

MICHEL DEMAZURE

# Cours d'algèbre

CASSINI



$$\hat{a} \; \mathbf{D}^3$$

**MICHEL DEMAZURE**, mathématicien, a été professeur aux universités de Strasbourg (1964-66) et de Paris-Sud (1966-1976), et à l'École polytechnique de Palaiseau (1976-1999).

Il a dirigé le Palais de la découverte (1991-1998) et présidé la Cité des sciences et de l'industrie (1998-2002).

Il préside le comité consultatif régional de la recherche de la région Languedoc-Roussillon (comité Arago). Depuis 2004, il réside en Touraine.

# Introduction

## Vingt-cinq siècles d'arithmétique

L'arithmétique et l'algèbre ont toujours mêlé deux traditions, théorique et pratique, que l'on peut symboliser par les noms d'Eudoxe et de Diophante, bien que leurs origines soient sans doute antérieures. Et de toujours a été présente de façon confuse la distinction entre calculs théoriquement possibles et calculs effectivement réalisables. Mais ce n'est que récemment, dans le double développement des théories de la complexité et des outils informatiques, que les idées se sont éclaircies, que des énoncés précis ont pu être formulés et que des applications souvent spectaculaires ont été développées.

C'est ainsi par exemple que la sécurité des transactions interbancaires par voie électronique repose essentiellement aujourd'hui sur la difficulté de la décomposition des grands nombres en facteurs premiers, alors même que cette difficulté n'est pas à ce jour prouvée. Étrange revanche pour une discipline, la théorie des nombres, jusque là modèle même « d'inutilité ». Qu'on ne s'y méprenne pas : loin de moi l'idée qu'on pourrait « justifier » l'arithmétique par la banque (non plus que l'inverse d'ailleurs !). Mais simplement d'illustrer l'étonnante plasticité de l'algèbre et, suivant l'expression bien connue de WIGNER, son « unreasonable effectiveness »<sup>1</sup>.

Cet ouvrage est ainsi consacré à quelques-unes des questions d'algèbre ou d'arithmétique qui sont revenues au premier plan du fait d'applications souvent spectaculaires. Il trouve son origine dans un cours que j'ai donné plusieurs années à l'École polytechnique dans le cadre d'un module d'enseignement intégré (« majeure ») comprenant algèbre, algorithmique, informatique et logique. Je veux d'ailleurs dire que cet enseignement est celui qui m'a laissé le meilleur souvenir en une vingtaine d'années, non seulement à cause de l'intérêt du thème et de son accueil par les élèves, mais aussi et d'abord grâce à la collaboration amicale avec les trois autres mousquetaires Jacques STERN, Jean-Marc STEYAERT et Jean VUILLEMIN.

---

1. Eugene Paul WIGNER, « The unreasonable effectiveness of mathematics in the natural sciences », *Communications in Pure and Applied Mathematics*, 13 (1960), 1-14.

## Guide de lecture

Les treize chapitres de cet ouvrage sont organisés en deux parties.

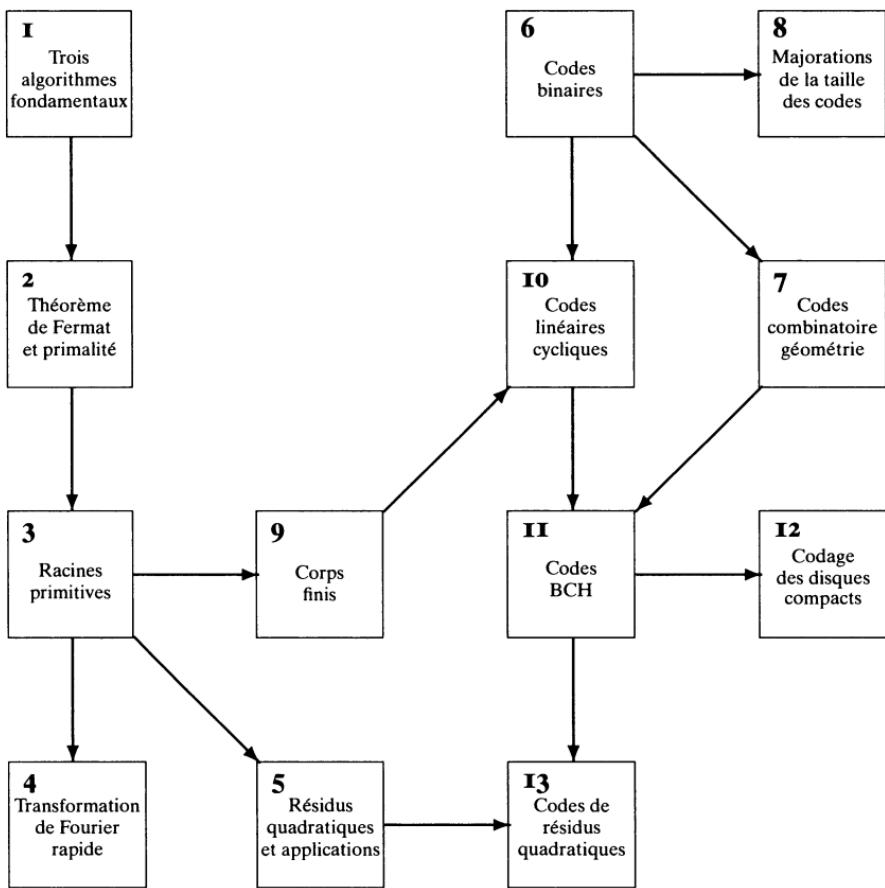
La première partie (chapitres 1 à 5) porte sur la primalité et ses applications, abordées de façon volontairement « concrète » et « calculatoire ». Elle contient de nombreux algorithmes, qui sont présentés chaque fois que cela est possible — et donc presque toujours étant donné leur simplicité — de façon indépendante du choix d'un langage (ou pseudo-langage) de programmation. On utilise la réécriture, que chacun pourra traduire dans son langage préféré. Des exemples variés de telles traductions sont donnés dans le premier chapitre. Dans la suite, j'ai privilégié un langage, à savoir Ruby ; j'expliquerai pourquoi un peu plus loin.

La seconde partie (chapitres 6 à 13) traite des codes correcteurs. Elle présente la problématique générale de la correction d'erreur et les principaux types de base de codes de blocs. Elle se conclut par deux applications de genre opposé. La première (chapitre 12) est « concrète », voire industrielle : elle présente le codage dit CIRC utilisé dans les disques compacts audio. La seconde (chapitre 13) est « mathématique » : on y voit comment la considération du *code de Golay* est le plus court chemin vers les plus jolis objets combinatoires (systèmes de Steiner, groupes de Mathieu).

Le diagramme ci-contre tente d'illustrer la dépendance mutuelle des divers chapitres. L'articulation entre les deux parties se fait dans le chapitre 9 (Corps finis), qui fait référence à la première partie (mais aussi en quelque sorte la conclut). Naturellement, on peut les lire dans l'ordre. Mais, comme on le voit, on peut attaquer directement la deuxième partie, dont les trois premiers chapitres (6, 7 et 8) sont indépendants de la première. On pourra continuer ensuite par le chapitre 9 et la suite, quitte à aller consulter les chapitres 2, 3 et 5 en tant que de besoin. Pour faciliter cette lecture, deux notions sont abordées à deux reprises : la construction d'anneaux par adjonction (3.2.6 et 9.1.2) ; la relation entre congruence modulo  $X^n - 1$  et permutations circulaires (4.1.2 et 10.3.2).

### Ruby

Venons-en au langage de programmation utilisé pour traduire les algorithmes présentés. L'objectif de ces traductions n'est pas de fournir au lecteur des programmes opérationnels, qui nécessiteraient des subtilités d'implémentation qui nous entraîneraient hors de notre sujet. Il s'agit simplement de permettre de « voir fonctionner » les algorithmes sur des exemples « réalistes », mais bien en deçà de ce que permettent les vrais systèmes opérationnels. Ainsi par exemple, dans l'exemple introductif de la première partie, on testera la primalité de  $(10^n - 1)/9$  pour  $n = 23$  et pas pour  $n = 1031$ . Mais pour ce faire, il faut déjà une arithmétique entière traitant des entiers à quelques



dizaines de chiffres décimaux, au delà des 32 voire 64 bits des entiers natifs des processeurs.

En quelque sorte, nous avons besoin d'une calculette programmable de précision arbitraire, disposant d'une syntaxe transparente. Cela m'a conduit à choisir un langage à typage dynamique disposant nativement de grands entiers, en l'espèce Ruby. Les traductions éventuelles en un autre langage du même type, par exemple Python seraient immédiates<sup>2</sup>. Les traductions dans un langage à typage statique (CAML-Light, Java) ne poseraient pas beaucoup de difficultés, sinon d'avoir à rendre explicites les types d'entiers utilisés, leurs opérateurs et leurs conversions.

2. Nous n'utiliserons aucune des constructions avancées de Ruby. Cela permet à la fois de ne pas avoir à spécifier la version choisie et de faciliter les traductions.

## Exercices

On trouvera tout le long du texte des exercices (plus de 260). Des indications sur leur solution sont données en fin d'ouvrage. Chaque exercice est affecté d'une lettre, suivant le code expliqué ci-dessous. Les lettres A à C indiquent des niveaux de difficulté. La lettre N est d'une autre nature.

- A** Un exercice noté [A] *doit* être résolu aussitôt rencontré. Cette résolution ne nécessite aucune écriture et doit être pratiquement instantanée. Dans le cas contraire, cela signifie soit que l'on n'a pas compris quelque chose d'essentiel... soit que l'auteur s'est trompé et que l'énoncé est faux !
- B** Un exercice noté [B] demande, soit de réfléchir un peu plus, soit d'écrire quelques lignes. Il ne contient aucune difficulté spéciale et ne demande pratiquement pas d'initiative. On devra en résoudre un nombre important. Le fait d'être bloqué dans un tel exercice est anormal.
- C** Un exercice noté [C] n'est pas forcément difficile, mais sa résolution nécessite quelque initiative.
- N** Les exercices notés [N] (pour « numériques ») peuvent nécessiter l'utilisation de machines. Leur niveau de difficulté est variable.

## Notations, index et glossaire

Ce livre n'utilise que des notations standard, à part un tout petit nombre. Celles-ci sont signalées en note à leur première apparition et figurent en tout cas dans *l'index des notations* qu'on trouvera en fin d'ouvrage, en compagnie de *l'index* et du *glossaire*. Pour permettre la consultation de la littérature en anglais, on donne en note de bas de page les traductions des termes principaux lorsqu'elles ne sont pas évidentes.

## Pour aller plus loin

La bibliographie qu'on trouvera en fin de volume contient, outre les livres auxquels renvoie le texte, quelques ouvrages de base sur les différents sujets abordés. À la fin de chacune des parties, on indique brièvement les références essentielles.

Les ressources du Web sont évidemment considérables. On en a indiqué quelques-unes dans le texte en espérant que les URL mentionnées ne changeraient pas trop vite.

Un *compagnon Web* de ce livre se trouve sur le site de Cassini : <http://cassini.fr>. Il y est accessible via la notice du livre et via la liste des sites compagnons. Il comprend notamment :

- ▷ les sources téléchargeables des programmes (ceux qui figurent dans le livre et d'autres),
- ▷ des mises à jour, notamment des url mentionnées dans le livre, des éditions des ouvrages cités, des records de factorisation...,
- ▷ des références complémentaires,

- ▷ les errata connus,
- ▷ des liens permettant de communiquer avec l'auteur et l'éditeur, notamment pour signaler des errata.

## La deuxième édition

Ce livre a connu une première édition, aujourd’hui épuisée. Cette seconde édition est marquée par des modifications considérables, dont voici les principales.

La première partie (Primalité) a été mise à jour. En particulier, on a ajouté le théorème de Agrawal, Kayal et Saxena (2004) permettant de décider si un nombre est premier par un algorithme déterministe à temps polynomial (5.4.4). Le langage Ruby a remplacé CAML-Light, pour les raisons expliquées plus haut.

La seconde partie (Codes correcteurs) a été *totalement refondue* et considérablement complétée. En particulier, les chapitres 7 (Codes, combinatoire, géométrie), 8 (Majorations de la taille des codes) et 12 (Le codage des disques compacts) sont intégralement nouveaux.

Pour compenser cet allongement et conserver un volume total raisonnable, on a supprimé l’ancien chapitre 7 (Divisibilité) qui, à vrai dire, détonnait un peu dans l’ensemble par son caractère plus abstrait. De l’ancien chapitre 6 (Anneaux), on n’a conservé que les résultats utiles au reste du livre, qui ont été répartis dans le texte aux endroits convenables, les définitions générales figurant, elles, dans le glossaire. (Le lecteur intéressé trouvera une rédaction améliorée de ces deux chapitres sur le *compagnon Web* du livre.)

Du fait de ces modifications, le livre a gagné en richesse et en cohérence. Son unité de contenu s'est faite autour des notions, évidemment liées, de nombre premier et de corps fini. Son unité de point de vue s'est affirmée, au cœur des interactions entre l'arithmétique, l'algèbre et l'informatique.

## Remerciements

Le manuscrit a été relu avec beaucoup de soin par Pierre-Vincent KOSELEFF qui m'a apporté de nombreuses suggestions. Mon éditeur André BELLAÏCHE a donné à ce livre une qualité de forme exceptionnelle, qui doit aussi beaucoup au travail réalisé par Alberto ARABIA pour la première édition. Je les remercie tous trois très vivement.

*Saint-Cyr sur Loire  
novembre 2008*



## **Première partie**

### **Primalité**



# Introduction à la première partie

## Un problème de primalité

Pour donner une idée des questions que nous allons traiter, prenons comme exemple la question suivante.

Rappelons qu'un entier  $n > 1$  est dit *composé* s'il peut s'écrire comme produit de deux entiers  $> 1$  et qu'il est dit *premier* s'il n'est pas composé.

Considérons les nombres entiers qui, écrits dans le système décimal, ne contiennent que le chiffre 1, c'est-à-dire 1, 11, 111, ... et demandons-nous lesquels sont premiers. D'abord, 1 n'est pas premier : c'est une question de définition. Ensuite, 11 est premier. Après, on commence à avoir des ennuis : 111 est divisible par 3, 1111 par 11, 11111 par 41, 111111 par 3, 1111111 par 239... Avant de continuer à diviser mécaniquement, il faudrait sans doute penser un peu.

Mettons d'abord un peu d'ordre. Choisissons une notation<sup>3</sup>, par exemple  $\mathbf{1}^{[n]}$ , pour une suite de  $n$  chiffres égaux à 1 :

$$\mathbf{1}^{[n]} = \frac{10^n - 1}{9}.$$

Il est clair que, dès que  $n$  est pair,  $\mathbf{1}^{[n]}$  est divisible par 11, le quotient s'exprimant par une suite alternée de 1 et de 0. Comme 11 n'est autre que  $\mathbf{1}^{[2]}$ , cela suggère que, dès que  $m$  divise  $n$ ,  $\mathbf{1}^{[m]}$  divise  $\mathbf{1}^{[n]}$ . Et cela est bien clair, le quotient ayant un chiffre tous les  $m$  égal à 1, les autres égaux à 0. D'ailleurs tout ceci n'a rien à voir avec la base 10 et reste valable pour une base quelconque  $b$ , puisqu'il s'agit en fait de l'identité polynomiale

$$\frac{b^{mr} - 1}{b - 1} = \frac{b^r - 1}{b - 1} (b^{(m-1)r} + \cdots + 1).$$

Par conséquent, pour que  $\mathbf{1}^{[n]}$  soit premier, il est nécessaire que  $n$  soit premier.

Notons au passage l'identité

$$\frac{b^n - 1}{b - 1} = b^{n-n'} \frac{b^{n'} - 1}{b - 1} + \frac{b^{n-n'} - 1}{b - 1}.$$

Ainsi, les diviseurs communs à  $\mathbf{1}^{[n]}$  et  $\mathbf{1}^{[n']}$  sont aussi les diviseurs communs à  $\mathbf{1}^{[n']}$  et  $\mathbf{1}^{[n-n']}$ . Une récurrence facile montre alors que si  $n$  et  $n'$  sont premiers entre eux,  $\mathbf{1}^{[n]}$  et  $\mathbf{1}^{[n']}$  sont premiers entre eux. Par conséquent, aucun diviseur

3. Ces nombres sont baptisés *repunits* en anglais et souvent notés  $R_n$ .

*premier d'un  $1^{[n]}$ , avec  $n$  premier, n'est aussi un diviseur d'un autre.* Les facteurs premiers des  $1^{[n]}$  donnent donc une infinité de nombres premiers distincts.

Nous pouvons donc désormais nous restreindre au cas où  $n$  est premier. Revenons à notre liste et d'abord au cas  $n = 5$  : pourquoi ce facteur 41 ? On peut en tout cas prouver<sup>4</sup> que si un nombre premier  $p$  divise  $1^{[n]}$ ,  $n$  étant premier, alors  $2n$  divise  $p - 1$ , donc  $p$  est l'un des entiers  $2n + 1, 4n + 1, \dots$ . Ainsi, les diviseurs premiers de  $1^{[5]}$  sont à chercher parmi les entiers de la forme  $10k + 1$  ; il n'est pas nécessaire d'essayer  $11 = 1^{[2]}$ , ni 21 qui n'est pas premier ; après un seul échec avec 31, on trouve la décomposition  $11111 = 41 \times 271$ . De même, les diviseurs premiers de  $1^{[7]}$  sont de la forme  $14k + 1$  ; on essaye avec 29, 43, 57, ... et on trouve  $1^{[7]} = 239 \times 4649$ .

### La méthode « bête »

De manière générale, si  $1^{[n]}$  n'est pas premier, il possède un diviseur inférieur à sa racine carrée, qui est de l'ordre de  $\frac{1}{3}10^{n/2}$  ; comme il suffit d'essayer un diviseur tous les  $2n$ , il faut faire au plus  $\frac{1}{6n}10^{n/2}$  essais. Ayant délaissé le calcul mental pour la calculette, puis la calculette pour le micro-ordinateur, on trouve ainsi  $1^{[11]} = 21649 \times 513239$ . On a ensuite une bonne surprise :  $1^{[13]}$  est divisible par 53 ; puis une mauvaise : au bout de plus de soixante mille essais, on trouve que  $1^{[17]}$  est divisible par 2071723.

Enfin, pour  $n = 19$ , après plus de  $2 \times 10^7$  essais infructueux, on conclut que  $1^{[19]}$  est premier. On atteint là les limites de la méthode « bête », même améliorée par le facteur  $2n$  ci-dessus : comme  $1^{[23]}$  est également premier, il faudrait plus de  $2 \times 10^9$  essais infructueux (où à chaque fois on divise un nombre à 23 chiffres par un nombre qui peut en avoir jusqu'à 12) pour le constater !

On voit bien où le bâton blesse : lorsqu'on passe du nombre premier  $n$  au suivant, qui est parfois  $n + 2$  ou  $n + 4$ , mais le plus souvent encore plus grand, on ajoute à  $1^{[n]}$  deux chiffres ou quatre, ou plus encore. Donc en gros on le multiplie par 100, par 10000, ou plus, et on multiplie le nombre d'essais par 10, par 100, ou encore beaucoup plus.

En définitive, ce qui condamne la méthode bête (même améliorée), ce n'est pas tant la *taille* du nombre des essais à faire pour un  $n$  donné, mais le *type de croissance* de ce nombre en fonction de  $n$ . Améliorer les performances de la machine utilisée par un facteur 100 ou 1000 permettra peut-être d'ajouter une valeur à la table, et encore. Quant on songe que  $1^{[1031]}$  est pre-

---

4. Voir la proposition 2.16.

mier<sup>5</sup>, on voit bien qu'il faut des méthodes autrement plus compliquées pour le démontrer !

Nous venons en fait de buter sur une règle pratique fondamentale : lorsque la complexité d'un algorithme est *exponentielle* comme dans le cas précédent, tout se passe comme s'il existait une limite absolue à son utilisation.

Outre cette énorme complexité, la méthode bête a un autre défaut. La formule  $1^{[11]} = 21649 \times 513239$ , qui est courte et peut se vérifier rapidement, est une démonstration du caractère composé du nombre  $1^{[11]}$  (c'est un « certificat court de non-primalité »). À l'inverse, le fait qu'on n'ait pas trouvé de diviseur de  $1^{[23]}$  inférieur à sa racine carrée ne peut guère être considéré comme une démonstration ; c'est plutôt un fait d'expérience, dont la seule vérification consiste en la répétition de l'expérience. C'est un énoncé purement « autoritaire », sans aucun pouvoir de conviction, qui se résume en définitive à :  $1^{[23]}$  est premier parce qu'il figure dans la liste des nombres premiers connus ! On voit ici apparaître une dissymétrie apparente entre le fait d'être composé et le fait d'être premier, tout au moins tant qu'on n'a pas prouvé que tout nombre premier possède des « certificats courts de primalité », ce que nous ferons en 3.2.2.

### Utilisation du petit théorème de Fermat

Il nous faut donc trouver d'autres méthodes d'attaque de la question. En voici une : le « petit théorème de Fermat » (proposition 2.10) affirme que si  $a$  et  $p$  sont deux entiers, et si  $p$  est premier, alors  $a^p - a$  est divisible par  $p$ . Dans la notation des congruences (voir 1.3.2), cela s'écrit  $a^p \equiv a \pmod{p}$ . Si  $1^{[n]}$  est premier et si on calcule modulo  $1^{[n]}$ , on a donc pour tout entier  $a$

$$a^{10^n} = a^{9 \cdot 1^{[n]} + 1} = (a^{1^{[n]}})^9 a \equiv a^9 a = a^{10}.$$

Il suffit par conséquent de trouver *un* entier  $a$  (par exemple  $a = 2$  qui conviendra le plus souvent) qui mette en défaut la congruence

$$a^{10^n} \equiv a^{10} \pmod{1^{[n]}}$$

pour en conclure que  $1^{[n]}$  n'est pas premier. Et il s'agit là d'un test assez rapide puisqu'il faut itérer  $n$  fois l'application  $a \pmod{1^{[n]}} \mapsto a^{10} \pmod{1^{[n]}}$ . Un tel  $a$  sera un « témoin de non-primalité » et on aura obtenu ainsi un « certificat court de non-primalité », mais sans pour autant avoir la moindre indication sur d'éventuels diviseurs.

---

5. En fait, 2, 19, 23, 317 et 1031 sont les cinq seuls entiers  $n$  pour lesquels on est sûr aujourd'hui (novembre 2008) que  $1^{[n]}$  est premier. On sait qu'il n'y en a pas d'autre avant  $1^{[49081]}$ , lequel n'a pas encore été prouvé premier, mais est « probablement premier » au sens que nous verrons plus loin.

Calculons par exemple en Ruby, à l'aide de la fonction `test(a,n)` ci-dessous qui affiche  $a^{10}$  et  $a^{10^n}$  modulo  $1^{[n]}$ , et le résultat de leur comparaison. La programmation est volontairement simpliste (on élève  $a$  à la puissance 10, *avant* de réduire ! Cela va bien ici, pour des petites valeurs de  $n$ ).

```
def test(a,n)
  repunit = (10**n-1)/9
  valk = val1 = (a**10) % repunit
  (n-1).times{ valk = (valk**10) % repunit }
  [val1, valk, valk == val1]
end
```

Le test avec  $a = 2$  nous permet de voir que  $1^{[n]}$  n'est pas premier lorsque  $n$  vaut 5, 7, 11, 13, 17 ou 31.

```
test(2,5) # => [1024, 6889, false]
test(2,7) # => [1024, 567467, false]
test(2,11) # => [1024, 1610031386, false]
test(2,13) # => [1024, 363509731629, false]
test(2,17) # => [1024, 7396339341874966, false]
test(2,31) # => [1024, 29989739325183025934329916551, false]
```

## Nombres probablement premiers

En revanche, pour  $n = 19$ , on obtient :

```
test(2,19) # => [1024, 1024, true]
```

Là, de deux choses l'une : ou bien  $1^{[19]}$  est premier, ou bien nous n'avons pas eu de chance. Continuons avec d'autres valeurs de  $a$  :

```
test(3,19) # => [59049, 59049, true]
test(5,19) # => [9765625, 9765625, true]
test(17,19) # => [2015993900449, 2015993900449, true]
test(12345,19) # => [930411748858376498, 930411748858376498, true]
test(987654321,19)
# => [1060837307766049730, 1060837307766049730, true]
```

Visiblement, nous n'arriverons pas à mettre en défaut notre critère. Nous pouvons soupçonner  $1^{[19]}$  d'être premier. Mais il se trouve que ce critère n'est pas infaillible : il existe des nombres, dits de Carmichael (définition 3.34), qui ne sont pas premiers mais qui le mettent en défaut (sauf un tout petit nombre de témoins).

On peut cependant améliorer la méthode de la façon suivante. On a  $1^{[n]} - 1 = 10 \times 1^{[n-1]}$ , donc  $1^{[n]} - 1 = 2t$  avec  $t = 5 \times 1^{[n-1]}$ . La congruence de Fermat s'écrit  $a(a^t + 1)(a^t - 1) \equiv 0 \pmod{1^{[n]}}$ . Si  $1^{[n]}$  est premier, et si  $a$  n'est pas divisible par  $1^{[n]}$ , on en conclut qu'on a l'une des deux congruences  $a^t \equiv 1$  ou  $a^t \equiv -1$ . Nous pouvons ainsi améliorer notre critère : si l'on trouve un  $a$  avec  $0 < a < 1^{[n]}$  qui mette en défaut l'alternative, alors  $1^{[n]}$  n'est pas premier.

Dans notre cas, cela ne marche pas mieux :

```
def test2(a,n)
    repunit = (10**n-1)/9
    valk = val1 = (a**5) % repunit
    (n-1).times{ valk = (valk**10) % repunit }
    [valk == val1, valk == repunit - val1]
end

test2(2,19) # => [true, false]
test2(3,19) # => [false, true]
test2(5,19) # => [true, false]
test2(17,19) # => [true, false]
test2(12345,19) # => [true, false]
test2(987654321,19) # => [false, true]
```

Mais nous verrons plus loin (théorème 2.24, noter que  $t$  est impair) que l'amélioration est en fait capitale, car si  $1^{[n]}$  n'est pas premier, les trois quarts au moins des  $1^{[n]} - 1$  valeurs possibles de  $a$  en témoigneront.

On voit donc que si l'on choisit des  $a$  « au hasard », et si l'on affirme au bout de  $N$  résultats positifs que  $1^{[n]}$  est premier, on court un risque d'erreur d'au plus  $4^{-N}$ . Après 50 essais, ce qui est très peu, le risque est inférieur à  $10^{-30}$ . Qui ne courrait un tel risque, visiblement inférieur à celui par exemple de mourir subitement en formulant le pari ? Cela nous conduit à une étrange classe de nombres : les nombres « probablement premiers », ou « premiers à usage commercial ».

### Moralité provisoire

Cette brève histoire nous montre que l'analyse de la primalité éventuelle d'un entier  $n$  débouche sur au moins quatre variantes :

- ▷ prouver que  $n$  n'est pas premier ;
- ▷ si  $n$  n'est pas premier, le décomposer ;
- ▷ garantir, avec un risque d'erreur faible, que  $n$  est premier ;
- ▷ certifier que  $n$  est premier.

Ces quatre questions sont de complexité très différentes. Pour un nombre de taille raisonnable, on arrivera assez vite à l'une des deux conclusions «  $n$  n'est pas premier et en voici une preuve » ou «  $n$  est probablement premier ». Décomposer  $n$  dans le premier cas, certifier sa primalité dans le second cas, sont des questions beaucoup plus difficiles. Elles sont d'ailleurs liées, car d'une part une véritable décomposition en facteurs premiers nécessite la certification des facteurs, et d'autre part on utilise souvent, pour certifier qu'un nombre  $n$  est premier, des décompositions d'entiers auxiliaires (tels que  $n - 1$  ou  $n + 1$ ).

Notons par ailleurs que dans la subdivision précédente se glisse un mécanisme de *secret* : si je choisis deux grands nombres premiers (ou probablement premiers)  $p$  et  $q$ , et si je rends public le produit  $n = pq$  accompagné même

si je le veux d'un certificat de non-primalité, je deviens ipso facto l'unique propriétaire d'un secret, celui de la décomposition de  $n$ . La méthode de cryptographie RSA (voir 2.3.5) est basée sur cette remarque.

En fin de compte, le sujet est assez vaste, et nous ne nous intéresserons vraiment parmi les quatre questions de la liste ci-dessus qu'à la première et à la troisième, auxquelles on peut donner des réponses efficaces avec des moyens techniques élémentaires. Les deux autres nécessitent pour être vraiment efficaces des outils algébriques plus avancés<sup>6</sup>.

Il est assez piquant de voir la décomposition en facteurs premiers, premier pas capital de l'histoire de l'arithmétique (Livres VII et IX des Éléments d'Euclide) redevenir après vingt-quatre siècles un sujet en plein développement. En hommage à Euclide (et à Mersenne<sup>7</sup>), contemplons :

Le nombre  $2^{43\,112\,609} - 1$  est premier.

---

6. On pourra consulter par exemple [Koblitz].

7. Voir 3.2.4.

# Chapitre I

## Trois algorithmes fondamentaux

Les résultats de ce chapitre conditionnent l'ensemble des algorithmes de primalité des chapitres suivants. Il expose en effet comment on peut faire rapidement (en un sens que nous préciserons) deux calculs arithmétiques fondamentaux : le pgcd de deux entiers et les puissances d'un entier. Notons au passage que, dans le chapitre 4, nous montrerons comment on peut multiplier rapidement de très grands entiers.

À ces deux algorithmes et leurs variantes, nous ajouterons celui de Floyd (1.6.3), qui a de multiples applications, notamment l'algorithme  $\rho$  de Pollard (1.6.5) qui permet de décomposer en facteurs les nombres entiers pas trop grands.

### § 1.1. Réécriture

#### 1.1.1. Règles de réécriture

La *réécriture* est une manière commode d'exprimer les algorithmes algébriques simples. Prenons l'exemple de la division euclidienne des entiers. Fixons le diviseur  $b > 0$ . Il s'agit, un entier  $a$  étant donné, d'obtenir des entiers  $q = a \div b$  (le quotient entier) et  $r = a \bmod b$  (le reste) tels qu'on ait  $a = bq + r$  et  $0 \leq r < b$ . Pour trouver le reste  $r$ , il suffit d'appliquer tant que c'est possible les *règles de réécriture* suivantes : si  $a$  est  $< 0$ , le remplacer par  $a + b$  ; si  $a$  est  $\geq b$ , le remplacer par  $a - b$  ; ce que nous écrirons :

$$\begin{aligned} [a < 0] : \quad a &\mapsto a + b \\ [a \geq b] : \quad a &\mapsto a - b \end{aligned}$$

Dans une telle écriture, chaque ligne est une règle qui décrit une application. Lorsque l'application n'est pas partout définie, son ensemble de définition est décrit par une condition entre crochets. Appliquer ces règles à un élément  $a$  signifie choisir, s'il en existe, une règle applicable à  $a$ , remplacer  $a$  par le résultat de la règle, disons  $a_1$ , recommencer avec  $a_1$  (avec une règle de la liste, celle dont on vient de se servir ou une autre), et ainsi de suite tant qu'on peut continuer.

De la même façon, pour trouver le couple  $(q, r)$  avec  $0 \leq r < b$  et  $a = bq + r$ , il suffit d'appliquer les règles de réécriture suivantes :

$$\begin{array}{rcl} a & \mapsto & (0, a) \\ [r < 0] : & (q, r) & \mapsto (q - 1, r + b) \\ [r \geq b] : & (q, r) & \mapsto (q + 1, r - b) \end{array}$$

Prenons par exemple  $b = 7$ . On a la suite de réécriture  $18 \mapsto (0, 18) \mapsto (1, 11) \mapsto (2, 4)$  et aucune règle ne peut s'appliquer à  $(2, 4)$ ; on a bien  $18 = 2 \times 7 + 4$  avec  $0 \leq 4 < 7$ . De même, on a  $-12 \mapsto (0, -12) \mapsto (-1, -5) \mapsto (-2, 2)$ .

Pour être non ambigu, il vaut mieux introduire dans l'algorithme la variable  $b$  (qui reste inchangée), et préciser les conditions de validité. Il est commode en outre d'introduire le nom (ici `div`) de la fonction calculée. On obtient ainsi :

---

#### ALGORITHME 1.1. — Division euclidienne

---

*Entrées* : entiers  $a$  et  $b$  avec  $b > 0$ .

*Sorties* :  $b$  inchangé, entiers  $q$  et  $r$  avec  $a = bq + r$  et  $0 \leq r < b$ .

*Règles* :

$$\begin{array}{rcl} \text{div}(a, b) & \mapsto & (0, a, b) \\ [r < 0] : & (q, r, b) & \mapsto (q - 1, r + b, b) \\ [r \geq b] : & (q, r, b) & \mapsto (q + 1, r - b, b) \end{array}$$


---

On aura ainsi par exemple la réécriture

$$\text{div}(18, 7) \mapsto (0, 18, 7) \mapsto (1, 11, 7) \mapsto (2, 4, 7)$$

La validité de cet algorithme est pratiquement évidente. Vérifions-la quand même, pour servir d'exemple à une méthode générale. Celle-ci consiste à décomposer le problème en deux parties : *terminaison* et *correction*.

Parlons d'abord de la *terminaison*<sup>1</sup>. Cela signifie que, si les données initiales satisfont aux conditions indiquées, la réécriture ne peut se poursuivre indéfiniment. On le vérifie souvent en prouvant qu'une quantité auxiliaire, à valeurs entières positives, diminue strictement à chaque application d'une règle (sauf peut-être les règles ad hoc d'initialisation, comme la première règle ci-dessus). C'est le cas ici par exemple de la distance de  $r$  au centre de l'intervalle  $\llbracket 0, b \rrbracket$ , disons de  $|2r - b|$ .

Pour vérifier la *correction*, on exhibe en général des *invariants* conservés par application des règles. Ici, outre  $b$  évidemment invariant, on considère l'expression  $bq + r$ , transformée suivant le cas en  $b(q - 1) + r + b$  ou en  $b(q + 1) + r - b$ , donc inchangée, et qui vaut au départ  $b \times 0 + a = a$ . Ainsi, si l'on considère une réécriture itérée de  $a$  qu'on ne peut prolonger,

---

1. On dit en français « terminer quelque chose » ou « se terminer ». L'expression « l'algorithme termine » des informaticiens est au moins un anglicisme, sinon un barbarisme à éviter.

son dernier élément  $(q, r)$  doit satisfaire à  $a = bq + r$  par invariance, et on a bien  $0 \leq r$  et  $r < b$  puisqu'aucune règle n'est applicable à  $(q, r)$ .

**EXERCICE I.1. [A]** — Vérifier la correction et la terminaison du premier système de règles.

Donnons un autre algorithme de division euclidienne, dans lequel on suppose disposer d'une opération de multiplication par 2. On se restreint pour simplifier à deux entiers  $a$  et  $b$  strictement positifs, et on détermine au préalable un entier  $n \geq 0$  tel que  $2^n b > a$ .

**ALGORITHME I.2. — Division euclidienne : variante binaire**

---

*Entrées* : entiers  $a, b$  et  $n$  avec  $2^n b > a > 0$ .

*Sorties* : entiers  $q$  et  $r$  avec  $a = bq + r$  et  $0 \leq r < b$ .

*Règles* :

$$\begin{array}{lll} \texttt{div}(a, b, n) & \mapsto & (n, 0, a, b) \\ [k > 0 \text{ et } r < 2^{k-1}b] : & (k, q, r, b) & \mapsto (k-1, 2q, r, b) \\ [k > 0 \text{ et } r \geq 2^{k-1}b] : & (k, q, r, b) & \mapsto (k-1, 2q+1, r-2^{k-1}b, b) \\ [k = 0] : & (k, q, r, b) & \mapsto (q, r) \end{array}$$


---

**EXERCICE I.2. [B]** — Démontrer la validité de cet algorithme.

**REMARQUE I.3.** — On peut aussi représenter en réécriture des algorithmes de division infinis, comme

$$\begin{array}{ll} (1 - X)^{-1} & \mapsto 1 + u \\ u & \mapsto X + uX \end{array}$$

qui donne la réécriture infinie

$$(1 - X)^{-1} \mapsto (1 + u) \mapsto (1 + X + uX) \mapsto (1 + X + X^2 + uX^2) \mapsto \dots$$

qui ne se termine pas, mais « converge » en un sens évident.

### I.II.2. Réécriture et déterminisme

La description des algorithmes par des règles de réécriture a plusieurs avantages.

Le premier avantage, c'est que l'on disjoint nettement les règles composant l'algorithme du mécanisme de contrôle de l'exécution. Si l'algorithme est déterministe (ce qui signifie qu'il n'arrive jamais que deux règles entrent en concurrence), cela n'apporte qu'une légère simplification.

Le premier algorithme de calcul de  $a \bmod b$  par exemple pourrait se traduire par l'expression conditionnelle

$$\begin{aligned} a \bmod b = & \text{if } a < 0 \text{ then } (a + b) \bmod b \\ & \text{elsif } a \geq b \text{ then } (a - b) \bmod b \\ & \text{else } a, \end{aligned}$$

tout aussi bien que par

```
a mod b = if a ≥ b then (a - b) mod b
           elseif a < 0 then (a + b) mod b
           else a.
```

En revanche, lorsque l'algorithme n'est pas déterministe, il faudra bien évidemment pour l'implanter le « déterminer » d'une façon ou d'une autre. Mais le fait que le système de réécriture soit valide impliquera que tout algorithme déterministe qu'on en déduit est valide. On notera d'ailleurs que l'indéterminisme des systèmes de réécriture est un *indéterminisme par indifférence* et non comme dans d'autres situations (Prolog) un *indéterminisme par ignorance*.

Voici un exemple *ad hoc* de réécriture non déterministe : il s'agit de résoudre par la méthode de substitution un système linéaire triangulaire, en l'espèce de calculer en fonction des paramètres  $a_i$  l'élément  $y = x_1 + \dots + x_n$ , où les  $x_i$  sont obtenus en résolvant le système linéaire :

$$\begin{aligned}x_1 &= a_1, \\x_2 - x_1 &= a_2, \\&\dots = \dots, \\x_n - x_{n-1} - \dots - x_1 &= a_n.\end{aligned}$$

On a en fait  $y = 2^{n-1}a_1 + \dots + 2a_{n-1} + a_n$ . On introduit naturellement le système de règles de réécriture

$$\begin{aligned}x_1 &\mapsto a_1, \\x_2 &\mapsto x_1 + a_2, \\&\dots \\x_n &\mapsto x_{n-1} + \dots + x_1 + a_n\end{aligned}$$

Pour  $n = 3$ , on pourra par exemple obtenir la réécriture

$$\begin{aligned}y &= x_1 + x_2 + x_3 \mapsto 2x_1 + 2x_2 + a_3 \\&\mapsto 4x_1 + 2a_2 + a_3 \mapsto 4a_1 + 2a_2 + a_3,\end{aligned}$$

en prenant les règles dans l'ordre 3, 2, 1, mais aussi

$$\begin{aligned}y &= x_1 + x_2 + x_3 \mapsto a_1 + x_2 + x_3 \\&\mapsto a_1 + a_2 + x_1 + x_3 \mapsto 2a_1 + a_2 + x_3 \mapsto \dots\end{aligned}$$

en prenant les règles dans l'ordre 1, 2, 1, 3, 1, 2, 1, ou encore toute autre possibilité.

On peut vérifier la terminaison du système précédent, ainsi que les remarques suivantes, qui expliquent sans doute la difficulté des débutants à résoudre les systèmes linéaires par la méthode dite « de substitution » :

- ▷ Toute stratégie de choix donne un nombre de pas compris entre  $n$  et  $2^n - 1$ .
- ▷ L'unique stratégie donnant  $n$  pas consiste à prendre à chaque fois la dernière règle applicable.
- ▷ L'unique stratégie donnant  $2^n - 1$  pas consiste à prendre à chaque fois la première règle applicable.

**EXERCICE I.3. [C]** — Vérifier ce qui précède.

**REMARQUE I.4.** — La différence exponentielle de complexité rencontrée ici entre la meilleure stratégie et la pire est un phénomène fréquent en réécriture non déterministe.

### I.I.3. Traduction dans un langage de programmation

Un algorithme décrit par des règles de réécriture se traduit très facilement dans tout langage de programmation raisonnable. Prenons d'abord un exemple tout à fait élémentaire, celui de la fonction *factorielle* sur les entiers  $> 0$  définie par le système de règles (en un sens élargi que le lecteur comprendra sans peine) :

**ALGORITHME I.5. — Factorielle, version 1** —

*Entrées* : entier  $n \geq 1$ . *Sorties* :  $n!$ .

*Règles* :

$$\begin{aligned} [n > 1] : \text{fact}(n) &\mapsto n \cdot \text{fact}(n-1) \\ [n = 1] : \text{fact}(n) &\mapsto 1 \end{aligned}$$

Commençons par les langages fonctionnels, évidemment particulièrement adaptés à la réécriture. Prenons d'abord Lisp. On peut traduire automatiquement les règles précédentes :

```
(defun fact(n)
  (cond
    ((> n 1) (* n (fact (1- n))))
    ((= n 1) 1)))
```

En utilisant *if* plutôt que *cond*, cela donne :

```
(defun fact(n)
  (if (> n 1) (* n (fact (1- n))) 1))
```

En CAML-Light, on écrira de manière analogue :

```
let rec fact (n) =
  if n=1 then 1 else n*fact(n-1);;
```

ou plutôt, dans un style plus proche de la réécriture :

```
let rec fact =
  fun 1 -> 1 | n -> n*fact(n-1);;
```

Dans un langage procédural, en C par exemple, on écrira :

```
int fact(int n)
{
  if (n>1) return n*fact(n-1);
  else return 1;
}
```

Bien évidemment, pour transformer un système de règles de réécriture en un programme, il faut d'une façon ou d'une autre en choisir une variante déterministe. Deux solutions « automatiques » existent. La plus fréquente consiste à appliquer la première règle possible dans l'ordre où elles sont écrites. C'est l'usage de CAML-Light par exemple. Cela amène à mettre en tête les cas particuliers. Une méthode plus astucieuse se base sur le « best match » : la règle  $1 \mapsto 1$  sera préférée à  $n \mapsto n * \text{fact}(n - 1)$  lorsque les deux s'appliquent, indépendamment de sa place dans la liste, car elle est « plus précise ».

## La question des grands entiers

Il convient de noter que ces programmes ne donneront les résultats voulus que pour de petites valeurs de  $n$ . En effet, le type `int` (explicite en C, implicite en CAML-Light) ne représente les entiers que modulo une puissance de 2 (en général  $2^{32}$ ). Pour utiliser des entiers de taille quelconque, on doit faire les déclarations de type nécessaires et modifier la syntaxe. Nous conserverons ci-après quelques traductions en CAML-Light ou en C, malgré cet inconvénient.

Pour permettre d'exécuter simplement ces algorithmes dans des situations réalistes, sans pour autant surcharger la syntaxe, nous utiliserons, comme une calculette, le langage Ruby qui ne nécessite ni typage explicite, ni syntaxe arithmétique spéciale et dispose d'un interpréteur.

En Ruby, on aura ainsi :

```
class Integer
  def fact
    (self == 1) ? 1 : self*(self-1).fact
  end
end
40.fact # => 815915283247897734345611269596115894272000000000
```

Évidemment, si on a affaire à de vraiment grands entiers dans une situation extrême où chaque fraction de cycle machine gagnée compte, on doit aller plus profond, comme le font les systèmes spécialisés tels que PARI<sup>2</sup>. Par exemple, savoir si un très grand nombre est pair ou impair, plus généralement trouver son reste modulo une puissance de 2, est immédiat lorsqu'on

---

2. Voir <http://pari.math.u-bordeaux.fr/>.

en connaît la représentation binaire interne. Nous simulerons le cas échéant ce genre de connaissances, comme dans :

```
class Integer
  def odd? ; self%2 == 1 ; end
  def even? ; self%2 == 0 ; end
  def half ; self/2 ; end
end
```

### 1.1.4. Récursion et itération

On sait bien que l'utilisation immodérée de la récursion peut donner des programmes extrêmement inefficaces, mais qu'un type particulier de récursion, dite *terminale*, est équivalent à l'itération. Or la réécriture permet presque toujours d'aboutir à des récursions *terminales*. On commence par bannir les symboles de fonctions (en tout cas dans la partie droite des règles). Dans le cas de la fonction factorielle, on prend par exemple :

---

**ALGORITHME 1.6. — Factorielle, version 2**

*Entrées* : entier  $n \geq 1$ . *Sorties* :  $n!$ .

Règles :

$$\begin{array}{l} \text{fact}(n) \mapsto (1, n) \\ [n > 1] : (f, n) \mapsto (nf, n - 1) \\ [n = 1] : (f, n) \mapsto f \end{array}$$


---

**EXERCICE 1.4. [A]** — Quel invariant assure la correction de cet algorithme ?

On peut alors introduire un nom de fonction pour le calcul intermédiaire, ce qui donne :

$$\begin{array}{l} \text{fact}(n) \mapsto \text{facrec}(1, n) \\ [n > 1] : \text{facrec}(f, n) \mapsto \text{facrec}(nf, n - 1) \\ [n = 1] : \text{facrec}(f, n) \mapsto f \end{array}$$

La traduction est maintenant automatique. En C, on obtient :

---

**PROGRAMME 1.7. — Factorielle en C, version 2**

```
int facrec(int f, int n)
{
    if (n>1) return facrec(n*f, n-1);
    else return f;
}
int fact(int n)
{
    return facrec(1, n);
}
```

---

En Lisp, on écrira :

```
(defun facrec (f n)
  (if (> n 1) (facrec (* n f) (1- n)) f))
(defun fact (n) (facrec 1 n))
```

ou encore, en rendant locale la fonction facrec :

PROGRAMME I.8. — *Factorielle en Lisp, version 2* ——————

```
(defun fact (n)
  (let facrec ((f 1) (n n))
    (if (> n 1) (facrec (* n f) (1- n)) f)))
```

On aura de même en CAML-Light :

```
let fact (n) =
  facrec (1,n) where rec
  facrec (f,n) =
    if n=1 then f else facrec (n*f,n-1);;
```

ou mieux, en « curryfiant » les fonctions :

PROGRAMME I.9. — *Factorielle en CAML-Light, version 2* ——————

```
let fact =
  facrec 1 where rec
  facrec f =
    fun 1 -> f | n -> facrec (n*f) (n-1);;
```

En Ruby, cela donnera directement :

PROGRAMME I.10. — *Factorielle en Ruby, version 2* ——————

```
def facrec(f,n)
  (n == 1) ? f : facrec(n*f, n-1)
end
class Integer
  def fact
    facrec(1, self)
  end
end
40.fact # => 815915283247897734345611269596115894272000000000
```

On peut aussi décalquer le style fonctionnel, avec le peu rubyistique

PROGRAMME I.11. — *Factorielle en Ruby, version 2bis* ——————

```
class Integer
  def fact
    (facrec = lambda{|f,n| n==1 ? f : facrec[n*f, n-1]}[1,self])
  end
end
40.fact # => 815915283247897734345611269596115894272000000000
```

La récursion terminale équivaut, comme on l'a dit, à l'itération. Les (bons) compilateurs la reconnaissent. Naturellement, dans un cas aussi simple, on peut programmer directement :

PROGRAMME I.12. — *Factorielle en C, version 2bis*


---

```
int fact(int n)
{
    int f = 1;
    while (n-- > 1) f = n*f;
    return f;
}
```

---

En Ruby, on pourra écrire de même :

PROGRAMME I.13. — *Factorielle en Ruby, version 2ter*


---

```
class Integer
  def fact
    (1..self).to_a.inject {|f,i| f = f*i}
  end
end
1000.fact / 10**2550 # => 402387260077093773
```

---

Nous arrêterons là ce jeu un peu pédant de transformation de programmes. Dans la suite, nous nous limiterons à Ruby et n'écrirons que les traductions les plus naïves.

**EXERCICE I.5. [C]** — Nous parlerons plus loin de la suite de Fibonacci définie récursivement par

$$F_0 = 0, \quad F_1 = 1, \quad F_{n+1} = F_n + F_{n-1}.$$

Donner des règles de réécriture calculant  $F_n$  sans utiliser de symbole pour  $F$ . Traduire en un programme dans le langage de son choix.

**EXERCICE I.6. [C]** — Généraliser à toute récurrence de la forme

$$f(n) = \Phi(f(n-1), \dots, f(n-k)),$$

où la fonction  $\Phi$  et l'entier  $k$  sont fixés.

## § I.2. Calcul rapide des puissances

Beaucoup des résultats que nous donnerons par la suite sont conditionnés par la possibilité de calculer rapidement une puissance  $a^m$  d'un élément  $a$ . La définition des puissances donne l'algorithme naïf :

$$\begin{aligned} \text{puiss}(a, m) &\mapsto (1, m) \\ [n > 0] : (b, n) &\mapsto (ba, n-1) \\ [n = 0] : (b, n) &\mapsto b \end{aligned}$$

d'invariant  $ba^n$  et dont le nombre de multiplications est  $m - 1$ . Par « calculer rapidement », nous entendons que le nombre de multiplications utilisées ne

doit pas croître linéairement avec  $m$ , comme dans l'algorithme naïf ci-dessus, mais seulement logarithmiquement.

L'idée de base est simple : pour calculer  $a^8$  par exemple, il suffit de calculer successivement  $a^2$ ,  $a^4$  et  $a^8$  par élévation au carré. Dans le cas où  $m$  n'est pas une puissance de 2, on utilise évidemment son écriture dans le système de numération binaire. On a alors deux variantes principales, suivant que l'on procède des poids faibles vers les poids forts ou en sens inverse. Nous les illustrerons par le cas simple  $n = 13 = 8 + 4 + 1$ . Si  $s$  est une suite de bits, on note  $(s)_2$  l'entier d'écriture binaire  $s$ . Ainsi on a  $13 = (1101)_2$ .

### 1.2.1. Des poids faibles vers les poids forts

Exposons d'abord la première méthode, nous verrons ensuite ses avantages et ses inconvénients.

Pour fixer les notations, nous considérons une loi de composition associative, notée multiplicativement, possédant un élément unité  $e$ . On part d'un élément  $a$  et d'un entier  $m > 0$ . On obtient alors  $a^m$  en appliquant les règles suivantes :

---

**ALGORITHME 1.14. — Calcul rapide des puissances**

---

*Entrées :  $a$ , entier  $m > 0$ . Sorties :  $a^m$ .*

*Règles :*

$$\begin{array}{lll} \text{puiss}(a, m) & \mapsto & (e, a, m) \\ [\text{n pair}] : & (y, x, n) & \mapsto (y, x^2, n/2) \\ [\text{n impair } \neq 1] : & (y, x, n) & \mapsto (yx, x^2, (n - 1)/2) \\ [n = 1] : & (y, x, n) & \mapsto yx \end{array}$$


---

Donnons un exemple : on a

$$\text{puiss}(a, 13) \mapsto (e, a, 13) \mapsto (a, a^2, 6) \mapsto (a, a^4, 3) \mapsto (a^5, a^8, 1) \mapsto a^{13}.$$

La terminaison de l'algorithme est assurée par le fait que, par application de la deuxième ou de la troisième règle, l'entier  $n$  est au moins divisé par 2. Ces règles ne peuvent donc être appliquées au total qu'au plus  $\log m$  fois, donc au plus  $\lfloor \log m \rfloor$  fois<sup>3</sup>. Par conséquent, cet algorithme utilise moins de  $2\lfloor \log m \rfloor$  multiplications. Par ailleurs,  $yx^n$  est invariant ; il vaut  $a^m$  au départ et  $yx$  à l'arrivée, ce qui prouve la correction de l'algorithme. On notera sur l'exemple précédent que les exposants des puissances calculées, à savoir 1, 2, 4, 5, 8 et 13 sont, outre les puissances de 2 inférieures à  $m$ , les entiers obtenus par troncature à gauche de l'écriture binaire de  $m = 13$ , soit  $(1)_2 = (01)_2 = 1$ ,  $(101)_2 = 5$  et  $(1101)_2 = 13$ .

---

3. Ici et dans tout le livre, on désigne par  $\log$  le logarithme à base 2 et par  $\lfloor \cdot \rfloor$  le plancher (partie entière par défaut).

**EXERCICE 1.7. [C] —** Calculer à l'aide de  $\lfloor \log m \rfloor$  et de  $S(m)$ , somme dans  $N$  des chiffres de l'écriture binaire de  $m$  (on a  $S(13) = 3$  par exemple), le nombre de multiplications nécessité par l'algorithme précédent.

Traduisons :

**PROGRAMME 1.15. — *Calcul rapide des puissances*** ——————

```
def puissrec(y,x,n)
    return x*y if n == 1
    puissrec((n.even? ? y : x*y), x*x, n.half)
end
class Integer
    def puiss(n)
        puissrec(1,self,n)
    end
end
3.puiss(100) # => 515377520732011331036461129765621272702107522001
```

---

**EXERCICE 1.8. [C] —** Revenons à la suite de Fibonacci. En se basant sur l'identité

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix},$$

donner un algorithme rapide de calcul de  $F_n$  et étudier le nombre de multiplications qu'il nécessite.

**REMARQUE 1.16. —** Bien évidemment, cela s'applique aussi lorsque la loi est notée additivement. Dans le cas de l'addition (dans un groupe commutatif d'élément unité noté 0), pour laquelle l'application « puissance » est donc la multiplication d'un élément  $a$  du groupe par un entier  $m > 0$ , on obtient l'algorithme de multiplication suivant, dit « égyptien » :

$$\begin{aligned} \text{mult}(a,m) &\mapsto (0,a,m) \\ [n \text{ pair}] : (y,x,n) &\mapsto (y,2x,n/2) \\ [n \text{ impair } \neq 1] : (y,x,n) &\mapsto (y+x,2x,(n-1)/2) \\ [n = 1] : (y,x,n) &\mapsto y+x \end{aligned}$$

### 1.2.2. Des poids forts vers les poids faibles

L'avantage essentiel de la méthode précédente, c'est qu'elle ne nécessite pas de connaître à l'avance l'écriture binaire de  $m$ . Elle possède deux inconvénients principaux. D'abord, lorsqu'on calcule dans une structure où la taille croît par multiplication et où donc la multiplication n'a pas un coût fixe (c'est le cas par exemple des entiers, par opposition à la situation d'un ensemble fini, tel que celui des entiers modulo un nombre fixé que nous verrons ci-dessous en 1.3.2), les deux « composantes »  $x$  et  $y$  croissent au cours du calcul et, par conséquent, les multiplications donnant  $x^2$  et  $xy$  sont toutes les deux de coût élevé.

Par ailleurs, cette méthode se généralise mal, contrairement à celle que nous allons expliquer maintenant, au calcul d'une fonction  $f(m)$  plus générale qu'une puissance  $a^m$ .

Le principe de la deuxième méthode est le suivant : tout nombre entier  $m \in \mathbb{N}$  peut se calculer à partir de 0 par itération des deux opérations suivantes  $D : n \mapsto 2n$ ,  $E : n \mapsto 2n + 1$ . Pour reprendre le cas  $m = 13$  ci-dessus, on a

$$13 = E6 = ED3 = EDE1 = EDEE0,$$

ce qui donne la chaîne de calcul  $0 \mapsto 1 \mapsto 3 \mapsto 6 \mapsto 13$ . Pour obtenir l'écriture d'un entier  $m$  sous la forme précédente, il suffit de prendre l'écriture binaire de  $m$ , ici  $13 = (1101)_2$ , de la renverser, et de remplacer 0 par D et 1 par E. Sous forme de réécriture, l'algorithme direct est le suivant :

$$\begin{aligned} [n \text{ pair } \neq 0] : \quad n &\mapsto Dn/2 \\ [n \text{ impair}] : \quad n &\mapsto E(n-1)/2 \end{aligned}$$

Par exemple :

```
def num_to_DE (n)
    return "" if n == 0
    (n.even? ? "D" : "E") + num_to_DE(n.half)
end
num_to_DE(123456) # => "DDDDDDDEDDEDDEEEE"

def DE_to_num(string)
    ret = 0
    str = string.dup
    until (letter = str[-1]) == nil
        case letter
            when "E"[-1] : ret = 2*ret + 1
            when "D"[-1] : ret = 2*ret
            else raise ArgumentError
        end
        str.chop!
    end
    ret
end
DE_to_num("DDDDDDDEDDEDDEEEE") # => 123456
```

Cela étant, si on a une application  $f$  de  $\mathbb{N}$  dans un ensemble de valeurs  $V$  et qu'on dispose de deux opérateurs  $D$  et  $E$  dans  $V$  tels que  $f(2n) = Df(n)$  et  $f(2n+1) = Ef(n)$ , il suffit d'appliquer à  $f(0)$  la même suite d'opérateurs. Par exemple, de  $13 = EDEE0$ , on tire  $f(13) = EDEEf(0)$ . On peut aussi supprimer le dernier E et partir de  $f(1)$ , comme dans  $f(13) = EDEf(1)$ . Cela donne, écrit sous forme récursive :

$$\begin{aligned} [n \text{ pair } \neq 0] : \quad f(n) &\mapsto Df(n/2) \\ [n \text{ impair}] : \quad f(n) &\mapsto Ef((n-1)/2) \end{aligned}$$

En partant de  $f(1)$ , on a la variante suivante :

$$\begin{aligned} [n \text{ pair}] : \quad f(n) &\mapsto D f(n/2) \\ [n \text{ impair } \neq 1] : \quad f(n) &\mapsto E f((n-1)/2) \end{aligned}$$

**EXERCICE 1.9. [B]** — Quel est le nombre de pas D et de pas E dans le calcul de  $f(n)$  ?

Dans le cas des puissances où  $f(n) = a^n$ , on aura  $Dx = x^2$  et  $Ex = ax^2$ . Comme on le voit, on retrouve essentiellement les mêmes applications de base que dans la première méthode, mais ici le multiplicateur intervenant dans E est fixe.

**EXERCICE 1.10. [C]** — On reprend l'exemple des nombres de Fibonacci. À l'aide de la représentation matricielle donnée à l'exercice 1.8, exprimer  $F_{2n}$ ,  $F_{2n+1}$  et  $F_{2n+2}$  à l'aide de  $F_n$  et  $F_{n+1}$ . Appliquer la méthode précédente en posant  $f(n) = (F_n, F_{n+1}) \in \mathbb{N}^2$  pour donner un algorithme de calcul de  $F_n$ . Étudier sa complexité.

### § 1.3. Complexité des algorithmes arithmétiques

Le nombre de chiffres d'un entier  $n > 0$  en représentation binaire est  $1 + \lfloor \log n \rfloor$ . Pour cette raison, et d'autres, on considère qu'une bonne définition de la « taille » d'un entier  $n$  (non nul) est le nombre réel  $\log |n|$ . Ainsi, la taille d'un produit est la somme de la taille des facteurs.

#### 1.3.1. Coût des opérations arithmétiques élémentaires

Suivant la taille des entiers que l'on manipule, « petits », « moyens », ou « gros », plusieurs modèles de coûts sont à envisager.

##### Le modèle à coûts fixes

Le modèle le plus simple est celui où l'on se restreint à de « petits » entiers, pour lesquels les opérations arithmétiques sont fournies avec la machine. Le coût d'une telle opération (addition, multiplication, division euclidienne) est alors indépendant de la taille des facteurs ; c'est le modèle à coûts fixes. Dans ce modèle, faire  $a$  additions (ou soustractions),  $m$  multiplications et  $d$  divisions a un coût de la forme  $Aa + Mm + Dd$ , donc majoré par une expression de la forme  $C(a + m + d)$ . C'est donc essentiellement le nombre total  $a + m + d$  d'opérations effectuées qui mesure le coût d'un calcul. Une variante, plus réaliste dans le cas fréquent des microprocesseurs où la multiplication est notablement plus lente que l'addition, consiste à ne considérer que le nombre  $m + d$  de multiplications et de divisions.

## Le modèle à coûts bilinéaires

Dès que l'on utilise des entiers de taille plus grande, le modèle à coûts fixes devient irréaliste, et l'on doit fixer pour les opérations élémentaires des coûts qui font intervenir la taille des arguments. Dans le cas le plus fréquent, celui des tailles « moyennes » (disons de quelques dizaines à une centaine de chiffres décimaux), le modèle le plus raisonnable reflète directement la manière la plus naïve de calculer « par tranches ». En notant  $c_+, c_\times$  et  $c_\div$  des constantes caractéristiques du matériel utilisé, on a donc :

- ▷ le coût d'une addition ou d'une soustraction  $r := m \pm n$  est majoré par  $c_+ \sup(\log |m|, \log |n|)$ ,
- ▷ le coût d'une multiplication  $r := m \times n$  est majoré par  $c_\times \log |m| \log |n|$ ,
- ▷ le coût de la division euclidienne de  $m$  par  $n$  avec  $0 < n \leq m$  est majoré par  $c_\div \log |m| \log |m/n| = c_\div \log |m| (\log |m| - \log |n|)$ .

Nous baptiserons ce modèle « modèle à coûts bilinéaires ». L'un des avantages du coût bilinéaire  $\log |m| \log |n|$  retenu pour la multiplication est son caractère « associatif » : le coût du calcul d'un produit de plusieurs facteurs ne dépend pas de la façon de le décomposer en suite de multiplications. En effet, le calcul de  $r := m \times n \times p$  par  $s := m \times n$  et  $r := s \times p$  est  $\log |m| \log |n| + \log |m \times n| \log |p|$ , soit

$$\log |m| \log |n| + \log |m| \log |p| + \log |n| \log |p|,$$

expression qui fait intervenir symétriquement les facteurs.

Bien entendu, de nombreuses et substantielles améliorations des algorithmes naïfs sont possibles. Parmi les plus évidentes, mentionnons que le coût de l'élévation au carré est en gros la moitié du coût d'une multiplication de nombres de taille analogue et qu'on peut utiliser des techniques de « pré-conditionnement » dans le cas où plusieurs multiplications font intervenir le même multiplicateur.

En outre, il existe des algorithmes de multiplication nettement plus rapides pour de très grands entiers. Nous verrons plus loin deux méthodes, celle de Pollard (4.3.4), suffisante pour tous les besoins pratiques, et celle de Schönhage-Strassen (4.3.5), d'un intérêt plus théorique, qui montre que la multiplication de deux entiers de taille  $n$  peut se faire avec un coût de la forme  $A \cdot n \cdot \log n \cdot \log \log n$ .

Lorsqu'on a affaire à des entiers qui atteignent le millier de chiffres décimaux, il devient intéressant d'utiliser ces techniques plus avancées.

Dans la suite, nous n'entrerons pas dans ces considérations et nous nous limiterons pour estimer les complexités au modèle à coûts bilinéaires.

### 1.3.2. Congruences dans $\mathbb{Z}$

La notion de congruence est ancienne. Les congruences modulo 2 ou 4 ont été utilisées par les Grecs (raisonnement par parité). La mise en forme

de cette notion est due à Legendre et Gauss.

Précisons qu'ici comme ailleurs, le mot *entier* signifie « entier de signe quelconque » et que les entiers  $0, 1, 2, \dots$  sont dits *naturels* ou *positifs*. On note **Z** l'anneau des entiers et **N** la partie formée des entiers naturels<sup>4</sup>.

Si  $a, b$  et  $N$  sont trois entiers, on dit que  $a$  et  $b$  sont *congrus modulo N*, et on écrit  $a \equiv b \pmod{N}$  si  $N$  divise  $a - b$ . Notons au passage que la congruence modulo  $-N$  est la même que la congruence modulo  $N$  et que la congruence modulo 0 est l'égalité.

On supposera donc ci-dessous  $N > 0$ ; dire que  $a$  et  $b$  sont congrus modulo  $N$  signifie alors qu'ils donnent le même reste par division euclidienne par  $N$ . L'ensemble des entiers congrus à  $a$  modulo  $N$ , s'appelle la *classe de congruence* de  $a$  modulo  $N$ . Il y a  $N$  classes de congruences modulo  $N$ , que l'on peut représenter par les entiers  $0, \dots, N - 1$ .

On peut additionner, soustraire, multiplier les congruences relatives au même module :

- ▷  $(x \equiv y \pmod{N} \text{ et } x' \equiv y' \pmod{N})$  implique  $x + x' \equiv y + y' \pmod{N}$ ,
- ▷  $(x \equiv y \pmod{N} \text{ et } x' \equiv y' \pmod{N})$  implique  $x - x' \equiv y - y' \pmod{N}$ ,
- ▷  $(x \equiv y \pmod{N} \text{ et } x' \equiv y' \pmod{N})$  implique  $xx' \equiv yy' \pmod{N}$ .

Notons aussi les propriétés suivantes :

- ▷  $x \equiv y \pmod{N}$  implique  $ax \equiv ay \pmod{aN}$ ,
- ▷ si  $N'$  divise  $N$ , alors  $x \equiv y \pmod{N}$  implique  $x \equiv y \pmod{N'}$ ,
- ▷ on a  $x \equiv y \pmod{1}$  quels que soient  $x$  et  $y$ .

En revanche, on ne peut pas toujours simplifier des congruences. De  $ax \equiv ay \pmod{N}$ , on ne peut déduire  $x \equiv y \pmod{N}$  que si  $a$  est premier à  $N$ . En fait, pour que  $a$  soit *inversible* modulo  $N$ , c'est-à-dire pour qu'on puisse trouver  $b$  avec  $ab \equiv 1 \pmod{N}$ , il faut et il suffit que  $a$  et  $N$  soient premiers entre eux (voir ci-dessous le corollaire 1.33).

Cela montre que la situation est meilleure lorsque le module est *premier* :

- ▷ Si  $p$  est premier, et si  $a \not\equiv 0 \pmod{p}$ , alors  $ax \equiv ay \pmod{p}$  implique  $x \equiv y \pmod{p}$ .

## L'anneau **Z/NZ**

On notera dans la suite **Z/NZ** l'ensemble à  $N$  éléments des classes de congruence modulo  $N$ . Soient  $\alpha$  et  $\beta$  deux classes. On a vu ci-dessus que pour  $a$  dans  $\alpha$  et  $b$  dans  $\beta$ , la classe de  $a + b$  ne dépend que de  $\alpha$  et  $\beta$ ; on la note évidemment  $\alpha + \beta$ . On définit de même le produit  $\alpha\beta$ . Muni de ces deux opérations, **Z/NZ** est un anneau, appelé *l'anneau quotient de Z* par la relation de congruence modulo  $N$ . Son élément nul est la classe de 0, son élément unité la classe de 1. Ses éléments inversibles sont les classes des

4. La notation **Z** provient de l'allemand « *Zahl* » (nombre) ... et non pas de l'anneau des-z-entiers ; quant à **N**, c'est sans doute l'initiale de « *nombre* » ou « *naturel* ».

entiers premiers à N. C'est un corps lorsque N est premier (on le note alors souvent  $\mathbb{F}_N$ , voir 9.1.3).

À chaque entier  $a$ , on peut associer sa classe  $a \bmod N$  dans  $\mathbb{Z}/\mathbb{Z}$ . Ainsi, la relation d'égalité  $a \bmod N = b \bmod N$  dans  $\mathbb{Z}/\mathbb{Z}$  est équivalente à la congruence  $a \equiv b \pmod{N}$  dans  $\mathbb{Z}$ . L'application  $a \mapsto a \bmod N$  de  $\mathbb{Z}$  dans  $\mathbb{Z}/\mathbb{Z}$  est par construction compatible avec l'addition et la multiplication et envoie 1 sur 1, donc est (par définition) un *homomorphisme d'anneaux*.

Signalons explicitement une ambiguïté de notations. On représente en général  $\mathbb{Z}/\mathbb{Z}$  par l'intervalle  $\{0, 1, \dots, N - 1\}$  de N. De ce fait, la classe  $a \bmod N$  de  $a$  est représentée par le reste de la division euclidienne de  $a$  par N, que l'on note alors souvent également  $a \bmod N$ . Lorsqu'on ne parle que d'un élément, il n'y a là aucun danger. Mais considérons maintenant deux entiers  $a$  et  $b$  et leurs classes. Nous avons noté  $a \bmod N + b \bmod N$  la somme des deux classes, qui est par définition la classe de la somme  $a + b$ , donc la classe représentée par le reste de  $a + b$ . Cette classe est aussi, bien évidemment, celle de la somme des restes de  $a$  et de  $b$ , mais ce dernier entier n'est pas forcément compris entre 0 et  $N - 1$ . Lorsqu'on écrit  $a \bmod N + b \bmod N$ , il faut donc savoir si l'on désigne ainsi l'addition des classes ou l'addition des restes. Dans toute la suite, c'est toujours de l'addition des classes qu'il s'agira. De même pour la multiplication.

### 1.3.3. Le coût du calcul modulaire

Supposons fixé un (grand) entier  $N > 0$  et intéressons-nous au calcul *modulo N*.

Nous représentons les classes modulo N par les entiers  $0, 1, \dots, N - 1$  et nous nous plaçons dans le modèle à coûts bilinéaires. Les opérations de base sont les suivantes :

a) L'opération de *réduction* modulo N d'un entier quelconque, notée  $a \mapsto a \bmod N$ , qui n'est autre que le calcul du reste de la division euclidienne. Son coût est donc majoré par  $c_{\pm} \log |a| \log |a/N|$ .

b) L'*addition* (ou soustraction) modulo N. On part donc  $a$  et  $b$  avec  $0 \leq a < N$  et  $0 \leq b < N$ . Puisqu'on a  $0 \leq a + b < 2N - 1$ , la somme de  $a$  et  $b$  modulo N est donnée par l'expression conditionnelle

$$\text{if } a + b - N < 0 \text{ then } a + b \text{ else } a + b - N.$$

On doit donc calculer  $a + b$ , puis  $a + b - N$ , et on prend comme résultat  $a + b$  si  $a + b - N$  est  $< 0$  et  $a + b - N$  sinon. La comparaison a un coût négligeable (elle porte sur un bit de signe), et le coût total est au plus de  $2c_{\pm} \log N$ . Le cas de la soustraction est analogue.

c) La *multiplication* modulo N. Partant de  $a$  et  $b$  comme ci-dessus, le produit  $ab \bmod N$  des classes de congruence  $a \bmod N$  et  $b \bmod N$  s'obtient par

division euclidienne du produit usuel  $ab$  par  $N$ . Le coût de la multiplication est au plus de  $c_x(\log N)^2$ , tandis que le coût de la division est au plus de

$$c_{\div} \log(N^2) \log N = 2c_{\div}(\log N)^2.$$

Au total, le coût est au plus égal à  $(c_x + 2c_{\div})(\log N)^2$ .

d) L'inversion modulo  $N$ . Soit l'entier  $a$  avec  $0 \leq a < N$ . Pour savoir si  $a$  est inversible modulo  $N$  et calculer son inverse, on utilise l'algorithme d'Euclide étendu (voir ci-dessous le programme 1.4.5), de coût majoré par  $c(\log N)^2$  avec  $c = c_{\div} + c_x$  (proposition 1.37).

### 1.3.4. Le coût de l'exponentielle

Nous avons vu plus haut que dans le modèle à coûts fixes, le calcul de  $a^n$  peut se faire par un algorithme dont le coût est de la forme  $c \log n$ . Nous nous proposons maintenant d'évaluer ce coût dans le modèle à coûts bilinéaires.

#### Le cas modulaire

Traitons d'abord un cas spécial, qui nous intéresse pour la suite (critères de primalité). Supposons fixé un (grand) entier  $N$  et proposons-nous de calculer  $a^n \bmod N$ , avec  $|a| \leq N$ .

On vient de voir que le coût de chaque multiplication modulo  $N$  est au plus égal à  $c(\log N)^2$  avec  $c = c_x + 2c_{\div}$ . L'algorithme rapide exposé dans le paragraphe précédent utilise au maximum  $2 \log n$  pas. On obtient par conséquent :

**PROPOSITION 1.17.** — *Le coût du calcul de  $a^n \bmod N$ , avec  $|a| < N$ , est majoré par  $(2c_x + 4c_{\div})(\log n)(\log N)^2$ .*

Compte-tenu du caractère approximatif du modèle de calcul retenu, la valeur précise des constantes importe assez peu, et nous retiendrons essentiellement le type de croissance des majorations trouvées. Pour simplifier le langage, nous abrégerons donc systématiquement les énoncés de ce genre, en *sous-entendant la constante et l'inégalité*. On dira par exemple : *le calcul de  $a^n \bmod N$  a un coût en  $(\log n)(\log N)^2$* .

Traduisons :

**PROGRAMME 1.18.** — *Calcul modulaire rapide des puissances* ——————

```
def puissmrec(y,x,n,m)
    return (y * x) % m if n == 1
    puissmrec((n.even? ? y : y*x % m), x*x % m, n.half, m)
end
class Integer
    def puissmod(n,m)
        puissmrec(1,self,n,m)
    end
end
```

Par exemple :

```
fermat = 2**32+1 # => 4294967297
2.puissmod(fermat,fermat) # => 2
3.puissmod(fermat,fermat) # => 497143886
```

## Le cas général

Passons maintenant au cas général, comme exemple des techniques que l'on peut mettre en œuvre dans ce genre de questions.

Nous supposons comme ci-dessus que le coût d'une multiplication  $r := m \times n$  est majoré par  $c_x \log |m| \log |n|$ . Fixons l'entier  $a$ . Notons  $C(n)$  le coût du calcul de  $a^n$  par le deuxième algorithme rapide exposé ci-dessus qui, rappelons-le, est basé sur les deux faits suivants :

$$a^{2n} := a^n \times a^n, \quad a^{2n+1} := a^n \times a^n \times a.$$

On a donc

$$\begin{aligned} C(2n) &= C(n) + c_x(n \log a)^2, \\ C(2n+1) &= C(n) + c_x(n \log a)^2 + c_x \log a(2n \log a), \end{aligned}$$

que nous compléterons par  $C(0) = C(1) = 0$ . Pour simplifier les notations, mettons en facteurs les constantes  $c_x$  et  $\log a$  en posant

$$C(n) = c_x(\log a)^2 \gamma(n),$$

de sorte qu'on a  $\gamma(0) = \gamma(1) = 0$ , et

$$\gamma(2n) = \gamma(n) + n^2, \quad \gamma(2n+1) = \gamma(n) + n^2 + 2n.$$

On peut donc calculer  $\gamma(n)$  pour tout  $n$  par récursion :

```
def gamma (n)
    return 0 if n < 2
    m = n.half
    gamma(m)+ m*m + (n.even? ? 0 : n-1)
end
(101...105).collect { |n| gamma(n)} # => [3441, 3492, 3594, 3609]
(101...105).collect { |n| n*(n+2)/3} # => [3467, 3536, 3605, 3674]
```

**EXERCICE I.11. [B]** — Montrer qu'on a  $\gamma(n) < \gamma(n+1)$  pour  $n > 0$ .

**LEMME I.19.** — On a  $\gamma(n) \leq \frac{1}{3}(n^2 + 2n)$ .

*Démonstration.* Pour imaginer la forme d'une majoration, supposons d'abord que l'on n'ait affaire qu'à des puissances de 2. On a alors

$$\gamma(n) = \frac{n^2}{4} + \gamma(n/2) = \frac{n^2}{4} + \frac{n^2}{16} + \gamma(n/4) = \dots$$

On voit apparaître la série convergente  $1/4 + 1/16 + 1/64 + \dots$ , donc une majoration de la forme  $\gamma(n) \leq an^2$ . Pour que cela marche, il faut que  $an^2 + n^2 \leq a(2n)^2 =$

$4an^2$ , et on peut prendre  $a = 1/3$ . D'ailleurs la somme  $a$  de la série précédente est évidemment telle que  $4a = 1 + a$ , soit  $a = 1/3$ . Mais cela ne suffit pas, car la formule pour  $\gamma(2n+1)$  donne alors  $\gamma(2n+1) \leq n^2/3 + n^2 + 2n = 4n^2/3 + 2n$ , alors que  $(2n+1)^2/3$  vaut seulement  $4n^2/3 + 4n/3 + 1/3$ . Si l'on suppose (sauvagement) la fonction  $\gamma$  définie pour tous les nombres réels positifs et croissante, on a dans tous les cas  $\gamma(n) \leq \gamma(n/2) + n^2/4 + n$ , d'où comme ci-dessus

$$\gamma(n) \leq \frac{n^2}{4} + n + \gamma(n/2) \leq n^2\left(\frac{1}{4} + \frac{1}{16}\right) + n\left(1 + \frac{1}{2}\right) + \gamma(n/4) = \dots$$

et on voit apparaître une deuxième série convergente  $1 + 1/2 + 1/4 + \dots$ , d'ailleurs de somme 2.

On va donc chercher une majoration de la forme  $\gamma(n) \leq an^2 + bn$ . On veut que  $an^2 + bn + n^2 \leq a(2n)^2 + b(2n)$  et  $an^2 + bn + n^2 + 2n \leq a(2n+1)^2 + b(2n+1)$ . On obtient alors sans difficulté la solution  $a = 1/3$  et  $b = 2/3$ , donc la majoration  $\gamma(n) \leq (n^2 + 2n)/3$  annoncée.  $\square$

On en tire  $\gamma(n) < (n+1)^2/3$ , soit

$$C(n) < \frac{c \times (\log a)^2}{3} (n+1)^2.$$

En résumé :

**PROPOSITION 1.20.** — *Dans le modèle à coûts bilinéaires, pour a fixé, le coût du calcul de  $a^n$  est en  $n^2$ .*

De manière équivalente, on peut retenir que dans le calcul de  $a^n$ , le dernier pas d'élévation au carré (de coût approximatif  $c \times (\log a)^2 (n/2)^2$ ) intervient en gros pour trois quarts dans le coût total, le quatrième quart étant essentiellement le calcul récursif de  $a^{\lfloor n/2 \rfloor}$ .

## § 1.4. Algorithmes d'Euclide

### 1.4.1. L'algorithme de base

L'algorithme originel donné par Euclide<sup>5</sup> pour calculer le plus grand commun diviseur (pgcd) de deux entiers positifs  $u$  et  $v$  consiste à soustraire le plus petit des deux entiers  $u$  et  $v$  de l'autre, et à répéter l'opération jusqu'à ce qu'un des deux nombres divise l'autre ; sa valeur donne alors le pgcd. Il revient essentiellement au même de continuer les soustractions jusqu'à ce qu'un des nombres s'annule, l'autre valant alors le pgcd. Cet algorithme s'exprime le plus commodément par des règles de réécriture :

---

5. Propositions 1 et 2 du livre 7 des Éléments, dû sans doute à Eudoxe (environ 375 avant notre ère).

**ALGORITHME 1.21.** — *Algorithme d'Euclide, version soustractive*

*Entrées* : entiers  $u \geq 0$  et  $v \geq 0$ . *Sorties* :  $\text{pgcd}(u, v)$ .

Règles :

$$\begin{aligned} [0 < u \leq v] &: (u, v) \mapsto (u, v - u) \\ [0 < v \leq u] &: (u, v) \mapsto (u - v, v) \\ [u = 0] &: (u, v) \mapsto v \\ [v = 0] &: (u, v) \mapsto u \end{aligned}$$


---

La terminaison est évidente : l'application d'une des deux premières règles fait diminuer strictement la somme  $u + v$ , et on doit arriver nécessairement à l'une des deux dernières.

Le fait que le résultat  $d$  est bien le plus grand commun diviseur de  $u$  et  $v$  est une conséquence des relations immédiates suivantes :

$$\begin{aligned} \text{pgcd}(u, v) &= \text{pgcd}(u, v - u) = \text{pgcd}(u - v, v), \\ \text{pgcd}(0, u) &= \text{pgcd}(u, 0) = u. \end{aligned}$$

On notera au passage que cet algorithme prouve en même temps l'existence et l'unicité du pgcd de deux entiers positifs.

**EXERCICE 1.12.** [B] — Soit  $f : \mathbf{N} \rightarrow \mathbf{N}$  une fonction telle qu'on ait pour tout couple d'entiers  $m \geq n$  la relation  $\text{pgcd}(f(m), f(n)) = \text{pgcd}(f(m-n), f(n))$ . Prouver que pour tout couple d'entiers  $u$  et  $v$ , de pgcd  $d$ , le pgcd de  $f(u)$  et  $f(v)$  est  $f(d)$ . Application :  $f(n) = a^n - b^n$  avec  $\text{pgcd}(a, b) = 1$ .

On notera dans l'algorithme 1.21 l'indétermination lorsque  $u = v$ . On peut lever cette indétermination et éviter de doubler chaque règle par la modification innocente suivante :

$$\begin{aligned} [0 < u \leq v] &: (u, v) \mapsto (u, v - u) \\ [u > v] &: (u, v) \mapsto (v, u) \\ [u = 0] &: (u, v) \mapsto v \end{aligned}$$

Remarquons alors que cet algorithme réalise automatiquement des divisions euclidiennes : chaque application de la première règle est obligatoirement répétée tant que le nouveau nombre n'est pas devenu strictement inférieur à l'autre. Notons  $a \div b$  le quotient entier de  $a$  par  $b$  : on a  $a = (a \div b)b + r$ , avec  $0 \leq r < b$ . On obtient alors un algorithme essentiellement équivalent :

$$\begin{aligned} [0 < u \leq v] &: (u, v) \mapsto (u, v - (v \div u)u) \\ [u > v] &: (u, v) \mapsto (v, u) \\ [u = 0] &: (u, v) \mapsto v \end{aligned}$$

Notons maintenant que chaque application de la règle de division renverse l'ordre des facteurs, donc nécessite l'application de la deuxième règle. Introduisant la règle composée, on obtient la version suivante :

$$\begin{aligned}[0 < u \leq v] : \quad (u, v) &\mapsto (v - (v \div u)u, u) \\ [u > v] : \quad (u, v) &\mapsto (v, u) \\ [u = 0] : \quad (u, v) &\mapsto v\end{aligned}$$

Sous cette forme, il est maintenant clair que la deuxième règle n'est utilisée qu'au plus une fois et en premier. Comme par ailleurs, il suffit d'étendre le champ d'application de la première règle pour qu'elle inclue la seconde, on obtient la version finale :

---

**ALGORITHME 1.22.** — *Algorithme d'Euclide, version classique*

---

*Entrées* : entiers  $u \geq 0$  et  $v \geq 0$ . *Sorties* :  $\text{pgcd}(u, v)$ .

*Règles* :

$$\begin{aligned}[u \neq 0] : \quad (u, v) &\mapsto (v - (v \div u)u, u) \\ [u = 0] : \quad (u, v) &\mapsto v\end{aligned}$$


---

Traduit en Ruby, cela donne :

---

**PROGRAMME 1.23.** — *Algorithme d'Euclide, version de base*

---

```
def pgcd(u,v)
  u == 0 ? v : pgcd(v%u, u)
end
a,b,d = 77777462, 12345653, 12345
pgcd(a*d, b*d) == d # => true
```

---

**REMARQUE 1.24.** — La valeur exacte de l'entier  $q = v \div u$  ne joue pas de rôle essentiel. En définitive, le centre de l'algorithme, c'est la transformation  $(u, v) \mapsto (v - qu, u)$  et on a de toutes façons  $\text{pgcd}(v - qu, u) = \text{pgcd}(u, v)$ . La vertu essentielle du choix  $q = v \div u$ , c'est d'assurer la terminaison de l'algorithme. Mais ce n'est pas le seul possible. Il suffit par exemple d'imposer  $|v - qu| < |u|$ , ce qui permet en général deux valeurs de  $q$ . On peut aussi imposer  $|v - qu| \leq |u|/2$ , ce qui donne un algorithme plus rapide<sup>6</sup>.

Notons qu'après le premier pas de division euclidienne, il est fort probable que les quotients successifs soient petits. En effet, la seule chose que l'on sache sur le premier reste  $w = u \bmod v$ , c'est qu'il est compris entre 0 et  $v$ , donc a une « probabilité » en gros  $1/2$  d'être supérieur à  $v/2$ , une « probabilité » en gros  $2/3$  d'être supérieur à  $v/3$ , etc, ce qui donne pour le deuxième quotient  $v \bmod w$  une « probabilité » en gros  $1/2$  d'être égal à 1, une « probabilité » en gros  $2/3$  d'être  $\leq 2$ , etc. Ainsi, l'algorithme classique ne différera pas beaucoup, en dehors du premier pas, de l'algorithme soustractif.

---

6. C'est en fait l'algorithme le plus rapide, voir [Naudin, Quitté], page 161, exercice 27.

**EXERCICE I.13.** [N] — Écrire et implanter un algorithme « mixte », commençant par un pas euclidien, puis continuant par des pas soustractifs. Expérimenter avec les trois algorithmes.

On peut calculer le ppcm de deux entiers à partir de leur pgcd (voir par exemple l'exercice I.24) :

```
def ppcm(u,v)
  u*(v/pgcd(u,v))
end
a,b,d = 77777462, 12345653, 12345
ppcm(a*d,b*d) == a*b*d # => true
```

### I.4.2. L'algorithme d'Euclide binaire

Avant d'aller plus loin, donnons une variante « binaire » de l'algorithme d'Euclide. On suppose avoir déterminé la puissance de 2 qui intervient dans le pgcd et avoir divisé les nombres considérés par cette puissance. On part donc de deux entiers dont l'un au moins est impair, disons  $u > 0$  et  $v \geq 3$  impair.

**ALGORITHME I.25.** — *Algorithme d'Euclide, version binaire*

*Entrées* : entiers  $u > 0$  et  $v \geq 3$  impair. *Sorties* :  $\text{pgcd}(u, v)$ .

Règles :

$$\begin{aligned} [u \text{ pair}] &: (u, v) \mapsto (u/2, v) \\ [u \text{ impair} > v] &: (u, v) \mapsto ((u - v)/2, v) \\ [u \text{ impair} < v] &: (u, v) \mapsto ((v - u)/2, u) \\ [u = v] &: (u, v) \mapsto u \end{aligned}$$

Si on utilise du « pattern-matching », et si l'on suppose que les variables ne prennent que des valeurs entières, on peut encore écrire cet algorithme sous la forme suivante :

$$\begin{aligned} (2n, 2m + 1) &\mapsto (n, 2m + 1) \\ [n > m] &: (2n + 1, 2m + 1) \mapsto (n - m, 2m + 1) \\ [n < m] &: (2n + 1, 2m + 1) \mapsto (m - n, 2n + 1) \\ [n = m] &: (2n + 1, 2m + 1) \mapsto 2n + 1 \end{aligned}$$

**EXERCICE I.14.** [B] — Démontrer la validité de cet algorithme.

En Ruby, en incorporant la réduction initiale :

**PROGRAMME I.26.** — *Algorithme d'Euclide, version binaire*

```
def pgcd(u,v)
  return u if u == v
  case
    when u.even? && v.even?
      2*pgcd(u.half, v.half)
```

```

when u.even?
    pgcd(u.half, v)
when v.even?
    pgcd(u, v.half)
else
    pgcd((u > v) ? (u-v).half : (v-u).half, u)
end
end
pgcd(2*12345*77777462, 2*12345*12345653) # => 24690

```

---

Cet algorithme est particulièrement rapide. En effet, pour  $0 \leq u \leq N$  et  $0 \leq v \leq N$ , le nombre de pas de réécriture est majoré par  $2 \log N$ . En outre, chaque pas comporte au plus une soustraction et une opération de décalage (division par 2 d'un entier pair). On obtient donc un coût théorique en  $c(\log N)^2$ . On verra ci-dessous qu'il en est de même pour l'algorithme classique, mais on doit s'attendre à une efficacité pratique supérieure pour l'algorithme binaire, compte-tenu notamment de la difficulté d'une programmation optimale de la division euclidienne des grands nombres.

**EXERCICE 1.15.** [N] — Implémenter cet algorithme et comparer son temps d'exécution réel à celui des trois autres proposés.

### 1.4.3. La suite de Fibonacci

La *suite de Fibonacci*<sup>7</sup> est définie récursivement par

$$F_0 = 0, \quad F_1 = 1, \quad F_{n+1} = F_n + F_{n-1}.$$

On a ainsi par exemple  $(F_n)_{0 \leq n \leq 6} = (0, 1, 1, 2, 3, 5, 8)$ . Notons l'identité

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix},$$

valable pour tout  $n > 0$ , qui implique notamment  $F_{n+1}F_{n-1} - F_n^2 = (-1)^n$ .

**EXERCICE 1.16.** [A] — Vérifier cette identité.

**EXERCICE 1.17.** [A] — On a  $F_{n+m} = F_{n+1}F_m + F_nF_{m-1}$ .

**EXERCICE 1.18.** [B] — En déduire les égalités  $\text{pgcd}(F_{n+m}, F_m) = \text{pgcd}(F_m, F_n)$ , puis  $\text{pgcd}(F_n, F_m) = F_{\text{pgcd}(m,n)}$ .

**EXERCICE 1.19.** [B] — On prolonge la définition des  $F_n$  en acceptant des indices  $n < 0$ . Que vaut  $F_n$  pour  $n < 0$ ?

**EXERCICE 1.20.** [B] — Considérons le « nombre d'or »  $\phi = (\sqrt{5} + 1)/2 = 1,6180\dots$  et son inverse  $1/\phi = \phi - 1 = (\sqrt{5} - 1)/2 = 0,6180\dots$  Prouver qu'on a  $F_n = \frac{1}{\sqrt{5}}(\phi^n - (-\phi)^{-n})$ . En déduire que  $F_n$  est l'entier le plus proche de  $\frac{1}{\sqrt{5}}\phi^n$ .

<sup>7</sup>. Cette suite a été introduite par Léonard de Pise, dit Fibonacci (filius Bonacci) en 1202 dans son *Liber Abbaci*.

### 1.4.4. Le coût de l'algorithme d'Euclide

**PROPOSITION 1.27** (Lamé, 1845). — Soient  $x$  et  $y$  deux entiers avec  $0 < y < x$  et soit  $d$  leur pgcd. Si l'algorithme d'Euclide usuel partant de  $(x, y)$  s'arrête au bout de  $n$  pas, on a

$$x \geq d F_{n+2}, \quad y \geq d F_{n+1}.$$

*Démonstration.* On raisonne par récurrence. Dire que  $n = 1$ , c'est dire que  $x$  est multiple de  $y$ ; on a alors  $y = d = d F_2$  et  $x \geq 2d = d F_3$ . Supposons  $n > 1$ . Alors le premier pas transforme  $(x, y)$  en  $(y, z)$  avec  $z = x - qy \leq x - y$ . On a par l'hypothèse de récurrence appliquée au couple  $(y, z)$ ,  $y \geq d F_{n+1}$  et  $z \geq d F_n$ , donc  $x \geq y + z \geq d F_n + d F_{n+1} = d F_{n+2}$ .  $\square$

On notera le cas minimal :

$$(F_{n+1}, F_{n+2}) \mapsto (F_n, F_{n+1}) \mapsto \dots \mapsto (2, 3) \mapsto (1, 2) \mapsto (0, 1) \mapsto 1.$$

**COROLLAIRE 1.28.** — Soient  $x$  et  $y$  deux entiers avec  $0 \leq y \leq x$ . L'algorithme d'Euclide calculant le pgcd de  $x$  et  $y$  prend au plus  $\frac{3}{2} \log y + 1$  pas.

*Démonstration.* En effet, si l'algorithme prend  $n$  pas, on a  $F_{n+1} \leq y$ . Par ailleurs, on a  $n \leq \frac{3}{2} \log(F_{n+1}) + 1$ .  $\square$

**EXERCICE 1.21. [B]** — Vérifier la majoration de  $n$  ci-dessus.

Évaluons maintenant le coût de cet algorithme dans le modèle à coûts bilinéaires : rappelons que l'on fait l'hypothèse que le coût du calcul du reste  $v \bmod u = v - (v \div u)u$  est majoré par  $c_{\div} \log v \log(v/u) = c_{\div}((\log v)^2 - \log v \log u)$ . Cela étant, le calcul est le suivant : on pose  $x_0 = v$ ,  $x_1 = u$ , puis  $x_{i+1} = x_{i-1} \bmod x_i$  et on s'arrête lorsque  $x_n = 0$ , avec alors  $x_{n-1} = \text{pgcd}(u, v)$ . Le coût total, au facteur multiplicatif  $c_{\div}$  près, est donc

$$(\log x_0)^2 - \log x_0 \log x_1 + (\log x_1)^2 - \log x_1 \log x_2 + \dots,$$

et on constate que les valeurs absolues des termes de cette somme alternée vont en décroissant. Elle est donc majorée par son premier terme :

**PROPOSITION 1.29.** — Dans le modèle à coûts bilinéaires, le coût du calcul du pgcd de  $u$  et  $v$ , avec  $0 \leq u < v$ , est majoré par  $c_{\div}(\log v)^2$ .

**EXERCICE 1.22. [C]** — Le modèle de coût retenu pour la division euclidienne n'est pas totalement réaliste. Au lieu de  $C_{\div} \log v \log(v/u)$ , il faudrait plutôt prendre  $C_{\div}(\log v)(1 + \log(v/u))$ , voire  $C_{\div}(1 + \log v)(1 + \log(v/u))$ . En utilisant le fait que le nombre de pas de l'algorithme est logarithmique, prouver que l'on garde cependant une majoration du coût total de la forme  $c(\log v)^2$ .

### 1.4.5. L'algorithme d'Euclide étendu

En fait, une variante de l'algorithme d'Euclide permet de déterminer, deux entiers  $x$  et  $y$  étant donnés, non seulement leur pgcd  $d$ , mais aussi deux

entiers  $a$  et  $b$  tels que  $d = ax + by$ . Il suffit en fait d'effectuer des réécritures, non pas sur des entiers  $u$ , mais sur des triplets  $(u, a, b)$  avec  $ax + by = u$ . Voici cet algorithme étendu et sa version Ruby :

**ALGORITHME I.30.** — *Algorithme d'Euclide étendu (version symétrique)*

*Entrées* : entiers positifs  $x$  et  $y$ .

*Sorties* : entiers  $d$ ,  $a$  et  $b$  avec  $d = \text{pgcd}(x, y)$  et  $ax + by = d$ .

*Règles* :

$$\begin{array}{lll} (x, y) & \mapsto & (x, 1, 0, y, 0, 1) \\ [u \neq 0] : (u, c, d, v, a, b) & \mapsto & (v - qu, a - qc, b - qd, u, c, d) \text{ où } q = v \div u \\ [u = 0] : (u, c, d, v, a, b) & \mapsto & (v, a, b) \end{array}$$


---

**EXERCICE I.23. [B]** — Vérifier la validité de cet algorithme.

**PROGRAMME I.31.** — *Algorithme d'Euclide étendu*

```
def bezrec(u,c,d,v,a,b)
    return [v,a,b] if u == 0
    q = v / u
    bezrec(v-q*u, a-q*c, b-q*d, u, c, d)
end
def bezout(x,y)
    bezrec(x,1,0,y,0,1)
end
d,a,b = bezout(12345,54321) # =>[3, 3617, -822]
d == a*12345+b*54321 # => true
```

---

On en déduit l'énoncé bien connu :

**PROPOSITION I.32.** — *Soient  $x$  et  $y$  deux entiers, et soit  $d$  leur pgcd. Il existe des entiers  $a$  et  $b$  tels que  $ax + by = d$  (« identité de Bézout<sup>8</sup> »).*

**COROLLAIRE I.33.** — *Soient  $x$  et  $y$  deux entiers. Les conditions suivantes sont équivalentes :*

- (i)  *$x$  et  $y$  sont premiers entre eux : ils n'ont pas de diviseur commun autre que 1 et  $-1$ ,*
- (ii) *il existe  $a$  et  $b$  dans  $\mathbf{Z}$  avec  $ax + by = 1$ .*

*Démonstration.* On vient de voir que (i) implique (ii). Inversement, si (ii) est vrai, tout diviseur commun à  $x$  et  $y$  divise  $ax + by$ , donc divise 1.  $\square$

8. On verra dans [Delahaye], page 148, que ce résultat est à attribuer à Bachet de Méziriac, plus de 100 ans avant la naissance de Bézout. Au passage, signalons que Bézout écrivait effectivement son nom avec un accent aigu.

**EXERCICE I.24. [B]** — Déduire de la proposition que les multiples communs de  $x$  et  $y$  sont exactement les multiples de  $xy/d$ , qui est donc le plus petit commun multiple (*ppcm*) de  $x$  et  $y$ .

En fait, l'algorithme précédent est inutilement compliqué. En effet, lorsqu'on connaît  $a$ , on peut récupérer  $b$  par la division (exacte) de  $d - ax$  par  $y$ . Supprimant le calcul de  $b$ , on obtient la variante suivante :

**ALGORITHME I.34. — Algorithme d'Euclide étendu (version dissymétrique)** —————

*Entrées* : entiers positifs  $x$  et  $y$ .

*Sorties* : entiers  $d$  et  $a$  avec  $d = \text{pgcd}(x, y)$  et  $ax \equiv d \pmod{y}$ .

*Règles* :

$$\begin{array}{lll} (x, y) & \mapsto & (x, 1, y, 0) \\ [u \neq 0] : & (u, c, v, a) & \mapsto (v - qu, a - qc, u, c) \text{ avec } q = v \div u \\ [u = 0] : & (u, c, v, a) & \mapsto (v, a) \end{array}$$


---

Ce deuxième algorithme peut aussi être vu comme celui de l'inversion de  $x$  modulo  $y$  : pour que  $x$  soit inversible modulo  $y$ , il faut et il suffit que  $d$  soit égal à 1, et dans ce cas  $a$  est un inverse de  $x$ .

Par exemple en Ruby :

**PROGRAMME I.35. — Inversion modulaire** —————

```
def bezrec2(u,c,v,a)
  return [v,a] if u == 0
  q = v/u
  bezrec2(v-q*u, a-q*c, u, c)
end
class Integer
  def invmod(y)
    v,a = bezrec2(self,1,y,0)
    a % y
  end
end
```

---

#### I.4.6. Le coût de l'algorithme d'Euclide étendu

Les deux algorithmes précédents ont le même nombre de pas de réécriture que l'algorithme d'Euclide usuel. Pour en déterminer le coût dans le modèle à coûts bilinéaires, il nous faut étudier d'un peu plus près leur fonctionnement. On part des entiers

$$x_0 = x, a_0 = 1, b_0 = 0, \quad x_1 = y, a_1 = 0, b_1 = 1,$$

puis on pose

$$x_{i+1} = x_{i-1} - q_i x_i, \quad a_{i+1} = a_{i-1} - q_i a_i, \quad b_{i+1} = b_{i-1} - q_i b_i,$$

avec  $q_i = x_{i-1} \div x_i$ . On s'arrête lorsque  $x_{n+1} = 0$ , et on obtient  $x_n = d$ ,  $a_n = a$ , et  $b_n = b$ . On a constamment  $a_i x + b_i y = x_i$ , et donc  $ax + by = d$ .

**LEMME 1.36.** — Pour  $1 \leq i \leq n + 1$ , les  $a_i$  sont de signes alternés, de valeurs absolues strictement croissantes, et on a  $a_{n+1} = (-1)^{n+1}y/d$ .

*Démonstration.* On a  $(-1)^{i+1}a_{i+1} = (-1)^{i-1}a_{i-1} + q_i(-1)^ia_i$ , et les  $q_i$  sont strictement positifs. Cela implique les deux premières assertions. D'autre part, un calcul immédiat donne

$$x_i a_{i+1} - x_{i+1} a_i = -(x_{i-1} a_i - x_i a_{i-1}),$$

donc  $x_i a_{i+1} - x_{i+1} a_i = (-1)^i(x_0 a_1 - x_1 a_0) = (-1)^{i-1}y$ . Prenant  $i = n$ , on obtient la dernière assertion.  $\square$

**PROPOSITION 1.37.** — Soient  $N$  un entier majorant  $x$  et  $y$ . Dans le modèle à coûts bilinéaires, le coût de l'algorithme d'Euclide étendu appliqué aux entiers  $x$  et  $y$  est majoré par  $c(\log N)^2$ , avec  $c = c_{\div} + 2c_{\times}$  pour l'algorithme symétrique et  $c = c_{\div} + c_{\times}$  pour l'algorithme dissymétrique.

*Démonstration.* Il faut en effet ajouter au coût de l'algorithme classique, que l'on sait déjà être de la forme précédente, celui du calcul récurrent des  $a_i$  et des  $b_i$ . Prenons le cas des  $a_i$ , celui des  $b_i$  étant analogue. On a d'après le lemme  $|a_i| \leq y/d \leq N$ . Le coût cumulé des multiplications  $a_i \times q_i$  est donc majoré par  $c_{\times} \sum \log q_i \log N$ . Or on a  $\sum_i \log q_i \leq \log(\prod q_i) \leq \log x \leq \log N$ .  $\square$

**REMARQUE 1.38.** — La relation  $a_{n+1} = (-1)^{n+1}y/d$  et la relation analogue  $b_{n+1} = (-1)^n x/d$  redonnent  $a_{n+1}x + b_{n+1}y = x_{n+1} = 0$ . On voit d'ailleurs au passage que pour tout couple d'entiers  $(u, v)$  tel que  $ux + vy = 0$ , il existe un entier  $r$  tel que  $u = ra_{n+1}$  et  $v = rb_{n+1}$ .

## § 1.5. Algorithmes d'Euclide pour les polynômes

### 1.5.1. Polynômes en une variable

Soit  $A$  un anneau<sup>9</sup>. Les polynômes à coefficients dans  $A$  en une variable  $X$  forment un anneau noté  $A[X]$ . Tout polynôme  $P$  s'écrit de façon unique  $\sum_{i \in \mathbb{N}} \alpha_i X^i$ , où les  $\alpha_i$  non nuls sont en nombre fini. On écrit aussi  $P(X)$  au lieu de  $P$  lorsqu'on veut attirer l'attention sur le nom de la variable. Si  $P$  n'est pas nul, son coefficient non nul d'indice le plus élevé (cet indice est le *degré* de  $P$ ) s'appelle son coefficient *dominant* (nous le noterons  $\text{dom}(P)$  si besoin est) et le terme correspondant son terme dominant. On dit que  $P$  est *unitaire* si son coefficient dominant est égal à 1.

On note  $\deg(P) \in \mathbb{N}$  le degré du polynôme non nul  $P$ . On pose  $\deg(0) = -\infty$ ; ainsi, dire que  $\deg(P) < m$  signifie que  $\alpha_m = \alpha_{m+1} = \dots = 0$ .

Soient  $P$  et  $Q$  deux polynômes non nuls. Si le coefficient dominant de  $P$  n'est pas un diviseur de zéro<sup>10</sup> (par exemple si  $P$  est unitaire), le produit des

9. Commutatif, comme tous les anneaux considérés dans ce livre.

10. C'est-à-dire si son produit par un élément non nul ne peut être nul.

termes dominants de  $P$  et  $Q$  est non nul, et on a par conséquent  $\deg(PQ) = \deg(P)\deg(Q)$ . En particulier,  $P(X)$  n'est pas non plus un diviseur de 0.

### 1.5.2. Division euclidienne des polynômes

**PROPOSITION 1.39.** — Soient  $U$  et  $V$  deux polynômes de  $A[X]$ . Supposons que le coefficient dominant de  $V$  soit inversible dans  $A$ . Il existe alors deux polynômes  $Q$  et  $R$ , uniquement déterminés, avec  $U = VQ + R$  et  $\deg(R) < \deg(V)$ .

*Démonstration.* Si  $U = VQ + R = VQ' + R'$ , alors  $R' - R = V(Q - Q')$ . Si  $Q \neq Q'$ , on a  $\deg(R' - R) = \deg(V) + \deg(Q - Q') \geq \deg(V)$ , ce qui est contradictoire. Cela prouve l'unicité.

Écrivons  $V = bX^m + \dots$ ,  $m = \deg(V)$ . Prouvons l'existence de  $Q$  et  $R$  par récurrence sur le degré  $n$  de  $U$ . Si  $n < m$ , on prend  $Q = 0$  et  $R = U$ . Supposons  $n \geq m$ , et écrivons  $U = aX^n + \dots$ , avec  $a \neq 0$ . Posons  $U' = U - ab^{-1}X^{n-m}V$ , de sorte que  $\deg(U') < n$ . Appliquant l'hypothèse de récurrence à  $U'$ , et écrivant  $U' = VQ' + R'$ , on obtient  $U = V(ab^{-1}X^{n-m} + Q') + R'$ .  $\square$

**EXERCICE 1.25. [A]** — On a  $\deg(Q) = \max(0, \deg(U) - \deg(V) + 1)$ .

**EXERCICE 1.26. [B]** — Qu'obtient-on pour  $R$  lorsque  $V(X) = X - a$  ? que vaut  $Q(a)$  ?

**EXERCICE 1.27. [B]** — Soient  $U$  et  $V$  deux polynômes de  $A[X]$ . Notons  $b$  le coefficient dominant de  $V$ , et soit  $r$  l'entier  $\max(0, \deg(U) - \deg(V) + 1)$ . Prouver qu'il existe deux polynômes  $Q$  et  $R$  avec  $b^r U = VQ + R$  et  $\deg(R) < \deg(V)$ .

On peut exprimer par un algorithme de réécriture la division euclidienne des polynômes.

---

#### ALGORITHME 1.40. — Division euclidienne des polynômes

---

*Entrées* : polynômes  $U$  et  $V$  avec  $\text{dom}(V)$  inversible.

*Sorties* : polynômes  $Q$  et  $R$  avec  $U = VQ + R$  et  $\deg(R) < \deg(V)$ .

*Règles* :

$$\begin{array}{lll} (U, V) & \mapsto & (0, U, V) \\ [\deg(R) \geq \deg(V)] : & (Q, R, V) & \mapsto (Q + A, R - AV, V) \\ & & \text{avec } A = \frac{\text{dom}(R)}{\text{dom}(V)} X^{\deg(R) - \deg(V)} \\ [\deg(R) < \deg(V)] : & (Q, R, V) & \mapsto (Q, R) \end{array}$$


---

**EXERCICE 1.28. [A]** — Vérifier la validité de cet algorithme.

### 1.5.3. Algorithmes d'Euclide

L'algorithme d'Euclide classique et l'algorithme d'Euclide étendu dans ses deux formes s'étendent immédiatement au cas des polynômes à une variable à coefficients dans un corps.

Considérons donc un corps  $K$  et l'anneau  $K[X]$  des polynômes à une variable à coefficients dans  $K$ .

Partons de deux polynômes  $P_0$  et  $P_1$ , avec  $\deg(P_1) \leq \deg(P_0)$ . Construisons par divisions euclidiennes successives deux suites de polynômes  $(P_i)$  pour  $i \geq 2$  et  $(Q_i)$  pour  $i \geq 1$ , avec pour  $i = 1, \dots$ :

$$\begin{aligned} P_{i-1} &= P_i Q_i + P_{i+1}, \\ \deg(Q_i) &= \deg(P_{i-1}) - \deg(P_i), \\ \deg(P_{i+1}) &< \deg(P_i), \end{aligned}$$

que l'on continue tant que  $P_i \neq 0$ . Mais, comme les degrés des  $P_i$  décroissent strictement, on arrive nécessairement à un entier  $n$  tel que  $P_n \neq 0$ ,  $P_{n+1} = 0$ . Notons qu'on suppose  $P_0 \neq 0$  et qu'on peut avoir à l'extrême  $P_1 = 0$ , donc  $n = 0$ .

Ce procédé constitue l'algorithme d'Euclide *stricto sensu*. L'algorithme d'Euclide *étendu* consiste à construire parallèlement deux autres suites  $(A_i)$  et  $(B_i)$ ,  $i$  variant de 0 à  $n + 1$ , de la façon suivante. Partant de

$$A_0 = B_1 = 1, \quad A_1 = B_0 = 0,$$

on écrit

$$A_{i+1} = A_{i-1} - Q_i A_i, \quad B_{i+1} = B_{i-1} - Q_i B_i.$$

On a par exemple  $A_1 = 1$ ,  $B_2 = -Q_1$ ,  $A_3 = -Q_2$ ,  $B_3 = 1 + Q_1 Q_2$ .

**LEMME 1.41.** — Pour  $2 \leq i \leq n + 1$ , on a  $\deg(A_i) = \deg(P_1) - \deg(P_{i-1})$  et  $\deg(B_i) = \deg(P_0) - \deg(P_{i-1})$ .

**EXERCICE 1.29. [B]** — Vérifier ce lemme.

**PROPOSITION 1.42.** — On a  $P_i = A_i P_0 + B_i P_1$  pour  $0 \leq i \leq n + 1$  et  $A_i B_{i+1} - A_{i+1} B_i = (-1)^i$  pour  $0 \leq i \leq n$ .

**EXERCICE 1.30. [B]** — Vérifier cette proposition.

**PROPOSITION 1.43.** — On a

$$P_0 = (-1)^n B_{n+1} P_n, \quad P_1 = (-1)^{n+1} A_{n+1} P_n.$$

*Démonstration.* On a d'une part

$$A_i P_{i+1} - A_{i+1} P_i = A_i (P_{i-1} - P_i Q_i) - (A_{i+1} - Q_i A_i) P_i = -(A_{i-1} P_i - A_i P_{i-1}),$$

donc

$$A_i P_{i+1} - A_{i+1} P_i = (-1)^{i-1} (A_1 P_0 - A_0 P_1) = (-1)^i P_1,$$

et d'autre part par un calcul analogue

$$B_i P_{i+1} - B_{i+1} P_i = (-1)^{i+1} P_0.$$

Puisque  $P_{n+1} = 0$ , on en déduit les résultats annoncés. □

Ainsi, on voit que  $P_n$  divise  $P_0$  et  $P_1$ . Inversement, on a « l'identité de Bézout »

$$P_n = A_n P_0 + B_n P_1,$$

qui implique que tout diviseur commun de  $P_0$  et  $P_1$  divise  $P_n$ . Par conséquent  $P_n$  est un « plus grand commun diviseur » (en abrégé *pgcd*) de  $P_0$  et  $P_1$ . En particulier :

**PROPOSITION I.44.** — *Soient P et Q deux polynômes de K[X]. Les conditions suivantes sont équivalentes :*

- (i) *tout diviseur commun de P et Q est constant ;*
- (ii) *il existe A et B dans K[X] avec AP + BQ = 1.*

On dit alors que P et Q sont *premiers entre eux*.

Comme dans le cas des entiers, on en déduit le résultat classique : si P est premier à Q et divise QR, il divise R. En effet, si  $AP + BQ = 1$ , alors  $R = APR + BQR$  est divisible par P, puisque AP et B(QR) le sont.

On dit qu'un polynôme P est *irréductible* s'il n'est pas constant et si, chaque fois que l'on a  $P = P_1 P_2$ , alors  $P_1$  ou  $P_2$  est constant. Un polynôme non constant qui n'est pas irréductible peut donc par définition se décomposer en produit de deux polynômes de degrés strictement inférieurs, ce qui montre par récurrence que *tout polynôme non constant se décompose en produit de polynômes irréductibles*.

**LEMME I.45.** — *Un polynôme irréductible qui divise un produit divise l'un des facteurs.*

*Démonstration.* Supposons que P soit irréductible, divise QR et ne divise pas Q. Soit F un pgcd de P et Q. Puisque F divise P, il doit être constant, de sorte que P est premier à Q et donc divise R.  $\square$

Comme dans le cas des entiers, on en déduit l'unicité de la décomposition d'un polynôme comme produit de polynômes irréductibles (à l'ordre près et à des facteurs constants près).

## § I.6. Algorithmes de recherche d'une période

### I.6.1. Exemple : écriture décimale d'un nombre rationnel

Naturellement, tous les algorithmes de réécriture ne se terminent pas nécessairement. Un exemple simple et classique est celui du calcul de l'expression décimale d'un nombre rationnel. Soient  $u$  et  $v$  deux entiers avec  $0 < u < v$ , déterminant le nombre rationnel  $r = u/v < 1$ , d'écriture décimale

$$r = 0, q_1 q_2 q_3 \dots$$

avec  $0 \leq q_i \leq 9$ . Imaginons l'algorithme usuel de division décimale. Une fois épuisée la partie initiale qui parcourt les différents chiffres de  $u$ , il se déroule comme suit : on a un reste  $r_i$  avec  $0 \leq r_i < v$ , on « abaisse un zéro », ce qui donne  $10r_i$ , et on divise le résultat par  $v$ , ce qui donne le reste suivant et le  $i$ -ième chiffre du quotient par

$$10r_i = vq_i + r_{i+1},$$

soit  $r_{i+1} \equiv 10r_i \pmod{v}$ . Puisque les  $r_i$  appartiennent à l'ensemble *fini* des entiers compris entre 0 et  $v - 1$ , il ne peuvent être tous distincts, et il existe nécessairement  $n$  et  $N > 0$  avec  $r_{n+N} = r_n$ . Par conséquent, l'algorithme se répète cycliquement à partir de ce moment là. On en déduit qu'après une séquence initiale la suite des chiffres  $(q_i)$  devient périodique.

**EXERCICE 1.31.** [N] — Écrire dans le langage de son choix un programme déterminant *tous* les chiffres décimaux d'une fraction  $u/v$  donnée.

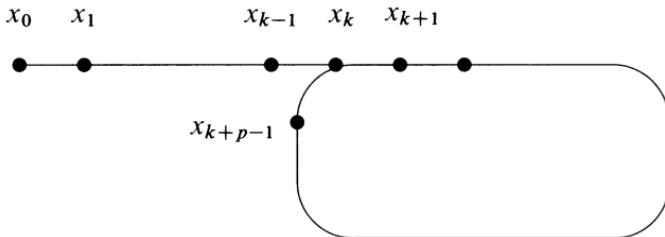
### 1.6.2. Période d'une suite récurrente

Considérons plus généralement la situation suivante. Soient  $E$  un ensemble,  $a$  un élément de  $E$  et  $f : E \rightarrow E$  une application. On considère la suite récurrente  $(x_i)$  d'éléments de  $E$  donnée par  $x_0 = a$  et  $x_{i+1} = f(x_i)$  pour  $i \geq 0$ . Alors, de deux choses l'une : ou bien les  $x_i$  sont tous différents (ce qui ne peut se produire que si  $E$  est infini), ou bien il existe  $k$  et  $l$  avec  $k \neq l$  et  $x_k = x_l$ . On dit alors que la suite est « ultimement périodique ». On a en effet le lemme suivant.

**LEMME 1.46.** — *Supposons la suite ultimement périodique. Soit  $l$  le plus petit indice tel que  $x_l$  soit égal à l'un des  $x_k$  pour  $k < l$ . Posons  $p = l - k$ . Alors les  $x_i$  pour  $0 \leq i < l$  sont tous distincts et forment toutes les valeurs de la suite. Pour tout couple d'indices  $i \neq j$ , la condition  $x_i = x_j$  équivaut à  $k \leq \inf(i, j)$  et  $i \equiv j \pmod{p}$ .*

*Démonstration.* La première assertion est évidente par définition même de  $l$ . Pour  $k \leq i$ , on a  $x_{i+p} = f^{i-k}(x_{k+p}) = f^{i-k}(x_l) = f^{i-k}(x_k) = x_i$ . Inversement, soient  $i < j$  avec  $x_i = x_j$ . On a alors par définition  $l \leq j$ , donc  $k \leq j - p$  et  $x_{j-p} = x_j$  d'après ce qui précède. Si  $j - p = i$ , on a bien  $k \leq i$  et  $i \equiv j \pmod{p}$ . Si  $j - p \neq i$ , on peut remplacer le couple  $(i, j)$  par le couple  $(i, j - p)$  et conclure par récurrence.  $\square$

L'entier  $k$  est appelé *l'indice d'entrée dans la période*. L'entier  $p$  est appelé *la période* de la suite. Les multiples  $> 0$  de  $p$ , qui d'après le lemme sont aussi les entiers de la forme  $j - i$  avec  $i < j$  et  $x_i = x_j$ , sont parfois appelés les périodes de la suite.



## Taille moyenne de la période

Soient  $E$  un ensemble fini à  $n$  éléments. À chaque couple  $(f, a)$  formé d'une application de  $E$  dans lui-même et d'un élément  $a$  de  $E$ , on peut associer la suite récurrente définie par  $x_0 = a$  et  $x_{i+1} = f(x_i)$ . Notons comme dans le lemme ci-dessus par  $k$  et  $l$  les plus petits entiers avec  $k < l$  et  $x_k = x_l$ . Si l'on considère les  $n \cdot n^n$  couples  $(f, a)$  comme équiprobables, les entiers  $k, l$  et  $p = l - k$  ont des valeurs moyennes que l'on peut calculer. On obtient respectivement  $(Q(n) - 1)/2$ ,  $Q(n)$ , et  $(Q(n) + 1)/2$ , où  $Q(n)$  est une fonction pour laquelle Donald KNUTH a donné une évaluation asymptotique dont le terme dominant est  $\sqrt{\pi n/2}$ . On renvoie pour l'analyse complète à [Naudin, Quitté], IV.4.3 et [Knuth 1], 2.II.3.

Il faut en retenir que, pour  $n$  grand, on doit s'attendre à des indices d'entrée et des périodes de l'ordre de  $\frac{5}{4}\sqrt{n}$ .

### I.6.3. Algorithme de Floyd

Le premier pas pour déterminer la période d'une suite (connue pour être ultimement périodique, par exemple parce que l'ensemble des valeurs possibles est fini) consiste à trouver un couple  $i < j$  avec  $x_i = x_j$ , donc une période  $j - i$ . Il suffit ensuite de tester les valeurs des  $x_{i+d}$  pour tous les diviseurs  $d$  de  $j - i$  pour déterminer la période. Pour trouver un tel couple  $i < j$ , la méthode naturelle consiste à calculer les  $x_i$  et à les comparer deux à deux, ce qui n'est guère économique.

L'algorithme inventé par Robert FLOYD est basé sur les remarques suivantes. Notons  $k$  et  $p > 0$  respectivement l'indice d'entrée et la période, de sorte qu'on a  $x_{k+p} = x_k$ . Si  $i$  est à la fois supérieur à  $k$  et multiple de  $p$ , on a  $x_{2i} = x_i$ ; inversement, cette condition implique que  $i$  est  $\geq k$  et multiple de  $p$ . Enfin  $x_{2i}$  s'obtient à partir de  $x_0 = a$  par itération de la fonction  $f \circ f$ . Cela donne l'algorithme suivant :

---

#### ALGORITHME I.47. — *Algorithme de Floyd*

*Entrées* : terme initial  $a$ . *Sorties* : un entier  $i > 0$  tel que  $x_{2i} = x_i$ .

Règles :

$$\begin{array}{lcl} a & \mapsto & (1, f(a), f(f(a))) \\ [x \neq y] : (i, x, y) & \mapsto & (i + 1, f(x), f(f(y))) \\ [x = y] : (i, x, y) & \mapsto & i \end{array}$$


---

EXERCICE 1.32. [A] — Prouver la validité de cet algorithme.

EXERCICE 1.33. [N] — Reprendre l'exercice 1.31 par la méthode de Floyd et comparer les vitesses obtenues.

#### 1.6.4. Algorithme de Brent

L'algorithme précédent a deux défauts. D'abord, si  $k$  est beaucoup plus grand que  $p$ , l'algorithme ne trouvera qu'un multiple élevé de la vraie période. Ensuite, chaque pas demande trois applications de la fonction  $f$ , ce qui peut se révéler coûteux. En définitive, il s'agit de tester l'égalité  $x_i = x_j$  le long d'une suite double d'entiers  $(i, j)$  qui a les vertus suivantes : l'entier  $i$  prend des valeurs aussi grandes que l'on veut et l'entier  $j - i$  prend toutes les valeurs assez grandes. De ce point de vue, la suite « de Floyd »  $\{(i, 2i)\}$  convient, mais n'est pas la meilleure possible. On doit à Richard BRENT une meilleure solution : la suite

$$\begin{aligned} 0 &\mapsto (0, 1) \\ &\mapsto (1, 2) \mapsto (1, 3) \\ &\mapsto (3, 4) \mapsto (3, 5) \mapsto (3, 6) \mapsto (3, 7) \\ &\mapsto (7, 8) \mapsto (7, 9) \mapsto (7, 10) \mapsto (7, 11) \cdots \mapsto (7, 15) \\ &\mapsto (15, 16) \cdots \end{aligned}$$

dont l'algorithme de construction peut être décrit ainsi :

$$\begin{array}{lcl} 0 & \mapsto & (0, 1) \\ [j \leq 2i] : (i, j) & \mapsto & (i, j + 1) \\ [j = 2i + 1] : (i, j) & \mapsto & (j, j + 1). \end{array}$$

On en tire directement un algorithme qui, partant du terme initial  $a$ , donne deux indices  $i$  et  $j$  avec  $x_i = x_j$ , donc une période  $j - i$  :

$$\begin{array}{lcl} a & \mapsto & (0, 1, a, f(a)) \\ [x \neq y \text{ et } j \leq 2i] : (i, j, x, y) & \mapsto & (i, j + 1, x, f(y)) \\ [x \neq y \text{ et } j = 2i + 1] : (i, j, x, y) & \mapsto & (j, j + 1, y, f(y)) \\ [x = y] : (i, j, x, y) & \mapsto & (i, j) \end{array}$$

On peut évidemment remplacer le test  $j = 2i + 1$  par  $j > 2i$ . Il est aussi un peu plus simple de manipuler le couple  $(i, j - i)$  au lieu du couple  $(i, j)$ , ce qui, en posant  $r = j - i$ , donne la forme définitive :

**ALGORITHME I.48. — Algorithme de Brent**

*Entrées* : terme initial  $a$ . *Sorties* : entiers  $i$  et  $r > 0$  avec  $x_{i+r} = x_i$ .

Règles :

$$\begin{array}{lll} a & \mapsto & (0, 1, a, f(a)) \\ [x \neq y \text{ et } r \leq i] : & (i, r, x, y) & \mapsto (i, r+1, x, f(y)) \\ [x \neq y \text{ et } r > i] : & (i, r, x, y) & \mapsto (i+r, 1, y, f(y)) \\ [x = y] : & (i, r, x, y) & \mapsto (i, r) \end{array}$$


---

Cet algorithme présente plusieurs avantages sur celui de Floyd : il donne d'abord la *vraie période*  $r$  et d'autre part, à chaque pas, on n'applique qu'une fois la fonction  $f$  (noter cependant qu'un pas de l'algorithme de Floyd correspond à deux pas de l'algorithme de Brent).

En CAML-Light<sup>11</sup>, l'algorithme de Brent est :

**PROGRAMME I.49. — Algorithme de Brent en CAML-Light**

```
let brent f a =
  brentrec 0 1 a (f(a)) where rec
    brentrec i r x y =
      if (x=y) then (i,r)
      else if (r>i) then brentrec (i+r) 1 y (f(y))
      else brentrec i (r+1) x (f(y));;
```

---

**EXERCICE I.34. [B]** — Démontrer que l'algorithme précédent donne effectivement la vraie période et pas un multiple.

**EXERCICE I.35. [N]** — Reprendre l'exercice I.31 par la méthode de Brent et comparer les vitesses obtenues par les trois méthodes.

**I.6.5. La méthode de factorisation  $\rho$  de Pollard**

On doit à John POLLARD une méthode de factorisation des entiers basée sur les algorithmes précédents.

Supposons donné un entier  $n$  et considérons comme ci-dessus une suite récurrente d'entiers modulo  $n$ , donnée par  $x_0 = a$  et  $x_{i+1} = f(x_i)$ . On a vu qu'on doit s'attendre, si  $a$  et  $f$  ont un comportement moyen, à une période de l'ordre de  $c\sqrt{n}$ , que l'on pourra trouver en utilisant suivant l'une des méthodes ci-dessus le test  $x_i = x_j$ .

Supposons maintenant que  $n$  ne soit pas premier. Il possède alors un diviseur  $p \leq \sqrt{n}$ . La suite formée des  $x_i \bmod p$  a elle aussi une période, estimée d'ordre  $c\sqrt{p} \leq c\sqrt[4]{n}$ . Si  $p$  était connu, on trouverait cette période avec un test  $x_i \equiv x_j \pmod{p}$ . Mais cette condition implique  $\text{pgcd}(x_i - x_j, n) \neq 1$ , qui ne fait pas intervenir explicitement  $p$ . De plus, ce pgcd ne

11. Évidemment, seul un langage fonctionnel permet une telle programmation.

devrait pas être  $n$ , car cela donnerait une période anormalement courte pour la suite de départ. On obtient ainsi un diviseur de  $n$ , différent de 1, et qui a aussi de fortes chances d'être différent de  $n$ .

Pour faire fonctionner cette méthode, il faut trouver une fonction  $f$  qui se comporte de façon pas trop anormale en ce qui concerne les périodes considérées. Or des tests numériques poussés ont montré que tel semble bien être le cas des fonctions quadratiques, et notamment de  $f(x) = x^2 + 1$ .

On obtient ainsi, en cherchant la période « à la Brent », l'algorithme suivant :

---

**ALGORITHME 1.50. — Algorithme  $\rho$  de Pollard**


---

*Entrées* : entier  $n$ . *Sorties* : un diviseur  $\neq 1$  de  $n$ .

Règles :

$n$	$\mapsto$	$(0, 1, 0, 1, n)$
[ $\text{pgcd}(x - y, n) = 1$ et $r \leq i$ ] :	$\mapsto$	$(i, r + 1, x, y^2 + 1 \bmod n, n)$
[ $\text{pgcd}(x - y, n) = 1$ et $r > i$ ] :	$\mapsto$	$(i + r, 1, y, y^2 + 1 \bmod n, n)$
[ $\text{pgcd}(x - y, n) \neq 1$ ] :	$\mapsto$	$\text{pgcd}(x - y, n)$

---

En Ruby :

---

**PROGRAMME 1.51. — Algorithme  $\rho$  de Pollard**


---

```
def rhorec(i, r, x, y, n)
  p = pgcd(x-y, n)
  z = (y*y+1) % n
  return p unless p == 1
  r>i ? rhorec(i+r, 1, y, z, n) : rhorec(i, r+1,x, z, n)
end
def rho(n)
  rhorec(0,1,0,1, n)
end
rho(2**32+1) # => 641 # diviseur d'un nombre de Fermat
rho(2**67-1) # => 193707721 # diviseur d'un nombre de Mersenne
repunit = (10**17-1)/9 # => 1111111111111111
rho(repunit) # => 2071723 # diviseur d'un repunit
```

---

**EXERCICE 1.36. [B] — Écrire la variante plus simple « à la Floyd ».**

De nombreuses optimisations de cet algorithme sont possibles. Il est notamment coûteux de calculer séparément tous les pgcd. On peut par exemple stocker le produit de plusieurs  $(x - y)$  consécutifs, et calculer le pgcd de ce produit avec  $n$  ; si ce pgcd est égal à 1, on poursuit ; dans le cas contraire, on revient en arrière et on recalcule les pgcd individuels. On synchronise évidemment ces phases successives avec celles de la méthode de Brent.

**EXERCICE 1.37. [N] — Implanter la méthode de Pollard et tester diverses améliorations.**



# Chapitre 2

## Théorème de Fermat et primalité

Le *petit théorème de Fermat*<sup>1</sup> a été énoncé par ce dernier en 1640 et démontré par Euler en 1736.

Il s'énonce très simplement : si  $p$  est un nombre premier et  $a$  un entier quelconque, alors  $a^p - a$  est divisible par  $p$ . Nous l'utiliserons de toutes les façons possibles. Il donne notamment un critère de non-primalité : si  $a$  et  $n$  sont deux entiers tels que  $a^n$  n'est pas congru à  $a$  modulo  $n$ , alors  $n$  est composé. C'est par exemple le cas pour  $n = 2^{32} + 1$  et  $a = 3$ , comme nous l'avons vu (1.3.4), ce qui montre que le nombre de Fermat  $2^{32} + 1$  n'est pas premier.

Le théorème de Fermat a de nombreuses variantes lorsque  $p$  n'est pas premier (théorème d'Euler 2.17, théorème de Carmichael que nous verrons en 3.3.2), comme aussi les critères de non-primalité qu'on en déduit. Ce chapitre est consacré à ces questions. En préambule, on traite du *théorème chinois* qui permet de réduire les congruences modulo un entier à celles modulo ses facteurs « primaires » (puissances de nombres premiers).

### § 2.1. Théorème chinois

#### 2.1.1. L'énoncé : forme classique

Le « théorème chinois »<sup>2</sup> apparaît pour la première fois<sup>3</sup> dans un traité appelé *Sun Tzu Suan Ching* ou *Jiuzhang suanshu* (date estimée : entre 280 et 473) :

Nous avons des choses dont nous ne connaissons pas le nombre ; si nous les comptons par paquets de trois, le reste est 2 ; si nous les comptons par paquets de cinq, le reste est 3 ; si nous les comptons par paquets de sept, le reste est 2. Combien y a-t-il de choses ? Réponse : 23.  
[...]

Pour chaque unité comme reste quand vous comptez par paquet de trois, posez 70 ; pour chaque unité comme reste quand vous comptez par paquet de cinq, posez 21 ; pour chaque unité comme reste quand vous

1. Le « grand » théorème conjecturé par Fermat a été démontré par Andrew WILES en novembre 1994.

2. En anglais, on dit « *Chinese remainder theorem* ».

3. Ces renseignements et l'extrait qui suit sont tirés de [Davis, Hersh], pages 177 et 178. Voir aussi <http://math.sfu.ca/histmath/China/3rdCenturyBC/STSC.html>.

comptez par paquet de sept, posez 15. Si c'est 106 ou plus, soustrayez-lui 105.

Traduisons en termes plus récents : les solutions du système des trois congruences

$$x \equiv a \pmod{3}, \quad x \equiv b \pmod{5}, \quad x \equiv c \pmod{7},$$

sont les entiers congrus modulo 105 à  $70a + 21b + 15c$ . On voit bien ce qu'est 105 : c'est le produit  $3 \times 5 \times 7$ ; et ce que sont 70, 21 et 15 : par exemple 70 est congru à 1 modulo 3 et à 0 modulo 5 et 7. Ce qui ne saute pas aux yeux sur cet exemple, c'est que cela fonctionne parce que 3, 5 et 7 sont premiers entre eux deux à deux.

L'énoncé général est le suivant :

**PROPOSITION 2.1.** — *Soient  $m_1, \dots, m_r$  des entiers premiers entre eux deux à deux, et soit  $n$  leur produit. Pour toute suite  $x_1, \dots, x_r$  d'entiers, il existe un entier  $x$ , uniquement déterminé modulo  $n$ , tel que  $x \equiv x_i \pmod{m_i}$  pour  $i = 1, \dots, r$ .*

### 2.1.2. L'énoncé : forme abstraite

Pour tout entier  $m > 0$ , on note  $\mathbf{Z}/m\mathbf{Z}$  l'anneau des classes de congruences modulo  $m$  (1.3.2). Reprenons les notations de la proposition précédente, et introduisons l'application qui, pour tout  $x \in \mathbf{Z}$ , associe à l'élément  $x \pmod{n}$  de  $\mathbf{Z}/n\mathbf{Z}$  la suite des  $x \pmod{m_i}$ , élément de l'anneau-produit  $\prod \mathbf{Z}/m_i \mathbf{Z}$ . On obtient ainsi un homomorphisme de l'anneau  $\mathbf{Z}/n\mathbf{Z}$  dans le produit des  $r$  anneaux  $\mathbf{Z}/m_i \mathbf{Z}$ . Cela étant, le théorème chinois s'exprime simplement en disant que cet homomorphisme est *bijectif*.

### 2.1.3. Les démonstrations du théorème chinois

Les démonstrations du théorème chinois ne manquent pas. L'unicité, c'est-à-dire le fait que l'application introduite ci-dessus est injective, est facile : puisque les  $m_i$  sont premiers entre eux, dire que  $x - y$  est divisible par le produit  $n$  des  $m_i$  équivaut à dire qu'il est divisible par chacun d'entre eux.

Pour l'existence, c'est-à-dire la surjectivité de l'application, on peut raisonner de plusieurs façons. La plus simple consiste à remarquer que les deux anneaux considérés ont le même nombre d'éléments, ce qui suffit compte tenu de l'unicité déjà démontrée. On peut aussi donner une raison plus constructive. Pour chaque  $i$ , les deux entiers  $m_i$  et  $n/m_i$  sont premiers entre eux ; il existe donc (identité de Bézout) des entiers  $a_i$  et  $b_i$  avec  $a_i m_i + b_i n/m_i = 1$  ; posons  $e_i = b_i n/m_i = 1 - a_i m_i$ , de sorte qu'on a

$$e_i \equiv 1 \pmod{m_i}, \quad e_i \equiv 0 \pmod{m_j} \text{ pour } j \neq i.$$

Il suffit alors de poser  $x = \sum_i x_i e_i$ , puisqu'on a  $\sum_i x_i e_i \equiv x_j \pmod{m_j}$  pour tout  $j$ .

Voici une troisième méthode, où on raisonne par récurrence sur le nombre de facteurs. Il est commode pour effectuer cette récurrence de considérer un problème un peu plus général : on se donne des entiers  $a$  et  $b$  avec  $a$  premier à  $n$ , c'est-à-dire premier à chacun des  $m_i$ , et on cherche à résoudre les congruences  $ax + b \equiv x_i \pmod{m_i}$ . Puisque  $a$  est premier à  $m_1$ , on peut déterminer un entier  $c_1$  tel que  $c_1 a \equiv 1 \pmod{m_1}$ ; soit  $y_1$  un entier tel que  $y_1 \equiv c_1(x_1 - b) \pmod{m_1}$ . Alors  $x$  doit s'écrire  $x = y_1 + m_1 x'$  et on a  $ax + b = am_1 x' + ay_1 + b$ . Il s'agit alors de résoudre les congruences  $a'x' + b' \equiv x_i \pmod{m_i}$  pour  $i \geq 2$ , avec  $a' = am_1$  et  $b' = ay_1 + b$ . Mais  $a'$  est premier à  $m_2 \cdots m_r$  et on conclut par récurrence.

**EXERCICE 2.1. [B]** — Soient  $m_1$  et  $m_2$  deux entiers, soit  $d$  leur pgcd et soient  $x_1$  et  $x_2$  deux entiers. Prouver que, pour qu'on puisse résoudre simultanément les deux congruences  $x \equiv x_i \pmod{m_i}$ , il faut et il suffit qu'on ait  $x_1 \equiv x_2 \pmod{d}$  et que, s'il en est ainsi,  $x$  est uniquement déterminé modulo  $m_1 m_2 / d$ .

**EXERCICE 2.2. [C]** — Combien y a-t-il de racines modulo  $n$  de l'équation  $x^2 = 1$ ? Indication :  $n$  doit diviser  $(x+1)(x-1)$ . Quels sont les  $n$  pour lesquels il y a exactement 2 solutions ?

#### 2.1.4. Un algorithme

Chaque démonstration du théorème chinois donne une méthode de résolution pour les congruences simultanées  $x \equiv x_i \pmod{m_i}$ ,  $i = 1, \dots, r$ . La première (« il existe une solution puisque l'application est surjective ») correspond à la méthode de recherche exhaustive (« essayer toutes les possibilités »). La seconde est effective : on détermine chaque  $e_i$  par l'algorithme d'Euclide étendu.

Si la méthode précédente est effective, elle présente un défaut, celui d'introduire les produits  $x_i e_i$  qui peuvent être beaucoup plus grands que  $n$  lui-même. Suivant Knuth, on peut donner un algorithme plus économique en mettant en action la troisième méthode. Cela donne<sup>4</sup> :

- ▷ **Données** : des entiers  $n$ ,  $a$  et  $b$ , avec  $n$  et  $a$  premiers entre eux, une décomposition de  $n$  en produit d'entiers deux à deux premiers entre eux  $m_i > 0$ ,  $i = 1, \dots, r$ , et des entiers  $x_i$ ,  $i = 1, \dots, r$ .
- ▷ **Sortie** : un entier  $x$  avec  $0 \leq x < n$ , et  $ax + b \equiv x_i \pmod{m_i}$ ,  $i = 1, \dots, r$ .
- ▷ Première phase, indépendante de  $b$  et des  $x_i$  : poser  $a_1 = a$  et  $a_i = a_{i-1} m_{i-1}$  pour  $i = 2, \dots, r$ , et déterminer des entiers  $c_i$  avec  $a_i c_i \equiv 1 \pmod{m_i}$  pour chaque  $i$ .

---

4. Il serait par trop artificiel d'écrire cet algorithme sous forme de réécriture.

- ▷ Deuxième phase : poser  $b_1 = b$  et calculer par récurrence  $y_1, b_2, y_2, b_3, \dots, b_r, y_r$ , satisfaisant à  $0 \leq y_i < m_i$ ,  $y_i \equiv c_i(x_i - b_i) \pmod{m_i}$  et  $b_{i+1} = b_i + a_i y_i$ .
- ▷ Troisième phase : poser  $z_{r+1} = 0$  et calculer  $z_r, \dots, z_1$  par  $z_i = y_i + m_i z_{i+1}$ ; enfin, prendre  $x = z_1$ .

En Ruby, on obtient par exemple :

---

PROGRAMME 2.2. — *Théorème chinois*

---

```
def chinois(a, b, liste)
  # liste est la liste des couples (x_i, m_i)
  # on renvoie x tel que ax + b ≡ x_i [mod m_i] pour tout i
  listy = []
  liste.each do |res, mod|
    c = a.invmmod(mod)
    y = (c * (res - b)) % mod
    listy << y
    b = b + a * y
    a = a * mod
  end
  z = 0
  (liste.size-1).downto(0) do |i|
    z = listy[i] + liste[i][1]*z
  end
  z
end

def residus(liste, n)
  liste.collect{|mod| [n % mod, mod]}
end

liste = residus([2, 3, 5, 7, 11], 13*2252+2)
# => [[0, 2], [1, 3], [3, 5], [4, 7], [7, 11]]
chinois(13, 2, liste) # => 2252
```

---

## § 2.2. L'indicateur d'Euler

### 2.2.1. Le groupe $(\mathbf{Z}/n\mathbf{Z})^*$

Dans tout anneau, les éléments inversibles (pour la multiplication) forment un groupe multiplicatif. Cela s'applique notamment à l'anneau  $\mathbf{Z}/n\mathbf{Z}$  des classes modulo  $n$ ,  $n > 0$ . Pour que la classe  $a \pmod{n}$  soit inversible dans l'anneau  $\mathbf{Z}/n\mathbf{Z}$ , il faut et il suffit (Bézout) que  $a$  soit premier à  $n$ . Ces classes forment le groupe multiplicatif  $(\mathbf{Z}/n\mathbf{Z})^*$ . L'inverse d'un élément se calcule par exemple par l'algorithme d'Euclide étendu.

Donnons quelques exemples. Si  $n$  est premier, tout entier non divisible par  $n$  est premier à  $n$ , de sorte que  $\mathbf{Z}/n\mathbf{Z}$  est un corps. Tous les éléments non nuls de  $\mathbf{Z}/n\mathbf{Z}$  sont alors inversibles et le groupe  $(\mathbf{Z}/n\mathbf{Z})^*$  a  $n - 1$  éléments, les classes de  $1, \dots, n - 1$ . Prenons par exemple  $n = 5$ . Alors, le groupe  $(\mathbf{Z}/5\mathbf{Z})^*$  a quatre éléments et on a  $2 \cdot 3 \equiv 1$  et  $4 \cdot 4 \equiv 1$ . Comme on dit (et nous reviendrons là-dessus en détail, voir le corollaire 3.9), ce groupe est

cyclique, car formé des puissances de la classe de 2 : on a  $2^2 \equiv 4, 2^3 \equiv 3$  et  $2^4 \equiv 1$ .

Prenons maintenant  $n = 8$ . Les classes inversibles sont celles de 1, 3, 5 et 7, qui sont chacune leur propre inverse. De plus, puisque  $3 \cdot 5 \cdot 7 \equiv 1$ , le produit de deux de ces trois classes est égal à la troisième.

Nous déterminerons plus loin la structure du groupe  $(\mathbf{Z}/n\mathbf{Z})^*$  dans le cas le plus général. Notons simplement pour le moment qu'il suffit de le faire lorsque  $n$  est une puissance d'un nombre premier. En effet, si on décompose  $n$  en facteurs premiers sous la forme  $n = \prod p_i^{r_i}$ , le groupe  $(\mathbf{Z}/n\mathbf{Z})^*$  s'identifie au produit des groupes  $(\mathbf{Z}/p_i^{r_i}\mathbf{Z})^*$ , comme il résulte par récurrence du lemme suivant :

**LEMME 2.3.** — *Soient  $m$  et  $n$  deux entiers positifs premiers entre eux. Alors le groupe  $(\mathbf{Z}/mn\mathbf{Z})^*$  s'identifie au produit des groupes  $(\mathbf{Z}/m\mathbf{Z})^*$  et  $(\mathbf{Z}/n\mathbf{Z})^*$ .*

*Démonstration.* D'après le théorème chinois, l'anneau  $\mathbf{Z}/mn\mathbf{Z}$  est le produit des anneaux  $\mathbf{Z}/m\mathbf{Z}$  et  $\mathbf{Z}/n\mathbf{Z}$ . Pour  $x \in \mathbf{Z}/m\mathbf{Z}$  et  $y \in \mathbf{Z}/n\mathbf{Z}$ , dire que le couple  $(x, y)$  est inversible dans l'anneau produit, c'est dire que  $x$  et  $y$  sont inversibles. Par conséquent, le groupe  $(\mathbf{Z}/mn\mathbf{Z})^*$  est le produit des groupes  $(\mathbf{Z}/m\mathbf{Z})^*$  et  $(\mathbf{Z}/n\mathbf{Z})^*$ .  $\square$

### 2.2.2. L'indicateur d'Euler

Soit  $n$  un entier  $> 0$ . On note  $\varphi(n)$  le nombre des éléments inversibles de l'anneau  $\mathbf{Z}/n\mathbf{Z}$ , soit

$$\varphi(n) = \#((\mathbf{Z}/n\mathbf{Z})^*),$$

c'est-à-dire encore le nombre des entiers  $a$  premiers à  $n$  et tels que  $0 \leq a < n$ , soit

$$\varphi(n) = \#\{a \mid 0 \leq a < n, a \text{ premier à } n\}.$$

Dans les lignes précédentes et dans tout le livre, on désigne par  $\#E$  le nombre des éléments d'un ensemble fini  $E$ . La fonction  $\varphi$  s'appelle *l'indicateur d'Euler*<sup>5</sup>.

On a

$$\varphi(1) = 1, \varphi(2) = 1, \varphi(3) = 2, \varphi(4) = 2, \dots$$

On a toujours  $\varphi(n) < n$ , sauf pour  $n = 1$ , et la condition  $\varphi(n) = n - 1$  équivaut au fait que  $\mathbf{Z}/n\mathbf{Z}$  est un corps, donc à la primalité de  $n$ .

**EXERCICE 2.3. [A]** — Quels sont les entiers  $n$  pour lesquels on a  $\varphi(n) = 1$  ?

**PROPOSITION 2.4.** — *a) Soit  $p$  un nombre premier. On a  $\varphi(p^s) = p^s - p^{s-1}$  pour tout  $s > 0$ . En particulier  $\varphi(p) = p - 1$  et  $\varphi(2^s) = 2^{s-1}$ .*

---

5. En anglais, « Euler totient function ».

b) L'indicateur d'Euler est multiplicatif<sup>6</sup> : si  $m$  et  $n$  sont premiers entre eux, on a  $\varphi(mn) = \varphi(m)\varphi(n)$ .

*Démonstration.* Puisque  $s$  est  $> 0$ , les entiers qui ne sont pas premiers à  $p^s$  sont les multiples de  $p$ . Il y en a  $p^{s-1}$  qui appartiennent à l'intervalle considéré. Cela prouve a). D'après le lemme ci-dessus, les groupes  $(\mathbf{Z}/mn\mathbf{Z})^*$  et  $(\mathbf{Z}/m\mathbf{Z})^* \times (\mathbf{Z}/n\mathbf{Z})^*$  sont isomorphes. Ils ont donc le même nombre d'éléments, ce qui implique b).  $\square$

D'après a) et b), l'indicateur d'Euler peut se calculer par décomposition en facteurs premiers. On a, pour des nombres premiers distincts  $p_1, \dots, p_r$  et pour des entiers  $s_1, \dots, s_r$ , tous  $> 0$

$$\varphi\left(\prod_{i=1}^r p_i^{s_i}\right) = \prod_{i=1}^r (p_i^{s_i} - p_i^{s_i-1}).$$

En d'autres termes :

$$\varphi(n) = n \prod_p \left(1 - \frac{1}{p}\right),$$

où le produit est étendu aux diviseurs premiers  $p$  de  $n$ .

On peut ainsi calculer brutallement les premières valeurs de  $\varphi$  en suivant le crible d'Eratosthène :

PROGRAMME 2.5. — *Calcul de l'indicateur d'Euler* ——————

```
def phi(max)
# table des valeurs de phi entre 0 et max
phi = Array.new
phi[0], phi[1], p = 1, 1, 2
until p.nil? do
  done = Array.new
  phi.each_with_index do |val, n|
    next if n == 0 || val.nil? || done[n]
    pp = p
    until (nn = n * pp) > max
      done[nn] = true
      phi[nn] = val * (pp - pp/p)
      pp = pp * p
    end
  end
  p = phi.index(nil) # nombre premier suivant
end
phi[0] = nil
phi
phi(1000) # => [nil, 1, 1, 2, 2, 4, 2, 6, 4, 6, 4, 10, ...]
phi(561).last # => 320
```

6. Une fonction  $f$  définie sur les entiers  $> 0$  est traditionnellement dite *multiplicative* si l'on a  $f(mn) = f(m)f(n)$  chaque fois que  $m$  et  $n$  sont premiers entre eux. Une telle fonction est donc déterminée par les valeurs  $f(p^r)$  avec  $p$  premier et  $r > 0$ .

Naturellement, cette méthode est désespérée pour  $n$  grand. *On ne connaît pas la complexité du calcul de  $\varphi(n)$ .* Elle est sans doute élevée puisque liée à celle de la décomposition en facteurs premiers. Si par exemple  $n$  est le produit de deux nombres premiers distincts  $p$  et  $q$ , on a  $\varphi(n) = (p - 1)(q - 1) = n - (p + q) + 1$ , donc  $p + q = n - 1 - \varphi(n)$ . Connaître simultanément  $n$  et  $\varphi(n)$  équivaut donc à connaître la somme et le produit des deux nombres  $p$  et  $q$ , c'est-à-dire en définitive la décomposition de  $n$ . On verra ci-dessous (2.3.5) que la base même de la méthode de cryptographie RSA est la difficulté (constatée, mais non prouvée pour le moment) de décomposer un tel entier  $n$ , convenablement choisi.

**EXERCICE 2.4. [A]** — Pour tout  $n > 2$ , l'entier  $\varphi(n)$  est pair. Quels sont les  $n$  pour lesquels  $\varphi(n) = 2$  ?

**EXERCICE 2.5. [B]** — Que vaut  $\varphi(2n)$  ?

**EXERCICE 2.6. [B]** — Montrer que l'ensemble des solutions de  $\varphi(n) = a$  est fini pour tout entier  $a$ .

**EXERCICE 2.7. [C]** — Montrer qu'il n'existe pas de nombre réel  $\alpha > 0$  tel que  $\varphi(n) > \alpha n$  pour tout  $n$  ; en d'autres termes, on a  $\liminf \frac{\varphi(n)}{n} = 0$ . On utilisera le fait que le produit infini  $\sum(1 - \frac{1}{p})^{-1}$  étendu à tous les nombres premiers  $p$  est égal à la somme  $\sum \frac{1}{n}$ , donc est infini.

**EXERCICE 2.8. [A]** — Montrer qu'on a  $\limsup \frac{\varphi(n)}{n} = 1$ .

**REMARQUE 2.6.** — On peut prouver<sup>7</sup> que, *en moyenne*,  $\varphi(n)$  est égal à  $\frac{6}{\pi^2}n$  et qu'il existe  $\alpha > 0$  avec  $\varphi(n) \geq \alpha n / \log \log n$ .

Signalons aussi :

**PROPOSITION 2.7.** *Pour tout entier  $n > 0$ , on a*

$$n = \sum_d \varphi(d),$$

où  $d$  parcourt les diviseurs de  $n$ .

*Démonstration.* Répartissons les entiers entre 0 et  $n - 1$  suivant la valeur, notée  $n/d$ , de leur pgcd avec  $n$  : chacun de ces entiers peut s'écrire de manière unique sous la forme  $yn/d$ , où  $d$  divise  $n$  et où  $y$  est premier à  $d$  et compris entre 0 et  $d - 1$ . Ainsi, le nombre total  $n$  de ces entiers est la somme des  $\varphi(d)$  pour  $d$  divisant  $n$ . □

**EXERCICE 2.9. [B]** — Si la fonction  $f$  est multiplicative, alors la fonction  $g$  définie par  $g(n) = \sum_d f(d)$  où  $d$  parcourt les diviseurs de  $n$ , l'est aussi.

**EXERCICE 2.10. [B]** — Déduire de l'exercice 2.9 une nouvelle démonstration de la proposition 2.7.

---

7. Voir par exemple [Hardy, Wright].

### § 2.3. Le petit théorème de Fermat

#### 2.3.1. Ordre d'un élément d'un groupe

Soit  $G$  un groupe, noté multiplicativement, d'élément neutre noté  $e$ , et soit  $g$  un élément de  $G$ . L'ensemble formé des puissances  $g^n$  de  $g$ , où  $n$  parcourt  $\mathbf{Z}$ , est un sous-groupe de  $G$ . Comme c'est évidemment le plus petit sous-groupe de  $G$  contenant  $g$ , on l'appelle le *sous-groupe engendré* par  $g$ . Notons-le ici  $H$ .

Distinguons alors deux cas. Ou bien les  $g^n$  sont tous distincts, et alors  $H$  est infini et on dit que  $g$  est *d'ordre infini*. Ou bien il existe  $n$  et  $n'$  distincts avec  $g^n = g^{n'}$ , donc  $g^{n-n'} = e$ . Soit alors  $m$  le plus petit entier  $> 0$  tel que  $g^m = e$ . Par division euclidienne, tout entier  $n$  s'écrit  $mq + r$ , de sorte que  $g^n = (g^m)^q g^r = g^r$ . Le groupe  $H$  est donc formé des  $m$  éléments  $g^i$  pour  $i = 0, \dots, m-1$ , et ceux-ci sont distincts (car si  $i$  et  $j$  sont tels que  $0 \leq i < j \leq m-1$  et  $g^i = g^j$ , on a  $g^{j-i} = e$ , et cela contredit la définition de  $m$ ). La condition  $g^n = g^{n'}$  équivaut donc à  $n \equiv n' \pmod{m}$ . On dit alors que  $g$  est *d'ordre*  $m$ .

**EXERCICE 2.11. [A]** — Dire que  $g$  est d'ordre  $m$  signifie que  $g^m = e$  et que  $g^{m/q} \neq e$  pour tout diviseur premier  $q$  de  $m$ .

**EXERCICE 2.12. [B]** — Si  $g$  est d'ordre  $m$ , et si  $k$  est premier à  $m$ ,  $g^k$  est aussi d'ordre  $m$ . S'il existe un élément d'ordre  $m$ , il y en a donc au moins  $\varphi(m)$ .

**EXERCICE 2.13. [B]** — Supposons  $g$  d'ordre  $m$  et soit  $n$  un entier  $> 0$ . Alors  $g^n$  est d'ordre  $m/d$  où  $d = \text{pgcd}(m, n)$ .

Traditionnellement, le nombre  $\#G$  des éléments d'un groupe fini  $G$  s'appelle l'*ordre* de  $G$ . Tout élément d'un groupe fini est d'ordre fini. Plus précisément :

**THÉORÈME 2.8 (Lagrange).** — *L'ordre de tout élément d'un groupe fini  $G$  divise l'ordre de  $G$  : pour tout  $g \in G$ , on a  $g^{\#G} = e$ .*

*Démonstration.* Donnons une démonstration tout à fait élémentaire dans le cas, qui nous suffira, où le groupe est commutatif. Notons  $n$  l'ordre de  $G$ . L'application  $x \mapsto gx$  est une bijection de  $G$  sur lui-même. Le produit des  $gx$  pour  $x$  parcourant  $G$  (qui ne dépend pas de l'ordre choisi, puisque  $G$  est supposé commutatif) est donc égal au produit des  $x$ . Regroupant les facteurs  $g$  (toujours la commutativité) et simplifiant, on obtient  $g^n = e$ .  $\square$

**REMARQUE 2.9.** — On attribue cet énoncé à Lagrange car il l'a prouvé en 1770 dans le cas particulier où  $G$  est le groupe des permutations : l'ordre de toute permutation de  $n$  lettres est un diviseur de  $n!$ .

**EXERCICE 2.14. [B]** — Le théorème de Lagrange est un cas particulier de l'énoncé suivant : si  $H$  est un sous-groupe du groupe fini  $G$ , alors  $\#H$  divise

#G. Démontrer cet énoncé (introduire pour chaque élément  $g$  de  $G$  la partie  $gH$  formée des  $gh$  où  $h$  parcourt  $H$ , montrer que chacune de ces parties a  $\#H$  éléments et que deux parties de cette forme sont disjointes ou identiques).

Une grande partie de ce qui va suivre est l'étude de l'ordre d'une classe de congruence : deux entiers premiers entre eux  $a$  et  $n$  étant donnés, on considère l'ordre de  $a$  modulo  $n$ , c'est-à-dire l'ordre de la classe  $a \bmod n$  dans le groupe  $(\mathbf{Z}/n\mathbf{Z})^*$ , soit explicitement le plus petit entier  $m > 0$  tel que  $a^m \equiv 1 \pmod{n}$ . Le premier renseignement est déduit du théorème de Lagrange : cet ordre divise l'ordre  $\varphi(n)$  du groupe.

Lorsque l'ordre de  $a$  est assez petit, on peut le déterminer par énumération brutale des puissances de  $a$  :

```
class Integer
  def ordre_mod(n)
    raise ArgumentError unless pgcd(self,n) == 1
    i = 1
    b = self
    until b == 1 do
      b = (self * b) % n
      i += 1
    end
    i
  end
end
[2,3,5,7].collect {|a| a.ordre_mod(71)} # => [35, 35, 5, 70]
```

Explicitons le cas de l'exemple donné, celui des entiers modulo 71. Le groupe  $(\mathbf{Z}/71\mathbf{Z})^*$  est d'ordre 70. Les classes de 2 et de 3 sont toutes deux d'ordre 35. Cela peut se vérifier par le calcul brutal ci-dessus. On peut aussi remarquer que  $3^7 = 2187 \equiv 128 = 2^7$  et  $3^5 = 243 \equiv 1024 = 2^{10}$ , ce qui donne  $2^{35} \equiv 3^{35} \equiv 2^{70}$ , donc  $2^{35} \equiv 1$  et  $3^{35} \equiv 1$ ; comme d'après ce qui précède, les classes de 2 et 3 ne sont ni d'ordre 5, ni d'ordre 7, elles doivent bien être d'ordre 35. La classe de 4 =  $2^2$  est aussi d'ordre 35. Quant à la classe de 5, elle est d'ordre 5, puisque  $5^5 - 1 = 3124 = 44 \times 71$ .

### 2.3.2. Le petit théorème de Fermat

On s'intéresse d'abord au cas des modules premiers.

**PROPOSITION 2.10** (« Petit » théorème de Fermat). — *Soit  $p$  un nombre premier. On a  $a^p \equiv a \pmod{p}$  pour tout entier  $a$ , et  $a^{p-1} \equiv 1 \pmod{p}$  pour tout entier  $a$  premier à  $p$ .*

Pour la démonstration, il est commode de généraliser un peu la situation. Rappelons qu'on appelle *corps* un anneau dans lequel tout élément non nul est inversible. C'est le cas notamment du quotient  $\mathbf{Z}/p\mathbf{Z}$  où  $p$  est un nombre premier.

**PROPOSITION 2.11.** — Soit  $K$  un corps fini<sup>8</sup>, à  $q$  éléments.

- a) On a  $\alpha^{q-1} = 1$  pour tout  $\alpha \in K^*$ .
- b) On a dans  $K[X]$  l'identité

$$\prod_{\alpha \in K} (X - \alpha) = X^q - X.$$

*Démonstration.* L'assertion a) résulte directement du théorème de Lagrange. On a donc  $\alpha^q = \alpha$  pour tout  $\alpha$  non nul et donc aussi pour tout  $\alpha$ . Le polynôme  $X^q - X$ , unitaire et de degré  $q$ , possède ainsi comme racines les  $q$  éléments  $\alpha$  de  $K$ . C'est donc le produit des  $X - \alpha$  (voir par exemple la proposition 3.4), d'où b).  $\square$

**REMARQUE 2.12.** — Si on applique le théorème de Lagrange au groupe additif  $K$ , on obtient la relation  $q.1_K = 0$ . En fait, on verra plus loin (théorème 9.5) que  $q$  est une puissance d'un nombre premier  $p$  avec  $p.1_K = 0$ .

**EXERCICE 2.15.** [B] — En calquant la démonstration du théorème 2.8, donner une démonstration directe de a).

**COROLLAIRE 2.13.** — Soit  $p$  un nombre premier.

a) On a dans l'anneau  $(\mathbf{Z}/p\mathbf{Z})[X]$  des polynômes à coefficients modulo  $p$  l'identité

$$\prod_{i=0}^{p-1} (X - i) = X^p - X.$$

b) On a dans  $(\mathbf{Z}/p\mathbf{Z})[X]$  l'identité  $(X + 1)^p = X^p + 1$ .

c) Pour tout entier  $i$  avec  $0 < i < p$ , le coefficient binomial  $\binom{p}{i}$  est divisible par  $p$ .

*Démonstration.* La partie a) résulte de la proposition précédente appliquée au corps  $K = \mathbf{Z}/p\mathbf{Z}$ . Par ailleurs, l'application  $\alpha \mapsto \alpha - 1$  est une permutation de  $K$ . On a donc dans  $K[X]$

$$X^p - X = \prod_{\alpha \in K} (X - \alpha) = \prod_{\alpha \in K} (X - (\alpha - 1)) = \prod_{\alpha \in K} ((X + 1) - \alpha) = (X + 1)^p - (X + 1).$$

Comparant les termes extrêmes, on obtient b). Développant b), on obtient c).  $\square$

**REMARQUE 2.14.** — L'identité a) est due à Lagrange ; il l'a publiée en 1771 et en a déduit le « théorème de Wilson » (voir l'exercice 2.16). D'ailleurs, ce dernier résultat, attribué par Waring en 1770 au juriste Wilson (1741-1793), avait été publié à la fin du dixième siècle par Ibn al-Haytham, aussi appelé Alhazen par ses traducteurs latins.

**EXERCICE 2.16.** [A] — Déduire de a) la relation  $(p - 1)! \equiv -1 \pmod{p}$  qui caractérise les nombres premiers (« théorème de Wilson »).

---

8. En vertu de nos conventions, tous les corps sont supposés commutatifs. Mais on peut prouver qu'un corps fini est nécessairement commutatif (théorème de Wedderburn) ; voir par exemple l'exercice 9.14.

**EXERCICE 2.17. [B]** — Prouver directement que dans tout corps fini le produit des éléments non nuls est égal à  $-1$ .

**EXERCICE 2.18. [B]** — Montrer que si  $n > 4$  n'est pas premier, on a  $(n - 1)! \equiv 0 \pmod{n}$ .

**EXERCICE 2.19. [N]** — Décomposer  $n! + 1$  en facteurs premiers pour  $n$  petit.

**EXERCICE 2.20. [B]** — Démontrer  $c)$  directement.

**EXERCICE 2.21. [B]** — En associant deux à deux les termes dans le théorème de Wilson (voir l'exercice 2.16 ci-dessus), construire un élément  $a$  de  $\mathbf{Z}/p\mathbf{Z}$  tel que  $(-1)^{(p-1)/2}a^2 = -1$ . En déduire que si on a  $p \equiv 1 \pmod{4}$ ,  $-1$  est un carré modulo  $p$ .

**COROLLAIRE 2.15.** — Soient  $A$  un anneau<sup>9</sup>,  $p$  un nombre premier, et  $x$  et  $y$  deux éléments de  $A$  tels que  $p \cdot xy = 0$ . Alors  $(x + y)^p = x^p + y^p$ .

*Démonstration.* En effet, tous les termes intermédiaires de la formule du binôme sont nuls, en vertu de la partie  $c)$  du corollaire précédent.  $\square$

### 2.3.3. Une application du théorème de Fermat à la factorisation

Il s'agit d'un théorème de Legendre, simplifiant la recherche des facteurs premiers des entiers de la forme  $a^n \pm b^n$ . Pour tout diviseur  $m$  de  $n$ , l'entier  $a^m - b^m$  divise  $a^n - b^n$ . De même  $a^m + b^m$  divise  $a^n + b^n$  lorsqu'en outre  $n/m$  est impair. Cette remarque permet déjà de mettre à part certains facteurs premiers de  $a^n \pm b^n$ , ceux qui divisent l'un des  $a^m \pm b^m$  pour  $m$  divisant  $n$  et distinct de  $n$ , et que l'on peut avoir déterminés au préalable. Le théorème de Legendre donne des renseignements sur les autres :

**PROPOSITION 2.16 (Legendre).** — Soient  $a$  et  $b$  deux entiers  $> 0$  premiers entre eux, et soit  $n$  un entier  $> 1$ .

a) Soit  $p$  un facteur premier de  $a^n + b^n$ , qui ne divise aucun des  $a^m + b^m$  pour  $m$  divisant  $n$  et distinct de  $n$ . Alors  $p \equiv 1 \pmod{2n}$ .

b) Soit  $p$  un facteur premier de  $a^n - b^n$  qui ne divise aucun des  $a^m - b^m$  pour  $m$  divisant  $n$  et distinct de  $n$ . Alors  $p \equiv 1 \pmod{n}$ . Si  $n$  est impair et  $> 1$ , on a même  $p \equiv 1 \pmod{2n}$ .

*Démonstration.* Traitons d'abord a). Plaçons-nous modulo  $p$ , et posons  $\alpha = a \pmod{p}$  et  $\beta = b \pmod{p}$ . On a  $\alpha^n = -\beta^n$ . Puisque  $a$  et  $b$  sont premiers entre eux, les classes  $\alpha$  et  $\beta$  ne sont pas simultanément nulles, donc aucune n'est nulle. Posant  $x = \alpha/\beta$ , on a  $x^n = -1$ , donc  $x^{2n} = 1$ . Notons  $r$  l'ordre de  $x$  dans le groupe  $(\mathbf{Z}/p\mathbf{Z})^*$ . Alors  $r$  divise  $2n$ . Si  $r$  divisait  $n$ , on aurait  $x^n = 1$ , donc  $1 \equiv -1 \pmod{p}$  et  $p = 2$ . Mais  $p = 2$  est impossible (si  $a^n + b^n$  est pair, alors  $a$  et  $b$  sont de même parité, donc  $a + b$  est pair). Par conséquent  $r = 2m$ , avec  $m$  divisant  $n$ . On a  $x^{2m} = 1$  et  $x^m \neq 1$ , donc

9. Commutatif ; dans le cas général, on doit supposer que  $xy = yx$ .

$x^m = -1$  (dans un corps l'équation  $X^2 = 1$  a au plus deux solutions), ce qui signifie que  $p$  divise  $a^m + b^m$ . Vu l'hypothèse faite sur  $p$ , cela implique que  $m = n$ . L'ordre de  $x$  est donc égal à  $2n$ . Puisque  $x^{p-1} = 1$  d'après le petit théorème de Fermat,  $2n$  divise  $p - 1$ .

Dans le cas  $b$ ), le raisonnement est analogue mais plus simple. On a  $x^n = 1$ , donc l'ordre  $m$  de  $x$  divise  $n$ . Mais  $m$  ne peut être  $< n$ , vu l'hypothèse sur  $p$ . Donc  $m = n$  et  $n$  divise  $p - 1$ . Si  $n$  est  $> 1$ , alors on a  $p \neq 2$  et  $p - 1$  est pair. Si de plus  $n$  est impair,  $p - 1$  est donc divisible par  $2n$ .  $\square$

EXERCICE 2.22. [B] — Décomposer en facteurs premiers  $(10^3 - 1)/(10 - 1)$ .

Comparer avec ce que dit la proposition précédente. Conclusion ?

### 2.3.4. Le théorème d'Euler

Le théorème de Fermat se généralise directement à un entier non nécessairement premier :

PROPOSITION 2.17 (Euler, 1760). — Soient  $n$  un entier  $> 0$  et  $a$  un entier premier à  $n$ . Alors  $a^{\varphi(n)} \equiv 1 \pmod{n}$ .

Démonstration. Il suffit d'appliquer le théorème de Lagrange à l'élément  $a \pmod{n}$  du groupe  $(\mathbf{Z}/n\mathbf{Z})^*$  dont l'ordre est par définition  $\varphi(n)$ .  $\square$

EXERCICE 2.23. [A] — Déduire du théorème de Lagrange que, pour tous entiers  $a \geq 1$  et  $n \geq 1$ ,  $n$  divise  $\varphi(a^n - 1)$ .

EXERCICE 2.24. [B] — Soient  $a$ ,  $m$  et  $n$  des entiers positifs. On suppose que  $m$  est premier et que  $n$  divise  $a^m - 1$ . Prouver que, soit  $n$  divise  $a - 1$ , soit  $m$  divise  $\varphi(n)$  (considérer l'ordre de  $a$  modulo  $n$ ). Application : tout facteur premier du nombre de Mersenne  $2^m - 1$ , où  $m > 2$  est premier, est congru à 1 modulo  $2m$ .

EXERCICE 2.25. [C-N] — Pour inverser modulo  $n$  un élément  $a$ , on dispose donc de deux méthodes. On peut d'une part calculer par l'algorithme d'Euclide étendu des entiers  $x$  et  $y$  avec  $ax + ny = 1$  et prendre comme inverse la classe de  $x$ . On peut aussi prendre pour inverse la classe de  $a^{\varphi(n)-1}$  (on suppose que  $n$  est fixé et que  $\varphi(n)$  a été déterminé au préalable). Comparer théoriquement et pratiquement les deux méthodes.

On peut renforcer le théorème d'Euler dans le cas où l'entier  $n$  est sans facteur multiple<sup>10</sup> :

LEMME 2.18. — Soit  $n$  un entier  $> 1$  sans facteur multiple. Soit  $m$  un entier tel que, pour tout diviseur premier  $p$  de  $n$ ,  $p - 1$  divise  $m - 1$ . On a alors  $a^{m-1} \equiv 1 \pmod{n}$  pour tout entier  $a$  premier à  $n$  et  $a^m \equiv a \pmod{n}$  pour tout entier  $a$ .

10. On dit qu'un entier est sans facteur multiple s'il n'est divisible par le carré d'aucun entier  $> 1$ , c'est-à-dire s'il est produit de nombres premiers distincts. Le terme anglais est « square-free » ; on trouve parfois en français le mot allemand « quadratfrei ».

*Démonstration.* Soit  $a$  un entier. Soit  $p$  un diviseur premier de  $n$ . Alors, d'après le petit théorème de Fermat,  $a^{p-1}$  est congru à 0 ou 1 modulo  $p$ , suivant que  $a$  est ou non divisible par  $p$ , et il en est de même de  $a^{m-1}$ . Il en résulte que  $a^m$  est congru à  $a$  modulo  $p$ . Puisque  $a^m - a$  est divisible par tous les facteurs premiers de  $n$ , il est divisible par  $n$  qui est leur produit. Ainsi on a  $a^m \equiv a \pmod{n}$ . Si  $a$  est premier à  $n$ , on peut diviser cette congruence par  $a$ .  $\square$

**PROPOSITION 2.19.** — *Soit  $n$  un entier  $> 1$  sans facteur multiple et  $r$  un multiple de  $\varphi(n)$ . On a  $a^r \equiv 1 \pmod{n}$  pour tout entier  $a$  premier à  $n$  et  $a^{r+1} \equiv a \pmod{n}$  pour tout entier  $a$ .*

*Démonstration.* En effet, d'après la proposition 2.4,  $\varphi(n)$  est le produit des  $p - 1$  où  $p$  parcourt les diviseurs premiers de  $n$ . On peut donc appliquer le lemme précédent à  $m = r + 1$ .  $\square$

**EXERCICE 2.26. [A]** — Donner un exemple de deux entiers  $a$  et  $n$  pour lesquels on n'a pas  $a^{\varphi(n)+1} \equiv a \pmod{n}$ .

### 2.3.5. Cryptographie à clés publiques et nombres premiers : la méthode RSA

Le principe de la cryptographie à clés publiques<sup>11</sup> est de façon simplifiée le suivant. On fixe un ensemble  $M$  (« messages ») et chaque interlocuteur choisit deux applications réciproques l'une de l'autre, soit

$$c_i : M \rightarrow M, \quad d_i : M \rightarrow M.$$

L'application  $c_i$  est l'application de *chiffrement* de l'interlocuteur  $i$ , l'application  $d_i$  son application de déchiffrement. Les  $c_i$  sont connues de tous, l'application  $d_i$  n'est connue que de son propriétaire, l'interlocuteur  $i$ . Pour envoyer le message  $m \in M$  à l'interlocuteur  $j$ , l'interlocuteur  $i$  le « chiffre » en lui appliquant la fonction  $c_j \circ d_i$  (il connaît « sa » fonction  $d_i$  et toutes les  $c_j$ ), et envoie donc  $m' = c_j(d_i(m))$ . À la réception, l'interlocuteur  $j$  applique au message chiffré  $m'$  la fonction  $c_i \circ d_j$  (il connaît sa propre fonction  $d_j$ , ainsi que toutes les  $c_i$ ). Ainsi, il récupère le message initial et vérifie au passage qu'il provient bien de l'interlocuteur  $i$  ; on a donc à la fois un mécanisme de chiffrement et un mécanisme d'authentification.

Pour qu'un tel mécanisme fonctionne dans la pratique, il faut que les opérations de chiffrement et de déchiffrement soient calculables par des algorithmes rapides, mais qu'en revanche le « décryptage » ne le soit pas. Toute l'astuce consiste donc à inventer des applications  $c : M \rightarrow M$  telles

11. Publié par Whitfield DIFFIE et Martin HELLMAN in 1976. On a appris bien plus tard que James ELLIS, qui travaillait au service britannique du chiffre (GCHQ, Government Communications Headquarters), avait eu cette idée peu avant. En 1973, C.C. Cocks décrivit (pour le même service du chiffre) ce qu'on a appelé depuis l'algorithme RSA. Ces informations ont été rendues publiques par le GCHQ en 1997.

qu'en connaissant  $c$ , et en disposant même d'un algorithme rapide de calcul de  $c(m)$  à partir de  $m$ , reconstituer inversement  $m$  à partir de  $c(m)$  demande un temps de calcul prohibitif. De telles situations existent : on peut prendre comme exemples les annuaires téléphoniques qui fournissent des applications nom  $\mapsto$  numéro à calcul rapide (trouver un nom par recherche dichotomique dans une liste alphabétique) sans fournir pour autant d'algorithme rapide pour l'application réciproque.

Le point de départ de la méthode RSA, publiée<sup>12</sup> par Ron RIVEST, Adi SHAMIR et Leonard ADLEMAN en 1977, est qu'un tel mécanisme existe dans le calcul des congruences. Sans entrer dans des détails techniques, l'idée de base est la suivante. Les messages sont représentés par des entiers. Une méthode publique de chiffrement  $c$  est déterminée par un grand entier  $n$  sans facteurs carrés et un entier  $r$  premier à  $\varphi(n)$ , tous deux publics. L'application de chiffrement est

$$c(m) = m^r \bmod n.$$

L'application (privée) de déchiffrement correspondante est

$$d(m') = m'^s \bmod n.$$

où  $s$  est choisi de façon que  $m^{rs} \equiv m \pmod{n}$  pour tout entier  $m$ . D'après le théorème d'Euler (proposition 2.19), on choisit donc  $s$  de sorte que  $rs \equiv 1 \pmod{\varphi(n)}$ . La fonction  $c$  se calcule rapidement et il en est de même de  $d$  lorsqu'on connaît l'entier  $s$  (calcul rapide des puissances).

Mais cet  $s$  n'est connu que de son propriétaire et la connaissance de  $s$  demande pratiquement celle de  $\varphi(n)$ . Or, c'est là qu'est la clé : pour calculer  $\varphi(n)$ , il faut essentiellement décomposer  $n$  en facteurs premiers. Et, pour autant qu'on le sache, il n'existe pas de méthode rapide pour le faire. Évidemment, il faudra bien que le propriétaire de  $s$  connaisse  $\varphi(n)$ . Mais justement voilà l'astuce : le propriétaire peut se donner  $n$  à partir de sa décomposition, mais ne publier que le produit. Par exemple, c'est le plus simple et le plus efficace, il choisit deux grands nombres premiers  $p$  et  $q$ , publie le produit  $n = pq$ , choisit  $r$  premier à  $\varphi(n) = (p-1)(q-1)$ , le publie, et résout par l'algorithme d'Euclide l'équation  $rs \equiv 1 \pmod{(p-1)(q-1)}$ .

**EXERCICE 2.27. [B]** — On choisit  $p$  et  $q$  congrus à 2 modulo 3. On peut alors prendre  $r = 3$ . Comment choisir  $s$  ?

En définitive, le chiffrement consiste essentiellement à coder un entier  $m$  par  $m' = m^r \bmod n$ . Pour quelqu'un ignorant  $s$ , le *décryptage* consiste à retrouver  $m$  (modulo  $n$ ) à partir de  $m'$  et  $r$ . Et tout se résume à la remarque surprenante que l'on ne connaît pas de méthode pour ce faire, tout au moins si on a bien choisi  $r$ .

---

12. Voir la note précédente.

Revenons au choix initial de  $p$  et  $q$ , car il semble poser a priori de grandes difficultés : comment diable trouver de grands nombres premiers s'il est si difficile de décomposer les grands nombres en facteurs premiers ? En fait, ce sont des problèmes différents. L'un consiste à savoir si un nombre donné est premier ou pas (test de primalité), l'autre à décomposer un nombre non premier en facteurs. Or, de fait, la seconde question est nettement plus difficile ; savoir qu'un nombre n'est pas premier n'aide en rien à le décomposer. Il existe, comme nous le verrons, des tests rapides de primalité, mais pour le moment on ne connaît pas de méthodes assez rapides de décomposition.

Mais le plus curieux est encore à venir. Il existe en effet, comme nous le verrons plus tard (voir 5.3.4), des tests probabilistes qui permettent de garantir économiquement que des nombres sont presque sûrement premiers. Et on peut remarquer que la méthode précédente peut fonctionner même lorsque  $p$  et  $q$  sont seulement « pratiquement premiers ». Supposons en effet que l'on parte de deux grands entiers  $p$  et  $q$ , qui ont résisté aux tests de non-primalité, et ont reçu un certificat de « primalité pratique ». Fabriquons comme ci-dessus deux entiers  $r$  et  $s$  avec  $rs \equiv 1 \pmod{(p-1)(q-1)}$ , et publions  $r$  et le produit  $n = pq$ .

Pour envoyer le message  $m$ , nous employons la procédure usuelle : nous le codons en  $m' = c'(d(m))$ , où  $c'$  est la fonction de chiffrement publique du destinataire. Mais, avant d'envoyer cet  $m'$ , nous vérifions si on a bien  $c(d(m)) = m \pmod{n}$ . Si oui, envoyons  $m'$  à notre correspondant, nous sommes assurés que son déchiffrement fonctionnera. Si non, alors nous avons trouvé un contre-exemple à la formule d'Euler, donc  $\varphi(n)$  n'est pas égal à  $(p-1)(q-1)$ , et il n'est pas vrai que  $p$  et  $q$  soient tous les deux premiers. Cela est extrêmement peu probable, si  $p$  et  $q$  sont de bonne qualité. Mais si cela arrive, rapportons  $p$  et  $q$  à notre fournisseur de nombres premiers, remplaçons-les par deux autres et recommençons.

En résumé, la viabilité de la méthode RSA repose sur une course de vitesse entre tests de « primalité pratique » et algorithmes de décomposition en facteurs. Chaque invention d'un algorithme de décomposition nouveau oblige à réexaminer RSA pour voir comment protéger les nombres effectivement utilisés des attaques de ce nouvel algorithme<sup>13</sup>. Renvoyons à la littérature spécialisée pour des précisions. Dans la suite, nous expliquerons quelques tests de primalité et de non-primalité. Les algorithmes de décomposition les plus performants font appel, eux, à des outils différents et souvent nettement plus difficiles sur le plan théorique.

Pour fixer les idées, signalons que le « record » atteint dans le cadre du défi lancé en 1991 par la société RSA<sup>14</sup>, est la décomposition de « RSA-200 »,

13. On recommande actuellement une longueur de clé de 2048 bits.

14. Voir <http://www.rsa.com/rsalabs/node.asp?id=2879>. La société RSA a mis un terme à ce défi à la fin 2007 (voir <http://www.rsa.com/rsalabs/node.asp?id=2092>).

un nombre de 200 chiffres décimaux (663 bits), factorisé en mai 2005 par une équipe de chercheurs allemands, au terme d'un calcul de plus d'un an sur 80 machines environ.

### 2.3.6. Critères de non-primalité tirés du petit théorème de Fermat

Le test de non-primalité le plus simple pour un entier  $n > 1$  traduit la définition même de la primalité :

▷  $n$  est composé s'il existe  $a$  avec  $1 < a < n$  qui divise  $n$ .

Un tel  $a$  sera un « témoin de non-primalité ». S'il n'en existe pas, le nombre est premier. La plupart des tests de non-primalité utilisent sur ce modèle un entier auxiliaire.

On peut améliorer le test précédent : si  $n$  n'est pas premier, il possède au moins deux facteurs ; le plus petit de ses diviseurs est donc inférieur à  $\sqrt{n}$ . On obtient ainsi :

▷  $n$  est composé s'il existe un diviseur  $a$  de  $n$  tel que  $1 < a \leq \sqrt{n}$ .

Cela est une petite amélioration, mais le principe même du test est mauvais. Prenons le pire des cas, celui où  $n$  est le produit de deux grands nombres premiers distincts, par exemple

$$20! + 1 = 2432902008176640001 = 20639383 \times 117876683047.$$

Si on commence à tester la divisibilité par 3, 5, ... on attendra longtemps. Si on prend des  $a$  au hasard, on n'a aucune chance de tomber sur l'un des deux facteurs. En résumé, les témoins du test précédent sont à la fois *grands et rares*. On ne peut raisonnablement espérer les découvrir, ni par une recherche exhaustive<sup>15</sup> ni par une recherche aléatoire.

Une seconde petite amélioration pourrait être apportée en remplaçant l'existence d'un diviseur  $a$  par l'existence d'un nombre  $a$  non premier à  $n$  (fait que l'on teste par l'algorithme d'Euclide) :

▷  $n$  est composé s'il existe  $a$  non premier à  $n$  avec  $1 < a \leq \sqrt{n}$ .

Si on reprend le cas de  $20! + 1 = pq$ , on voit qu'on n'a guère amélioré les choses : en effet, la probabilité qu'un entier soit multiple de  $p$  est  $1/p$ , celle qu'il soit multiple de  $p$  ou multiple de  $q$  est  $1/p + 1/q - 1/pq$ .

Une nouvelle lignée de critères provient du petit théorème de Fermat. Le plus simple est le suivant :

▷  $n$  est composé s'il existe  $a$  avec  $a^{n-1} \not\equiv 1 \pmod{n}$  et  $1 < a < n$ .

La condition est évidemment suffisante, puisqu'elle contredit la conclusion du petit théorème de Fermat. Elle est nécessaire pour une raison stupide : si  $a$  n'est pas premier à  $n$ , alors aucune puissance de  $a$  ne peut être congrue à 1

---

<sup>15</sup>. Une recherche exhaustive serait éventuellement possible dans cet exemple, que nous avons choisi assez petit pour simplifier, mais sûrement pas pour un produit de deux nombres premiers d'une centaine de chiffres.

modulo  $n$ , car  $a$  serait alors inversible modulo  $n$ . En fait, le critère précédent ajoute aux témoins venant de la divisibilité les « vrais » contre-exemples au petit théorème de Fermat : les  $a$  premiers à  $n$  tels que  $a^{n-1} \not\equiv 1 \pmod{n}$ , et que l'on peut appeler des *témoins de Fermat*. Le grand avantage est maintenant qu'il y a fréquemment de très petits témoins de Fermat. En particulier, on constate expérimentalement que  $a = 2$  (ou à défaut  $a = 3$ ) suffit à détecter la majorité des nombres composés de petite taille !

**EXERCICE 2.28. [B]** — On suppose que  $n = pq$  est le produit de deux nombres premiers tels que  $\text{pgcd}(p - 1, q - 1) = 2$ . Montrer que  $2^{n-1} \not\equiv 1 \pmod{n}$ . Généraliser au cas où, posant  $d = \text{pgcd}(p - 1, q - 1)$ , on a  $2^d \leq n$ .

**EXERCICE 2.29. [N]** — Calculer  $3^{2^{32}}$  modulo  $2^{32} + 1$ . En conclure que  $2^{32} + 1$  n'est pas premier.

**EXERCICE 2.30. [N]** — On utilise le critère suivant pour vérifier la primalité :  $m$  est premier à 30 (c'est-à-dire n'est divisible ni par 2, ni par 3, ni par 5),  $2^{m-1} \equiv 1 \pmod{m}$  et  $3^{m-1} \equiv 1 \pmod{m}$ . Quels sont les plus petites valeurs de  $m$  pour lesquelles ce critère est en défaut ?

Il y a malheureusement des cas où le « critère de Fermat » ne marche pas mieux que les précédents. Il existe en effet des nombres composés  $n$  tels que l'on ait  $a^{n-1} \equiv 1 \pmod{n}$  pour tout  $a$  premier à  $n$  et  $a^n \equiv a \pmod{n}$  pour tout  $a$ , donc *sans témoins de Fermat*. On les appelle les *nombres de Carmichael* (définition 3.34). Le plus petit est 561, on sait depuis peu qu'il en existe une infinité.

**EXERCICE 2.31. [C]** — Montrer que, ou bien  $n$  ne possède aucun témoin de Fermat, ou bien il en possède au moins  $n/2$ .

Continuons dans la même veine. Soit  $n$  un nombre premier impair<sup>16</sup> et soit  $a$  un entier avec  $1 < a < n$ . On a  $(a^{(n-1)/2})^2 \equiv 1 \pmod{n}$ . Or il n'y a dans le corps  $\mathbf{Z}/n\mathbf{Z}$  que deux racines carrées de 1, à savoir 1 et  $-1$ . Cela implique  $a^{(n-1)/2} \equiv \pm 1 \pmod{n}$ . Si un entier  $a$  met en défaut cette égalité, c'est que  $n$  ne pouvait être premier :

▷  $n$  impair est composé s'il existe  $a$  avec  $a^{(n-1)/2} \not\equiv \pm 1 \pmod{n}$  et  $1 < a < n$ .

En fait, ce nouveau critère introduit de nouveaux témoins pour la plupart des nombres de Carmichael, mais pas pour tous. Il existe en effet des nombres composés  $n$  pour lesquels, pour tout  $a$  premier à  $n$ , on a  $a^{(n-1)/2} \equiv 1 \pmod{n}$ . C'est par exemple le cas du nombre  $7 \cdot 13 \cdot 19 = 1729$ .

**EXERCICE 2.32. [C]** — Soit  $n$  un entier impair tel que  $a^{(n-1)/2} \equiv \pm 1 \pmod{n}$  pour tout  $a$  premier à  $n$ . En utilisant la proposition 3.33, montrer que, s'il existe un  $a$  tel que  $a^{(n-1)/2} \equiv -1 \pmod{n}$ , alors  $n$  est premier.

---

16. Évidemment, le seul nombre premier pair est 2 ! Mais l'usage est de dire « nombre premier impair » plutôt que « nombre premier  $\neq 2$  ». D'ailleurs, suivant une plaisanterie américaine classique, *two is the oddest of all primes* !

### 2.3.7. Le critère de Miller-Rabin

La difficulté créée par les nombres précédents peut être levée par la remarque suivante, due à Gary MILLER : si on trouve  $a^{(n-1)/2} \equiv 1 \pmod{n}$  et si  $(n - 1)/2$  est pair, on peut recommencer, et ainsi de suite.

**PROPOSITION 2.20.** — *Soit  $p > 2$  un nombre premier. Écrivons  $p - 1 = 2^s t$  avec  $t$  impair. Soit  $a$  un entier non divisible par  $p$ . Alors, ou bien  $a^t \equiv 1 \pmod{p}$ , ou bien il existe un entier  $i$  avec  $0 \leq i < s$  et  $a^{2^i t} \equiv -1 \pmod{n}$ .*

*Démonstration.* Posons  $a_i = a^{2^i t} \pmod{n}$  pour  $i = 0, \dots, s$ . On a  $a_s = 1$  d'après Fermat. Ainsi, de deux choses l'une : ou bien tous les  $a_i$  et en particulier  $a_0$  sont égaux à 1, ou bien il existe  $i$  avec  $0 \leq i < s$ ,  $a_i \neq 1$  et  $a_{i+1} = 1$ . Mais, puisque  $a_i^2 = a_{i+1} = 1$  et que  $p$  est premier, cela implique  $a_i = -1$ .  $\square$

**COROLLAIRE 2.21.** — *Soit  $n > 1$  un entier impair. Écrivons  $n - 1 = 2^s t$  avec  $t$  impair. Supposons qu'il existe un entier  $a$  avec  $1 < a < n$ ,  $a^t \not\equiv 1 \pmod{n}$  et  $a^{2^i t} \not\equiv -1 \pmod{n}$  pour  $i = 0, 1, \dots, s - 1$ . Alors  $n$  est composé.*

Appelons un tel entier  $a$  un *témoin de Miller*.

**PROGRAMME 2.22.** — *Témoins de Miller*

---

```
class Integer
  def is_a_Miller_witness_for(n)
    s, t = 0, n-1
    until t.odd? do
      s, t = s+1, t/2
    end
    x = self.puissmod(t,n)
    return false if x == 1 || x == n-1
    (s-1).times {return false if (x = x*x % n) == n-1}
    return true
  end
end
```

---

Si  $n$  est composé, il possède des témoins de Miller, par exemple ses diviseurs, ce qui n'est pas nouveau. Mais, ce qui est nouveau, c'est qu'il en a beaucoup :

**PROPOSITION 2.23.** — *Si le nombre impair  $n$  est composé, au moins les trois quarts des  $n - 2$  entiers  $a$  tels que  $1 < a < n$  sont des témoins de Miller pour  $n$ .*

Donnons quelques exemples.

D'abord celui du nombre de Carmichael 561. On a  $a^{560} \equiv 1 \pmod{561}$  pour tout  $a$  premier à 561, par exemple pour  $a = 2$ . Mais on a  $560 = 2^4 \cdot 35$ ,  $2^{2^3 \cdot 35} \equiv 1 \pmod{561}$  et  $2^{2^2 \cdot 35} \equiv 67 \pmod{561}$ , de sorte que 2 est un témoin de Miller. En fait, il y en a 550 sur les 559 possibles :

```
(1...561).select{|a| a.is_a_Miller_witness_for(561) == true}.size
# => 550
```

Regardons aussi un cas extrême, celui du plus petit nombre composé impair  $n = 9$ . Les entiers  $2, \dots, 7$  sont des témoins de Miller et il y en a 6 sur 7 possibles et on a bien  $6 \geq (3/4) \cdot 7$ .

Il y a souvent de très petits témoins de Miller. Le plus petit nombre pour lequel ni 2, ni 3, ni 5 ne sont des témoins de Miller est 25 326 001 :

```
pseudo = 2251*11251 # => 25326001
[2,3,5,7].collect { |a| a.is_a_Miller_witness_for(pseudo)}
# => [false, false, false]
```

La proposition précédente est conséquence d'un théorème un peu plus précis que nous démontrerons plus loin (voir 3.3.6) :

**THÉORÈME 2.24 (Rabin).** — *Soit  $n$  un entier impair composé et  $> 9$ . Posons  $n - 1 = 2^s t$  avec  $t$  impair. Les entiers  $a$  compris entre 1 et  $n$  et qui satisfont à la condition  $a^t \equiv 1 \pmod{n}$  ou à l'une des conditions  $a^{2^i t} \equiv -1 \pmod{n}$  pour  $i = 0, 1, \dots, s - 1$  sont en nombre au plus  $\varphi(n)/4$ .*

Nous verrons aussi plus tard comment le résultat précédent est la base d'un test *probabiliste* de primalité. Notons dès à présent que pour  $a$  donné, le test demande le calcul de  $a^t \pmod{n}$  et les  $s$  élévations au carré successives donnant les  $a^{2^i t} \pmod{n}$ . Le premier a un coût en  $(\log t)(\log n)^2$ , chacun des autres un coût en  $(\log n)^2$ . Au total, puisque  $s + \log t = \log(n - 1) \leq \log n$ , chaque instance du test a un coût en  $(\log n)^3$ .



# Chapitre 3

## Racines primitives

Le chapitre précédent a été consacré au théorème de Fermat : si  $p$  est un nombre premier et  $a$  est premier à  $p$ , alors  $a^{p-1}$  est congru à 1 modulo  $p$ . Mais naturellement, il se peut que  $p - 1$  ne soit pas le plus petit exposant  $r > 1$  tel que  $a^r \equiv 1 \pmod{p}$ . Le point essentiel autour duquel tourne tout ce chapitre, c'est qu'on peut trouver des  $a$  (que l'on appelle *racines primitives modulo p*) pour lesquels  $p - 1$  est effectivement le plus petit de ces exposants (corollaire 3.9).

Comme dans le chapitre précédent, cela va nous donner des critères de non-primalité, mais aussi, ce qui est nouveau, des critères de primalité, car l'existence d'une racine primitive modulo un entier  $n > 1$  implique que  $n$  est premier. Ce qui nous permettra d'étudier la primalité des nombres de Fermat (de la forme  $2^r + 1$ ) et de Mersenne (de la forme  $2^r - 1$ ).

### § 3.1. Structure du groupe $\mathbf{K}^*$

#### 3.1.1. Groupes cycliques

On appelle *groupe cyclique* un groupe fini qui peut être engendré par un seul élément (qu'on dit alors être un générateur). Dire que  $g \in G$  engendre  $G$ , ou encore par définition que tout élément de  $G$  est une puissance de  $g$ , c'est aussi dire que l'ordre de  $g$  est égal à l'ordre du groupe  $G$ .

Si  $G$  est un groupe cyclique, d'ordre  $r$ , et  $g$  un générateur de  $G$ , l'application  $(n \bmod r) \mapsto g^n$  est un isomorphisme du groupe  $\mathbf{Z}/r\mathbf{Z}$  des entiers modulo  $r$  sur  $G$  (voir 2.3.1). Deux groupes cycliques de même ordre sont donc isomorphes. Tout élément  $h$  de  $G$  s'écrit  $g^m$  pour un unique entier  $m$  tel que  $0 \leq m < r$ . Le sous-groupe engendré par  $h$  est formé des puissances  $g^{mn}$ , où  $n$  parcourt  $\mathbf{Z}$ . Pour que ce sous-groupe soit égal à  $G$  tout entier, il faut et il suffit que  $m$  soit inversible modulo  $r$ . Ainsi *le nombre des générateurs d'un groupe cyclique à r éléments est le nombre  $\varphi(r)$  des entiers compris entre 0 et r et premiers à r*.

#### 3.1.2. Exposant d'un groupe commutatif fini

Soit  $G$  un groupe fini. D'après le théorème de Lagrange (théorème 2.8), on a  $x^{\#G} = e$  pour tout élément  $x$  de  $G$ , ce qui signifie aussi que  $\#G$  est un multiple commun des ordres de tous les éléments de  $G$ . Mais ce n'est pas

nécessairement le plus petit entier  $> 0$  ayant cette propriété, ce qui amène à la définition suivante.

On note  $\omega(G)$ , et on appelle *exposant* de  $G$ , le ppcm des ordres des éléments de  $G$ . C'est donc le plus petit entier  $> 0$  tel que  $x^{\omega(G)} = e$  pour tout  $x \in G$ . Il divise  $\#G$ .

**EXERCICE 3.1. [A]** — Le groupe des permutations de trois éléments, qui est d'ordre 6, est d'exposant 6. Il possède des éléments d'ordre 2 et des éléments d'ordre 3, mais pas d'élément d'ordre 6.

La situation est particulièrement agréable lorsque le groupe  $G$  est *commutatif* :

**LEMME 3.1.** — *Soient  $x$  et  $y$  deux éléments d'ordre fini dans un groupe, tels que  $xy = yx$ . Si l'ordre  $m$  de  $x$  et l'ordre  $n$  de  $y$  sont premiers entre eux, alors l'ordre de  $xy$  est  $mn$ .*

*Démonstration.* On a d'abord  $(xy)^{mn} = (x^m)^n(y^n)^m = e$ . Inversement, soit  $r$  un entier  $> 0$  tel que  $(xy)^r = e$ . On a  $e = (xy)^{rm} = (x^m)^r(y^{rm}) = y^{rm}$ , donc  $n$  divise  $rm$  ; puisque  $n$  est premier à  $m$ , il divise  $r$ . On voit de même que  $m$  divise  $r$ . Ainsi  $r$  est un multiple commun de  $n$  et de  $m$ , donc un multiple de  $mn$ .  $\square$

**LEMME 3.2.** — *Dans un groupe commutatif, l'ensemble des ordres des éléments est stable par ppcm.*

*Démonstration.* Soient en effet  $x$  un élément d'ordre  $m$  et  $y$  un élément d'ordre  $n$ . Il s'agit de construire un élément dont l'ordre est le ppcm de  $m$  et  $n$ , soit  $r$ . Or on peut écrire  $r = m'n'$ , où  $m'$  divise  $m$ ,  $n'$  divise  $n$ , et les deux entiers  $m'$  et  $n'$  sont premiers entre eux (cela résulte de la décomposition en facteurs premiers, mais on peut aussi le faire en n'utilisant que l'algorithme du pgcd, voir l'exercice 3.2 ci-dessous). Alors  $x^{m/m'}$  est d'ordre  $m'$ , et  $y^{n/n'}$  est d'ordre  $n'$ , et on peut appliquer le lemme.  $\square$

**EXERCICE 3.2. [C]** — Montrer comment on peut calculer des entiers  $m'$  et  $n'$  explicitement, en utilisant uniquement l'algorithme du pgcd.

On déduit aussitôt du lemme précédent :

**PROPOSITION 3.3.** — *Soit  $G$  un groupe fini commutatif. Il existe un élément de  $G$  d'ordre  $\omega(G)$ .*

Ainsi, parmi les groupes commutatifs finis, l'égalité  $\omega(G) = \#G$  caractérise les groupes cycliques.

**EXERCICE 3.3. [C]** — Soient  $G$  et  $G'$  deux groupes finis commutatifs. Prouver la relation  $\omega(G \times G') = \text{ppcm}(\omega(G), \omega(G'))$ . En déduire que, pour que le groupe produit  $G \times G'$  soit cyclique, il faut et il suffit que  $G$  et  $G'$  soient cycliques et d'ordres premiers entre eux.

### 3.1.3. Racines primitives de l'unité

Soit  $A$  un anneau commutatif. Un élément  $\zeta$  de  $A$  est appelé une *racine de l'unité* si c'est un élément d'ordre fini du groupe  $A^*$ , c'est-à-dire s'il existe un entier  $r > 0$  tel que  $\zeta^r = 1_A$  (on dit alors que  $\zeta$  est une *racine  $r$ -ième de l'unité* dans  $A$ ).

Un élément d'ordre  $r$  du groupe  $A^*$  est appelé une racine *primitive  $r$ -ième de l'unité* dans  $A$ . Dire que  $\zeta$  est une racine primitive  $r$ -ième de l'unité dans  $A$  signifie que  $\zeta^r = 1$ , et que  $\zeta^{r/q} \neq 1$  pour tout diviseur premier  $q$  de  $r$  : en effet la première condition implique que l'ordre  $s$  de  $\zeta$  divise  $r$  et la seconde que  $r/s$  n'est divisible par aucun nombre premier.

**EXERCICE 3.4. [A]** — Prouver que l'ensemble des  $r$  tels qu'il existe une racine primitive  $r$ -ième de l'unité dans  $A$  est stable par ppcm.

Rappelons qu'un anneau (commutatif) est dit *intègre* s'il est non nul et si le produit de deux éléments non nuls est toujours non nul.

**EXERCICE 3.5. [B]** — Supposons  $A$  intègre, soit  $n$  un entier  $\geq 1$  et soit  $\zeta \in A$ . Montrer que, pour que  $\zeta$  soit une racine primitive  $(2^n)$ -ième de l'unité, il faut et il suffit que  $2 \cdot 1_A \neq 0$  et que  $\zeta^{2^{n-1}} = -1$ .

Dans un anneau intègre, un polynôme ne peut avoir qu'un nombre fini de racines. Plus précisément :

**PROPOSITION 3.4.** — *Supposons l'anneau  $A$  intègre, et soit  $P \in A[X]$ . Soient  $a_1, \dots, a_m$  des éléments distincts de  $A$  tels que  $P(a_i) = 0$ . Alors  $P(X)$  est divisible dans  $A[X]$  par le produit  $(X - a_1) \cdots (X - a_m)$ .*

*Démonstration.* En effectuant la division euclidienne (1.39) de  $P$  par le polynôme unitaire  $X - a_m$ , on peut écrire  $P(X) = Q(X)(X - a_m) + b$ , avec  $b \in A$ . Prenant  $X = a_m$ , on trouve  $b = P(a_m) = 0$ , de sorte que  $P(X) = Q(X)(X - a_m)$ . Pour  $i < m$ , on a  $P(a_i) = 0$ , donc  $Q(a_i)(a_i - a_m) = 0$ , ou encore  $Q(a_i) = 0$  puisque  $A$  est supposé intègre. Ainsi le polynôme  $Q(X)$  a  $a_1, \dots, a_{m-1}$  comme racines et on conclut par récurrence.  $\square$

**COROLLAIRE 3.5.** — *Si l'anneau  $A$  est intègre, un polynôme de degré  $n$  a au plus  $n$  racines dans  $A$ .*

**PROPOSITION 3.6.** — *Soit  $A$  un anneau intègre. Soit  $G$  un sous-groupe fini, d'ordre  $r$ , du groupe  $A^*$ . Alors  $G$  est cyclique, et c'est l'ensemble des racines  $r$ -ièmes de l'unité dans  $A$ . On a dans  $A[X]$  la relation*

$$X^r - 1 = \prod_{\zeta \in G} (X - \zeta).$$

*Démonstration.* Soit  $s = \omega(G)$  l'exposant de  $G$ . Alors  $s$  divise  $r$  et on a  $x^s = 1$  pour tout  $x \in G$ . Le polynôme  $X^s - 1$  admet donc les  $r$  éléments de  $G$  comme racines.

Mais il ne peut avoir au plus que  $s$  racines dans  $A$ , ce dernier étant supposé intègre. Puisque  $s \leq r = \#G$ , il en résulte d'une part que  $r = s$ , ce qui signifie que  $G$  est cyclique (proposition 3.3), et d'autre part que ce polynôme a exactement  $r$  racines qui sont les éléments de  $G$ . Cela implique la relation annoncée (proposition 3.4).  $\square$

**EXERCICE 3.6. [B]** — Montrer qu'on a en outre  $r \cdot 1_A \neq 0$ .

**COROLLAIRE 3.7.** — Soient  $A$  un anneau intègre et  $n$  un entier  $> 0$ .

a) Le nombre des racines  $n$ -ièmes de l'unité dans  $A$  divise  $n$ .

b) Les conditions suivantes sont équivalentes :

- (i) il y a exactement  $n$  racines  $n$ -ièmes de l'unité dans  $A$  ;
- (ii) il y a dans  $A$  au moins une racine primitive  $n$ -ième de l'unité ;
- (iii) il y a dans  $A$  exactement  $\varphi(n)$  racines primitives  $n$ -ièmes de l'unité.

c) Soit  $\zeta$  une racine primitive  $n$ -ième de l'unité dans  $A$ . Les  $n$  racines  $n$ -ièmes de l'unité dans  $A$  sont les  $\zeta^i$ , où  $i$  parcourt  $\mathbf{Z}$  modulo  $n$ . Pour que  $\zeta^i$  soit primitive, il faut et il suffit que  $i$  soit premier à  $n$ .

*Démonstration.* Les racines  $n$ -ièmes de l'unité dans  $A$  forment un sous-groupe de  $A^*$ . Ce groupe est fini, puisque le polynôme  $X^n - 1 = 0$  ne peut avoir qu'un nombre fini de racines dans l'anneau intègre  $A$ . On peut alors lui appliquer la proposition précédente.  $\square$

**EXERCICE 3.7. [B]** — Finir la démonstration.

### 3.1.4. Racines primitives modulo $p$

La proposition précédente s'applique notamment au cas d'un corps commutatif fini  $K$ , en prenant pour  $G$  le groupe multiplicatif  $K^*$  :

**THÉORÈME 3.8.** — Le groupe multiplicatif d'un corps fini est cyclique.

En particulier :

**COROLLAIRE 3.9.** — Soit  $p$  un nombre premier.

a) Le groupe  $(\mathbf{Z}/p\mathbf{Z})^*$  est cyclique, d'ordre  $p - 1$  : il existe  $\varphi(p - 1)$  racines primitives  $(p - 1)$ -ièmes de l'unité dans le corps  $\mathbf{Z}/p\mathbf{Z}$ .

b) Pour qu'il existe des racines primitives  $d$ -ièmes de l'unité dans  $\mathbf{Z}/p\mathbf{Z}$ , il faut et il suffit que  $d$  divise  $p - 1$ .

c) Pour tout diviseur  $d$  de  $p - 1$ , il existe  $\varphi(d)$  racines primitives  $d$ -ièmes de l'unité dans le corps  $\mathbf{Z}/p\mathbf{Z}$ .

**REMARQUE 3.10.** — L'existence de racines primitives  $(p - 1)$ -ièmes modulo  $p$  a été énoncée par Euler en 1773. Sa démonstration a été complétée par Gauss en 1801.

Voici un exemple. Pour  $p = 71$ , 7 est une racine primitive, et on a  $2 \equiv 7^6$ ,  $3 \equiv 7^{26}$  et  $5 \equiv 7^{28}$ , donc aussi  $4 \equiv 7^{12}$  et  $6 \equiv 7^{32}$ ; ainsi, chacune des classes de 2, 3, 4, 5 et 6 est une puissance paire de la classe de 7, donc n'est pas une racine primitive.

### 3.I.5. Recherche des racines primitives

Il n'existe pas de méthode universelle simple autre que la recherche exhaustive pour trouver des racines primitives modulo un nombre premier donné  $p$ . On voit au passage sur cet exemple la distance qu'il y a entre l'énoncé *théorique* trivial qui affirme qu'en tant que groupe cyclique à  $p - 1$  éléments, le groupe  $(\mathbf{Z}/p\mathbf{Z})^*$  est isomorphe à  $\mathbf{Z}/(p - 1)\mathbf{Z}$ , et sa contrepartie *constructive* consistant à exhiber un isomorphisme, c'est-à-dire à donner une racine primitive modulo  $p$ .

En utilisant l'hypothèse de Riemann généralisée, conjecture très célèbre<sup>1</sup> de la théorie des nombres, on a pu montrer qu'il existe une racine primitive modulo  $p$  inférieure à  $C \cdot (\log p)^6$ , pour une constante  $C$  convenable. Cela justifie la recherche d'une telle racine par énumération des entiers successifs :

PROGRAMME 3.II. — *Plus petite racine primitive modulo p* ——————

```
def least_primitive_root_mod(p)
    for a in 2...p do
        return a if a.ordre_mod(p) == p-1
    end
    raise ArgumentError, "#{p} is not prime"
end
[3,7,71,191,719].collect { |p| least_primitive_root_mod(p)}
# => [2, 3, 7, 19, 11]
```

Les nombres premiers qui sont donnés ci-dessus sont des cas extrêmes. Par exemple, tout nombre premier  $\leq 67$  a une racine primitive  $\leq 5$ . On saute à 7 pour 71 et tout nombre premier  $\leq 181$  une racine primitive  $\leq 7$ . On saute à 19 pour 191, à 21 pour 401...

REMARQUE 3.I2. — La vérification qu'un candidat  $a$  est ou non une racine primitive modulo  $p$  peut être accélérée par la remarque suivante (on suppose  $p > 2$ ) : si  $a$  est une racine primitive modulo  $p$ , alors  $a^{(p-1)/2} \equiv -1 \pmod{p}$  ; comme c'est une racine carrée de l'unité dans le corps  $\mathbf{Z}/p\mathbf{Z}$ , ce doit être  $-1$ . La condition nécessaire  $a^{(p-1)/2} \equiv -1$  élimine la moitié des candidats ; elle exprime le fait que  $a$  est un résidu quadratique modulo  $p$  et peut se vérifier ou s'infirmer très rapidement, comme nous le verrons (5.3.2).

En tout cas, l'existence d'un entier  $a$  tel que  $a^{(n-1)/2} \equiv -1 \pmod{n}$  est donc une condition nécessaire pour que  $n$  soit premier. On dit souvent que l'entier  $n$  est un « probable prime for base  $a$  » si  $a^{(n-1)/2} \equiv -1 \pmod{n}$ .

Notons qu'il y a  $\varphi(p - 1)$  racines primitives parmi les  $p - 1$  entiers  $1, \dots, p - 1$  (on pourrait évidemment se restreindre à  $2, \dots, p - 2$ ), donc que la probabilité qu'un entier choisi au hasard dans cet intervalle convienne est le quotient  $\varphi(p - 1)/(p - 1)$ , qui n'est autre que le produit des  $1 - 1/q$ ,

1. Et aussi très difficile, mais le cas spécifiquement utilisé a été numériquement vérifié pour les tailles de nombres qui nous intéressent.

où  $q$  parcourt les diviseurs premiers de  $p - 1$  (voir 2.2.2). Or, ce produit peut être très petit si  $p - 1$  a beaucoup de facteurs premiers (voir l'exercice 2.7). Chercher une racine primitive au hasard n'est donc pas une bonne stratégie générale (voir cependant la remarque 3.13 ci-dessous).

**REMARQUE 3.13.** — On peut donner une méthode « probabiliste » pour trouver des racines primitives modulo  $p$  lorsqu'on connaît la décomposition en facteurs premiers de  $p - 1$ , soit

$$p - 1 = q_1^{r_1} \cdots q_s^{r_s}.$$

Il suffit de trouver pour chaque  $i$  un élément  $a_i$  d'ordre  $q_i^{r_i}$ , et de faire le produit des  $a_i$  (lemme 3.1). Or si  $x$  est un entier quelconque non divisible par  $p$ , et si l'on pose  $a_i = x^{(p-1)/q_i^{r_i}}$ , alors  $a_i$  convient à moins que l'on ait  $x^{(p-1)/q_i} \equiv 1 \pmod{p}$ , ce qui ne se produit que dans un cas sur  $q_i$ .

### § 3.2. Critères de primalité

On a déjà remarqué que la définition même des nombres premiers n'est pas effective : il faudrait diviser le nombre donné par tous les nombres inférieurs (ou en tout cas par beaucoup d'entre eux). Dans ce paragraphe, nous montrons qu'on peut donner des critères effectifs de primalité, en nous restreignant aux deux types de critères les plus simples : ceux « à la Lehmer », où l'on suppose donnée une décomposition en facteurs premiers de  $p - 1$ , et ceux « à la Lucas », où l'on suppose donnée une décomposition en facteurs premiers de  $p + 1$ . Ces critères s'appliquent notamment au cas des nombres de Fermat ( $p = 2^s + 1$ ) et de Mersenne ( $p = 2^s - 1$ ).

#### 3.2.1. Critères de primalité « à la Lehmer »

Soit  $n > 1$  un entier. Si  $n$  est premier, il existe une racine primitive  $(n - 1)$ -ième de l'unité dans  $\mathbf{Z}/n\mathbf{Z}$ . Inversement, s'il existe une racine primitive  $(n - 1)$ -ième de l'unité  $a$  dans  $\mathbf{Z}/n\mathbf{Z}$ , alors  $(\mathbf{Z}/n\mathbf{Z})^*$  est d'ordre  $\geq n - 1$ , donc  $\mathbf{Z}/n\mathbf{Z}$  est un corps, et  $n$  est premier. Calculer toutes les puissances de  $a$  modulo  $n$  pour trouver son ordre ne fournira pas une méthode nettement plus rapide que celle que nous avons rejetée ci-dessus. Mais, si on connaît à l'avance la décomposition en facteurs premiers de  $n - 1$ , on peut donner un critère utilisable :

**PROPOSITION 3.14** (Critère de Lehmer). — *Soit  $n > 1$  un entier impair tel que l'on connaisse tous les facteurs premiers de  $n - 1$ . Les conditions suivantes sont équivalentes :*

- (i) *n est premier ;*
- (ii) *il existe un entier a tel que  $a^{n-1} \equiv 1 \pmod{n}$  et  $a^{(n-1)/q} \not\equiv 1 \pmod{n}$  pour tout facteur premier q de n - 1.*

*Démonstration.* On a vu que (i) implique (ii) (on prend pour  $a$  une racine primitive  $(n - 1)$ -ième de l'unité). Supposons inversement (ii) satisfaite. Alors l'ordre de l'élément  $a \bmod n$  de  $(\mathbf{Z}/n\mathbf{Z})^*$  divise  $n - 1$ , mais ne divise aucun des  $(n - 1)/q$ . Il est donc égal à  $n - 1$ . Par conséquent,  $(\mathbf{Z}/n\mathbf{Z})^*$  possède au moins  $n - 1$  éléments (les puissances de  $a \bmod n$ ) et  $n$  est premier.  $\square$

On sait que le coût du calcul de  $a^m \bmod n$  est en  $(\log m)(\log n)^2$ . La décomposition de  $n - 1$  étant connue et le nombre  $a$  étant choisi, le coût de la vérification du critère de Lehmer est donc au plus en  $k(\log n)^3$ , où  $k$  est le nombre de facteurs premiers de  $n - 1$ . On peut majorer très brutalement  $k$  par  $\log n$  (puisque chaque facteur est  $\geq 2$ ), et on obtient *un coût total au plus en*  $(\log n)^4$ .

On peut donner une variante utile du critère de Lehmer :

**COROLLAIRE 3.15.** — Soit  $n$  un entier impair  $> 2$ . Les conditions suivantes sont équivalentes :

- (i)  $n$  est premier ;
- (ii) il existe un entier  $a$  tel que  $a^{(n-1)/2} \equiv -1 \pmod{n}$  et  $a^{(n-1)/q} \not\equiv 1 \pmod{n}$  pour tout facteur premier impair  $q$  de  $n - 1$ .

*Démonstration.* En effet, dans un corps, le seul élément d'ordre 2 est  $-1$ . La conjonction des conditions  $a^{n-1} \equiv 1 \pmod{n}$  et  $a^{(n-1)/2} \not\equiv 1 \pmod{n}$  peut donc être remplacée par  $a^{(n-1)/2} \equiv -1 \pmod{n}$ .  $\square$

On peut donner aussi des variantes du critère précédent qui « séparent les contributions des différents facteurs premiers de  $n - 1$  ».

**LEMME 3.16** (Critère de Pocklington). — Soit  $n$  un entier  $> 1$ . Écrivons  $n - 1 = q^r m$ , avec  $q$  premier et  $r \geq 1$ . Supposons qu'il existe un entier  $a$  avec  $a^{q^r} \equiv 1 \pmod{n}$  et  $\text{pgcd}(a^{q^{r-1}} - 1, n) = 1$ . Alors tout facteur premier de  $n$  est congru à 1 modulo  $q^r$ .

*Démonstration.* Soit  $p$  un facteur premier de  $n$ . On a  $a^{q^r} \equiv 1 \pmod{p}$  et  $a^{q^{r-1}} \not\equiv 1 \pmod{p}$ . Cela signifie que l'ordre de  $a \bmod p$  dans le groupe  $(\mathbf{Z}/p\mathbf{Z})^*$  divise  $q^r$  et ne divise pas  $q^{r-1}$ , donc est égal à  $q^r$ . Donc  $q^r$  divise  $p - 1$  (théorème de Lagrange).  $\square$

**EXERCICE 3.8. [A]** — On n'a pas fait l'hypothèse que  $m$  est premier à  $q$ . Commenter.

**PROPOSITION 3.17** (Critère de Lehmer-Pocklington). — Soit  $n$  un entier  $> 1$ . Écrivons  $n - 1 = uv$ , les facteurs premiers de  $u$  étant connus. Supposons qu'il existe pour chaque facteur premier  $q$  de  $u$ , en désignant par  $q^r$  la plus grande puissance de  $q$  qui divise  $u$ , un entier  $a_q$  avec  $a_q^{q^r} \equiv 1 \pmod{n}$  et  $\text{pgcd}(a_q^{q^{r-1}} - 1, n) = 1$ . Alors tout facteur premier  $p$  de  $n$  est congru à 1 modulo  $u$ . Si on a de plus  $v \leq u + 1$ , alors  $n$  est premier.

*Démonstration.* Soit  $u = q_1^{r_1} \cdots q_s^{r_s}$  la décomposition de  $u$  en facteurs premiers et soit  $p$  un facteur premier de  $n$ . D'après le lemme précédent,  $p - 1$  est divisible par chacun des  $q_i^{r_i}$ , donc est divisible par  $u$ . Cela implique en particulier  $p > u$ ; si on a  $v \leq u + 1$ , on a  $n = 1 + uv < (u + 1)^2 \leq p^2$ . Par conséquent, tout facteur premier  $p$  de  $n$  est  $> \sqrt{n}$ , et  $n$  est premier.  $\square$

**EXERCICE 3.9.** [N] — On prend  $n = 12289$ . Calculer  $3^{2^8}$  modulo  $n$ . En conclure que  $n$  est premier.

### 3.2.2. Certificats de primalité

Revenons au critère de Lehmer. Pour l'utiliser pour *certifier* qu'un nombre  $n$  est premier, il faut connaître la décomposition de  $n - 1$  en facteurs premiers, et notamment certifier au préalable que ces facteurs sont bien premiers. On aboutit ainsi à une notion récursive de *certificat de primalité à la Lehmer*: un certificat de primalité de  $n$  est la donnée d'une racine primitive  $a$ , de la décomposition  $n - 1 = 2^{e_0} q_1^{e_1} \cdots q_s^{e_s}$  et d'un certificat de primalité de chacun des  $q_i$ .

Pour clarifier les choses, nous donnons ci-dessous une définition non récursive.

**LEMME 3.18.** — *Soit  $p$  un nombre premier impair. Il existe une suite de nombres premiers impairs*

$$p_0 > p_1 > \cdots > p_r$$

*tels que  $p_0 = p$ , que chaque facteur premier impair d'un des entiers  $p_i - 1$  soit l'un des  $p_j$ , et qu'inversement chaque  $p_j$  avec  $j > 0$  divise l'un des entiers  $p_i - 1$ . De plus, une telle suite est uniquement déterminée et on a  $\sum \log p_i \leq (\log p)^2$ .*

*Démonstration.* Raisonnons par récurrence sur  $p$ . Soient  $q_1, \dots, q_s$  les facteurs premiers impairs distincts de  $p - 1$ . Il est clair que la suite cherchée est nécessairement la réunion de  $\{p\}$  et des suites correspondantes pour les  $q_i$ . Cela prouve existence et unicité. Posons  $S(p) = \sum \log p_i$ . L'argument précédent montre que  $S(p) \leq \log p + \sum S(q_i)$ . Puisque, par récurrence, on a  $S(q_i) \leq (\log q_i)^2$ , on en déduit  $S(p) \leq \log p + \sum (\log q_i)^2$ . Mais, puisque  $q_1 \cdots q_s \leq p/2$ , on a  $\log q_1 + \cdots + \log q_s \leq \log p - 1$ , et donc  $(\log q_1)^2 + \cdots + (\log q_s)^2 \leq (\log p - 1)^2$ . En définitive, cela donne  $S(p) \leq \log p + (\log p - 1)^2 \leq (\log p)^2$ .  $\square$

Nous appellerons *suite de décomposition* (de  $p$ ) l'unique suite  $p_0, \dots, p_r$  donnée par la proposition précédente. Voici des exemples de suites de décomposition : (3), (5), (19, 3), (31, 5, 3), (71, 7, 5, 3).

**DÉFINITION 3.19.** — *Soit  $p$  un entier impair. On appelle certificat de primalité de  $p$  un couple de suites d'entiers  $(p_0, p_1, \dots, p_r)$  et  $(a_0, a_1, \dots, a_r)$ , de même longueur, satisfaisant aux propriétés suivantes :*

- (i) On a  $p = p_0 > p_1 > \dots > p_r > 2$ .
- (ii) Pour chaque  $i$ ,  $p_i$  est impair, et  $p_i - 1$  est le produit d'une puissance de 2 et de certains des  $p_j$  (éventuellement répétés).
- (iii) Pour chaque  $i$ , on a  $1 < a_i < p_i$ ,  $a_i^{(p_i-1)/2} \equiv -1 \pmod{p_i}$  et

$$a_i^{\frac{p_i-1}{p_j}} \not\equiv 1 \pmod{p_i}$$

pour tout  $j$  tel que  $p_j$  divise  $p_i - 1$ .

**PROPOSITION 3.20.** — a) Tout nombre premier impair  $p$  possède un certificat de primalité tel que  $\sum \log p_i \leq (\log p)^2$ .

b) Si  $(p_0, p_1, \dots, p_r)$  et  $(a_0, a_1, \dots, a_r)$  forment un certificat de primalité de l'entier  $p$ , alors pour chaque  $i$ , l'entier  $p_i$  est premier et  $a_i$  est une racine primitive  $(p_i - 1)$ -ième de l'unité modulo  $p_i$ . En particulier  $p$  est premier et  $a_0$  est une racine primitive  $(p - 1)$ -ième de l'unité modulo  $p$ .

*Démonstration.* Pour construire un certificat de primalité de  $p$ , il suffit de prendre une suite de décomposition de  $p$ , et pour chaque  $i$  une racine primitive  $(p_i - 1)$ -ième de l'unité modulo  $p_i$ . Cela prouve a).

Inversement, partons d'un certificat de primalité de  $p$ . Notons que pour chaque  $i$ , le couple formé de  $(p_i, \dots, p_r)$  et  $(a_i, \dots, a_r)$  est un certificat de primalité de  $p_i$ . Raisonnant par récurrence sur  $p$ , on peut donc supposer que chacun des  $p_i$ , pour  $i = 1, \dots, r$  est premier. Mais alors chacun des facteurs premiers impairs de  $p - 1$  se trouve dans la suite des  $p_i$ , et on peut appliquer à  $p$  et  $a_0$  le critère de Lehmer. Cela prouve b).  $\square$

Par exemple, le couple formé des suites  $(71, 7, 5, 3)$  et  $(7, 3, 2, 2)$  est un certificat de primalité de 71.

On notera que la vérification que des suites forment bien un certificat de primalité de  $p$  a un coût au plus en  $\sum (\log p_i)^4$ . Pour un certificat de primalité « minimal » obtenu comme ci-dessus à l'aide d'une suite de décomposition, on a  $\sum \log p_i \leq (\log p)^2$  et obtient en définitive que *la vérification d'un tel certificat de primalité de  $p$  a un coût qui est au plus en  $(\log p)^8$* .

**EXERCICE 3.10. [N]** — Écrire des programmes construisant (resp. vérifiant) des certificats de primalité. Faire des tests de complexité et préciser les majorations ci-dessus.

### 3.2.3. Les nombres de Fermat

Ce qui précède fait apparaître le cas particulier des *nombres premiers de Fermat* : ce sont les nombres premiers  $p$  qui ont les certificats de primalité les plus courts, autrement dit ceux pour lesquels  $p - 1$  est une puissance de 2. Alors  $p = 2^s + 1$ , mais  $s$  est à son tour une puissance de 2 (si  $s$  peut s'écrire

$ab$  avec  $a$  impair, alors  $2^s + 1 = (2^b)^a + 1$  est divisible par  $2^b + 1$ ). Tout nombre premier de Fermat est donc de la forme

$$\text{Fer}_n = 2^{2^n} + 1.$$

Pour  $n$  variant de 0 à 4, on obtient les nombres  $2^1 + 1 = 3$ ,  $2^2 + 1 = 5$ ,  $2^4 + 1 = 17$ ,  $2^8 + 1 = 257$ ,  $2^{16} + 1 = 65537$  qui sont premiers. Fermat pensait<sup>2</sup> que tous les  $\text{Fer}_n$  étaient premiers. Mais  $\text{Fer}_5 = 2^{32} + 1 = 4294967297$  ne l'est pas ; nous l'avons d'ailleurs prouvé par une méthode que Fermat aurait pu employer ! (Exercice 2.29.) En fait,  $\text{Fer}_5 = 641 \times 6700417$ .

**EXERCICE 3.11. [B]** — Montrer que, pour  $n \neq m$ , les entiers  $\text{Fer}_n$  et  $\text{Fer}_m$  sont premiers entre eux, et qu'il y a donc une infinité de nombres premiers qui sont facteurs de nombres de Fermat.

**EXERCICE 3.12. [B]** — Soit  $p$  un facteur premier de  $\text{Fer}_n$ . Montrer que l'ordre de 2 dans le groupe des éléments inversibles modulo  $p$  est exactement  $2^{n+1}$ . En déduire que l'on a  $p \equiv 1 \pmod{2^{n+1}}$ . On verra d'ailleurs plus tard qu'on a même  $p \equiv 1 \pmod{2^{n+2}}$ .

**EXERCICE 3.13. [N]** — D'après l'exercice précédent, tout facteur premier de  $\text{Fer}_n$  est congru à 1 modulo  $2^{n+1}$ . Ainsi tout facteur premier de  $\text{Fer}_5$  est congru à 1 modulo 64, ce qui donne une liste de facteurs possibles : 193, 257, ... (en fait, 257 est exclu, pourquoi?). Pour chaque élément  $p$  de cette liste, calculer  $2^{32}$  modulo  $p$  et en déduire la décomposition de  $\text{Fer}_5$ .

**EXERCICE 3.14. [A]** — Posons  $a = 2^7$  et  $b = 5$ . Montrer qu'on a  $1 + ab - b^4 = 2^4$ , donc  $2^{32} + 1 = 2^4(2^7)^4 + 1 = (1 + ab - b^4)a^4 + 1$ . Ainsi  $2^{32} + 1$  est divisible par  $1 + ab = 641$ .

Le critère de Lehmer prend une forme particulièrement simple pour les nombres de Fermat :

**LEMME 3.21.** — *Pour que  $\text{Fer}_n$  soit premier, il faut et il suffit qu'il existe  $a$  avec  $a^{(\text{Fer}_n-1)/2} \equiv -1 \pmod{\text{Fer}_n}$ .*

**EXERCICE 3.15. [A]** — On considère un entier  $n$  de la forme  $2^r m + 1$  avec  $m \leq 2^r + 1$ . Que dit le critère de Lehmer-Pocklington ?

**EXERCICE 3.16. [A-B]** — On a  $2 \equiv -1 \pmod{3}$  et  $2^2 \equiv -1 \pmod{5}$ . Mais pour  $n > 1$ , on a  $2^{(\text{Fer}_n-1)/2} \equiv 1 \pmod{\text{Fer}_n}$ .

**EXERCICE 3.17. [N]** — On a  $3^2 \equiv -1 \pmod{5}$ ,  $3^8 \equiv -1 \pmod{17}$ ,  $3^{128} \equiv -1 \pmod{257}$ , donc 5, 17 et 257 sont premiers. Prouver par la même méthode que 65537 est premier.

**PROPOSITION 3.22 (Critère de Pépin).** — *Supposons  $n > 1$ . Pour que le nombre de Fermat  $\text{Fer}_n = 2^{2^n} + 1$  soit premier, il faut et il suffit que  $3^{2^{2^n}-1} \equiv -1 \pmod{\text{Fer}_n}$ .*

2. Il l'écrivit dans une lettre à Mersenne, en 1640.

*Démonstration.* La condition est suffisante en vertu du lemme. Inversement, on a  $\text{Fer}_n \equiv 5 \pmod{12}$  pour  $n > 1$  et nous montrerons plus loin (remarque 5.15) que pour tout nombre premier  $p$  congru à 5 modulo 12, on a  $3^{(p-1)/2} \equiv -1 \pmod{p}$ .  $\square$

En fait, ce critère est encore valable avec 5 ou 7 au lieu de 3 (voir l'exercice 5.16).

Le critère précédent semble remarquable, puisque la vérification de la condition  $3^{(\text{Fer}_n-1)/2} \equiv -1 \pmod{\text{Fer}_n}$  a un coût en  $(\log \text{Fer}_n)^3$ , donc en  $2^{3n}$ . Il suffit de poser  $a_0 = 3$  et  $a_{i+1} = a_i^2 \pmod{\text{Fer}_n}$ , et de vérifier si  $a_{n-1} \equiv -1$ .

Hélas, on n'a trouvé *aucun* nombre de Fermat premier au delà du cinquième ( $\text{Fer}_4 = 65537$ ), et il se pourrait fort bien qu'il n'en existe aucun ! De plus, la décomposition en facteurs premiers des nombres de Fermat est une question numériquement très difficile, car ces facteurs sont de fait très grands. C'est ainsi par exemple :

- a) qu'il a fallu attendre 1988 pour connaître la factorisation complète

$$\begin{aligned} \text{Fer}_{11} = & 319489 \times 974849 \times 167988556341760475137 \times \\ & \times 3560841906445833920513 \times p_{564}, \end{aligned}$$

où  $p_{564}$ , qui a 564 chiffres décimaux, est premier ;

- b) qu'on ne connaît pas encore<sup>3</sup> la décomposition totale de  $\text{Fer}_{13}$ , mais seulement le résultat partiel (le quatrième facteur a été trouvé par Brent en juin 1995)

$$\begin{aligned} \text{Fer}_{13} = & 2710954639361 \times 2663848877152141313 \times \\ & \times 3603109844542291969 \times \\ & \times 319546020820551643220672513 \times c_{2391}, \end{aligned}$$

où  $c_{2391}$  est composé et possède 2391 chiffres décimaux ;

- c) que l'on sait depuis 1963 (Selfridge et Hurwitz) que  $\text{Fer}_{14}$  est composé, sans en connaître encore<sup>4</sup> un seul facteur explicite !

### 3.2.4. Nombres de Mersenne

On appelle *nombres de Mersenne* les entiers de la forme  $2^s - 1$ . En vertu du fait que  $2^a - 1$  divise  $2^{ab} - 1$ , un tel nombre ne peut être premier que si  $s$  est premier. Si  $2^s - 1$  est premier pour  $s = 2, 3, 5, 7$ , il n'en est pas de même pour  $s = 11$ , car  $2^{11} - 1 = 2047 = 23 \times 89$ .

La détermination des nombres de Mersenne premiers est un très ancien défi<sup>5</sup>. Il y a peu de nombres de Mersenne premiers. On ne sait s'il en existe une infinité.

---

3. En novembre 2008.

4. En novembre 2008.

5. On trouvera dans [Knuth 2], 4.5.4, quelques éléments de cette histoire, lancée par une lettre de Mersenne à Fermat en 1644.

**REMARQUE 3.23.** — À l'inverse, on ne sait même pas prouver qu'il existe une infinité de nombres de Mersenne composés. Cela résulterait d'une conjecture classique : l'existence d'une infinité de nombres premiers  $p$  tels que  $2p+1$  soit premier (nombres dits « de Sophie Germain »).

En fait, on ne connaît aujourd'hui<sup>6</sup> que 46 nombres de Mersenne premiers. Les deux plus grands sont  $2^{43\,112\,609} - 1$ , découvert en août 2008, qui a 12 978 189 chiffres décimaux<sup>7</sup> et  $2^{37\,156\,667} - 1$ , inférieur, mais découvert un peu plus tard, en septembre 2008, et qui a 11 185 272 chiffres décimaux. Ces nombres ont été découverts dans le cadre du projet GIMPS<sup>8</sup>, tout comme leurs dix prédecesseurs depuis 1996. À l'exception du dernier, ces douze nombres ont été, chacun à son tour (et jusqu'à la découverte du suivant), le plus grand nombre premier connu. Il s'agit là de calculs intensifs. Par exemple, la simple vérification de la primalité de  $2^{43\,112\,609} - 1$  a pris 13 jours sur douze machines SPARC64 totalisant 32 processeurs.

Si c'est parmi les nombres de Mersenne que l'on cherche et que l'on trouve les nombres premiers record, cela résulte de l'existence d'un critère de primalité particulièrement rapide, dû à Lucas et amélioré par Lehmer, que nous verrons ci-dessous (théorème 3.31). Ce critère est un cas particulier d'un énoncé plus général et utilisable dans la pratique (proposition 3.29), qui permet d'étudier la primalité d'un entier  $n$  lorsqu'on connaît la décomposition en facteurs premiers de  $n + 1$ , ce qui est évidemment le cas pour les nombres de Mersenne.

Pour énoncer ce critère, nous aurons besoin de résultats auxiliaires sur les « suites de Lucas ».

### 3.2.5. Suites de Lucas

Supposons d'abord disposer dans un anneau  $A$  d'un élément inversible  $x$ . Possons  $a = x + x^{-1} \in A$  et  $V_n = x^n + x^{-n}$  pour tout  $n \in \mathbf{Z}$ . On a  $V_{-n} = V_n$  pour tout  $n$ .

On a par ailleurs la relation

$$\begin{aligned} aV_n &= (x + x^{-1})(x^n + x^{-n}) \\ &= x^{n+1} + x^{n-1} + x^{-n+1} + x^{-n-1} = V_{n+1} + V_{n-1}, \end{aligned}$$

d'où la formule de récurrence  $V_{n+1} = aV_n - V_{n-1}$  qui, jointe aux deux relations initiales  $V_0 = 2 \cdot 1_A$  et  $V_1 = a$ , permet de calculer de proche en

---

6. En novembre 2008.

7. L'écriture décimale de ce nombre se trouve dans le fichier <http://prime.isthe.com/no.index/chongo/merdigit/long-m43112609/prime-c.html>. Attention : ne pas cliquer sur le lien, car cela lance le chargement d'un fichier de plus de 16 megaoctets.

8. *Great Internet Mersenne Prime Search*, voir le site [http://www.mersenne.org/french\\_prime.htm](http://www.mersenne.org/french_prime.htm) où on trouvera d'autres détails.

proche les  $V_n$  pour  $n > 0$ . On voit que ce calcul récursif ne fait en aucune façon intervenir  $x$  et permet d'exprimer chaque  $V_n$  comme un polynôme en  $a$ . Posons donc la définition suivante :

**DÉFINITION 3.24.** — Soient  $A$  un anneau et  $a$  un élément de  $A$ . On appelle suite de Lucas associée à  $a$ , la suite  $(V_n)_{n \geq 0}$  d'éléments de  $A$  définie par récurrence par  $V_0 = 2 \cdot 1_A$ ,  $V_1 = a$  et  $V_{n+1} = aV_n - V_{n-1}$ .

On définira alors  $V_n$  pour  $n < 0$  par la relation  $V_{-n} = V_n$ .

Inversement, s'il existe un élément inversible  $x$  de  $A$  tel que  $a = x + x^{-1}$ , on aura  $V_n = x^n + x^{-n}$  comme on le prouve aussitôt par récurrence. Notons aussi que la relation  $V_{n+1} + V_{n-1} = aV_n$  qui sert à définir les  $V_n$  est valable pour tout  $n$  : elle est vraie pour  $n > 0$  par construction ; changeant de signe chaque indice, on obtient la relation pour  $n < 0$  ; enfin le cas  $n = 0$ , soit  $V_1 + V_{-1} = aV_0$ , s'écrit  $a + a = 2 \cdot a$ . Cette relation est d'ailleurs le cas particulier  $m = 1$  de l'énoncé suivant :

**PROPOSITION 3.25.** — On a  $V_n V_m = V_{n+m} + V_{n-m}$  pour tout  $n \in \mathbf{Z}$  et tout  $m \in \mathbf{Z}$ .

*Démonstration.* La relation est invariante par changement de  $n$  en  $-n$ , et par changement de  $m$  en  $-m$ . On peut donc supposer  $n \geq 0$  et  $m \geq 0$ . Elle est vraie pour  $m = 0$  puisque  $V_0 = 2 \cdot 1_A$  et pour  $m = 1$  puisque c'est la définition de  $V_{n+1}$ . Raisonnons donc par récurrence sur  $m$ . Supposons la relation vraie pour  $m$  et  $m - 1$ . On a alors

$$\begin{aligned} V_n V_{m+1} &= aV_n V_m - V_n V_{m-1} = aV_{n+m} + aV_{n-m} - V_{n+m-1} - V_{n-m+1} \\ &= (aV_{n+m} - V_{n+m-1}) + (aV_{m-n} - V_{m-n-1}) \\ &= V_{n+m+1} + V_{m-n+1} = V_{n+m+1} + V_{n-(m+1)}. \end{aligned} \quad \square$$

**COROLLAIRE 3.26.** — On a pour tout  $n$  les relations

$$V_{2n-1} = V_n V_{n-1} - a,$$

$$V_{2n} = V_n^2 - 2,$$

$$V_{2n+1} = aV_n^2 - V_n V_{n-1} - a.$$

*Démonstration.* Les deux premières égalités sont les cas particuliers  $m = n-1$  et  $m = n$  de la proposition. La troisième s'en déduit, puisque  $V_{2n+1} = aV_{2n} - V_{2n-1}$ .  $\square$

Notons  $f(n)$  le couple  $(V_{n-1}, V_n) \in A^2$ . On a  $f(1) = (2, a)$  et on vient d'exprimer  $f(2n)$  et  $f(2n+1)$  en termes de  $f(n)$  : on a  $f(2n) = D(f(n))$  et  $f(2n+1) = E(f(n))$  avec

$$D(x, y) = (xy - a, y^2 - 2), \quad E(x, y) = (y^2 - 2, ay^2 - xy - a).$$

Utilisant la construction donnée en 1.2.2, on en déduit un *algorithme rapide de calcul* des suites de Lucas. Chaque pas D contient deux multiplications et

deux soustractions, chaque pas E contient trois multiplications (dont l'une a un facteur  $a$  fixe) et trois soustractions. Le nombre total d'opérations pour le calcul de  $V_n$  est au plus de  $3 \log n$  multiplications et autant d'additions.

EXERCICE 3.18. [N] — Calculer une table des polynômes  $V_n \in \mathbf{Z}[a]$ .

### 3.2.6. Construction d'anneaux par adjonction

On peut établir les résultats précédents d'une autre manière, par introduction d'un anneau auxiliaire, dans lequel  $a$  s'exprime effectivement sous la forme  $x + x^{-1}$ . Expliquons d'abord cette construction de façon plus générale.

Soit  $A$  un anneau et soit  $P(X)$  un polynôme unitaire de degré  $n$  à coefficients dans  $A$ . Écrivons  $P$  sous la forme

$$P(X) = X^n - R(X).$$

Dans l'anneau  $A[X]$ , on peut considérer la congruence modulo  $P$  : on a par définition  $U \equiv U' \pmod{P}$  si il existe  $Q \in A[X]$  avec  $U - U' = PQ$ . On peut traduire la proposition 1.39 de la façon suivante : tout polynôme est congru modulo  $P$  à un unique polynôme de degré  $< n$ .

On peut alors définir un « anneau-quotient », noté  $A[X]/(P)$ , en calquant la construction donnée dans le cas des entiers pour les anneaux  $\mathbf{Z}/n\mathbf{Z}$  (1.3.2). Les polynômes de degré  $< n$  forment un système de représentants des classes modulo  $P$ . On peut donc identifier cet anneau-quotient à l'ensemble des polynômes de degré  $< n$  ; cette identification respecte l'addition et la multiplication par les constantes ; le produit des classes représentées par  $U$  et  $U'$  est représenté par le reste de la division de  $UU'$  par  $P$ .

Une autre manière de dire essentiellement la même chose est la suivante : notons  $\xi = X \pmod{P}$  l'image de  $X$  dans l'anneau-quotient  $B = A[X]/(P)$ . On a  $P(\xi) = 0$ , ou encore  $\xi^n = R(\xi)$  et chaque élément de  $B$  s'écrit de manière unique  $a_0 + a_1\xi + \cdots + a_{n-1}\xi^{n-1}$ , avec des  $a_i \in A$ .

Cela donne une définition directe de l'anneau-quotient : on introduit un symbole  $\xi$  et on considère les expressions formelles

$$a_0 + a_1\xi + \cdots + a_{n-1}\xi^{n-1}, \quad a_i \in A.$$

On additionne de tels éléments de manière évidente terme à terme et on les multiplie en utilisant la relation  $\xi^n = R(\xi)$  pour éliminer les termes de degré  $\geq n$ .

On dit alors que l'anneau  $B$  est obtenu à partir de  $A$  par *adjonction de  $\xi$ , soumis à la relation  $\xi^n = R(\xi)$* . On notera que  $A$  est un sous-anneau de  $B$ .

REMARQUE 3.27. — Le fait que  $A$  soit un sous-anneau de  $B$ , c'est-à-dire que l'application évidente de  $A$  dans  $B$  soit injective, joue un rôle essentiel dans les applications. C'est en effet lui qui permet de vérifier une relation entre éléments de  $A$  « en se

plaçant dans B ». Il résulte de l'existence de la division euclidienne, qui fonctionne parce que le polynôme P est supposé unitaire (ou au moins de coefficient dominant inversible).

Nous utiliserons ci-dessous le cas  $n = 2$ . Écrivons P sous la forme  $P = X^2 - aX - b$ . Alors les éléments de B s'écrivent  $\alpha + \beta\xi$ , avec  $\alpha$  et  $\beta$  dans A et on a  $\xi^2 = a\xi + b$ ; par exemple, pour  $P = X^2 - b$ , on obtient la construction algébrique de l'anneau  $A(\sqrt{b})$ . Le cas de  $C = \mathbf{R}[X]/(X^2 + 1) = \mathbf{R}(\sqrt{-1})$ , avec  $i = X \bmod (X^2 + 1) \in C$ , est bien connu.

**EXERCICE 3.19. [B]** — Dans l'anneau  $A[X]/(X^2 - aX - b)$ , on considère la classe de  $X + c$ ; on obtient ainsi une nouvelle description de cet anneau sous la forme  $A[Y]/(Y^2 - a'Y - b')$ . Que valent  $a'$  et  $b'$ ? Que vaut  $a'^2 - 4b'$ ?

Revenons maintenant à nos suites de Lucas. Soient donc, comme plus haut, A un anneau et  $a$  un élément de A. Considérons l'anneau B obtenu par adjonction à A d'un élément  $x$  soumis à la relation  $x^2 = ax - 1$ . On a par construction dans B la relation  $x(a - x) = 1$ , ce qui montre que  $x$  est inversible dans B avec  $x + x^{-1} = a$ . On a par conséquent  $x^n + x^{-n} = V_n$ . Ainsi, pour démontrer une relation entre les  $V_n \in A$ , il suffit de la prouver dans B; en d'autres termes, lorsqu'on a à prouver une relation entre les  $V_n$ , on peut toujours supposer qu'il existe un élément  $x$  avec  $x + x^{-1} = a$ , donc avec  $x^n + x^{-n} = V_n$ .

**EXERCICE 3.20. [B]** — En utilisant la remarque ci-dessus, démontrer qu'on a  $V_n(V_m(a)) = V_{nm}(a)$ .

**EXERCICE 3.21. [C]** — Pour construire les polynômes  $V_n$ , il suffit en fait de considérer le cas de l'anneau de polynômes  $A = \mathbf{Z}[a]$ . Vérifier que l'on peut alors identifier B à  $\mathbf{Z}[x, x^{-1}] = \mathbf{Z}[x, y]/(xy - 1)$ . Cela donne dans l'anneau  $\mathbf{Z}[x, y]$  une relation  $x^n + y^n = V_n(x + y) + (xy - 1)W_n(x, y)$ . Comment calculer les  $W_n$ ?

### 3.2.7. Le critère de primalité de Lucas-Lehmer

Comme annoncé, nous allons établir ici le critère, dû à Lucas (1878) et Lehmer (1930), qui permet d'étudier la primalité d'un entier  $n$  lorsqu'on connaît la décomposition en facteurs premiers de  $n + 1$ .

Nous établirons d'abord un résultat auxiliaire. Fixons un nombre premier impair  $p$  et un entier  $a$ . Notons  $k$  le corps fini  $\mathbf{Z}/p\mathbf{Z}$  des classes modulo  $p$ , qui a  $p$  éléments. Adjoignons-lui, comme expliqué ci-dessus, un élément  $x$  soumis à la relation  $x^2 = ax - 1$ . Autrement dit, considérons l'anneau A formé des expressions  $u + vx$ , où  $u$  et  $v$  sont deux éléments de  $k$ , que l'on additionne et que l'on multiplie de façon naturelle, compte-tenu de la relation donnée  $x^2 = ax - 1$ . Notons que  $1$  désigne  $1_k$  et que le multiple  $ax$  de  $x$  est aussi le produit de  $x$  par la classe  $\bar{a} = a \cdot 1_k = a \bmod p$ .

La relation  $x^2 - \bar{a}x + 1_k = 0$  s'écrit aussi  $x(\bar{a} - x) = 1_k = 1_A$ . Il s'ensuit que  $x$  est inversible dans  $A$  et que l'on a  $x^{-1} = \bar{a} - x$ , donc

$$x + x^{-1} = \bar{a}.$$

Posons  $\Delta = a^2 - 4 \in \mathbf{Z}$ , de sorte que

$$(x - x^{-1})^2 = (x + x^{-1})^2 - 4 \cdot 1_k = (\bar{a})^2 - 4 \cdot 1_k = \Delta \bmod p.$$

**LEMME 3.28.** — *Supposons  $\Delta$  premier à  $p$ .*

a) *On a, ou bien  $\Delta^{(p-1)/2} \equiv 1 \pmod p$  et alors  $x^{p-1} = 1_A$ , ou bien  $\Delta^{(p-1)/2} \equiv -1 \pmod p$  et alors  $x^{p+1} = 1_A$ .*

b) *Pour entier  $m$ , les relations  $x^m = 1_A$  et  $x^m + x^{-m} = 2 \cdot 1_A$  sont équivalentes.*

*Démonstration.* Posons pour simplifier les notations  $\delta = \Delta \bmod p \in k$  et  $\varepsilon = \delta^{(p-1)/2} = \pm 1_k$ . Le petit théorème de Fermat implique  $\varepsilon^2 = \delta^{p-1} = 1_k$ , donc  $\varepsilon = \pm 1_k$ . On a  $(2x - \bar{a})^2 = (x - x^{-1})^2 = \delta$  ce qui, en élévant à la puissance  $(p-1)/2$ , donne  $\varepsilon = (2x - \bar{a})^{p-1}$ . Multipliant encore une fois par  $(2x - \bar{a})$  et utilisant 2.15 et le petit théorème de Fermat pour  $2 \cdot 1_k$  et  $\bar{a}$ , on obtient

$$\varepsilon(2x - \bar{a}) = (2x - \bar{a})^p = 2x^p - \bar{a}.$$

Si  $\varepsilon = 1_k$ , on obtient  $2x^p = 2x$ , donc  $x^p = x$  puisque 2 est inversible modulo  $p$ ,  $p$  étant impair. Si  $\varepsilon = -1_k$ , on obtient de même  $x^p = \bar{a} - x = x^{-1}$ . Cela prouve a).

Prouvons b). La relation  $x^m + x^{-m} = 2 \cdot 1_A$  équivaut à  $(x^m - 1_A)^2 = 0$ . Il suffit donc de prouver que dans l'anneau  $A$  la relation  $t^2 = 0$  implique  $t = 0$ . Si l'on pose  $t = u + vx$ , on a  $t^2 = u^2 + 2uvx + v^2x^2 = (u^2 - v^2) + (2uv + \bar{a}v^2)x$ . Il s'agit donc de voir que dans  $k$  le système des deux équations  $(u + v)(u - v) = 0$  et  $v(2u + \bar{a}v) = 0$  n'a comme solution que  $u = v = 0$ . Mais, puisqu'il s'agit d'un corps, la première relation donne  $u = v$  ou  $u = -v$ ; et la seconde  $v = 0$  ou  $2u + \bar{a}v = 0$ . On conclut aussitôt, étant donné que  $\bar{a} - 2 \cdot 1$  et  $\bar{a} + 2 \cdot 1$  sont inversibles, puisque leur produit vaut  $\delta$ , qui est inversible.  $\square$

**PROPOSITION 3.29** (Critère de primalité de Lucas-Lehmer). — *Soit  $n > 1$  un entier impair tel que l'on connaisse la décomposition de  $n + 1$  en facteurs premiers. Soit  $a$  un entier tel que  $n$  et  $a^2 - 4$  soient premiers entre eux, et soit  $V_n$  la suite (de Lucas) définie par*

$$V_0 = 2, \quad V_1 = a, \quad V_{i+1} = aV_i - V_{i-1}.$$

*Si  $V_{n+1} \equiv 2 \pmod n$  et si  $\text{pgcd}(V_{(n+1)/q} - 2, n) = 1$  pour tout facteur premier  $q$  de  $n + 1$ , alors  $n$  est premier.*

*Démonstration.* Soit  $p$  un facteur premier de  $n$ . Considérons l'anneau  $A$  et l'élément  $x \in A$  introduit ci-dessus, de sorte que  $x + x^{-1} = a \cdot 1_A$ . On a d'après le lemme  $x^{p\pm 1} = 1_A$ .

D'autre part, pour tout entier  $m > 0$ , la relation  $x^m = 1$  équivaut à  $x^m + x^{-m} = 2$ , donc d'après la définition même des suites de Lucas (voir 3.24) à  $V_m \equiv 2 \pmod{p}$ . L'hypothèse faite implique donc que  $x^{n+1} = 1_A$  et  $x^{(n+1)/q} \neq 1_A$  pour tout facteur premier  $q$  de  $n + 1$ . Mais cela signifie que l'ordre de  $x$  dans le groupe multiplicatif  $A^*$  est égal à  $n + 1$ . Puisque  $x^{p\pm 1} = 1$ , il en résulte que  $n + 1$  divise  $p \pm 1$ ; puisque  $p$  est au plus égal à  $n$ , cela implique  $n + 1 = p + 1$ , donc  $n = p$  et  $p$  est premier.  $\square$

Comme on l'a remarqué précédemment, on peut calculer  $V_m$  modulo  $n$  par un algorithme rapide, qui a un coût en  $C(\log m)(\log n)^2$ . Pour  $a$  fixé, et la décomposition en facteurs premiers de  $n + 1$  étant connue, la vérification du critère précédent a donc un coût en  $k(\log n)^3$ , où  $k$  est le nombre de facteurs premiers de  $n + 1$ , donc au plus en  $(\log n)^4$ .

**EXERCICE 3.22.** [C] — On remarquera la similitude de cet énoncé avec le critère de Lehmer. L'exercice consiste à le généraliser sur le modèle des critères de Pocklington et Lehmer-Pocklington.

### 3.2.8. Critère de primalité des nombres de Mersenne

Nous allons appliquer le critère précédent au cas du nombre de Mersenne  $n = 2^s - 1$ . On a  $n + 1 = 2^s$  et nécessairement  $q = 2$ . Il faut donc considérer les deux éléments  $V_{2^s}$  et  $V_{2^s-1}$ . Mais on a démontré dans le corollaire 3.26 la relation

$$V_{2^i} = V_{2^{i-1}}^2 - 2.$$

**COROLLAIRE 3.30** (Lucas). — Soient  $s$  et  $a$  deux entiers, avec  $s > 1$ . Définissons la suite  $(L_i)_{i \geq 1}$  par  $L_1 = a$ ,  $L_{i+1} = L_i^2 - 2$ . Supposons qu'on ait  $L_{s-1} \equiv 0 \pmod{2^s - 1}$ , et que  $a^2 - 4$  soit premier à  $2^s - 1$ . Alors le nombre de Mersenne  $2^s - 1$  est premier.

*Démonstration.* En effet, on a pour tout  $i$  la relation  $L_i = V_{2^i-1}$ . La relation donnée implique donc modulo  $2^s - 1$  la congruence  $V_{2^{s-2}} \equiv 0$ , d'où l'on tire  $V_{2^{s-1}} \equiv 0^2 - 2 = -2$ , puis  $V_{2^s} \equiv (-2)^2 - 2 = 2$ . On applique alors la proposition.  $\square$

Prenons par exemple  $s = 5$ , donc  $2^s - 1 = 31$ , et  $a = 4$ . On a successivement  $L_1 = 4$ ,  $L_2 = 14$ ,  $L_3 = 14^2 - 2 = 194 \equiv 8$  et  $L_4 \equiv 8^2 - 2 \equiv 0$ ; il s'ensuit que 31 est premier. Pour  $s = 11$ , donc  $2^s - 1 = 2047$ , on trouve modulo 2047 la suite 4, 14, 194, 788, 701, 119, -170, 240, 282, et en définitive,  $L_{10} \equiv -311$ . Le critère ne permet donc pas (heureusement !) de conclure à la primalité de 2047. Mieux, il implique sa non-primalité, car l'unique essai avec  $a = 4$  suffit à détecter tous les nombres de Mersenne premiers, en vertu du théorème suivant :

**THÉORÈME 3.31** (Lucas-Lehmer). — Soit  $s$  un entier impair  $> 1$ , et soit  $n = 2^s - 1$ . Définissons une suite  $(L_i)$  d'entiers modulo  $n$  par  $L_1 = 4$  et  $L_{i+1} \equiv L_i^2 - 2$ . Pour que  $n$  soit premier, il faut et il suffit qu'on ait  $L_{s-1} \equiv 0 \pmod{n}$ .

*Démonstration.* Notons d'abord qu'on a  $n \equiv 7 \pmod{12}$ ; en effet, on a  $n \equiv -1 \pmod{4}$  et  $n \equiv -2 \pmod{3}$ .

Pour  $a = 4$ , on a  $\text{pgcd}(a^2 - 4, n) = \text{pgcd}(12, n) = \text{pgcd}(12, 7) = 1$ . On peut donc appliquer le corollaire précédent et la condition est bien suffisante pour que  $n$  soit premier. Il nous reste à prouver la réciproque, et nous supposons désormais que  $n$  est premier.

Nous utiliserons le fait suivant, conséquence de la congruence  $n \equiv 7 \pmod{12}$  comme nous le verrons (remarque 5.15) :

$$a) \text{ on a } 3^{(n-1)/2} \equiv -1 \pmod{n},$$

ainsi que le suivant, qui résulte, lui, de la congruence  $n \equiv -1 \pmod{8}$  (voir proposition 5.16),

$$b) \text{ on a } 2^{(n-1)/2} \equiv 1 \pmod{n}.$$

Notons A l'anneau obtenu en adjoignant au corps  $k = \mathbf{Z}/n\mathbf{Z}$  un élément  $\beta$  tel que  $\beta^2 = 3 \cdot 1_k$ . Posons  $\alpha = 2 \cdot 1_k + \beta$ , de sorte que  $\alpha^2 - 4\alpha + 1_k = 0$ . Par définition de la suite de Lucas associée à  $a = 4$ , on a dans A

$$L_s \cdot 1_k = V_{2^{s-1}} \cdot 1_k = \alpha^{2^{s-1}} + \alpha^{2^{-(s-1)}}.$$

On a  $2\alpha = 4 \cdot 1_k + 2\beta = (1_k + \beta)^2$  et de même  $2\alpha^{-1} = 4 \cdot 1_k - 2\beta = (1_k - \beta)^2$ , ce qui implique (compte-tenu du fait que  $2 \cdot 2^{s-1} = n + 1$ )

$$2^{\frac{n+1}{2}} L_s \cdot 1_k = (1_k + \beta)^{n+1} + (1_k - \beta)^{n+1}.$$

Puisque  $\beta^2 = 3 \cdot 1_k$ , la formule du binôme donne (dans k c'est-à-dire modulo n) la relation

$$2^{\frac{n+1}{2}} L_s \equiv 2 \sum_{i=0}^{\frac{n+1}{2}} \binom{n+1}{2i} 3^i = 2 \sum_{i=0}^{\frac{n+1}{2}} \left( \binom{n}{2i} + \binom{n}{2i-1} \right) 3^i.$$

Puisque  $n$  est premier impair, seuls sont donc non nuls modulo  $n$  les termes de cette somme qui correspondent à  $i = 0$  et  $i = \frac{n+1}{2}$ . On obtient en définitive modulo  $n$

$$2^{\frac{n+1}{2}} L_s \equiv 2(1 + 3^{\frac{n+1}{2}}).$$

D'après a) et b) ci-dessus, on a modulo  $n$  les deux congruences  $2^{\frac{n+1}{2}} \equiv 2$  et  $3^{\frac{n+1}{2}} \equiv -3$ , ce qui donne en définitive  $L_s \equiv -2 \pmod{n}$ . Puisque  $L_s = L_{s-1}^2 - 2$ , cela implique bien  $L_{s-1} \equiv 0 \pmod{n}$ , ce qu'on voulait démontrer.  $\square$

Le calcul récursif de  $L_{s-1}$  demande  $s - 2$  élévations au carré modulo  $2^s - 1$ . Le coût total est donc en  $s^3$ . L'efficacité de ce critère (et de ses variantes) explique pourquoi les nombres de Mersenne premiers (et les nombres analogues de la forme  $r \cdot 2^s - 1$  avec  $r$  petit) fournissent les plus grands nombres premiers connus. Voici une rédaction possible en Ruby :

PROGRAMME 3.32. — *Test de Lucas pour les nombres de Mersenne*

```

def test_Lucas(s)
    # returns true if  $2^s - 1$  is prime
    n = 2**s - 1
    l = 4
    (s-2).times {l = (l*l - 2) % n}
    l == 0
end
[61, 67, 107, 257, 1279].collect{|s| test_Lucas(s)}
# => [true, false, true, false, true]

```

EXERCICE 3.23. [N] — Déterminer les premiers nombres de Mersenne premiers.

### § 3.3. Indicateur et nombres de Carmichael

#### 3.3.1. Nombres de Carmichael

PROPOSITION 3.33 (Carmichael). — *Soit  $n$  un entier  $> 1$ . Les conditions suivantes sont équivalentes :*

- (i)  *$n$  est sans facteur multiple, et  $p - 1$  divise  $n - 1$  pour tout facteur premier  $p$  de  $n$  ;*
- (ii) *on a  $a^n \equiv a \pmod{n}$  pour tout entier  $a$  ;*
- (iii) *on a  $a^{n-1} \equiv 1 \pmod{n}$  pour tout entier  $a$  premier à  $n$ .*

Démonstration. L’implication (i)  $\Rightarrow$  (ii) résulte directement du lemme 2.18. L’assertion (ii)  $\Rightarrow$  (iii) est claire : puisque  $a \pmod{n}$  est inversible, on peut simplifier la congruence par  $a$ . Il reste à prouver que (iii) implique (i). Supposons donc la condition (iii) satisfaite. Prouvons d’abord que  $n$  est sans facteur multiple. Raisonnons par l’absurde en considérant un nombre premier  $p$  tel que  $p^2$  divise  $n$ , donc avec  $n = p^2m$ . Posons  $a = 1 + pm$  ; par la formule du binôme, on a  $a^p \equiv 1 \pmod{n}$  ; l’ordre de  $a$  modulo  $n$  est donc  $p$  (il ne peut être égal à 1) ; comme  $p$  ne divise pas  $n - 1$ , cela contredit (iii), et on a bien prouvé que  $n$  est sans facteur multiple. Écrivons donc  $n$  comme produit de nombres premiers distincts  $p_1, \dots, p_m$ . Soit  $a_i$  une racine primitive modulo  $p_i$  (corollaire 3.9). D’après le théorème chinois, il existe  $a$  avec  $a \equiv a_i \pmod{p_i}$  pour tout  $i$ . D’après (iii),  $a^{n-1}$  est congru à 1 modulo  $n$ , donc modulo  $p_i$ . Cela donne  $(a_i)^{n-1} \equiv 1 \pmod{p_i}$ , donc  $n - 1$  est un multiple de  $p_i - 1$ , vu le choix de  $a_i$ . Ceci prouve (i) et achève la démonstration.  $\square$

Les nombres premiers satisfont évidemment aux conditions précédentes.

DÉFINITION 3.34. — *On dit que l’entier  $n > 1$  est un nombre de Carmichael s’il est composé et s’il satisfait aux conditions de la proposition précédente.*

Les nombres de Carmichael sont donc ceux qui, bien que composés, réussissent le test de primalité évident tiré du théorème de Fermat.

EXERCICE 3.24. [A] — Montrer que l’entier  $3 \times 11 \times 17$  est un nombre de Carmichael.

**EXERCICE 3.25. [B]** — Soit  $n$  un entier impair  $> 1$ . Prouver que les conditions suivantes sont équivalentes :

(i)  $n$  est sans facteur multiple, et  $p - 1$  divise  $(n - 1)/2$  pour tout facteur premier  $p$  de  $n$  ;

(ii) on a  $a^{(n-1)/2} \equiv 1 \pmod{n}$  pour tout entier  $a$  premier à  $n$ .

**EXERCICE 3.26. [B]** — Soit  $m$  un entier  $> 0$  tel que les trois entiers  $6m + 1$ ,  $12m + 1$ ,  $18m + 1$  soient premiers. Alors  $n = (6m + 1)(12m + 1)(18m + 1)$  est un nombre de Carmichael. Il vérifie même les conditions de l'exercice précédent lorsque  $m$  est impair. Application :  $n = 7 \times 13 \times 19 = 1729$ .

**EXERCICE 3.27. [N]** — Trouver les premières valeurs de  $m$  satisfaisant aux conditions de l'exercice précédente.

Le plus petit des nombres de Carmichael est 561. On sait (1992) qu'il en existe une infinité. Un nombre de Carmichael possède un facteur premier inférieur à sa racine cubique :

**PROPOSITION 3.35.** — *Tout nombre de Carmichael est un produit de nombres premiers impairs distincts, en nombre  $\geq 3$ .*

*Démonstration.* Soit  $n$  un nombre de Carmichael. C'est un produit de facteurs premiers distincts, en nombre au moins égal à 2. Si  $n = pq$  avec  $p < q$ , où  $p$  et  $q$  sont premiers, alors  $q - 1$  doit diviser  $pq - 1 = p(q - 1) + (p - 1)$ , donc doit diviser  $p - 1$ , ce qui est impossible. Enfin, puisque  $n$  est composé, on a  $n \geq 4$ , donc  $-1 \not\equiv 1 \pmod{n}$ ; mais la relation  $(-1)^n \equiv -1 \pmod{n}$  implique alors que  $n$  est impair.  $\square$

**EXERCICE 3.28. [C]** — On considère les nombres de Carmichael de la forme  $pqr$  où  $p$ ,  $q$  et  $r$  sont premiers. Montrer que si  $r$  est donné, il n'existe qu'un nombre fini de tels nombres. Trouvez tous ceux pour lesquels  $r = 3$  ou  $r = 5$ .

### 3.3.2. L'indicateur de Carmichael

Soit  $n > 1$  un entier. Le théorème d'Euler (proposition 2.17) dit qu'on a  $a^{\varphi(n)} \equiv 1 \pmod{n}$  pour tout entier  $a$  premier à  $n$ .

On peut améliorer cet énoncé : notons  $\lambda(n) = \omega((\mathbf{Z}/n\mathbf{Z})^*)$  l'exposant du groupe  $(\mathbf{Z}/n\mathbf{Z})^*$ , qui est par définition le ppcm des ordres des éléments de ce groupe. On a donc la caractérisation suivante :

- ▷ on a  $a^{\lambda(n)} \equiv 1 \pmod{n}$  pour tout entier  $a$  premier à  $n$  ;
- ▷ tout entier  $m$  tel qu'on ait  $a^m \equiv 1 \pmod{n}$  pour tout entier  $a$  premier à  $n$  est multiple de  $\lambda(n)$ .

**EXERCICE 3.29. [A]** — Sauf pour  $n = 1$  et  $n = 2$ ,  $\lambda(n)$  est toujours pair.

On dit que  $\lambda$  est l'*indicateur de Carmichael*. Les propriétés suivantes résultent de ce qu'on a déjà vu, et notamment de la proposition 3.3 :

- ▷  $\lambda(n)$  divise  $\varphi(n)$  ;
- ▷ il existe dans  $(\mathbf{Z}/n\mathbf{Z})^*$  un élément d'ordre  $\lambda(n)$  ;

- ▷ pour qu'on ait  $\lambda(n) = \varphi(n)$ , il faut et il suffit que le groupe  $(\mathbf{Z}/n\mathbf{Z})^*$  soit cyclique ;
- ▷ si  $n$  est premier, on a  $\lambda(n) = \varphi(n) = n - 1$  ;
- ▷ pour que le nombre composé  $n$  soit un nombre de Carmichael, il faut et il suffit que  $\lambda(n)$  divise  $n - 1$ .

On peut calculer  $\lambda(n)$  en décomposant  $n$  en facteurs premiers :

**PROPOSITION 3.36.** — *Si  $m$  et  $n$  sont premiers entre eux,  $\lambda(mn)$  est le ppcm de  $\lambda(m)$  et de  $\lambda(n)$ .*

*Démonstration.* D'après le théorème chinois, la condition  $a^r \equiv 1 \pmod{mn}$  équivaut à la conjonction des deux conditions  $a^r \equiv 1 \pmod{m}$  et  $a^r \equiv 1 \pmod{n}$ . La proposition résulte alors directement de la définition de la fonction  $\lambda$ .  $\square$

Ainsi, il suffit de calculer les  $\lambda(p^n)$  où  $p$  est premier, ce que nous ferons plus tard (proposition 3.43). On peut déjà remarquer :

**PROPOSITION 3.37.** — *Soit  $n$  un entier  $> 1$  sans facteur multiple. Alors  $\lambda(n)$  est le ppcm des  $p - 1$ , lorsque  $p$  parcourt les facteurs premiers de  $n$ . De plus, on a  $a^{\lambda(n)+1} \equiv a \pmod{n}$  pour tout entier  $a$ .*

*Démonstration.* La première assertion résulte de la proposition 3.36, et la seconde du lemme 2.18.  $\square$

**EXERCICE 3.30. [A]** — Donner un exemple d'entiers  $a$  et  $n$  pour lesquels on n'a pas  $a^{\lambda(n)+1} \equiv a \pmod{n}$ .

### 3.3.3. Structure du groupe $(\mathbf{Z}/p^n\mathbf{Z})^*$ , $p$ premier impair

Dans les deux lemmes qui suivent, on désigne par  $p$  un nombre premier impair.

**LEMME 3.38.** — *Soient  $n$  un entier positif et  $a$  un entier.*

- On a  $(1 + p^n a)^p \equiv 1 + p^{n+1} a \pmod{p^{n+2}}$ .*
- On a  $(1 + pa)^{p^n} \equiv 1 + p^{n+1} a \pmod{p^{n+2}}$ .*

*Démonstration.* On a

$$(1 + p^n a)^p = 1 + pp^n a + \binom{p}{2} p^{2n} a^2 + p^{3n} (\dots),$$

et on remarque que  $\binom{p}{2}$  est divisible par  $p$  et que l'on a  $2n + 1 \geq n + 2$  et  $3n \geq n + 2$ . Cela prouve a). On démontre b) par récurrence sur  $n$ . Supposons en effet que l'on ait  $(1 + pa)^{p^{n-1}} = 1 + p^n a + p^{n+1} b$ . On en déduit en appliquant a)

$$(1 + pa)^{p^n} = (1 + p^n(a + pb))^p \equiv 1 + p^{n+1}(a + pb) \pmod{p^{n+2}},$$

ce qui démontre b).  $\square$

**EXERCICE 3.31.** [B] — Où a servi l'hypothèse  $p \neq 2$  ?

**LEMME 3.39.** — Soient  $n \geq 2$  un entier et  $u$  un entier congru à 1 modulo  $p$ . Alors  $u^{p^{n-1}} \equiv 1 [\text{mod } p^n]$ . Pour que  $u \bmod p^n$  soit un élément d'ordre  $p^{n-1}$  du groupe  $(\mathbf{Z}/p^n\mathbf{Z})^*$ , il faut et il suffit qu'on ait  $u \not\equiv 1 [\text{mod } p^2]$ .

*Démonstration.* Posons  $u = 1 + pa$ . D'après le lemme 3.38 b) (pour la valeur  $n - 1$ ), on a  $u^{p^{n-1}} \equiv 1 [\text{mod } p^n]$  et l'ordre de  $u \bmod p^n$  divise  $p^{n-1}$ . Dire qu'il lui est égal, c'est dire qu'on a  $u^{p^{n-2}} \not\equiv 1 [\text{mod } p^n]$ . Or le même lemme (pour la valeur  $n - 2$ ) dit que cette condition équivaut à  $a \not\equiv 0 [\text{mod } p]$ , c'est-à-dire à  $u \not\equiv 1 [\text{mod } p^2]$ .  $\square$

**PROPOSITION 3.40.** — Soient  $p$  un nombre premier impair et  $n \geq 2$  un entier. Le groupe  $(\mathbf{Z}/p^n\mathbf{Z})^*$  est cyclique, d'ordre  $p^n - p^{n-1}$ .

*Démonstration.* Notons  $G$  ce groupe. Il est d'ordre  $\varphi(p^n) = p^n - p^{n-1} = p^{n-1}(p-1)$ . Puisque  $p^{n-1}$  et  $p-1$  sont premiers entre eux, il suffit, compte tenu du lemme 3.2, de prouver qu'il possède un élément d'ordre  $p^{n-1}$  et un élément d'ordre multiple de  $p-1$ . La première assertion est claire, car l'élément  $1 + p$  convient, vu le lemme précédent. Par ailleurs, il existe des entiers  $a$  d'ordre  $p-1$  modulo  $p$  (corollaire 3.9), donc d'ordre multiple de  $p-1$  modulo  $p^n$ .  $\square$

### 3.3.4. Structure du groupe $(\mathbf{Z}/2^n\mathbf{Z})^*$

Nous nous proposons de déterminer la structure du groupe  $(\mathbf{Z}/2^n\mathbf{Z})^*$ . Son ordre est  $2^n - 2^{n-1} = 2^{n-1}$ .

Le groupe  $(\mathbf{Z}/2\mathbf{Z})^*$  est réduit à l'élément neutre. Si  $n > 1$ , les deux éléments 1 et  $-1$  forment un sous-groupe cyclique d'ordre 2 de  $(\mathbf{Z}/2^n\mathbf{Z})^*$ , auquel il se réduit si  $n = 2$ .

**EXERCICE 3.32.** [A] — Le groupe  $(\mathbf{Z}/8\mathbf{Z})^*$  est un « Vierergruppe » : il possède quatre éléments, les trois éléments non neutres sont de carré 1 et chacun est le produit des deux autres.

Supposons  $n \geq 3$  et notons  $U(n)$  l'ensemble des classes modulo  $2^n$  des entiers congrus à 1 modulo 4. C'est un sous-groupe de  $(\mathbf{Z}/2^n\mathbf{Z})^*$ , d'ordre  $2^{n-2}$ . Tout élément de  $(\mathbf{Z}/2^n\mathbf{Z})^*$  est congru à 1 ou  $-1$  modulo 4, donc s'écrit  $\pm a$ , avec  $a \in U(n)$ . Ainsi  $(\mathbf{Z}/2^n\mathbf{Z})^*$  s'identifie au produit du groupe  $\{-1, +1\}$  et du groupe  $U(n)$ .

**PROPOSITION 3.41.** — Soit  $n$  un entier  $\geq 3$ . Le groupe  $U(n)$  est cyclique, d'ordre  $2^{n-2}$  et engendré par la classe de 5.

*Démonstration.* Il suffit de vérifier que  $5^{2^{n-3}} \equiv 1 + 2^{n-1} [\text{mod } 2^n]$ , ce qui montrera que l'ordre de  $5 \bmod 2^n$  ne divise pas  $2^{n-3}$ , donc est égal à  $2^{n-2}$ . Cette vérification se fait immédiatement par récurrence sur  $n$ .  $\square$

**EXERCICE 3.33. [B]** — Faire la vérification. Généraliser en remplaçant 5 par un entier congru à 1 modulo 4.

**EXERCICE 3.34. [B]** — Démontrer que la classe d'un entier congru à 1 modulo 8 est un carré dans le groupe  $U(n)$ .

**COROLLAIRE 3.42.** — Pour  $n > 2$ , l'ordre maximal d'un élément de  $(\mathbf{Z}/2^n\mathbf{Z})^*$  est  $2^{n-2}$ . Soient  $u$  un entier impair et  $n$  un entier  $> 3$ . Pour que  $u$  soit d'ordre  $2^{n-2}$  modulo  $2^n$ , il faut et il suffit que  $u$  soit congru à 3 ou 5 modulo 8.

*Démonstration.* Puisque la classe de 5 engendre le groupe  $U(n)$ , on a une congruence  $u \equiv \varepsilon 5^a \pmod{2^n}$ , avec  $\varepsilon = \pm 1$ , pour un entier  $a$  convenable. On a  $u^{2^{n-2}} \equiv 1$ , et  $u^{2^{n-3}} \equiv (5^a)^{2^{n-3}}$ . Pour qu'on ait  $u^{2^{n-3}} \not\equiv 1$ , il faut et il suffit que la classe de  $5^a$  soit un générateur de  $U(n)$ , c'est-à-dire que  $a$  soit impair. Mais modulo 8, on a  $5^{2b} \equiv 25^b \equiv 1^b \equiv 1$ , tandis que  $5^{2b+1} \equiv 5 \cdot 25^b \equiv 5$ . La condition «  $a$  est impair » équivaut donc à  $5^a \equiv 5 \pmod{8}$ , c'est-à-dire à  $u \equiv \pm 5 \pmod{8}$ .  $\square$

Les cas non couverts par la proposition précédente sont les suivants :

- ▷  $(\mathbf{Z}/2\mathbf{Z})^*$  est réduit à la classe de 1.
- ▷  $(\mathbf{Z}/4\mathbf{Z})^*$  a deux éléments. La classe de 3 est d'ordre 2.
- ▷  $(\mathbf{Z}/8\mathbf{Z})^*$  a quatre éléments. Les classes de 3, 5 et 7 sont d'ordre 2.

### 3.3.5. Calcul de l'indicateur de Carmichael

On peut résumer les résultats obtenus :

**PROPOSITION 3.43.** — a) Si  $m$  et  $n$  sont premiers entre eux,  $\lambda(mn)$  est le ppcm de  $\lambda(m)$  et de  $\lambda(n)$ .

b) Si  $p$  est un nombre premier impair, on a  $\lambda(p^r) = \varphi(p^r) = p^{r-1}(p-1)$  pour tout  $r \geq 1$ .

c) On a  $\lambda(2) = 1$ ,  $\lambda(4) = 2$  et  $\lambda(2^r) = 2^{r-2} = \varphi(2^r)/2$  pour tout entier  $r \geq 3$ .

*Démonstration.* Ces trois énoncés résultent respectivement de la proposition 3.36, de la proposition 3.40 et du corollaire 3.42.  $\square$

Cette proposition permet de calculer la valeur de l'indicateur de Carmichael pour un entier que l'on sait décomposer en facteurs premiers.

**EXERCICE 3.35. [A]** — Le groupe  $(\mathbf{Z}/n\mathbf{Z})^*$  est un produit de groupes cycliques tous d'ordre pair. Quel en est le nombre ?

**EXERCICE 3.36. [B]** — (Suite du précédent.) Combien le groupe  $(\mathbf{Z}/n\mathbf{Z})^*$  a-t-il d'éléments d'ordre 2 ? Pour que le groupe  $(\mathbf{Z}/n\mathbf{Z})^*$  soit cyclique, il faut et il suffit que la congruence  $a^2 \equiv 1 \pmod{n}$  implique  $a \equiv \pm 1 \pmod{n}$ .

**EXERCICE 3.37. [B]** — Les entiers  $n$  tels que le groupe  $(\mathbb{Z}/n\mathbb{Z})^*$  soit cyclique, c'est-à-dire tels que  $\lambda(n) = \varphi(n)$ , sont 1, 2, 4, et les  $p^r$  et  $2p^r$  où  $p$  est un nombre premier impair.

**EXERCICE 3.38. [C]** — (Suite du précédent.) Lorsque ce groupe est cyclique, caractériser ses générateurs.

En calquant le programme 2.2.2, on obtient (`lambda` est un mot-clé de Ruby, donc nous choisissons un autre nom) :

**PROGRAMME 3.44. — Calcul de l'indicateur de Carmichael** ——————

```
def carmichael(max)
  carmichael = Array.new
  carmichael[0], carmichael[1], p = 1, 1, 2
  until p.nil? do
    done = Array.new
    carmichael.each_with_index do |val, n|
      next if n == 0 || val.nil? || done[n]
      pp = p
      until (nn = n * pp) > max
        done[nn] = true
        carm_pp = pp - pp/p
        carm_pp = carm_pp/2 if p == 2 && pp > 4
        carmichael[nn] = ppcm(val, carm_pp)
        pp = pp * p
      end
    end
    p = carmichael.index(nil) # premier suivant
  end
  carmichael[0] = nil
  carmichael
end
carmichael(561).last # => 80 # notons que 80 divise 560.
```

---

### 3.3.6. Preuve du théorème de Rabin

Nous nous proposons dans ce paragraphe de démontrer le théorème 2.24.

#### Puissances dans un groupe cyclique

**LEMME 3.45.** — Soit  $G$  un groupe cyclique d'ordre  $r$ , d'élément neutre noté  $e$ . Soient  $m$  un entier et  $z$  un élément de  $G$ . Notons  $k$  le pgcd de  $m$  et  $r$ . Pour que l'équation  $x^m = z$  ait des solutions dans  $G$ , il faut et il suffit que  $z^{r/k} = e$ . S'il en est ainsi, elle a  $k$  solutions.

*Démonstration.* Notons  $g$  un générateur de  $G$ , de sorte que  $z$  s'écrit  $g^i$  et qu'il s'agit de déterminer les entiers  $j$  définis modulo  $r$  tels que  $g^{mj} = g^i$ , c'est-à-dire  $mj \equiv i \pmod{r}$ . Si on pose  $m = km'$  et  $r = kr'$ , la relation  $mj \equiv i \pmod{r}$  s'écrit aussi  $km'j \equiv i \pmod{kr'}$ ; elle ne peut avoir de solution que si  $i$  est divisible par  $k$ , ce qui signifie que  $ir'$  est divisible par  $kr' = r$ , ou encore que  $z^{r'} = e$ . Supposons qu'il en soit ainsi, et posons  $i = ki'$ . On doit donc résoudre  $m'j \equiv i' \pmod{r'}$ . Puisque  $m'$

et  $r'$  sont premiers entre eux, il existe des entiers  $u$  et  $v$  avec  $um' + vr' = 1$ , et la relation précédente équivaut à  $j \equiv ui \pmod{r'}$ . Ainsi  $j$  est déterminé modulo  $r'$  et il y a  $r/r' = k$  solutions.  $\square$

## Un résultat préliminaire

Supposons donnés un entier pair  $m$ , un groupe commutatif  $G$  et un élément  $\alpha$  d'ordre 2 de  $G$ . Ecrivons  $m = 2^s t$  avec  $t$  impair et  $s \geq 1$ .

Considérons les  $s + 1$  équations suivantes dans  $G$  :

$$\begin{aligned} x^t &= 1, & (C_0(G, \alpha)) \\ x^t &= \alpha, & (C_1(G, \alpha)) \\ &\dots \\ x^{2^{j-1}t} &= \alpha, & (C_j(G, \alpha)) \\ &\dots \\ x^{2^{s-1}t} &= \alpha. & (C_s(G, \alpha)) \end{aligned}$$

Supposons d'abord que  $G$  soit un *groupe cyclique d'ordre pair*  $\phi$  et écrivons  $\phi = 2^u v$  avec  $v$  impair et  $u \geq 1$ .

**LEMME 3.46.** — Posons  $r = \min(u, s)$  et soit  $w$  le pgcd de  $t$  et  $v$ . L'équation  $C_0(G, \alpha)$  a  $w$  solutions. Pour  $1 \leq j \leq r$ , l'équation  $C_j(G, \alpha)$  a  $2^{j-1}w$  solutions. Pour  $j \geq r$ , l'équation  $C_j(G, \alpha)$  n'a pas de solution.

*Démonstration.* Traitons le cas  $1 \leq j \leq s$ , le cas  $j = 0$  étant analogue (et plus simple). Posons  $k = \text{pgcd}(\phi, 2^{j-1}t)$ . Pour que  $C_j(G, \alpha)$  ait des solutions, il faut et il suffit d'après le lemme précédent que  $\alpha^{\phi/k} = 1$ . Or  $k = \text{pgcd}(2^u v, 2^{j-1}t) = 2^{\min(u, j-1)} w$ . La condition signifie que  $\phi/k$  est pair, ce qui se traduit par  $\min(u, j-1) < u$ , soit  $j-1 < u$ , et en définitive  $j \leq r$ . On a alors  $k = 2^{j-1}w$ , et on a  $k$  solutions d'après le lemme.  $\square$

## La démonstration

Venons-en maintenant à la démonstration du théorème de Rabin. On considère donc un entier impair composé  $n > 1$ , et on écrit  $n - 1 = 2^s t$  avec  $t$  impair. On considère les  $s + 1$  équations  $C_j(G, -1)$ , où  $G$  est le groupe multiplicatif  $(\mathbb{Z}/n\mathbb{Z})^*$ . On va prouver l'assertion suivante, légèrement plus précise que le théorème annoncé :

**PROPOSITION 3.47.** — a) Si  $n = p^a$  a un seul facteur premier, les  $s + 1$  équations précédentes ont au total  $p - 1 = \varphi(n)/p^{a-1}$  solutions.

b) Si  $n$  a plus d'un facteur premier, les  $s + 1$  équations précédentes ont au total au plus  $\varphi(n)/4$  solutions.

*Démonstration.* Soit

$$n = \prod_{i=1}^N p_i^{a_i}$$

la décomposition de  $n$  en facteurs premiers. Par le théorème chinois, le groupe  $G$  se décompose en le produit des groupes  $G_i = (\mathbf{Z}/p_i^{a_i} \mathbf{Z})^*$ . Pour  $j = 0, \dots, s$ , le nombre des solutions de  $C_j(G, -1)$  est le produit du nombre des solutions de chacune des  $C_j(G_i, -1)$ . Par ailleurs, chacun des  $G_i$  est cyclique, d'ordre  $\phi_i = p_i^{a_i-1}(p_i - 1)$  (proposition 3.40). On a  $\varphi(n) = \prod \phi_i$ . Il ne reste plus qu'à appliquer ce qui précède.

Posons donc  $\phi_i = 2^{u_i} v_i$ , avec  $v_i$  impair,  $w_i = \text{pgcd}(t, v_i)$  et  $v'_i = v_i/w_i$ . On a  $\varphi(n) = 2^U V$  avec  $U = \sum u_i$  et  $V = \prod v_i$ . On posera pour simplifier  $V' = \prod v'_i$ .

Par ailleurs, pour que  $C_j(G, -1)$  ait des solutions, il faut que  $j$  soit inférieur à chacun des  $u_i$ . Posons donc  $u_{\min} = \inf_i(u_i)$  et  $r = \inf(u_{\min}, s)$ . L'équation  $C_j(G, -1)$  n'a donc de solutions que si l'on a  $j \leq r$ . Pour  $j = 0$ , le nombre de ses solutions est

$$A_0 = \prod_i w_i = V/V' ;$$

pour  $1 \leq j \leq r$ , c'est

$$A_j = \prod_i 2^{j-1} w_i = 2^{N(j-1)} V/V'.$$

Le nombre total des solutions est donc

$$A = \sum_j A_j = (1 + 1 + 2^N + \dots + 2^{N(r-1)}) V/V'.$$

Distinguons maintenant deux cas, suivant que  $n$  possède un ou plusieurs facteurs premiers.

a) Cas  $N = 1$ . On a donc  $n = p^a$ ,  $n - 1 = p^a - 1 = 2^s t$ ; posons  $p - 1 = 2^u w$ , avec  $w$  impair. Puisque  $p - 1$  divise  $n - 1$ , on a  $u \leq s$ , donc  $\inf(u, s) = u$ . Par ailleurs, on a  $\varphi(n) = p^{a-1}(p - 1) = 2^u v$ , avec  $v = p^{a-1}w$  impair. Notons que  $w$  divise  $p - 1$ , donc aussi  $p^a - 1$ , donc aussi  $t$ , tandis que  $p^{a-1}$  est premier à  $n - 1$  donc aussi à  $t$ . Il en résulte que le pgcd de  $t$  et  $v$  est  $w$ . La relation établie ci-dessus, avec  $N = 1$ ,  $r = u$  et  $V/V' = w$  donne  $A = (1 + 1 + \dots + 2^{u-1})w = 2^u w = p - 1$ .

b) Supposons maintenant  $N > 1$ , et revenons à la formule donnant  $A$ . La somme se majore par  $2^{N(r-1)+1} = 2^{Nr} 2^{1-N}$ . Puisqu'on a  $\varphi(n) = 2^U V$ , on en déduit

$$\frac{\varphi(n)}{A} \geq 2^{U-Nr} 2^{N-1} V'.$$

Mais, puisque chaque  $u_i$  est par définition  $\geq r$ , on a  $U = \sum u_i \geq Nr$ . Notons par ailleurs que  $t$  divise  $n - 1$ , donc est premier avec chacun des  $p_i$ ; si on a  $a_i > 1$ , le facteur  $p_i^{a_i-1}$  de  $v_i$  n'apparaît donc pas dans  $w_i$ , donc apparaît dans  $v'_i$ . Ainsi,  $V'$  est divisible par chacun des  $p_i$  pour lesquels  $a_i$  est  $> 1$ . On voit ainsi que  $\varphi(n)/A$  ne peut être  $< 4$  que si l'on a simultanément  $N = 2$ ,  $a_1 = a_2 = 1$ , et  $U = 2r$ , donc  $u_1 = u_2 = r$ , et aussi  $V' = 1$ , donc  $v'_1 = v'_2 = 1$ . Résumons : on a

$$n = p_1 p_2, \quad n - 1 = 2^s t, \quad p_1 - 1 = 2^r v_1, \quad p_2 - 1 = 2^r v_2,$$

où  $v_1$  et  $v_2$  divisent  $t$  et où on a  $r \leq s$ . Mais cela implique que  $p_1 - 1$  et  $p_2 - 1$  divisent  $n - 1$ , ce qui est impossible : on a en effet

$$n - 1 = p_1 p_2 - 1 = (p_1 - 1)(p_2 - 1) + (p_1 - 1) + (p_2 - 1),$$

et il en résulterait que  $p_1 - 1$  divise  $p_2 - 1$  et réciproquement.  $\square$

REMARQUE 3.48. — La démonstration précédente montre que le nombre des solutions est en général très inférieur à  $\varphi(n)/4$ .

EXERCICE 3.39. [C] — Quels sont les nombres composés pour lesquels on a exactement  $\varphi(n)/4$  solutions ?



# Chapitre 4

## Transformation de Fourier rapide

Profitant de l'introduction des racines primitives dans le chapitre précédent, nous faisons dans ce chapitre une petite diversion en montrant comment l'adaptation au calcul modulaire de la transformation de Fourier rapide permet de construire des algorithmes rapides de multiplication des grands entiers.

### § 4.1. Transformation de Fourier discrète

#### 4.1.1. Racines principales de l'unité

Pour des applications aux anneaux  $\mathbf{Z}/N\mathbf{Z}$  qui peuvent avoir des diviseurs de zéro, nous avons besoin de renforcer un peu la notion de racine primitive de l'unité.

Soient  $A$  un anneau,  $n$  un entier  $> 0$  et  $\omega$  une racine  $n$ -ième de l'unité dans  $A$ . Les conditions suivantes sont équivalentes :

(i) pour tout entier  $i$  avec  $0 < i < n$ , l'élément  $1 - \omega^i$  n'est pas diviseur de zéro dans  $A$  ;

(ii) pour tout couple d'entiers  $i$  et  $j$  non congrus modulo  $n$ , l'élément  $\omega^i - \omega^j$  n'est pas diviseur de zéro dans  $A$ .

En effet,  $\omega$  est inversible et on a  $\omega^i - \omega^j = \omega^i(1 - \omega^{j-i})$ .

Sous ces conditions on dira que  $\omega$  est une *racine principale*  $n$ -ième de l'unité. En particulier,  $\omega^i$  est différent de 1 pour  $0 < i < n$  et  $\omega$  est une racine primitive  $n$ -ième de l'unité. Inversement, si l'anneau  $A$  est intègre, tout racine primitive de l'unité est une racine principale. Par exemple,  $e^{2\pi i/n}$  est une racine principale  $n$ -ième de l'unité dans  $\mathbf{C}$ .

**EXERCICE 4.1. [B]** — Quelles sont les racines principales secondes de l'unité ?

**EXERCICE 4.2. [A]** — Montrer que, si  $\omega \in A$  est une racine principale  $n$ -ième de l'unité et si  $n = n'n''$ , alors  $\omega^{n'}$  est une racine principale  $n''$ -ième de l'unité.

**PROPOSITION 4.1.** — Soit  $\omega$  une racine principale  $n$ -ième de l'unité dans  $A$ .

a) On a dans  $A[X]$  la relation

$$X^n - 1_A = \prod_{i=0}^{n-1} (X - \omega^i).$$

b) L'élément  $n \cdot 1_A$  n'est pas diviseur de zéro dans  $A$  et on a

$$n \cdot 1_A = \prod_{i=1}^{n-1} (1 - \omega^i).$$

*Démonstration.* Démontrons a). Il suffit de reprendre la démonstration usuelle (voir la proposition 3.4). Posons  $P(X) = X^n - 1$ . On a  $P(1) = 0$ , donc  $P(X)$  s'écrit  $(X - 1)Q(X)$ , avec  $Q$  unitaire. On a  $P(\omega) = 0$ , donc  $(\omega - 1)Q(\omega) = 0$ . Mais  $\omega - 1$  ne divise pas zéro, donc  $Q(\omega) = 0$  et  $Q(X)$  s'écrit  $(X - \omega)R(X)$ . Ainsi  $P$  s'écrit  $(X - 1)(X - \omega)R(X)$ , avec  $R$  unitaire. Donc  $(\omega^2 - 1)(\omega^2 - \omega)R(\omega^2) = 0$ , etc. Cela prouve a). Si on simplifie les deux membres par  $X - 1$  et qu'on fait  $X = 1$ , on trouve la relation de b), et il en résulte que  $n \cdot 1_A$  est produit de non diviseurs de zéro.  $\square$

#### 4.1.2. L'anneau $A[X]/(X^n - 1)$

On peut représenter l'anneau quotient  $A[X]/(X^n - 1)$  comme l'ensemble des polynômes de degré  $< n$ , muni de l'addition ordinaire et de la multiplication modulo  $X^n - 1$ . À son tour, un tel polynôme peut s'interpréter comme le vecteur formé de ses coefficients. Pour tenir compte du fait que l'on travaille modulo  $X^n - 1$ , il est commode de considérer comme ensemble d'indices, non pas l'intervalle  $\{0, \dots, n - 1\}$ , mais l'ensemble  $\mathbf{Z}/n\mathbf{Z}$  des classes modulo  $n$ .

On considère donc formellement l'ensemble des applications de  $\mathbf{Z}/n\mathbf{Z}$  dans  $A$ , c'est-à-dire l'ensemble

$$E = A^{\mathbf{Z}/n\mathbf{Z}}$$

formé des familles  $(a_i)$ , avec  $i \in \mathbf{Z}/n\mathbf{Z}$ .

En associant à chaque élément  $a \in E$  le polynôme

$$P_a(X) = \sum_{i=0}^{n-1} a_i X^i = a_0 + \cdots + a_{n-1} X^{n-1} \in A[X],$$

puis sa classe modulo  $X^n - 1$ , on obtient une bijection de  $E$  sur l'anneau quotient  $A[X]/(X^n - 1)$ . Cette bijection préserve l'addition et la multiplication par les scalaires (éléments de  $A$ ). La multiplication par  $X$  correspond à la permutation circulaire des indices  $i \mapsto i - 1 \bmod n$ , qui applique  $(a_0, \dots, a_{n-1})$  sur  $(a_{n-1}, a_0, \dots, a_{n-2})$ , puisque

$$\begin{aligned} X(a_0 + \cdots + a_{n-1} X^{n-1}) &= a_0 X + \cdots + a_{n-1} X^n \\ &\equiv a_{n-1} + a_0 X + \cdots + a_{n-2} X^{n-1}. \end{aligned}$$

L'élément unité de  $A[X]/(X^n - 1)$  est l'image de l'élément  $\delta$  de  $E$  tel que  $\delta_0 = 1$  et  $\delta_i = 0$  pour  $i \neq 0$  : on a  $P_\delta = 1$ .

Pour obtenir la multiplication des polynômes, on doit munir  $E$  du *produit de convolution*. Il s'agit de l'application  $(a, b) \mapsto a * b$  définie par

$$(a * b)_k = \sum_{i+j=k} a_i b_j;$$

précisons que l'addition est à prendre au sens de  $\mathbf{Z}/n\mathbf{Z}$ , donc modulo  $n$  :

$$(a * b)_k = a_k b_0 + \cdots + a_0 b_k + a_{n-1} b_{k+1} + \cdots + a_{k+1} b_{n-1}.$$

On a par construction

$$P_{a*b} \equiv P_a P_b [\text{mod } (X^n - 1)].$$

**EXERCICE 4.3. [B]** — Vérifier.

Puisque la convolution correspond à la multiplication dans l'anneau quotient  $A[X]/(X^n - 1)$ , on a  $a * (b + c) = a * b + a * c$ ,  $a * b = b * a$  et  $\delta * a = a * \delta = a$ .

On peut aussi munir  $E$  de sa multiplication d'anneau produit, composante par composante, que nous noterons simplement  $a \cdot b$  : on a  $(a \cdot b)_i = a_i b_i$  pour tout  $i$ . L'élément unité de  $E$  pour cette autre structure d'anneau est le vecteur  $\mathbf{1} = (1, \dots, 1)$ .

#### 4.1.3. Définition de la transformation de Fourier

Supposons maintenant donnée une racine principale  $n$ -ième de l'unité dans  $A$ , soit  $\omega$ .

On appelle *transformations de Fourier* les applications

$$\mathcal{F} : E \rightarrow E, \quad \overline{\mathcal{F}} : E \rightarrow E,$$

définies par

$$(\mathcal{F}a)_j = \sum_{i \in \mathbf{Z}/n\mathbf{Z}} \omega^{ij} a_i, \quad (\overline{\mathcal{F}}a)_j = \sum_{i \in \mathbf{Z}/n\mathbf{Z}} \omega^{-ij} a_i,$$

c'est-à-dire

$$(\mathcal{F}a)_j = \sum_{i=0}^{n-1} \omega^{ij} a_i = P_a(\omega^j), \quad (\overline{\mathcal{F}}a)_j = \sum_{i=0}^{n-1} \omega^{-ij} a_i = P_a(\omega^{-j}),$$

pour  $j = 0, \dots, n-1$ .

On a immédiatement

$$\mathcal{F}(a + b) = \mathcal{F}a + \mathcal{F}b, \quad \overline{\mathcal{F}}(a + b) = \overline{\mathcal{F}}a + \overline{\mathcal{F}}b.$$

Par ailleurs, on a  $\mathcal{F}\delta = \overline{\mathcal{F}}\delta = \mathbf{1}$ .

Puisque  $P_{a*b} \equiv P_a P_b [\text{mod } (X^n - 1)]$ , on a  $P_{a*b}(\alpha) = P_a(\alpha)P_b(\alpha)$  pour tout racine  $n$ -ième de l'unité  $\alpha$ . Par conséquent :

**PROPOSITION 4.2.** — *On a  $\mathcal{F}(a * b) = \mathcal{F}a \cdot \mathcal{F}b$  et  $\overline{\mathcal{F}}(a * b) = \overline{\mathcal{F}}a \cdot \overline{\mathcal{F}}b$ .*

Les deux transformations de Fourier sont presque réciproques l'une de l'autre :

**PROPOSITION 4.3.** — *On a  $\mathcal{F}(\overline{\mathcal{F}}a) = n \cdot a$  et  $\overline{\mathcal{F}}(\mathcal{F}a) = n \cdot a$  pour tout  $a \in E$ .*

*Démonstration.* Il s'agit en effet de calculer le produit des matrices  $(\omega^{ij})$  et  $(\omega^{-jk})$ . Or  $\sum_{j=0}^{n-1} \omega^{ij} \omega^{-jk} = \sum_{j=0}^{n-1} \omega^{j(i-k)}$ . Pour  $i = k$ , on trouve  $n1_A$ . Pour  $i \neq k$ , on trouve 0 : en effet  $1 - \omega^{i-k}$  n'est pas diviseur de zéro par hypothèse et on a

$$(1 - \omega^{i-k}) \sum_{j=0}^{n-1} \omega^{j(i-k)} = 1 - \omega^{n(i-k)} = 0.$$
□

On en déduit les relations  $\mathcal{F}\mathbf{1} = \overline{\mathcal{F}}\mathbf{1} = n\delta$ . Si  $n1_A$  est inversible, il résulte de la proposition que l'application  $\mathcal{F}$  est bijective, de bijection réciproque  $\frac{1}{n}\overline{\mathcal{F}}$ .

Remarquons d'ailleurs que les applications  $\mathcal{F}$  et  $\overline{\mathcal{F}}$  ne diffèrent que très peu : notons  $\sigma : E \rightarrow E$  l'application telle que  $\sigma(a)_i = a_{-i}$  pour  $i \in \mathbf{Z}/n\mathbf{Z}$ , ou encore

$$\sigma(a_0, \dots, a_{n-1}) = (a_0, a_{n-1}, \dots, a_1).$$

On a alors  $\sigma \circ \sigma = Id$  et  $\overline{\mathcal{F}} = \mathcal{F} \circ \sigma = \sigma \circ \mathcal{F}$ , de sorte que la proposition précédente s'écrit aussi  $\mathcal{F} \circ \mathcal{F} = n\sigma$ .

**EXERCICE 4.4. [A]** — Vérifier ces relations.

**EXERCICE 4.5. [B]** — Montrer que les applications  $\mathcal{F}$  et  $\overline{\mathcal{F}}$  sont injectives.

**EXERCICE 4.6. [B]** — Sachant que  $n \cdot 1_A$  n'est pas diviseur de zéro, en déduire les relations  $n \cdot \mathcal{F}(ab) = \mathcal{F}(a) * \mathcal{F}(b)$  et  $n \cdot \overline{\mathcal{F}}(ab) = \overline{\mathcal{F}}(a) * \overline{\mathcal{F}}(b)$ .

En fait, inverser l'application  $\mathcal{F}$ , c'est retrouver le polynôme  $P_a(X)$  par interpolation entre ses valeurs pour  $X = \omega^i$ . La proposition 4.3 signifie qu'au coefficient  $1/n$  près, cette interpolation se ramène à l'évaluation aux mêmes points (dans l'ordre opposé) du polynôme  $P_{\mathcal{F}a}$  construit avec les valeurs de  $P_a$ .

## § 4.2. Transformation de Fourier rapide

### 4.2.1. Le principe

Nous allons particulariser ce qui précède au cas où  $n$  est une puissance de 2. Supposons d'abord seulement  $n$  pair, soit  $n = 2n'$ ; alors  $\omega^{n'}$  est une racine carrée principale de l'unité, donc est égale à  $-1_A$  (voir l'exercice 4.1).

De même,  $\omega' = \omega^2$  est une racine principale  $n'$ -ième de l'unité dans A. Notons  $\mathcal{F}'$  la transformation de Fourier correspondante :

$$(\mathcal{F}'b)_j = \sum_{i=0}^{n'-1} \omega'^{ij} b_i, \quad j = 0, \dots, n' - 1.$$

Décomposons la famille  $a$  en deux parties suivant la parité de l'indice  $i$  :

$$a_i^{\text{pair}} = a_{2i}, \quad a_i^{\text{impair}} = a_{2i+1}, \quad i = 0, \dots, n' - 1,$$

ce qu'on peut aussi écrire

$$P_a(X) = P_{a^{\text{pair}}}(X^2) + X P_{a^{\text{impair}}}(X^2).$$

On a aussitôt

$$(\mathcal{F}a)_j = \sum_{i=0}^{n'-1} \omega^{2ij} a_{2i} + \sum_{i=0}^{n'-1} \omega^{(2i+1)j} a_{2i+1}.$$

Mais, pour  $0 \leq i < n'$  et  $0 \leq j < n'$ , on a

$$\begin{aligned} \omega^{2ij} &= \omega'^{ij}, & \omega^{2i(n'+j)} &= \omega'^{ij}, \\ \omega^{(2i+1)j} &= \omega^j \omega'^{ij}, & \omega^{(2i+1)(n'+j)} &= \omega^{n'} \omega^j \omega'^{ij} = -\omega^j \omega'^{ij}. \end{aligned}$$

Par conséquent :

$$\begin{aligned} (\mathcal{F}a)_j &= (\mathcal{F}'a^{\text{pair}})_j + \omega^j (\mathcal{F}'a^{\text{impair}})_j, & j &= 0, \dots, n' - 1 \\ (\mathcal{F}a)_{n'+j} &= (\mathcal{F}'a^{\text{pair}})_j - \omega^j (\mathcal{F}'a^{\text{impair}})_j, & j &= 0, \dots, n' - 1. \end{aligned}$$

Ainsi, le calcul de la transformée de Fourier à  $n$  points  $\mathcal{F}a$  se ramène au calcul des deux transformées de Fourier à  $n'$  points  $\mathcal{F}'a^{\text{pair}}$  et  $\mathcal{F}'a^{\text{impair}}$ .

Le mécanisme de la *transformation de Fourier rapide*<sup>1</sup> consiste à prendre pour  $n$  une puissance de 2 et à itérer récursivement le procédé précédent.

Posons donc  $n = 2^k$ , notons  $m_F(k)$  le nombre total de multiplications par des racines de l'unité et  $a_F(k)$  le nombre total d'additions et de soustractions utilisées dans le calcul d'une transformée de Fourier à  $n = 2^k$  points. On a alors sans difficultés

$$m_F(k) = 2m_F(k-1) + (2^{k-1} - 1), \quad a_F(k) = 2a_F(k-1) + 2^k,$$

ce qui, compte-tenu de  $m_F(0) = a_F(0) = 0$  donne

$$m_F(k) = (k-2)2^{k-1} + 1, \quad a_F(k) = k2^k,$$

soit  $m_F(k) + a_F(k) \leq 3k2^{k-1} = \frac{3}{2}n \log n$ .

EXERCICE 4.7. [A] — Vérifier.

---

1. En anglais : « *Fast Fourier transform* »; en argot : « FFT ».

### 4.2.2. L'algorithme

Si l'on revient aux formules de base reliant  $\mathcal{F}$  et  $\mathcal{F}'$ , on constate que si l'on écrit  $i$  (indice de  $a$ ) et  $j$  (indice de  $\mathcal{F}a$ ) en numération binaire, le « découpage en quatre » faisant passer de  $n$  à  $n/2$  utilise le bit de poids faible de  $i$  et le bit de poids fort de  $j$ . Par ailleurs, tant  $\mathcal{F}a$  d'un côté que de l'autre  $\mathcal{F}'a^{\text{pair}}$  et  $\mathcal{F}'a^{\text{impair}}$  simultanément, peuvent être représentées par des vecteurs à  $n$  composantes. Explicitant tous les calculs intermédiaires, on arrive à l'algorithme suivant, dans l'exposition duquel nous suivons de près [Knuth 2], 4.3.3-C.

Pour toute suite  $(s_{r-1}, \dots, s_0)$  d'entiers égaux à 0 ou 1, on note

$$(s_{r-1}, \dots, s_0)_2 = \sum_{m=0}^{r-1} 2^m s_m = s_{r-1} 2^{r-1} + \dots + s_0$$

l'entier dont les  $s_m$  sont les chiffres binaires.

On fixe comme ci-dessus un anneau  $A$  et un entier  $k > 0$  et on considère une racine  $(2^k)$ -ième de l'unité  $\omega \in A$ . On se donne une famille  $(a_i)$ , où  $0 \leq i < 2^k$ , d'éléments de  $A$ , et on cherche à calculer les

$$(\mathcal{F}a)_j = \sum_{i=0}^{2^k - 1} \omega^{ij} a_i.$$

Nous écrirons  $i$  et  $j$  sous la forme (noter l'inversion)

$$i = (t_{k-1}, \dots, t_0)_2, \quad j = (s_0, \dots, s_{k-1})_2.$$

Remarquons au passage que la valeur de  $\omega^{ij}$  ne dépend que de la classe de  $ij$  modulo  $2^k$ , et que modulo  $2^k$  on a

$$ij = \sum_{r=0}^{k-1} 2^r t_r (s_0, \dots, s_{k-1})_2 \equiv \sum_{r=0}^{k-1} 2^r t_r (s_r, \dots, s_{k-1})_2,$$

ce qui donne

$$\omega^{ij} = \prod_{r=0}^{k-1} \omega^{2^r t_r (s_r, \dots, s_{k-1})_2}.$$

Cela étant, posons

$$F^{(0)}(t_{k-1}, \dots, t_0) = a_{(t_{k-1}, \dots, t_0)_2},$$

et définissons par récurrence pour  $m = 1, \dots, k$ ,

$$\begin{aligned} F^{(m)}(s_{k-1}, \dots, s_{k-m}, t_{k-m-1}, \dots, t_0) \\ = F^{(m-1)}(s_{k-1}, \dots, s_{k-m+1}, 0, t_{k-m-1}, \dots, t_0) \\ + \omega^{2^{k-m}(s_{k-m}, \dots, s_{k-1})_2} \cdot F^{(m-1)}(s_{k-1}, \dots, s_{k-m+1}, 1, t_{k-m-1}, \dots, t_0). \end{aligned}$$

Dans la formule précédente,  $s_{k-1}, \dots, s_{k-m}, t_{k-m-1}, \dots, t_0$  sont simplement  $k$  variables muettes, mais on les a nommées de façon à mettre en évidence le mécanisme qui « fait passer une par une les variables du côté  $t$  au côté  $s$  ».

**PROPOSITION 4.4.** — *On a*

$$F^{(k)}(s_{k-1}, \dots, s_0) = (\mathcal{F}a)_{(s_0, \dots, s_{k-1})_2}.$$

*Démonstration.* En effet, la définition de  $F^{(m)}(s_{k-1}, \dots, s_{k-m}, t_{k-m-1}, \dots, t_0)$  peut aussi s'écrire

$$\sum_{t_{k-m}} \omega^{2^{k-m} t_{k-m}(s_{k-m}, \dots, s_{k-1})_2} \cdot F^{(m-1)}(s_{k-1}, \dots, s_{k-m+1}, t_{k-m}, t_{k-m-1}, \dots, t_0).$$

En itérant de  $m = k$  à  $m = 1$ , on voit que  $F^{(k)}(s_{k-1}, \dots, s_0)$  peut aussi s'écrire

$$\begin{aligned} \sum_{t_0} \omega^{t_0(s_0, \dots, s_{k-1})_2} \cdots \sum_{t_{k-1}} \omega^{2^{k-1} t_{k-1}(s_{k-1})_2} \cdot a_{(t_{k-1}, \dots, t_0)_2} \\ = \sum_{t_0, \dots, t_{k-1}} \omega^{(t_{k-1}, \dots, t_0)_2(s_0, \dots, s_{k-1})_2} \cdot a_{(t_{k-1}, \dots, t_0)_2}, \end{aligned}$$

et on retrouve la définition de  $(\mathcal{F}a)_{(s_0, \dots, s_{k-1})_2}$ .  $\square$

**EXERCICE 4.8. [B]** — On voit ainsi que le calcul de  $\mathcal{F}$  demande  $k$  pas, chacun d'eux formés de  $2^k$  calculs élémentaires de la forme  $a + b\omega^r$ . Cela donne bien  $k2^k$  additions, mais semble donner aussi  $k2^k$  multiplications, soit nettement plus que l'estimation donnée ci-dessus. Où est l'erreur ?

### § 4.3. Applications

#### 4.3.1. Transformation de Fourier rapide modulo N

Ce qui précède montre l'intérêt qu'il y a à disposer de racines principales  $n$ -ièmes de l'unité, dans le cas où  $n$  est une puissance de 2, soit  $n = 2^k$ , avec  $k > 0$ .

Dire que  $\omega$  est une racine primitive  $(2^k)$ -ième de l'unité signifie que  $u = \omega^{2^{k-1}}$  est une racine carrée de l'unité distincte de 1. Si  $A$  est intègre, cela signifie que  $u = -1_A$  et que  $-1_A \neq 1_A$ , donc  $2 \cdot 1_A \neq 0$ . Ce critère reste valable dans le cas général pour les racines principales :

**LEMME 4.5.** — *Soient A un anneau tel que  $2 \cdot 1_A$  ne soit pas diviseur de zéro, k un entier  $> 0$  et  $\omega$  un élément de A tel que  $\omega^{2^{k-1}} = -1$ . Alors  $\omega$  est une racine principale  $(2^k)$ -ième de l'unité.*

*Démonstration.* On a  $\omega^{2^k} = (-1)^2 = 1$ . Soit  $i$  un entier tel que  $0 < i < 2^k$ . Il faut prouver que la condition  $(\omega^i - 1)a = 0$  implique  $a = 0$ . Or il existe  $r$  et  $s$  avec

$i = 2^r s$ ,  $0 \leq r < k$  et  $s$  impair. Puisque  $a = \omega^i a$ , on a aussi  $a = (\omega^i)^{2^{k-r}-1} a$ . Mais  $i 2^{k-r-1} = 2^{k-1}s$  et on obtient  $a = \omega^{2^{k-1}s} a = (-1)^s a = -a$ , donc  $2a = 0$  et en définitive  $a = 0$ .  $\square$

Cela s'applique notamment au cas où  $A$  est de la forme  $\mathbf{Z}/N\mathbf{Z}$  avec  $N$  impair (de sorte que 2 est inversible dans  $A$ ), et où  $\omega$  est la classe modulo  $N$  d'un entier  $a$  tel que  $a^{2^{k-1}} \equiv -1 \pmod{N}$ . On dira alors que  $\omega$  (ou  $a$ ) est une racine principale ( $2^k$ )-ième de l'unité modulo  $N$ .

Donnons trois exemples.

a) Soit d'abord  $a > 0$  un entier pair et posons  $N = a^{2^{k-1}} + 1$ . Alors  $a \pmod{N}$  est une racine principale ( $2^k$ )-ième de l'unité modulo  $N$ .

On peut prendre notamment pour  $N$  un nombre de Fermat, soit  $N = \text{Ferm}_{m+k} = 2^{2^{m+k}} + 1$ , avec  $\omega = 2^{2^{m+1}} \pmod{N}$ . Si  $N$  est premier, donc un nombre premier de Fermat (rappelons qu'on n'en connaît aucun au delà de  $\text{Fer}_4$ ), alors  $A$  est intègre et  $\omega$  est une racine principale ( $2^k$ )-ième de l'unité. Mais ce qui précède montre que ce dernier fait reste vrai sans l'hypothèse de primalité sur  $N$ . On peut donc appliquer le mécanisme de la transformation de Fourier rapide à  $2^k$  points sur l'anneau  $A = \mathbf{Z}/(2^{2^{m+k}} + 1)\mathbf{Z}$ . On notera au passage que les opérations à effectuer sont des additions/soustractions et des multiplications par des puissances de 2, qui se réalisent par des décalages.

b) On prend pour  $N$  un nombre premier  $p$  avec  $p \equiv 1 \pmod{2^k}$ . Alors  $p-1$  est multiple de  $2^k$ ; d'après le corollaire 3.9, il existe dans le corps  $\mathbf{Z}/p\mathbf{Z}$  des racines primitives de l'unité d'ordre  $2^k$  (qui sont donc principales puisque  $\mathbf{Z}/p\mathbf{Z}$  est un corps).

c) Plus généralement, on prend pour  $N$  un produit  $p_1 \cdots p_r$  de nombres premiers distincts, avec  $p_i \equiv 1 \pmod{2^k}$  pour tout  $i$ . On choisit pour chaque  $i$  une racine primitive de l'unité d'ordre  $2^k$  modulo  $p_i$ , soit  $\omega_i$ , et on prend  $\omega$  congru à  $\omega_i$  modulo  $p_i$  pour chaque  $i$  (théorème chinois, 2.1.1). On a alors  $\omega^{2^{k-1}} \equiv -1 \pmod{p_i}$  pour tout  $i$ , donc aussi  $\omega^{2^{k-1}} \equiv -1 \pmod{N}$ , et  $\omega$  est une racine principale ( $2^k$ )-ième de l'unité modulo  $N$ .

### 4.3.2. Applications arithmétiques

Supposons donc choisis un entier  $N$  et une racine principale ( $2^k$ )-ième de l'unité modulo  $N$ .

a) On peut calculer la transformée de Fourier d'une suite de  $2^k$  entiers modulo  $N$ . Cela demande comme on l'a vu  $m_F$  multiplications modulo  $N$  et  $a_F$  additions modulo  $N$  avec  $m_F = (k-2)2^{k-1} + 1$  et  $a_F = k2^k$ .

b) On peut calculer le produit de convolution de deux suites  $a$  et  $b$  de  $2^k$  entiers modulo  $N$ . Cela demande le calcul de  $\mathcal{F}(a)$ , de  $\mathcal{F}(b)$ , du produit  $c = \mathcal{F}(a) \cdot \mathcal{F}(b)$ , de  $\overline{\mathcal{F}}(c)$ , et enfin de  $\overline{\mathcal{F}}(c)/2^k$ , soit trois transformées de Fourier, et deux fois  $2^k$  multiplications. Ce qui fait au total  $m_C$  multiplications

modulo N et  $a_C$  additions modulo N, avec  $m_C = 3m_F + 2^k + 2^k = (3k - 2)2^{k-1} + 3 \leqslant 3k2^{k-1}$  et  $a_C = 3a_F = 3k2^k$ , donc  $m_C + a_C \leqslant 9k2^{k-1}$ . Si on pose  $n = 2^k$ , cela donne  $a_C = 3n \log n$  et  $m_C + a_C \leqslant \frac{9}{2}n \log n$ .

c) On peut calculer avec les mêmes coûts que dans b) le produit de deux polynômes P et Q à coefficients dans  $\mathbf{Z}/\mathbf{N}\mathbf{Z}$  avec  $\deg(P) + \deg(Q) < 2^k$ . En effet, ce produit coïncide alors avec le produit de convolution, la congruence modulo  $X^{2^k} - 1$  « n'ayant pas la place d'intervenir ».

d) On peut calculer avec les mêmes coûts que dans b) le produit de deux polynômes P et Q à coefficients entiers satisfaisant aux inégalités

$$\deg(P) + \deg(Q) < 2^k, \quad \min(\deg(P), \deg(Q)) = d$$

pourvu que leurs coefficients appartiennent tous à l'intervalle  $\llbracket 0, M \rrbracket$  (resp. à l'intervalle  $\llbracket -M, M \rrbracket$ ), avec  $dM^2 < N$  (resp.  $2dM^2 < N$ ). En effet, les coefficients du polynôme produit appartiennent tous à l'intervalle  $\llbracket 0, dM^2 \rrbracket$  (resp.  $\llbracket -dM^2, dM^2 \rrbracket$ ), et il suffit de les connaître modulo N.

### 4.3.3. Multiplication des grands entiers

La technique précédente s'applique à la multiplication des grands entiers.

Supposons d'abord fixée une base de numération  $M > 0$  et considérons deux entiers naturels écrits dans le système à base M sous la forme de nombres à  $r$  et  $s$  chiffres respectivement :

$$\begin{aligned} a &= a_0 + a_1M + \cdots + a_rM^{r-1} < M^r, \\ b &= b_0 + b_1M + \cdots + b_sM^{s-1} < M^s, \end{aligned}$$

où les  $a_i$  et les  $b_i$  appartiennent à l'intervalle  $\llbracket 0, M - 1 \rrbracket$ . Introduisant les deux polynômes

$$A(X) = a_0 + a_1X + \cdots + a_rX^{r-1},$$

$$B(X) = b_0 + b_1X + \cdots + b_sX^{s-1},$$

on a  $a = A(M)$  et  $b = B(M)$ , donc  $ab = C(M)$ , où  $C(X)$  est le polynôme produit

$$C(X) = A(X)B(X) = \bar{c}_0 + \cdots + \bar{c}_{r+s-2}X^{r+s-2}.$$

Les entiers  $\bar{c}_i$  sont compris entre 0 et  $(M - 1)^2 \inf(r, s)$ . Pour obtenir à partir d'eux les chiffres  $c_i$  du produit  $ab$ , il faut tenir compte des retenues et de leur propagation. De façon précise, écrivant comme à l'accoutumée  $x = (x \div M)M + (x \bmod M)$ , on pose  $c'_0 = \bar{c}_0$  et  $c_0 = c'_0 \bmod M$ , puis pour  $i = 1, \dots, r + s - 1$

$$c'_i = \bar{c}_i + c'_{i-1} \div M, \quad c_i = c'_i \bmod M,$$

avec en définitive  $0 \leqslant c_i < M$  et  $ab = c_0 + \cdots + c_{r+s-1}M^{r+s-1}$ .

On a ainsi ramené la multiplication d'un entier  $a \in \llbracket 0, M^r \rrbracket$  et d'un entier  $b \in \llbracket 0, M^s \rrbracket$  à deux opérations successives : la multiplication de deux polynômes de degrés respectivement  $< r$  et  $< s$  à coefficients dans l'intervalle  $[0, M]$  et le calcul récurrent de propagation de retenues. Cette multiplication de polynômes peut s'effectuer par transformation de Fourier rapide comme expliqué ci-dessus en 4.3.2, *d*) : on prend deux entiers  $N$  et  $k$  tels qu'il existe une racine principale  $(2^k)$ -ième de l'unité modulo  $N$ , et on impose  $r + s - 2 < 2^k$  et  $\min(r, s)(M - 1)^2 < N$ , soit pour simplifier  $r \leq 2^{k-1}$ ,  $s \leq 2^{k-1}$  et  $2^{k-1}(M - 1)^2 < N$ . Si on mesure le coût total en nombre d'opérations modulo  $N$ , la propagation des retenues a un coût au plus en  $2^k$  et la multiplication des polynômes un coût majoré par  $9k2^{k-1}$ . Résumons :

**PROPOSITION 4.6.** — *Soient  $N$ ,  $k$  et  $M$  des entiers naturels ayant les propriétés suivantes :*

- a) il existe une racine principale d'ordre  $2^k$  de l'unité modulo  $N$ ;*
- b) on a  $2^{k-1}(M - 1)^2 < N$ .*

*On peut alors multiplier deux entiers naturels inférieurs à  $M^{2^{k-1}}$  à l'aide d'opérations modulo  $N$ , en nombre au plus  $9k2^{k-1}$  (essentiellement  $3k2^{k-1}$  multiplications).*

#### 4.3.4. La méthode de Pollard

Pour planter en machine cette méthode, on a intérêt à prendre une base  $M$  qui tienne dans un mot machine, mais qui ne soit pas nettement plus petite. La condition *b*) ci-dessus montre alors qu'on ne peut éviter de prendre  $N$  assez grand, disons d'une taille entre deux et trois mots. On voit donc qu'il est raisonnable de prendre pour  $N$  un produit de trois nombres premiers tenant chacun dans un mot. Précisons ces divers choix dans un cas précis.

On suppose que la machine dont on dispose permet de calculer directement et exactement sur des nombres  $a$  avec  $0 \leq a < 2^s$  (on prendra par exemple  $s = 16$  sur une machine à 32 bits ne disposant pas d'opérations ayant un résultat en longueur double). On choisit trois nombres premiers distincts  $p_1, p_2, p_3$ , inférieurs à  $2^s$  et de la forme  $p_i = a_i 2^k + 1$ , où  $k$  est aussi grand que possible. Pour chaque  $p_i$ , on détermine une racine primitive  $\omega_i$  de l'unité modulo  $p_i$ , d'ordre  $2^k$ .

Pour  $s = 16$ , on pourra ainsi prendre  $k = 12$ , avec

$$p_1 = 40961 = 10 \times 2^{12} + 1,$$

$$p_2 = 12289 = 3 \times 2^{12} + 1,$$

$$p_3 = 61441 = 15 \times 2^{12} + 1.$$

Un rapide calcul donne des racines primitives  $\omega_1 = 28, \omega_2 = 41, \omega_3 = 19$ :

```
[40961, 12289, 61441].collect do |p|
  for i in 2...p do break if i.ordre_mod(p) == 4096 end
  i
end # => [28, 41, 19]
```

On pose  $N = p_1 p_2 p_3$ ; on représente les entiers modulo  $N$  par leurs trois restes modulo les  $p_i$ , donc par trois mots-machine, et on calcule modulo  $N$  via le théorème chinois. Par exemple, une multiplication modulo  $N$  demande un nombre fixe (disons une petite dizaine) de multiplications sur un mot machine.

On applique alors la proposition 4.6 avec une base  $M = 2^m$  telle que  $2^{2m+k-1} < N$ . Puisqu'on a  $p_i > 2^k$ , donc  $N > 2^{3k}$ , on peut prendre  $m = k$ . On pourra alors multiplier des entiers inférieurs à  $M^{2^{k-1}} = 2^{m2^{k-1}}$ , donc à  $m2^{k-1}$  bits.

Dans notre exemple, on a  $k = 12$ , mais  $N = p_1 p_2 p_3 > 5 \cdot 3 \cdot 15 \cdot 2^{37} > 2^{44}$ , et on peut même aller jusqu'à  $M = 2^{16}$ , de sorte que  $m2^k = 16 \times 2^{11} = 2^{15}$ . Cela donne  $2^{15}$  bits, donc *plus de dix mille chiffres décimaux*.

Dans la pratique, on ne fait pas les calculs modulo  $N$ , mais modulo les  $p_i$ , et on n'utilise le théorème chinois qu'au dernier moment et non à chaque pas. De façon plus précise :

a) on part des entiers naturels  $a$  et  $b$  inférieurs à  $2^{m2^{k-1}}$ , écrits en base  $2^m$  ou, ce qui revient au même des polynômes  $A$  et  $B$  à coefficients dans l'intervalle  $\llbracket 0, 2^m \rrbracket$  et de degré au plus  $2^{k-1} - 1$ ;

b) on réduit ces polynômes modulo les  $p_i$  pour obtenir six polynômes  $A_1, A_2, A_3, B_1, B_2, B_3$ ;

c) on multiplie ces polynômes deux à deux, par transformation de Fourier, ce qui donne  $C_1 = A_1 \cdot B_1 \bmod p_1$ , etc.;

d) en appliquant le théorème chinois, on déduit de ces trois polynômes le polynôme  $C = AB$ , de degré au plus  $2^k - 2$ , à coefficients dans l'intervalle  $\llbracket 0, 2^{2m+k-1} \rrbracket$ ;

e) on propage les retenues pour obtenir les chiffres dans la base  $2^m$  du produit  $c = ab$ .

#### 4.3.5. La méthode de Schönhage-Strassen

Les tailles obtenues ci-dessus sont suffisantes pour pratiquement toute application concrète. La méthode de Schönhage-Strassen, dont l'intérêt est plus théorique que pratique, permet de multiplier deux nombres de tailles  $n$  en un nombre d'opérations élémentaires de la forme  $C \cdot n \cdot \log n \cdot \log \log n$ .

Elle consiste en une application récursive de ce qu'on a vu ci-dessus. Expliquons-en brièvement le principe, en renvoyant à [Knuth 2], 4.3.3 pour les détails. Par transformation de Fourier rapide, on ramène en effet le produit de deux entiers inférieurs à  $A = M^{2^{k-1}}$  à un certain nombre de

multiplications d'entiers inférieurs à N, où M, N et  $k$  sont reliés par les conditions de la proposition 4.6. On choisit M de la forme  $2^{2^m+1}$ , de sorte que  $A = 2^{2^k+m}$ . On choisit pour N un nombre (de Fermat) de la forme  $2^{2^r} + 1$  (et pour racine principale la puissance convenable de 2), de sorte que les calculs modulo N se ramènent sans difficultés à des calculs modulo B =  $2^{2^r}$ . On poursuit alors récursivement. Il reste à ajuster convenablement à chaque pas les choix relatifs de  $m$ ,  $k$  et  $r$  et à compter avec précision les diverses opérations.

# Chapitre 5

## Résidus quadratiques et applications

Ce chapitre est consacré à une nouvelle descendance du théorème de Fermat. Si  $p$  est un nombre premier  $> 2$  et si  $a$  est un entier premier à  $p$ , alors  $(a^{(p-1)/2})^2$  est congru à 1 modulo  $p$ , donc  $a^{(p-1)/2}$  est congru à 1 ou à  $-1$ . Le premier cas est celui des *résidus quadratiques* (congrus à un carré modulo  $p$ ), l'autre celui des non-résidus.

D'un autre côté, étant donnés deux entiers premiers entre eux  $a$  et  $n$ ,  $n$  étant impair, on peut définir le *symbole de Jacobi*  $(\frac{a}{n})$ , qui se calcule rapidement et vaut  $\pm 1$ . L'un de ses vertus fondamentales est que, lorsque  $p$  est premier, la condition  $(\frac{a}{p}) = 1$  caractérise les résidus quadratiques, de sorte qu'on a  $a^{(p-1)/2} \equiv (\frac{a}{p}) [\text{mod } 1]$ . Cela nous donne une nouvelle condition nécessaire de primalité, donc un critère de non-primalité (Solovay-Strassen).

Ce critère, comme celui de Miller-Rabin rencontré au chapitre 2, donne naissance à un algorithme *probabiliste* de primalité. Cela nous amènera à une brève excursion dans les classes de complexité  $\mathcal{P}$ ,  $\mathcal{NP}$ ,  $\mathcal{RP}$  et  $\mathcal{ZPP}$ .

### § 5.1. Résidus quadratiques

#### 5.1.1. Carrés dans un corps fini

Soit  $K$  un corps fini (par exemple un corps  $\mathbf{Z}/p\mathbf{Z}$  avec  $p$  premier). Posons  $q = \#K$  et supposons que  $q$  soit *impair*. On ne peut avoir  $2 \cdot 1_K = 0$ , car on a  $q \cdot 1_K = 0$  d'après le théorème de Lagrange et cela impliquerait  $1_K = 0$ . On a donc  $-1_K \neq 1_K$ . On dit par ailleurs pour abréger qu'un élément  $x$  de  $K^*$  est un carré si c'est le carré d'un élément de  $K^*$ .

**PROPOSITION 5.1.** — *Parmi les  $q - 1$  éléments de  $K^*$ , la moitié exactement sont des carrés. Si  $a \in K^*$  est un carré, on a  $a^{(q-1)/2} = 1_K$ . Sinon, on a  $a^{(q-1)/2} = -1_K$ .*

*Démonstration.* Pour tout  $x \in K^*$ , on a  $x \neq -x$  mais  $x^2 = (-x)^2$ . Pour  $a \neq 0$ , l'équation  $X^2 = a$  qui a au plus deux racines en a exactement zéro ou deux. On en conclut qu'exactement  $(q - 1)/2$  des éléments de  $K^*$  sont des carrés. Pour tout  $x \in K^*$ , on a  $x^{q-1} = 1$ , donc  $x^{(q-1)/2} = \pm 1$ . Chacune des équations  $X^{(q-1)/2} = \pm 1$  a au plus  $(q - 1)/2$  racines, donc exactement  $(q - 1)/2$  racines. Si  $a = x^2$  est un carré, on a  $a^{(q-1)/2} = x^{q-1} = 1$ . La proposition résulte immédiatement de là.  $\square$

Par exemple, dans  $(\mathbf{Z}/17\mathbf{Z})^*$ , il y a 8 carrés (qui sont  $\pm 1, \pm 2, \pm 4$  et  $\pm 8$ ) et 8 non carrés (qui sont  $\pm 3, \pm 5, \pm 6$  et  $\pm 7$ ).

**EXERCICE 5.1. [A]** — Le vérifier.

**COROLLAIRE 5.2.** — Pour que  $-1$  soit un carré dans  $K$ , il faut et il suffit que l'on ait  $q \equiv 1 \pmod{4}$ .

**EXERCICE 5.2. [C]** — Soit  $r$  un diviseur de  $q - 1$ . Montrer que  $K$  contient  $r$  racines  $r$ -ièmes de l'unité, qu'il existe dans  $K^*$  exactement  $(q-1)/r$  puissances  $r$ -ièmes et qu'elles sont caractérisées par la relation  $a^{(q-1)/r} = 1$ .

**REMARQUE 5.3.** — On sait que le groupe  $K^*$  est cyclique. On peut donc aussi déduire la proposition 5.1 des remarques plus générales suivantes. Soit  $G$  un groupe cyclique fini, d'ordre  $r > 1$ , d'élément neutre noté  $e$ .

Si  $r$  est impair, soit  $r = 2m - 1$ , alors tout élément  $a$  de  $G$  est un carré (que  $G$  soit cyclique ou non d'ailleurs), car on a  $a^r = e$ , donc  $a = a^{r+1} = a^{2m} = (a^m)^2$ .

Il n'en est pas de même si  $r$  est pair, soit  $r = 2m$ . Il existe alors dans  $G$  un unique élément d'ordre deux, soit  $\varepsilon$ , et pour tout élément  $a$  de  $G$ , on a soit  $a^m = e$  auquel cas  $a$  est un carré, soit  $a^m = \varepsilon$  auquel cas  $a$  n'est pas un carré. En effet, choisissons un générateur  $g$  de  $G$ , posons  $\varepsilon = g^m$  et écrivons un élément quelconque de  $G$  sous la forme  $a = g^n$  avec  $0 \leq n < 2r$ . Alors la condition  $a^2 = e$  signifie que  $2n$  est divisible par  $2m$ , donc que l'on a, soit  $n = 0$  et  $a = e$ , soit  $n = m$  et  $a = \varepsilon$ . On a par ailleurs  $(g^s)^2 = g^{2s}$ , de sorte que  $a$  est un carré exactement lorsqu'on peut résoudre  $n \equiv 2s \pmod{2m}$ , c'est-à-dire lorsque  $n$  est pair ; d'autre part on a  $a^m = g^{nm} = (g^m)^n = \varepsilon^n$  ; ainsi, dire que  $a^m = e$ , c'est dire que  $n$  est pair, ce qui prouve notre assertion. On a vu au passage que  $G$  possède  $m$  carrés et  $m$  non carrés.

### 5.1.2. Le symbole de Legendre

Soient  $m$  et  $n$  deux entiers. On dit que  $n$  est un *résidu quadratique* modulo  $m$  s'il existe un entier  $a$  tel que  $n \equiv a^2 \pmod{m}$ , c'est-à-dire si  $n \bmod m$  est un carré dans l'anneau  $\mathbf{Z}/m\mathbf{Z}$ . Dans le cas contraire, on dit souvent que  $n$  est un *non-résidu*. Bien évidemment, l'ensemble des résidus quadratiques modulo  $m$  est stable par congruence modulo  $m$  et par multiplication. Bien évidemment aussi, tout multiple de  $m$  est un résidu quadratique modulo  $m$ .

**EXERCICE 5.3. [A]** — Montrer que, pour que  $n$  soit un résidu quadratique modulo 4, il faut et il suffit qu'il soit congru à 0 ou à 1 modulo 4.

**EXERCICE 5.4. [A]** — Montrer que, si  $n$  est un résidu quadratique modulo  $m$ , alors c'est aussi un résidu quadratique modulo tout diviseur de  $m$ .

**EXERCICE 5.5. [B-C]** — Montrer que, si  $n$  est un résidu quadratique modulo  $m$  et modulo  $m'$ , c'est aussi un résidu quadratique modulo le ppcm de  $m$  et  $m'$ .

**DÉFINITION 5.4.** — Pour tout entier  $n$  et tout nombre premier  $p$ , on note  $(\frac{n}{p})$  l'entier défini comme suit : on a

$$\left(\frac{n}{p}\right) = \begin{cases} 0 & \text{si } n \text{ est divisible par } p \\ 1 & \text{si } n \text{ n'est pas divisible par } p \text{ et est un résidu quadratique modulo } p \\ -1 & \text{si } n \text{ n'est pas un résidu quadratique modulo } p \end{cases}$$

L'expression  $(\frac{n}{p})$  s'appelle *symbole de Legendre*. Par exemple, pour  $p = 17$  et  $n = -3$ , on a  $(\frac{-3}{17}) = -1$ .

Par construction,  $(\frac{n}{p})$  ne dépend que de la classe de  $n$  modulo  $p$ . Par exemple,  $(\frac{n}{2})$  vaut 0 si  $n$  est pair et 1 si  $n$  est impair, ce qui implique  $(\frac{n}{2}) \equiv n \pmod{2}$ . De même, d'après le corollaire 5.2, on a  $(\frac{-1}{p}) = 1$  lorsque  $p \equiv 1 \pmod{4}$  et  $(\frac{-1}{p}) = -1$  lorsque  $p \equiv -1 \pmod{4}$ .

**EXERCICE 5.6. [A]** — Montrer que  $(\frac{n}{3}) \equiv n \pmod{3}$ .

**EXERCICE 5.7. [A]** — Supposons  $p > 2$ . Si  $n \pmod{p}$  est une racine primitive  $(p-1)$ -ième modulo  $p$ , on a  $(\frac{n}{p}) = -1$ .

**PROPOSITION 5.5 (Euler).** — Soit  $p$  un nombre premier impair. Parmi les  $p-1$  éléments de  $(\mathbf{Z}/p\mathbf{Z})^*$ , la moitié exactement sont des carrés. Pour tout entier  $n$ , on a  $(\frac{n}{p}) \equiv n^{(p-1)/2} \pmod{p}$ .

*Démonstration.* La première partie résulte de la proposition 5.1, ainsi que la formule proposée lorsque  $n$  n'est pas divisible par  $p$ . Lorsque  $n$  est divisible par  $p$ , les deux membres de la formule sont nuls.  $\square$

La *formule d'Euler*  $(\frac{n}{p}) \equiv n^{(p-1)/2} \pmod{p}$  permet de calculer  $(\frac{n}{p})$  via l'algorithme rapide de calcul des puissances. On retrouve par exemple la valeur de  $(\frac{-3}{17})$  par le calcul suivant modulo 17 :

$$(\frac{-3}{17}) \equiv (-3)^8 \equiv 9^4 \equiv 81^2 \equiv (-4)^2 \equiv 16 \equiv -1.$$

Le coût de ce calcul (non compris la réduction initiale de  $n$  modulo  $p$ ) est en  $(\log p)^3$ . Nous verrons un peu plus loin (5.3.2) un algorithme de calcul en  $(\log p)^2$ .

**COROLLAIRE 5.6.** — On a pour tous entiers  $n$  et  $n'$

$$(\frac{nn'}{p}) = (\frac{n}{p})(\frac{n'}{p})$$

et  $(\frac{n^2n'}{p}) = (\frac{n'}{p})$  si  $n$  est premier à  $p$ .

**REMARQUE 5.7.** — 1) Il est clair a priori que le produit de deux carrés est un carré, et que l'inverse ( $\mathbf{Z}/p\mathbf{Z}$  est un corps) d'un carré est un carré. Il en résulte que le produit d'un non carré par un carré n'est pas un carré. Le seul point nouveau du corollaire précédent est que le produit de deux non carrés est un carré.

2) On a  $X^p - X = X(X^{(p-1)/2} - 1)(X^{(p-1)/2} + 1)$ . Les racines dans  $\mathbf{Z}/p\mathbf{Z}$  des trois facteurs de  $X^p - X$  ainsi mis en évidence sont respectivement 0, les carrés, les non carrés.

3) Avec un peu de théorie des corps, on peut donner une autre démonstration de la proposition précédente, comme suit. Posons  $a = n \pmod p \in \mathbf{Z}/p\mathbf{Z}$ . Supposons  $a \neq 0$ . S'il existe  $b \in \mathbf{Z}/p\mathbf{Z}$  avec  $a = b^2$ , alors  $a^{(p-1)/2} = b^{p-1} = 1$  d'après Fermat. Sinon, l'anneau  $(\mathbf{Z}/p\mathbf{Z})[X]/(X^2 - a)$  est un corps, et dans ce corps la classe  $b$  de  $X$  est une racine carrée de  $a$ . Si on avait  $a^{(p-1)/2} = 1$ , on aurait  $b^p = b$ , ce qui impliquerait, puisque cette équation ne peut avoir qu'au plus  $p$  racines, que  $b$  appartiendrait à  $\mathbf{Z}/p\mathbf{Z}$ , contrairement à l'hypothèse.

Il est immédiat de fabriquer autant de résidus quadratiques modulo  $p$  que l'on veut : il suffit de prendre  $a$  quelconque et de considérer  $n = a^2$ . Il est moins immédiat d'exhiber un non-résidu. Si  $p \equiv -1 \pmod 4$ ,  $-1$  est un non-résidu. Si  $p \equiv 3 \pmod 8$  ou  $p \equiv 5 \pmod 8$ , on verra ci-dessous que  $2$  est un non-résidu. Si  $p \equiv 5 \pmod {12}$  ou  $p \equiv 7 \pmod {12}$ , on verra de même que  $3$  est un non-résidu. Et ainsi de suite.

En utilisant l'hypothèse de Riemann généralisée (voir 3.1.5), Eric BACH a démontré qu'il existe un non-résidu modulo  $p$  inférieur à  $2 \ln(p)^2$ , où  $\ln$  désigne le logarithme népérien, donc inférieur à  $\log(p)^2$ . Cela rend raisonnable la recherche exhaustive. Une autre méthode est probabiliste : on choisit  $n$  au hasard et on calcule  $(\frac{n}{p})$ ; à chaque tirage, on a une chance sur deux de réussir, d'où en  $N$  tirages une « probabilité de réussite » de  $1 - 2^{-N}$ .

Voici une implantation possible en Ruby :

---

#### PROGRAMME 5.8. — Recherche d'un non-résidu modulo $p$

---

```
class Integer
  # provisoire : on utilisera plus tard le symbole de Legendre
  def is_residue_mod?(p)
    puissmod((p-1).half, p) == 1
  end
end
[1,2,3,4,5].collect{ |n| n.is_residue_mod?(17)}
# => [true, true, false, true, false]

def find_non_residue_mod(p)
  loop do
    a = 1 + rand(p-2)
    return a unless a.is_residue_mod?(p)
    # mieux : return a unless jacobi(a,p) == 1
  end
end
find_non_residue_mod(641) # => 247
```

---

### 5.1.3. Calcul d'une racine carrée dans $\mathbf{Z}/p\mathbf{Z}$

Soit toujours  $p$  un nombre premier impair, et soit  $n$  un entier premier à  $p$ . La formule d'Euler permet de vérifier si  $n$  est un résidu quadratique modulo  $p$ , c'est-à-dire s'il existe  $a$  avec  $a^2 \equiv n \pmod{p}$ , mais ne donne aucune indication sur la valeur de  $a$ .

Le cas le plus simple est celui où on a  $p \equiv -1 \pmod{4}$ , c'est-à-dire où  $p - 1 = 2t$  avec  $t$  impair. On a alors pour tout  $n$  premier à  $p$  la relation

$$(n^{\frac{p+1}{4}})^2 = n \cdot n^{\frac{p-1}{2}} \equiv (\frac{n}{p})n$$

et  $n^{\frac{p+1}{4}}$  est une racine carrée de  $n$  ou de  $-n$  selon que  $n$  est un résidu quadratique ou non.

Dans le cas général, la chose est plus difficile. Écrivons  $p - 1 = 2^{r+1}t$  avec  $t$  impair, de sorte que  $(\frac{n}{p}) \equiv n^{2^rt}$ . L'algorithme suivant (dû à Daniel SHANKS) donne une réponse lorsqu'on suppose disposer d'un entier auxiliaire  $z$  tel que  $z^{2^r} \equiv -1 \pmod{p}$ ; pour construire un tel  $z$ , il suffit de partir d'un non-résidu  $m$  et de prendre  $z = m^t$ .

---

**ALGORITHME 5.9. — Racine carrée modulo  $p$**  —

---

*Entrées* : entiers  $n$ ,  $z$  et  $r$ . *Sorties* : entier  $x$ .

Règles :

$$\begin{aligned} (n, z) &\mapsto (n^t, n^{\frac{t+1}{2}}, z, r) \\ [r \geq 1 \text{ et } b^{2^{r-1}} = 1] : (b, x, y, r) &\mapsto (b, x, y^2, r-1) \\ [r \geq 1 \text{ et } b^{2^{r-1}} = -1] : (b, x, y, r) &\mapsto (by^2, xy, y^2, r-1) \\ [r = 0] : (b, x, y, r) &\mapsto x \end{aligned}$$


---

**PROPOSITION 5.10. —** Partant d'entiers  $n$  et  $z$  avec  $z^{2^r} \equiv -1 \pmod{p}$  et  $(\frac{n}{p}) = 1$ , l'algorithme précédent réécrit en  $r + 2$  pas le couple  $(n, z)$  en un entier  $x$  tel que  $x^2 \equiv n \pmod{p}$ .

*Démonstration.* Chaque quadruplet  $(b, x, y, r)$  considéré satisfait aux conditions suivantes

$$nb = x^2, \quad y^{2^r} \equiv -1 \pmod{p}, \quad b^{2^r} \equiv 1 \pmod{p}.$$

C'est en effet vrai au premier pas, puisqu'on a  $n \cdot n^t = (n^{\frac{t+1}{2}})^2$  et par hypothèse  $z^{2^r} \equiv -1$  et  $(n^t)^{2^r} = n^{2^rt} \equiv (\frac{n}{p}) = 1$ . Par ailleurs, les deux règles centrales conservent ces conditions. Puisqu'on a toujours  $b^{2^r} \equiv 1$ , l'une des deux conditions  $b^{2^{r-1}} \equiv \pm 1$  est satisfaite, et la réécriture se poursuit jusqu'au moment où  $r = 0$ . Mais on a alors  $b \equiv 1$ , donc  $n \equiv nb = x^2$ .  $\square$

En Ruby, voici une implantation possible :

PROGRAMME 5.II. — *Calcul d'une racine carrée modulo p* ——————

```

class Integer
  def root_mod(p)
    unless self.is_residue_mod?(p)
      raise ArgumentError, "#{self} n'est pas un résidu modulo #{p}"
    end
    r, t = -1, p-1
    until t.odd? do
      r, t = r+1, t/2
    end
    y = find_non_residue_mod(p).puissmod(t,p)
    b = self.puissmod(t,p)
    x = self.puissmod((t+1)/2,p)
    until r == 0 do
      y2 = y.puissmod(2,p)
      if b.puissmod(2**((r-1),p) == p - 1
        b, x = b*y2 % p, x*y % p
      end
      y, r = y2, r-1
    end
    x
  end
end
(47*47).root_mod(65537) # => 65490 # -47, une chance sur deux !

```

---

5.I.4. Carrés dans  $\mathbf{Z}/p^n\mathbf{Z}$ 

Soit d'abord  $p$  un nombre premier impair. Dès qu'un entier premier à  $p$  est un résidu quadratique modulo  $p$ , c'est aussi un résidu quadratique modulo toutes les puissances de  $p$  :

**PROPOSITION 5.12.** — *Soient  $p$  un nombre premier impair,  $r$  un entier  $\geq 1$  et  $n$  un entier. Si  $(\frac{n}{p}) = 1$ , il existe  $a \in \mathbf{Z}$  avec  $n \equiv a^2 \pmod{p^r}$ .*

*Démonstration.* On peut donner deux démonstrations.

La première consiste à appliquer la remarque 5.3 : le groupe  $(\mathbf{Z}/p^r\mathbf{Z})^*$  est cyclique d'ordre  $p^{r-1}(p-1)$  (proposition 3.40) ; par ailleurs, si  $n^{(p-1)/2} \equiv 1 \pmod{p}$ , on a  $n^{p^{r-1}(p-1)/2} \equiv 1 \pmod{p^r}$  (lemme 3.38).

La seconde démonstration consiste à construire de proche en proche, en partant d'un entier  $a$  tel que  $a^2 \equiv n \pmod{p}$ , un entier  $a_r$  tel que  $a_r^2 \equiv n \pmod{p^r}$ . On trouvera les détails dans l'exercice ci-dessous.  $\square$

**EXERCICE 5.8. [B]** — Terminer cette démonstration en utilisant la « méthode de Newton » : itérer l'application  $N(x) = x + \frac{p-1}{2}b(x^2 - n)$ , où  $b$  est choisi de façon que  $ab \equiv 1 \pmod{p}$ .

Le cas du nombre premier 2 est un peu différent du cas général, car le groupe  $(\mathbf{Z}/2^r\mathbf{Z})^*$  n'est pas cyclique pour  $r > 1$  :

**PROPOSITION 5.13.** — Soient  $n$  un entier impair et soit  $r \geq 3$ . Pour que  $n$  soit un résidu quadratique modulo  $2^r$ , il faut et il suffit que  $n \equiv 1 \pmod{8}$ .

*Démonstration.* La condition est nécessaire, puisque  $n$  doit être un résidu quadratique modulo 8 et que  $(\pm 1)^2 = 1$  et  $(\pm 3)^2 = 9$  sont congrus à 1 modulo 8. Il s'agit de prouver qu'elle est suffisante. Cela peut se démontrer par itération : on pose  $n = 1 + 8b$ , et on cherche une racine carrée sous la forme  $a = 1 + 4y$  ; il faut alors résoudre l'équation  $y + 2y^2 \equiv b$ , ce qu'on fait par la méthode de Newton en itérant l'application  $N(y) = b - 2y^2$  à partir de  $y = 0$ .

On peut aussi déduire la proposition de la structure du groupe  $(\mathbf{Z}/2^r\mathbf{Z})^*$ . En effet, la proposition 3.41 implique que tout élément de ce groupe peut s'écrire  $\pm 5^s$ . Mais, modulo 8, on a  $5^{2t} \equiv 1$ ,  $-5^{2t} \equiv 7$ ,  $5^{2t+1} \equiv 5$  et  $-5^{2t+1} \equiv 3$ . Par conséquent, si  $n$  est congru à 1 modulo 8, alors  $n \pmod{2^r}$  est de la forme  $5^{2t}$ , donc est un carré.  $\square$

**EXERCICE 5.9. [B]** — Terminer la première démonstration.

**REMARQUE 5.14.** — Le plus petit entier congru à 1 modulo 8 qui n'est pas un carré parfait est 17. Aucun raisonnement basé sur la parité ne peut prouver que 17 n'est pas un carré, puisqu'on peut résoudre la congruence  $n^2 \equiv 17 \pmod{2^r}$  pour tout  $r$ . Peut-être est-ce l'interprétation qu'il faut donner au passage où Platon parle de l'existence d'une ancienne démonstration (antérieure à démonstration générale utilisant la décomposition en facteurs premiers) de l'irrationalité non seulement de  $\sqrt{2}$ , mais aussi de  $\sqrt{3}$ , « ... et ainsi de suite jusqu'à  $\sqrt{17}$ ... ».

### 5.1.5. Les signes $\varepsilon(n)$ , $\omega(n)$ et $\theta(a, b)$

Il est commode pour la suite d'introduire les deux fonctions suivantes, définies sur l'ensemble des entiers impairs et à valeurs dans  $\{\pm 1\}$  :

$$\begin{aligned}\varepsilon(n) &= \begin{cases} 1 & \text{si } n \equiv 1 \pmod{4}, \\ -1 & \text{si } n \equiv 3 \pmod{4}; \end{cases} \\ \omega(n) &= \begin{cases} 1 & \text{si } n \equiv 1 \pmod{8} \text{ ou si } n \equiv 7 \pmod{8}, \\ -1 & \text{si } n \equiv 3 \pmod{8} \text{ ou si } n \equiv 5 \pmod{8}. \end{cases}\end{aligned}$$

Si on tient à des formules explicites, on peut poser

$$\varepsilon(n) = (-1)^{(n-1)/2}, \quad \omega(n) = (-1)^{(n^2-1)/8}.$$

On a aussitôt

$$\varepsilon(a)\varepsilon(b) = \varepsilon(ab), \quad \omega(a)\omega(b) = \omega(ab).$$

Avec ces notations :

- ▷ On a  $(\frac{-1}{p}) = \varepsilon(p)$  pour  $p$  premier impair.
- ▷ Dire que l'entier impair  $n$  est un résidu quadratique modulo 4, c'est dire que  $\varepsilon(n) = 1$ .

▷ Dire que l'entier impair  $n$  est un résidu quadratique modulo 8 (ou toute puissance de 2 supérieure à 8), c'est dire que  $\varepsilon(n) = \omega(n) = 1$ .

**EXERCICE 5.10. [B]** — Quels sont les homomorphismes du groupe  $(\mathbf{Z}/2^r\mathbf{Z})^*$  dans  $\{\pm 1\}$  ?

Nous aurons aussi besoin plus loin de la fonction de deux entiers impairs définie par

$$\theta(a, b) = \begin{cases} 1 & \text{si } a \equiv 1 \pmod{4} \text{ ou } b \equiv 1 \pmod{4}, \\ -1 & \text{si } a \equiv 3 \pmod{4} \text{ et } b \equiv 3 \pmod{4}. \end{cases}$$

On peut aussi l'écrire

$$\theta(a, b) = (-1)^{(a-1)(b-1)/4} = \varepsilon(a)^{(b-1)/2} = \varepsilon(b)^{(a-1)/2}.$$

On a de plus  $\theta(a, b) = \theta(b, a)$  et

$$\theta(aa', b) = \theta(a, b)\theta(a', b), \quad \theta(a, bb') = \theta(a, b)\theta(a, b').$$

**EXERCICE 5.11. [A]** — Vérifier ces relations.

## § 5.2. Réciprocité quadratique

Pour calculer le symbole de Legendre  $(\frac{n}{p})$ , où  $p$  est un nombre premier impair et où  $n$  est premier à  $p$ , on peut utiliser la méthode suivante. Posons  $a = n \pmod{p} \in \mathbf{Z}/p\mathbf{Z}$ , et supposons donnés un anneau  $A$  contenant le corps  $\mathbf{Z}/p\mathbf{Z}$ , et dans  $A$  un élément  $b$  tel que  $b^2 = a$ . Alors, d'après la proposition 5.5,  $(\frac{n}{p}) \pmod{p}$  est égal à  $a^{(p-1)/2} = b^{p-1} = b^p/b$ ; on notera que  $b$  est inversible, puisque  $a$  l'est. Par conséquent, on a  $(\frac{n}{p}) = 1$  si  $b^p = b$  et  $(\frac{n}{p}) = -1$  si  $b^p = -b$ . Il s'agit alors de calculer  $b^p$ . L'astuce consiste à exprimer  $b$  comme combinaison linéaire de racines de l'unité, et à utiliser le fait que l'application  $x \mapsto x^p$  respecte l'addition (corollaire 2.15).

### 5.2.1. Deux exemples

Avant de traiter le cas général, appliquons la méthode ci-dessus pour calculer  $(\frac{-3}{p})$ , en nous souvenant du fait que les racines cubiques de l'unité dans le corps des complexes font intervenir une racine de  $-3$ . Considérons donc l'anneau  $A$  obtenu en adjoignant au corps  $k = \mathbf{Z}/p\mathbf{Z}$  un élément  $\alpha$  avec  $\alpha^2 + \alpha + 1_k = 0$ , et donc  $\alpha^3 = 1_k$ . Ainsi  $\alpha$  est inversible et on a  $\alpha^{-1} = \alpha^2$ . Le discriminant du trinôme précédent est égal à  $-3$ , mais c'est aussi le carré de la différence des deux racines. Posons donc  $b = \alpha - \alpha^{-1}$ . On a effectivement

$$b^2 = (\alpha - \alpha^{-1})^2 = \alpha^2 + \alpha^{-2} - 2 = \alpha^2 + \alpha - 2 = -3 \cdot 1_k.$$

On en tire

$$\left(\frac{-3}{p}\right) = (-3)^{(p-1)/2} = (b^2)^{(p-1)/2} = \frac{b^p}{b}.$$

Mais, d'après le corollaire 2.15, on a  $b^p = (\alpha - \alpha^{-1})^p = \alpha^p - \alpha^{-p}$ . Comme on a  $\alpha^p = \alpha$  pour  $p \equiv 1 \pmod{3}$  et  $\alpha^p = \alpha^{-1}$  pour  $p \equiv -1 \pmod{3}$ , on voit que  $b^p$  vaut  $b$  si  $p \equiv 1 \pmod{3}$  et que  $b^p$  vaut  $-b$  si  $p \equiv -1 \pmod{3}$ , ce qu'on peut écrire  $b^p = b \left(\frac{p}{3}\right)$ . On a par conséquent

$$\left(\frac{-3}{p}\right) = \left(\frac{p}{3}\right).$$

C'est le premier cas de la loi de réciprocité quadratique, objet du théorème 5.20 ci-dessous. On a par exemple

$$\left(\frac{-3}{17}\right) = \left(\frac{17}{3}\right) = \left(\frac{2}{3}\right) = -1.$$

**REMARQUE 5.15.** — Traduisons : on a  $\left(\frac{3}{p}\right) = \left(\frac{-1}{p}\right) \left(\frac{-3}{p}\right) = \varepsilon(p) \left(\frac{-3}{p}\right)$ , donc  $\left(\frac{3}{p}\right) \equiv \varepsilon(p)p \pmod{3}$ . Puisque  $p$  est premier, il est congru à 1, 5, 7 ou 11 modulo 12, ce qui donne  $\left(\frac{3}{p}\right) = 1$  pour  $p \equiv \pm 1 \pmod{12}$  et  $\left(\frac{3}{p}\right) = -1$  pour  $p \equiv \pm 5 \pmod{12}$ .

**EXERCICE 5.12. [B]** — Si  $p = 2^n - 1$  est un nombre de Mersenne premier  $> 3$ , alors  $\left(\frac{3}{p}\right) = -1$ .

**EXERCICE 5.13. [C]** — Soit  $p$  un nombre premier. Quel est le nombre de racines cubiques de l'unité dans  $\mathbf{Z}/p\mathbf{Z}$ ? Quel est le nombre de résidus cubiques modulo  $p$ ?

Nous verrons ci-dessous comment le calcul précédent se généralise au cas de  $\left(\frac{\pm q}{p}\right)$ , où  $q$  est un nombre premier impair. Nous allons auparavant calculer  $\left(\frac{2}{p}\right)$ , car le cas du nombre premier 2 est comme à l'habitude un peu différent du cas général. Il s'agit de trouver un anneau dans lequel 2 est un carré. Pour cela, il faut aller jusqu'aux racines huitièmes de l'unité (le calcul des racines carrées complexes de  $i$  fait intervenir  $\sqrt{2}$ ).

On considère donc l'anneau  $A = (\mathbf{Z}/p\mathbf{Z})[X]/(X^4 + 1)$  et la classe  $\alpha$  de  $X$  dans  $A$ , qui satisfait par construction à la relation  $\alpha^4 + 1 = 0$ . Ainsi,  $\alpha$  est inversible et on a  $\alpha^{-1} = -\alpha^3$ . Posons  $b = \alpha + \alpha^{-1}$ . On a

$$b^2 = (\alpha + \alpha^{-1})^2 = \alpha^2 + \alpha^{-2} + 2 = \alpha^{-2}(\alpha^4 + 1) + 2 = 2,$$

donc  $\left(\frac{2}{p}\right) = (b^2)^{(p-1)/2} = b^p/b$ . Mais  $b^p = (\alpha + \alpha^{-1})^p = \alpha^p + \alpha^{-p}$ . Comme on a  $\alpha^p = \alpha$  pour  $p \equiv 1 \pmod{8}$ ,  $\alpha^p = \alpha^3 = -\alpha^{-1}$  pour  $p \equiv 3 \pmod{8}$ ,  $\alpha^p = \alpha^5 = -\alpha$  pour  $p \equiv 5 \pmod{8}$  et  $\alpha^p = \alpha^{-1}$  pour  $p \equiv 7 \pmod{8}$ , on voit que  $b^p$  vaut  $b$  si  $p$  est congru à 1 ou 7 modulo 8, et que  $b^p$  vaut  $-b$  si  $p$  est congru à 3 ou 5 modulo 8. Par conséquent :

**PROPOSITION 5.16.** — Soit  $p$  un nombre premier impair. On a  $\left(\frac{2}{p}\right) = \omega(p) = (-1)^{(p^2-1)/8}$ . Pour que 2 soit un carré modulo  $p$ , il faut et il suffit qu'on ait  $p \equiv \pm 1 \pmod{8}$ .

**EXERCICE 5.14. [C] —** Soit  $p$  un facteur premier du nombre de Fermat  $\text{Fer}_n = 2^{2^n} + 1$ , avec  $n \geq 2$ . Montrer qu'on a  $p \equiv 1 \pmod{2^{n+2}}$ .

### 5.2.2. Sommes de Gauss

Soit  $q$  un nombre premier impair. On peut calculer  $\sqrt{\pm q}$  comme combinaison de racines  $q$ -ièmes de l'unité. En effet :

**PROPOSITION 5.17 (Gauss).** — Soient  $q$  un nombre premier impair,  $\mathbf{A}$  un anneau et  $\alpha$  un élément de  $\mathbf{A}$  satisfaisant à la condition

$$\alpha^{q-1} + \alpha^{q-2} + \cdots + \alpha + 1 = 0.$$

On a  $\alpha^q = 1$ . Considérons la somme de Gauss

$$\tau = \sum_{i \in \mathbf{Z}/q\mathbf{Z}} \left(\frac{i}{q}\right) \alpha^i = \sum_{i=0}^{q-1} \left(\frac{i}{q}\right) \alpha^i = \sum_{i=1}^{q-1} \left(\frac{i}{q}\right) \alpha^i.$$

a) On a  $\tau^2 = \varepsilon(q)q$ .

b) Soit  $p$  un nombre premier impair distinct de  $q$ , et supposons qu'on ait dans  $\mathbf{A}$  la relation  $p\alpha = 0$ . On a alors

$$\tau^p = \left(\frac{p}{q}\right) \tau.$$

*Démonstration.* Multipliant la relation donnée par  $\alpha - 1$ , on trouve  $\alpha^q = 1$ .

Démontrons a). On a successivement (tous les indices de sommation varient dans  $\mathbf{Z}/q\mathbf{Z}$ ) :

$$\begin{aligned} \varepsilon(q)\tau^2 &= \left(\frac{-1}{q}\right) \tau \tau = \sum_{i,j} \left(\frac{-1}{q}\right) \left(\frac{i}{q}\right) \left(\frac{j}{q}\right) \alpha^i \alpha^j \\ &= \sum_{i,j} \left(\frac{-ij}{q}\right) \alpha^{i+j} = \sum_k \left( \sum_i \left(\frac{i(i-k)}{q}\right) \right) \alpha^k. \end{aligned}$$

Il s'agit donc de calculer les sommes

$$s_k = \sum_i \left(\frac{i(i-k)}{q}\right) = \sum_{i \neq 0} \left(\frac{i(i-k)}{q}\right).$$

Distinguons deux cas. Si  $k = 0$ , alors  $\left(\frac{i(i-k)}{q}\right) = \left(\frac{i^2}{q}\right) = 1$ , et on a  $s_0 = q - 1$ . Pour  $k \neq 0$ , on a  $s_k = -1$ ; admettons ce fait, que nous prouverons ci-dessous (lemme 5.18). On trouve alors

$$(-1)^{(q-1)/2} \tau^2 = (q-1) - \sum_{k \neq 0} \alpha^k = q - \sum_{k=0}^{q-1} \alpha^k = q.$$

Démontrons maintenant  $b$ ). Puisque  $p\alpha = 0$ , on peut appliquer le corollaire 2.15. On obtient successivement

$$\tau^p = \left( \sum_{i \in \mathbb{Z}/q\mathbb{Z}} \left( \frac{i}{q} \right) \alpha^i \right)^p = \sum_{i \in \mathbb{Z}/q\mathbb{Z}} \left( \frac{i}{q} \right)^p \alpha^{ip} = \sum_{i \in \mathbb{Z}/q\mathbb{Z}} \left( \frac{i}{q} \right) \alpha^{ip}.$$

Puisque  $p$  est inversible modulo  $q$ , l'application  $i \mapsto ip$  est bijective. On a par ailleurs  $\left( \frac{p}{q} \right) \left( \frac{i}{q} \right) = \left( \frac{ip}{q} \right)$ . Par conséquent :

$$\left( \frac{p}{q} \right) \tau^p = \sum_{i \in \mathbb{Z}/q\mathbb{Z}} \left( \frac{ip}{q} \right) \alpha^{ip} = \sum_{j \in \mathbb{Z}/q\mathbb{Z}} \left( \frac{j}{q} \right) \alpha^j = \tau,$$

ce qui achève la démonstration de  $b$ ).  $\square$

Il ne nous reste donc plus qu'à prouver le résultat admis au passage :

**LEMME 5.18.** — Soient  $q$  un nombre premier impair et  $k$  un entier avec  $k \not\equiv 0 \pmod{q}$ . On a

$$\sum_{i=1}^{q-1} \left( \frac{i(i-k)}{q} \right) = -1.$$

*Démonstration.* Notons  $i^{-1}$  l'inverse de  $i$  modulo  $q$ . On a

$$\left( \frac{i(i-k)}{q} \right) = \left( \frac{i^2(1-ki^{-1})}{q} \right) = \left( \frac{1-ki^{-1}}{q} \right).$$

Mais lorsque  $i$  parcourt les éléments non nuls de  $\mathbb{Z}/q\mathbb{Z}$ , l'élément  $1-ki^{-1}$  parcourt les éléments distincts de 1. Notre somme est donc la somme des  $\left( \frac{i}{q} \right)$  pour  $i \neq 1$ . Comme la somme étendue à tous les  $i$  est nulle, puisqu'il y a autant de carrés que de non-carrés dans  $(\mathbb{Z}/q\mathbb{Z})^*$ , on trouve  $-\left( \frac{1}{q} \right)$  qui vaut bien  $-1$ .  $\square$

**EXERCICE 5.15.** [B] — Pourquoi a-t-on  $\tau = \sum_{i=0}^{q-1} \alpha^{i^2}$  ?

**REMARQUE 5.19.** — Supposons  $A = C$ . On a donc suivant la valeur de  $\varepsilon(q)$  l'une des relations  $\tau = \pm \sqrt{q}$  ou  $\tau = \pm i \sqrt{q}$ . La détermination du signe, extrêmement subtile, est également due à Gauss.

### 5.2.3. La loi de réciprocité quadratique

Cette loi est due à Carl Friedrich GAUSS, qui en a donné plusieurs démonstrations, dont celle qui suit.

**THÉORÈME 5.20 (Gauss).** — Soient  $p$  et  $q$  deux nombres premiers impairs distincts. On a

$$\left( \frac{p}{q} \right) = \left( \frac{\varepsilon(q)q}{p} \right) = \theta(p, q) \left( \frac{q}{p} \right) = (-1)^{(p-1)(q-1)/4} \left( \frac{q}{p} \right).$$

En d'autres termes, on a  $\left( \frac{p}{q} \right) = \left( \frac{q}{p} \right)$  si  $p$  ou  $q$  est congru à 1 modulo 4, et  $\left( \frac{p}{q} \right) = -\left( \frac{q}{p} \right)$  dans le cas contraire.

*Démonstration.* Il s'agit de prouver la première égalité, les autres s'en déduisant immédiatement. Considérons l'anneau  $A$  obtenu en adjoignant au corps  $\mathbf{Z}/p\mathbf{Z}$  un élément  $\alpha$  soumis à la relation  $\Phi_q(\alpha) = 0$  avec

$$\Phi_q(X) = (X^q - 1)/(X - 1) = X^{q-1} + \cdots + 1.$$

Considérons dans  $A$  la somme de Gauss

$$\tau = \sum_{i \in \mathbf{Z}/q\mathbf{Z}} \left(\frac{i}{q}\right) \alpha^i$$

On a par construction  $\Phi_q(\alpha) = 0$  et évidemment  $p\alpha = 0$ . On peut donc appliquer la proposition 5.17. On a  $\tau^2 = \varepsilon(q)q$ . On en déduit  $(\frac{\varepsilon(q)q}{p}) = (\tau^2)^{(p-1)/2} = \tau^{p-1}$ . Mais, puisque qu'on a aussi  $\tau^p = (\frac{p}{q})\tau$ , cela donne  $(\frac{\varepsilon(q)q}{p})\tau = (\frac{p}{q})\tau$ . Il ne reste plus qu'à noter que  $\tau$  est inversible dans  $A$ , puisque son carré l'est ( $q$  est premier à  $p$ ).  $\square$

On a par exemple :

$$\left(\frac{5}{p}\right) = \left(\frac{p}{5}\right) \equiv p^2 \pmod{5}.$$

**EXERCICE 5.16. [B]** — Soit  $p = 2^{2^n} + 1$  un nombre de Fermat *premier*, avec  $n > 2$ . Montrer que  $\left(\frac{3}{p}\right) = \left(\frac{5}{p}\right) = \left(\frac{7}{p}\right) = -1$ . En appliquant le lemme 3.22, en déduire que les conditions suivantes sont équivalentes :

- (i) le nombre de Fermat  $\text{Fer}_n = 2^{2^n} + 1$  est premier ( $n > 1$ ),
- (ii) on a  $3^{2^{2^n}-1} \equiv -1 \pmod{\text{Fer}_n}$ ,
- (iii) on a  $5^{2^{2^n}-1} \equiv -1 \pmod{\text{Fer}_n}$ ,
- (iv) on a  $7^{2^{2^n}-1} \equiv -1 \pmod{\text{Fer}_n}$ .

### § 5.3. Symbole de Jacobi

#### 5.3.1. Définition et réciprocité

Pour utiliser pleinement la puissance du théorème précédent, il est commode d'étendre le domaine de définition du symbole de Legendre en introduisant, avec la même notation, le *symbole de Jacobi*  $(\frac{m}{n})$ . Il est défini pour deux entiers  $n$  et  $m$  avec  $n \geq 3$  impair. On l'obtient ainsi : on décompose  $n$  en produit de facteurs premiers (répétés éventuellement), soit  $n = p_1 \cdots p_r$ , et on pose  $(\frac{m}{n}) = (\frac{m}{p_1}) \cdots (\frac{m}{p_r})$ . C'est un entier égal à 0, 1 ou  $-1$ . Il vaut 0 lorsque  $n$  et  $m$  ne sont pas premiers entre eux. Dans le cas contraire, il vaut 1 ou  $-1$ . Il ne dépend que de la classe de  $m$  modulo  $n$  et satisfait aux propriétés suivantes de multiplicativité :

$$\left(\frac{mm'}{n}\right) = \left(\frac{m}{n}\right)\left(\frac{m'}{n}\right), \quad \left(\frac{m}{nn'}\right) = \left(\frac{m}{n}\right)\left(\frac{m}{n'}\right).$$

On a en particulier, lorsque  $n$  et  $m$  sont premiers entre eux,

$$\left(\frac{m^2}{n}\right) = 1, \quad \left(\frac{m}{n^2}\right) = 1.$$

On a par exemple

$$\left(\frac{14}{51}\right) = \left(\frac{14}{3}\right)\left(\frac{14}{17}\right) = \left(\frac{-1}{3}\right)\left(\frac{-3}{17}\right) = (-1)(-1) = 1.$$

**REMARQUE 5.21.** — Kronecker a introduit une généralisation du symbole de Jacobi au cas où  $n$  n'est pas nécessairement impair. C'est un peu plus compliqué. Comme nous nous intéressons en fait à des critères de primalité pour  $n$ , le cas où  $n$  est impair nous suffit.

**PROPOSITION 5.22** (Loi de réciprocité). — *Soient  $n$  et  $m$  deux entiers impairs, positifs et premiers entre eux. On a*

$$\left(\frac{m}{n}\right) = \left(\frac{\varepsilon(n)n}{m}\right) = \theta(m, n) \left(\frac{n}{m}\right) = (-1)^{(m-1)(n-1)/4} \left(\frac{n}{m}\right).$$

*Démonstration.* Lorsque  $n$  et  $m$  sont premiers, c'est le théorème de réciprocité de Gauss. Mais  $n$  et  $m$  sont produits de nombres premiers, et les membres de la formule sont multiplicatifs relativement à  $n$  et à  $m$ .  $\square$

**PROPOSITION 5.23** (Lois complémentaires). — *Soit  $n$  un entier impair positif. On a*

$$\left(\frac{-1}{n}\right) = \varepsilon(n) = (-1)^{(n-1)/2}, \quad \left(\frac{2}{n}\right) = \omega(n) = (-1)^{(n^2-1)/8}.$$

*Démonstration.* En effet, cela est vrai lorsque  $n$  est premier. Le cas général s'en déduit, puisque les diverses expressions considérées dépendent multiplicativement de  $n$ .  $\square$

**EXERCICE 5.17. [B]** — Montrer que le symbole de Jacobi  $\left(\frac{m}{n}\right)$  ne dépend que de  $m$  modulo  $n$  et de  $n$  modulo  $4|m|$ .

Il est important de signaler explicitement deux confusions à éviter, toutes deux dues à l'extension abusive de propriétés du symbole de Legendre au symbole de Jacobi.

En premier lieu, si  $m$  est un résidu quadratique modulo  $n$ , donc s'il existe un entier  $a$  (premier à  $n$ ) avec  $m \equiv a^2 \pmod{n}$ , on a  $\left(\frac{m}{n}\right) = \left(\frac{a^2}{n}\right) = 1$ . La réciproque est vraie (par définition) lorsque  $n$  est premier, mais pas dans le cas général. On a par exemple  $\left(\frac{14}{51}\right) = 1$ , mais 14 n'est pas un résidu quadratique modulo 51 (puisque c'est déjà faux modulo 3).

**EXERCICE 5.18. [A]** — Montrer que, pour que  $m$  soit un résidu quadratique modulo  $n$ , il faut et il suffit que  $\left(\frac{m}{p}\right) = 1$  pour tout diviseur premier  $p$  de  $n$ .

La seconde confusion possible est la suivante. Lorsque  $n$  est premier (impair), le symbole de Jacobi coïncide avec le symbole de Legendre, et peut se calculer par la formule d'Euler  $\left(\frac{m}{n}\right) \equiv m^{(n-1)/2} \pmod{n}$ . Dans le cas général, il n'y a *a priori* pas de rapport entre le symbole de Jacobi et

l'expression  $m^{(n-1)/2} \bmod n$ . Cette remarque est à la base du test de primalité de Solovay et Strassen dont nous allons parler un peu plus loin.

Par exemple, bien que  $(\frac{14}{51}) = 1$ , on a  $14^{25} \equiv 20 \pmod{51}$  : cela peut se vérifier par un calcul direct, mais aussi comme suit. On a modulo 3 les congruences  $14^{25} \equiv (-1)^{25} \equiv -1$ . Modulo 17, on a  $14^{25} \equiv (-3)^{25} \equiv (-3)^9$  d'après Fermat et  $(-3)^9 \equiv (-3)(-3)^8 \equiv -3(\frac{-3}{17}) = 3$ . Le résultat cherché est donc congru à  $-1$  modulo 3 et à 3 modulo 17. C'est bien 20 (théorème chinois).

### 5.3.2. Algorithmes de calcul du symbole de Jacobi

Le symbole de Jacobi peut se calculer par un algorithme rapide, parallèle à l'algorithme d'Euclide et basé sur la loi de réciprocité quadratique. Cet algorithme part d'un couple  $(u, v)$  avec  $v \geq 3$  impair, et aboutit au symbole de Jacobi  $(\frac{u}{v})$ . Son coût est en  $(\log n)^2$ . Le voici :

---

**ALGORITHME 5.24. — Symbole de Jacobi — algorithme euclidien**

Entrées : entier  $u$ , entier impair  $v \geq 3$ . Sorties :  $(\frac{u}{v})$ .

Règles :

$[u \geq 0]$ :	$\text{jacobi}(u, v)$	$\mapsto$	$(u, v, 1)$
$[u < 0]$ :	$\text{jacobi}(u, v)$	$\mapsto$	$(-u, v, \varepsilon(v))$
$[u \text{ pair } > 0]$ :	$(u, v, \varepsilon)$	$\mapsto$	$(u/2, v, \varepsilon\omega(v))$
$[u \text{ impair } > 1]$ :	$(u, v, \varepsilon)$	$\mapsto$	$(v - (v \div u)u, u, \varepsilon\theta(u, v))$
$[u = 1]$ :	$(u, v, \varepsilon)$	$\mapsto$	$\varepsilon$
$[u = 0]$ :	$(u, v, \varepsilon)$	$\mapsto$	0

---

**EXERCICE 5.19. [B] — Démontrer la validité de cet algorithme.**

En Ruby, cela donne par exemple, en traduisant de façon directe :

---

**PROGRAMME 5.25. — Calcul du symbole de Jacobi**

```
class Integer
  def epsilon?
    self % 4 == 1
  end
  def omega?
    self % 8 == 1 || self % 8 == 7
  end
end

def theta?(u,v)
  u.epsilon? or v.epsilon?
end

def jacrec(u,v,a)
  case
  when u == 0 : 0
  when u == 1 : a
  else
    if v <= 0
      jacrec(-u,-v,a)
    else
      if u.even?
        jacrec(u/2,v,a)
      else
        if v.even?
          jacrec((v-(v/u)*u),u,a)
        else
          if theta?(u,v)
            jacrec(-u,-v,a)
          else
            a * jacrec(1,v,a)
          end
        end
      end
    end
  end
end
```

```

when u.even?
    jacrec(u.half, v, (v.omega? ? a : -a))
else
    jacrec(v % u, u, (theta?(u,v) ? a : -a))
end
end

def jacobi(u,v)
    raise ArgumentError, "#{v} est pair !" if v.even?
    u>0 ? jacrec(u,v,1) : jacrec(-u,v, (v.epsilon? ? 1 : -1))
end

jacobi(14,51) # => 1

```

---

Comme dans le cas du pgcd, on peut donner une variante n'utilisant pas la division euclidienne, mais uniquement la division par 2 : comme le précédent, cet algorithme part d'un couple  $(u, v)$  avec  $v \geq 3$  impair, et aboutit au symbole de Jacobi  $(\frac{u}{v})$ .

**ALGORITHME 5.26. — Symbole de Jacobi — algorithme binaire**

---

*Entrées :* entier  $u$ , entier impair  $v \geq 3$ . *Sorties :*  $(\frac{u}{v})$ .

Règles :

$[u = 0]$ :	$\text{jacobi}(u, v)$	$\mapsto 0$
$[u > 0]$ :	$\text{jacobi}(u, v)$	$\mapsto (u, v, 1)$
$[u < 0]$ :	$\text{jacobi}(u, v)$	$\mapsto (-u, v, \varepsilon(v))$
$[u \text{ pair}]$ :	$(u, v, \varepsilon)$	$\mapsto (u/2, v, \varepsilon\omega(v))$
$[u \text{ impair} > v]$ :	$(u, v, \varepsilon)$	$\mapsto ((u-v)/2, v, \varepsilon\omega(v))$
$[u \text{ impair} < v]$ :	$(u, v, \varepsilon)$	$\mapsto ((v-u)/2, u, \varepsilon\omega(u)\theta(u, v))$
$[1 < u = v]$ :	$(u, v, \varepsilon)$	$\mapsto 0$
$[u = 1]$ :	$(u, v, \varepsilon)$	$\mapsto \varepsilon$

---

La terminaison de cet algorithme résulte de ce qu'en dehors des trois règles initiales, dont une et une seule est exécutée au début et des deux règles finales, dont une et une seule est exécutée à la fin, les trois règles essentielles divisent au moins par 2 le produit  $uv$ . Il n'y a donc qu'au plus  $(\log u + \log v + 2)$  pas, dont chacun consiste au plus en une soustraction, une division par 2 d'un entier pair et deux changements de signe.

La correction de l'algorithme vient de ce que les trois règles essentielles, opérant sur les triplets  $(u, v, \varepsilon)$ , respectent les inégalités  $u > 0$  et  $v > 2$  et la congruence  $v \equiv 1 \pmod{2}$ , et laissent invariant l'entier  $(\frac{u}{v})\varepsilon$ .

On obtient par exemple la réécriture suivante :

$$\begin{aligned} \text{jacobi}(14, 51) &\mapsto (14, 51, 1) \mapsto (7, 51, -1) \mapsto (22, 7, 1) \mapsto \\ &\mapsto (11, 7, 1) \mapsto (2, 7, 1) \mapsto (1, 7, 1) \mapsto 1 \end{aligned}$$

qui implique  $(\frac{14}{51}) = 1$ .

En Ruby, cela donne, en supprimant les définitions intermédiaires :

PROGRAMME 5.27. — *Calcul du symbole de Jacobi, variante binaire* ——————

```

def jacrec(u,v,a)
  case
    when u == v : 0
    when u == 1 : a
    when u.even?
      jacrec(u.half, v, (v%8==1||v%8==7 ? a : -a))
    when u > v
      jacrec((u-v).half, v, (v%8==1||v%8==7 ? a : -a))
    else
      sign = (u%8==1||u%8==7 ? 1 : -1) * (u%4==1||v%4==1 ? 1 : -1)
      jacrec((v-u).half, u, sign*a)
    end
  end
end
def jacobi(u,v)
  raise ArgumentError, "#{v} est pair !" if v.even?
  u>0 ? jacrec(u,v,1) : jacrec(-u,v,(v%4 == 1 ? 1 : -1))
end

```

---

## 5.3.3. Le critère de Solovay et Strassen

Soit  $n$  un entier impair  $> 2$ . Si  $n$  est premier, on a  $(\frac{a}{n}) \equiv a^{(n-1)/2} [\text{mod } n]$  pour tout entier  $a$  premier à  $n$ . Inversement :

**PROPOSITION 5.28** (Solovay-Strassen). — *Soit  $n$  un entier impair  $> 2$  tel que  $(\frac{a}{n}) \equiv a^{(n-1)/2} [\text{mod } n]$  pour tout entier  $a$  premier à  $n$ . Alors  $n$  est premier.*

*Démonstration.* On a d'abord pour tout  $a$  premier à  $n$  la relation  $a^{n-1} \text{ mod } n = (\frac{a}{n})^2 = 1$ . Ainsi, si  $n$  n'est pas premier, c'est un nombre de Carmichael (définition 3.34), donc un produit de facteurs premiers tous distincts (proposition 3.35), soit  $n = p_1 \cdots p_r$ , avec  $r \neq 1$ . Soit  $a$  un entier premier à  $m$ , c'est-à-dire premier à chacun des  $p_i$ ; notons  $\alpha_i$  sa classe modulo  $p_i$ .

On a d'une part

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right) \cdots \left(\frac{a}{p_r}\right) = \left(\frac{\alpha_1}{p_1}\right) \cdots \left(\frac{\alpha_r}{p_r}\right).$$

D'autre part, réduisant modulo  $p_1$  l'égalité initiale, on obtient

$$\left(\frac{a}{n}\right) \equiv \alpha_1^{(n-1)/2} [\text{mod } p_1].$$

Comparant, on obtient

$$\left(\frac{\alpha_1}{p_1}\right) \cdots \left(\frac{\alpha_r}{p_r}\right) \equiv \alpha_1^{(n-1)/2} [\text{mod } p_1].$$

D'après le théorème chinois, on peut choisir les différents  $\alpha_i$  indépendamment. Mais le second membre ne dépend que de  $\alpha_1$ , tandis qu'on peut changer le signe du premier sans modifier  $\alpha_1$ , par exemple en modifiant  $\alpha_r$ . Cela est absurde et contredit l'hypothèse  $r \neq 1$ .  $\square$

Notons maintenant que l'ensemble des  $a \in (\mathbf{Z}/n\mathbf{Z})^*$  tels que  $(\frac{a}{n}) = a^{(n-1)/2}$  est un sous-groupe de  $(\mathbf{Z}/n\mathbf{Z})^*$ , puisque les deux membres de l'égalité à vérifier dépendent multiplicativement de  $a$ . Or un sous-groupe d'un groupe à  $r$  éléments, qui n'est pas le groupe entier, possède au plus  $r/2$  éléments. Cela résulte de la forme forte du théorème de Lagrange : l'ordre d'un sous-groupe divise l'ordre du groupe, mais cela peut aussi se prouver directement. Notons en effet  $G$  le groupe et  $G'$  le sous-groupe et soit  $a \notin G'$ . Alors, pour tout  $g \in G'$ , on a  $ag \notin G'$ ; il en résulte que  $G'$  et  $aG'$  sont disjoints, ce qui implique l'assertion cherchée. Par conséquent :

**COROLLAIRE 5.29.** — Soit  $n$  un entier impair > 2.

a) Si  $n$  est premier, on a  $(\frac{a}{n}) \equiv a^{(n-1)/2} \pmod{n}$  pour tout entier  $a$  premier à  $n$ .

b) Si  $n$  est composé, l'ensemble des  $a$  premiers à  $n$  tels que  $0 < a < n$  et  $(\frac{a}{n}) \equiv a^{(n-1)/2} \pmod{n}$  a au plus  $\varphi(n)/2$  éléments.

Notons que,  $a$  étant choisi, le test précédent demande le calcul du symbole de Jacobi  $(\frac{a}{n})$  et de  $a^{(n-1)/2} \pmod{n}$ . Son coût est donc en  $(\log n)^3$ . Dans un langage déjà utilisé, on dira que  $a$  est un *témoin de Solovay* si l'on n'a pas  $(\frac{a}{n}) \equiv a^{(n-1)/2} \pmod{n}$ . L'existence d'un tel témoin implique que  $n$  est composé. En Ruby, cela donne :

**PROGRAMME 5.30.** — Test de Solovay

---

```
class Integer
  def is_a_Solovay_witness_for(n)
    raise ArgumentError, "#{n} est pair !" if n.even?
    self.puissmod((n-1).half, n) != jacobi(self, n) % n
  end
end

pseudo = 2251*11251 # => 25326001
z = [2,3,5,7].collect{|a| a.is_a_Solovay_witness_for(pseudo)}
# => [false, false, false, true]

mersenne = 2**67 -1 # => 147573952589676412927
z = [2,3,5,7].collect{|a| a.is_a_Solovay_witness_for(mersenne)}
# => [false, true, true, true]
```

---

### 5.3.4. Tests probabilistes de primalité

Ces tests dérivent des critères de non-primalité et sont basés sur l'idée qu'un nombre qui survit à beaucoup d'instances de ces critères a une bonne chance d'être premier. De façon plus précise, on veut savoir si un entier (impair, sinon c'est trop facile !)  $n \geq 3$  est premier à l'aide d'un critère de non-primalité, disons  $P(a, n)$ , qui utilise un « témoin »  $a$  avec  $0 < a < n$ . On

fixe un nombre  $\alpha$  avec  $0 \leq \alpha < 1$ , et on suppose que  $P$  satisfait aux deux propriétés suivantes :

a) Si  $n$  est premier, alors  $P(a, n)$  est vrai pour tout  $a$  compris entre 1 et  $n - 1$ .

b) Si  $n$  est composé, alors, sur les  $n - 1$  entiers  $a$  compris entre 1 et  $n - 1$ , il y en a au plus  $\alpha \cdot (n - 1)$  pour lesquels  $P(a, n)$  est vrai.

Naturellement, ce qui va suivre n'a d'intérêt pratique que si le test  $P(a, n)$  est rapide, ce qui veut dire qu'un algorithme en donne la réponse en un temps majoré par une expression du type  $A(\log n)^r$ .

Nous avons rencontré deux exemples :

1) *Le test de Miller et Rabin* (corollaire 2.21) : on écrit  $n - 1 = 2^s t$  avec  $t$  impair, et on considère la propriété suivante :

$$P_{\text{MR}}(a, n) \Leftrightarrow \left( a^t \equiv 1 \pmod{n} \text{ ou il existe } i \in [0, s[ \text{ avec } a^{2^i t} \equiv -1 \pmod{n} \right).$$

On prend  $\alpha = 1/4$  (théorème de Rabin 2.24) ; si  $n$  est composé, bien que  $P(a, n)$  soit vraie, on dit souvent que  $n$  est un « *strong pseudoprime for base a* ».

2) *Le test de Solovay et Strassen* (corollaire 5.29) : on pose

$$P_{\text{SS}}(a, n) \Leftrightarrow \left( \text{pgcd}(a, n) = 1 \text{ et } \left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n} \right).$$

On prend  $\alpha = 1/2$  (corollaire 5.29) ; si  $n$  est composé, bien que  $P(a, n)$  soit vraie, on dit souvent que  $n$  est un « *Euler pseudoprime for base a* ».

Cela étant, l'algorithme est le suivant. On fixe un entier  $N$  et on répète  $N$  fois l'opération suivante :

- ▷ on choisit au hasard un entier  $a$  avec  $0 < a < n$  et on exécute le test  $P(a, n)$  ;
- ▷ si le test échoue,  $n$  est composé et on a terminé ;
- ▷ si le test réussit, on recommence.

Si au bout de  $N$  fois, le test n'a jamais échoué, on déclare que  $n$  est premier. Naturellement, cela ne donne aucune certitude. Mais si  $n$  est composé, chaque test a une probabilité de réussite  $\leq \alpha$ , et une série de  $N$  tests a une probabilité de réussite  $\leq \alpha^N$ . Ainsi, le « risque d'erreur » est majoré par  $\alpha^N$ . Pour fixer les idées, 50 répétitions du test de Solovay-Strassen donnent un risque d'erreur inférieur à  $2^{-50} < 10^{-15}$ .

**REMARQUE 5.31.** — Notons bien qu'il s'agit là de la probabilité pour qu'un nombre composé soit déclaré premier après  $N$  répétitions du test. C'est donc une probabilité conditionnelle, qu'il ne faut pas confondre avec la probabilité d'erreur de l'algorithme lui-même, erreur classique dans ce genre de questions (les « faux positifs » des tests médicaux). On trouvera des détails dans [Delahaye], page 310.

En Ruby, cela donne par exemple ce qui suit :

PROGRAMME 5.32. — *Tests probabilistes de primalité*

```

def repeat_random_test(k, n, &test)
  k.times do
    a = 1 + rand(n-1)
    return a if test[a]
  end
  return nil
end

def find_Solovay_witness(k,n)
  repeat_random_test(k, n) { |a| a.is_a_Solovay_witness_for(n)}
end

def find_Miller_witness(k,n)
  repeat_random_test(k, n) { |a| a.is_a_Miller_witness_for(n)}
end

find_Miller_witness(50, 2**61 - 1) # => nil
find_Solovay_witness(50, 2**67 - 1) # => 105518523544836389858
find_Miller_witness(50, 2**257 - 1) # => 875103766

```

On voit ainsi que les nombres de Mersenne  $M_{67}$  et  $M_{257}$  ne sont pas premiers, tandis que  $M_{61} = 2^{61} - 1$  l'est très probablement, choses que nous savions déjà par le test de Lucas. Nous avons un peu triché pour  $M_{257}$ . Notre programme est purement démonstratif, car donner explicitement un témoin n'a aucun intérêt. Comme un nombre tiré au sort entre 1 et  $M_{257}$  a toutes les chances d'avoir presque autant de chiffres que ce dernier, le résultat est désagréable à imprimer et à regarder. Pour exhiber un témoin de petite taille, nous avons donc exécuté le programme précédent en y remplaçant la ligne `a = 1 + rand(n-1)` par `a = 1 + rand(2**50)`.

De même, si on veut décomposer  $x = 1 + 12!$  dont on sait déjà qu'il est divisible par 13 (« théorème de Wilson », voir l'exercice 2.16), on obtient

```

x = (1 + 12.fact)
find_Miller_witness(50, x/13) # => 23805021

```

Donc  $x$  n'est pas premier. Utilisant l'algorithme de Pollard (1.6.5) on obtient un deuxième facteur 13, puis un quotient probablement premier.

```

y = (1 + 12.fact)/13
rho(y) # => 13
y/13 # => 2834329
find_Miller_witness(50, y/13) # => nil

```

### 5.3.5. Comparaison des algorithmes de Miller-Rabin et de Solovay-Strassen

En fait, l'algorithme de Miller-Rabin est toujours meilleur que celui de Solovay-Strassen : non seulement parce que  $1/4 < 1/2$ , mais aussi parce que

la propriété  $P_{MR}$  est plus forte que  $P_{SS}$  comme on va le voir. Posons donc  $n - 1 = 2^s t$  avec  $t$  impair.

Traitons d'abord comme exemple le cas le plus simple, celui où  $n \equiv 3 \pmod{4}$ , donc  $s = 1$ . Alors  $P_{MR}(a, n)$  s'écrit  $a^t \equiv \pm 1 \pmod{n}$ , tandis que  $P_{SS}(a, n)$  s'écrit  $a^t \equiv \left(\frac{a}{n}\right) \pmod{n}$ . Il s'agit en fait de prouver que ces propriétés sont équivalentes. Or on a d'abord

$$\left(\frac{a}{n}\right) = \left(\frac{a(a^{(t-1)/2})^2}{n}\right) = \left(\frac{a^t}{n}\right).$$

Mais, puisque  $n \equiv 3 \pmod{4}$ , on a  $\left(\frac{-1}{n}\right) = -1$ . Si  $a^t \equiv \pm 1$ , on a donc  $\left(\frac{a^t}{n}\right) \equiv a^t$ , soit encore  $\left(\frac{a}{n}\right) \equiv a^t$ , c'est-à-dire  $P_{SS}(a, n)$ .

Passons au cas général :

**PROPOSITION 5.33.** — *On a  $P_{MR}(a, n) \Rightarrow P_{SS}(a, n)$ .*

*Démonstration.* Supposons d'abord que  $a^t \equiv 1 \pmod{n}$ . Puisque  $t$  est impair, on a  $\left(\frac{a}{n}\right) = \left(\frac{a}{n}\right)^t = \left(\frac{a^t}{n}\right) = \left(\frac{1}{n}\right) = 1$ . Par ailleurs, on a  $(n-1)/2 = 2^{s-1}t$ , donc  $a^{(n-1)/2} \equiv 1 \pmod{n}$ , d'où  $P_{SS}(a, n)$ .

Supposons maintenant que  $a^{2^i t} \equiv -1 \pmod{n}$  avec  $0 \leq i \leq s-1$ . Alors, ou bien  $i < s-1$  et alors  $a^{(n-1)/2} \equiv 1 \pmod{n}$ , ou bien  $i = s-1$  et alors  $a^{(n-1)/2} \equiv -1 \pmod{n}$ . Il s'agit donc de prouver que  $\left(\frac{a}{n}\right) = 1$  si  $i < s-1$  et  $\left(\frac{a}{n}\right) = -1$  si  $i = s-1$ .

Or, soit  $p$  un diviseur premier de  $n$ . Posons  $p-1 = 2^u v$  avec  $v$  impair. On a d'une part  $a^{2^i t} \equiv -1 \pmod{p}$ , donc  $a^{2^i t v} \equiv -1 \pmod{p}$  puisque  $v$  est impair, et d'autre part  $a^{2^{u-1} v} \equiv \left(\frac{a}{p}\right) \pmod{p}$ , donc  $a^{2^{u-1} t v} \equiv \left(\frac{a}{p}\right) \pmod{p}$  puisque  $t$  est impair. Cela implique que l'on a nécessairement  $u \geq i+1$ , avec  $\left(\frac{a}{p}\right) = 1$  si  $u > i+1$  et  $\left(\frac{a}{p}\right) = -1$  si  $u = i+1$ . Dans le premier cas, on a  $p \equiv 1 \pmod{2^{i+2}}$ ; dans le second cas, on a  $p \equiv 1 + 2^{i+1} \pmod{2^{i+2}}$ . Si on note  $k$  le nombre des diviseurs premiers  $p$  de  $n$  (comptés avec leurs multiplicités) pour lesquels l'entier  $v$  correspondant vaut  $i+1$ , on en déduit d'une part que  $\left(\frac{a}{n}\right)$ , qui est le produit des  $\left(\frac{a}{p}\right)$ , vaut  $(-1)^k$ , d'autre part que l'on a  $n \equiv (1 + 2^{i+1})^k \pmod{2^{i+2}}$ . Mais, puisque  $(1 + 2^{i+1})^k \equiv 1 + k2^{i+1} \pmod{2^{i+2}}$ , on a  $n \not\equiv 1 \pmod{2^{i+2}}$  si  $k$  est impair et  $n \equiv 1 \pmod{2^{i+2}}$  si  $k$  est pair. Or  $n = 2^s t$  avec  $t$  impair. On a donc  $s = i+1$  si  $k$  est impair, et  $s > i+1$  si  $k$  est pair. Comme  $\left(\frac{a}{n}\right) = (-1)^k$ , on obtient bien les résultats recherchés.  $\square$

## § 5.4. Algorithmes probabilistes

### 5.4.1. Parties de type $\mathcal{P}$

Une fonction  $f : \mathbf{R}_+ \rightarrow \mathbf{R}$  sera dite à croissance (au plus) polynomiale s'il existe un entier  $r \geq 0$  tel que pour  $t \in \mathbf{R}_+$  assez grand, on ait  $f(t) \leqslant t^r$ . Il revient au même de dire qu'il existe des entiers  $A$  et  $r$  avec  $f(t) \leqslant A + t^r$ , ou encore qu'il existe un polynôme  $P$  tel que  $f(t) \leqslant P(t)$  pour tout  $t$ . Par exemple, les fonctions  $t^3$ ,  $t \log t$ ,  $\sqrt{t}$  sont à croissance polynomiale.

L'ensemble des fonctions à croissance polynomiale est stable par addition, multiplication et composition.

Comme précédemment, nous mesurerons la taille  $\|\mathbf{n}\|$  d'un entier  $n > 0$  par son logarithme binaire (on conviendra, ce qui n'a du reste aucune importance, que  $\|0\| = 0$ ). De même la taille  $\|\mathbf{n}\|$  d'une suite finie  $\mathbf{n} = (n_1, \dots, n_s)$  d'entiers  $\geq 0$  sera la somme  $\|n_1\| + \dots + \|n_s\|$ .

On s'intéresse à des parties de  $\mathbb{N}$  ou plus généralement de  $\mathbb{N}^s$ , où  $s$  est un entier  $> 0$ . Il n'existe en fait qu'une notion vraiment raisonnable de partie calculable, c'est-à-dire de partie  $A$  pour laquelle il existe un algorithme pouvant répondre par oui ou non à la question : la famille d'entiers  $\mathbf{n} = (n_1, \dots, n_s)$  appartient-elle à  $A$  ? La « thèse de Church » (c'est une thèse au sens philosophique du terme, ce n'est donc pas un énoncé formel) affirme en effet que, quel que soit le modèle retenu pour la notion d'algorithme et pour la machine qui les exécute (dès lors qu'il s'agit d'une machine déterministe à mémoire illimitée), on obtient en définitive le même ensemble de parties calculables. Ce sont mathématiquement les parties *récursives*.

Une partie  $A$  de  $\mathbb{N}^s$  est dite de type  $\mathcal{P}$  s'il existe un algorithme permettant de décider de l'appartenance d'une famille  $\mathbf{n}$  à  $A$ , et dont le temps d'exécution est une fonction à croissance polynomiale de  $\|\mathbf{n}\|$ . La « thèse de Church généralisée » affirme que cette notion est, comme ci-dessus, indépendante du modèle de calcul retenu ; ce n'est que lorsqu'on veut être plus précis, et parler par exemple d'algorithmes en  $\|\mathbf{n}\|^2$ , qu'il faut préciser le modèle choisi.

Par exemple, la partie  $D$  de  $\mathbb{N}^3$  formée des triplets  $(a, b, c)$  avec  $a > 1$ ,  $b > 1$ ,  $c > 1$  et  $a = bc$  est de type  $\mathcal{P}$ .

**EXERCICE 5.20. [A]** — Vérifier ce point.

**EXERCICE 5.21. [A]** — Prouver que toute partie finie est de type  $\mathcal{P}$ .

**EXERCICE 5.22. [B]** — Prouver que l'ensemble des parties de type  $\mathcal{P}$  est stable par réunion finie, intersection finie, complémentation, produit cartésien.

#### 5.4.2. Parties de type $\mathcal{NP}$

Il existe plusieurs définitions (équivalentes) des parties de type  $\mathcal{NP}$ . La plus commode pour nous est la suivante :

**DÉFINITION 5.34.** — *Une partie  $A$  de  $\mathbb{N}^s$  est de type  $\mathcal{NP}$  s'il existe un entier  $t \geq 0$ , une partie  $B$  de  $\mathbb{N}^s \times \mathbb{N}^t$  qui est de type  $\mathcal{P}$ , et une fonction à croissance polynomiale  $f$  avec les deux propriétés suivantes :*

- ▷ si  $\mathbf{n} \in \mathbb{N}^s$  appartient à  $A$ , il existe  $\mathbf{a} \in \mathbb{N}^t$  avec  $(\mathbf{n}, \mathbf{a}) \in B$  et  $\|a\| \leq f(\|\mathbf{n}\|)$ ;
- ▷ si  $\mathbf{n} \in \mathbb{N}^s$  n'appartient pas à  $A$ , il n'existe aucun  $\mathbf{a} \in \mathbb{N}^t$  avec  $(\mathbf{n}, \mathbf{a}) \in B$ .

En termes imagés, d'une part l'appartenance de  $\mathbf{n}$  à A peut se décider par l'existence d'un « témoin »  $\mathbf{a}$ , et d'autre part la taille de ce témoin ainsi que le temps de vérification sont à croissance polynomiale.

Par exemple, l'ensemble C des nombres composés est de type  $\mathcal{NP}$  puisque l'appartenance de  $a$  à C équivaut à l'existence d'un triplet  $(a, b, c)$  appartenant à la partie D définie plus haut, et que dans ce cas, on a  $\|b\| < \|a\|$  et  $\|c\| < \|a\|$ .

Il est évident que toute partie de type  $\mathcal{P}$  est de type  $\mathcal{NP}$ . Par ailleurs, avec les notations de la définition, il existe par hypothèse un algorithme de décision polynomial pour B ; comme la taille d'un témoin possible  $\mathbf{a}$  pour un entier  $\mathbf{n}$  est majorée par une expression de la forme  $\|\mathbf{n}\|^r$ , le nombre de témoins possibles est majoré par une expression de la forme  $2^{\|\mathbf{n}\|^r}$  ; la recherche exhaustive d'un témoin donne ainsi un algorithme de décision pour A, à temps d'exécution a priori *exponentiel* en  $\|\mathbf{n}\|$ .

D'autres interprétations de la définition ci-dessus sont utiles à connaître. On peut par exemple imaginer une machine « non déterministe » (c'est la signification du N de l'expression  $\mathcal{NP}$ ), ou penser à un jeu entre deux joueurs, l'un qui possède une puissance de calcul infinie et qui propose le témoin, l'autre qui a une puissance polynomiale et vérifie. On peut aussi parler de preuve et de vérification de preuve. Il nous est impossible d'en dire plus ici, et nous renvoyons le lecteur intéressé aux ouvrages spécialisés de théorie de la complexité ou d'informatique.

## Le problème $\mathcal{P} \neq \mathcal{NP}$

La classe  $\mathcal{NP}$  est énorme et couvre la plus grande partie des questions « difficiles » classiques (problèmes de graphes, voyageur de commerce, etc.) pour lesquelles on ne connaît pas d'algorithme déterministe polynomial. Cela rend extrêmement vraisemblable la conjecture selon laquelle il existe des parties de type  $\mathcal{NP}$  qui ne sont pas déjà de type  $\mathcal{P}$ . On a des quantités de candidats (les parties dites  $\mathcal{NP}$ -complètes), mais on ne sait pas prouver qu'elles ne sont pas de type  $\mathcal{P}$ .

C'est là certainement le problème ouvert le plus important et le plus fameux de l'informatique théorique. C'est l'un des sept défis mathématiques que la Fondation Clay a doté en 2000 d'un prix d'un million de dollars. C'est sans doute le plus frappant d'entre eux : d'une part c'est le seul qui peut s'exprimer sans dénaturation majeure dans le langage de tous les jours et du même coup, malgré la conviction spontanée qu'on a que  $\mathcal{P} \neq \mathcal{NP}$ , on ne voit guère comment aborder la question.

Notons que si le complémentaire d'une partie de type  $\mathcal{P}$  est de type  $\mathcal{P}$ , l'analogie pour le type  $\mathcal{NP}$  n'a aucune raison d'être vrai (sauf par exemple s'il s'avérait que  $\mathcal{P} = \mathcal{NP}$ !). Cela amène à introduire les parties de type  $\text{co}\mathcal{NP}$ , complémentaires des parties de type  $\mathcal{NP}$ .

Les parties qui sont à la fois de type  $\mathcal{NP}$  et de type  $\text{co}\mathcal{NP}$  sont donc celles pour lesquelles le « problème de décision » (décider si un élément donné s'il appartient ou non à la partie) peut être résolu par un algorithme non-déterministe à temps polynomial. Par exemple, l'ensemble des nombres premiers est de type  $\text{co}\mathcal{NP}$ , puisque l'ensemble des nombres composés est de type  $\mathcal{NP}$ . Mais on a vu qu'il est aussi de type  $\mathcal{NP}$  : d'après 3.2.2, tout nombre premier possède un certificat *court* de primalité possédant une vérification *rapide* (Pratt).

On a évidemment

$$\mathcal{P} \subset \mathcal{NP} \cap \text{co}\mathcal{NP}.$$

### 5.4.3. Parties de type $\mathcal{RP}$

On a pu penser longtemps que, pour qu'une partie soit « pratiquement calculable », il fallait au moins qu'elle soit de type  $\mathcal{P}$ . Mais l'existence et l'efficacité des algorithmes probabilistes ont conduit à modifier ce point de vue et à introduire le type  $\mathcal{RP}$ .

Les parties de type  $\mathcal{RP}$  sont obtenues par le renforcement suivant de la définition précédente :

**DÉFINITION 5.35.** — *Une partie A de  $\mathbf{N}^s$  est dite de type  $\mathcal{RP}$  s'il existe un entier  $t \geq 0$ , une partie B de  $\mathbf{N}^s \times \mathbf{N}^t$  qui est de type  $\mathcal{P}$ , une fonction à croissance polynomiale  $f$ , et une constante  $\alpha > 0$  avec les deux propriétés suivantes :*

- ▷ *si  $\mathbf{n} \in \mathbf{N}^s$  appartient à A, la proportion des  $\mathbf{a} \in \mathbf{N}^t$  tels que  $(\mathbf{n}, \mathbf{a})$  appartienne à B, parmi ceux tels que  $\|\mathbf{a}\| \leq f(\|\mathbf{n}\|)$ , est au moins égale à  $\alpha$  ;*
- ▷ *si  $\mathbf{n} \in \mathbf{N}^s$  n'appartient pas à A, il n'existe aucun  $\mathbf{a} \in \mathbf{N}^t$  avec  $(\mathbf{n}, \mathbf{a}) \in B$ .*

Si une partie A est de type  $\mathcal{RP}$ , on dispose d'un algorithme probabiliste pour décider de l'appartenance d'une suite  $\mathbf{n}$  à A : on tire au sort des  $\mathbf{a}$  dans le domaine donné par la majoration de l'énoncé. Si on en trouve un avec  $(\mathbf{n}, \mathbf{a}) \in B$ , alors on est sûr que  $\mathbf{n}$  appartient à A. Si, au contraire, on n'en a trouvé aucun au bout de N tirages, on déclare que  $\mathbf{n}$  n'appartient pas à A, avec une « probabilité d'erreur » inférieure à  $(1 - \alpha)^N$ , donc exponentiellement petite en fonction du temps passé.

**EXERCICE 5.23. [C]** — Montrer que le *temps moyen* d'exécution pour l'algorithme déterministe basé sur la recherche exhaustive d'un témoin est polynomial (on suppose pouvoir effectivement tirer des témoins aléatoires indépendants en temps polynomial).

**REMARQUE 5.36.** — Évidemment, la lettre R vient de « random ». À dire vrai, cette histoire d'algorithme probabiliste est moins claire qu'il n'y paraît, car « tirer au sort » des  $\mathbf{a}$  est plus facile à dire qu'à faire, surtout qu'on doit en tirer plusieurs de façon indépendante. On voit bien en particulier qu'il est contradictoire dans les termes de

parler de générateurs déterministes de nombres aléatoires. Mais on peut remarquer qu'il suffit que les tirages apparaissent comme aléatoires à quelqu'un qui n'a qu'une puissance de calcul polynomiale. En fait, il y a un lien direct entre les questions dont nous parlons et le problème de l'engendrement de suites pseudo-aléatoires ; renvoyons sur ce sujet aux textes spécialisés.

Comme illustration directe de ce qui précède, notons que nous avons vu précédemment que la partie C formée des nombres composés est de type  $\mathcal{RP}$  (critères de Miller-Rabin ou de Solovay-Strassen).

Il n'y a aucune raison pour que le complémentaire d'une partie de type  $\mathcal{RP}$  soit de type  $\mathcal{RP}$  (sauf encore une fois s'il s'avérait que  $\mathcal{P} = \mathcal{NP}$ ). Cela amène à introduire les parties de type  $\text{co}\mathcal{RP}$ , complémentaires des parties de type  $\mathcal{RP}$ . On a

$$\mathcal{P} \subset \mathcal{RP} \subset \mathcal{NP}, \quad \mathcal{P} \subset \text{co}\mathcal{RP} \subset \text{co}\mathcal{NP}.$$

On pose  $Z\mathcal{PP} = \mathcal{RP} \cap \text{co}\mathcal{RP}$ , de sorte

$$\mathcal{P} \subset Z\mathcal{PP} \subset \mathcal{NP} \cap \text{co}\mathcal{NP}.$$

Les trois classes exhibées ci-dessus sont donc formées des parties pour lesquelles on dispose d'un algorithme de décision qui fonctionne en temps polynomial et qui est respectivement : déterministe, probabiliste, non-déterministe. Comme on ne sait pas démontrer que  $\mathcal{NP}$  est distinct de  $\mathcal{P}$ , on ne sait pas a fortiori démontrer que  $Z\mathcal{PP}$  est distinct de  $\mathcal{P}$ .

#### 5.4.4. Le cas des nombres premiers

Comme on l'a vu, l'ensemble des nombres premiers est de type  $\text{co}\mathcal{RP}$ . Cela est connu depuis longtemps : l'algorithme de Miller et Rabin date de 1976, celui de Solovay et Strassen de 1977. En 1992, Adleman et Huang ont produit un algorithme, utilisant des mathématiques bien plus difficiles, montrant qu'il est de type  $\mathcal{RP}$ , donc de type  $Z\mathcal{PP}$ .

On sait maintenant qu'il est de type  $\mathcal{P}$ . C'est en 2002 que Agrawal, Kayal et Saxena ont exhibé un algorithme polynomial pour décider de la primalité d'un entier, basé sur une idée à la fois nouvelle et, de manière surprenante, élémentaire.

Soit  $n > 1$  un entier. Supposons d'abord que  $n$  soit premier. Pour tout entier  $a$ , on a dans l'anneau des polynômes à coefficients entiers

$$(X + a)^n \equiv X^n + a \pmod{n}.$$

C'est une forme du petit théorème de Fermat : les coefficients intermédiaires de la formule du binôme sont nuls modulo  $n$ , et  $a^n$  est congru à  $a$  modulo  $n$ . Pour  $X = 0$  on retrouve d'ailleurs Fermat. Mais cette forme ne rencontre

pas les mêmes ennuis (nombres de Carmichael) : la condition  $(X + 1)^n \equiv X^n + 1 \pmod{n}$  implique déjà  $n$  est premier. En effet, si  $p$  est un facteur premier de  $n$  et si on écrit  $n = p^r m$  avec  $m$  non divisible par  $p$ , le coefficient du binôme  $\binom{n}{p}$  est divisible par  $p^{r-1}$ , mais pas par  $p^r$ , donc  $n$  est pas divisible par  $n$ . Le développement de  $(X + 1)^n$  modulo  $n$  contient donc un terme non nul en  $X^p$ , ce qui implique  $p = n$  d'après l'hypothèse.

Naturellement cette condition ne peut servir telle quelle de critère polynomial de primalité, même pour une seule valeur de  $a$  ( $a = 1$  par exemple), car le nombre de termes à calculer (essentiellement  $n/2$ ) est exponentiel en la taille  $\log n$  de  $n$ . L'astuce consiste à réduire le degré du polynôme à manipuler, ce qu'on fait en travaillant modulo un polynôme de la forme  $X^r - 1 \in \mathbf{Z}/n\mathbf{Z}[X]$ , avec  $r$  « assez petit ».

Étendant la notation des congruences, nous écrirons pour deux polynômes  $P$  et  $Q$  à coefficients entiers

$$P(X) \equiv Q(X) \pmod{(n, X^r - 1)}$$

si on peut trouver deux polynômes  $A(X)$  et  $B(X)$  avec  $P(X) - Q(X) = nA(X) + (X^r - 1)B(X)$ . Cela revient à dire que  $P$  et  $Q$  donnent le même reste lorsqu'on les divise d'abord par  $X^r - 1$  dans  $\mathbf{Z}[X]$ , puis qu'on réduit les coefficients modulo  $n$ .

Cela étant, l'algorithme AKS est basé sur le théorème suivant, publié par les trois auteurs en 2004 :

**THÉORÈME 5.37** (Agrawal, Kayal, Saxena). — *Soit  $n$  un entier  $> 1$ . Soit  $r$  un entier  $> 1$  premier à  $n$ , choisi de telle façon que l'ordre de la classe de  $n$  dans le groupe multiplicatif  $(\mathbf{Z}/r\mathbf{Z})^*$  (qui a  $\varphi(r)$  éléments) soit  $> (\log n)^2$ . Posons  $s = \sqrt{\varphi(r)} \log n$ . Supposons que pour tout entier  $a$  avec  $1 \leq a \leq s$ , on ait*

$$(X + a)^n \equiv X^n + a \pmod{(n, X^r - 1)}.$$

*Alors, ou bien  $n$  possède un facteur premier inférieur à  $s$ , ou bien  $n$  est une puissance d'un nombre premier.*

La démonstration, qui utilise des notions (élémentaires) sur les corps finis sera donnée plus loin (9.5). L'algorithme AKS en résulte immédiatement. Étant donné  $n$ , les pas sont les suivants :

1. Éliminer d'abord le cas où  $n$  est une puissance  $m^k$  avec  $k \geq 2$ ,
2. Essayer les entiers  $2, 3, \dots$  successivement jusqu'à trouver un  $r$  qui convienne ; calculer  $s$ ,
3. Éliminer le cas où  $n$  a un facteur premier inférieur à  $s$ ,
4. Tester la congruence pour  $a = 1, \dots, s$ , et conclure.

Il reste à vérifier que l'algorithme est bien polynomial. Reprenons les divers pas, sans entrer dans tous les détails.

Si  $n$  est une puissance  $k$ -ième, on a nécessairement  $k < \log n$  et on essaye tous les  $k$ . Pour un  $k$  donné, on calcule une racine  $k$ -ième approchée (à moins de  $1/2$  près) de  $n$ , on prend l'entier le plus proche  $m$  et on regarde si  $m^k$  vaut  $n$ .

Pour le second pas, on montre qu'on a  $r \leq (\log n)^5$ . Chaque test a un coût en  $(\log r)(\log n)^2$  et on arrive à un coût en  $(\log n)^{7+\varepsilon}$ . On a en outre  $s \leq \sqrt{r} \log n \leq (\log n)^{7/2}$ .

Le troisième pas consiste à diviser  $n$  par les entiers impairs inférieurs à  $s$ , avec un coût total en  $s(\log s)(\log n)$ , ce qui nous amène à  $(\log n)^{9/2+\varepsilon}$ .

Enfin, pour chaque valeur de  $a$ , la vérification de la congruence demande de calculer une puissance  $n$ -ième, dans l'espace des  $r$  coefficients modulo  $n$ , ce qui donne une complexité en  $r(\log n)^2$ . Au total, cela donne une complexité en  $s r (\log n)^2$ , donc  $r^{3/2} (\log n)^3$  et en définitive  $(\log n)^{21/2}$ .

Comme on le voit, c'est le quatrième pas qui domine, donnant une complexité en  $(\log n)^{21/2}$  et c'est la majoration de  $r$  qui est le point crucial.

Des améliorations ont été apportées à cet algorithme. On peut notamment remplacer les polynômes auxiliaires  $X^r - 1$  par d'autres et obtenir des complexités nettement plus basses (en  $(\log n)^{6+\varepsilon}$ ). Néanmoins, l'algorithme AKS et ses dérivés restent actuellement d'intérêt plus théorique (théorique, mais fondamental !) que pratique, car ils n'atteignent pas l'efficacité concrète des algorithmes probabilistes.

# Pour aller plus loin sur la primalité

Comme nous l'avons dit en introduction, nous n'avons abordé que les questions les plus faciles sur le plan des outils mathématiques. On trouvera dans les ouvrages cités (voir ci-dessous) les algorithmes plus avancés.

La primalité est vraiment un sujet éternel. La question est facile à comprendre. Mais depuis au moins vingt-cinq siècles qu'on y travaille, on a l'impression de ne pas finir d'en gratter la surface. Le mélange de régularité et d'irrégularité (toutes deux démontrées, démonstrations qui restent d'ailleurs difficiles) de la répartition des nombres premiers reste un phénomène « naturel » fascinant. La conjecture de Riemann reste ouverte et, semble-t-il, hors d'atteinte.

L'aspect algorithmique n'est pas en reste. Les progrès de la voie élémentaire, variantes du petit théorème de Fermat (Lucas, Lehmer, Miller...) ont été extrêmement lents et pourtant, vu a posteriori, tout cela est bien facile. On a pu croire il n'y a pas si longtemps (certains l'ont même écrit) que les nombres premiers allaient fournir un exemple d'ensemble d'entiers qui ne serait à la fois ni de type  $\mathcal{P}$ , ni  $\mathcal{NP}$ -complet. Puis on a dit que si on prouvait qu'il était de type  $\mathcal{P}$ , alors le système RSA s'effondrerait. Et deux doctorants indiens, Neeraj KAYAL et Nitin SAXENA, poursuivant une voie nouvelle ouverte par leur patron de thèse Manindra AGRAWAL, ont apporté une preuve, élémentaire, du fait qu'il était bien de type  $\mathcal{P}$ . Mais RSA, qui a maintenant 30 ans, tient toujours, sans qu'on puisse encore l'asseoir sur une démonstration. Nul ne sait à quoi s'attendre demain.

## Bibliographie

Nous ne signalons que des livres (les références complètes se trouvent en page 329). En ce qui concerne les articles, on trouvera une bibliographie de plus de 3000 entrées sur le site de Shallit :

<http://www.cs.uwaterloo.ca/~shallit/antbib.html>

On ne peut évidemment pas ne pas commencer par l'immense traité de KNUTH aux multiples éditions, sans cesse renouvelées, dont les deux premiers volumes ([Knuth 1] et [Knuth 2]) couvrent, parmi bien d'autres choses, la plupart des sujets abordés ici. Bien des erreurs de programmation dont certaines ont fait couler pas mal d'encre (le « bug » du Pentium, les erreurs de calcul d'Excel) n'auraient pas eu lieu si leurs auteurs l'avaient consulté.

Nous avons maintenu dans cette bibliographie le livre de NAUDIN et QUITTÉ, actuellement épuisé, mais consultable en bibliothèque.

L’algorithmique de la théorie des nombres est l’objet du livre de COHEN et du monumental traité de BACH et SHALLIT, dont le premier volume est paru.

Comme l’indique son titre, le livre de Koblitz est plus spécifique.

Les nombres premiers, notamment dans l’aspect calculatoire, sont l’objet de nombreux livres. Citons ceux de RIESEL et de CRANDALL et POMERANCE. Il faut aussi mentionner le livre « grand public » de DELAHAYE et le *livre des records* de RIBENBOIM. On trouvera une foule de choses dans les *Prime Pages* de Chris Caldwell : <http://primes.utm.edu>.

Pour la complexité, citons le livre historique de GAREY et JOHNSON et la troisième partie du traité de SIPSER.

Pour la cryptographie<sup>1</sup> s’imposent le traité de SCHNEIER, qui a été traduit en français et le *Handbook of Coding Theory* ([Menezes et al.]), dont la cinquième édition est disponible au téléchargement sur <http://www.cacr.math.uwaterloo.ca/hac/>.

En ce qui concerne Ruby, signalons les deux ouvrages de références : le *Pickaxe* de THOMAS et HUNT et *The Ruby Way* de FULTON. Le livre de COOPER est une bonne introduction. On trouvera aussi nombre de ressources sur le Web et notamment la première édition du Pickaxe ([Thomas, Hunt 1]) : <http://www.rubycentral.com/pickaxe>.

---

1. Dont RSA n'est qu'un tout petit morceau contrairement à l'impression que donnent certains textes de vulgarisation.

## **Deuxième partie**

# **Codes correcteurs**



## Introduction à la deuxième partie

L'une des grandes nouveautés de l'époque actuelle est la *numérisation* de l'information. Cette numérisation ouvre entre autres deux possibilités nouvelles : la conservation illimitée de l'information, la protection contre les erreurs.

La conservation de l'information n'était jusqu'alors envisageable que pour l'écrit *stricto sensu* et au prix de recopies successives ; c'est d'ailleurs de cette façon que de très anciens textes ont été sauvegardés et nous sont parvenus.

En revanche, qu'il s'agisse de sons ou d'images, les procédés d'enregistrement « analogique » qui remontent au XIX<sup>e</sup> siècle, la photographie (NIEPCE, 1823), le phonographe (EDISON, 1877), la cinématographie (MAREY, 1888), ne permettent qu'une reproduction dégradée. On connaît bien en vidéo analogique le phénomène des *générations* successives, chaque recopie se traduisant par une détérioration supplémentaire de la qualité.

La numérisation de l'information ouvre au contraire la possibilité (au moins théorique) d'une conservation illimitée de l'enregistrement, puisque les recopies, qui sont de toute façon indispensables puisqu'aucun support n'a une durée de vie infinie, peuvent se faire sans dégradation aucune du contenu. Notons au passage qu'il est étrange de se dire par exemple que nous ne disposons d'aucun moyen pour reconstituer le jeu au piano de Frédéric CHOPIN ou Franz SCHUBERT, alors que celui de tel ou tel interprète contemporain pourra être conservé indéfiniment.

Si l'on va un peu plus loin, on doit noter que cette capacité de lecture et de recopie sans erreur n'est que théorique, pour de nombreuses raisons, ne seraient-ce que les limites données par la mécanique quantique. Car il n'existe de toutes façons pas de procédé d'enregistrement véritablement « numérique » et c'est toujours de façon analogique que sont en définitive enregistrées les données. Ainsi, même si l'on parle simplement d'une information codée par des « zéros » et des « uns », ces bits sont représentés sur le support par une quantité physique prenant deux valeurs, disons  $z$  et  $u$ , et dans les faits, plutôt des « valeurs proches de  $z$  » et des « valeurs proches de  $u$  ». Et, sans entrer dans des détails techniques, on sent bien que plus les dispositifs sont miniaturisés et plus les vitesses augmentent, plus aussi augmente la probabilité d'une erreur de lecture. En définitive, la précision a un coût et la précision absolue a un coût infini.

On ne peut donc bannir totalement les erreurs. Il s'agit en définitive de les prendre en compte pour en minimiser, voire en compenser totalement

les effets. Et c'est là qu'apparaît la deuxième vertu de la numérisation : la capacité de maîtriser les erreurs de lecture.

## Le principe de la correction d'erreurs

Le principe même de la correction d'erreurs est facile à comprendre à partir de l'expérience courante consistant à corriger des « fautes d'orthographe » dans un texte en langue naturelle. Une telle correction est possible parce qu'une langue naturelle comme le français possède une syntaxe (et au delà, une sémantique) permettant de distinguer mots corrects et mots erronés, et aussi d'induire d'un mot erroné le mot correct le plus vraisemblable. Sans aller chercher les règles de conjugaison, d'accord, etc., le niveau le plus élémentaire de la correction consiste à utiliser le fait que parmi tous les mots que l'on pourrait écrire, seuls certains sont autorisés par la langue.

L'idée d'un code correcteur est justement, partant d'un texte initial qu'on suppose pour simplifier n'être contraint par aucune grammaire, de le traduire en un nouveau texte soumis, lui, à des contraintes syntaxiques permettant une correction des erreurs. Un exemple banal est donné par le *code aéronautique international* qui facilite la transmission vocale d'une suite arbitraire de lettres en la traduisant en une suite de mots choisis pour être deux à deux très distinguables à l'audition grâce au jeu des voyelles (depuis « alpha », « bravo », et « charlie », jusqu'à « yankee » et « zoulou »). Dans le même cadre d'ailleurs, on dit « niner » pour le chiffre 9 au lieu de « nine » pour éviter la confusion nine-five<sup>2</sup>.

Un autre exemple est celui du *numéro d'inscription au répertoire national d'identification des personnes*, ou NIR, couramment (et improprement) appelé « numéro de sécurité sociale ». Il est composé de deux parties, un nombre de 13 chiffres décimaux, et une clé de 2 chiffres. La première partie est composé de divers morceaux : un chiffre pour le sexe, deux pour l'année de naissance, deux pour le mois de naissance, etc. Notons  $n$  ce nombre et  $c$  la clé, de sorte que le NIR à 15 chiffres est  $100n + c$ . On calcule  $c$  comme le complément à 97 du reste de  $n$  modulo 97. Autrement dit,  $c$  est l'entier compris entre 01 et 97 tel que  $n + c$  soit divisible par 97. Ce mécanisme permet de détecter certaines erreurs de copie. Par exemple, si un des chiffres de  $n$  est erroné, alors  $n$  devient  $n' = n + 10^k a$ , avec  $-9 \leq a \leq 9$ , de sorte que son reste modulo 97 est modifié et  $n' + c$  n'est pas divisible par 97. Mais il ne permet pas d'identifier cette erreur et donc de la corriger : puisque  $100 \equiv 3 \pmod{97}$ , ajouter 1 à un chiffre de  $n$  aura le même effet sur la clé qu'ajouter 3 au chiffre qui est deux positions à droite.

---

2. Pierre-Vincent Koseleff m'a signalé l'usage analogue en allemand de « zwau » pour 2, dans le but d'éviter la confusion zwei-drei.

## Le cadre général

De façon très générale, entre un *émetteur* et un *récepteur*, une information emprunte un *canal de transmission*. Ce peut être aussi bien un canal « direct » comme une ligne téléphonique ou un faisceau hertzien, qu'une chaîne complexe d'enregistrement, de stockage et de lecture comme dans le cas des disques informatiques ou des disques optiques (audio, vidéo ou de données).

Or, la transmission par tout canal est susceptible d'introduire des erreurs. On veut donc construire un mécanisme de *codage*, préliminaire à la transmission par le canal, qui rende possibles la détection et la correction de ces erreurs de transmission, sous l'hypothèse évidemment qu'elles ne sont pas trop nombreuses. Il est évident que la contrepartie inévitable d'une telle possibilité de correction est la dilution du message original. Il y donc un certain compromis à faire, et le point d'équilibre dépend des contraintes imposées.

À l'une des extrémités du spectre, lorsque la rapidité est prioritaire, que les erreurs sont rares et que l'opération de transmission peut être répétée (transfert entre composantes d'une même machine, mémoire et unité centrale par exemple), il suffira de détecter la présence d'erreurs. À l'autre extrémité, lorsqu'il faut donner la priorité à la sécurité d'une transmission plus hasardeuse et qu'on ne peut répéter — par exemple pour la réception en temps réel d'images transmises par des satellites d'exploration spatiale lointaine, comme dans l'exemple historique de Mariner 9 (voir 7.2.3) — il faudra une grande capacité de correction.

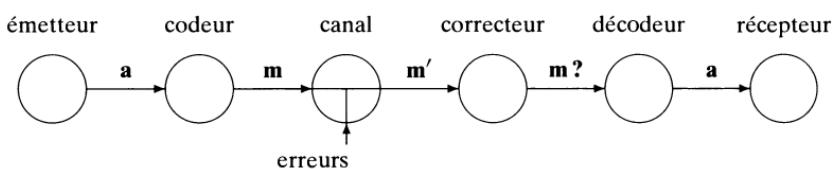
Le but de la théorie est donc de quantifier ces différentes qualités (taux d'information et capacité de correction principalement) d'un *code correcteur d'erreurs* — on dit aussi « code correcteur » tout court — et de montrer comment construire des codes pratiquement « optimaux » dans les divers cas envisagés. Cela suppose qu'on se place dans un certain *modèle* de transmission, impliquant un certain type d'erreurs. Il ne faut pas oublier non plus que le codage et le décodage (c'est surtout ce dernier qui présente la plus grande difficulté) doivent être réalisés par des algorithmes de complexité compatible avec les dispositifs techniques considérés. Cela amène en fait souvent dans la pratique à choisir des codes « loin de l'optimum », mais pour lesquels on dispose d'un algorithme de décodage efficace.

## Le modèle de base

Plaçons nous dans le cadre le plus simple. Il va s'agir de transmettre un *message* de longueur donnée, disons une suite  $a$  de  $k$  bits. Si l'on avait à transmettre un message de longueur quelconque, on le découperait en blocs successifs de  $k$  bits ; c'est pourquoi on parle souvent de *codes de blocs* pour les codes que nous considérons. Le mécanisme de *codage* consistera à associer à

chacun des  $2^k$  mots possibles un *mot de code* bien choisi de  $n$  bits, où  $n$  est un entier  $\geq k$  fixé ; on complétera par exemple le message initial par  $n - k$  bits convenablement choisis. Les  $2^k$  mots de code correspondant aux  $2^k$  messages possibles forment alors par définition le *code* considéré.

Le mot de code  $\mathbf{m}$  déduit du message  $\mathbf{a}$  est ensuite transmis. Le canal de transmission pouvant introduire des erreurs, on obtient à la sortie de ce canal, au lieu de  $\mathbf{m}$ , un mot  $\mathbf{m}'$ . Il s'agit, à partir de  $\mathbf{m}'$ , de reconstituer dans la mesure du possible  $\mathbf{m}$  (*correction*), puis  $\mathbf{a}$  (*décodage*). Le schéma suivant résume l'enchaînement des diverses opérations.



Dans tout cela, l'application de codage (toujours choisie injective pour des raisons évidentes) et son application réciproque de décodage ne jouent que des rôles mineurs. Le problème essentiel est la construction du code lui-même, donc d'un ensemble de mots de  $n$  bits, en nombre  $2^k$ , choisis de façon à rendre possible (et numériquement praticable) l'opération de correction.

### Comment décrire un code

Lorsque  $k$  est petit, on peut décrire un tel code en énumérant ses éléments, mais cela devient impraticable dès que  $k$  atteint quelques dizaines ; il faut donc bien trouver des méthodes plus efficaces. Et puis, comment comprendre la situation à partir d'une telle énumération ? D'ailleurs, on peut toujours « renuméroter les bits », c'est-à-dire appliquer aux  $n$  bits des mots de code une permutation arbitraire (la même pour tous les mots) et obtenir ce qu'on appelle de façon naturelle un code *équivalent* au code initial ; il est donc tout autant souhaitable de trouver des descriptions qui soient moins vulnérables à la renumérotation. On peut par exemple identifier mots de  $n$  bits et parties d'un ensemble fixé  $E$  à  $n$  éléments : fixant une énumération de  $E$ , on associe à la partie  $A$  le mot dont le  $i$ -ième bit vaut 1 si  $A$  contient le  $i$ -ième élément de  $E$  et vaut 0 sinon. Deux énumérations différentes donneront deux codes équivalents.

Il s'agit donc en définitive de distinguer  $2^k$  parties d'un ensemble fini  $E$  à  $n$  éléments, parmi les  $2^n$  possibles. Or, un tel programme, à savoir distinguer des parties privilégiées d'un ensemble fini, ressortit exactement de ce qu'on appelle la *géométrie finie*. Sans entrer dans des détails inutiles à ce niveau de généralité, on peut dire que l'expérience montre que ce n'est qu'en munissant  $E$  de *structures auxiliaires* que l'on peut espérer caractériser simplement un

type de parties, par référence à ces structures. Nous rencontrerons ainsi par exemple les codes de Reed-Muller (7.3.2), lorsque  $E$  est un espace vectoriel fini ; les codes cycliques (10.3) lorsque  $E$  est muni d'un ordre cyclique ; les codes de résidus quadratiques (13.1) lorsque  $E$  est un corps  $\mathbf{Z}/p\mathbf{Z}$ , avec  $p \equiv -1 \pmod{8}$ , soit  $p = 7, 23, \dots$ .

D'ailleurs, c'est souvent à l'inverse en partant d'un code — comme par exemple le code de Golay (13.2), pour lequel  $E = \mathbf{Z}/23\mathbf{Z}$  — que l'on entre le plus simplement dans une situation combinatoire complexe — dans ce cas un « système de Steiner » de type  $(5, 8, 24)$ , voir (7.5.1).

Outre son importance technologique propre, la théorie des codes correcteurs d'erreurs présente ainsi l'avantage de faire parcourir plusieurs thèmes intéressants de mathématique finie et de faire rencontrer quelques-uns des plus jolis objets combinatoires, comme le code de Golay déjà nommé.



# Chapitre 6

## Codes binaires

Dans ce chapitre, nous présentons les premiers éléments de la théorie des codes binaires, ainsi qu'un exemple fondamental, celui des *codes de Hamming*. Le premier exemplaire intéressant de ces codes, le code de Hamming de longueur 7, que nous décrirons en détail, reviendra à plusieurs reprises dans la suite comme élément de plusieurs autres familles de codes.

### § 6.1. Définitions générales

#### 6.1.1. Le corps $\mathbf{F}_2$

Précisons d'abord quelques notations. Nous étudierons plus tard en détail les corps finis (voir le chapitre 9). Pour le moment, nous n'aurons besoin que du corps à deux éléments, traditionnellement noté  $\mathbf{F}_2$ .

On note donc  $\mathbf{F}_2$  l'ensemble à deux éléments formés des bits 0 et 1. Pour l'addition et la multiplication binaires (la seule règle non évidente est  $1 + 1 = 0$ ), c'est un *corps*. Notons que 0 et 1 peuvent aussi s'écrire respectivement faux et vrai et qu'alors l'addition et la multiplication sont respectivement les opérations logiques XOR et AND. Une troisième interprétation consiste à définir  $\mathbf{F}_2$  comme l'anneau  $\mathbf{Z}/2\mathbf{Z}$  des entiers modulo 2, de sorte que 0 représente pair, tandis que 1 représente impair, l'addition et la multiplication étant déduites de celles des entiers. On appellera souvent *parité* d'un entier  $n$  l'élément de  $\mathbf{F}_2$  qui est le reste de la division de  $n$  par 2.

L'ensemble des mots de  $n$  bits, soit  $\mathbf{F}_2^n$ , a une structure naturelle d'espace vectoriel sur le corps  $\mathbf{F}_2$  : l'addition (qui est aussi la soustraction !) se fait bit à bit, la multiplication par les scalaires est évidente (puisque  $0\mathbf{m} = \mathbf{0}$  et  $1\mathbf{m} = \mathbf{m}$  pour tout  $\mathbf{m} \in \mathbf{F}_2^n$ ). Nous notons **0** le *mot nul* qui a toutes ses composantes égales à 0 et de même **1** le *mot plein*, qui a toutes ses composantes égales à 1.

Nous utiliserons librement dans ce cadre tous les résultats usuels de l'algèbre linéaire.

#### 6.1.2. Le modèle

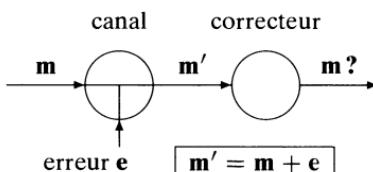
Reprendons le cadre fixé dans l'introduction. Il s'agit donc de transmettre un *message* formé de  $k$  bits. Le *codage* consiste à associer à chacun des  $2^k$  messages possibles *un mot de code* à  $n$  bits.

Il serait stupide de choisir un codage non injectif, aussi supposerons-nous toujours que l'application de codage est injective. Cela nous permettra de l'oublier dans la théorie et d'identifier simplement messages et mots de codes. Parmi les  $2^n$  mots de  $\mathbf{F}_2^n$ , les  $2^k$  mots de codes forment le *code C* considéré ; c'est, comme on dit, un code binaire de *longueur n* à  $2^k$  éléments.

**DÉFINITION 6.1.** — *On appelle code binaire de longueur n une partie C de  $\mathbf{F}_2^n$ . Si C a  $2^k$  éléments, le rapport  $k/n$  est appelé le taux d'information ou simplement le taux<sup>1</sup> de C.*

**REMARQUE 6.2.** — Au niveau de généralité où nous sommes, il n'est pas nécessaire d'imposer que le nombre N des éléments de C soit une puissance de 2. On appelle alors taux d'information le rapport  $(\log N)/n$ . Mais, comme nous nous intéresserons essentiellement aux codes linéaires (voir ci-dessous) dont le nombre d'éléments est une puissance de 2, ce cas plus général ne nous servira pas.

Avec la simplification consistant à éliminer les phases de codage et de décodage, le schéma général de l'introduction se présente comme suit :



Le mot de code  $\mathbf{m} \in C$  est transmis à travers un *canal de transmission* susceptible d'introduire des erreurs. À la sortie de ce canal, on obtient au lieu de  $\mathbf{m}$  un mot  $\mathbf{m}'$ , et il s'agit de reconstituer dans la mesure du possible  $\mathbf{m}$ .

Dans l'espace vectoriel  $\mathbf{F}_2^n$ , on a  $\mathbf{m}' = \mathbf{m} + \mathbf{e}$ , où  $\mathbf{e}$  est l'*erreur de transmission* : un bit de  $\mathbf{e}$  vaut 1 lorsque les bits correspondants de  $\mathbf{m}$  et  $\mathbf{m}'$  diffèrent. Nous appellerons *poids* d'un mot  $\mathbf{m} \in \mathbf{F}_2^n$  et noterons  $w(\mathbf{m})$  le nombre des bits de  $\mathbf{m}$  qui sont égaux à 1. Ainsi, dire que  $\mathbf{m}$  et  $\mathbf{m}'$  diffèrent en  $i$  places, c'est dire que  $w(\mathbf{e}) = i$  ; au lieu de dire que l'erreur est de poids  $i$ , on dit aussi que l'on a  $i$  erreurs.

**EXERCICE 6.1. [A]** — Notons  $\mathbf{x} \cdot \mathbf{y}$  le produit bit à bit des mots  $\mathbf{x}$  et  $\mathbf{y}$ . On a  $w(\mathbf{x} + \mathbf{y}) = w(\mathbf{x}) + w(\mathbf{y}) - 2w(\mathbf{x} \cdot \mathbf{y}) \leq w(\mathbf{x}) + w(\mathbf{y})$ .

1. En anglais, « *rate* », d'où la notation fréquente  $R = k/n$  et l'utilisation en français du mot « rendement ».

### 6.1.3. Le modèle d'erreur : le canal binaire symétrique

Plusieurs modèles de comportement du canal peuvent être envisagés. Celui que nous choisissons est le suivant : on suppose que chaque bit de  $\mathbf{m}$  a la même probabilité  $p < 1/2$  d'être transformé en le bit opposé et que les événements concernant les différents bits de  $\mathbf{m}$  sont indépendants. Il s'agit, comme on dit, d'un *canal binaire symétrique sans mémoire* : « sans mémoire » car les bits ont des comportements indépendants (nous verrons plus tard le cas différent des « bouffées d'erreurs ») ; « symétrique », car les modifications  $0 \mapsto 1$  et  $1 \mapsto 0$  ont la même probabilité.

**REMARQUE 6.3.** — On notera que si l'on avait  $p = 1/2$ , la sortie du canal serait purement aléatoire, indépendante de l'entrée, tandis que si l'on avait  $p > 1/2$ , il suffirait de changer chaque bit de  $\mathbf{m}'$  en son opposé pour se ramener au cas  $p < 1/2$ .

Avec les hypothèses et notations précédentes, une erreur  $\mathbf{e}$  donnée, dont le poids est  $i$ , a une probabilité d'apparition égale à

$$p^i(1-p)^{n-i} = \left(\frac{p}{1-p}\right)^i(1-p)^n.$$

Puisqu'on a supposé  $p < 1/2$ , on a  $p/(1-p) < 1$  et la probabilité précédente décroît lorsque  $i$  croît : *une erreur  $\mathbf{e}$  est d'autant plus probable que son poids est plus faible.*

Comme il y a  $\binom{n}{i}$  mots de poids  $i$  et qu'une erreur de poids  $i$  a une probabilité égale à  $p^i(1-p)^{n-i}$ , on a  $P(w(\mathbf{e}) \leq t) = \sum_{i=0}^t \binom{n}{i} p^i(1-p)^{n-i}$ , et de manière équivalente

$$P(w(\mathbf{e}) > t) = 1 - \sum_{i=0}^t \binom{n}{i} p^i(1-p)^{n-i} = \sum_{i=t+1}^n \binom{n}{i} p^i(1-p)^{n-i}.$$

**REMARQUE 6.4.** — Le poids de l'erreur suit donc une loi binomiale de moyenne  $np$  et de variance  $np(1-p)$ .

### 6.1.4. Le décodage

Lorsqu'on a reçu le message transmis  $\mathbf{m}'$ , les différentes valeurs possibles pour l'erreur  $\mathbf{e}$  sont les  $\mathbf{m}' - \mathbf{m}$ , où  $\mathbf{m}$  parcourt le code C. Puisqu'une erreur est d'autant plus probable que son poids est plus petit, le mot de code initial  $\mathbf{m}$  est d'autant plus probable que  $w(\mathbf{m}' - \mathbf{m})$  est plus petit. La stratégie naturelle est donc de rechercher le *maximum de vraisemblance* et de choisir  $\mathbf{m}$  de façon à minimiser  $w(\mathbf{m}' - \mathbf{m})$ . C'est cela que doit accomplir l'algorithme de *décodage*.

## Distance minimale

La *distance* (dite de Hamming) de deux mots est par définition le poids de leur différence<sup>2</sup>, c'est-à-dire le nombre de positions en lesquels leurs bits respectifs diffèrent :

$$d(\mathbf{m}, \mathbf{m}') = w(\mathbf{m} - \mathbf{m}').$$

C'est bien une distance, puisqu'on a  $w(\mathbf{x} - \mathbf{z}) \leq w(\mathbf{x} - \mathbf{y}) + w(\mathbf{y} - \mathbf{z})$ .

**EXERCICE 6.2. [B]** — Donner une démonstration très courte de cette inégalité.

On appelle *distance minimale* d'un code C la plus petite valeur  $d$  de la distance de deux mots distincts du code :

$$d = \inf_{\substack{\mathbf{m} \in C, \mathbf{m}' \in C \\ \mathbf{m} \neq \mathbf{m}'}} w(\mathbf{m} - \mathbf{m}').$$

## Détection et correction

On peut *détecter* l'existence d'une erreur  $\mathbf{e}$  telle que  $w(\mathbf{e}) < d$ . En effet, si le mot de code  $\mathbf{m} \in C$  est perturbé par l'erreur  $\mathbf{e} \neq 0$  avec  $w(\mathbf{e}) < d$ , alors le mot reçu  $\mathbf{m}' = \mathbf{m} + \mathbf{e}$  n'appartient pas à C; car sinon, on aurait  $\mathbf{e}' = \mathbf{m}' - \mathbf{m} \in C$ , ce qui contredirait la définition de  $d$ .

On peut aussi *corriger* une erreur  $\mathbf{e}$  telle que  $w(\mathbf{e}) < d/2$ . En effet, si  $\mathbf{m}' \in \mathbf{F}_2^n$  peut s'écrire  $\mathbf{m}_1 + \mathbf{e}_1$  avec  $\mathbf{m}_1 \in C$  et  $w(\mathbf{e}_1) < d/2$ , et aussi  $\mathbf{m}_2 + \mathbf{e}_2$  avec  $\mathbf{m}_2 \in C$  et  $w(\mathbf{e}_2) < d/2$ , on a

$$w(\mathbf{m}_1 - \mathbf{m}_2) = w(\mathbf{e}_2 - \mathbf{e}_1) \leq w(\mathbf{e}_2) + w(\mathbf{e}_1) < d,$$

donc  $\mathbf{m}_1 = \mathbf{m}_2$  par définition de la distance minimale  $d$ . Il existe donc un unique mot de code  $\mathbf{m}$  tel que  $w(\mathbf{m}' - \mathbf{m}) < d/2$ ; le *décodage* consiste ainsi à déduire du mot erroné  $\mathbf{m}'$  ce mot de code  $\mathbf{m}$ .

## Effacements

Il existe une situation un peu différente, dans laquelle on suppose que le canal de transmission cause des *effacements*, c'est-à-dire que certains des bits transmis sont perdus. La différence avec les erreurs est que la position des effacements est connue. Un code dont la distance minimale est  $> e$  permet de corriger  $e$  effacements. En effet, deux mots de codes qui diffèrent chacun du mot reçu en au plus  $e$  positions connues, diffèrent l'un de l'autre au plus en ces mêmes positions, donc sont à une distance  $\leq e < d$ .

**EXERCICE 6.3. [B]** — Un code de distance minimale  $d$  permet de corriger simultanément  $t$  erreurs et  $e$  effacements, si  $2t + e < d$ .

---

2. Ici, comme plus haut, on parle de différence et on utilise le signe « moins » bien que somme et différence coïncident. Mais d'une part c'est plus parlant, et d'autre part les définitions ainsi données s'appliquent encore à des corps différents de  $\mathbf{F}_2$ .

### 6.1.5. Codes linéaires

Le cas le plus simple et le plus important est celui où le code  $C$  est un sous-espace vectoriel de  $\mathbf{F}_2^n$ , ce qui signifie simplement que la somme (c'est aussi la différence) de deux mots de code quelconques est encore un mot de code. On dit alors que  $C$  est un code binaire *linéaire*.

Puisque la différence de deux mots de code est encore un mot de code, la *distance minimale d'un code linéaire n'est autre que le poids minimal d'un mot de code non nul* :

$$d = \inf_{\substack{\mathbf{m} \in C \\ \mathbf{m} \neq \mathbf{0}}} w(\mathbf{m}).$$

Par ailleurs, il est équivalent de dire que  $C$  possède  $2^k$  éléments, ou que sa dimension comme espace vectoriel sur le corps  $\mathbf{F}_2$  est  $k$ .

**DÉFINITION 6.5.** — *On dit que la longueur  $n$ , la dimension  $k$  et la distance minimale  $d$  sont les paramètres du code linéaire  $C$ , ou que  $C$  est un code binaire linéaire de paramètres  $(n, k, d)$ . La codimension de  $C$  est la différence  $n - k$ .*

## § 6.2. Exemples

### 6.2.1. Les codes triviaux : les cas $k = 0, 1, n - 1, n$

Donnons quelques exemples rudimentaires. Il y a d'abord évidemment le *code plein* formé de tous les mots. C'est un code linéaire de paramètres  $(n, n, 1)$ . Sa distance minimale est 1, inutile de dire qu'il ne peut rien corriger !

Une façon rudimentaire de permettre la correction des erreurs est basée sur la « répétition pure » : on répète  $n$  fois chaque bit transmis. Il a donc  $k = 1$  et s'agit donc du code de longueur  $n$  formé des deux mots  $\mathbf{0} = [0 \cdots 0]$  et  $\mathbf{1} = [1 \cdots 1]$ . C'est un code linéaire de paramètres  $(n, 1, n)$ . Il permet de corriger des erreurs de transmission en nombre  $< n/2$  par un simple algorithme majoritaire : on compte le nombre de 0 et le nombre de 1 et le plus nombreux l'emporte. Naturellement, le prix à payer est une dilution extrême de l'information, puisque le taux d'information est  $1/n$ . Il faut par exemple prendre  $n = 3$  pour pouvoir corriger une erreur.

Plus généralement, pour tout mot non nul  $\mathbf{m} \in \mathbf{F}_2^n$ , le code  $\{\mathbf{0}, \mathbf{m}\}$  est un code linéaire de paramètres  $(n, 1, w(\mathbf{m}))$ .

Un exemple universellement connu en informatique est celui des *codes de parité*. Le plus fréquent consiste à compléter chaque octet binaire par un bit supplémentaire de façon que la somme des 9 bits ainsi obtenus soit nulle (modulo 2, donc paire). Le code obtenu est le code de longueur 9

formé des  $2^n$  mots de poids pair. Sa distance minimale est 2. Il permet de détecter l'existence d'au plus une erreur de transmission, qui change la parité du résultat, mais pas de la localiser. Il permet tout au plus de corriger un effacement.

De même, pour tout  $n$ , le code de parité de longueur  $n$  est le code binaire linéaire de paramètres  $(n, n - 1, 2)$  formé de tous les mots de poids pair dans  $\mathbf{F}_2^n$ . C'est le sous-espace vectoriel de codimension 1, noyau de la forme linéaire  $\varepsilon : (m_0, \dots, m_{n-1}) \mapsto m_0 + \dots + m_{n-1} \in \mathbf{F}_2$ .

Plus généralement, les codes linéaires de codimension 1 sont les noyaux des formes linéaires non nulles sur  $\mathbf{F}_2^n$ , une telle forme étant simplement la somme de certaines des composantes du mot.

Les trois types de codes linéaires précédents sont souvent appelés *codes linéaires triviaux*. On peut ajouter à cette liste le *code nul* réduit au seul mot  $\mathbf{0}$  de  $\mathbf{F}_2^n$ . Il est linéaire. Sa distance minimale n'est pas clairement définie ; on le dira de paramètres  $(n, 0, ?)$ . Les codes linéaires triviaux de longueur  $n$  sont donc en définitive les codes de dimension 0, 1,  $n - 1$  et  $n$ .

### 6.2.2. Codes $t$ -correcteurs, capacité de correction, codes parfaits

*La boule de Hamming*, ou simplement boule, de centre  $\mathbf{m}$  et de rayon  $t$  est par définition l'ensemble des mots  $\mathbf{m}'$  avec  $w(\mathbf{m}' - \mathbf{m}) \leq t$ .

**PROPOSITION 6.6.** — Soient  $C$  un code binaire de longueur  $n$ , de distance minimale  $d$ , et  $t$  un entier  $\geq 0$ . Les conditions suivantes sont équivalentes :

- (i) pour tout mot  $\mathbf{m}'$  dans  $\mathbf{F}_2^n$ , il existe au plus un mot  $\mathbf{m}$  de  $C$  tel que  $w(\mathbf{m}' - \mathbf{m}) \leq t$ ;
- (ii) les boules de rayon  $t$  centrées en les différents mots de  $C$  sont disjointes ;
- (iii) on a  $2t + 1 \leq d$ , c'est-à-dire  $t \leq (d - 1)/2$ .

*Démonstration.* Les conditions (i) et (ii) sont évidemment équivalentes, et on a vu ci-dessus qu'elles sont impliquées par (iii). Inversement, supposons qu'on ait  $2t \geq d$ . Il existe alors deux mots de  $C$ , soient  $\mathbf{m}_1$  et  $\mathbf{m}_2$ , qui diffèrent en au plus  $2t$  positions. On peut donc construire un mot  $\mathbf{m}'$  avec  $w(\mathbf{m}' - \mathbf{m}_1) \leq t$  et  $w(\mathbf{m}' - \mathbf{m}_2) \leq t$ , ce qui contredit (ii).  $\square$

Un code  $C$  satisfaisant aux conditions précédentes est dit  *$t$ -correcteur* puisqu'il permet, vu (i), de retrouver le mot de code initial  $\mathbf{m}$  connaissant le mot  $\mathbf{m}'$  transmis, pourvu que ce dernier ne comporte qu'au plus  $t$  bits erronés. Évidemment, un code  $t$ -correcteur est  $t'$ -correcteur pour  $t' \leq t$ . D'après la proposition, le plus grand  $t$  possible est  $\lfloor (d - 1)/2 \rfloor$ .

Ainsi, un code est d'autant meilleur du point de vue des possibilités de correction qu'il offre que sa distance minimale  $d$  est plus grande. Le rapport  $d/n$  mesure ainsi la *capacité de correction* du code.

Le décodage consiste en définitive à trouver le point de  $C$  le plus proche du message reçu  $\mathbf{m}'$ ; si le poids de l'erreur est  $\leq t$ , le point  $\mathbf{m}'$  appartient à l'une des boules considérées ci-dessus et le décodage consiste à trouver laquelle. Évidemment, le code sera d'autant plus économique qu'il y aura moins de place perdue. La limite est le cas des *codes parfaits*, hélas fort rares comme nous le verrons (remarque 6.9) :

**DÉFINITION 6.7.** — *On dit qu'un code binaire de longueur  $n$  est  $t$ -correcteur parfait si les boules de rayon  $t$  centrées en les divers mots du code forment une partition de l'espace  $\mathbf{F}_2^n$ .*

**EXERCICE 6.4. [A]** — Le code plein est 0-correcteur parfait.

**EXERCICE 6.5. [A]** — Si  $n$  est impair, le code de répétition pure est parfait.

**EXERCICE 6.6. [B]** — Si  $C$  est  $t$ -correcteur parfait, sa distance minimale est  $2t + 1$ .

**PROPOSITION 6.8.** — *Soit  $C$  un code binaire 1-correcteur de longueur  $n$  et soit  $N$  le nombre des éléments de  $C$ . On a  $N(n + 1) \leq 2^n$ . De plus, les conditions suivantes sont équivalentes :*

- (i)  *$C$  est 1-correcteur parfait ;*
- (ii) *on a  $N(n + 1) = 2^n$  ;*
- (iii) *il existe un entier  $r > 1$ , avec  $n = 2^r - 1$  et  $N = 2^{n-r}$ .*

*Démonstration.* Une boule de rayon 1 a  $n+1$  éléments (un au centre et  $n$  à distance 1). La réunion des  $N$  boules disjointes de rayon 1 centrées aux mots de code possède donc  $N(n + 1)$  éléments. Ce nombre est inférieur à  $2^n$ , l'égalité caractérisant les codes parfaits. De plus, si l'on a  $N(n + 1) = 2^n$ , alors  $N$  est un diviseur de  $2^n$ , donc de la forme  $2^{n-r}$ , et on obtient  $n + 1 = 2^r$ . Enfin, on a nécessairement  $n \geq 3$ , donc  $r \geq 2$ .  $\square$

Dans la proposition précédente, on n'a pas supposé  $C$  linéaire, et d'ailleurs il existe des codes 1-correcteurs parfaits non linéaires. Mais la proposition implique que le nombre des éléments d'un tel code est une puissance de 2. Dans le cas linéaire, la condition (iii) signifie que le code a des paramètres de la forme  $(2^r - 1, 2^r - 1 - r, 3)$ , donc  $(3, 1, 3), (7, 4, 3), (15, 11, 3)\dots$

**EXERCICE 6.7. [B]** — À quels codes correspond le cas  $r = 2$  ?

**EXERCICE 6.8. [B]** — On suppose que  $C$  est  $t$ -correcteur parfait. Quelle relation numérique doit-il y avoir entre  $n$ ,  $N$  et  $t$  ?

**REMARQUE 6.9.** — Il n'y a en fait qu'un seul code binaire  $t$ -correcteur parfait avec  $t > 1$  qui ne soit pas trivial : c'est le code de Golay (voir ci-dessous le théorème 13.17).

**REMARQUE 6.10.** — Il convient sans doute un peu plus précis. D'après la définition précédente, un code est  $t$ -correcteur s'il donne la possibilité de corriger une

erreur de poids  $\leq t$ . Dans la pratique, ce qu'on a envie d'appeler *code t-correcteur*, c'est plutôt le couple formé d'un tel code et d'un algorithme explicite calculant l'*application de correction*  $\mathbf{m}' \mapsto \mathbf{m}$ . Il y a d'ailleurs là plusieurs variantes suivant le comportement attendu de l'algorithme sur les mots  $\mathbf{m}'$  « trop erronés » (à distance  $> t$  de tout mot de code) ; on peut par exemple exiger ou non que le mot « corrigé » soit un mot de code, ou encore demander ou non que l'algorithme signale explicitement ces mots. Nous reviendrons là-dessus plus loin (voir 8.4). Signalons aussi qu'il existe (voir par exemple la remarque 11.8) des codes théoriquement  $t$ -correcteurs, mais pour lesquels l'algorithme naturel de correction ne corrige que des erreurs de poids  $\leq t'$ , avec  $t' < t$ . En outre, la complexité<sup>3</sup> de l'algorithme de correction est évidemment un des paramètres fondamentaux du choix d'un code.

### 6.2.3. Le code de Hamming $H_3$ de longueur 7

Donnons sans plus attendre un exemple moins rudimentaire que celui des codes triviaux, à savoir un code linéaire de paramètres  $(7, 4, 3)$ , donc 1-correcteur, d'ailleurs parfait d'après la proposition 6.8.

Dans l'espace vectoriel  $\mathbf{F}_2^7$ , on considère le vecteur  $\mathbf{m}_0 = [1101000]$  et les trois vecteurs qu'on en déduit par décalage, soient  $\mathbf{m}_1 = [0110100]$ ,  $\mathbf{m}_2 = [0011010]$  et  $\mathbf{m}_3 = [0001101]$ . L'application linéaire de codage  $\gamma : \mathbf{F}_2^4 \rightarrow \mathbf{F}_2^7$ , associe au mot  $\mathbf{a} = [a_0 a_1 a_2 a_3]$  le mot  $a_0 \mathbf{m}_0 + a_1 \mathbf{m}_1 + a_2 \mathbf{m}_2 + a_3 \mathbf{m}_3$ . Cela veut simplement dire que  $\gamma(\mathbf{a})$  s'obtient en ajoutant les  $\mathbf{m}_i$  pour les  $i$  qui correspondent aux bits égaux à 1 dans  $\mathbf{a}$ . On obtient explicitement :

$$\begin{aligned}\gamma[0000] &= [0000000], & \gamma[1000] &= [1101000], & \gamma[0100] &= [0110100], \\ \gamma[0010] &= [0011010], & \gamma[0001] &= [0001101], & \gamma[1100] &= [1011100], \\ \gamma[1010] &= [1110010], & \gamma[1001] &= [1100101], & \gamma[0110] &= [0101110], \\ \gamma[0101] &= [0111001], & \gamma[0011] &= [0010111], & \gamma[1110] &= [1000110], \\ \gamma[1101] &= [1010001], & \gamma[1011] &= [1111111], & \gamma[0111] &= [0100011]. \\ \gamma[1111] &= [1001011].\end{aligned}$$

L'application linéaire  $\gamma$  est injective. Son image  $C$ , formée des 16 mots de la liste précédente, est le code considéré, que l'on note  $H_3$  et que l'on appelle le *code de Hamming de longueur 7*. Il contient un mot de poids 0, sept mots de poids 3, sept mots de poids 4, un mot de poids 7. Pour un usage ultérieur, remarquons qu'on constate, par vérification directe, que  $C$  est stable par décalage circulaire (on dit que c'est un code « cyclique »).

**EXERCICE 6.9. [B]** — Montrer qu'on peut considérer les mots du code comme formés de 4 bits de message et de 3 bits de contrôle.

---

3. Signalons que, convenablement formulé, le problème général de la correction est  $\mathcal{NP}$ -complet.

### § 6.3. Outils de construction de codes

Plusieurs procédés permettent de construire des codes à partir d'un ou de plusieurs autres. Nous nous restreignons dans la suite au cas des codes linéaires, bien que la plupart de ces constructions soient valables dans le cas général.

#### 6.3.1. Constructions élémentaires

##### Somme directe

Soient  $C \subset \mathbf{F}_2^n$  et  $C' \subset \mathbf{F}_2^{n'}$  deux codes linéaires de paramètres respectifs  $(n, k, d)$  et  $(n', k', d')$ . La *somme directe* de  $C$  et  $C'$  est l'ensemble  $C \oplus C' \subset \mathbf{F}_2^{n+n'}$  formé des mots  $(\mathbf{m}, \mathbf{m}')$  obtenus par *concaténation* d'un mot quelconque  $\mathbf{m}$  de  $C$  et d'un mot quelconque  $\mathbf{m}'$  de  $C'$ . C'est un code linéaire. On a évidemment  $w((\mathbf{m}, \mathbf{m}')) = w(\mathbf{m}) + w(\mathbf{m}')$ , et  $(\mathbf{m}, \mathbf{m}')$  est non nul dès que  $\mathbf{m}$  ou  $\mathbf{m}'$  est non nul, de sorte que la distance minimale de  $C \oplus C'$  est la borne inférieure des distances minimales de  $C$  et  $C'$ . Ainsi  $C \oplus C'$  est de paramètres  $(n + n', k + k', \inf(d, d'))$ .

##### Construction diagonale

Soit  $C \subset \mathbf{F}_2^n$  un code linéaire de type  $(n, k, d)$ . L'ensemble  $C^{[2]} \subset \mathbf{F}_2^{2n}$  des mots  $(\mathbf{m}, \mathbf{m})$  où  $\mathbf{m}$  parcourt  $C$ , est un code linéaire de longueur  $2n$  et de dimension  $k$ . On a évidemment  $w(\mathbf{m}, \mathbf{m}) = 2w(\mathbf{m})$ , de sorte que la distance minimale de  $C^{[2]}$  vaut  $2d$ . Ainsi  $C^{[2]}$  est de paramètres  $(2n, k, 2d)$ ; on notera que  $C \oplus C$  est de paramètres  $(2n, 2k, d)$ .

##### Construction « $\inf(2d, d')$ »

Cette construction combine les deux précédentes. Soient  $C \subset \mathbf{F}_2^n$  et  $C' \subset \mathbf{F}_2^{n'}$  deux codes linéaires de même longueur. Considérons le code linéaire  $D \subset \mathbf{F}_2^{2n}$  formé des mots de la forme  $(\mathbf{a}, \mathbf{a} + \mathbf{a}') = (\mathbf{a}, \mathbf{a}) + (0, \mathbf{a}')$ , où  $\mathbf{a}$  parcourt  $C$  et  $\mathbf{a}'$  parcourt  $C'$ .

**LEMME 6.11.** — Soient  $d$  la distance minimale de  $C$  et  $d'$  celle de  $C'$ . La distance minimale de  $D$  est  $\inf(2d, d')$ .

*Démonstration.* Notons d'abord qu'il existe effectivement des mots  $(\mathbf{a}, \mathbf{a})$  de poids  $2d$  et des mots  $(0, \mathbf{a}')$  de poids  $d'$ . Il s'agit donc de minorer  $w(\mathbf{a}, \mathbf{a} + \mathbf{a}')$  pour  $(\mathbf{a}, \mathbf{a} + \mathbf{a}') \neq (\mathbf{0}, \mathbf{0})$ . Distinguons deux cas. Si  $\mathbf{a}' = \mathbf{0}$ , alors  $\mathbf{a} \neq \mathbf{0}$  et on a  $w(\mathbf{a}, \mathbf{a}) = 2w(\mathbf{a}) \geq 2d$ . Si  $\mathbf{a}' \neq \mathbf{0}$ , alors  $w(\mathbf{a}') \geq d'$ ; on a par ailleurs

$$w(\mathbf{a}, \mathbf{a} + \mathbf{a}') = w(\mathbf{a}) + w(\mathbf{a} + \mathbf{a}') \geq w(\mathbf{a}) + w(\mathbf{a}) + w(\mathbf{a}') - 2w(\mathbf{a} \cdot \mathbf{a}'),$$

où  $\mathbf{a} \mathbf{a}'$  est le produit bit à bit de  $\mathbf{a}$  et  $\mathbf{a}'$ . Puisque  $w(\mathbf{a}) \geq w(\mathbf{a} \mathbf{a}')$ , on en tire  $w(\mathbf{a}, \mathbf{a} + \mathbf{a}') \geq w(\mathbf{a}') \geq d'$ , ce qui conclut dans le second cas.  $\square$

Si  $C$  est de paramètres  $(n, k, d)$  et  $C'$  de paramètres  $(n, k', d')$ , le code  $D$  est donc de paramètres  $(2n, k + k', \inf(2d, d'))$ .

## Renumérotation

C'est l'opération la plus simple. Elle consiste à *renuméroter* les indices, donc à appliquer aux  $n$  bits de chacun des mots de code une permutation fixée (la même pour tous les mots). On obtient ainsi un nouveau code qui garde les caractères de l'ancien (linéarité, paramètres). On dit que deux codes sont *équivalents* s'ils se déduisent l'un de l'autre par une renumérotation. On peut naturellement appliquer des renumérotations avant et/ou après chacune des opérations répertoriées ci-dessus.

### 6.3.2. Extension paire

Considérons un code binaire linéaire  $C \subset \mathbf{F}_2^n$ . L'application  $\varepsilon$  qui associe à un mot  $\mathbf{m} = [m_0, \dots, m_{n-1}]$ , la somme dans  $\mathbf{F}_2$  des  $m_i$  est une forme linéaire sur  $\mathbf{F}_2^n$ . On ne confondra pas  $w(\mathbf{m})$  qui est, si l'on veut, la somme dans  $\mathbf{N}$  des  $m_i$ , considérés comme des entiers valant 0 ou 1, avec  $\varepsilon(\mathbf{m})$  qui est la parité de  $w(\mathbf{m})$ . On dira que  $\mathbf{m}$  est *pair* s'il est de poids pair, qu'il est *impair* sinon. La somme de deux mots pairs ou de deux mots impairs est paire, la somme d'un mot pair et d'un mot impair est impaire.

Ainsi, puisque  $C$  est supposé linéaire, de deux choses l'une : ou bien tous les mots de  $C$  sont pairs (on dit alors que le code est *pair*), ou bien  $C$  contient autant de mots impairs que de mots pairs (on dit souvent que le code est *impair*). Dans ce deuxième cas, on peut déduire de  $C$  deux codes pairs. D'abord les mots pairs de  $C$  forment le sous-code pair  $C^{\text{pair}}$  de codimension 1 dans  $C$ .

Mais on peut aussi ajouter au code  $C$  un *bit de parité*. On considère donc l'application linéaire de  $\mathbf{F}_2^n$  dans  $\mathbf{F}_2^{n+1} = \mathbf{F}_2^n \times \mathbf{F}_2$  qui à  $\mathbf{m}$  associe  $(\mathbf{m}, \varepsilon(\mathbf{m}))$ . L'image de  $C$  par cette application est un code pair  $\bar{C}$  de longueur  $n + 1$  et de même dimension que  $C$ , code que l'on appelle *l'extension paire de C*. Soit  $d$  la distance minimale de  $C$ . Il est clair que la distance minimale de  $\bar{C}$  est  $d$  si  $d$  est paire,  $d + 1$  sinon.

Notons aussi qu'on peut faire cette construction même lorsque  $C$  est déjà pair, mais que le résultat n'est guère intéressant, puisque le bit de parité est constamment égal à 0 et qu'ainsi la dernière composante de  $\bar{C}$  « ne sert à rien ».

Traitons l'exemple du *code de Hamming étendu de longueur 8*, noté  $\bar{H}_3$ , que l'on obtient en ajoutant un bit de parité au code de Hamming  $H_3$  explicité plus haut. L'application de codage correspondante est

$$\begin{aligned}
 \bar{\gamma}[0000] &= [00000000], & \bar{\gamma}[1000] &= [11010001], & \bar{\gamma}[0100] &= [01101001], \\
 \bar{\gamma}[0010] &= [00110101], & \bar{\gamma}[0001] &= [00011011], & \bar{\gamma}[1100] &= [10111000], \\
 \bar{\gamma}[1010] &= [11100100], & \bar{\gamma}[1001] &= [11001010], & \bar{\gamma}[0110] &= [01011100], \\
 \bar{\gamma}[0101] &= [01110010], & \bar{\gamma}[0011] &= [00101110], & \bar{\gamma}[1110] &= [10001101], \\
 \bar{\gamma}[1101] &= [10100011], & \bar{\gamma}[1011] &= [11111111], & \bar{\gamma}[0111] &= [01000111], \\
 \bar{\gamma}[1111] &= [10010110].
 \end{aligned}$$

Le code obtenu est un code linéaire de dimension 4, de longueur 8. Il possède le mot **0** de poids 0 et le mot **1** de poids 8, les quatorze autres étant de poids 4. Sa distance minimale est 4. Ses paramètres sont donc (8, 4, 4). Nous verrons plus loin (en 7.1.2) une interprétation combinatoire de ce code.

### 6.3.3. Orthogonal d'un code

Définissons sur  $\mathbf{F}_2^n$  un produit scalaire par la formule usuelle

$$\langle \mathbf{m} | \mathbf{n} \rangle = \sum_{i=1}^n m_i n_i \in \mathbf{F}_2.$$

De manière équivalente, on a

$$\langle \mathbf{m} | \mathbf{n} \rangle = \varepsilon(\mathbf{m} \cdot \mathbf{n}).$$

On a en particulier  $\langle \mathbf{1} | \mathbf{m} \rangle = \varepsilon(\mathbf{m})$ .

Soit  $C \subset \mathbf{F}_2^n$  un code binaire linéaire de longueur  $n$ . Notons  $C^\perp$  l'orthogonal de  $C$ , c'est-à-dire l'ensemble des  $\mathbf{n} \in \mathbf{F}_2^n$  tels que  $\langle \mathbf{m} | \mathbf{n} \rangle = 0$  pour tout  $\mathbf{m} \in C$ . C'est évidemment un code linéaire. C'est le *code orthogonal* de  $C$  (on dit parfois *code dual* au lieu de code orthogonal).

**EXERCICE 6.10. [A]** — Quel est l'orthogonal du code de répétition pure ? du code de parité ?

Les mots de  $C^\perp$  peuvent aussi s'interpréter comme les formes linéaires sur  $\mathbf{F}_2^n$  qui s'annulent identiquement sur  $C$ . L'un des résultats clés de l'algèbre linéaire, le fait que la codimension d'un sous-espace peut se calculer comme le nombre minimum d'équations linéaires qu'il faut pour le définir, se traduit par la partie *a*) de la proposition suivante :

**PROPOSITION 6.12.** — *a)* On a  $\dim(C) + \dim(C^\perp) = n$ .  
*b)* On a  $(C^\perp)^\perp = C$ .

**EXERCICE 6.11. [B]** — Vérifier *b*).

On dit que deux codes  $C$  et  $C'$  de même longueur sont *orthogonaux* si l'on a  $\langle \mathbf{m} | \mathbf{m}' \rangle = 0$  pour tout  $\mathbf{m} \in C$  et tout  $\mathbf{m}' \in C'$ , ce qui peut aussi s'écrire  $C \subset C'^\perp$ , ou encore symétriquement  $C' \subset C^\perp$ .

On notera une différence fondamentale avec le cas usuel de l'algèbre linéaire réelle : le carré scalaire d'un vecteur non nul peut très bien être nul. En fait, le carré scalaire d'un vecteur n'est autre que sa parité : il est nul si le vecteur est pair, et vaut 1 sinon. On peut même rencontrer plus surprenant : on dit que  $C$  est *auto-orthogonal* s'il est orthogonal à lui-même, c'est-à-dire si on a  $\langle \mathbf{m} | \mathbf{n} \rangle = 0$  pour tous  $\mathbf{m}$  et  $\mathbf{n}$  dans  $C$ . Il suffit évidemment de vérifier cette condition lorsque  $\mathbf{m}$  et  $\mathbf{n}$  parcouruent une base de  $C$ .

Nous renconterons souvent la situation suivante :

**PROPOSITION 6.13.** — *Considérons un code linéaire impair  $C \subset \mathbf{F}_2^n$  et son extension paire  $\overline{C} \subset \mathbf{F}_2^{n+1}$ . Alors l'orthogonal de  $\overline{C}$  est formé des éléments  $(\mathbf{a}, \alpha)$ , où  $\mathbf{a}$  est orthogonal au sous-code pair de  $C$  et  $\alpha$  vaut 0 si  $\mathbf{a}$  est orthogonal à  $C$  et vaut 1 sinon.*

*Démonstration.* Les mots de  $\overline{C}$  sont par définition les  $(\mathbf{m}, \varepsilon(\mathbf{m}))$  pour  $\mathbf{m}$  parcourant  $C$ . Soit  $(\mathbf{a}, \alpha)$  un mot quelconque de  $\mathbf{F}_2^n \times \mathbf{F}_2$ . On a

$$\langle (\mathbf{m}, \varepsilon(\mathbf{m})) | (\mathbf{a}, \alpha) \rangle = \langle \mathbf{m} | \mathbf{a} \rangle + \varepsilon(\mathbf{m})\alpha.$$

Si  $\mathbf{m}$  est pair, on obtient  $\langle \mathbf{m} | \mathbf{a} \rangle$ . Si l'on veut que  $(\mathbf{a}, \alpha)$  soit orthogonal à  $\overline{C}$ , il faut donc déjà que  $\mathbf{a}$  soit orthogonal au sous-code pair de  $C$ . Mais, puisque la différence de deux éléments impairs de  $C$  est paire, il en résulte alors que  $\langle \mathbf{m} | \mathbf{a} \rangle$  prend la même valeur pour tous les éléments impairs  $\mathbf{m}$  de  $C$ . Cette valeur vaut 0 si  $\mathbf{a}$  est orthogonal à  $C$ , et vaut 1 sinon. La proposition en résulte.  $\square$

**COROLLAIRE 6.14.** — *Le code orthogonal de  $C$  est formé des mots  $m$  tels que  $(m, 0)$  soit orthogonal à  $\overline{C}$ .*

EXERCICE 6.12. [A] — Vérifier.

**COROLLAIRE 6.15.** — *Les conditions suivantes sont équivalentes :*

- (i)  $\overline{C}$  est son propre code orthogonal ;
- (ii) l'orthogonal de  $C$  est le sous-code pair de  $C$ .

EXERCICE 6.13. [A] — Vérifier.

EXERCICE 6.14. [B] — Montrer que la situation de ce corollaire se rencontre dans le cas du code  $H_3$ .

## § 6.4. Matrices génératrices et vérificatrices d'un code linéaire

### Avertissement

Puisqu'on écrit les mots comme des suites « horizontales » de bits, la tradition en théorie des codes, *contraire à l'usage classique en algèbre linéaire*, est de représenter les mots de code par des matrices-lignes plutôt que par des matrices-colonnes. Mais on représente quand même également par des lignes les formes linéaires, ce qui entraîne la présence de transposées dans beaucoup de formules.

### 6.4.1. Matrices génératrices et vérificatrices

Par définition, un code binaire linéaire  $C$  de longueur  $n$  et de dimension  $k$  est un sous-espace vectoriel de dimension  $k$  de  $\mathbf{F}_2^n$ . On peut donc le décrire explicitement des diverses façons suivantes :

- ▷ comme sous-espace engendré par  $k$  mots linéairement indépendants ;
- ▷ comme image d'une application linéaire injective  $\mathbf{F}_2^k \rightarrow \mathbf{F}_2^n$  ;
- ▷ comme noyau commun de  $n - k$  formes linéaires indépendantes ;
- ▷ comme noyau d'une application linéaire surjective  $\mathbf{F}_2^n \rightarrow \mathbf{F}_2^{n-k}$ .

Les deux premières descriptions ci-dessus reviennent à fixer une *matrice génératrice*  $G$  à  $k$  lignes et  $n$  colonnes, de rang  $k$ , dont les lignes forment une base de  $C$  (c'est-à-dire transposée de la matrice usuelle de l'application linéaire considérée). Les deux dernières reviennent à choisir une *matrice vérificatrice*  $P$  à  $n - k$  lignes et  $n$  colonnes, de rang  $n - k$ , dont les lignes donnent des équations linéaires indépendantes définissant  $C$  ou, ce qui revient au même, forment une base du code orthogonal  $C^\perp$ . On a par construction  $\tilde{P}G = \mathbf{0}$ , où  $\tilde{P}$  est la transposée de  $P$ .

**EXERCICE 6.15. [A]** — Soient  $G$  et  $P$  des matrices respectivement génératrice et vérificatrice de  $C$ . Alors  $P$  est une matrice génératrice du code  $C^\perp$ , et  $G$  une matrice génératrice du code  $C^\perp$ .

**EXERCICE 6.16. [B]** — Pour que l'on ait  $C \subset C^\perp$ , il faut et il suffit que les lignes de  $G$  soient deux à deux orthogonales. Pour que l'on ait  $C^\perp \subset C$ , il faut et il suffit que les lignes de  $P$  soient deux à deux orthogonales.

On a par exemple donné en 6.2.3 pour le code de Hamming de longueur 7 la matrice génératrice

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

### 6.4.2. Changements de base

Si les mots  $\mathbf{m}_1, \dots, \mathbf{m}_k$  forment une base de  $C$  ou, ce qui revient au même, sont les  $k$  lignes d'une matrice génératrice de  $C$ , alors les éléments de  $C$  sont les combinaisons linéaires des  $\mathbf{m}_i$ , c'est-à-dire simplement les  $2^k$  mots obtenus comme somme d'une des  $2^k$  parties de l'ensemble des  $\mathbf{m}_i$ .

Si on remplace la base du code par une autre, alors la nouvelle matrice génératrice  $G'$  s'écrit

$$G' = AG,$$

où  $A$  une matrice carrée inversible d'ordre  $k$ . On pourra prendre pour nouvelle matrice vérificatrice le produit  $P' = \tilde{A}^{-1}P$ .

En particulier, puisque la matrice  $G$  est de rang  $k$  par hypothèse, on peut en extraire des matrices carrées inversibles d'ordre  $k$ . Quitte à permute les colonnes (ce qui remplace le code par un code équivalent), on peut supposer que les  $k$  premières colonnes de  $G$  forment une matrice inversible. Prenant alors pour  $A$  l'inverse de cette matrice, on obtient une nouvelle matrice génératrice de la forme

$$G = [ I_k \quad B ],$$

où  $B$  est une matrice à  $n - k$  colonnes et  $k$  lignes.

On pourra donc toujours se ramener au cas d'une matrice génératrice de cette forme. Pour  $i = 1, \dots, k$ , notons  $e_i$  le  $i$ -ième vecteur de base de  $\mathbf{F}_2^k$  et  $b_i \in \mathbf{F}_2^{n-k}$  la  $i$ -ième ligne de la matrice  $B$ . Le code est donc engendré par les vecteurs  $(e_i, b_i)$ . C'est l'ensemble des vecteurs  $(\mathbf{m}, b(\mathbf{m}))$ , où  $b : \mathbf{F}_2^k \rightarrow \mathbf{F}_2^{n-k}$  est l'application linéaire qui envoie  $e_i$  sur  $b_i$  pour tout  $i$ .

**EXERCICE 6.17. [B]** — Appliquer cela au code de Hamming de longueur 7.

### 6.4.3. Conditions de parité généralisées

Gardons les notations précédentes. Les mots du code sont les  $(\mathbf{m}, b(\mathbf{m}))$  avec  $\mathbf{m} \in \mathbf{F}_2^k$ . Ils sont donc obtenus par concaténation d'un message  $\mathbf{m}$  de  $k$  bits et d'un *contrôle de parité généralisé*  $b(\mathbf{m})$ . De manière équivalente, les mots du code sont les  $(\mathbf{m}, \mathbf{m}')$  qui satisfont à la *condition de parité généralisée*  $\mathbf{m}' = b(\mathbf{m})$ . Le code est donc le noyau de l'application linéaire  $(\mathbf{m}, \mathbf{m}') \mapsto -b(\mathbf{m}) + \mathbf{m}'$ , de matrice<sup>4</sup>

$$P = [ -\tilde{B} \quad I_{n-k} ].$$

À cause de cet exemple, on dit souvent *matrice de parité* au lieu de matrice vérificatrice.

Donnons un exemple. Choisissons une matrice  $B$  symétrique (cela nous évitera de distinguer matrice initiale et transposée) d'ordre 3

$$B = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Elle permet de construire un code de dimension 3 et de longueur 6, donné au choix par la matrice génératrice  $G$ , ou la matrice vérificatrice  $P$ , suivantes :

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix},$$

---

4. Dans le cas binaire où nous sommes, le signe « moins » est psychologique ; mais les formules resteront valables lorsque nous passerons au cas général où  $\mathbf{F}_2$  sera remplacé par un corps fini quelconque.

$$P = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Ce code est formé des mots  $\mathbf{m} = (m_0, m_1, m_2, m_3, m_4, m_5)$  satisfaisant aux trois conditions de parité  $m_3 = m_0 + m_1$ ,  $m_4 = m_0 + m_2$  et  $m_5 = m_1 + m_2$ .

#### 6.4.4. Extension paire

L'extension paire d'un code se décrit simplement en termes de matrices génératrices : il suffit en effet de compléter une matrice génératrice du code initial par une colonne de façon que chaque ligne devienne paire. Pour le code de Hamming par exemple, les quatre lignes de la matrice génératrice  $G$  donnée plus haut étant impaires, la nouvelle matrice génératrice s'obtient en ajoutant à  $G$  une dernière colonne entièrement formée de 1 :

$$\bar{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Mais la description de l'extension paire est également très simple en termes de matrices vérificatrices. Supposons en effet que le code  $C$  soit défini par une matrice vérificatrice  $P$ . Il est donc formé des mots  $\mathbf{m}$  de longueur  $n$  tels que le vecteur colonne transposé, disons  $M$ , satisfasse à  $P \cdot M = 0$ . L'extension paire est formé de vecteurs  $\bar{M}$  de longueur  $n + 1$  dont la première partie  $M$  satisfait à  $P \cdot M = 0$  et qui sont pairs, c'est-à-dire tels que  $\bar{1} \cdot \bar{M} = 0$ , où on désigne par  $\bar{1}$  le mot formé de  $n + 1$  bits égaux à 1. On obtient ainsi la matrice vérificatrice pour le code étendu :

$$\bar{P} = \begin{bmatrix} P & \mathbf{0} \\ \bar{1} & 1 \end{bmatrix}.$$

#### 6.4.5. Matrice vérificatrice et syndrome

La donnée d'une matrice vérificatrice  $P$  ou, ce qui revient au même, d'une application linéaire surjective  $\pi : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^{n-k}$  de noyau  $C$ , permet de construire un algorithme de décodage. Pour tout mot  $\mathbf{x} \in \mathbb{F}_2^n$ , on appelle *syndrome* de  $\mathbf{x}$  le mot  $\pi(\mathbf{x}) \in \mathbb{F}_2^{n-k}$ . Si, au mot de code initial  $\mathbf{m}$  s'ajoute l'erreur  $\mathbf{e}$ , donnant le mot transmis erroné  $\mathbf{m}' = \mathbf{m} + \mathbf{e}$ , on a  $\pi(\mathbf{m}') = 0$  donc  $\pi(\mathbf{m}') = \pi(\mathbf{e})$  et le syndrome du mot transmis est égal au syndrome de l'erreur.

Dire que la distance minimale de  $C$  est  $\geq d$ , c'est dire qu'aucun mot  $\mathbf{x}$  non nul de poids  $< d$  n'a un syndrome nul. De manière équivalente, dire que

C est  $t$ -correcteur, c'est dire que les différents mots  $\mathbf{e}$  de poids  $\leq t$  donnent des syndromes  $\pi(\mathbf{e})$  tous différents, ce qui permet de retrouver l'erreur par inspection du syndrome.

Dans l'exemple précédent par exemple, les différents mots de poids 1 donnent comme syndrome les six colonnes de  $P$ , toutes non nulles et différentes, de sorte que C est 1-correcteur. De manière équivalente, aucune colonne, ni aucune somme de deux colonnes distinctes de C n'est nulle, ce qui signifie que la distance minimale de C est  $> 2$ . Mais ce code contient des mots de poids 3, par exemple les colonnes de  $G$ ; sa distance minimale est donc 3 et on a construit un code linéaire de paramètres  $(6, 3, 3)$ .

Dans le cas général, le syndrome d'un mot  $\mathbf{x}$  est la somme des colonnes de  $P$  qui correspondent aux bits non nuls de  $\mathbf{x}$ . On en déduit :

**PROPOSITION 6.16.** — *Pour que le code linéaire C, défini par la matrice vérificatrice P, soit de distance minimale  $\geq d$ , il faut et il suffit qu'aucune somme de colonnes de P, en nombre m avec  $0 < m < d$ , ne soit nulle.*

**EXERCICE 6.18. [B]** — En considérant le rang de la matrice P, prouver l'inégalité de Singleton  $d \leq n - k + 1$ .

## § 6.5. Les codes binaires de Hamming

### 6.5.1. Construction

Proposons-nous de construire par le procédé précédent des codes linéaires 1-correcteurs, de longueur  $n$ , de codimension  $r$ . Il s'agit donc d'exhiber une matrice vérificatrice  $P$  à  $r$  lignes et  $n$  colonnes, dont les colonnes sont non nulles et toutes différentes. Or, il y a en tout  $2^r - 1$  vecteurs non nuls dans  $\mathbf{F}_2^r$ , ce qui donne  $n \leq 2^r - 1$ . Pour atteindre cette borne, il faut constituer la matrice P avec *toutes* les colonnes non nulles possibles sans répétition. Seul l'ordre de ces colonnes reste à fixer, *choisissons-en un*.

On obtient alors une matrice à  $r$  lignes et  $2^r - 1$  colonnes, donc un code binaire linéaire de paramètres  $(2^r - 1, 2^r - r - 1, 3)$ , que l'on appelle un *code de Hamming*. Par construction, les codes de Hamming de même longueur sont équivalents (6.3.1). Ces codes ont été découverts par Richard HAMMING et Marcel GOLAY en 1949-1950.

On note souvent  $H_r$ ,  $r \geq 2$ , un code de Hamming de longueur  $2^r - 1$ . On a ainsi des codes de Hamming de paramètres  $(3, 1, 3)$ ,  $(7, 4, 3)$ ,  $(15, 11, 3)$ ,  $(31, 26, 3)$ ... Si nécessaire, on notera  $H_1$  le code nul de longueur 1, de paramètres  $(1, 0, ?)$ .

**EXERCICE 6.19. [A]** — Qu'est le code  $H_2$  ?

**EXERCICE 6.20. [B]** — Pourquoi ces codes sont-ils de distance minimale exactement 3 ?

Résumons les résultats obtenus :

**PROPOSITION 6.17.** — *a) Les codes de Hamming sont 1-correcteurs et parfaits.*

*b) Tout code binaire linéaire 1-correcteur parfait est un code de Hamming.*

*c) Tout code binaire 1-correcteur parfait a les mêmes paramètres qu'un code de Hamming.*

*Démonstration.* La partie *c*) et le fait que les codes de Hamming sont parfaits résultent de la proposition 6.8. Les autres assertions ont déjà été démontrées.  $\square$

**REMARQUE 6.18.** — Il existe effectivement des codes binaires 1-correcteurs parfaits non linéaires.

Dire que le code de Hamming  $C \subset \mathbb{F}_2^n$  est 1-correcteur parfait, c'est par définition dire que tout mot de  $\mathbb{F}_2^n$  est, ou bien un mot de  $C$ , ou bien à distance 1 d'un mot de  $C$ , autrement dit « un mot de  $C$  avec un bit erroné ». L'algorithme de décodage des codes de Hamming est ainsi particulièrement simple : on reçoit le mot  $\mathbf{m}'$ , on calcule le syndrome  $\pi(\mathbf{m}')$  ; si ce syndrome est nul, alors  $\mathbf{m}' \in C$  et on prend  $\mathbf{m} = \mathbf{m}'$  ; si ce syndrome est non nul, c'est l'une des colonnes de la matrice  $P$ , disons la  $i$ -ième ; on change alors le  $i$ -ième bit de  $\mathbf{m}'$ , ce qui donne le mot de code  $\mathbf{m}$  dont  $\mathbf{m}'$  est à distance 1.

Pour construire explicitement un code de Hamming, la codimension  $r$  étant donnée, le seul choix possible est celui de l'ordre des colonnes de la matrice  $P$ . Comme on l'a vu, il peut être intéressant de placer en fin une sous-matrice unité. Pour  $r = 2$ , on a  $n = 3, k = 1$ , on peut ainsi prendre

$$P = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix},$$

et on obtient le code de répétition pure de longueur 3 (on doit avoir  $m_1 = m_0$  et  $m_2 = m_0$ ).

Le premier cas vraiment intéressant est  $r = 3$ , donc  $n = 7$  et  $k = 4$ . On peut ainsi prendre la matrice

$$P = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Pour des raisons qui apparaîtront ultérieurement, il est meilleur de choisir un ordre particulier des colonnes (et on verra qu'on peut le faire pour tout  $r$ ), de façon que les lignes de la matrice se déduisent les unes des autres par décalage à droite, comme par exemple :

$$P = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix},$$

Les trois contraintes de parité sur les mots  $\mathbf{m} = (m_0, \dots, m_6)$  de  $C$  sont ici  $m_0 + m_2 + m_3 + m_4 = 0, m_1 + m_3 + m_4 + m_5 = 0, m_2 + m_4 + m_5 + m_6 = 0$ . On aboutit ainsi à la description explicite de  $H_3$  donnée en 6.2.3. En effet :

**EXERCICE 6.21.** [A] — Vérifier que les vecteurs  $\mathbf{m}_i, i = 0, \dots, 3$ , introduits en 6.2.3 satisfont à ces conditions de parité.

**EXERCICE 6.22.** [B] — Montrer que l'orthogonal du code  $H_3$  est son sous-code pair.

### 6.5.2. Les codes de Hamming étendus

Les codes de Hamming, étant de distance minimale impaire ( $d = 3$ ), sont tous impairs. On peut donc les enrichir d'un bit de parité comme expliqué en 6.3.2 et obtenir les *codes de Hamming étendus*, de paramètres  $(2^r, 2^r - r - 1, 4)$ , pour  $r \geq 2$ , soit  $(4, 1, 4), (8, 4, 4), (16, 11, 4), (32, 26, 4) \dots$

On note souvent  $\bar{H}_r$  un code de Hamming étendu de longueur  $2^r$ . Si nécessaire, on note  $\bar{H}_1$  le code nul de longueur 2 et  $\bar{H}_0$  le code nul de longueur 1.

**EXERCICE 6.23.** [A] — Qu'est le code  $\bar{H}_2$  ?

Nous allons, comme ci-dessus pour le cas  $r = 3$ , décrire en général les codes de Hamming par des matrices vérificatrices. Supposons donc avoir défini le code de Hamming  $H_r$ , de longueur  $2^r - 1$ , comme le noyau de la matrice vérificatrice  $P_r$ , à  $r$  lignes et  $2^r - 1$  colonnes obtenue, rappelons-le, en énumérant (arbitrairement) toutes les colonnes non nulles de hauteur  $r$ . On peut alors prendre comme matrice  $P_{r+1}$

$$P_{r+1} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} \\ P_r & \mathbf{0} & P_r \end{bmatrix},$$

car on a bien écrit ainsi toutes les colonnes possibles (les chiffres gras **1** et **0** désignent des colonnes ou des lignes obtenues en répétant le bit correspondant).

Le code étendu  $\bar{P}_r$  est décrit par la matrice

$$\bar{P}_r = \begin{bmatrix} P_r & \mathbf{0} \\ \mathbf{1} & 1 \end{bmatrix}.$$

La matrice  $\bar{P}_{r+1}$  vaut alors

$$\bar{P}_{r+1} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} & 0 \\ P_r & \mathbf{0} & P_r & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & 1 \end{bmatrix},$$

c'est-à-dire

$$\bar{P}_{r+1} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \bar{P}_r & \bar{P}_r \end{bmatrix}.$$

On obtient ainsi une construction récursive des matrices

$$\begin{aligned}\bar{P}_0 &= [ \ 1 \ ], \\ \bar{P}_1 &= \left[ \begin{array}{cc} 1 & 0 \\ 1 & 1 \end{array} \right], \\ \bar{P}_2 &= \left[ \begin{array}{cccc} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{array} \right], \\ \bar{P}_3 &= \left[ \begin{array}{ccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \right] \dots\end{aligned}$$

Supprimant la dernière ligne et la dernière colonne, on obtient

$$\begin{aligned}P_1 &= [ \ 1 \ ], \\ P_2 &= \left[ \begin{array}{ccc} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{array} \right], \\ P_3 &= \left[ \begin{array}{ccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \right] \dots\end{aligned}$$

**PROPOSITION 6.19.** — Soit  $\bar{H}_r \subset \mathbf{F}_2^{2^r}$  un code de Hamming étendu de longueur  $2^r$ . Considérons dans  $\mathbf{F}_2^{2^{r+1}} = \mathbf{F}_2^{2^r} \times \mathbf{F}_2^{2^r}$ , le sous-espace formé des couples  $(\mathbf{a}, \mathbf{a} + \mathbf{a}')$ , avec  $\mathbf{a}$  pair et  $\mathbf{a}' \in \bar{H}_r$ . C'est un code de Hamming étendu de longueur  $2^{r+1}$ .

*Démonstration.* Revenons à la description donnée ci-dessus pour la matrice  $\bar{P}_{r+1}$ . Son noyau  $H_{r+1}$  est formé des couples  $(\mathbf{a}, \mathbf{b}) \in \mathbf{F}_2^{2^r} \times \mathbf{F}_2^{2^r}$  qui satisfont aux deux conditions  $\varepsilon(\mathbf{a}) = 0$  et  $\bar{P}_r(\mathbf{a} + \mathbf{b}) = \mathbf{0}$ . Posant  $\mathbf{a} + \mathbf{b} = \mathbf{a}'$ , on obtient la description annoncée.  $\square$

On reconnaît la construction «  $\inf(2d, d')$  » donnée dans 6.3.1.

**EXERCICE 6.24. [B]** — Décrire récursivement le code  $H_r$  de manière analogue.



# Chapitre 7

## Codes, combinatoire, géométrie

Il y a une connexion très étroite entre codes binaires et structures combinatoires finies. Cette correspondance peut être utilisée dans deux directions inverses. Ou bien, on part d'une structure combinatoire connue, comme par exemple une géométrie finie, et on en déduit un code ; c'est le cas des codes de Reed-Muller (7.3.2) ou des codes simplex (7.2.2). Ou bien, on part d'un code connu et on en déduit une structure combinatoire ; c'est le cas par exemple de la construction de certains systèmes de Steiner (7.5.1), comme celui qui provient du code de Golay (voir 7.5.3 ou le théorème 13.14).

### § 7.1. Les codes binaires comme codes de parties

#### 7.1.1. Traductions

Considérons un code binaire  $C \subset \mathbf{F}_2^n$ . Chaque mot  $\mathbf{m}$  du code est bien défini par l'ensemble  $\{i \mid m_i = 1\}$  des numéros de ses bits égaux à un, qui est une partie de l'ensemble  $I = \{0, \dots, n-1\}$  que l'on pourrait appeler son *support*. En associant à chaque mot du code son support, on transforme  $C$  en un ensemble de parties de  $I$ .

La construction inverse consiste à interpréter  $\mathbf{F}_2^n$  comme l'espace de toutes les applications de  $I$  dans  $\mathbf{F}_2$  et à associer à chaque partie  $A$  de  $I$  sa *fonction caractéristique*  $\chi_A : I \rightarrow \mathbf{F}_2$  qui vaut 1 sur  $A$  et 0 sur le complémentaire de  $A$ . Cela donne le dictionnaire suivant :

mot de longueur $n$	partie de $I$
mot nul $\mathbf{0}$	partie vide
mot plein $\mathbf{1} = [1 \cdots 1]$	partie pleine
poids d'un mot	cardinal d'une partie
somme de deux mots $a + b$	différence symétrique <sup>1</sup> $A \oplus B$
$w(b) = w(a) + w(b - a)$	$B$ contient $A$
produit bit à bit de deux mots	intersection de deux parties
code binaire $C$ de longueur $n$	ensemble $C$ de parties de $I$
$C$ est pair	# $A$ est pair pour tout $A \in C$
$C$ est linéaire	$C$ est stable par diff. sym.

1. Rappelons que la différence symétrique de deux ensembles  $A$  et  $B$  est  $A \oplus B = A \cup B - A \cap B$ , complémentaire de leur intersection dans leur réunion.

D'une manière un peu plus générale, on peut considérer un ensemble fini  $I$  quelconque et l'espace vectoriel  $\mathbf{F}_2^I$  de toutes les applications de  $I$  dans  $\mathbf{F}_2$ . À chaque partie  $A$  de  $I$ , on associe comme ci-dessus sa fonction caractéristique  $\chi_A \in \mathbf{F}_2^I$ . Tout ensemble  $C$  de parties de  $I$  se traduit ainsi en un sous-ensemble de  $\mathbf{F}_2^I$ .

On appellera *énumération* de l'ensemble fini  $I$  à  $n$  éléments une bijection de  $\{0, \dots, n-1\}$  sur  $I$ . Pour chaque énumération de  $I$ , disons  $i_0, \dots, i_{n-1}$ , on obtient une bijection entre parties de  $I$  et mots de  $\mathbf{F}_2^n$ . Cette bijection transforme  $C$  en une partie de  $\mathbf{F}_2^n$ , donc en un code binaire. Les différents codes obtenus ainsi, pour toutes les énumérations de  $I$ , sont équivalents (6.3.1) puisqu'ils se déduisent les uns des autres par des renumérotations. En fait, on a souvent intérêt, comme on le verra, à rester dans ce cadre un peu plus général d'un ensemble fini  $I$ , sans nécessairement en choisir une énumération privilégiée. On parlera alors de *codes de parties de  $I$* .

Les correspondances du tableau précédent restent valables dans ce cadre général.

## Extension paire

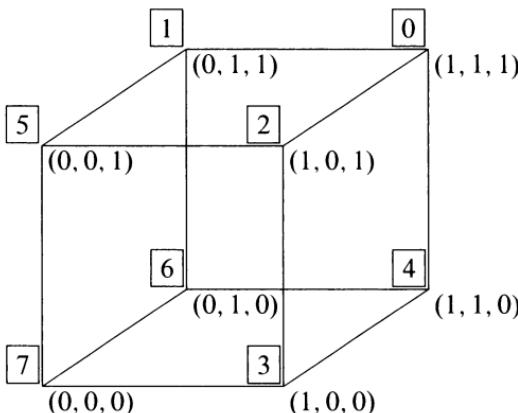
Notons au passage la chose suivante. Considérons un code  $C$  de parties d'un ensemble fini  $I$ . Une *extension paire*  $\Gamma = \bar{C}$  s'obtient en ajoutant à  $I$  un élément  $p$  qui « portera le bit de parité », de sorte que  $\Gamma$  sera un code de parties de  $J = I \cup \{p\}$ . Pour chaque élément  $A \subset I$  de  $C$ , l'élément correspondant de  $\Gamma$  sera  $A$  ou  $A \cup \{p\}$  selon que  $\#A$  est pair ou impair.

Mais inversement, prenons un code  $\Gamma$  de parties d'un ensemble fini  $J$  et un élément  $p$  de  $J$ , et soit  $I$  le complémentaire de  $p$  dans  $J$ . Pour chaque mot  $B \in \Gamma$ ,  $B \cap I$  est une partie de  $I$ . L'ensemble de ces parties forment un code  $C$  de parties de  $I$ . Supposons  $\Gamma$  pair. Alors on aura  $B = (B \cap I) \cup \{p\}$  si  $(B \cap I)$  est impair et  $B = B \cap I$  sinon. Donc  $\Gamma$  est une extension paire de  $C$ . En d'autres termes, et contrairement à l'apparence, le bit de parité d'une extension paire n'a rien de spécial : *tout bit d'un code pair est le bit de parité du code obtenu en le supprimant*.

**EXERCICE 7.1. [B]** — Ce qu'on vient de dire ne signifie pas que les divers codes obtenus en supprimant les divers bits soient équivalents : donner un contre-exemple.

### 7.1.2. Un exemple

Reprenons l'exemple du code de Hamming étendu de longueur 8 vu en 6.3.2. Numérotions les bits de 0 à 7 et répartissons ces numéros aux sommets du cube unité comme le montre le dessin ci-dessous :



On identifie ainsi l'ensemble  $\{0, \dots, 7\}$  à l'ensemble des huit sommets du cube unité usuel (dans  $\mathbb{R}^3$ ). À chaque mot de code, associons alors son support, partie formée des numéros des bits valant 1 dans ce mot. On obtient seize parties, à savoir la partie vide et la partie pleine, ainsi que quatorze parties à 4 éléments, que l'on retrouve dans la première et la troisième colonnes du tableau suivant :

1567	$X_0 = 0$	0234	$X_0 = 1$
2357	$X_1 = 0$	0146	$X_1 = 1$
3467	$X_2 = 0$	0125	$X_2 = 1$
0457	$X_0 = X_1$	1236	$X_0 + X_1 = 1$
0267	$X_0 = X_2$	1345	$X_0 + X_2 = 1$
0137	$X_1 = X_2$	2456	$X_1 + X_2 = 1$
1247		0356	

Les douze premières sont contenues dans les douze plans dont l'équation est donnée. Les deux autres sont les deux tétraèdres inscrits dans le cube. Mais, si l'on considère maintenant les coordonnées 0 ou 1 des sommets du cube non plus comme des nombres réels, mais comme des bits, de sorte que ces huit points forment en fait l'espace vectoriel  $\mathbb{F}_2^3$ , on constate alors ces deux autres parties ont elles aussi des équations linéaires à coefficients dans  $\mathbb{F}_2$ . On obtient maintenant le tableau suivant :

1567	$X_0 = 0$	0234	$X_0 = 1$
2357	$X_1 = 0$	0146	$X_1 = 1$
3467	$X_2 = 0$	0125	$X_2 = 1$
0457	$X_0 + X_1 = 0$	1236	$X_0 + X_1 = 1$
0267	$X_0 + X_2 = 0$	1345	$X_0 + X_2 = 1$
0137	$X_1 + X_2 = 0$	2456	$X_1 + X_2 = 1$
1247	$X_0 + X_1 + X_2 = 0$	0356	$X_0 + X_1 + X_2 = 1$

En définitive, on voit apparaître les quatorze équations linéaires possibles et les parties que l'on a distinguées sont tout simplement les *plans affines* de  $\mathbf{F}_2^3$  (voir ci-dessous 7.2.1).

Se restreindre au code de Hamming proprement dit revient à ignorer le sommet numéro 7 (l'origine). Les sept mots de poids 3 du code de Hamming correspondent aux sept plans passant par l'origine. Les sept mots de poids 4 sont les complémentaires des précédents.

Cet exemple montre qu'en fait, il n'est pas forcément plus simple de raisonner sur l'ensemble  $\{0, \dots, n-1\}$  plutôt que sur le cas plus général en apparence d'un ensemble fini  $I$  à  $n$  éléments. Dans le cas qu'on vient de voir en effet, en remplaçant l'ensemble « naïf »  $\{0, \dots, 7\}$  par le cube  $I = \mathbf{F}_2^3$ , on obtient une description en définitive plus simple et plus compréhensible.

**EXERCICE 7.2. [C]** — Nous avons déjà remarqué que la permutation circulaire des indices 0 à 6 transforme mots de code en mots de code. Comment le vérifie-t-on dans ce modèle ?

## § 7.2. Codes d'hyperplans affines

Nous allons maintenant généraliser l'exemple précédent en dimension supérieure, le mot dimension étant pris au sens « géométrique » (donc ici  $m = 3$ ), et non au sens de la dimension du code (ici  $k = 4$ ), ou de sa longueur (ici  $n = 2^m = 2^3 = 8$ ). Nous le ferons en deux temps, d'abord en considérant des hyperplans, puis des sous-espaces plus généraux.

### Notations

Nous fixons pour la fin de cette section un entier  $m > 0$ . Nous allons travailler dans « l'hypercube »  $I = \mathbf{F}_2^m$ , qui a  $2^m$  éléments. Nous noterons  $J$  le complémentaire de l'origine, qui a donc  $2^m - 1$  éléments. L'ensemble des parties de  $I$  (avec la différence symétrique pour addition) est une version géométrique de l'espace des mots de  $2^m$  bits ; de même pour  $J$  et les mots de  $2^m - 1$  bits.

Les codes que nous considérerons seront des ensembles de parties de  $I$  ou de  $J$ , et donc de longueur  $n = 2^m$  ou  $n = 2^m - 1$ . Les codes de parties de  $I$  pourront être raccourcis en codes de parties de  $J$  ; les extensions paires des codes de parties de  $J$  seront des codes de parties de  $I$ , l'origine de l'hypercube « portant » ainsi le bit de parité.

Pour  $m < 3$ , nous aurons des codes triviaux, pour  $m = 3$ , nous retrouverons les cas traités ci-dessus.

### 7.2.1. Hyperplans

Par définition, un *hyperplan vectoriel* de  $I = \mathbf{F}_2^m$  est l'ensemble des zéros d'une forme linéaire  $\phi$  non nulle ; une telle forme est d'ailleurs tout simplement la somme d'une partie non vide de  $\{X_0, \dots, X_{m-1}\}$ . Il y a  $2^m - 1$  hyperplans vectoriels, autant que de formes linéaires non nulles.

Un *hyperplan affine* est par définition un translaté d'un hyperplan vectoriel. Pour chaque hyperplan vectoriel  $L$ , d'équation  $\phi = 0$ , le complémentaire de  $L$  est donné par l'équation  $\phi = 1$  ; c'est l'unique hyperplan affine parallèle à  $L$  et distinct de  $L$ . Il y a ainsi  $2^m - 1$  hyperplans affines ne passant pas par l'origine, donc en tout  $2^{m+1} - 2$  hyperplans affines. Chacun d'eux possède  $2^{m-1}$  points.

### 7.2.2. L'orthogonal du code de Hamming : le code simplexe

Nous allons maintenant nous intéresser à l'orthogonal d'un code de Hamming et d'abord expliciter un peu plus la définition des codes  $H_m$ . Rappelons qu'un tel code est défini par une matrice vérificatrice  $P$ , à  $m$  lignes, dont les colonnes sont exactement les  $n = 2^m - 1$  vecteurs non nuls de  $I = \mathbf{F}_2^m$ . Mais il n'existe pas d'ordre privilégié dans lequel prendre ces colonnes. Il est naturel en revanche, non pas de les énumérer, mais bien de prendre pour ensemble d'indices cet ensemble de colonnes, donc le complémentaire  $J$  de l'origine dans  $I$ . Ainsi, la colonne correspondant au vecteur non nul  $\mathbf{x} \in J$  est ce vecteur lui-même !

C'est donc  $J$  qui est l'ensemble d'indices naturel pour les mots du code  $H_m$ . C'est aussi l'ensemble d'indices de son orthogonal, engendré comme espace vectoriel par les lignes de  $P$ . Chaque ligne de la matrice  $P$  est ainsi la fonction caractéristique d'une partie de  $J$ , correspondant à un mot de base du code orthogonal. La ligne numéro  $i$  par exemple correspond à la partie de  $J$  formée des colonnes (vecteurs)  $\mathbf{x} = (x_0, \dots, x_{m-1})$  tels que  $x_i = 1$ . De même, la somme de la ligne  $i$  et la ligne  $j$  correspond à la partie formée des  $\mathbf{x}$  tels que  $x_i + x_j = 1$ . Plus généralement, les combinaisons linéaires des lignes de  $P$ , c'est-à-dire les mots du code orthogonal à  $H_m$ , correspondent simplement aux parties de  $J$  formées des  $\mathbf{x}$  tels que  $\phi(\mathbf{x}) = 1$ , où  $\phi$  est une forme linéaire quelconque sur  $\mathbf{F}_2^m$ . Pour  $\phi = 0$ , c'est la partie vide, pour  $\phi \neq 0$ , c'est un hyperplan affine. On a donc obtenu :

**PROPOSITION 7.1.** — *Interprété comme un ensemble de parties de  $J$ , le code orthogonal de  $H_m$  est formé du mot nul (partie vide) et de tous les hyperplans affines de  $\mathbf{F}_2^m$  ne passant pas par l'origine.*

On appelle ce code le *code simplexe* d'ordre  $m$  ; on le note  $S_m$ . Tous ses mots non nuls sont de poids  $2^{m-1}$ . Il est en particulier de distance minimale  $2^{m-1}$ , donc de paramètres  $(2^m - 1, m, 2^{m-1})$ .

Cela nous donne une construction géométrique du code de Hamming : c'est le dual du code simplexe.

**EXERCICE 7.3. [A] —** Que sont  $S_1$ ,  $S_2$  et  $S_3$  ?

**REMARQUE 7.2.** — La dénomination « simplexe » vient du fait que ce code est fait de mots dont les distances mutuelles sont toutes égales, ce qui est analogue à la situation d'un simplexe régulier ( $N + 1$  points dans  $\mathbf{R}^N$  dont les distances deux à deux sont toutes égales, comme un tétraèdre régulier dans  $\mathbf{R}^3$ ).

### 7.2.3. L'orthogonal du code de Hamming étendu : le code $R_{1,m}$

Passons maintenant au code de Hamming étendu. Notons  $P$  la matrice vérificatrice du code de Hamming introduite ci-dessus : ses lignes sont au nombre de  $m$ , ses colonnes sont les éléments du complémentaire  $J$  de l'origine dans  $I = \mathbf{F}_2^m$ . Le code de Hamming  $H_m$  et son orthogonal sont formés de parties de  $J$ . Pour obtenir le code étendu, on doit ajouter à  $J$  un élément portant le bit de parité ; il est naturel de prendre l'origine et on obtient ainsi  $\bar{H}_m$  comme ensemble de parties de  $I$  lui-même. De même pour son orthogonal. On a alors :

**PROPOSITION 7.3.** — *L'orthogonal du code de Hamming étendu  $\bar{H}_m$  est formé du mot vide, du mot plein, et de tous les hyperplans affines de  $\mathbf{F}_2^m$ .*

*Démonstration.* La matrice vérificatrice du code étendu est

$$\bar{P} = \begin{bmatrix} P & \mathbf{0} \\ \mathbf{1} & 1 \end{bmatrix},$$

où la dernière colonne commence par le vecteur origine de  $\mathbf{F}_2^m$ . Le même raisonnement que ci-dessus montre que les combinaisons linéaires des  $m$  premières lignes donnent toutes les équations  $\phi(\mathbf{x}) = 1$  avec  $\phi$  linéaire. L'adjonction de la dernière ligne donne toutes les équations  $\phi(\mathbf{x}) = 0$ . Pour  $\phi$  non nul, on obtient ainsi tous les hyperplans. Pour  $\phi = 0$ , la partie vide et la partie pleine.  $\square$

Ce code est noté  $R_{1,m}$ . Ses paramètres sont  $(2^m, m + 1, 2^{m-1})$ .

Par exemple, les paramètres du code  $R_{1,5}$  sont  $(32, 6, 16)$  ; ses taux sont  $k/n = 3/16$  et  $d/n = 1/2$ . Le code  $R_{1,5}$  a été utilisé pour coder sur 32 bits les 64 ( $= 2^6$ ) niveaux de gris des pixels des images de Mars transmises par le satellite *Mariner 9* en 1972. La raison de ce choix était technologique : les codes  $R_{1,m}$  possèdent des algorithmes de décodage très simples, réalisables par les circuits électroniques élémentaires disponibles à l'époque.

**EXERCICE 7.4. [A] —** Que sont les codes  $R_{1,1}$ ,  $R_{1,2}$  et  $R_{1,3}$  ?

### § 7.3. Codes de Reed-Muller

Nous allons généraliser ce qui précède à des codes de parties de  $I = \mathbf{F}_2^m$  définies par des conditions de degré supérieur.

*Cette section, assez difficile, peut être sautée en première lecture.*

#### 7.3.1. Fonctions booléennes et polynômes

On notera  $\mathcal{E}_m$  l'ensemble (c'est un espace vectoriel) des applications de  $\mathbf{F}_2^m$  dans  $\mathbf{F}_2$ . Si l'on fait correspondre comme d'habitude 0 à faux et 1 à vrai,  $\mathcal{E}_m$  est aussi l'ensemble des fonctions booléennes  $f(X_0, \dots, X_{m-1})$  de  $m$  variables booléennes.

Chaque partie A de l'hypercube I peut être définie par sa *fonction caractéristique*  $\chi_A$ , qui est l'élément de  $\mathcal{E}_m$  valant 1 sur A et 0 en dehors ; on peut associer inversement à chaque fonction  $f$  de  $\mathcal{E}_m$  son *support*, qui est la partie A de I formée des  $\mathbf{x}$  tels que  $f(\mathbf{x}) = 1$ . La somme des fonctions correspond alors à la différence symétrique des parties, c'est-à-dire à l'addition sur les mots. On identifie ainsi l'espace vectoriel  $\mathcal{E}_m$ , de dimension  $n = 2^m$ , à l'ensemble des parties de I. Dans la version « booléenne », cette identification n'est autre que l'association à chaque fonction booléenne de sa *table de vérité*. Nous voilà donc avec une nouvelle description de notre espace des mots de longueur  $n$ .

Prenons comme exemple le cas  $m = 2$ , donc  $n = 4$ , et choisissons pour fixer les idées d'énumérer les éléments de  $I = \mathbf{F}_2^2$  dans l'ordre  $(0, 0), (1, 0), (0, 1), (1, 1)$ . Pour  $f \in \mathcal{E}_2$ , posons

$$\vec{f} = (f(0, 0), f(1, 0), f(0, 1), f(1, 1)).$$

Alors l'application  $f \mapsto \vec{f}$  est un isomorphisme de l'espace vectoriel  $\mathcal{E}_2$  sur  $\mathbf{F}_2^4$ .

Rappelons par ailleurs que les opérations booléennes et les opérations arithmétiques dans  $\mathbf{F}_2$  sont reliées par les relations

$$ab = a \text{ AND } b, \quad a + b = a \text{ XOR } b, \quad a + b + ab = a \text{ OR } b, \quad 1 + a = \text{NOT } a.$$

Comme on sait que toute fonction  $f \in \mathcal{E}_m$  peut s'exprimer à l'aide des  $X_i$  et des opérateurs booléens fondamentaux, on voit qu'on peut aussi l'exprimer comme un polynôme (à coefficients dans  $\mathbf{F}_2$ ) en les  $X_i$ . Mais, puisqu'on a  $X_j^2 = X_j$  pour tout indice  $j$ , donc aussi  $X_j^r = X_j$  pour tout indice  $j$  et tout entier  $r > 1$ , on peut faire disparaître tous les exposants  $\geq 2$ .

Ainsi, si l'on appelle *monômes réduits* les monômes dont le degré en chaque variable ne dépasse pas 1 (c'est-à-dire les produits des parties de l'ensemble  $\{X_0, \dots, X_{m-1}\}$ ), *toute fonction de  $\mathcal{E}_m$  s'exprime comme somme de monômes réduits*.

On notera que ces monômes réduits sont au nombre de  $2^m = n$ , donc qu'ils forment une base de l'espace vectoriel des fonctions booléennes. Pour éviter une confusion assez naturelle, signalons explicitement que cette base *n'est pas* la base naturelle, formée des fonctions caractéristique des  $n$  points de  $\mathbf{F}_2^m$ . Prenons comme exemple le cas  $m = 2$ , donc  $n = 4$ , avec comme ci-dessus

$$\vec{f} = (f(0, 0), f(1, 0), f(0, 1), f(1, 1)).$$

Alors les monômes réduits sont 1,  $X_0$ ,  $X_1$  et  $X_0X_1$ , et on a  $\vec{1} = (1, 1, 1, 1)$ ,  $\vec{X_0} = (0, 1, 0, 1)$ ,  $\vec{X_1} = (0, 0, 1, 1)$  et  $\vec{X_0X_1} = (0, 0, 0, 1)$ . On reconnaît bien une base, car la matrice obtenue est triangulaire de diagonale unité.

Revenons au cas général. Il est clair que la fonction constante 1 correspond à la partie pleine, et qu'à l'autre extrême le monôme  $X_0 \cdots X_{m-1}$  donne la partie réduite à l'unique point  $(1, \dots, 1)$ . Plus généralement, un monôme réduit de degré  $r$ , c'est-à-dire un produit de  $r$  variables, donne une partie à  $2^{n-r}$  éléments : on donne la valeur 1 aux  $r$  variables du monôme, et des valeurs arbitraires aux  $n-r$  autres. *En particulier ce poids est pair sauf lorsque  $r = n$ .*

### 7.3.2. Définition des codes de Reed-Muller

Pour chaque  $r = 0, \dots, m$ , on appelle *code de Reed-Muller* (ou code de type RM) d'ordre  $r$  et de longueur  $n = 2^m$ , et on note  $R_{r,m}$ , le code linéaire formé des parties de  $\mathbf{F}_2^m$  dont la fonction caractéristique est un polynôme de degré total  $\leq r$ .

On peut de manière équivalente, et ce sera commode pour les calculs qui suivent, définir le code  $R_{r,m}$  comme le sous-espace de  $\mathcal{E}_m$  formé des fonctions booléennes qui peuvent s'exprimer par des polynômes de degré total  $\leq r$ .

Les codes  $R_{r,m}$  sont emboîtés par construction : on a

$$R_{0,m} \subset R_{1,m} \subset \cdots \subset R_{m-1,m} \subset R_{m,m}.$$

Puisque les monômes basiques de degré exactement  $s$  sont au nombre de  $\binom{m}{s}$ , on a :

$$\dim R_{r,m} = 1 + \binom{m}{1} + \cdots + \binom{m}{r}.$$

Les cas extrêmes redonnent les codes triviaux :

- ▷  $R_{0,m}$  est formé des deux fonctions constantes, c'est le code de répétition pure ;
- ▷  $R_{m-1,m}$ , qui est de codimension 1 et engendré par des vecteurs de poids pair, est l'ensemble des vecteurs de poids pair ;
- ▷  $R_{m,m}$  est le code total.

Nous avons déjà rencontré  $R_{1,m}$ . En effet, dire que la fonction booléenne  $f = \chi_A$  s'exprime comme un polynôme de degré 1 signifie que A est un

hyperplan affine. Ainsi, le code  $R_{1,m}$  est formé de la partie vide, de la partie pleine et des hyperplans affines ; comme on l'a vu plus haut, c'est l'orthogonal du code de Hamming étendu.

**PROPOSITION 7.4.** — Pour  $r < m$ , le code orthogonal de  $R_{r,m}$  est  $R_{m-r-1,m}$ .

*Démonstration.* Par construction, pour  $\mathbf{a} \in R_{r,m}$  et  $\mathbf{b} \in R_{m-r-1,m}$ , on a  $\mathbf{a} \cdot \mathbf{b} \in R_{m-1,m}$ , de sorte que  $w(\mathbf{a} \cdot \mathbf{b})$  est pair, ce qui signifie que  $\langle \mathbf{a} | \mathbf{b} \rangle = 0$ . Ainsi,  $R_{r,m}$  et  $R_{m-r-1,m}$  sont orthogonaux. Mais leurs dimensions sont complémentaires, puisqu'on a

$$\begin{aligned}\dim R_{r,m} + \dim R_{m-r-1,m} &= (1 + \cdots + \binom{m}{r}) + ((\binom{m}{m-r-1} + \cdots + 1) \\ &= (1 + \cdots + \binom{m}{r}) + ((\binom{m}{r+1} + \cdots + \binom{m}{m})) \\ &= 2^m = n.\end{aligned}$$

□

**REMARQUE 7.5.** — Il est naturel de convenir que  $R_{-1,m}$  est le code nul, orthogonal du code plein  $R_{m,m}$ , ce qui concorde avec la proposition précédente.

### 7.3.3. Description itérative des codes de Reed-Muller

On peut construire les codes  $R_{r,m}$  de façon itérative. Chaque élément de  $\mathcal{E}_{m+1}$  s'écrit de façon unique  $a = u + X_m v$ , avec  $u \in \mathcal{E}_m$  et  $v \in \mathcal{E}_m$ . On a d'ailleurs  $a(X_0, \dots, X_{m-1}, 0) = u(X_0, \dots, X_{m-1})$  et  $a(X_0, \dots, X_{m-1}, 1) = u(X_0, \dots, X_{m-1}) + v(X_0, \dots, X_{m-1})$ . Pour un ordre convenable des composantes des vecteurs de  $\mathbb{F}_2^{2^m+1}$ , cela s'écrit

$$a = (u, u + v).$$

Dire que l'on a  $a \in R_{r,m+1}$ , c'est dire que l'on a à la fois  $u \in R_{r,m}$  et  $v \in R_{r-1,m}$ .

Ainsi,  $R_{r,m+1}$  se déduit de  $R_{r,m}$  et  $R_{r-1,m}$  par la construction baptisée  $\inf(2d, d')$  en 6.3.1. Voici deux applications :

**PROPOSITION 7.6.** — La distance minimale de  $R_{r,m}$  est  $2^{m-r}$ .

*Démonstration.* Si on note  $\delta_{r,m}$  la distance minimale de  $R_{r,m}$ , on déduit du lemme 6.1.1 la relation

$$\delta_{r,m+1} = \inf(2\delta_{r,m}, \delta_{r-1,m}).$$

Mais on a vu que l'on a  $\delta_{0,m} = n = 2^m$  et  $\delta_{m,m} = 1$  pour tout  $m$ . Écrivons  $\delta_{r,m} = 2^{m-r} + \varepsilon_{r,m}$ , de sorte que  $\varepsilon_{0,m} = 0$  et  $\varepsilon_{m,m} = 0$ . On a alors  $\varepsilon_{r,m+1} = \inf(2\varepsilon_{r,m}, \varepsilon_{r-1,m})$ . On en déduit sans difficulté par récurrence double qu'on a  $\varepsilon_{r,m} = 0$ , ce qui implique la proposition. □

**PROPOSITION 7.7.** — Pour  $m \geq 2$ , le code  $R_{m-2,m}$  est un code de Hamming étendu de longueur  $2^m$ .

*Démonstration.* Donnons deux démonstrations de la proposition. On peut la déduire de la proposition 7.4 et de l'identification de  $R_{1,m}$  à l'orthogonal du code de Hamming étendu. On peut aussi la vérifier par récurrence sur  $m$ . Pour  $m = 2$ , les codes  $R_{0,2}$  et  $\bar{H}_2$  sont tous deux le code de répétition pure. Mais, puisque  $R_{m-1,m}$  est le code de parité, on voit que la construction qui fait passer de  $R_{m-2,m}$  à  $R_{m-1,m+1}$  est la même que celle qui a été faite dans 6.19 pour passer de  $\bar{H}_m$  à  $\bar{H}_{m+1}$ .  $\square$

### 7.3.4. Mots de poids minimal du code $R_{r,m}$

#### Fonctions caractéristiques des sous-espaces affines

Pour passer du cas  $r = 1$  au cas général, nous aurons besoin de la notion générale de *sous-espace affine* de  $\mathbf{F}_2^m$ . Ce sont par définition les parties de  $\mathbf{F}_2^m$  qui peuvent s'écrire comme intersection d'hyperplans affines ou, de manière équivalente, sont obtenues par translation à partir d'un sous-espace vectoriel (on néglige le cas de la partie vide). Une telle partie a une codimension  $r$  (le nombre minimum d'hyperplans dont elle est intersection) et une dimension  $m-r$  (la dimension au sens usuel de l'espace vectoriel dont elle est translatée). Elle a  $2^{m-r}$  éléments.

Nous utiliserons ci-dessous la forme suivante du *théorème d'échange* de l'algèbre linéaire.

**LEMME 7.8.** — Soit  $A$  un sous-espace affine de codimension  $r$  de  $\mathbf{F}_2^m$ . Quitte à modifier l'ordre des variables  $X_0, \dots, X_{m-1}$ , on peut supposer que  $A$  est décrit par  $r$  équations de la forme

$$X_{m-r} = a_{m-r,0}X_0 + \cdots + a_{m-r,m-r-1}X_{m-r-1} + b_{m-r},$$

...

$$X_{m-1} = a_{m-1,0}X_0 + \cdots + a_{m-1,m-r-1}X_{m-r-1} + b_{m-1},$$

avec  $a_{i,j} \in \mathbf{F}_2$  et  $b_i \in \mathbf{F}_2$ .

Démontrons d'abord un résultat intermédiaire :

**PROPOSITION 7.9.** — Soit  $A$  un sous-espace affine de codimension  $r$  de  $\mathbf{F}_2^m$  et soit  $\chi_A$  sa fonction caractéristique.

a) Alors  $\chi_A$  est un mot de  $R_{r,m}$ , de poids  $2^{m-r}$ .

b) Soit  $s$  un entier avec  $r \leq s \leq m$  et soit  $f \in R_{s,m}$  une fonction booléenne, telle que  $f(x) = 0$  pour tout  $x \notin A$ . Alors  $f$  peut s'écrire comme produit  $\chi_A g$ , avec  $g \in R_{s-r,m}$ .

*Démonstration.* Par définition,  $A$  peut se définir par la conjonction de  $r$  équations affines, que l'on peut écrire sous la forme  $\phi_i = 1$ , avec  $\phi_i \in R_{1,m}$ . On a donc  $\chi_A = \prod \phi_i \in R_{r,m}$ . Par ailleurs, on a  $w(\chi_A) = \#A = 2^{m-r}$ . On a ainsi prouvé a).

Pour démontrer b), appliquons le lemme 7.8 : on peut fixer un ordre convenable des variables et choisir les  $\phi_i$  de la forme  $\phi_i = X_i + a_i$ , où  $a_i$  ne dépend que des

variables  $X_0, \dots, X_{m-r-1}$  et où  $i$  parcourt l'intervalle  $\llbracket m-r, m-1 \rrbracket$ . On peut écrire comme plus haut  $f = u + X_{m-1}v$ , avec  $u \in R_{s,m-1}$  et  $v \in R_{s-1,m-1}$ . Cela donne

$$f = (X_{m-1} + a_{m-1})v + b = \phi_{m-1}v + b,$$

avec  $v \in R_{s-1,m-1}$  et  $b = u + a_{m-1}v \in R_{s,m-1}$ .

Démontrons que l'on a  $b = 0$ . Soit donc  $y = (x_0, \dots, x_{m-2}) \in F_2^{m-1}$  et prouvons que  $b(y) = 0$ . Posons  $x_{m-1} = a_{m-1}(x_0, \dots, x_{r-m+1})$  et  $x = (x_0, \dots, x_{m-1})$ . On a alors par construction  $\phi_{m-1}(x) = 0$ , donc  $x \notin A$ . L'hypothèse faite sur  $f$  implique alors  $f(x) = 0$ . Mais, puisque  $f(x) = \phi_{m-1}(x)v(x) + b(y)$ , on obtient  $b(y) = 0$ , ce qu'on voulait démontrer. Ainsi  $b$  est bien nul et l'on a  $f = \phi_{m-1}v$ .

On écrit alors de même  $v = \phi_{m-2}w + c$ , avec  $w \in R_{s-2,m-2}$  et  $c \in R_{s-1,m-2}$ . Raisonnant comme ci-dessus, on obtient  $c = 0$ , donc  $f = \phi_{m-1}\phi_{m-2}w$ . On continue jusqu'à trouver une relation  $f = \phi_{m-1} \cdots \phi_{m-r}g = \chi_A g$  avec  $g \in R_{s-r,m-r}$ .  $\square$

**REMARQUE 7.10.** — Sous l'hypothèse de  $b$ ), on a aussitôt  $(f(x) = 1) \Rightarrow (\chi_A(x) = 1)$ , autrement dit  $f \Rightarrow \chi_A$ , ou encore  $f = \chi_A$  AND  $f$ , soit aussi  $f = \chi_A f$ , de sorte que  $f$  s'écrit bien sous la forme  $\chi_A g$ , avec  $g = f$ . Mais le degré du quotient  $g$  obtenu ainsi est  $s$ , et non  $s-r$  comme on le désire.

**THÉORÈME 7.11.** — *Les mots de poids minimal du code  $R_{r,m}$  sont exactement les fonctions caractéristiques des sous-espaces affines de codimension  $r$ .*

*Démonstration.* On a déjà vu que la distance minimale du code  $R_{r,m}$  vaut  $2^{m-r}$  et que les fonctions caractéristiques des sous-espaces affines de codimension  $r$  sont des mots de  $R_{r,m}$  de poids  $2^{m-r}$ . Il reste à prouver la réciproque.

Soit donc  $A$  une partie de  $F_2^m$ , avec  $\#A = 2^{m-r}$ , telle que  $a = \chi_A$  appartienne à  $R_{r,m}$ . Il s'agit de prouver que  $A$  est un sous-espace affine. Nous allons raisonner par récurrence sur  $m$ , l'énoncé étant connu pour  $m = 1$ . Écrivons comme plus haut  $a = u + X_{m-1}v$ , avec  $u \in R_{r,m-1}$  et  $v \in R_{r-1,m-1}$  et distinguons deux cas suivant que  $v$  est nul ou non.

Si  $v = 0$ , alors  $a = u$  ne dépend pas de  $X_m$ . Soit  $U$  la partie de  $F_2^{m-1}$  telle que  $u = \chi_U$ . On a  $A = U \times F_2$ . Le poids de  $a$  est double du poids de  $u$ , de sorte que  $u$  est un mot de poids minimal de  $R_{r,m-1}$ . Par récurrence, on en conclut que  $U$  est un sous-espace affine de  $F_2^{m-1}$ , donc que  $A$  est un sous-espace affine de  $F_2^m$ .

Plaçons-nous enfin dans le second cas  $v \neq 0$  et notons  $V$  la partie de  $F_2^{m-1}$  telle que  $v = \chi_V$ . Soit  $x = (x_0, \dots, x_{m-1}) = (y, x_{m-1})$  un point quelconque de  $F_2^m$ . Pour  $y \in V$ , on a  $v(y) = 1$ , donc  $a(x) = x_{m-1} + u(y)$ . Ainsi, si l'on prend  $x_{m-1} = u(y) + 1$ , on obtient un point de  $A$ . Cela montre que  $\#V \leq \#A = 2^{m-r}$ . L'hypothèse de récurrence appliquée à  $v$  montre alors que  $V$  est un sous-espace affine de codimension  $r-1$  de  $F_2^{m-1}$ , et aussi que l'on a ainsi obtenu tous les points de  $A$ . Il n'existe donc aucun point  $x = (y, x_{m-1})$  de  $A$  avec  $v(y) = 0$ . Par conséquent, pour  $v(y) = 0$ , on a nécessairement  $u(y) = 0$ , sinon on obtiendrait deux nouveaux points de  $A$ , à savoir  $(y, 0)$  et  $(y, 1)$ . Appliquant à  $u$  et  $V$  la partie b) de la proposition 7.9, on en conclut que l'on peut écrire  $u = vg$ , avec  $g \in R_{1,m-1}$ . Il en résulte que  $a = (X_{m-1} + g)v$ , de sorte que  $(y, x_{m-1}) \in A$  équivaut à  $y \in V$  et  $x_{m-1} = g(y) + 1$ . Mais cela décrit bien  $A$  comme sous-espace affine, ce qu'on voulait obtenir.  $\square$

**REMARQUE 7.12.** — On peut prouver que les mots qu'on vient de déterminer, à savoir les  $\chi_A$  avec  $A$  affine de codimension  $r$ , engendrent le sous-espace vectoriel  $R_{r,m}$ , ce qui donne une définition directe de ce code, comme sous-espace vectoriel engendré par ces fonctions. Ce fait est notablement plus difficile à démontrer que le précédent. On trouvera une démonstration dans [MacWilliams, Sloane], ch. 13, §5, *theorem 10*.

## § 7.4. Géométries finies

### 7.4.1. Corps finis

Pour chaque entier premier  $p$ , l'anneau à  $p$  éléments  $\mathbf{Z}/p\mathbf{Z}$  des classes modulo  $p$  est un *corps*. Plus généralement, nous verrons plus tard que, pour tout entier premier  $p$  et tout entier  $r > 0$ , il existe un corps fini à  $p^r$  éléments (théorème 9.26) et qu'inversement le nombre des éléments d'un corps fini est toujours une puissance d'un nombre premier (théorème 9.5). Il existe ainsi par exemple des corps à 2, 3, 4, 5, 7, 8, 9 éléments, mais pas de corps à 6 ou à 10 éléments.

### 7.4.2. Espaces affines ou projectifs finis

Pour tout corps commutatif  $K$  et tout entier  $m$ , on peut considérer l'espace vectoriel  $K^m$ . Si  $K$  est fini, à  $q$  éléments, cet espace a  $q^m$  éléments. On rappelle qu'un *sous-espace affine* de dimension  $r$  de  $K^m$  est par définition un translaté d'un sous-espace vectoriel  $V$  de  $K^m$ , de dimension  $r$ . On parle de droite affine, de plan affine, d'hyperplan affine, si  $r$  vaut 1, 2, ou  $m - 1$ . On parle de droite vectorielle, de plan vectoriel, ou d'hyperplan vectoriel lorsque le sous-espace considéré passe par l'origine. Si  $K$  a  $q$  éléments, un sous-espace affine de dimension  $r$  a  $q^r$  éléments.

L'*espace projectif*  $\mathbf{P}_{m-1}(K)$  s'obtient en identifiant entre eux les points de  $K^m$  qui sont alignés avec l'origine. Ainsi, les points de  $\mathbf{P}_{m-1}(K)$  correspondent bijectivement aux droites vectorielles de  $K^m$ . Si  $K$  est fini à  $q$  éléments, alors  $\mathbf{P}_{m-1}(K)$  possède  $(q^m - 1)/(q - 1)$  éléments. Une *droite projective* de  $\mathbf{P}_{m-1}(K)$  s'obtient à partir d'un plan vectoriel de  $K^m$ . C'est l'ensemble des points de  $\mathbf{P}_{m-1}(K)$  qui correspondent à des droites du plan considéré ; il a  $(q^2 - 1)/(q - 1) = q + 1$  éléments.

Considérons successivement trois cas particuliers.

a) Prenons d'abord  $K = \mathbf{F}_2$ , donc  $q = 2$ . Alors chaque droite vectorielle ne comprend en plus de l'origine qu'un *seul point*. Ainsi, l'espace projectif  $\mathbf{P}_{m-1}(\mathbf{F}_2)$  est simplement le complémentaire de l'origine dans l'hypercube  $\mathbf{F}_2^m$ . Les droites projectives de  $\mathbf{P}_{m-1}(\mathbf{F}_2)$  ont trois points et sont les complémentaires de l'origine dans les plans vectoriels de  $\mathbf{F}_2^m$ , qui en ont quatre.

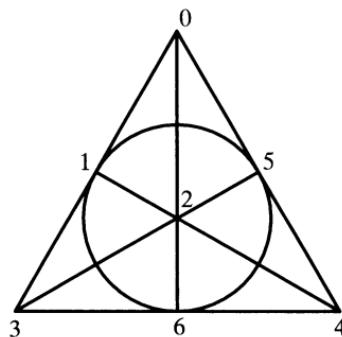
Le code simplexe  $S_m$  est le code de parties de  $\mathbf{P}_{m-1}(\mathbf{F}_2)$  formé de la partie vide et des complémentaires des hyperplans.

b) Revenons à un corps  $K$  général et prenons  $m = 2$ , donc  $m - 1 = 1$ . Il s'agit donc de la *droite projective*  $\mathbf{P}_1(K)$ , dont les points sont par définition les droites vectorielles du plan  $K^2$ . Mais, à tout point  $(x, y) \neq (0, 0)$  de  $K^2$ , associons le quotient  $x/y$ , qui est un élément de  $K$  lorsque  $y \neq 0$  et le symbole  $\infty$  lorsque  $y = 0$ . Deux points alignés avec l'origine donnent la même image. Cela permet d'identifier la droite projective  $\mathbf{P}_1(K)$  avec  $K \cup \{\infty\}$ , c'est-à-dire avec la droite affine complétée d'un point à l'infini.

c) Regardons enfin le cas des *plans projectifs* (donc  $m = 3$  et  $m - 1 = 2$ ). On considère donc l'espace vectoriel  $V = K^3$ , et  $P = \mathbf{P}_2(K)$  est par définition l'ensemble des droites vectorielles de  $V$ . Il sera commode de démultiplier les notations en disant plutôt que les points  $x$  de  $P$  repèrent les droites vectorielles  $V_x$  de  $V$ . Pour chaque sous-espace vectoriel  $W$  de dimension 2 de  $V$  (c'est-à-dire chaque plan de  $V$  passant par l'origine), on obtient une droite projective  $d$  de  $P$  : l'ensemble des  $x \in P$  tels que  $V_x \subset W$ . De manière équivalente, on peut dire que les droites  $d$  de  $P$  repèrent les plans vectoriels  $W_d$  de  $V$ , et les propriétés  $x \in d$  et  $V_x \subset W_d$  sont équivalentes.

**REMARQUE 7.13.** — Notons  $V^*$  l'espace vectoriel dual de  $V$ . Les plans vectoriels de  $V$  sont exactement les orthogonaux des droites vectorielles de  $V^*$ . On peut donc aussi considérer que les points (resp. droites) de  $P$  correspondent aux droites vectorielles de  $V$  (resp.  $V^*$ ) et alors la relation d'incidence  $x \in d$  n'est autre que la relation d'orthogonalité entre droites vectorielles de  $V$  et de  $V^*$ .

Le cas le plus simple est celui où  $K$  est le corps  $\mathbf{F}_2$  à deux éléments. Le plan projectif  $\mathbf{P}_2(\mathbf{F}_2)$  possède 7 points et 7 droites, chaque droite étant formée de 3 points, chaque point étant commun à 3 droites. On l'appelle souvent le « plan de Fano ».



C'est la situation que nous avons rencontrée dans la description géométrique du code de Hamming  $H_3$  en 7.1.2 : les mots de codes sont la partie vide, la partie pleine, les 7 droites et leurs 7 complémentaires ; la partie vide et les sept complémentaires de droites forment le code simplexe  $S_3$ , partie paire de  $H_3$ .

### 7.4.3. Plans projectifs « abstraits »

On généralise la situation précédente en appelant *plan projectif* un ensemble  $P$  dont certaines parties sont appelées *droites*, de manière que les axiomes suivants soient satisfaits :

(PP0)  $P$  n'est pas réduit à une droite, ni à la réunion de deux droites ;

(PP1) par deux points distincts passe une et une seule droite ;

(PP2) deux droites distinctes se coupent en un point (ce point est unique d'après l'axiome précédent).

L'axiome (PP0) élimine par exemple le cas où  $P$  se réduirait à trois points, formant deux à deux trois droites.

**REMARQUE 7.14.** — En fait la relation entre points et droites est parfaitement symétrique. On pourrait tout aussi bien se donner deux ensembles  $P$  et  $P^*$ , appeler points les éléments  $x$  de  $P$  et droites les éléments  $d$  de  $P^*$ , et supposer donnée une *relation d'incidence* «  $d$  passe par  $x$  » satisfaisant à (PP1) et (PP2). Il résulte en effet de ces axiomes qu'une droite est bien déterminée par la donnée des points par lesquels elle passe, donc s'interprète aussi comme une partie de  $P$ . Cette nouvelle présentation a l'avantage de montrer qu'on peut intervertir les rôles de  $P$  et  $P^*$ , les points de l'un correspondant aux droites de l'autre (*dualité*).

Dans le cas usuel, avec  $P = \mathbf{P}_2(K)$ , où l'on suppose en outre que le corps  $K$  est fini, à  $q$  éléments, alors  $P$  possède  $\frac{q^3-1}{q-1} = q^2 + q + 1$  points et de même  $q^2 + q + 1$  droites, et chaque droite possède  $\frac{q^2-1}{q-1} = q + 1$  points. En fait, ce dénombrement reste valable dans le cas général :

**PROPOSITION 7.15.** — *Soit  $P$  un plan projectif fini. Il existe un entier  $a > 1$  tel que chaque droite possède  $a + 1$  points, que par chaque point passe  $a + 1$  droites, et que  $P$  possède  $a^2 + a + 1$  points et  $a^2 + a + 1$  droites.*

*Démonstration.* Montrons d'abord que toutes les droites ont le même nombre d'éléments. Soient  $d$  et  $d'$  deux droites. Il existe un point  $x$  qui n'appartient à aucune des deux (axiome (PP0)). Pour chaque point  $y$  de  $d$ , la droite passant par  $x$  et  $y$  recoupe  $d'$  en un point  $z$ , et l'application  $y \mapsto z$  est une bijection de  $d$  sur  $d'$ . Ainsi toutes les droites ont le même nombre d'éléments ; notons-le  $a + 1$ . Soit maintenant  $x$  un point de  $P$ . Choisissons une droite  $d$  ne passant pas par  $x$ . Chaque droite passant par  $x$  recoupe  $d$  en un point, et on construit ainsi une bijection entre droites passant par  $x$  et points de  $d$ . Il existe donc exactement  $a + 1$  droites passant par  $x$ . Il reste à compter le nombre de points et de droites de  $P$ . Mais, dans la situation précédente, chacune des droites considérées possède  $a$  points distincts de  $x$ , ce qui donne en tout  $(a + 1)a + 1$  points. Le calcul du nombre de droites est analogue.  $\square$

On dit que  $a$  est l'*ordre* du plan projectif  $P$ .

La liste exacte des entiers  $a$  pour lesquels il existe un plan projectif d'ordre  $a$  n'est pas connue. Elle contient en tout cas tous les ordres des corps finis, donc tous les puissances des nombres premiers. On ne connaît aucun  $a$  qui

ne soit pas une puissance d'un nombre premier et pour lequel existe un plan projectif d'ordre  $a$ .

**REMARQUE 7.16.** — Mais, pour les  $a$  qui sont les ordres des plans projectifs classiques, il peut exister d'autres plans projectifs que ces derniers.

Les plus petits cas à traiter sont 6, 10, 12, 14... On a un critère numérique partiel<sup>1</sup> qui implique par exemple qu'il n'existe pas de plan projectif d'ordre 6, 14 ou 21. Le premier cas « difficile »,  $a = 10$ , est resté ouvert très longtemps ; on sait maintenant, après de très longs calculs sur ordinateur, que la réponse est négative : il n'existe pas de plan projectif d'ordre 10. Le cas suivant,  $a = 12$ , est toujours ouvert.

## § 7.5. Systèmes de Steiner

### 7.5.1. Définition des systèmes de Steiner

**DÉFINITION 7.17.** — Soient  $r$ ,  $s$  et  $n$  trois entiers avec  $0 < r < s < n$ . Fixons un ensemble  $I$  à  $n$  éléments. Un système de Steiner de type  $(r, s, n)$  est un ensemble  $S$  de parties de  $I$ , qui ont toutes  $s$  éléments, possédant la propriété suivante : toute partie à  $r$  éléments de  $I$  est contenue dans un unique élément de  $S$ .

Cette définition se décompose en deux. L'assertion d'unicité peut se traduire par : pour  $a$  et  $b$  dans  $S$ , avec  $a \neq b$ , on a  $\#a \cap b < r$ . Si on suppose cette condition satisfaite, et qu'on compte les parties à  $r$  éléments de  $I$  qui sont contenues dans un élément (non fixé) de  $S$ , on en trouve  $\binom{s}{r} \#S$  ; l'assertion d'existence s'écrit donc  $\binom{n}{r} = \binom{s}{r} \#S$ , ou encore

$$\#S = \frac{n(n-1)\cdots(n-r+1)}{s(s-1)\cdots(s-r+1)}.$$

### 7.5.2. Exemples

Le premier cas est  $r = 1$ . La condition signifie simplement que  $S$  est une partition de  $I$  et la formule précédente s'écrit  $\#S = n/s$ .

Dans le cas  $r = 2$ , la condition signifie que deux éléments distincts de  $I$  appartiennent à un élément de  $S$  et un seul. On a alors envie de baptiser « droites » les éléments de  $S$ . La géométrie finie va nous fournir des exemples de systèmes de Steiner de type  $(2, s, n)$ .

Ainsi, si  $K$  est un corps fini à  $q$  éléments, les droites du plan affine  $K^2$  forment un système de Steiner de type  $(2, q, q^2)$  : par deux points distincts quelconques passe une droite et une seule.

2. Le théorème de Bruck-Ryser-Chowla : si  $a$  est congru à 1 ou 2 modulo 4, et s'il existe un plan projectif d'ordre  $a$ , alors  $a$  est somme de deux carrés.

De même, dans l'espace projectif  $\mathbf{P}_m(K)$  de dimension  $m > 1$  sur  $K$ , les droites projectives forment un système de Steiner de type  $(2, q + 1, n)$  avec

$$n = \#\mathbf{P}_m(K) = \frac{q^{m+1} - 1}{q - 1} = q^m + \cdots + q + 1.$$

## Plans projectifs

Cela se généralise aux plans projectifs généraux, vu la proposition 7.15 :

**PROPOSITION 7.18.** — *Les droites d'un plan projectif d'ordre  $a$  forment un système de Steiner de type  $(2, a + 1, a^2 + a + 1)$ .*

En fait, la réciproque est vraie :

**PROPOSITION 7.19.** — *Tout système de Steiner de type  $(2, a + 1, a^2 + a + 1)$ , avec  $a > 1$ , est formé des droites d'un plan projectif d'ordre  $a$ .*

*Démonstration.* Baptisons « droites » les parties distinguées du système de Steiner. Chaque droite est formée de  $s = a + 1$  points. Puisque  $P$  possède  $n = a^2 + a + 1$  points, le nombre de droites est

$$\frac{n(n-1)}{s(s-1)} = \frac{n(a^2+a)}{(a+1)a} = n.$$

Les axiomes (PP0) et (PP1) sont satisfaits. Il s'agit de prouver l'axiome (PP2). Pour cela, nous allons compter le nombre de couples  $(d, d')$  de droites qui se coupent ; il s'agit de prouver qu'il vaut  $n(n-1)$ . Mais, vu (PP1), c'est aussi le nombre de triplets  $(x, d, d')$  où  $x$  est un point,  $d$  et  $d'$  sont des droites et où  $d \cap d' = \{x\}$ . Or, il y a  $n$  valeurs possibles pour  $x$ , et ensuite  $(a+1)a$  valeurs possibles pour le couple  $(d, d')$ , ce qui donne bien la valeur attendue, puisque  $a(a+1) = n-1$ .  $\square$

## Espaces affines sur $\mathbf{F}_2$

Dans un espace affine de dimension  $m > 2$  sur le corps  $\mathbf{F}_2$  (on a donc  $n = 2^m$ ), les plans affines forment un système de Steiner de type  $(3, 4, n)$  : en effet, toute partie à 3 éléments est contenue dans un unique plan affine (car trois points ne peuvent jamais être alignés).

### 7.5.3. Codes et systèmes de Steiner

Beaucoup de codes permettent de construire des systèmes de Steiner. Supposons en effet donné un code binaire  $C$ , de longueur  $n$ , de distance minimale  $d$ . Considérons les mots de codes comme des parties de l'ensemble  $I$  des  $n$  bits et notons  $S$  l'ensemble des mots de codes de poids minimal (donc de poids  $d$ ).

Soit d'abord  $r > d/2$  un entier. Alors toute partie  $a$  de  $I$  ayant  $r$  éléments est contenue dans *au plus* un élément de  $S$ . Si en effet  $m$  et  $m'$  sont deux

éléments de  $S$  contenant  $a$ , on a  $\#(m \cap m') \geq r$ , et la différence symétrique  $m \oplus m'$  a au plus  $2(d - r) < d$  éléments, ce qui implique  $m = m'$ , par définition de  $d$ . Pour prouver que  $S$  est un système de Steiner de type  $(r, d, n)$ , il suffira alors de vérifier l'assertion d'existence, par exemple en prouvant que  $\#S$  a bien la valeur voulue.

Considérons inversement un système de Steiner  $S$  de type  $(r, d, n)$ , et supposons  $r > d/2$ . Identifions les éléments de  $S$  à des mots de  $\mathbf{F}_2^n$  de poids  $d$ . Soit  $m$  un mot tel que  $w(m) \geq r$ . On peut donc écrire  $m = a + m'$  avec  $w(a) = r$  et  $w(m') = w(m) - r$ . Il existe un mot  $u$  de  $S$  contenant  $a$  (on ne se sert en fait que de la propriété d'existence), de sorte que l'on peut écrire  $u = a + u'$  avec  $w(u') = d - r < r$ . Ainsi, on a  $w(m + u) = w(m' + u') \leq w(m') + w(u') < w(m) - r + r = w(m)$ . Itérant cette opération et raisonnant par récurrence sur la longueur de  $m$ , on en déduit que tout mot peut s'écrire comme somme d'un certain nombre d'éléments de  $S$  et d'un mot de poids  $< r$ . Si en particulier, le système  $S$  se trouve être formé des mots de poids minimal d'un code linéaire  $C$ , on en déduit que tout mot de  $C$  est somme de mots de poids minimal (puisque le mot résiduel est de poids  $< r < d$ , donc est nul).

**PROPOSITION 7.20.** — *Soit  $C$  un code binaire parfait<sup>3</sup>, de longueur  $n$ , de distance minimale impaire  $d = 2t + 1$ , donc  $t$ -correcteur.*

a) *Les mots de poids minimal de  $C$  forment un système de Steiner de type  $(t + 1, 2t + 1, n)$ . Ils engendrent le code  $C$ .*

b) *Les mots de poids minimal du code étendu  $\bar{C}$  forment un système de Steiner de type  $(t + 2, 2t + 2, n + 1)$ . Ils engendrent le code  $\bar{C}$ .*

*Démonstration.* Comme on l'a vu ci-dessus, il s'agit de prouver les assertions d'existence. Soit d'abord  $a$  une partie à  $t + 1$  éléments de  $I = \{0, \dots, n - 1\}$ . Pour démontrer a), il faut prouver qu'il existe un mot  $m \in C$ , de poids  $2t + 1$ , contenant  $a$ . Or, puisque  $C$  est parfait, il existe un mot de code  $m \in C$  avec  $w(a - m) \leq t$ ; mais cela implique  $1 \leq w(m) \leq 2t + 1 = d$ , donc  $w(m) = d = 2t + 1$ .

Prouvons maintenant b). La distance minimale de  $\bar{C}$ , qui est paire, est évidemment  $d + 1 = 2t + 2$ . Soit  $b$  une partie de  $I \cup \{n\}$  à  $t + 2$  éléments. Il s'agit de prouver qu'il existe un mot du code  $\bar{C}$  de poids  $2t + 2$  contenant  $b$ . Or les mots de  $\bar{C}$  sont de deux types : les mots  $(m, 1)$ , où  $m$  est un mot de poids  $2t + 1$  de  $C$ , et les mots  $(m, 0)$ , où  $m$  est un mot de poids  $2t + 2$  de  $C$ . Distinguons deux cas, suivant que  $b = (a, 1)$ , avec  $w(a) = t + 1$ , ou  $b = (a, 0)$ , avec  $w(a) = t + 2$ . Dans le premier cas, tout mot contenant  $b$  a nécessairement son dernier bit à 1, et on est ramené au cas déjà traité. Supposons donc  $b = (a, 0)$  avec  $w(a) = t + 2$ . Puisque  $C$  est parfait, il existe  $m \in C$  avec  $w(a - m) \leq t$ , donc  $2 \leq w(m) \leq 2t + 2$ . Comme  $2 < d = 2t + 1$ , cela implique

3. On ne se méprendra pas sur l'apparente généralité de cet énoncé. Rappelons en effet (remarque 6.9) qu'en dehors du code 3-correcteur de Golay, auquel nous appliquerons plus loin cette proposition, les seuls codes binaires parfaits non triviaux sont 1-correcteurs.

$w(m) = 2t + 2$  ou  $w(m) = 2t + 1$ . Dans le premier cas  $(m, 0)$  convient, dans le second cas  $(m, 1)$  convient.  $\square$

**COROLLAIRE 7.21.** — a) Le code  $C$  possède  $\frac{n(n-1)\cdots(n-t)}{(2t+1)\cdots(t+1)}$  mots de poids  $2t + 1$ .

b) Le code  $C$  possède  $\frac{(n-2t-1)n(n-1)\cdots(n-t)}{(2t+2)(2t+1)\cdots(t+1)}$  mots de poids  $2t + 2$ .

c) Le code  $\bar{C}$  possède  $\frac{(n+1)n\cdots(n-t)}{(2t+2)\cdots(t+1)}$  mots de poids  $2t + 2$ .

*Démonstration.* Les parties a) et c) résultent directement de la proposition, et des remarques qui la précèdent. On en déduit b) par différence.  $\square$

Prenons comme premier exemple le cas du code de Hamming de type  $(7, 4, 3)$  et du code étendu de type  $(8, 4, 4)$ . Les mots de  $C$  de poids 3 forment un système de Steiner de type  $(2, 3, 7)$ ; ils engendrent  $C$  et sont au nombre de  $(7 \times 6)/(3 \times 2) = 7$ ; ce sont en fait les droites projectives du plan de Fano  $P_2(\mathbf{F}_2)$ .

De même, les mots de  $\bar{C}$  de poids 4 forment un système de Steiner de type  $(3, 4, 8)$ ; ils engendrent  $\bar{C}$  et sont au nombre de  $(8 \times 7 \times 6)/(4 \times 3 \times 2) = 14$ ; ce sont en fait les plans de l'espace affine  $\mathbf{F}_2^3$ .

Plus généralement, si  $C$  est un code 1-correcteur parfait, auquel cas on sait que  $n$  est de la forme  $2^r - 1$  (voir la proposition 6.8), on obtient

$$n(n-1)/6 = (2^r - 1)(2^{r-1} - 1)/3$$

mots de poids 3 et

$$(n-3)n(n-1)/24 = (2^r - 1)(2^{r-1} - 1)(2^{r-2} - 1)/3$$

mots de poids 4.

Dans le (dernier) cas du code de Golay, de paramètres  $(23, 12, 7)$ , que nous verrons en 13.2 et qui est 3-correcteur parfait, les mots de poids minimaux de son extension paire forment un système de Steiner de type  $(5, 8, 24)$ .

## § 7.6. Automorphismes

Considérons un code binaire, représenté comme ensemble de parties d'un ensemble fini  $I$  (par exemple  $\{0, \dots, n-1\}$ ). On appelle *automorphismes* du code les permutations de l'ensemble  $I$  qui transforment tout mot de code en un mot de code. Les automorphismes du code forment un sous-groupe du groupe des permutations de  $I$ .

**EXERCICE 7.5. [A]** — Quel est le groupe d'automorphismes du code de parité ? du code de répétition pure ?

**EXERCICE 7.6. [B]** — Quel est le groupe d'automorphismes du code orthogonal d'un code ?

**EXERCICE 7.7. [B]** — Quelle relation y a-t-il entre automorphismes d'un code et automorphismes de son extension paire ?

## Un résultat sur la distance minimale

Rappelons qu'un groupe  $G$  de permutations d'un ensemble  $I$  est dit *transitif* si pour tout couple d'éléments  $i$  et  $j$  de  $I$ , il existe  $g \in G$  tel que  $g(i) = j$ . Nous utiliserons à plusieurs reprises le lemme suivant.

**LEMME 7.22.** — *Soient  $C \subset \mathbf{F}_2^n$  un code binaire et  $\bar{C} \subset \mathbf{F}_2^{n+1}$  son extension paire. Supposons que le groupe des automorphismes de  $\bar{C}$  soit transitif. Alors la distance minimale de  $C$  est impaire.*

*Démonstration.* Soit  $\mathbf{m}$  un mot du code  $C$  de poids minimal. Il s'agit de prouver que son poids  $w(\mathbf{m})$  est impair. Supposons qu'il soit pair ; alors  $(\mathbf{m}, 0)$  est un mot de  $\bar{C}$ . Soit  $i$  un indice tel que  $m_i \neq 0$ . Il existe par hypothèse un automorphisme  $g$  de  $\bar{C}$  qui envoie l'indice  $i$  sur l'indice  $n$  du bit de parité. Alors  $g$  transforme  $(\mathbf{m}, 0)$  en un mot de la forme  $(\mathbf{n}, 1)$ . Mais on a  $w(\mathbf{n}) = w(\mathbf{m}) - 1$ , ce qui contredit le caractère minimal de  $\mathbf{m}$ .  $\square$

**EXERCICE 7.8. [A]** — Il résulte en particulier du lemme que le code  $C$  est impair. Pourquoi était-ce évident ?

**EXERCICE 7.9. [A]** — Sous les mêmes hypothèses, prouver que tous les codes obtenus en « supprimant » un bit de  $\bar{C}$  sont équivalents à  $C$ .

### 7.6.1. Exemple : les codes cycliques

On dit que le code  $C \subset \mathbf{F}_2^n$  est cyclique si pour tout mot  $(m_0, \dots, m_{n-1})$  de  $C$ , le mot  $(m_{n-1}, m_0, \dots, m_{n-2})$  qu'on en déduit par décalage circulaire appartient encore à  $C$ . Les codes cycliques forment une classe particulièrement intéressante que nous étudierons en détail plus tard (voir 10.3).

**EXERCICE 7.10. [A]** — La partie paire d'un code cyclique est cyclique.

**EXERCICE 7.11. [B]** — Le code orthogonal d'un code cyclique est cyclique.

Dire que  $C$  est cyclique, c'est dire que le groupe des automorphismes de  $C$  contient la permutation circulaire  $\sigma = (1, \dots, n-1, 0)$ , et donc aussi ses  $n$  puissances, qui sont toutes les permutations circulaires.

Inversement, soit  $C$  un code de parties d'un ensemble  $I$  à  $n$  éléments. Supposons que le groupe des automorphismes de  $C$  contienne un cycle d'ordre  $n$ , c'est-à-dire un élément  $\sigma$  tel que pour un élément  $\alpha \in I$  (et ce sera alors vrai pour tous), les  $\sigma^i(\alpha)$  pour  $i = 0, \dots, n-1$  soient distincts, donc énumèrent les éléments de  $I$ . Numérotons  $I$  en associant à l'entier  $i$  le point  $\sigma^i(\alpha)$ . Alors le code obtenu est cyclique. On dira alors naturellement que  $C$  est cyclique.

Notons que, dans le cas où  $I = \{0, \dots, n-1\}$ , cette définition est plus générale que la première que nous avons donnée, car on peut être amené à changer l'ordre des éléments, donc à remplacer le code par un code équivalent. Cette ambiguïté n'est pas gênante. Beaucoup de codes ne sont en

effet définis qu'à équivalence près. C'est le cas par exemple des codes de Hamming, dont nous verrons plus tard (11.1.1) qu'ils sont cycliques. Nous avons d'ailleurs construit  $H_3$  dans 6.2.3 de façon qu'il soit cyclique dans l'ordre donné. On peut le vérifier sur le plan de Fano (7.4.2) dont nous avons numéroté les sommets de façon que le décalage modulo 7 transforme droites en droites, donc soit un automorphisme du code.

### 7.6.2. Exemple : les transformations affines

Revenons à notre code de Hamming étendu de longueur 8, formé de parties du cube  $I = \mathbf{F}_2^3$ . Il est défini en termes de géométrie affine. Une permutation de  $I$  qui respecte cette géométrie va nécessairement respecter le code, donc être un automorphisme du code.

Précisons. Une *transformation affine* de  $\mathbf{F}_2^m$  est une transformation de la forme

$$x \mapsto Ax + b$$

où  $A$  est une matrice carrée d'ordre  $m$  à coefficients dans  $\mathbf{F}_2$  qui est inversible et où  $b$  est un vecteur de  $\mathbf{F}_2^m$  (on écrit ici les éléments de  $\mathbf{F}_2^m$  comme des colonnes). Ces transformations forment le *groupe affine*  $\mathbf{GA}_m(\mathbf{F}_2)$ . Celles de ces transformations pour lesquelles  $b = 0$ , c'est-à-dire qui laissent fixe l'origine, forment le *groupe linéaire*  $\mathbf{GL}_m(\mathbf{F}_2)$ .

Toute transformation affine de  $\mathbf{F}_2^3$  permute entre eux les plans affines, donc permute entre eux les mots du code de Hamming étendu  $\bar{H}_3$ , donc est un automorphisme de ce code. Celles qui laissent fixe l'origine induisent dans son complémentaire des automorphismes de  $H_3$ .

**EXERCICE 7.12. [B]** — Il y a exactement 168 transformations linéaires et 1344 transformations affines de  $\mathbf{F}_2^3$ .

Inversement, tout automorphisme du code doit permuter entre eux les plans affines de  $\mathbf{F}_2^3$ . On peut prouver que cela implique que c'est une transformation affine. Ainsi le groupe des automorphismes du code de Hamming étendu de longueur 8 est exactement le groupe  $\mathbf{GA}_3(\mathbf{F}_2)$  des transformations affines de  $\mathbf{F}_2^3$ . Le groupe des automorphismes de  $H_3$  est exactement le stabilisateur de l'origine, donc le groupe linéaire  $\mathbf{GL}_3(\mathbf{F}_2)$ .

**REMARQUE 7.23.** — Le fait que  $H_3$  soit cyclique s'interprète ainsi donc en termes de groupes : le groupe linéaire  $\mathbf{GL}_3(\mathbf{F}_2)$ , opérant dans le complémentaire de l'origine dans  $\mathbf{F}_2^3$  qui à 7 éléments, contient un cycle d'ordre 7.

Cela se généralise sans difficultés. Le groupe des automorphismes d'un code de Reed-Muller  $R_{r,m}$  contient par construction le groupe  $\mathbf{GA}_m(\mathbf{F}_2)$ . On peut prouver qu'il est égal à  $\mathbf{GA}_m(\mathbf{F}_2)$  pour  $1 \leq r \leq m - 2$ . Par dualité, on voit que le groupe d'automorphismes du code de Hamming étendu  $\bar{H}_m$  est

**GA<sub>m</sub>(F<sub>2</sub>)** pour  $m \geq 3$ . De même, pour  $m \geq 3$ , le groupe d'automorphisme du code simplexe S<sub>m</sub> ou du code de Hamming H<sub>m</sub> est **GL<sub>m</sub>(F<sub>2</sub>)**.

EXERCICE 7.13. [A] — Que se passe-t-il pour les autres valeurs de  $r$  ou de  $m$  ?

### 7.6.3. Codes et géométries finies

En 1872, dans ce qu'on appelle le *programme d'Erlangen*, le mathématicien allemand Felix KLEIN a défini la géométrie comme l'étude des propriétés invariantes par un groupe de transformations. À chaque géométrie correspond ainsi son groupe : à la géométrie affine le groupe affine, à la géométrie vectorielle le groupe linéaire, etc.

En dehors de définitions explicites, par énumération de ses mots (ou explicitation d'une base dans le cas linéaire), la méthode naturelle pour définir un code de parties d'un ensemble fini I est de le faire en termes d'une structure combinatoire connue sur l'ensemble I. Les automorphismes de cette structure donneront alors des automorphismes du code, comme on a pu le voir ci-dessus.

Inversement d'ailleurs, le détour par un code peut être un moyen commode pour définir un groupe d'automorphismes ou une structure combinatoire. C'est le cas par exemple pour le code binaire de Golay G<sub>24</sub> que nous étudierons plus tard, qui permet de construire à moindres frais un système de Steiner de type (5, 8, 24), et aussi le groupe de Mathieu M<sub>24</sub> comme groupe d'automorphismes de G<sub>24</sub>.



# Chapitre 8

## Majorations de la taille des codes

Comme nous l'avons dit en introduction, chaque code présente un certain compromis entre taux d'information et capacité de correction.

Il est intéressant de savoir quelles sont les contraintes théoriques en la matière. De façon précise, quels sont les triplets  $(n, k, d)$  que l'on peut obtenir comme paramètres d'un code binaire linéaire ? On notera qu'un triplet  $(n, k, d)$  donné ne peut être obtenu que pour un nombre *fini* de codes, tout simplement parce qu'il n'existe qu'un nombre fini de codes de longueur  $n$  donnée : ce sont en effet des parties de l'ensemble fini  $\mathbf{F}_2^n$ .

Cette question se pose dans deux cadres différents. On veut d'un côté des réponses « explicites » lorsque  $n$  est petit ; cela amène à la notion de code *optimal* (voir 8.1.1). Mais on s'intéresse aussi à la situation « asymptotique » pour  $n$  grand et c'est plutôt en termes des deux rapports  $k/n$  et  $d/n$  que l'on veut exprimer les résultats.

### § 8.1. Conditions nécessaires

#### 8.1.1. Majorations, codes optimaux

La première direction que l'on peut prendre est celle des conditions nécessaires pour l'existence d'un code de paramètres  $(n, k, d)$ . La première remarque est alors la suivante. S'il existe un code de paramètres  $(n, k, d)$ , alors il en existe aussi un de paramètres  $(n + 1, k, d)$ . Il suffit en effet de munir chaque mot du code initial d'un bit supplémentaire égal à 0. Par conséquent, si on note  $N(k, d)$  la dimension minimale d'un code binaire linéaire de dimension  $k > 0$  et de distance minimale  $d > 0$ , l'existence d'un code de paramètres  $(n, k, d)$  équivaut à l'inégalité

$$n \geq N(k, d).$$

Nous dirons qu'un code binaire linéaire est *optimal*<sup>1</sup> si ses paramètres satisfont à l'égalité  $n = N(k, d)$ , autrement dit s'il n'existe pas de codes de paramètres  $(m, k, d)$  avec  $m < n$ .

---

1. Il y a plusieurs notions d'optimalité possibles. Selon la définition habituelle, C est optimal s'il n'existe pas de code de paramètres  $(n, k', d)$  avec  $k' > k$ . La notion donnée ici est plus forte, voir l'exercice 8.3.

Dans ce qui suit, nous établirons diverses inégalités de la forme  $n \geq f(k, d)$ , que l'on a coutume d'appeler des *majorations*, car elles permettent de majorer  $k$  connaissant  $n$  et  $d$ . Dire qu'une majoration  $n \geq f(k, d)$  est valable pour tous les codes signifie qu'on a  $N(k, d) \geq f(k, d)$ . En particulier, si un code satisfait à  $n = f(k, d)$ , alors on a  $N(k, d) = f(k, d)$  et le code est optimal.

### 8.1.2. Projections et raccourcissements d'un code

Soit  $C \subset \mathbf{F}_2^n$  un code binaire linéaire de paramètres  $(n, k, d)$ . On suppose  $C$  non nul, de sorte qu'on a  $k \geq 1$  et  $d \geq 1$ . Fixons un entier  $n'$  avec  $0 \leq n' \leq n$  et posons  $n'' = n - n'$ . Tout mot du code  $C$  peut se décomposer en  $\mathbf{m} = (\mathbf{m}', \mathbf{m}'')$  où  $\mathbf{m}'$  est de longueur  $n'$  et  $\mathbf{m}''$  de longueur  $n''$ .

Deux constructions permettent de déduire de  $C$  un code linéaire de longueur  $n'$ . La première est la *projection* : lorsque  $\mathbf{m}$  parcourt  $C$ ,  $\mathbf{m}'$  parcourt un code  $C' \subset \mathbf{F}_2^{n'}$ . En termes d'algèbre linéaire,  $C'$  est la projection de  $C$  parallèlement au sous-espace  $E'' \subset \mathbf{F}_2^n$  formé des mots dont les  $n'$  premiers bits sont nuls. La seconde est le *raccourcissement* : les mots  $\mathbf{m}'$  provenant des mots  $\mathbf{m} \in C$  pour lesquels  $\mathbf{m}'' = 0$  forment l'intersection de  $C$  avec le sous-espace  $E'$  formé des mots dont les  $n''$  derniers bits sont nuls.

On peut évidemment choisir de découper l'ensemble des bits en deux de façon quelconque, ce qui se ramène à la construction précédente par renumérotation. L'interprétation en termes de parties est particulièrement simple : si  $C$  est un code de parties (voir 7.1.1) de l'ensemble fini  $I$ , et si  $J$  est un sous-ensemble de  $I$ , la projection sur  $J$  du code  $C$  est formée des  $A \cap J$  où  $A$  parcourt  $C$ , tandis que le code obtenu par raccourcissement est formé des parties  $A$  du code  $C$  qui sont en fait déjà contenues dans  $J$ .

**EXERCICE 8.1. [A]** — Que forment les mots du code  $C$  dont la projection est vide (nulle) ?

**EXERCICE 8.2. [A]** — On considère un code  $C$  de longueur  $n$  et son extension paire  $\bar{C}$ . Qu'obtient-on par projection de ce dernier sur les  $n$  premiers bits ? Même question pour le raccourcissement.

**PROPOSITION 8.1.** — Soit  $C$  un code linéaire non nul, de paramètres  $(n, k, d)$ .

a) Si l'on a  $k > 1$ , il existe un code, obtenu par raccourcissement de  $C$ , de paramètres  $(n - 1, k - 1, d)$  ;

b) si l'on a  $d > 1$ , il existe un code projeté de  $C$ , de paramètres  $(n - 1, k, d - 1)$  ;

c) il existe un code projeté de  $C$ , de paramètres  $(n - d, k - 1, d')$  avec  $d' \geq d/2$ .

*Démonstration.* Choisissons un mot  $\mathbf{m}$  de  $C$  de poids  $d$ .

Si nous raccourcissons le code  $C$  en supprimant un bit qui ne figure pas dans  $\mathbf{m}$ , alors le mot  $\mathbf{m}$  appartient au code raccourci, qui sera donc aussi de distance minimale  $d$ . Comme on a imposé une condition linéaire, la dimension du code raccourci est  $k$  ou  $k - 1$ . Dire qu'elle vaut  $k$ , c'est dire que tous les mots du code ont le bit en question à 0. Si c'est vrai pour tous les bits qui ne figurent pas dans  $\mathbf{m}$ , cela signifie que  $C$  est réduit à 0 et à  $\mathbf{m}$ , donc que  $k = 1$ . Cela prouve  $a$ .

Projetons le code en supprimant un bit qui figure dans  $\mathbf{m}$ . Alors le code projeté aura un mot de poids  $d - 1$ , donc sera de distance  $d - 1$ . S'il n'est pas de dimension  $k$ , cela signifie que le noyau de la projection n'est pas nul, donc que le mot réduit au bit supprimé appartient à  $C$ , et donc que  $d = 1$ . Cela prouve  $b$ .

Projetons maintenant le code parallèlement aux bits figurant dans  $\mathbf{m}$ , que l'on suppose avoir placé en dernier, de sorte qu'on a  $\mathbf{m} = (\mathbf{m}', \mathbf{m}'')$ , avec  $\mathbf{m}' = 0$ . Le noyau de la projection est formé des mots de code « contenus dans  $\mathbf{m}$  », donc est réduit à 0 et  $\mathbf{m}$ . On a donc obtenu un code  $C'$  de longueur  $n - d$  et de dimension  $k - 1$ . Soit  $\mathbf{n}'$  un mot non nul de  $C'$ . Pour démontrer  $c$ ), il s'agit de prouver que l'on a  $w(\mathbf{n}') \geq d/2$ . Or il existe par construction un mot de  $C$  de la forme  $\mathbf{n} = (\mathbf{n}', \mathbf{n}'')$  et l'on a  $w(\mathbf{n}') + w(\mathbf{n}'') \geq d$ . Mais l'on a aussi  $(\mathbf{n}', \mathbf{n}'' + \mathbf{m}'') = \mathbf{n} + \mathbf{m} \in C$ , ce qui donne  $w(\mathbf{n}') + w(\mathbf{n}'' + \mathbf{m}'') \geq d$ , et en additionnant  $2w(\mathbf{n}') + w(\mathbf{n}'') + w(\mathbf{n}'' + \mathbf{m}'') \geq 2d$ . Or, puisque tous les bits de  $\mathbf{m}''$  sont égaux à 1,  $\mathbf{n}'' + \mathbf{m}''$  est le « complémentaire » de  $\mathbf{n}''$ , ce qui implique  $w(\mathbf{n}'') + w(\mathbf{n}'' + \mathbf{m}'') = d$ . On en déduit  $2w(\mathbf{n}') \geq d$ , d'où  $c$ ).  $\square$

**COROLLAIRE 8.2 (Majoration de Singleton).** — *On a  $n \geq d + k - 1$ .*

*Démonstration.* Si on applique  $a$ ) de façon répétée, on obtient un code de paramètres  $(n - k + 1, 1, d)$ . On a donc  $d \leq n - k + 1$ .  $\square$

On peut traduire la majoration de Singleton par l'inégalité

$$N(k, d) \geq d + k - 1.$$

Si on prend pour  $C$  un code optimal, on a  $n = N(k, d)$  et on déduit aussitôt des parties  $a$ ) et  $b$ ) de la proposition les inégalités

$$N(k, d) > N(k - 1, d), \quad N(k, d) > N(k, d - 1).$$

**EXERCICE 8.3. [A]** — Soit  $C$  un code optimal de paramètres  $(n, k, d)$ . Il n'existe pas de codes de paramètres  $(n, k + 1, d)$  ou  $(n, k, d + 1)$ .

**EXERCICE 8.4. [A]** — S'il existe un code de paramètres  $(n, k, d)$ , alors il existe des codes de paramètres  $(n, k', d')$  pour  $1 \leq k' \leq k$  et  $1 \leq d' \leq d$ .

**EXERCICE 8.5. [B]** — Si  $a$  et  $b$  sont deux entiers  $\geq 0$ , on a  $N(k + a, d + b) \geq N(k, d) + a + b$ .

### 8.1.3. Majoration de Griesmer

**PROPOSITION 8.3 (Majoration de Griesmer).** — *Soit  $C$  un code binaire linéaire non nul, de paramètres  $(n, k, d)$ . On a<sup>2</sup>*

---

2. On note  $\lceil x \rceil$  le plafond (partie entière par excès) du nombre réel  $x$ .

$$n \geq \sum_{0 \leq i < k} \lceil \frac{d}{2^i} \rceil = d + \lceil \frac{d}{2} \rceil + \cdots + \lceil \frac{d}{2^{k-1}} \rceil.$$

*Démonstration.* On a d'abord  $n \geq d$ , puisque  $C$  est non nul. Appliquant la partie *c*) de la proposition 8.1, on déduit de  $C$  un code  $C_1$ , de paramètres  $(n-d, k-1, d_1)$ , avec  $d_1 \geq \lceil d/2 \rceil$ . Si  $k > 1$ ,  $C_1$  est non nul et on a  $n-d \geq d_1$ , donc  $n \geq d+d_1 \geq d+\lceil d/2 \rceil$ . Recommençant avec  $C_1$ , on construit  $C_2$ , de paramètres  $(n-d-d_1, k-2, d_2)$ , avec  $d_2 \geq \lceil d_1/2 \rceil \geq \lceil d/4 \rceil$ , ce qui donne  $n \geq d+d_1+d_2$  si  $k > 2$ . On continue ainsi, jusqu'à un code de paramètres  $(n-d-d_1-\cdots-d_{k-2}, 1, d_{k-1})$ , et on obtient  $n-d-d_1-\cdots-d_{k-2} \geq d_{k-1}$ , d'où la majoration annoncée.  $\square$

EXERCICE 8.6. [A] — En déduire la majoration de Singleton  $n \geq d+k-1$ .

EXERCICE 8.7. [B] — Quels sont les cas où la majoration de Singleton est atteinte ?

Donnons deux exemples. Si  $d = 2^{k-1}$ , on obtient alors  $n \geq 2^k - 1$ . La borne est atteinte pour le code simplexe  $S_k$  de paramètres  $(2^k - 1, k, 2^{k-1})$ . Si  $d = 2^{k-2}$ , on obtient  $n \geq 2^{k-1}$ . La borne est atteinte pour le code de Reed-Muller  $R_{1,k-1}$  de paramètres  $(2^{k-1}, k, 2^{k-2})$ . Ainsi, les codes  $S_m$  et  $R_{1,m}$  sont optimaux.

### 8.1.4. Majorations de Plotkin

De la majoration de Griesmer on déduit :

COROLLAIRE 8.4 (Majorations de Plotkin). — a) On a  $n \geq 2d - \lfloor d/2^{k-1} \rfloor$ .

b) Si  $d \leq 2^{k-1}$ , on a  $n \geq 2d + k - 2 - \lfloor \log d \rfloor$ .

*Démonstration.* Le second membre de la majoration de Griesmer est supérieur à la somme  $d + d/2 + \cdots + d/2^{k-1}$ , qui vaut  $2d - d/2^{k-1}$ . Cela donne a).

Prouvons b). Posons  $a = \lfloor \log d \rfloor \leq k-1$ , de sorte que  $2^a \leq d < 2^{a+1}$ . Les  $k-1-a$  derniers termes de la majoration de Griesmer sont chacun  $\geq 1$ , leur somme est donc  $\geq k-1-a$ . Par ailleurs la somme des  $a+1$  premiers est  $\geq 2d - d/2^a$ , par le même calcul que dans a). Mais puisque  $1 \leq d/2^a < 2$ , cette somme, qui est un entier, est  $\geq 2d - 1$ . En définitive, on obtient

$$n \geq (2d - 1) + (k - 1 - a) = 2d + k - 2 - \lfloor \log d \rfloor.$$

$\square$

EXERCICE 8.8. [B] — Si  $2d > n$ , on a  $2^k \leq 2d/(2d-n)$ . Si  $d$  est impair et  $2d+1 > n$ , on a  $2^k \leq 2(d+1)/(2d+1-n)$ .

### 8.1.5. Majoration de Hamming

Fixons un entier  $n \geq 0$ . Soit  $\mathbf{m} \in \mathbf{F}_2^n$  et soit  $r$  un entier avec  $0 \leq r \leq n$ . La boule de rayon  $r$  centrée en  $\mathbf{m}$  est formée par définition des  $\mathbf{m}' \in \mathbf{F}_2^n$  avec  $w(\mathbf{m}' - \mathbf{m}) \leq r$ .

Le nombre d'éléments d'une boule de  $\mathbf{F}_2^n$  de rayon  $r$  est

$$V(n, r) = \sum_{i=0}^r \binom{n}{i} = 1 + n + \binom{n}{2} + \cdots + \binom{n}{r}.$$

EXERCICE 8.9. [A] — Pourquoi ?

Soit  $C \subset \mathbf{F}_2^n$  un code de longueur  $n$ , à  $N$  éléments. Dire que  $C$  est  $t$ -correcteur, c'est dire par définition que les  $N$  boules de rayon  $t$  centrées au divers points du code sont disjointes, ce qui implique qu'on a  $N \cdot V(n, t) \leq 2^n$ . Rappelons que  $C$  est dit *parfait* (définition 6.7) si ces boules forment une partition de l'espace total  $\mathbf{F}_2^n$ . Par conséquent :

**PROPOSITION 8.5** (Majoration de Hamming). — *Soit  $C$  un code binaire  $t$ -correcteur de longueur  $n$ . Posons  $N = \#C$ .*

a) *On a  $V(n, t) \leq 2^n / N$ .*

b) *Pour qu'on ait l'égalité, il faut et il suffit que  $C$  soit  $t$ -correcteur parfait.*

Pour  $t = 0$ , on a  $V(n, t) = 1$  et cela donne  $N \leq 2^n$ , ce qui n'est guère surprenant. Pour  $t = 1$ , on a  $V(n, t) = 1 + n$  et on retrouve la proposition 6.8. Si  $C$  est linéaire, de dimension  $k$ , on a  $N = 2^k$ , et on obtient :

**COROLLAIRE 8.6.** — *Soit  $C$  un code binaire linéaire de paramètres  $(n, k, d)$ . Posons  $t = \lfloor (d - 1)/2 \rfloor$ , de sorte que  $C$  est  $t$ -correcteur. On a  $V(n, t) \leq 2^{n-k}$ , et l'égalité caractérise les codes parfaits.*

## § 8.2. Conditions de Gilbert-Varshamov

Nous allons voir maintenant une condition suffisante d'existence d'un code de paramètres donnés. Nous en donnons deux variantes.

**PROPOSITION 8.7.** — *Soient  $n, d$  et  $N$  des entiers avec  $1 \leq d \leq n$ ,  $N > 0$  et  $V(n, d - 1) \leq 2^n / N$ . Il existe des codes de longueur  $n$ , de distance minimale  $\geq d$ , à  $N$  éléments.*

*Démonstration.* Construisons un tel code en choisissant ses éléments un par un. Supposons donc disposer déjà de  $m$  éléments de  $\mathbf{F}_2^n$ , dont les distances mutuelles sont  $\geq d$ . Pour trouver un élément supplémentaire, il faut le prendre hors des  $m$  boules de rayon  $d - 1$  centrées en les éléments déjà choisis. Or, la réunion de ces boules interdites a au plus  $m \cdot V(n, d - 1)$  éléments, ce qui est  $< 2^n$  tant qu'on a  $m < N$ .  $\square$

On peut rassembler les deux propositions 8.5 et 8.7 de la façon suivante.

**COROLLAIRE 8.8.** — Soient  $n$  et  $t$  deux entiers, avec  $t \geq 0$  et  $2t + 1 \leq n$ . Soit  $C(n, t)$  le nombre maximum d'éléments d'un code binaire  $t$ -correcteur de longueur  $n$ . On a alors

$$\frac{2^n}{V(n, 2t)} \leq C(n, t) \leq \frac{2^n}{V(n, t)}.$$

**EXERCICE 8.10. [A]** — Que dit cet énoncé pour  $t = 0$  ?

Une variante de 8.7 concerne l'existence de codes linéaires :

**PROPOSITION 8.9 (Gilbert-Varshamov).** — Soient  $n, d$  et  $k$  des entiers avec  $1 \leq k \leq n$  et  $2 \leq d \leq n$ . Si  $V(n - 1, d - 2) < 2^{n-k}$ , il existe des codes linéaires de longueur  $n$ , de distance minimale  $\geq d$  et de dimension  $k$ .

*Démonstration.* Posons  $r = n - k$ , de sorte que la condition de l'énoncé s'écrit

$$(*) \quad 1 + \binom{n-1}{1} + \cdots + \binom{n-1}{d-2} < 2^r.$$

Rappelons la construction donnée en 6.4.3. Considérons une matrice binaire  $P$  à  $r$  lignes et  $n$  colonnes, et soit  $\Gamma \subset \mathbb{F}_2^n$  le code linéaire qui est le *noyau* de cette matrice de vérification. Il est de dimension  $\geq n - r = k$ . Supposons qu'aucun ensemble de  $m$  colonnes de  $P$ , avec  $0 < m < d$ , n'est de somme nulle. Alors la distance minimale de  $\Gamma$  est  $\geq d$  (proposition 6.16). Si on prend un sous-espace  $C$  de  $\Gamma$ , de dimension  $k$ , sa distance minimale sera au moins égale à celle de  $\Gamma$ , donc au moins égale à  $d$ .

Il nous suffit donc de prouver qu'on peut construire une matrice  $P$ , à  $r$  lignes et  $n$  colonnes, satisfaisant à la condition énoncée précédemment, c'est-à-dire qu'on peut trouver  $n$  vecteurs de  $\mathbb{F}_2^r$  (les colonnes de la matrice) tels qu'aucune somme de  $m$  d'entre eux, avec  $0 < m < d$ , ne soit nulle.

Nous allons raisonner par récurrence sur  $n$ . Puisque le premier membre de  $(*)$  croît avec  $n$ , l'hypothèse est vérifiée pour  $n - 1$ , et on peut supposer avoir construit une famille de  $n - 1$  vecteurs ayant la propriété voulue. Il s'agit de trouver le  $n$ -ième. La condition que doit satisfaire ce vecteur est d'être différent des sommes de toutes les familles d'au plus  $d - 2$  vecteurs pris parmi les  $n - 1$  donnés. Cela est sûrement possible si le nombre des parties ayant au plus  $d - 2$  éléments d'un ensemble à  $n - 1$  éléments est strictement inférieur au nombre  $2^r$  de tous les vecteurs disponibles. Or c'est exactement ce qu'affirme  $(*)$ .  $\square$

### § 8.3. Formes asymptotiques

Nous allons maintenant nous intéresser aux formes *asymptotiques* des majorations précédentes, que nous allons exprimer en terme des rapports  $k/n$  (taux d'information) et  $d/n$  (capacité de correction).

#### 8.3.1. Le diagramme ( $d/n, k/n$ )

Il est commode de représenter graphiquement la situation. À chaque code linéaire binaire, de paramètres  $(n, k, d)$ , associons le point de coordonnées

$(x = d/n, y = k/n)$ , qui se trouve dans le carré formé des points  $(x, y)$  avec  $0 \leq x \leq 1$  et  $0 \leq y \leq 1$ . Chaque point  $(x, y)$  de ce carré, dont les coordonnées sont rationnelles, représente donc tous les codes dont les paramètres peuvent s'écrire  $(n, ny, nx)$ .

Chacune des majorations obtenues précédemment s'interprète comme une condition liant les coordonnées du point et la longueur des codes que ce point peut représenter. Regardons d'abord les majorations élémentaires.

La majoration de Singleton  $d + k \leq n + 1$  peut s'écrire  $x + y \leq 1 + 1/n$  et montre déjà qu'un point situé hors du triangle de sommets  $O = (0, 0)$ ,  $A = (0, 1)$  et  $B = (1, 0)$  ne peut représenter qu'un nombre fini de codes. En fait, seuls les codes triviaux figurent hors de ce triangle (exercice 8.7). La majoration de Plotkin (corollaire 8.4, b)) donne un énoncé plus précis :

**PROPOSITION 8.10.** — *Pour tout nombre réel  $\varepsilon$ , avec  $\varepsilon > 0$ , il n'existe qu'un nombre fini de codes représentés dans la région  $2x + y \geq 1 + \varepsilon$ ,  $y \geq \varepsilon$ .*

*Démonstration.* Considérons un code de paramètres  $(n, k, d)$  avec  $k \geq \varepsilon n$  et  $2d + k \geq n + \varepsilon n$ . Il s'agit de montrer que  $n$  est borné. Puisque  $d \leq n$ , on a nécessairement  $d \leq 2^{\varepsilon n - 1} \leq 2^{k-1}$  dès que  $n$  est assez grand. Appliquons alors la majoration b) de Plotkin : on a  $n \geq 2d + k - 2 - \lfloor \log d \rfloor$ . Comme on a  $2d + k \geq n + \varepsilon n$  par hypothèse, on en tire l'inégalité  $2 + \lfloor \log d \rfloor \geq \varepsilon n$  et a fortiori  $2 + \log n \geq \varepsilon n$ . Mais cela est impossible dès que  $n$  est assez grand.  $\square$

### 8.3.2. Le domaine des codes

Nous allons nous intéresser aux *points d'accumulation* des points représentatifs des codes. Ce sont par définition les points  $(x, y)$  pour lesquels on peut trouver une suite infinie de codes  $(C_i)$  dont les paramètres  $(n_i, k_i, d_i)$  sont tels que

$$\lim d_i/n_i = x, \quad \lim k_i/n_i = y.$$

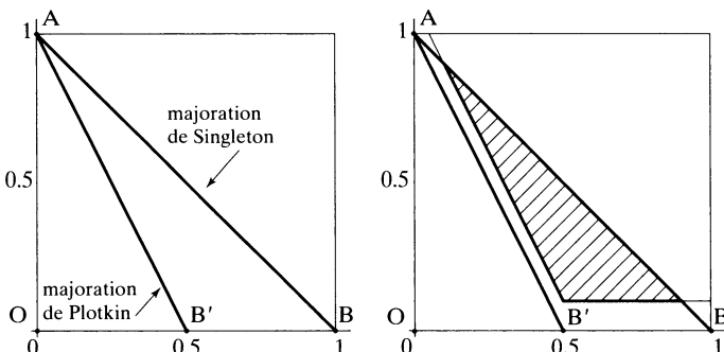
Ce sera le cas en particulier si le point  $(x, y)$  représente une infinité de codes. Remarquons encore une fois qu'il n'y a qu'un nombre fini de codes de longueur donnée, donc que  $n_i$  augmente indéfiniment.

On appelle *domaine des codes* la partie du carré unité formée de ces points d'accumulation. C'est une partie *fermée* comme l'est tout ensemble de points d'accumulation. La proposition précédente montre qu'elle est contenue dans la réunion du triangle

$$x \geq 0, \quad y \geq 0, \quad 2x + y \leq 1,$$

de sommets  $O$ ,  $A$  et  $B' = (1/2, 0)$ , et du segment  $\{1/2 \leq x \leq 1, y = 0\}$  joignant  $B'$  à  $B$ .

**EXERCICE 8.11. [A]** — Montrer que les points  $A$  et  $B$  appartiennent au domaine des codes.



À gauche, points représentatifs des codes (cas limite  $n = \infty$ ). À droite, cas de  $n$  fini.

**EXERCICE 8.12. [B]** — Montrer que si  $(x, y)$  appartient au domaine des codes, il en est de même pour tous les points  $(x', y')$  avec  $0 \leq x' \leq x$  et  $0 \leq y' \leq y$ .

**EXERCICE 8.13. [B]** — Déduire des exercices précédents que le domaine des codes est décrit par une relation de la forme  $0 \leq y \leq f(x)$ , où  $f$  est une fonction décroissante, avec  $f(0) = 1$  et  $f(x) = 0$  pour  $1/2 \leq x \leq 1$ .

**REMARQUE 8.11.** — Il existe de nombreuses majorations asymptotiques plus fortes, remplaçant le segment de droite frontière  $AB'$  du triangle par un segment de courbe joignant  $A$  et  $B'$  à l'intérieur du triangle. On peut par exemple donner une forme asymptotique de la majoration de Hamming.

Nous allons maintenant nous intéresser aux majorations « en sens inverse », c'est-à-dire à des conditions garantissant l'existence de points représentatifs de codes dans certaines zones.

### 8.3.3. Préliminaires

On convient dans la suite que  $x^x = 1$  pour  $x = 0$ . Rappelons qu'on désigne par  $V(n, r)$  le nombre d'éléments d'une boule de rayon  $r$  de  $\mathbb{F}_2^n$ .

**LEMME 8.12.** — Soient  $n$  et  $r$  deux entiers avec  $0 \leq r \leq n/2$ . On a

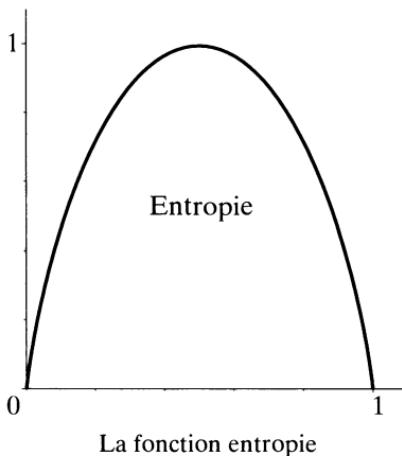
$$V(n, r) \leq \frac{n^n}{r^r (n-r)^{n-r}}.$$

*Démonstration.* Le cas  $r = 0$  est trivial, et on peut supposer  $r > 0$ . Fixons  $x$  avec  $0 < x \leq 1$ . On a successivement

$$(1+x)^n = \sum_{i=0}^n \binom{n}{i} x^i \geq \sum_{i=0}^r \binom{n}{i} x^i \geq x^r \sum_{i=0}^r \binom{n}{i} = x^r V(n, r),$$

soit  $V(n, r) \leq (1+x)^n x^{-r}$ .

Cela est vrai pour tout  $x$  dans l'intervalle considéré. Prenant  $x = r/(n-r)$ , qui est bien compris entre 0 (strictement) et 1, on a  $(1+x)^n x^{-r} = (\frac{n}{n-r})^n (\frac{r}{n-r})^{-r}$ , d'où le lemme.  $\square$



Dans ce qui suit, nous utilisons la fonction réelle  $H$ , dite *entropie*, définie sur l'intervalle  $[0, 1]$  par

$$H(x) = -x \log(x) - (1-x) \log(1-x).$$

Cette fonction est croissante entre 0 et  $1/2$ , décroissante entre  $1/2$  et 1. On a  $H(1-x) = H(x)$ ,  $H(0) = 0$ ,  $H(1/2) = 1$ ,  $H(1) = 0$ ,  $H'(0) = +\infty$ ,  $H'(1/2) = 0$ ,  $H'(1) = -\infty$ .

**PROPOSITION 8.13.** — Soient  $n$  et  $r$  deux entiers avec  $0 \leq r \leq n/2$ . On a

$$V(n, r) \leq 2^{nH(r/n)}.$$

*Démonstration.* Un calcul immédiat donne

$$\begin{aligned} nH(r/n) &= -r \log(r/n) - (n-r) \log((n-r)/n) \\ &= -r \log(r) - (n-r) \log(n-r) + n \log(n), \end{aligned}$$

et on applique le lemme 8.12.  $\square$

**COROLLAIRE 8.14.** — Soient  $n$ ,  $k$  et  $d$  des entiers avec  $0 \leq k \leq n$ ,  $2 \leq d \leq n/2$  et

$$\frac{k}{n} < 1 - \frac{n-1}{n} H\left(\frac{d-2}{n-1}\right).$$

Il existe un code binaire linéaire de longueur  $n$ , de dimension  $k$  et de distance minimale  $\geq d$ .

*Démonstration.* Nous allons utiliser la proposition 8.9. D'après la majoration que l'on vient de voir, on a  $V(n-1, d-2) \leq 2^{(n-1)H(\frac{d-2}{n-1})}$ , de sorte que la condition de la proposition est certainement vérifiée si l'on a  $(n-1)H(\frac{d-2}{n-1}) < n-k$ . Le corollaire en résulte immédiatement.  $\square$

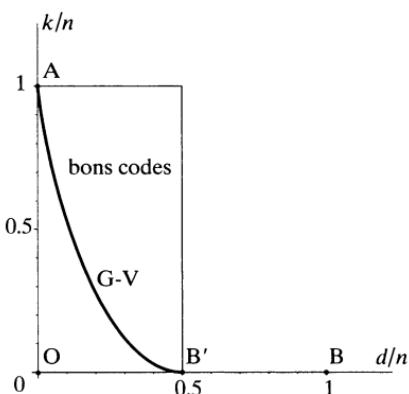
### 8.3.4. Existence de bons codes

On utilise le plus souvent le corollaire précédent sous une forme un peu affaiblie, qui a l'avantage de ne faire intervenir que les rapports  $k/n$  et  $d/n$ . Pour l'exprimer commodément, introduisons une définition :

**DÉFINITION 8.15.** — *On dit que le code linéaire binaire de longueur  $n$ , de dimension  $k$  et de distance minimale  $d$  est bon si l'on a  $d/n < 1/2$  et*

$$\frac{k}{n} > 1 - H\left(\frac{d}{n}\right).$$

Dans le plan  $(x, y)$  introduit plus haut, les bons codes sont donc ceux qui sont représentés par des points du rectangle  $0 \leq x \leq 1/2, 0 \leq y \leq 1$ , de sommets O, A, B' et A + B', qui sont situés « à droite » du segment de la courbe d'équation  $y + H(x) = 1$  contenu dans la bande  $0 \leq x \leq 1/2$ , donc qui joint A à B' et que nous appellerons *courbe de Gilbert-Varshamov*.



**THÉORÈME 8.16.** — *Soit  $k_0$  et  $n_0$  des entiers avec  $0 < k_0 < n_0$ . Pour tout entier  $a > 0$  assez grand, il existe un bon code de longueur  $an_0$  et de dimension  $ak_0$ .*

*Démonstration.* Posons  $y_0 = k_0/n_0$ . On a  $0 < y_0 < 1$  et il existe un unique  $\delta$  avec  $0 < \delta < 1/2$  et  $y_0 = 1 - H(\delta)$ . Soit  $a$  un entier  $> 0$ , posons  $k = ak_0$  et  $n = an_0$ , de sorte que  $k/n = y_0 = 1 - H(\delta)$ . Soit  $d$  l'unique entier tel que  $n\delta < d \leq n\delta + 1$ . Si  $a$  est assez grand pour qu'on ait  $\delta + 1/n < 1/2$ , on a  $d/n < 1/2$ , ce que nous supposerons désormais. On a par construction  $n\delta < d < n\delta + 2 - \delta$ , ce qui s'écrit

aussi  $\frac{d-2}{n-1} < \delta < \frac{d}{n}$ , ou encore, puisque la fonction  $H$  est strictement croissante sur l'intervalle  $[0, 1/2]$ ,  $H(\frac{d-2}{n-1}) < H(\delta) < H(\frac{d}{n})$ , soit enfin

$$(*) \quad 1 - H\left(\frac{d}{n}\right) < y_0 = \frac{k}{n} < 1 - H\left(\frac{d-2}{n-1}\right).$$

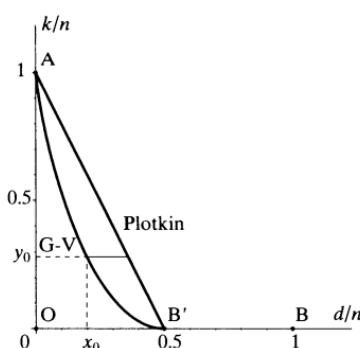
La seconde inégalité de  $(*)$  implique a fortiori  $\frac{k}{n} < 1 - \frac{n-1}{n} H(\frac{d-2}{n-1})$ . Appliquant le corollaire précédent, on en déduit qu'il existe un code linéaire binaire  $C$  de longueur  $n$ , de dimension  $k$ , et de distance minimale  $d' \geq d$ . Le point représentatif de  $C$  est de la forme  $(x, y_0)$ , avec  $x = d'/n > d/n$ . D'après la proposition 8.10, il n'existe qu'un nombre fini de codes avec  $2x + y \geq 1 + y_0$ . Il s'ensuit qu'on a  $x < 1/2$  dès que  $n$  est assez grand. Comme on a alors  $1 - H(x) \leq 1 - H(\frac{d}{n})$  et aussi  $1 - H(\frac{d}{n}) < y_0$  d'après la première inégalité de  $(*)$ , le code  $C$  est bon.  $\square$

Traduisons géométriquement :

**COROLLAIRE 8.17.** — Pour tout nombre rationnel  $y_0$  avec  $0 < y_0 < 1$ , il existe une infinité de codes dont les points représentatifs sont de la forme  $(x, y_0)$  avec  $x < 1/2$  et  $H(x) > 1 - y_0$ .

Considérons l'horizontale d'ordonnée rationnelle  $y = y_0$ . La courbe de Gilbert-Varshamov et la verticale  $x = 1/2$  découpent sur cette horizontale un segment  $x_0 \leq x \leq 1$ , avec  $H(x_0) = 1 - y_0$ . Ce segment contient une infinité de points représentatifs et par conséquent des points d'accumulation. Il existe donc un point du domaine des codes de la forme  $(x, y_0)$ , avec  $H(x) \geq 1 - y_0$ . Vu l'exercice 8.12, cela implique que le point  $(x_0, y_0)$  et tous les points  $(x, y_0)$  avec  $0 \leq x \leq x_0$  appartiennent au domaine des codes. Et comme le domaine des codes est fermé, on en conclut :

**COROLLAIRE 8.18.** — Le domaine des codes contient le triangle curviligne limité par les segments OA et OB' et la courbe de Gilbert-Varshamov.



La majoration de Plotkin (proposition 8.10) montre le domaine des codes ne dépasse pas le segment  $AB'$ . Comme nous l'avons déjà dit, on peut remplacer le segment  $AB'$  par diverses courbes plus proches de la courbe (GV) de Gilbert-Varshamov. Mais en fait, on conjecture que la condition asymptotique de Gilbert-Varshamov ne peut être améliorée, autrement dit *qu'il n'y a aucun point d'accumulation « à droite » de cette courbe*, ou encore que le domaine des codes est décrit exactement par le corollaire 8.18 précédent<sup>3</sup>.

### 8.3.5. Bonnes familles de codes

Bien que les démonstrations des propositions 8.7 et 8.9 prouvent l'existence de bons codes soient d'une certaine façon « explicites » car elles consistent en des algorithmes de construction de tels codes, les résultats précédents ne sont malheureusement pas vraiment constructifs.

La plupart des codes utilisés proviennent de constructions générales qui permettent de décrire explicitement des familles infinies de codes. Nous avons vu les codes de Reed-Muller (RM), nous verrons aussi les codes de Bose-Chaudhuri-Hocquenghem (BCH), les codes de Reed-Solomon (RS)...

Il serait très intéressant de savoir construire explicitement des familles de bons codes, mais c'est en fait beaucoup trop optimiste. Considérons une famille infinie de codes binaires linéaires. En affaiblissant notamment les exigences précédentes, on dira que cette famille est *bonne* si les points d'accumulation des points représentant ces codes ne se trouvent pas tous sur les axes du diagramme, c'est-à-dire s'il existe une constante  $\varepsilon > 0$  telle que la famille contienne une infinité de codes dont les paramètres  $(n, k, d)$  soient tels que  $k/n \geq \varepsilon$  et  $d/n \geq \varepsilon$ .

À l'inverse, une famille de codes binaires linéaires est dite *mauvaise* si, pour tout  $\varepsilon > 0$ , il n'existe qu'un nombre fini de codes de la famille dont les paramètres  $(n, k, d)$  sont tels que  $k/n \geq \varepsilon$  et  $d/n \geq \varepsilon$ . Eh bien, malheureusement, la majorité des familles de codes de définition simple, notamment celles que nous avons citées ci-dessus, sont mauvaises. La première construction d'une bonne famille, dérivée des codes RS, a été donnée en 1972 par Jorn JUSTENSEN<sup>4</sup>.

---

3. Ce fait, s'il est vrai, est particulier aux codes binaires. Dans le cas général, que nous attaquerons dans le chapitre 10, il existe un énoncé tout à fait analogue, mais on peut construire des familles de codes qui s'accumulent au-delà de l'analogue de la courbe de Gilbert-Varshamov.

4. Voir le chapitre 10, §11 de [MacWilliams, Sloane].

## § 8.4. Et Shannon dans tout cela ?

Claude T. SHANNON est universellement connu comme fondateur de la théorie mathématique de la communication, dans deux articles célèbres parus au *Bell Systems Technical Journal* en 1948. Il y formulait un énoncé, justement fameux, prouvant l'existence de codes optimaux en un sens très fort. Mais cet approche probabiliste et non constructive n'a pas permis pendant très longtemps<sup>5</sup> de construire de tels codes.

L'approche algébrique introduite peu après (1949/50) par Marcel GOLAY et Richard HAMMING<sup>6</sup>, et que nous suivons, peut ainsi être considérée par les spécialistes de la communication comme un détour historique devant ce blocage. En témoigne cet extrait de [MacKay] :

Many coding theorists take as their fundamental problem the task of packing as many spheres as possible, with radius as large as possible, into an  $N$ -dimensional space, with no spheres overlapping. Prizes are awarded to people who find packings that squeeze in an extra few spheres. While this is a fascinating mathematical topic, we shall see that the aim of maximizing the distance between codewords in a code has only a tenuous relationship to Shannon's aim of reliable communication.

Inutile sans doute de préciser que les pages que vous avez entre les mains sont consacrées à ce *fascinating mathematical topic*. Nous en dirons un peu plus en conclusion de ce chapitre.

### 8.4.1. L'approche probabiliste

Revenons à la situation initiale décrite en 6.1.2. On code des messages de  $k$  bits par les mots d'un code linéaire  $C$  de longueur  $n$  (et de dimension  $k$ ). On transmet ces mots de  $n$  bits à travers un canal binaire symétrique sans mémoire, dont la probabilité d'erreur est  $p < 1/2$  : chaque bit a de façon indépendante des autres une probabilité  $p$  d'être transformé dans le bit opposé. Un mot de code inconnu  $\mathbf{m}$  est transmis et on reçoit le mot  $\mathbf{m}'$ , entaché de l'erreur  $\mathbf{e} = \mathbf{m}' - \mathbf{m}$ , également inconnue. On cherche à reconstruire  $\mathbf{m}$ . On suppose que tous les mots de code ont une probabilité égale (soit  $2^{-k}$ ) d'avoir été transmis.

Le cadre où nous nous sommes situés jusqu'à présent, et qui nous a permis *d'algébriser* totalement la situation, c'est de prendre un entier  $t > 0$  tel que  $C$  soit  $t$ -correcteur (et évidemment le meilleur possible tant qu'on reste sur le plan théorique) et d'utiliser la dichotomie suivante : si  $\mathbf{m}'$  est à une distance

5. En fait jusqu'à l'invention en 1993 des *turbo-codes* par Claude BERROU, Alain GLAVIEUX et Punya THITIMAJSHIMA, qui a ouvert la voie à une nouvelle génération de codes, coeurs des systèmes GSM et CDMA.

6. Golay, Hamming, Shannon, un beau trio des *Bell Labs* !

$\leq t$  d'un mot  $\mathbf{m}$  de  $C$ , alors on produit  $\mathbf{m}$  comme mot corrigé ; sinon, on sait que l'erreur est de poids  $> t$  et on ne produit pas de  $\mathbf{m}$ .

On notera au passage que même dans le premier cas on n'est pas sûr que la réponse soit correcte. Mais ce qu'on sait de toute façon, c'est qu'on produit une réponse correcte exactement dans les cas où l'erreur est de poids  $\leq t$ . Autrement dit, la probabilité d'une réponse erronée (ou d'une absence de réponse) est égale à la probabilité que l'erreur soit de poids  $> t$ , qui est, comme nous l'avons vu en 6.1.3 :

$$(*) \quad P(w(\mathbf{e}) > t) = \sum_{i=t+1}^n \binom{n}{i} p^i (1-p)^{n-i}.$$

### 8.4.2. Le décodage total

Par opposition à la stratégie précédente, le *décodage total* consiste à appliquer vraiment (totalement !) le principe du maximum de vraisemblance. Pour chaque mot  $\mathbf{m}'$  reçu, on produira le mot de code  $\mathbf{m}$  le plus proche (ou l'un des plus proches s'il y a plusieurs mots à distance minimale de  $\mathbf{m}'$ ), autrement dit de la forme  $\mathbf{m}' - \mathbf{e}$  avec  $w(\mathbf{e})$  minimal. Le décodage total produira la même réponse que la méthode précédente lorsque celle-ci donnait une réponse, puisqu'elle donnait alors nécessairement l'unique mot de code le plus proche, les boules étant supposées disjointes. Ainsi, l'algorithme total aura une probabilité d'erreur plus faible que ce que donne la formule (\*) ci-dessus.

Une manière commode de présenter cet algorithme utilise la notion de syndrome introduite en 6.4.5. Décrivons donc  $C$  comme le noyau d'une application linéaire surjective

$$\pi : \mathbf{F}_2^n \longrightarrow \mathbf{F}_2^{n-k},$$

$\pi(\mathbf{m}')$  étant par définition le syndrome de  $\mathbf{m}'$ . On cherche un mot  $\mathbf{e}$  tel que  $\pi(\mathbf{e}) = \pi(\mathbf{m}')$  et que  $w(\mathbf{e})$  soit minimal pour cette propriété. Ainsi, définissons une application

$$\sigma : \mathbf{F}_2^{n-k} \longrightarrow \mathbf{F}_2^n,$$

en prenant pour  $\sigma(y)$  un élément de poids minimal de  $\pi^{-1}(y)$ , pour tout syndrome  $y \in \mathbf{F}_2^{n-k}$ . On a par construction  $\pi(\sigma(y)) = y$  pour tout  $y$  (ce qu'on traduit en disant que  $\sigma$  est une « section » de la projection  $\pi$ ). Inversement, la condition  $\mathbf{n} = \sigma(\pi(\mathbf{n}))$  caractérise l'image de la section, les mots de la forme  $\sigma(y)$ .

EXERCICE 8.14. [B] — Si  $C$  est  $t$ -correcteur et si  $w(\mathbf{n}) \leq t$ , on a  $\mathbf{n} = \sigma(\pi(\mathbf{n}))$ .

On définit alors l'application de décodage total par

$$\mathbf{m}' \mapsto \mathbf{m}' - \sigma(\pi(\mathbf{m}')).$$

**LEMME 8.19.** — Soit  $\mathbf{e}$  l'erreur réelle de transmission d'un mot  $\mathbf{m}$ . Pour que l'application de décodage total fournit la valeur correcte de  $\mathbf{m}$ , il faut et il suffit qu'il existe  $y \in \mathbf{F}_2^{n-k}$  tel que  $\mathbf{e} = \sigma(y)$ .

*Démonstration.* En effet, un tel  $y$  est nécessairement égal à  $\pi(\sigma(y)) = \pi(\mathbf{e}) = \pi(\mathbf{m}')$ .  $\square$

On appelle *probabilité d'erreur*<sup>7</sup> du code (relativement à la probabilité d'erreur  $p$  du canal) et on note  $P_{\text{err}}(C, p)$  la probabilité que l'application de décodage total donne un résultat erroné. On déduit donc du lemme précédent :

**PROPOSITION 8.20.** — *On a*

$$P_{\text{err}}(C, p) = 1 - \sum_{\mathbf{e}} p^{w(\mathbf{e})}(1-p)^{n-w(\mathbf{e})},$$

où  $\mathbf{e}$  parcourt l'image de l'application  $\sigma$ .

### 8.4.3. Le théorème de Shannon et sa réciproque

On appelle *capacité* du canal le nombre

$$K(p) = 1 - H(p) = 1 + p \log p + (1-p) \log(1-p).$$

On a  $K(0) = 1$ ,  $K(1/2) = 0$  et la fonction  $K$  est décroissante sur l'intervalle  $[0, 1/2]$ .

Le théorème ci-dessous montre que le nombre  $K(p)$  partage l'intervalle  $[0, 1]$  des taux  $k/n$  possibles en deux parties donnant des comportements totalement opposés lorsque  $n$  augmente indéfiniment.

**THÉORÈME 8.21.** — Soient  $\varepsilon$  et  $R$  deux nombres réels, avec  $\varepsilon > 0$  et  $0 \leq R \leq 1$ .

a) Supposons  $R < K(p)$ . Il existe un entier  $N$  tel que, pour tout  $n > N$ , il existe un code binaire linéaire  $C$  de longueur  $n$  et de dimension  $k$ , avec

$$k/n \geq R, \quad P_{\text{err}}(C, p) \leq \varepsilon.$$

b) Supposons  $R > K(p)$ . Il existe un entier  $N$  tel que, pour tout  $n > N$  et tout code binaire linéaire  $C$  de longueur  $n$  et de dimension  $k$  avec  $k/n \geq R$ , on ait  $P_{\text{err}}(C, p) \geq 1 - \varepsilon$ .

**REMARQUE 8.22.** — La partie a) est le cas particulier, appliqué aux canaux binaires symétriques, du théorème de Shannon proprement dit qui se place dans un cadre beaucoup plus général. De même pour la partie b) et la « réciproque forte » de ce

---

7. L'application de décodage total étant basée sur le principe du maximum de vraisemblance, on peut montrer que tout autre mécanisme de décodage donnera une probabilité au moins égale. La dénomination choisie n'est donc pas usurpée.

théorème<sup>8</sup>. Comme nous le verrons un peu plus loin (8.4.5), la méthode de Shannon consiste à évaluer la moyenne de  $P_{\text{err}}(C, p)$  lorsque  $C$  parcourt tous les codes de taux voisins de  $R$  et à montrer que cette moyenne tend vers 0 lorsque  $n$  augmente indéfiniment. Cela entraîne non seulement qu'il existe des codes satisfaisant à la condition voulue, mais que, « asymptotiquement », un code choisi au hasard conviendra.

#### 8.4.4. Quel rapport avec ce qui précède ?

On peut (doit) être quelque peu surpris du fait que les deux théorèmes d'existence de « bons » codes, le théorème 8.16 et la partie *a*) du théorème précédent ont de nombreuses similitudes, mais aussi des différences majeures.

Ce sont en effet deux énoncés asymptotiques portant sur le taux  $k/n$  que peut avoir un « bon » code et la frontière est dans les deux cas de la forme  $k/n = 1 - H(x)$ , pour la même fonction entropie  $H$ , mais les valeurs du  $x$  en question semblent n'avoir aucun rapport. Essayons de comprendre. Partons d'un code  $C$ , de taux  $k/n$ . Soit  $x$  l'unique élément de  $\llbracket 0, 1/2 \rrbracket$  tel que  $k/n = 1 - H(x)$ .

D'un côté, Gilbert et Varshamov nous disent que si  $n$  est grand et si  $C$  est bien choisi, il sera (presque<sup>9</sup>) de distance  $d$  avec  $k/n = 1 - H(d/n)$ , donc  $d/n = x$ . Il sera alors  $t$ -correcteur, avec  $t = d/2$  ou  $t = (d-1)/2$ , soit essentiellement  $t = nx/2$ .

De l'autre, Shannon nous dit que si  $n$  est grand et si  $C$  est bien choisi<sup>10</sup>, il pourra supporter la transmission par un canal de probabilité d'erreur  $p$  avec (presque<sup>11</sup>)  $k/n = 1 - H(p)$ , donc  $p = x$  et qu'il permettra ainsi de corriger environ  $np = nx$  bits erronés.

Comparant, on tombe sur la surprise  $np = 2t$ . On a maintenant effectivement trouvé un rapport entre les deux énoncés, mais on est encore plus troublé, car ils semblent se contredire : suivant la façon de voir le problème, on peut corriger  $t$  ou  $2t$  erreurs. En fait, l'explication va se révéler simple et se comprendra à la lumière de la démonstration du théorème de Shannon.

#### 8.4.5. Une idée de la démonstration

Fixons d'abord  $k$  et  $n$ . Considérons un code  $C$  de longueur  $n$  à  $2^k$  éléments. Prenons un entier  $r$  avec  $0 \leq r < n/2$  et regardons les boules de Hamming centrées aux points de  $C$  et de rayon  $r$ . L'algorithme de décodage total a évidemment la propriété suivante : si le mot reçu  $\mathbf{m}'$  appartient à une et une seule de ces boules,

---

8. Le théorème direct, énoncé par Shannon en 1948, a été démontré rigoureusement sous sa forme générale par Amiel FEINSTEIN en 1954. La réciproque forte est due à Jacob WOLFOWITZ (1957).

9. Rappelons qu'on conjecture qu'asymptotiquement, la majoration de G-V ne peut être améliorée.

10. En fait, choisi essentiellement au hasard !

11. L'égalité est exclue dans le théorème, mais il s'agit ici d'un raisonnement heuristique.

alors il donnera comme réponse le centre de la boule et cette réponse sera exacte. Il ne peut donc faire une erreur que dans deux cas : ou bien lorsque  $\mathbf{m}'$  n'appartient à aucune boule (autrement dit lorsque l'erreur est de poids  $> r$ ), ou bien lorsque  $\mathbf{m}'$  appartient à au moins deux boules.

On va donc chercher un  $r$  tel que ces deux possibilités aient des probabilités aussi petites que possible, ce qui va contraindre  $r$  dans deux sens opposés. Remarquons au passage que la vision « algébrique » consiste à interdire le second cas ; comme nous allons le voir dans un moment, c'est dans cette différence que gît l'explication du facteur 2 qui nous a surpris.

Le premier cas d'erreur est facile à maîtriser. En effet, comme nous l'avons vu ci-dessus, le poids de l'erreur suit une loi binomiale. La loi des grands nombres montre alors que pour  $n$  grand, il suffit de prendre  $r$  suffisamment supérieur à la moyenne  $np$ , en fait d'assurer une inégalité de la forme

$$(i) \quad r/n \geq p + \alpha,$$

avec  $\alpha > 0$ , pour que cette première probabilité tende vers 0 (et d'ailleurs exponentiellement) lorsque  $n$  augmente indéfiniment.

Le second cas est plus intéressant, puisque c'est lui qui fait la différence. Il s'agit de voir dans quel cas l'erreur  $\mathbf{e}$  sur le mot  $\mathbf{m}$ , supposée déjà appartenir à la boule de centre  $\mathbf{m}$ , appartient en plus à une autre. On va maintenant raisonner *en moyenne*,  $C$  parcourant tous les codes de longueur  $n$  et de dimension  $k$ . Sans entrer dans tous les détails, lorsqu'on fait la moyenne pour tous les  $\mathbf{e}$  et tous les  $C$ , cela revient à estimer la probabilité  $\beta$  pour qu'un mot de  $\mathbf{F}_2^n$  donné au hasard appartienne à une boule. Or le volume total occupé par ces boules est majoré par  $2^k V(n, r)$ , de sorte que cette probabilité est majorée par  $2^{k-n} V(n, r)$ . Utilisant maintenant la proposition 8.13, on obtient

$$\beta \leq 2^{k-n+nH(r/n)} = 2^{n(k/n+H(r/n)-1)},$$

de sorte qu'il suffira d'assurer une inégalité de la forme

$$(ii) \quad k/n \leq 1 - H(r/n) - \alpha',$$

avec  $\alpha' > 0$ , pour que  $\beta$  tende vers 0 (exponentiellement elle aussi) lorsque  $n$  augmente indéfiniment.

Comme la fonction  $H(x)$  est strictement croissante sur  $[0, 1/2[$ , la conjonction des conditions (i) et (ii) peut s'écrire :

$$(iii) \quad k/n \leq 1 - H(r/n) - \alpha' \leq 1 - H(p + \alpha) - \alpha'.$$

Mettions-nous maintenant dans les conditions de l'énoncé du théorème de Shannon (théorème 8.21, a)). Fixons donc  $\varepsilon > 0$  et  $R < 1 - H(p)$ . Choisissons  $\rho$  avec  $R < \rho < 1 - H(p)$  et imposons au taux  $k/n$  des codes considérés la contrainte  $R \leq k/n \leq \rho$ . Alors, si on choisit  $\alpha$  et  $\alpha'$  assez petits pour que  $\rho < 1 - H(p + \alpha) - \alpha'$ , la condition  $\rho \leq 1 - H(x) \leq 1 - H(p + \alpha) - \alpha'$  décrira un intervalle de valeurs de  $x$  non réduit à un point. Si  $n$  est assez grand, on pourra alors choisir un  $r$  tel que  $r/n$  appartienne à cet intervalle.

Récapitulons. On a prouvé que la moyenne des  $P_{\text{err}}(C, p)$ , lorsque  $C$  décrit les codes de taux compris entre  $R$  et  $\rho$ , est majorée par une quantité qui tend vers 0

(exponentiellement) lorsque  $n$  augmente indéfiniment. Il existe donc un entier  $N$  tel que cette moyenne soit  $\leq \varepsilon$  pour  $n > N$ . Mais cela implique qu'il existe des  $C$  (et même beaucoup), avec  $\text{Perr}(C, p) \leq \varepsilon$ , ce que nous voulions démontrer.

#### 8.4.6. Moralité

En fait, cette démonstration nous apprend un peu plus. Non seulement elle nous dit qu'il existe des codes  $C$  de taux  $k/n \geq R$  dont la probabilité d'erreur est  $\leq \varepsilon$ , mais elle fournit en même temps un algorithme de décodage, plus simple que le décodage total, dont la probabilité d'erreur est  $\leq \varepsilon$ . Il suffit de choisir une distance  $r$ , tout juste supérieure à l'espérance  $np$  de l'erreur et de produire pour chaque  $\mathbf{m}'$  un mot de code  $\mathbf{m}$ , s'il en existe, avec  $w(\mathbf{m}' - \mathbf{m}) \leq r$  (et n'importe quoi s'il n'en existe pas).

Cela nous donne la clé de la situation paradoxale évoquée en 8.4.4. Plaçons-nous à nouveau dans le cas limite  $k/n = 1 - H(p)$ . L'approche *algébrique* consiste à trouver un code de distance minimale  $d$  la meilleure possible, soit  $d = np$ , et à décoder en utilisant les boules disjointes de rayon  $t = d/2$ , tandis que l'approche *probabiliste* consiste à prendre un code aussi générique que possible et à décoder en utilisant les boules de rayon double  $r = d$  qui ne sont pas disjointes, certes, mais le sont « presque », au sens que le volume relatif des intersections est asymptotiquement négligeable.

Si on en revient à la citation de MacKay donnée en tête de ce paragraphe on comprend bien qu'il a une contradiction assez profonde entre deux orientations : d'un côté trouver des codes assez génériques pour approcher l'optimum de Shannon, ce qui est l'objectif fondamental des professionnels des communications, et de l'autre trouver des codes assez spécifiques pour être des objets mathématiques riches de liens avec d'autres structures intéressantes. Au fond, s'il est vrai « qu'il n'y a de science que du général », on peut dans doute dire en forçant un peu le trait « qu'il n'y a d'algèbre intéressante que du particulier » !

# Chapitre 9

## Les corps finis

Les corps finis sont l'un des outils essentiels de construction de codes. Il y a par exemple un lien très étroit, que nous étudierons dans les chapitres suivants, entre les corps à  $2^n$  éléments et de nombreuses classes de codes binaires, en premier lieu les codes BCH que nous rencontrerons au chapitre 11.

De même, certains codes se décrivent agréablement comme codes de parties d'un corps fini. C'est le cas des codes de résidus quadratiques (chapitre 13).

### § 9.1. Structure des corps finis

Rappelons qu'on appelle *corps* un anneau non nul dans lequel tout élément non nul possède un inverse. Tous les corps que nous considérons sont *commutatifs* par convention<sup>1</sup>.

#### 9.1.1. Éléments algébriques, polynômes minimaux

Soient  $K$  un corps et  $k$  un sous-corps de  $K$ . L'addition dans  $K$  et la multiplication par les éléments de  $k$  munissent  $K$  d'une *structure d'espace vectoriel sur  $k$* .

On dit qu'un élément  $\alpha$  de  $K$  est *algébrique* sur  $k$  si ses puissances ont linéairement dépendantes sur  $k$ , c'est-à-dire s'il existe un polynôme non nul  $Q$  à coefficients dans  $k$  tel que  $Q(\alpha) = 0$ . Sous ces conditions, on a :

**PROPOSITION 9.1.** — Soit  $P \in K[X]$  le polynôme unitaire de plus petit degré tel que  $P(\alpha) = 0$ . Alors les polynômes  $Q \in k[X]$  tels que  $Q(\alpha) = 0$  sont exactement les multiples de  $P$ . De plus,  $P$  est irréductible dans  $k[X]$ .

*Démonstration.* Soit  $Q \in K[X]$  tel que  $Q(\alpha) = 0$ . Écrivons par division euclidienne  $Q = AP + R$ . On a alors  $R(\alpha) = 0$  ce qui, d'après le caractère minimal de  $P$  implique  $R = 0$ .  $\square$

**EXERCICE 9.1. [A] —** Vérifier que  $P$  est irréductible.

1. En anglais, « *field* » signifie « corps commutatif ». Les corps non commutatifs, que l'on appelle aussi en français « corps gauches », sont appelés en anglais « *division algebras* ». En fait, comme nous l'avons déjà dit, les corps finis sont automatiquement commutatifs : c'est le théorème de Wedderburn — voir l'exercice 9.14.

**DÉFINITION 9.2.** — *Le polynôme unitaire  $P$  s'appelle le polynôme minimal de l'élément algébrique  $\alpha$  sur  $k$ . Le degré  $d$  de  $P$  est aussi appelé le degré de  $\alpha$  sur  $k$ .*

Considérons maintenant l'ensemble, noté usuellement  $k(\alpha)$ , formé des éléments de  $K$  qui peuvent s'exprimer sous la forme  $Q(\alpha)$ , où  $Q$  parcourt  $k[X]$ , autrement dit qui peuvent s'exprimer comme combinaison linéaire à coefficients dans  $k$  des puissances de  $\alpha$ .

**PROPOSITION 9.3.** — *a)  $k(\alpha)$  est un sous-corps de  $K$ .*

*b) Les  $d$  premières puissances de  $\alpha$ , à savoir  $1, \alpha, \dots, \alpha^{d-1}$ , forment une base de  $k(\alpha)$  comme espace vectoriel sur  $k$ .*

*Démonstration.* Il est clair que  $k(\alpha)$  est stable par addition et multiplication. Pour prouver *a*), il faut vérifier que si  $Q(\alpha) \neq 0$ , il existe  $U \in k[X]$  avec  $Q(\alpha)U(\alpha) = 1$ . Mais, puisque  $P$  est irréductible et que  $Q$  n'est pas multiple de  $P$ , les polynômes  $P$  et  $Q$  sont premiers entre eux ; par Bézout, il existe alors  $A$  et  $B$  dans  $k[X]$  avec  $AP + BQ = 1$ , et on peut prendre  $U = B$ . Prouvons maintenant *b*). Si  $R \in k[X]$  est le reste de la division euclidienne de  $Q$  par  $P$ , on a  $Q(\alpha) = R(\alpha)$  et  $\deg(R) < d$ . Tout élément de  $k(\alpha)$  est donc combinaison linéaire des éléments proposés. Mais ceux-ci sont linéairement indépendants, car une relation non triviale entre eux donnerait un polynôme non nul à coefficients dans  $k$ , de racine  $\alpha$  et de degré  $< d$ .  $\square$

Ainsi,  $k(\alpha)$  est le plus petit sous-corps de  $K$  qui contienne  $k$  et  $\alpha$  ; c'est un espace vectoriel sur  $k$  de dimension égale au degré  $d$  de  $\alpha$ . On dit que  $\alpha$  est un élément primitif de  $K$  (sous-entendu relativement à  $k$ ) si  $K = k(\alpha)$ .

Par exemple, l'élément  $i$  de  $C$  est algébrique sur  $R$ , de polynôme minimal  $X^2 + 1$ , et c'est un élément primitif de  $C$ . D'ailleurs, tout nombre complexe non réel est un élément primitif de  $C$  relativement à  $R$ .

### 9.1.2. Construction de corps par adjonction

En termes plus abstraits, on peut traduire les résultats précédents en disant que le corps  $k(\alpha)$  s'identifie au quotient  $k[X]/(P)$  de l'anneau de polynômes  $k[X]$  par la congruence modulo le polynôme irréductible  $P$ , donc à l'anneau obtenu par l'adjonction à  $k$  de l'élément  $\alpha$  soumis à la condition  $P(\alpha) = 0$  (voir 3.2.6).

Inversement, si on part d'un corps  $k$  et d'un polynôme irréductible  $P$  de  $k[X]$ , on peut considérer cet anneau quotient  $K = k[X]/(P)$ . De façon concrète, si  $P$  est de degré  $d$ , on peut identifier  $K$  à l'ensemble des polynômes de degré  $< d$  de  $k[X]$ , que l'on ajoute normalement et que l'on multiple modulo  $P$  : le produit de  $A$  par  $B$  est le reste de la division euclidienne par  $P$  du produit usuel  $AB$ .

Les mêmes arguments que ci-dessus montrent alors que  $K$  est un corps contenant  $k$ , et la classe  $\alpha$  de  $X$  modulo  $P$  est par construction un élément primitif de  $K$ , dont le polynôme minimal est le polynôme unitaire associé à  $P$ .

Considérons maintenant deux corps  $K$  et  $K'$  contenant  $k$  et supposons donné un élément primitif  $\alpha$  de  $K$  de polynôme minimal  $P \in k[X]$ . Si  $P$  possède une racine dans  $K'$ , disons  $\alpha'$ , alors le sous-corps  $k(\alpha')$  de  $K'$  s'identifie à  $k[X]/(P)$ , tout comme  $K$ . On peut donc plonger  $K$  comme sous-corps de  $K'$  : de façon plus précise, il existe un unique homomorphisme d'anneaux  $f : K \rightarrow K'$  tel que  $f(x) = x$  pour  $x \in k$  et  $f(\alpha) = \alpha'$ , et cet homomorphisme induit une bijection de  $K$  sur le sous-corps  $k(\alpha')$  de  $K'$  ; on définit  $f$  par  $f(Q(\alpha)) = Q(\alpha')$  pour tout  $Q \in k[X]$ .

### 9.1.3. Corps premiers

Soit  $K$  un corps. Considérons l'élément  $1_K$  du groupe additif  $K$  et son ordre. De deux choses l'une.

Si cet ordre est infini, l'application  $n \mapsto n \cdot 1_K$  est injective. Les  $n \cdot 1_K$  forment un sous-anneau de  $K$  et les  $(n \cdot 1_K)/(m \cdot 1_K)$ , avec  $m \neq 0$  un sous-corps de  $K$ , qui s'identifie naturellement au corps  $\mathbf{Q}$  des nombres rationnels. On dit alors que  $K$  est de caractéristique 0.

Si cet ordre est fini, c'est nécessairement un nombre premier  $p$  et on dit que  $K$  est de caractéristique  $p$ .

EXERCICE 9.2. [A] — Pourquoi  $p$  est-il premier ?

Alors les éléments  $n \cdot 1_K$  forment alors un sous-corps de  $K$ , que l'on peut identifier à  $\mathbf{Z}/p\mathbf{Z}$ .

Dans tous les cas, la relation  $n \cdot 1_k = n' \cdot 1_k$  équivaut à  $n \equiv n' \pmod{p}$ , où  $p$  est la caractéristique de  $K$ .

On appelle *corps premiers* le corps  $\mathbf{Q}$  et les corps  $\mathbf{F}_p = \mathbf{Z}/p\mathbf{Z}$  pour  $p$  premier. On vient ainsi de voir que tout corps  $K$  contient un sous-corps premier, qui est  $\mathbf{Q}$  lorsque la caractéristique de  $K$  est 0, et est  $\mathbf{F}_p$  lorsque la caractéristique de  $K$  est le nombre premier  $p$ . Notons au passage que tout sous-corps de  $K$  est de même caractéristique que  $K$ .

Rappelons, en les complétant, des résultats obtenus précédemment :

**PROPOSITION 9.4.** — *Soit  $K$  un corps de caractéristique  $p \neq 0$ . L'application  $x \mapsto x^p$  est un homomorphisme d'anneaux de  $K$  dans  $K$  : elle est compatible avec l'addition et la multiplication. La condition  $x^p = x$  équivaut à  $x \in \mathbf{F}_p$ . Soit  $Q \in K[X]$  ; pour que les coefficients de  $Q$  appartiennent au sous-corps premier  $\mathbf{F}_p$ , c'est-à-dire pour qu'on ait  $Q \in \mathbf{F}_p[X]$ , il faut et il suffit que  $Q(X^p) = Q(X)^p$ .*

*Démonstration.* Il résulte du corollaire 2.15, c) que  $(x + y)^p = x^p + y^p$  pour  $x$  et  $y$  dans  $K$  ; comme on a aussi  $(xy)^p = x^p y^p$  et  $1^p = 1$ , cela implique la première

assertion. Pour tout  $x \in \mathbf{F}_p$ , on a  $x^p = x$ ; la seconde assertion résulte alors de ce que l'équation  $x^p = x$  a au plus  $p$  solutions. Soit enfin  $Q \in K[X]$ . Si  $Q = \sum a_i X^i$ , on a d'une part  $Q(X^p) = \sum a_i X^{ip}$  et d'autre part  $Q(X)^p = \sum a_i^p X^{ip}$  d'après ce qu'on vient de voir. La condition  $Q(X^p) = Q(X)^p$  équivaut donc au fait que tous les coefficients de  $Q$  sont égaux à leur puissance  $p$ -ième, c'est-à-dire appartiennent à  $\mathbf{F}_p$ .  $\square$

### 9.1.4. Structure des corps finis

**THÉORÈME 9.5.** — *Soit  $K$  un corps fini.*

a) *La caractéristique  $p$  de  $K$  est  $\neq 0$  et le nombre des éléments de  $K$  est de la forme  $p^n$  avec  $n$  entier  $> 0$ .*

b) *On a  $\alpha^{p^n} = \alpha$  pour tout  $\alpha \in K$ ; on a dans l'anneau de polynômes  $K[X]$  la relation  $X^{p^n} - X = \prod_{\alpha \in K} (X - \alpha)$ .*

c) *Le groupe  $K^*$  est cyclique, d'ordre  $p^n - 1$ .*

d) *Le nombre des éléments de tout sous-corps de  $K$  est de la forme  $p^r$  où  $r$  divise  $n$ .*

e) *Inversement, pour tout diviseur  $r$  de  $n$ , il existe un unique sous-corps de  $K$  à  $p^r$  éléments : c'est l'ensemble des  $\alpha \in K$  tels que  $\alpha^{p^r} = \alpha$ .*

*Démonstration.* Puisque  $K$  est fini, son sous-corps premier ne peut être  $\mathbf{Q}$ , et  $p$  est  $\neq 0$ . Alors  $K$  est un espace vectoriel sur  $\mathbf{F}_p$ . Comptant les éléments à l'aide d'une base de cet espace vectoriel, on voit que  $\#K = p^n$ , où  $n$  est la dimension de l'espace. Cela prouve a). La partie b) résulte directement de la proposition 2.11, et la partie c) du théorème 3.8.

Soit  $k$  un sous-corps de  $K$ ; alors  $\#k$  est une puissance  $p^r$  de  $p$  d'après a). Pour prouver que  $r$  divise  $n$ , on peut donner deux démonstrations. La première consiste à remarquer que  $K$  est un espace vectoriel sur  $k$ , ce qui implique comme ci-dessus que  $\#K$  est une puissance de  $\#k$ , donc que  $r$  divise  $n$ . On peut aussi noter que  $k^*$  possède un élément d'ordre  $p^r - 1$ , et que cet ordre doit diviser l'ordre  $p^n - 1$  de  $K^*$ , ce qui implique aussi que  $r$  divise  $n$  (voir l'exercice 9.4 ci-dessous). On a donc prouvé d). Notons de plus que, pour tout  $\alpha \in k$ , on a  $\alpha^{p^r} = \alpha$ ; mais cette équation a au plus  $p^r$  solutions, qui sont donc les éléments de  $k$ . Cela montre que  $k$  est l'ensemble des racines dans  $K$  de l'équation précédente, donc est l'unique sous-corps de  $K$  à  $p^r$  éléments.

Inversement, soit  $r$  un diviseur de  $n$ . Notons  $k$  l'ensemble des  $\alpha \in K$  tels que  $\alpha^{p^r} = \alpha$ . C'est un sous-corps de  $K$  (voir l'exercice 9.3 ci-dessous), et qui a au plus  $p^r$  éléments. Puisque  $K^*$  est cyclique d'ordre  $p^n - 1$  et que  $p^r - 1$  divise  $p^n - 1$ ,  $K^*$  possède un élément d'ordre  $p^r - 1$ , disons  $a$ , ce qui implique que l'équation définissant  $k$  a au moins  $p^r$  solutions (à savoir 0 et les puissances de  $a$ ). Ainsi  $k$  a bien  $p^r$  éléments, ce qui achève la démonstration de e).  $\square$

**EXERCICE 9.3. [A]** — Vérifier que  $k$  est bien un corps.

**EXERCICE 9.4. [A]** — Soient  $p, n$  et  $r$  trois entiers  $> 0$ , avec  $p > 1$ , tels que  $p^r - 1$  divise  $p^n - 1$ . Prouver que  $r$  divise  $n$ .

Mettons en garde contre une erreur extrêmement fréquente : si l'on désigne par  $q$ , comme on le fait traditionnellement, le nombre  $p^n$  des éléments de  $K$ , le noyau de l'application naturelle  $x \mapsto x \cdot 1_K$  de  $\mathbf{Z}$  dans  $K$  est l'ensemble des multiples de  $p$ , et non pas celui des multiples de  $q$ .

Soient  $p$  un nombre premier et  $\mathbf{F}_p$  le corps premier à  $p$  éléments. Si  $P \in \mathbf{F}_p[X]$  est un polynôme irréductible de degré  $r$ , alors  $K = \mathbf{F}_p[X]/(P)$  est un corps à  $q = p^r$  éléments. Inversement, tous les corps finis peuvent s'obtenir par la construction précédente :

**PROPOSITION 9.6.** — *Soit  $K$  un corps fini, et soit  $p$  sa caractéristique. Il existe un élément primitif de  $K$  sur  $\mathbf{F}_p$ .*

*Démonstration.* On a vu que le groupe  $K^*$  est cyclique. Soit  $\alpha$  un générateur de ce groupe. Alors tout élément non nul de  $K$  est une puissance de  $\alpha$ , et tout sous-anneau contenant  $\alpha$  est égal à  $K$ .  $\square$

En d'autres termes, toute racine primitive de l'unité d'ordre  $\#K - 1$  dans  $K$  est un élément primitif de  $K$ .

### 9.1.5. Polynômes minimaux sur $\mathbf{F}_p$

**PROPOSITION 9.7.** — *Soient  $K$  un corps fini,  $\mathbf{F}_p$  son sous-corps premier,  $\alpha$  un élément de  $K$  et  $r$  son degré sur  $\mathbf{F}_p$ . Alors  $r$  est le plus petit entier tel que  $\alpha^{p^r} = \alpha$ , les  $\alpha^{p^i}$ , pour  $i = 0, \dots, r-1$ , sont tous distincts et le polynôme minimal de  $\alpha$  sur  $\mathbf{F}_p$  est  $(X - \alpha)(X - \alpha^p) \cdots (X - \alpha^{p^{r-1}})$ .*

*Démonstration.* Si on note  $p^n$  le nombre des éléments de  $K$ , on a  $\alpha^{p^n} = \alpha$ . Il existe donc un plus petit entier  $s > 0$  tel que  $\alpha^{p^s} = \alpha$ . Alors les  $\alpha^{p^i}$  pour  $0 \leq i < s$  sont tous distincts. Si on avait en effet  $\alpha^{p^i} = \alpha^{p^j}$  avec  $0 \leq i < j < s$ , on en déduirait  $\alpha^{p^{i+s-j}} = (\alpha^{p^i})^{p^{s-j}} = (\alpha^{p^j})^{p^{s-j}} = \alpha^{p^s} = \alpha$ , ce qui contredirait la définition de  $s$ . Cela étant acquis, notons  $P \in \mathbf{F}_p[X]$  le polynôme minimal de  $\alpha$ , qui est de degré  $r$  par hypothèse. D'après la proposition 9.4, on a  $P(X^p) = P(X)^p$ , ce qui implique que l'ensemble des racines de  $P$  dans  $K$  est stable par l'application  $u \mapsto u^p$  et donc que les  $\alpha^{p^i}$  pour  $i > 0$  sont racines de  $P$ . Par conséquent,  $P$  est un multiple dans  $K[X]$  du polynôme  $Q = (X - \alpha)(X - \alpha^p) \cdots (X - \alpha^{p^{s-1}})$ . Comme  $P$  est irréductible dans  $\mathbf{F}_p[X]$ , il suffit maintenant de prouver que  $Q$  est à coefficients dans  $\mathbf{F}_p$ , ce qui impliquera  $P = Q$  et  $r = s$ . Mais on a

$$\begin{aligned} (Q(X))^p &= (X - \alpha)^p (X - \alpha^p)^p \cdots (X - \alpha^{p^{s-1}})^p \\ &= (X^p - \alpha^p)(X^p - \alpha^{p^2}) \cdots (X^p - \alpha) = Q(X^p), \end{aligned}$$

et on applique la proposition 9.4.  $\square$

### 9.1.6. Automorphismes d'un corps fini

Soit  $K$  un corps. Par définition, un *automorphisme* de  $K$  est une application bijective  $f : K \rightarrow K$  qui est un homomorphisme d'anneaux, c'est-à-dire telle que  $f(1) = 1$ ,  $f(x + y) = f(x) + f(y)$  et  $f(xy) = f(x)f(y)$ .

**EXERCICE 9.5. [B]** — La condition  $f(1) = 1$  est conséquence des autres.

**EXERCICE 9.6. [A]** — On a  $f(x - y) = f(x) - f(y)$  et  $f(x^{-1}) = f(x)^{-1}$  pour  $x \neq 0$ .

Les automorphismes de  $K$  forment un groupe pour la composition, l'élément neutre étant l'automorphisme identique  $I : x \mapsto x$ . Par exemple, si  $K$  est fini à  $p^n$  éléments, l'application  $u : x \mapsto x^p$  est un automorphisme, dont l'ordre (dans le groupe des automorphismes de  $K$ ) est  $n$  : on a vu en effet dans le théorème 9.5 que  $u^n = I$  et que  $u^r \neq I$  pour tout diviseur  $r$  de  $n$  distinct de  $n$ .

On notera que si  $f$  et  $g$  sont deux automorphismes de  $K$ , l'ensemble des éléments  $x$  tels que  $f(x) = g(x)$  est un sous-corps de  $K$ . En particulier, l'ensemble des  $x$  tels que  $f(x) = x$  est un sous-corps de  $K$ . Il en résulte que les éléments du sous-corps premier sont laissés fixes par tout automorphisme de  $K$ . De façon équivalente, tout automorphisme d'un corps premier est l'identité.

**EXERCICE 9.7. [C]** — Prouver que tout automorphisme du corps  $\mathbb{R}$  est l'identité.

**PROPOSITION 9.8.** — *Soient  $K$  un corps fini,  $p$  sa caractéristique et  $f : K \rightarrow K$  un automorphisme. Il existe un entier  $s \geq 0$  tel que  $f(x) = x^{p^s}$  pour tout  $x$ . En d'autres termes, si  $\#K = p^n$ , le groupe des automorphismes de  $K$  est cyclique d'ordre  $n$ , engendré par  $x \mapsto x^p$ .*

*Démonstration.* Soit  $\alpha \in K$  un élément primitif sur  $\mathbf{F}_p$  (corollaire 9.6) et soit  $P$  le polynôme minimal de  $\alpha$  sur  $\mathbf{F}_p$ . Les coefficients de  $P$  appartiennent à  $\mathbf{F}_p$  et on a  $f(a) = a$  pour tout  $a \in \mathbf{F}_p$ , ce qui implique  $P(f(\alpha)) = f(P(\alpha)) = f(0) = 0$ . D'après la proposition 9.7, il existe  $s$  avec  $f(\alpha) = \alpha^{p^s}$ . Le sous-corps de  $K$  défini par l'équation  $f(x) = x^{p^s}$  contient  $\alpha$ , donc coïncide avec  $K$ .  $\square$

## § 9.2. Polynômes cyclotomiques

Avant de passer à la construction explicite des corps finis, il nous faut étudier plus précisément la construction des racines de l'unité dans le cas complexe.

### 9.2.1. Le polynôme $\Phi_n$

Dans le corps  $\mathbf{C}$  des nombres complexes, les racines de l'unité sont les  $e^{2\pi i r}$ , avec  $r \in \mathbf{Q}$ . Dire que  $e^{2\pi i r}$  est une racine  $n$ -ième de l'unité signifie que  $nr \in \mathbf{Z}$ ; dire que c'est une racine primitive  $n$ -ième de l'unité signifie que  $n$  est le dénominateur de la fraction  $r$  mise sous forme irréductible. Notons que, pour  $k \in \mathbf{Z}$ ,  $e^{2\pi i k/n}$  ne dépend que de la classe de  $k$  modulo  $n$ ; cela permet de donner un sens à l'expression  $e^{2\pi i k/n}$  avec  $k \in \mathbf{Z}/n\mathbf{Z}$ . On obtient ainsi lorsque  $k$  parcourt  $\mathbf{Z}/n\mathbf{Z}$  les  $n$  racines  $n$ -ièmes de l'unité; les racines primitives correspondent aux éléments  $k$  qui sont inversibles dans  $\mathbf{Z}/n\mathbf{Z}$ , c'est-à-dire aux  $\varphi(n)$  éléments du groupe multiplicatif  $(\mathbf{Z}/n\mathbf{Z})^*$ .

**DÉFINITION 9.9.** — Soit  $n$  un entier  $> 0$ . On appelle  $n$ -ième polynôme cyclotomique<sup>2</sup> et on note  $\Phi_n(X)$  le produit  $\prod_{\zeta} (X - \zeta)$ , où  $\zeta$  parcourt les racines primitives  $n$ -ièmes de l'unité dans  $\mathbf{C}$ .

On a donc

$$\Phi_n(X) = \prod_{k \in (\mathbf{Z}/n\mathbf{Z})^*} (X - e^{2\pi i k/n}),$$

à comparer avec

$$X^n - 1 = \prod_{k \in \mathbf{Z}/n\mathbf{Z}} (X - e^{2\pi i k/n}).$$

**PROPOSITION 9.10.** — a)  $\Phi_n$  est un polynôme unitaire à coefficients entiers, de degré  $\varphi(n)$ .

b) Le polynôme  $X^n - 1$  est le produit des  $\Phi_d(X)$ , pour tous les diviseurs  $d$  de  $n$ .

*Démonstration.* La partie b) est claire : il suffit de regrouper les racines  $n$ -ièmes de l'unité suivant leur ordre. Par ailleurs,  $\Phi_n$  est unitaire et de degré  $\varphi(n)$ . Démontrons qu'il est à coefficients entiers par récurrence sur  $n$ . Si l'on utilise la décomposition donnée dans b), l'hypothèse de récurrence implique que tous les facteurs du produit, sauf justement  $\Phi_n$ , sont à coefficients entiers. On peut donc écrire dans  $\mathbf{C}[X]$  la relation  $X^n - 1 = \Phi_n(X)h(X)$ , où  $h(X)$  est un polynôme unitaire de  $\mathbf{Z}(X)$ . Par division euclidienne dans  $\mathbf{Z}[X]$  (proposition 1.39), on peut écrire  $X^n - 1 = g(X)h(X) + r(X)$ , avec  $g(X)$  et  $r(X)$  à coefficients entiers et  $\deg(r) < \deg(h)$ . Mais l'unicité de la division euclidienne dans  $\mathbf{C}[X]$  donne  $\Phi_n = g \in \mathbf{Z}[X]$ .  $\square$

La partie b) permet de calculer les  $\Phi_n(X)$  par récurrence. On obtient par exemple

---

2. Cyclotomie signifie en grec « division du cercle ».

$$\begin{array}{ll} \Phi_1(X) = X - 1 & \Phi_5(X) = X^4 + X^3 + X^2 + X + 1 \\ \Phi_2(X) = X + 1 & \Phi_6(X) = X^2 - X + 1 \\ \Phi_3(X) = X^2 + X + 1 & \dots \\ \Phi_4(X) = X^2 + 1 & \Phi_{12}(X) = X^4 - X^2 + 1. \end{array}$$

**EXERCICE 9.8.** [A] — Prouver, pour  $n > 1$  et pour tout nombre réel  $x \geq 1$ , l'inégalité  $\Phi_n(x) > (x - 1)^{\varphi(n)}$ .

**EXERCICE 9.9.** [A] — Calculer  $\Phi_p$  pour  $p$  premier.

**EXERCICE 9.10.** [B] — Montrer que  $\Phi_{p^r}(X) = \Phi_p(X^{p^{r-1}})$  pour  $p$  premier et  $r > 0$ .

**EXERCICE 9.11.** [B] — Calculer  $\Phi_{2n}$  en termes de  $\Phi_n$ .

**EXERCICE 9.12.** [B] — D'après la proposition 2.16, il semble que si un nombre premier  $p$  divise  $\Phi_n(a)$ , alors  $p \equiv 1 \pmod{n}$ . Est-ce vrai ?

**EXERCICE 9.13.** [N] — Calculer  $\Phi_n$  pour  $n$  petit, disons  $n < 200$ . Les coefficients de  $\Phi_n$  sont-ils toujours égaux à 0, 1 ou  $-1$  ?

**EXERCICE 9.14.** [C] — Dans cet exercice, on donne une démonstration du théorème de Wedderburn. Soit  $K$  un corps fini (non supposé commutatif). Notons  $Z$  son centre, c'est-à-dire l'ensemble des  $x \in K$  tels que  $xy = yx$  pour tout  $y \in K$ .

- 1) Prouver que  $Z$  est un corps (commutatif).
- 2) Posons  $q = \#Z$ . Prouver que  $\#K$  s'écrit  $q^n$  et que, pour tout sous-corps  $E$  de  $K$  contenant  $Z$ , on a  $\#E = q^r$ , où  $r$  divise  $n$ .
- 3) Pour tout  $x \in K^*$ , notons  $C(x) \subset K$  l'ensemble des  $yxy^{-1}$  où  $y$  parcourt  $K^*$ . Prouver que  $\#C(x)$  est de la forme  $\frac{q^n - 1}{q^r - 1}$ , où  $r$  divise  $n$ .
- 4) En décomposant  $K$  en réunion de parties de la forme  $C(x)$ , prouver que  $\Phi_n(q)$  divise  $q - 1$ .
- 5) En déduire que  $n = 1$  en utilisant l'exercice 9.8.

### 9.2.2. Racines des polynômes cyclotomiques

Puisque  $\Phi_n(X)$  est à coefficients entiers, on peut calculer  $\Phi_n(a)$  pour tout élément  $a$  d'un anneau (commutatif)  $A$ .

**PROPOSITION 9.11.** — Soient  $A$  un anneau intègre et  $n$  un entier  $> 0$  tel que  $n1_A \neq 0$ .

a) Les racines  $n$ -ièmes de l'unité dans  $A$  sont des racines simples du polynôme  $X^n - 1$ . Plus généralement,  $X^n - 1$  est sans facteur multiple<sup>3</sup>.

b) Les racines de  $\Phi_n$  dans  $A$  sont exactement les racines primitives  $n$ -ièmes de l'unité dans  $A$ .

3. Cela signifie par définition qu'il n'existe pas de polynôme non constant  $Q \in A[X]$  tel que  $Q^2$  divise  $X^n - 1$ , voir plus loin 9.6.1.

*Démonstration.* Écrivons  $X^n - 1 = Q^2(X)R(X)$  et dérivons : on obtient

$$nX^{n-1} = Q(X)(2Q'(X)R(X) + Q(X)R'(X)).$$

Par combinaison, on obtient dans  $A[X]$  la relation

$$n\mathbb{1}_A = X(nX^{n-1}) - n(X^n - 1) = Q(X)(2XQ'(X)R(X) + XQ(X)R'(X) - nQ(X)R(X))$$

et  $Q(X)$ , divisant la constante *non nulle*  $n\mathbb{1}_A$ , est constant. Cela prouve *a*). Ainsi, une racine  $n$ -ième de l'unité dans  $A$ , c'est-à-dire une racine du polynôme  $X^n - 1$  est une racine *d'exactement un* des polynômes  $\Phi_d$  pour  $d$  divisant  $n$ . Mais l'ensemble des racines des  $\Phi_d$  pour  $d$  divisant  $n$  et distinct de  $n$  est aussi l'ensemble des racines des  $X^d - 1$  pour  $d$  divisant  $n$  et distinct de  $n$ , c'est-à-dire l'ensemble des racines  $n$ -ièmes de l'unité qui ne sont pas primitives. Cela prouve *b*).  $\square$

**REMARQUE 9.12.** — On peut donner une variante de la proposition précédente, sans hypothèse sur  $A$ , mais utilisant la notion de racine principale introduite en 4.1.1. Les deux conditions suivantes sont en effet équivalentes :

- (i)  $a$  est une racine principale d'ordre  $n$  ;
- (ii) on a  $\Phi_n(a) = 0$ , et  $n \cdot \mathbb{1}_A$  n'est pas diviseur de zéro.

**EXERCICE 9.15.** [C] — Démontrer cette équivalence.

### 9.2.3. Irréductibilité sur $\mathbf{Q}$ des polynômes cyclotomiques

Nous utiliserons ci-dessous les deux lemmes suivants :

**LEMME 9.13.** — Soient  $k$  un corps,  $f$  et  $g$  deux polynômes non nuls de  $k[X]$ ,  $A$  un anneau contenant  $k$ ,  $d'$  élément unité  $1_k$  et  $z$  un élément de  $A$  tel que  $f(z) = g(z) = 0$ . Si  $f$  est irréductible dans  $k[X]$ , il divise  $g$ .

*Démonstration.* Puisque  $f$  est irréductible, il s'agit de prouver que  $f$  et  $g$  ne sont pas premiers entre eux. Or, dans le cas contraire, il existerait  $a$  et  $b$  dans  $k[X]$  avec  $1 = a(X)f(X) + b(X)g(X)$  et, prenant  $X = z$ , on obtiendrait  $1_k = 0$ .  $\square$

**LEMME 9.14 (Lemme de Gauss).** — Soient  $f$  et  $g$  deux polynômes unitaires de  $\mathbf{Q}[X]$  tels que  $fg$  soit à coefficients entiers. Alors  $f$  et  $g$  sont à coefficients entiers.

*Démonstration.* Soit  $d$  le plus petit dénominateur commun des coefficients de  $f$ . Alors les coefficients de  $F = df$  sont entiers. Définissons de même  $e$  et posons  $G = eg$ , de sorte que l'on a dans l'anneau de polynômes  $\mathbf{Z}[X]$  la relation  $FG = de(fg)$ . Il s'agit de prouver que  $d$  et  $e$  sont égaux à 1. Sinon, prenons un nombre premier  $p$  qui divise  $de$  et réduisons la relation précédente modulo  $p$ . Puisque  $p$  divise  $de$ , cela donne dans  $\mathbf{F}_p[X]$  la relation  $\bar{F}\bar{G} = 0$ . Mais, comme  $\mathbf{F}_p$  est un corps, cela implique que l'un des polynômes  $\bar{F}$  ou  $\bar{G}$  est nul. Supposons par exemple que l'on ait  $\bar{F} = 0$ . Alors  $p$  divise tous les coefficients de  $F$ , de sorte que  $(1/p)F = (d/p)f$  est à coefficients entiers. Mais, puisque  $f$  a été supposé unitaire, le coefficient dominant de  $F$  est  $d$ , de sorte que  $p$  divise  $d$  et que  $d/p$  est entier. Ainsi  $d/p$  est un dénominateur commun des coefficients de  $f$ , ce qui contredit le choix de  $d$ .  $\square$

La décomposition des polynômes  $X^n - 1$  en produit de polynômes à coefficients *rationnels* donnée dans la proposition 9.10 ne peut être améliorée. En effet :

**THÉORÈME 9.15 (Gauss).** — *Pour chaque entier  $n > 0$ , le polynôme  $\Phi_n(X)$  est irréductible dans  $\mathbf{Q}[X]$ .*

*Démonstration.* Soit  $f \in \mathbf{Q}[X]$  un facteur irréductible de  $\Phi_n$ , supposé unitaire. Il s'agit de prouver que  $f = \Phi_n$ , ou encore que toute racine primitive  $n$ -ième de l'unité dans  $\mathbf{C}$  est une racine de  $f$ . Si  $\zeta$  et  $\zeta'$  sont deux racines primitives  $n$ -ième de l'unité, on peut écrire  $\zeta' = \zeta^m$ , avec  $m$  premier à  $n$ , donc  $m$  produit de nombres premiers ne divisant pas  $n$ . On voit donc qu'on aura terminé lorsqu'on aura démontré l'assertion suivante :

(\*) si le nombre premier  $p$  ne divise pas  $n$  et si  $f(\zeta) = 0$ , alors  $f(\zeta^p) = 0$ .

Remarquons d'abord que l'on peut écrire  $X^n - 1 = f(X)g(X)$  avec  $g \in \mathbf{Q}[X]$ , ce qui d'après le lemme 9.14 implique  $f \in \mathbf{Z}[X]$  et  $g \in \mathbf{Z}[X]$ . Pour prouver que  $f(\zeta^p) = 0$ , il suffit de montrer que  $g(\zeta^p) \neq 0$ . Or, dans le cas contraire, le polynôme  $g(X^p)$  aurait  $\zeta$  comme racine. Mais cela impliquerait qu'il est multiple de  $f$  (lemme 9.13), et on pourrait écrire  $g(X^p) = f(X)h(X)$ . Comme ci-dessus, on voit que  $h \in \mathbf{Z}[X]$ . Résumons : on dispose d'un entier  $p$  premier à  $n$  et de trois polynômes unitaires  $f, g$  et  $h$  de  $\mathbf{Z}[X]$  avec

$$X^n - 1 = f(X)g(X), \quad g(X^p) = f(X)h(X).$$

Réduisons maintenant tout ceci modulo  $p$ . On obtient dans  $(\mathbf{Z}/p\mathbf{Z})[X]$  des identités analogues, soit

$$X^n - 1 = \bar{f}(X)\bar{g}(X), \quad \bar{g}(X^p) = \bar{f}(X)\bar{h}(X).$$

Mais cette fois-ci, on a  $\bar{g}(X^p) = \bar{g}(X)^p$  (proposition 9.4). Ainsi, tout facteur irréductible  $P$  de  $\bar{f}$  divise  $\bar{g}^p$  donc  $\bar{g}$ ; donc  $P^2$  divise  $X^n - 1$  dans  $(\mathbf{Z}/p\mathbf{Z})[X]$ , ce qui contredit la proposition 9.11, a).  $\square$

#### 9.2.4. Corps cyclotomiques

Fixons un entier  $n > 0$ . On appelle *corps cyclotomique de niveau  $n$*  et on note  $\mathbf{R}_n(\mathbf{C})$ , ou simplement  $\mathbf{R}_n$ , le plus petit sous-corps de  $\mathbf{C}$  qui contienne toutes les racines  $n$ -ièmes de l'unité. Naturellement, ce sous-corps contient le sous-corps premier  $\mathbf{Q}$ , donc est un espace vectoriel sur  $\mathbf{Q}$ .

**PROPOSITION 9.16.** — *Le corps  $\mathbf{R}_n$  est un espace vectoriel de dimension  $\varphi(n)$  sur  $\mathbf{Q}$ . Plus précisément, soit  $\zeta$  une racine primitive  $n$ -ième de l'unité dans  $\mathbf{C}$ . Alors les éléments  $\zeta^i$ , pour  $i = 0, \dots, \varphi(n) - 1$ , forment une base de cet espace vectoriel.*

*Démonstration.* Notons d'abord qu'on a  $\Phi_n(\zeta) = 0$  et que  $\Phi_n$  est irréductible. Le plus petit sous-corps de  $\mathbf{C}$  contenant  $\zeta$  s'identifie donc à  $\mathbf{Q}[X]/(\Phi_n)$ . Comme les

racines  $n$ -ièmes de l'unité sont exactement les puissances de  $\zeta$ , elles appartiennent à ce corps, qui est donc le corps  $R_n$  cherché. La proposition résulte immédiatement de là.  $\square$

**EXERCICE 9.16.** [C] — Soit  $G$  le groupe des automorphismes du corps  $R_n$ . Montrer qu'il existe un homomorphisme de groupes  $u : G \rightarrow (\mathbf{Z}/n\mathbf{Z})^*$ , uniquement déterminé, tel que  $g(\zeta) = \zeta^{u(g)}$  pour toute racine  $n$ -ième de l'unité  $\zeta$  et tout  $g \in G$ , et que  $u$  est bijectif.

### 9.2.5. Décomposition des polynômes cyclotomiques dans un corps fini

D'après ce qu'on a vu, tout corps fini peut se construire en adjoignant à un corps premier  $F_p$  un élément algébrique bien choisi, racine d'un polynôme irréductible  $P \in F_p[X]$ . Mais comme tout élément non nul d'un corps fini, cet élément est nécessairement une racine de l'unité, et donc  $P$  est un facteur irréductible d'un polynôme  $X^n - 1$ . Or, on a vu qu'un tel polynôme se décomposait déjà dans  $\mathbf{Z}[X]$  en produit des polynômes cyclotomiques  $\Phi_d$ , pour  $d$  divisant  $n$ . Par conséquent,  $P$  est un facteur irréductible (dans  $F_p[X]$ ) d'un des polynômes cyclotomiques  $\Phi_d$ . Il existe ainsi une relation directe entre la construction des corps finis de caractéristique  $p$  et la décomposition des polynômes cyclotomiques dans  $F_p[X]$ .

Soient  $p$  un nombre premier et  $n$  un entier  $> 0$ . Bien que le polynôme  $\Phi_n$  soit irréductible sur  $\mathbf{Q}$  comme on vient de le prouver, il n'en n'est pas nécessairement de même lorsqu'on le réduit modulo  $p$ . Par exemple, si  $\Phi_7 = 1 + X + \cdots + X^6$  est bien irréductible sur  $\mathbf{Q}$ , il ne l'est plus modulo 2, puisqu'on a

$$1 + X + \cdots + X^6 \equiv (1 + X + X^3)(1 + X^2 + X^3) \pmod{2}.$$

Nous allons plus généralement déterminer ce qu'il advient lorsque l'on considère un corps fini  $k$  à  $q = p^m$  éléments et le polynôme à coefficients dans  $k$  obtenu en remplaçant chaque coefficient  $a_i$  de  $\Phi_n$  par son image  $a_i 1_k$  dans  $k$ . Cela consiste évidemment à réduire ces coefficients modulo  $p$  (et non pas modulo  $q$  !), mais comme il y a plus d'éléments dans  $k$  que dans  $F_p$ , les facteurs irréductibles de  $\Phi_n$  dans  $F_p[X]$  ne restent pas nécessairement irréductibles dans  $k[X]$ . La proposition suivante donne la réponse complète.

**PROPOSITION 9.17.** — Soit  $k$  un corps fini à  $q$  éléments, et soit  $n > 0$  un entier premier à  $q$ . Notons  $r$  l'ordre de la classe de  $q$  dans le groupe  $(\mathbf{Z}/n\mathbf{Z})^*$ , c'est-à-dire le plus petit entier  $r > 0$  tel que  $q^r \equiv 1 \pmod{n}$ . Alors le polynôme cyclotomique  $\Phi_n(X)$  se décompose dans  $k[X]$  en produit de polynômes unitaires irréductibles de degré  $r$ , tous différents.

*Démonstration.* Nous en donnerons deux démonstrations, chacune mettant en avant des idées importantes.

Le polynôme  $\Phi_n$  n'est pas divisible par le carré d'un polynôme non constant de  $k[X]$  (proposition 9.11) et il suffit de prouver que tout facteur irréductible de  $\Phi_n(X)$  dans  $k[X]$  est de degré  $r$ . Soit  $P$  un tel facteur, soit  $s$  son degré et soit  $K$  le corps  $k[X]/(P)$ .

On a  $\#K = q^s$  et tout élément non nul  $\alpha$  de  $K$  satisfait à  $\alpha^{q^s-1} = 1$ . Le corps  $K$  contient par construction une racine de  $\Phi_n$ , qui est d'après la proposition 9.11 une racine primitive  $n$ -ième de l'unité, soit  $\zeta$ . Puisque  $\zeta^{q^s-1} = 1$ , alors  $n$  divise  $q^s - 1$ , ce qui signifie que  $q^s \equiv 1 \pmod{n}$ . Par conséquent,  $s$  est au moins égal à  $r$  (c'en est en fait un multiple).

Inversement, puisque  $\zeta^n = 1$  et que  $n$  divise  $q^r - 1$ , on a  $\zeta^{q^r} = \zeta$ . Le sous corps de  $K$  formé des racines de l'équation  $X^{q^r} = X$  contient  $\zeta$ , donc est égal à  $K$  tout entier, puisque  $\zeta$  est un élément primitif de  $K$ . On a donc  $q^s = \#K \leq q^r$ , donc  $s \leq r$ . En définitive, on obtient  $r = s$ , ce qu'on voulait démontrer.

Exposons maintenant brièvement une seconde démonstration, où apparaît la notion de classe cyclotomique que nous étudierons plus loin dans 10.4.2. Nous laissons la vérification des détails au lecteur.

Notons  $K$  un corps à  $q^r$  éléments contenant  $k$  (il en existe d'après le théorème 9.26 ci-dessous). Alors, puisque  $n$  divise  $q^r - 1$ ,  $K$  contient un élément  $\alpha$  d'ordre  $n$ . Notons  $I \subset \mathbf{Z}/n\mathbf{Z}$  l'ensemble des classes modulo  $n$  des entiers  $i$  premiers à  $n$ . On a  $\#I = \varphi(n) = \deg \Phi_n$ ; mais chacun des  $\alpha^i$  est aussi d'ordre  $n$  (voir par exemple l'exercice 2.12), donc est une racine de  $\Phi_n$  d'après la proposition 9.11. On a ainsi

$$\Phi_n(X) = \prod_{i \in I} (X - \alpha^i).$$

Considérons maintenant la multiplication par  $q$  dans  $I$ . Pour chaque  $i \in I$ , on a  $q^r i \equiv i$ . Mais, puisque  $i$  est premier à  $n$  et que  $q$  est d'ordre  $r$  modulo  $n$ , les classes de  $i, q_i, \dots, q^{r-1}i$  sont toutes distinctes. On peut ainsi partitionner  $I$  en classes, toutes à  $r$  éléments, stables par multiplication par  $q$ . Cela correspond à une décomposition de  $\Phi_n$  en produit de polynômes, tous de degré  $r$ . Il ne reste plus qu'à prouver, comme dans la démonstration de la proposition 9.7 que ces polynômes sont à coefficients dans  $k$  et irréductibles.  $\square$

Donnons quelques exemples.

Pour  $q = 3$  et  $n = 11$ , on a  $r = 5$ , et le polynôme  $\Phi_{11} = 1 + X + \dots + X^{10}$  se décompose dans le corps  $\mathbf{F}_3$  en produit de deux polynômes irréductibles de degré 5.

Dire que  $r = 1$ , c'est dire que  $\Phi_n$  possède des racines dans  $k$ . Cela signifie que  $k$  contient des racines primitives  $n$ -ièmes de l'unité. La proposition montre que pour qu'il en soit ainsi, il faut et il suffit que  $n$  divise  $q - 1$ . On retrouve ainsi le fait que  $k^*$  est cyclique.

À l'opposé, pour qu'il n'y ait qu'un facteur dans la décomposition irréductible de  $\Phi_n$  sur  $k$ , il faut que  $r = \deg \Phi_n = \varphi(n)$ . Par conséquent :

**COROLLAIRE 9.18.** — *Soit  $k$  un corps fini à  $q$  éléments, et soit  $n$  un entier positif premier à  $q$ . Pour que le polynôme cyclotomique  $\Phi_n$  soit irréductible sur  $k$ , il faut et il suffit que la classe de  $q$  engende le groupe  $(\mathbf{Z}/n\mathbf{Z})^*$ .*

Par exemple, puisque la classe de 2 est un générateur de  $(\mathbf{Z}/3\mathbf{Z})^*$  (resp.  $(\mathbf{Z}/5\mathbf{Z})^*$ ), le polynôme  $\Phi_3$  (resp.  $\Phi_5$ ) est irréductible sur  $\mathbf{F}_2$ .

Si l'on prend maintenant  $k = \mathbf{F}_p$ , où  $p$  est premier, et  $n$  de la forme  $p^m - 1$ , alors on a  $r = m$  et on obtient :

**COROLLAIRE 9.19.** — *Dans l'anneau  $\mathbf{F}_p[X]$ , le polynôme  $\Phi_{p^r-1}$  est un produit de polynômes unitaires irréductibles de degré  $r$ , tous différents.*

On retrouve ainsi pour  $p = 2$  et  $r = 3$  la décomposition

$$1 + X + \cdots + X^6 = (1 + X + X^3)(1 + X^2 + X^3) \in \mathbf{F}_2[X].$$

**EXERCICE 9.17. [B]** — On déduit de cet énoncé que  $\varphi(p^r - 1)$  est divisible par  $r$ . Le prouver directement.

On déduit aussitôt du corollaire précédent que pour tout nombre premier  $p$  et tout entier  $r > 0$ , il existe dans  $\mathbf{F}_p[X]$  des polynômes irréductibles de degré  $r$ . Mais ce corollaire est en fait plus fort. Introduisons une définition commode :

**DÉFINITION 9.20.** — *Un polynôme irréductible  $P \in \mathbf{F}_p[X]$ , de degré  $r$ , est dit primitif s'il divise  $\Phi_{p^r-1}$  ou, de manière équivalente, si ses racines sont des racines primitives ( $p^r - 1$ )-ièmes de l'unité.*

Avec ce vocabulaire :

**PROPOSITION 9.21.** — *Pour tout nombre premier  $p$  et tout entier  $r > 0$ , il existe dans  $\mathbf{F}_p[X]$  des polynômes irréductibles primitifs de degré  $r$ .*

### § 9.3. Construction des corps finis

#### 9.3.1. Polynômes irréductibles sur $\mathbf{F}_p$

On vient de voir que pour tout nombre premier  $p$  et tout entier  $n > 0$ , il existe des polynômes irréductibles de degré  $n$  dans  $\mathbf{F}_p[X]$ . Ce résultat, que nous avons déduit d'une analyse un peu fine des polynômes cyclotomiques, peut aussi s'obtenir par une énumération qui fournira en outre une estimation du nombre de ces polynômes.

Notons donc  $m_r(p)$  le nombre de polynômes unitaires irréductibles de degré  $r$  sur le corps  $\mathbf{F}_p$ . Les polynômes unitaires de degré 1 sont tous irréductibles ; ce sont les  $X - a$  pour  $a \in \mathbf{F}_p$  ; on a donc

$$m_1(p) = p.$$

Il y a  $p^2$  polynômes unitaires de degré 2. Parmi eux, les polynômes réductibles sont les  $(X - a)^2$  et les  $(X - a)(X - b)$  pour  $a \neq b$  ; il y en a  $p(p-1)/2 = p(p+1)/2$ . On a donc

$$m_2(p) = p(p-1)/2.$$

**EXERCICE 9.18. [B]** — On suppose  $p \neq 2$ . Prouver que, pour que le polynôme  $X^2 - sX + t$  soit irréductible, il faut et il suffit que  $\delta = s^2 - 4t$  ne soit pas un carré dans  $\mathbf{F}_p$ . Retrouver ainsi la valeur de  $m_2(p)$ .

**PROPOSITION 9.22.** — Soient  $n$  un entier  $> 0$ . Dans  $\mathbf{F}_p[X]$ , le polynôme  $X^{p^n} - X$  est exactement le produit de tous les polynômes unitaires irréductibles dont le degré divise  $n$ .

*Démonstration.* Soit d'abord  $P$  un polynôme irréductible, de degré  $r$ . Alors dans le corps fini  $K = \mathbf{F}_p[X]/(P)$ , qui a  $p^r$  éléments, la classe  $\alpha$  de  $X$  satisfait à  $\alpha^{p^r} = \alpha$ . Si  $r$  divise  $n$ , on a aussi  $\alpha^{p^n} = \alpha$ , ce qui signifie que  $X^{p^n} - X$  est un multiple de  $P$  (lemme 9.13). Inversement, si  $X^{p^n} - X$  est un multiple de  $P$ , on a  $\alpha^{p^n} = \alpha$ . Mais les  $a \in K$  tels que  $a^{p^n} = a$  forment un sous-anneau de  $K$ . Comme ce sous-anneau contient  $\alpha$ , il est égal à  $K$ . Tout élément non nul  $a$  de  $K$  satisfait donc à  $a^{p^n-1} = 1$ . Or  $K^*$  contient un élément d'ordre  $p^r - 1$  (théorème 3.8), et il en résulte que  $p^r - 1$  divise  $p^n - 1$ , donc que  $r$  divise  $n$  (voir l'exercice 9.4 plus haut).

Ainsi, si on décompose en facteurs irréductibles le polynôme unitaire  $X^{p^n} - X$ , on obtient comme facteurs tous les polynômes considérés et uniquement ceux-là. Il reste à prouver qu'il n'y a pas de facteur multiple. Or, si le polynôme  $X^{p^n} - X$  est divisible par  $P^2$ , son polynôme dérivé est divisible par  $P$ , mais ce polynôme dérivé vaut  $p^n X^{p^n-1} - 1 = -1$ .  $\square$

**COROLLAIRE 9.23.** — On a

$$\frac{p^n - p^{\lfloor n/2 \rfloor + 1}}{n} \leq m_n(p) \leq \frac{p^n}{n}.$$

*Démonstration.* Écrivons pour simplifier  $m_r$  au lieu de  $m_r(p)$ . Le degré  $p^n$  de  $X^{p^n} - X$  est la somme des degrés de tous les polynômes unitaires irréductibles dont le degré divise  $n$ , ce qui s'écrit aussi

$$p^n = \sum r m_r,$$

où la somme est étendue à tous les diviseurs  $r$  de  $n$ . On en tire d'abord la majoration  $m_n \leq p^n/n$ . En appliquant cette majoration aux  $m_r$  pour les diviseurs  $r$  de  $n$  distincts de  $n$ , on en déduit que  $p^n - nm_n$  est majoré par la somme des  $p^r$  pour les  $r$  précédents. À son tour cette somme est majorée par la somme des  $p^r$  étendue à tous les entiers  $r \leq n/2$ , qui vaut  $(p^{\lfloor n/2 \rfloor + 1} - 1)/(p - 1) < p^{\lfloor n/2 \rfloor + 1}$ . D'où la minoration annoncée.  $\square$

La minoration ci-dessus implique notamment que  $m_n(p)$  est  $> 0$  pour tout  $n > 1$  (donc qu'il existe des polynômes irréductibles de degré  $n$ ).

Par ailleurs, on voit que la fonction  $n \mapsto m_n(p)$  est équivalente pour  $n$  grand à  $p^n/n$  : un polynôme unitaire de grand degré  $n$  choisi au hasard a en gros une chance sur  $n$  d'être irréductible.

On peut déduire aussi de la proposition précédente un critère d'irréductibilité :

**COROLLAIRE 9.24.** — Pour qu'un polynôme  $P$  de  $\mathbf{F}_p[X]$  de degré  $n > 0$  soit irréductible, il faut et il suffit qu'il satisfasse aux deux conditions suivantes :

- a)  $P(X)$  divise  $X^{p^n} - X$ ,
- b) pour tout facteur premier  $q$  de  $n$ ,  $P(X)$  est premier à  $X^{p^{n/q}} - X$ .

*Démonstration.* Considérons en effet les degrés des facteurs irréductibles de  $P$ . La condition a) signifie que ce sont tous des diviseurs de  $n$  et la condition b) qu'aucun d'entre eux ne peut en être un diviseur strict.  $\square$

Pour appliquer concrètement ce critère, on se place dans l'anneau quotient  $\mathbf{F}_p[X]/(P(X))$  et on utilise l'algorithme de calcul rapide des puissances (voir 1.2) pour déterminer les puissances de  $X$  modulo  $P(X)$ . Notons par exemple  $Q_i$  le reste ainsi obtenu de la division euclidienne de  $X^{p^i}$  par  $P(X)$ . Alors la condition a) s'écrit  $Q_n = X$  et la condition b) s'écrit  $\text{pgcd}(Q_{n/q} - X, P) = 1$ .

**EXERCICE 9.19. [B]** — On prend  $P(X) = X^p - X - a$  avec  $a \in \mathbf{F}_p^*$ . Prouver que  $P$  est irréductible.

**REMARQUE 9.25.** — Les énoncés de cette section restent valables, avec les mêmes démonstrations, lorsqu'on y remplace  $\mathbf{F}_p$  par un corps fini  $K$  quelconque et  $p$  par  $q = \#K$ .

### 9.3.2. Relation entre ordre et degré

Synthétisons ce qui a été obtenu précédemment. Soient  $K$  un corps fini,  $p$  la caractéristique de  $K$  et  $\alpha$  un élément non nul de  $K$ . Alors  $\alpha$  est une racine de l'unité et possède à la fois un *degré*  $r$  sur  $\mathbf{F}_p$  et un *ordre*  $n$  dans le groupe  $K^*$ .

Le *degré*  $r$  est défini par l'une quelconque des propriétés suivantes :

- a)  $\alpha$  est racine d'un polynôme irréductible  $P \in \mathbf{F}_p$  de degré  $r$  ;
- b) le corps  $\mathbf{F}_p(\alpha)$  a  $p^r$  éléments ;
- c)  $r$  est le plus petit entier  $> 0$  tel que  $\alpha^{p^r} = \alpha$ .

L'*ordre*  $n$  est défini par l'une quelconque des propriétés suivantes :

- a)  $n$  est le plus petit entier  $> 1$  tel que  $\alpha^n = 1$  ;
- b)  $n$  est premier à  $p$  et on a  $\Phi_n(\alpha) = 0$ .

Si  $r$  est donné, alors  $n$  est un diviseur de  $p^r - 1$ . Si  $n$  est donné, alors  $r$  est le plus petit entier tel que  $p^r \equiv 1 \pmod{n}$ .

#### Exemple : les corps à 16 éléments

Pour mieux comprendre, détaillons à titre d'exemple le cas d'un corps  $K$  à  $2^4 = 16$  éléments.

On a d'abord la décomposition en polynômes cyclotomiques, valable dans tout anneau de base

$$\begin{aligned} X^{16} - X &= X \Phi_1(X) \Phi_3(X) \Phi_5(X) \Phi_{15}(X) \\ &= X(X-1)(X^2 + X + 1)(X^4 + X^3 + X^2 + X + 1) \times \\ &\quad \times (X^8 - X^7 + X^5 - X^4 + X^3 - X + 1), \end{aligned}$$

dont les facteurs sont irréductibles dans  $\mathbf{Z}[X]$ .

Si on réduit modulo 2, les quatre premiers facteurs restent irréductibles dans  $\mathbf{F}_2[X]$ . Le dernier se décompose en produit de deux polynômes irréductibles (primitifs) de degré 4, conformément au corollaire 9.19 : on a modulo 2

$$X^8 + X^7 + X^5 + X^4 + X^3 + X + 1 \equiv (X^4 + X^3 + 1)(X^4 + X + 1).$$

Dans  $\mathbf{F}_2[X]$ , les polynômes irréductibles<sup>4</sup> de degré divisant 4 sont donc au nombre de six :

- ▷ les deux polynômes de degré 1, à savoir  $X$  et  $X + 1$ ,
- ▷ un polynôme de degré 2, à savoir  $\Phi_3(X) = X^2 + X + 1$ ,
- ▷ le polynôme cyclotomique  $\Phi_5(X)$ , de degré 4,
- ▷ les deux facteurs de  $\Phi_{15}(X)$ , qui sont primitifs et de degré 4.

Quant aux éléments de  $K$ , chacun d'eux est racine d'un des polynômes de cette liste. On obtient la liste correspondante de ces seize éléments :

- ▷ les deux éléments 0 et 1 du sous-corps premier,
- ▷ les deux racines de  $\Phi_3$ , qui sont les deux autres éléments du sous-corps à quatre éléments, et sont de degré 2 et d'ordre 3,
- ▷ les quatre racines de  $\Phi_5(X)$ , de degré 4 et d'ordre 5
- ▷ les huit racines de  $\Phi_{15}(X)$ , de degré 4 et d'ordre 15.

### 9.3.3. « Le » corps à $q$ éléments

**THÉORÈME 9.26.** — *a) Il existe pour tout nombre premier  $p$  et tout entier  $n > 0$  un corps à  $p^n$  éléments.*

*b) Deux corps finis qui ont le même nombre d'éléments sont isomorphes.*

*c) Plus généralement, si  $K$  est un corps fini à  $p^n$  éléments et  $K'$  un corps fini à  $p^m$  éléments, où  $n$  divise  $m$ , alors  $K$  se plonge comme un sous-corps de  $K'$ .*

*Démonstration.* On a vu qu'il existe au moins un polynôme irréductible de degré  $n$  dans  $\mathbf{F}_p[X]$ . Alors  $\mathbf{F}_p[X]/(P)$  est un corps à  $p^n$  éléments, ce qui prouve *a*). La partie *b*) est le cas particulier  $n = m$  de *c*), que nous démontrons maintenant.

D'après ce qu'on a vu, on peut décrire  $K$  à l'aide d'un élément primitif  $\alpha$  annulant un polynôme unitaire irréductible  $P \in \mathbf{F}_p[X]$ , de degré  $n$ . D'après la proposition 9.22, le polynôme  $P$  divise  $X^{p^m} - X$  dans  $\mathbf{F}_p[X]$ . D'un autre côté, on a dans  $K'[X]$  la relation  $X^{p^m} - X = \prod_{a \in K'} (X - a)$  (proposition 2.11). Il en résulte que le polynôme  $P$  qui divise dans  $K'[X]$  le produit précédent, s'y décompose en produit de facteurs

---

4. Noter que sur  $\mathbf{F}_2$ , tous les polynômes non nuls sont unitaires !

linéaires, donc possède des racines dans  $K'$ . Ainsi, comme on l'a vu précédemment (9.1.2),  $K$  se plonge dans  $K'$ .  $\square$

### La notation $F_q$ et ses dangers

Soit  $q$  une puissance d'un nombre premier. Comme on vient de voir, il n'y a essentiellement qu'un corps fini à  $q$  éléments. On s'autorise parfois de cela pour noter  $F_q$  un corps indéterminé à  $q$  éléments. Cette notation est dangereuse pour la raison que nous allons expliquer. Dans le cas d'un nombre premier  $p$ , si  $K$  et  $K'$  sont deux corps à  $p$  éléments, il n'y a aucun danger à écrire  $K = K' = F_p$ , car il n'y a qu'une façon de faire se correspondre les éléments de  $K$  et les éléments de  $K'$  : à  $1_K$  correspond  $1_{K'}$ , à  $2_K$  correspond  $2_{K'}$ , ... Cela n'est pas vrai lorsque le nombre des éléments n'est plus premier.

Prenons comme exemple celui des corps à 4 éléments. Soit  $K$  un tel corps. Il contient le sous-corps premier  $F_2$ , dont les éléments sont 0 et 1, et deux autres éléments, disons  $\alpha$  et  $\beta$ . On a  $\alpha + \alpha = 2\alpha = 0$  et de même  $\beta + \beta = 0$ . Puisque  $\alpha + 1$  ne peut être ni 0, ni 1, ni  $\alpha$ , on a forcément  $\alpha + 1 = \beta$ , donc  $\beta + 1 = \alpha$ . De même, on a forcément  $\alpha^2 = \beta$ ,  $\beta^2 = \alpha$  et  $\alpha\beta = 1$ , et on a déterminé totalement les tables d'addition et de multiplication.

**EXERCICE 9.20. [A]** — Les trois anneaux  $\mathbf{Z}/4\mathbf{Z}$ ,  $(\mathbf{Z}/2\mathbf{Z}) \times (\mathbf{Z}/2\mathbf{Z})$  et  $K$  sont deux à deux non isomorphes.

**EXERCICE 9.21. [C]** — Y a-t-il d'autres anneaux à quatre éléments que les trois ci-dessus (bien entendu, on raisonne à isomorphisme près) ?

Cela étant, il n'y a aucun moyen de distinguer l'un de l'autre les éléments  $\alpha$  et  $\beta$ . Si  $K'$  est un autre corps à 4 éléments, et si on note  $\gamma$  et  $\delta$  ses deux éléments distincts de 0 et 1, il y a deux façons distinctes d'identifier  $K$  et  $K'$ . On peut décider d'égaler  $K$  à  $K'$  de façon que  $\alpha = \gamma$  et  $\beta = \delta$ , ou que  $\alpha = \delta$  et  $\beta = \gamma$ .

**REMARQUE 9.27.** — On notera d'ailleurs que ce problème se pose déjà dans le cas du corps  $\mathbf{C}$ , où l'on ne peut algébriquement distinguer  $i$  de  $-i$ .

Cette situation se reproduit dans tous les cas : si l'on a deux corps  $K$  et  $K'$  de même nombre d'éléments  $q = p^n$  et si l'on a identifié  $K$  à  $K'$  par l'application  $f : K \rightarrow K'$ , on peut aussi bien le faire par l'application  $x \mapsto f(x)^p = f(x^p)$  par exemple. C'est l'existence de l'*automorphisme* non identique  $x \mapsto x^p$  de  $K$  qui est la cause de nos ennuis.

Il s'agit là d'un phénomène très général en mathématiques. On peut souvent définir un objet par des propriétés caractéristiques, ce qui signifie que deux objets possédant ces propriétés sont isomorphes. Mais il faut prendre garde lorsque cet isomorphisme n'est pas uniquement déterminé. C'est ainsi que l'on parle d'objets « uniques à isomorphisme près » ou « uniques à isomorphisme unique près ». Par exemple, le corps à 3 éléments est unique à isomorphisme unique près, mais « le » corps à 4 éléments ne l'est pas.

En résumé, il faut être très prudent lorsqu'on abrège l'expression «  $K$  est un corps à  $q$  éléments » par  $K = \mathbf{F}_q$ . Sinon par exemple, ayant écrit  $K = \mathbf{F}_q$ , puis  $K' = \mathbf{F}_{q^2}$ , on en tire  $K = K'$ , ce qui, étant donnés des éléments  $x$  de  $K$  et  $x'$  de  $K'$ , amène à la question interdite : a-t-on  $x = x'$  ?

Tout ceci n'est pas du purisme, et traduit une difficulté pratique considérable : si on veut calculer effectivement dans un corps fini, disons  $\mathbf{F}_{256}$ , il faut bien être capable d'en nommer les éléments et de les manipuler. La méthode la plus naturelle (que l'on pourrait appeler « additive »), c'est de choisir un polynôme irréductible  $P$  de degré 8 sur  $\mathbf{F}_2$  (on sait qu'il en existe, mais il faut en trouver un — et aussi savoir comment on discute avec celui qui en a choisi un autre !) et de prendre pour  $\mathbf{F}_{256}$  le quotient  $\mathbf{F}_2[X]/(P)$  correspondant. Une autre solution (« multiplicative » celle-là) est d'identifier  $\mathbf{F}_{256}^*$  au groupe cyclique  $\mathbf{Z}/255\mathbf{Z}$ , ce qui revient à choisir une racine primitive 255-ième dans le corps inconnu<sup>5</sup>, puis à déterminer la table d'addition. On peut combiner ces deux approches, en choisissant comme polynôme  $P$  un polynôme *primitif*, facteur irréductible de  $\Phi_{255}$  dans  $\mathbf{F}_2[X]$ . Nous expliquerons dans le paragraphe suivant cette méthode sur l'exemple un peu plus maniable de  $\mathbf{F}_{16}$ .

### 9.3.4. Racines de l'unité dans un corps fini

Soit  $k$  un corps fini, à  $q$  éléments et soit  $n$  un entier premier à  $q$ . Un corps  $K$  contenant  $k$  est appelé une *extension cyclotomique de niveau n de k* s'il possède  $n$  racines  $n$ -ièmes de l'unité, ce qui signifie aussi qu'il possède une racine primitive  $n$ -ième de l'unité, et s'il est minimal pour cette propriété. Cela signifie donc qu'il est obtenu par adjonction à  $k$  d'une racine primitive  $n$ -ième de l'unité. Le polynôme minimal d'une telle racine primitive est un facteur irréductible dans  $k[X]$  du polynôme cyclotomique  $\Phi_n$ . Or on a déterminé précédemment (proposition 9.17) le degré  $r$  d'un tel facteur, et il en résulte que le corps  $K$  a  $q^r$  éléments. Il est donc bien déterminé à isomorphisme près, ce qui permet de le noter  $R_n(k)$ , avec le même danger que celui qu'on a signalé pour  $\mathbf{F}_{p^r} = R_{p^r-1}(\mathbf{F}_p)$ . En résumé :

**PROPOSITION 9.28.** — *Soit  $k$  un corps fini à  $q$  éléments, et soit  $n$  un entier positif premier à  $q$ . Alors le corps  $R_n(k)$  a  $q^r$  éléments, où  $r$  est l'ordre de l'élément  $q$  dans le groupe  $(\mathbf{Z}/n\mathbf{Z})^*$ .*

Prenons par exemple  $q = 2$ . Alors  $q$  est d'ordre 4 à la fois modulo 5 et modulo 15. On a donc  $R_5(\mathbf{F}_2) = \mathbf{F}_{16}$  et aussi  $R_{15}(\mathbf{F}_2) = \mathbf{F}_{16}$ .

**EXERCICE 9.22. [A]** — Que dit la proposition pour  $n = 2$  ?

Prenons  $n = 3$ ; on a  $\Phi_3 = X^2 + X + 1$ . D'après ce qui précède, il revient au même de dire que  $X^2 + X + 1$  est irréductible dans  $k[X]$ , ou que  $R_3(k)$  a

---

5. Il y en a en fait  $\varphi(255) = \varphi(3)\varphi(5)\varphi(17) = 2 \times 4 \times 16 = 128$ .

$q^2$  éléments, ou que  $q$  est congru à  $-1$  modulo 3. On peut ainsi écrire

$$\mathbf{F}_{q^2} = \mathbf{R}_3(\mathbf{F}_q) = \mathbf{F}_q[X]/(X^2 + X + 1), \text{ pour } q \equiv -1 \pmod{3}.$$

En prenant  $n = 4$ , on obtient de même

$$\mathbf{F}_{q^2} = \mathbf{R}_4(\mathbf{F}_q) = \mathbf{F}_q[X]/(X^2 + 1), \text{ pour } q \equiv -1 \pmod{4}.$$

EXERCICE 9.23. [C] — Lorsque  $q \equiv -1 \pmod{12}$ , comparer les deux descriptions précédentes.

## § 9.4. Calculs explicites dans un corps fini

Compte tenu des applications que nous voulons en faire aux codes, nous nous spécialiserons au cas des corps de caractéristique 2.

### 9.4.1. Les corps à $2^m$ éléments

Pour décrire un corps  $\mathbf{F}_{2^m}$ , le plus simple est de le définir à partir du corps  $\mathbf{F}_2$  par adjonction, donc comme un quotient  $K = \mathbf{F}_2[X]/(P)$ , où  $P$  est un polynôme irréductible de degré  $m$  dans  $\mathbf{F}_2[X]$ . Ainsi, on a  $K = \mathbf{F}_2(\alpha)$ , avec  $P(\alpha) = 0$ . Tout élément s'écrit comme un polynôme en  $\alpha$  de degré  $\leq m - 1$ . L'addition est celle des polynômes ; pour la multiplication, on tient compte de la relation  $P(\alpha) = 0$  pour ramener le produit à être de degré  $\leq m - 1$ .

Comme tout élément non nul d'un corps fini, le générateur  $\alpha$  fixé est une racine de l'unité. Son ordre divise  $2^m - 1$  ; c'est aussi, comme on l'a vu, l'unique entier impair  $d$  tel que  $P$  divise  $\Phi_d$ . Il est commode de s'arranger pour que  $d$  soit maximal, donc  $d = 2^m - 1$ , car tout élément non nul de  $K$  est alors une puissance de  $\alpha$ . On doit prendre pour  $P$  un facteur irréductible de  $\Phi_{2^m-1}$ , ce qu'on a appelé un polynôme *primitif* de degré  $m$ .

Voici une liste de polynômes primitifs de degrés variant de 2 à 16.

$X^2 + X + 1$	$X^3 + X + 1$	$X^4 + X + 1$
$X^5 + X^2 + 1$	$X^6 + X + 1$	$X^7 + X + 1$
$X^8 + X^6 + X^5 + X^4 + 1$	$X^9 + X^4 + 1$	$X^{10} + X^3 + 1$
$X^{11} + X^2 + 1$	$X^{12} + X^7 + X^4 + X^3 + 1$	
$X^{13} + X^4 + X^3 + X + 1$	$X^{14} + X^{12} + X^{11} + X + 1$	
$X^{15} + X + 1$	$X^{16} + X^5 + X^3 + X^2 + 1$	

### 9.4.2. Exemple : le corps $\mathbf{F}_{16}$

Pour donner jusqu'au bout un exemple de basse complexité, nous considérons le (on devrait sans doute dire « un » et non pas « le ») corps  $\mathbf{F}_{16}$ . On pourrait le décrire comme le quotient  $\mathbf{R}_5(\mathbf{F}_2) = \mathbf{F}_2[X]/(X^4 + X^3 + X^2 + X + 1)$ . Mais il est plus commode, comme on l'a déjà dit, de le considérer

comme  $R_{15}(\mathbf{F}_2)$ , c'est-à-dire d'utiliser une racine primitive 15-ième de l'unité. Comme on l'a vu plus haut, on a deux polynômes minimaux possibles, à savoir les deux facteurs irréductibles de  $\Phi_{15}$  sur  $\mathbf{F}_2$ , qui sont  $X^4 + X^3 + 1$  et  $X^4 + X + 1$ . Choisissons par exemple le second, ce qui donne la relation de départ

$$\alpha^4 = \alpha + 1.$$

Les éléments de  $\mathbf{F}_{16}$  sont 0 et les quinze puissances  $\alpha^i$  avec  $i$  variant modulo 15, par exemple  $i = 0, \dots, 14$ . On pose aussi  $0 = \alpha^{-\infty}$ .

La multiplication de deux tels éléments est immédiate : on ajoute les exposants modulo 15. Naturellement, on a  $-\infty + i = -\infty$ . Pour  $i \neq -\infty$ , l'inverse de  $\alpha^i$  est  $\alpha^{-i} = \alpha^{15-i}$ .

Pour l'addition, on conserve une table exprimant chaque  $\alpha^i$  comme un polynôme de degré < 4 en  $\alpha$  (« table de logarithmes »). Si l'on pose  $[abcd] = a\alpha^3 + b\alpha^2 + c\alpha + d \in \mathbf{F}_{16}$ , où  $a, b, c$  et  $d$  sont des bits (éléments de  $\mathbf{F}_2$ ), l'addition se fait bit à bit sans retenue. La relation initiale  $\alpha + 1 = \alpha^4$  s'écrit  $\alpha^4 = [0011]$ . Pour calculer cette table de logarithmes, on multiplie les polynômes modulo  $\alpha^4 + \alpha + 1$ . On obtient ainsi successivement

$$\begin{aligned} 0 &= [0000], \\ 1 &= [0001], \\ \alpha &= [0010], \\ \alpha^2 &= [0100], \\ \alpha^3 &= [1000], \\ \alpha^4 &= \alpha + 1 = [0011], \\ \alpha^5 &= \alpha^2 + \alpha = [0110], \\ \alpha^6 &= \alpha^3 + \alpha^2 = [1100], \\ \alpha^7 &= \alpha^4 + \alpha^3 = \alpha^3 + \alpha + 1 = [1011], \\ \alpha^8 &= \alpha^4 + \alpha^2 + \alpha = \alpha^2 + 1 = [0101], \\ \alpha^9 &= \alpha^3 + \alpha = [1010], \\ \alpha^{10} &= \alpha^4 + \alpha^2 = \alpha^2 + \alpha + 1 = [0111], \\ \alpha^{11} &= \alpha^3 + \alpha^2 + \alpha = [1110], \\ \alpha^{12} &= \alpha^4 + \alpha^3 + \alpha^2 = \alpha^3 + \alpha^2 + \alpha + 1 = [1111], \\ \alpha^{13} &= \alpha^4 + \alpha^3 + \alpha^2 + \alpha = \alpha^3 + \alpha^2 + 1 = [1101], \\ \alpha^{14} &= \alpha^4 + \alpha^3 + \alpha = \alpha^3 + 1 = [1001]. \end{aligned}$$

On calcule ainsi par exemple :

$$\alpha^5 + \alpha^8 = [0110] + [0101] = [0011] = \alpha^4.$$

On notera que les éléments 0, 1,  $\alpha^5$  et  $\alpha^{10}$  forment le sous-corps à 4 éléments. On a d'ailleurs  $1 + \alpha^5 + \alpha^{10} = 0$ .

### 9.4.3. Le logarithme de Zech

Il existe une solution plus simple au problème de l'addition, qui reprend une idée due à Z. LEONELLI dans le cas des nombres réels. La table de Leonelli, publiée à Bordeaux en 1803 et légèrement simplifiée un peu plus tard par ZECH, donnait les valeurs de  $\log(1 + x)$  en fonction de  $\log(x)$ , ce qui permettait de calculer le logarithme d'une somme à partir des logarithmes des facteurs, en écrivant  $\log(a + b) = \log(a) + \log(1 + b/a)$  et en cherchant  $\log(1 + b/a)$  dans la table, en face de  $\log(b/a) = \log(b) - \log(a)$ .

Revenons au cas général de  $\mathbf{F}_{2^m}$ , et supposons choisie une racine primitive  $\alpha$ . Exprimons tout élément non nul du corps comme une puissance  $\alpha^i$ , l'exposant  $i$  variant modulo  $2^m - 1$ . On écrira aussi  $0 = \alpha^{-\infty}$ . Puisque  $0 + x = x$  et  $x + x = 0$ , il suffit de savoir ajouter des éléments non nuls et distincts. Mais on a  $\alpha^i + \alpha^j = \alpha^i(1 + \alpha^{j-i})$ ; si  $1 + \alpha^{j-i} = \alpha^k$ , alors  $\alpha^i + \alpha^j = \alpha^{i+k}$ .

Notons donc  $z$  l'application de l'ensemble des classes non nulles modulo  $2^m - 1$  dans lui-même définie par

$$1 + \alpha^i = \alpha^{z(i)}.$$

On a

$$\alpha^i + \alpha^j = \alpha^{i+z(j-i)} \text{ pour } i < j.$$

On ajoute évidemment

$$z(0) = -\infty, \quad z(-\infty) = 0.$$

Il suffit ainsi de pré-calculer la « table de Zech »  $z$  pour connaître intégralement la structure du corps. Ce calcul est simplifié par les trois faits suivants :

- ▷  $z(2i) = 2z(i)$ , puisque  $\alpha^{2z(i)} = (\alpha^{z(i)})^2 = (1 + \alpha^i)^2 = 1 + \alpha^{2i}$ ,
- ▷  $z(z(i)) = i$ , puisque  $1 + \alpha^{z(i)} = \alpha^i$ ,
- ▷  $z(-i) = z(i) - i$ , puisque  $1 + \alpha^{-i} = \alpha^{-i}(1 + \alpha^i) = \alpha^{-i}\alpha^{z(i)}$ .

Traitons maintenant l'exemple de  $\mathbf{F}_{16}$ . On peut déterminer la fonction  $z$  à l'aide de la table de logarithmes précédente. Mais on peut aussi la calculer directement comme suit.

On a au départ  $\alpha^4 = 1 + \alpha$ , donc  $z(1) = 4$ , d'où ensuite  $z(2) = 8$ , puis  $z(4) = 16 = 1$  et  $z(8) = 2$ . Ensuite, on a  $z(14) = z(-1) = z(1) - 1 = 4 - 1 = 3$ , d'où  $z(13) = z(28) = 2z(14) = 6$ ,  $z(11) = z(26) = 2z(13) = 12$ ,  $z(7) = z(22) = 2z(11) = 24 = 9$ . Symétriquement, on a  $z(3) = 14$ ,  $z(6) = 13$ ,  $z(12) = 11$  et  $z(9) = 7$ .

Il reste à calculer  $z(5)$  et  $z(10)$ . Comme toutes les valeurs sauf 5 et 10 ont été obtenues, on a nécessairement  $z(5) = 10$  et  $z(10) = 5$ , ce qu'on peut aussi vérifier dans la table (car on y lit  $\alpha^5 = [0110]$  et  $\alpha^{10} = [0111]$ ), ou directement comme suit :

$$\begin{aligned}\alpha^{10} &= \alpha^2\alpha^8 = \alpha^2(1+\alpha)^2 = \alpha^2(1+\alpha^2) = \alpha^2 + \alpha^4 = \\ &= \alpha^2 + \alpha + 1 = \alpha(\alpha + 1) + 1 = \alpha\alpha^4 + 1 = 1 + \alpha^5.\end{aligned}$$

On obtient ainsi la table suivante :

$i$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$z(i)$	4	8	14	1	10	13	9	2	7	5	12	11	6	3

Retenant le même exemple que précédemment, on aura

$$\alpha^5 + \alpha^8 = \alpha^5(1 + \alpha^3) = \alpha^{5+14} = \alpha^4.$$

EXERCICE 9.24. [B] — Traiter le cas du corps  $\mathbf{F}_{32}$ , défini par le polynôme minimal  $X^5 + X^2 + 1$ .

EXERCICE 9.25. [B] — Traiter le cas du corps  $\mathbf{F}_{64}$ , défini par le polynôme minimal  $X^6 + X + 1$ .

EXERCICE 9.26. [N] — Écrire un ensemble de programmes présentant de la façon précédente les corps  $\mathbf{F}_{2^r}$ . Le tester sur le corps  $\mathbf{F}_{256}$ .

### § 9.5. Démonstration du théorème AKS

Nous allons maintenant démontrer le théorème de Agrawal, Kayal et Saxena signalé en (5.4.4). Rappelons-en l'énoncé :

THÉORÈME 9.29. — Soit  $n$  un entier  $> 1$ . Soit  $r$  un entier  $> 1$  premier à  $n$ , choisi de telle façon que l'ordre de la classe de  $n$  dans le groupe multiplicatif  $(\mathbf{Z}/r\mathbf{Z})^*$  (qui a  $\varphi(r)$  éléments) soit  $> (\log n)^2$ . Posons  $s = \sqrt{\varphi(r)} \log n$ . Supposons que pour tout entier  $a$  avec  $1 \leq a \leq s$ , on ait

$$(X + a)^n \equiv X^n + a \pmod{(n, X^r - 1)}.$$

Alors, ou bien  $n$  possède un facteur premier inférieur à  $s$ , ou bien  $n$  est une puissance d'un nombre premier.

La démonstration est une suite astucieuse de pas élémentaires.

Pour le moment, considérons seulement la situation suivante : on se donne un nombre premier  $p$ , un multiple  $n = pm$  de  $p$ , et un entier  $r > 1$  premier à  $n$ . On considère une partie  $A$  du corps  $\mathbf{F}_p$  et on suppose qu'on a dans  $\mathbf{F}_p[X]$  la congruence

$$(H) \quad (X + a)^n \equiv X^n + a \pmod{(X^r - 1)} \text{ pour tout } a \in A.$$

Considérons le corps  $K = R_r(\mathbf{F}_p)$ , extension cyclotomique de niveau  $r$  de  $\mathbf{F}_p$ , et soit  $\zeta \in K$  une racine primitive  $r$ -ième de l'unité dans  $K$  (voir 9.3.4).

On a  $n \cdot 1_K = 0$  et  $(\zeta^i)^r = 1$  pour tout entier  $i$ , de sorte que l'hypothèse (H) implique

$$(H') \quad (\zeta^i + a)^n = \zeta^{in} + a, \quad i \in \mathbf{Z}/r\mathbf{Z}, \quad a \in A.$$

Notons J l'ensemble des entiers  $j \in \mathbf{N}$  qui sont premiers à  $r$  et tels qu'on ait  $(\zeta^i + a)^j = \zeta^{ij} + a$  pour tout  $i \in \mathbf{Z}/r\mathbf{Z}$  et tout  $a \in A$ .

**LEMME 9.30.** — *L'ensemble J est stable par multiplication et contient p, m et n.*

*Démonstration.* Si  $j$  et  $j'$  sont deux éléments de J, on a pour tout  $a \in A$

$$(\zeta^i + a)^{jj'} = ((\zeta^i + a)^j)^{j'} = (\zeta^{ij} + a)^{j'} = \zeta^{ijj'} + a.$$

Par ailleurs, J contient  $p$  d'après Fermat, et contient  $n = pm$  par l'hypothèse (H'). Démontrons qu'il contient  $m$ . Soit  $a \in A$ . Alors  $((\zeta^i + a)^m - \zeta^{im} - a)^p = (\zeta^i + a)^n - \zeta^{in} - a = 0$ . Comme on est dans un corps, cela implique bien  $(\zeta^i + a)^m - \zeta^{im} - a = 0$ .  $\square$

Notons Z l'ensemble des  $\zeta^j$  pour  $j \in J$  et posons  $t = \#Z$ .

**LEMME 9.31.** — *On a  $\omega \leq t \leq \varphi(r)$ , où  $\omega$  est l'ordre de la classe de n dans le groupe multiplicatif  $(\mathbf{Z}/r\mathbf{Z})^*$ .*

*Démonstration.* Puisque  $\zeta$  est d'ordre  $r$  et que Z contient les puissances de  $\zeta$ ,  $t$  est au moins égal à l'ordre de la classe de  $n$  dans le groupe multiplicatif  $\mathbf{Z}/r\mathbf{Z}$ . Mais il est aussi au plus égal à l'ordre  $\varphi(r)$  de ce groupe.  $\square$

Pour chaque famille (partie avec répétitions) E d'éléments de A, posons

$$P_E(X) = \prod_{a \in E} (X + a) \in \mathbf{F}_p[X], \quad x_E = P_E(\zeta) = \prod_{a \in E} (\zeta + a) \in K.$$

Alors, par définition, on a pour tout  $j \in J$ ,

$$P_E(\zeta^j) = P_E(\zeta)^j = x_E^j.$$

**LEMME 9.32.** — *Les  $x_E$ , pour  $\#E < t$ , sont tous distincts.*

*Démonstration.* Si  $x_E = x_{E'}$ , alors les polynômes  $P_E$  et  $P_{E'}$  prennent les mêmes valeurs en tous les éléments de Z, donc  $P_E - P_{E'}$  a au moins  $t$  racines. Si E et E' ont moins de  $t$  termes, cela implique  $P_E = P_{E'}$ , donc  $E = E'$ .  $\square$

Si l'on pose  $s = \#A - 1$ , ces  $x_E$  sont au nombre de  $\binom{t+s}{s+1}$ .

**LEMME 9.33.** — *Si n n'est pas une puissance de p, les  $x_E$  sont en nombre au plus  $n^{\lfloor \sqrt{t} \rfloor}$ .*

*Démonstration.* Si  $n$  n'est pas une puissance de  $p$ , alors  $m = n/p$  n'est pas non-plus une puissance de  $p$ . Les entiers de la forme  $j = p^u m^v$  sont donc tous distincts. Ce sont des éléments de  $J$ . Considérons ceux pour lesquels on a  $0 \leq u, v \leq \sqrt{t}$ . Ils sont majorés par  $n^{\lfloor \sqrt{t} \rfloor}$ . Ils sont en nombre  $(1 + \lfloor \sqrt{t} \rfloor)^2 > t$ . Comme les  $\zeta^j$  sont au nombre de  $t$ , il existe au moins<sup>6</sup> deux entiers distincts de cette forme  $j = p^u m^v$  et  $j' = p^{u'} m^{v'}$ , avec  $\zeta^j = \zeta^{j'}$ .

Cela donne pour tout  $E$

$$x_E^j = P_E(\zeta^j) = P_E(\zeta^{j'}) = x_E^{j'},$$

ce qui signifie que le polynôme  $X^j - X^{j'}$  possède tous les  $x_E$  comme racine. Mais il est de degré  $\leq n^{\lfloor \sqrt{t} \rfloor}$ , ce qui achève la démonstration.  $\square$

Confrontant les résultats obtenus :

**PROPOSITION 9.34.** — *Supposons que  $n$  ne soit pas une puissance de  $p$ . On a alors  $\binom{t+s}{s+1} \leq n^{\lfloor \sqrt{t} \rfloor}$ , avec  $s = \#A - 1$  et  $\omega \leq t \leq \varphi(r)$ .*

Venons-en maintenant à la démonstration du théorème.

*Démonstration.* Soit  $p$  un facteur premier de  $n$ . Si on a  $p \leq s$ , ou si  $n$  est une puissance de  $p$ , on a terminé. Sinon, on choisit pour  $A$  l'ensemble des classes modulo  $p$  des entiers entre 0 et  $s$ , qui sont deux à deux distinctes. La proposition précédente donne l'inégalité

$$(*) \quad \binom{t+s}{s+1} \leq n^{\lfloor \sqrt{t} \rfloor}$$

avec par hypothèse  $(\log n)^2 < \omega \leq t \leq \varphi(n)$ .

Montrons que c'est impossible. Posons  $\alpha = \lfloor \sqrt{t} \log n \rfloor$ , de sorte que  $2 \leq \alpha \leq s$ . On a d'un côté  $\sqrt{t} > \log n$ , donc  $t > \sqrt{t} \log n$ , soit  $t \geq \alpha + 1$ , et aussi  $s \geq \alpha$ . Cela donne

$$\binom{t+s}{s+1} \geq \binom{\alpha+1+s}{s+1} = \binom{\alpha+1+s}{\alpha} \geq \binom{2\alpha+1}{\alpha}.$$

Mais on a

$$\begin{aligned} \binom{2\alpha+1}{\alpha} &= \frac{(2\alpha+1)\cdots(\alpha+2)}{\alpha\cdots 1} = \frac{2\alpha+1}{\alpha} \cdots \frac{\alpha+3}{2} \frac{\alpha+2}{1} \\ &> 2^{\alpha-1} \cdot 4 = 2^{\alpha+1}. \end{aligned}$$

En définitive, le premier membre de  $(*)$  est strictement supérieur à  $2^{\alpha+1}$ .

D'un autre côté, on a  $\sqrt{t} \log n \leq \alpha + 1$ , donc le second membre de  $(*)$  est inférieur à  $2^{\alpha+1}$ , ce qui est contradictoire et achève la démonstration.  $\square$

6. Ce type de démonstration a été baptisé « principe des tiroirs » par Dirichlet en 1836 : si on a  $t$  tiroirs et  $m$  chaussettes, avec  $m > t$ , l'un des tiroirs contient au moins deux chaussettes ; l'anecdote selon laquelle l'idée lui en est venue en regardant effectivement ses tiroirs à chaussettes semble vérifique.

## § 9.6. Décomposition des polynômes dans $\mathbf{F}_p[X]$

Ce paragraphe est consacré à des algorithmes de décomposition de polynômes en facteurs irréductibles. Notons d'abord qu'il suffit de posséder un algorithme qui, partant d'un polynôme  $P$ , soit réponde qu'il est irréductible, soit fournit *un* facteur non trivial, disons  $Q$  : il n'y aura qu'à appliquer récursivement l'algorithme à  $Q$  et  $P/Q$ .

Par ailleurs, les algorithmes (notamment celui de Berlekamp) que nous présentons ci-dessous s'appliquent à des polynômes sans facteur multiple, cas auquel on peut toujours se ramener par une réduction préliminaire que nous exposons d'abord.

### 9.6.1. Polynômes sans facteur multiple

Fixons un corps  $K$  et soit  $P$  un polynôme unitaire de  $K[X]$ . On sait que  $P$  admet une décomposition de la forme

$$P = \prod_{i=1}^n P_i^{m_i},$$

où les  $m_i$  sont des entiers  $> 0$  et les  $P_i$  des polynômes unitaires irréductibles deux à deux distincts. Notre but est de calculer cette décomposition.

Le polynôme dérivé  $P'$  s'écrit

$$P' = Q \cdot \prod_{i=1}^n P_i^{m_i-1},$$

où  $Q$  est une somme de  $n$  termes

$$Q = m_1 P'_1 P_2 \cdots P_n + m_2 P_1 P'_2 \cdots P_n + \cdots.$$

Si  $m_1$  n'est pas nul dans  $K$ ,  $Q$  n'est pas divisible par  $P_1$ , et ainsi de suite. On voit donc que le pgcd de  $P$  et  $P'$  peut s'exprimer par

$$\operatorname{pgcd}(P, P') = \prod_{i=1}^n P_i^{m'_i},$$

où  $m'_i = m_i$  si  $m_i 1_K = 0$  et  $m'_i = m_i - 1$  sinon. Dire que  $\operatorname{pgcd}(P, P') = 1$ , c'est dire que tous les  $m_i$  sont égaux à 1. Par conséquent :

**PROPOSITION 9.35.** — *Les conditions suivantes sont équivalentes :*

- (i)  $P$  et  $P'$  sont premiers entre eux ;
- (ii) les facteurs irréductibles de  $P$  apparaissent tous avec la multiplicité 1 ;
- (iii) il n'existe pas de polynôme non constant  $Q$  tel que  $Q^2$  divise  $P$ .

**EXERCICE 9.27. [C]** — Démontrer que la propriété suivante est équivalente aux trois autres :

(iv)  $P$  n'a de racines multiples dans aucun corps contenant  $K$ .

Un tel polynôme est dit *sans facteur multiple*<sup>7</sup>.

Si on regroupe les facteurs de  $P$  suivant leur multiplicité, on voit qu'on peut écrire

$$P = Q_1 Q_2^2 \cdots Q_r^r,$$

où les  $Q_i$  sont premiers entre eux et sans facteur multiple. Si on suppose pour simplifier que  $K$  est de caractéristique 0, on a

$$\text{pgcd}(P, P') = Q_2 Q_3^2 \cdots Q_r^{r-1}.$$

L'algorithme d'Euclide permet de calculer rapidement le pgcd de  $P$  et  $P'$ , et on voit qu'en réitérant cette opération on obtiendra  $Q_r$ , puis  $Q_{r-1}$ , et ainsi de suite. La décomposition précédente peut donc s'obtenir rapidement par cette méthode lorsque  $K$  est de caractéristique 0.

Mais on peut faire plus simple dans tous les cas. En effet, si l'on calcule un pgcd de  $P$  et  $P'$ , alors de deux choses l'une : ou bien on obtient un facteur non constant de  $P$  et on a essentiellement terminé ; ou bien on obtient une constante, et alors  $P$  est sans facteur multiple. Il suffit donc de savoir décomposer les polynômes sans facteur multiple pour pouvoir décomposer tout polynôme.

**EXERCICE 9.28. [B]** — Prouver que tout polynôme unitaire  $P$  s'écrit de façon unique  $Q^2 R$ , où  $Q$  et  $R$  sont unitaires, et  $R$  est sans facteur multiple.

**EXERCICE 9.29. [C]** — Déduire de l'exercice précédent que le nombre de polynômes unitaires sans facteur multiple de degré  $n$  sur  $\mathbf{F}_p$  vaut  $p$  pour  $n = 1$  et vaut  $p^n - p^{n-1}$  pour  $n > 1$ .

### 9.6.2. L'algorithme de Berlekamp

Considérons un nombre premier  $p$  et un polynôme unitaire  $P \in \mathbf{F}_p[X]$  sans facteur multiple. On sait que  $P$  peut s'écrire

$$P = P_1 \cdots P_r$$

où les  $P_i$  sont unitaires irréductibles et premiers entre eux, et il s'agit de les déterminer. Chaque anneau-quotient  $K_i = \mathbf{F}_p[X]/(P_i)$  est un corps, et l'anneau  $A = \mathbf{F}_p[X]/(P)$  est isomorphe au produit des  $K_i$  d'après le théorème chinois. De façon précise, on peut identifier  $A$  à  $K_1 \times \cdots \times K_r$  par l'application

$$(Q \bmod P) \mapsto (Q \bmod P_1, \dots, Q \bmod P_r).$$

---

7. Comme pour les entiers, on dit « *square-free* » en anglais, « *quadratfrei* » en allemand.

Dans chaque  $K_i$ , l'équation  $a^p = a$  possède exactement  $p$  solutions, à savoir les éléments du sous-corps premier. Par conséquent, l'équation  $a^p = a$  possède exactement  $p^r$  solutions dans A. De façon plus précise, il y a  $p^r$  polynômes Q modulo P qui sont solutions de la congruence  $Q^p \equiv Q \pmod{P}$ , et chaque solution correspond à l'un des  $p^r$  vecteurs  $(\alpha_1, \dots, \alpha_r) \in \mathbf{F}_p^r$  par

$$Q \equiv \alpha_i \pmod{P_i}, \quad \alpha_i \in \mathbf{F}_p, \quad i = 1, \dots, r.$$

Mais l'application  $a \mapsto a^p$  est *linéaire* sur le corps  $\mathbf{F}_p$ . Si  $n$  est le degré de P, on peut prendre pour base de l'espace vectoriel A les classes  $x^i$  des  $X^i$  pour  $0 \leq i < n$ , et on obtiendra la matrice de l'application S :  $a \mapsto a^p$  par un calcul (rapide) des puissances faisant intervenir à chaque pas une division euclidienne : si  $X^{ip}$  est congru à  $a_{i,0} + a_{i,1}X + \dots + a_{i,n-1}X^{n-1}$  modulo P, on a

$$S(x^i) = \sum a_{i,j} x^j.$$

On détermine alors le noyau N de  $S - I$  par un algorithme usuel d'algèbre linéaire. Sa dimension est  $r$ , ce qui donne déjà un critère d'irréductibilité. Si Q (on devrait sans doute dire  $Q(x)$ ) est un élément de N, il existe des  $\alpha_i \in \mathbf{F}_p$  tels que  $Q - \alpha_i$  soit divisible par  $P_i$  pour tout  $i$ . Si on choisit  $\alpha \in \mathbf{F}_p$ , le pgcd de P et de  $Q - \alpha$  est le produit des  $P_i$  tels que  $\alpha_i = \alpha$ . On a donc

$$P = \prod_{\alpha \in \mathbf{F}_p} \text{pgcd}(P, Q - \alpha).$$

Si Q n'est pas constant, alors les  $\alpha_i$  ne sont pas tous égaux, et il existe un  $\alpha$  tel que le pgcd de P et de  $Q - \alpha$  soit distinct de 1 et de P. Prenant successivement  $\alpha = 0, 1, \dots, p - 1$ , cela permet d'obtenir un diviseur non trivial de P (et en fait la décomposition complète de P en faisant parcourir à Q une base de N).

**EXERCICE 9.30. [B] –** Déduire directement l'identité précédente de l'égalité

$$\prod_{\alpha \in \mathbf{F}_p} (Q - \alpha) = Q^p - Q \equiv 0 \pmod{P}.$$

### 9.6.3. Une variante probabiliste

L'algorithme précédent est raisonnablement rapide lorsque  $p$  est petit, mais impraticable pour de très grands  $p$ . En effet, si la première partie (calcul de la matrice S et du noyau de  $S - I$ ) reste raisonnable même pour de grands  $p$ , il n'en est pas de même de la seconde qui demande d'essayer  $p$  valeurs de  $\alpha$ .

On peut alors utiliser une autre idée. Si Q est un élément du noyau N de  $S - I$ , l'image  $\alpha_i$  de Q dans chacun des corps  $K_i$  appartient au sous-corps premier et on a, soit  $\alpha_i = 0$ , soit  $(\alpha_i)^{(p-1)/2} = \pm 1$ . Par conséquent, chacun

des éléments  $Q, Q^{(p-1)/2} - 1$  et  $Q^{(p-1)/2} + 1$  est divisible exactement par ceux des  $P_i$  tel que  $\alpha_i$  soit respectivement nul, un résidu quadratique, un non-résidu (on a supposé  $p > 2$ ). Dit autrement, puisque  $P$  divise le produit  $Q(Q^{(p-1)/2} - 1)(Q^{(p-1)/2} + 1)$ , on a la décomposition

$$P = \text{pgcd}(P, Q) \text{pgcd}(P, Q^{(p-1)/2} - 1) \text{pgcd}(P, Q^{(p-1)/2} + 1)$$

et cette décomposition n'est triviale que si les  $\alpha_i$  sont tous nuls, tous résidus ou tous non résidus. Or la « probabilité » pour qu'un tel événement se produise est  $\leq 2^{1-r}$ , donc est  $\leq 1/2$  si  $P$  est réductible.

On voit ainsi comment fabriquer un algorithme probabiliste : on tire au sort successivement des éléments de  $N$  (c'est-à-dire leurs coefficients dans la base déterminée dans la première phase). Si on trouve par le pgcd précédent un facteur non trivial de  $P$ , tant mieux. Si on n'en trouve pas au bout de  $m$  répétitions, on décide que  $P$  est irréductible, avec un risque d'erreur majoré par  $1/2^m$ .

#### 9.6.4. L'algorithme de Cantor-Zassenhaus

L'idée précédente est à la base d'un algorithme probabiliste encore plus simple. Soit toujours  $P(X) \in \mathbf{F}_p[X]$  un polynôme sans facteur multiple. On remarque d'abord que, d'après la proposition 9.22, le pgcd de  $P(X)$  et  $X^{p^r} - X$  est le produit des facteurs irréductibles de  $P$  dont le degré divise  $r$ . En déterminant ces pgcd pour  $r = 1, \dots, \deg(P)$ , on peut écrire  $P$  sous la forme  $P_1 \cdots P_d$  où, pour chaque  $r$ ,  $P_r$  est un produit de polynômes irréductibles de degré  $r$  exactement. Il nous suffit donc de savoir décomposer de tels polynômes. On suppose comme plus haut que  $p > 2$ .

Soit donc  $P$  un polynôme dont on sait à l'avance que tous ses facteurs irréductibles sont distincts et de degré  $r$ , donc qu'il divise  $X^{p^r} - X$ . Notons que, pour tout polynôme  $Q(X) \in \mathbf{F}_p[X]$ ,  $X^{p^r} - X$  divise  $Q^{p^r} - Q$  : en effet, si  $Q(X) = \sum a_i X^i$ , on a  $Q^{p^r} - Q = \sum a_i (X^{ip^r} - X^i)$ . Mais on a par ailleurs la décomposition

$$Q^{p^r} - Q = Q(Q^{(p^r-1)/2} - 1)(Q^{(p^r-1)/2} + 1),$$

Ces trois facteurs étant premiers entre eux, on en tire l'identité

$$P = \text{pgcd}(P, Q) \text{pgcd}(P, Q^{(p^r-1)/2} - 1) \text{pgcd}(P, Q^{(p^r-1)/2} + 1).$$

On tire alors au sort  $Q$  parmi les polynômes unitaires de degré  $< 2r$ .

**EXERCICE 9.31. [C]** — Vérifier que si  $P$  n'est pas irréductible, la probabilité que la décomposition précédente soit triviale est  $\leq 1/2$ .

**REMARQUE 9.36.** — Naturellement, pour calculer le pgcd de  $P$  et de polynômes du genre  $X^{p^r} - X$  ou  $Q^{(p^r-1)/2} \pm 1$ , on calcule d'abord  $X^{p^r}$  modulo  $P$  ou  $Q^{(p^r-1)/2}$

modulo  $P$  par l'algorithme rapide de calcul des puissances dans l'anneau quotient  $\mathbf{F}_p[X]/(P(X))$ .

### 9.6.5. Décomposition des polynômes dans $\mathbf{Q}[X]$

Donnons seulement quelques indications. D'abord, le lemme de Gauss permet de ramener la décomposition en facteurs dans  $\mathbf{Q}[X]$  à la décomposition en facteurs dans  $\mathbf{Z}[X]$ . Cela étant, si  $P(X) \in \mathbf{Z}[X]$  s'écrit  $Q(X)R(X)$ , avec  $Q(X) \in \mathbf{Z}[X]$  et  $R(X) \in \mathbf{Z}[X]$ , une décomposition analogue est valable dans  $(\mathbf{Z}/p\mathbf{Z})[X]$  pour tout nombre premier  $p$ . Or, on dispose des algorithmes précédents pour résoudre ce dernier problème. En définitive, on se ramène au problème de la « remontée » : reconnaître si une décomposition en facteurs modulo  $p$  peut ou non se relever dans  $\mathbf{Z}$ . Pour cela, on dispose de deux méthodes. L'une consiste à changer de nombre premier  $p$  et à utiliser en définitive le théorème chinois. L'autre consiste à remonter de proche en proche dans  $\mathbf{Z}/p^n\mathbf{Z}$  par la méthode dite de Hensel. Nous renvoyons pour plus de détails à [Knuth 2].



# Chapitre 10

## Codes linéaires cycliques

Nous avons jusqu'à présent parlé de codes formés de mots *binaires*. Nous allons généraliser ce que nous avons dit au cas où les symboles élémentaires des mots ne sont plus nécessairement de simples bits (des éléments de  $\mathbf{F}_2$ ), mais appartiennent à un corps fini  $\mathbf{F}_q$  quelconque. Dans les applications, en dehors du cas binaire  $q = 2$ , on a le plus souvent  $q = 2^s$  (auquel cas les éléments de  $\mathbf{F}_q$  peuvent être codés par des suites de  $s$  bits, opération dite de « démultiplication » que nous détaillerons plus loin), mais on rencontre aussi par exemple le cas  $q = 3$ .

Comme dans le cas binaire, nous allons pour simplifier nous restreindre aux codes linéaires.

### § 10.1. Codes linéaires sur $\mathbf{F}_q$

Les définitions données pour les codes binaires s'étendent sans difficulté aux codes généraux. Faisons-le rapidement.

#### 10.1.1. Paramètres d'un code linéaire

On fixe donc un corps fini  $\mathbf{F}_q$ . Considérons deux entiers  $k$  et  $n$  avec  $0 \leq k \leq n$ .

DÉFINITION 10.1. — *On appelle code linéaire de longueur  $n$  et de dimension  $k$  sur  $\mathbf{F}_q$ , un sous-espace vectoriel  $C$  de dimension  $k$  de  $\mathbf{F}_q^n$ .*

Si  $q = 2$ , on dit comme précédemment que le code est *binaire*. On parle de code *ternaire* lorsque  $q = 3$ .

On définit le *poids*  $w(\mathbf{m})$  d'un mot  $\mathbf{m} \in \mathbf{F}_q^n$  comme le nombre de composantes non nulles de  $\mathbf{m}$ . La *distance de Hamming* sur  $\mathbf{F}_q^n$  est donnée par  $d(\mathbf{m}, \mathbf{m}') = w(\mathbf{m} - \mathbf{m}')$ . La *distance minimale* du code  $C$  est la valeur minimale  $d$  de  $w(\mathbf{m})$  lorsque  $\mathbf{m}$  parcourt les éléments non nuls de  $C$ . C'est aussi la distance minimale de deux éléments distincts de  $C$ . On dit souvent que  $q$ ,  $n$ ,  $k$  et  $d$  sont les *paramètres du code*, ou que  $C$  est de paramètres  $(q; n, k, d)$ .

REMARQUE 10.2. — Il faut noter que la distance de Hamming de deux mots  $\mathbf{m}$  et  $\mathbf{m}'$  ne fait intervenir que le nombre d'indices  $i$  tels que la différence  $\mathbf{m}_i - \mathbf{m}'_i$  soit non

nulle, et non pas la valeur de ces différences. Cela est naturel dans le cas où  $q = 2$  (puisque il n'existe qu'une valeur non nulle possible), mais pas nécessairement dans le cas général.

**EXERCICE 10.1. [B]** — Soit  $P$  une matrice sur  $\mathbf{F}_q$  à  $n$  lignes et  $r$  colonnes. Notons aussi  $P : \mathbf{F}_q^n \mapsto \mathbf{F}_q^r$  l'application linéaire associée. Supposons que cette application soit surjective, c'est-à-dire que la matrice soit de rang  $r$ . Alors le noyau  $\text{Ker}(P)$  est un sous-espace vectoriel de dimension  $n - r$  de  $\mathbf{F}_q^n$ , c'est-à-dire un code linéaire de longueur  $n$  et de dimension  $n - r$ . Décrire la distance minimale  $d$  de  $\text{Ker}(M)$  en termes de la matrice  $M$ . À quelle condition a-t-on  $d \geq 3$  ?

**EXERCICE 10.2. [B]** — (Codes de Hamming.) On fixe un entier  $r > 0$  et on considère tous les vecteurs non nuls de  $\mathbf{F}_q^r$  ayant la propriété suivante : leur premier coefficient non nul est égal à 1. Combien y en a-t-il ? On place ces vecteurs dans un ordre arbitraire pour former les colonnes une matrice, et on considère le code noyau de cette matrice, comme dans l'exercice 10.1. Montrer que la distance minimale de ce code est 3.

### 10.1.2. Décodage

Tout se passe exactement comme dans le cas  $q = 2$ . Soit  $d$  la distance minimale de  $C$ . On peut détecter l'existence d'une erreur  $\mathbf{e}$  telle que  $w(\mathbf{e}) < d$ . En effet, si le mot de code  $\mathbf{m} \in C$  est perturbé par l'erreur  $\mathbf{e} \neq 0$  avec  $w(\mathbf{e}) < d$ , on a  $\mathbf{m}' = \mathbf{m} + \mathbf{e} \notin C$ ; sinon, on aurait  $\mathbf{e} = \mathbf{m}' - \mathbf{m} \in C$ , ce qui contredit la définition de  $d$ . On peut corriger une erreur  $\mathbf{e}$  telle que  $w(\mathbf{e}) < d/2$ . En effet, si  $\mathbf{m}' \in \mathbf{F}_q^n$  peut s'écrire  $\mathbf{m}_1 + \mathbf{e}_1$  avec  $\mathbf{m}_1 \in C$  et  $w(\mathbf{e}_1) < d/2$ , et aussi  $\mathbf{m}_2 + \mathbf{e}_2$  avec  $\mathbf{m}_2 \in C$  et  $w(\mathbf{e}_2) < d/2$ , on a

$$w(\mathbf{m}_1 - \mathbf{m}_2) = w(\mathbf{e}_2 - \mathbf{e}_1) \leqslant w(\mathbf{e}_2) + w(\mathbf{e}_1) < d,$$

donc  $\mathbf{m}_1 = \mathbf{m}_2$  par définition. L'opération consistant à déduire du mot erroné  $\mathbf{m} + \mathbf{e}$  le mot de code  $\mathbf{m}$  s'appelle le *décodage*. Si  $2t < d$ , le code permet donc de corriger une erreur  $\mathbf{e}$  avec  $w(\mathbf{e}) \leqslant t$ , il est dit *t-correcteur*.

**EXERCICE 10.3. [B]** — On peut corriger simultanément  $t$  erreurs et  $f$  effacements, si  $2t + f < d$ .

La méthode de décodage la plus évidente consiste, le mot reçu  $\mathbf{m}'$  étant donné, à inspecter les  $q^k$  mots  $\mathbf{m}$  du code jusqu'à ce qu'on en trouve un tel que  $w(\mathbf{m}' - \mathbf{m}) \leqslant t$ . Il s'agit là bien évidemment d'un algorithme très lent, impraticable lorsque  $q$  ou  $k$  est grand. Toute la difficulté consiste à trouver des algorithmes de décodage praticables.

### 10.1.3. Codes parfaits

Fixons un entier  $t \geqslant 0$ . À tout mot de code  $\mathbf{m}$ , on peut associer la boule de rayon  $t$  centrée en  $\mathbf{m}$  : elle est formée des  $\mathbf{m}' \in \mathbf{F}_q^n$  avec  $w(\mathbf{m}' - \mathbf{m}) \leqslant t$ .

Dire que  $C$  est  $t$ -correcteur, c'est dire que ces  $q^k$  boules sont disjointes. On dit que  $C$  est *parfait* si ces boules forment une partition de l'espace total  $\mathbf{F}_q^n$ .

Le nombre d'éléments d'une boule de rayon  $t$  est

$$1 + n(q - 1) + \binom{n}{2}(q - 1)^2 + \cdots + \binom{n}{t}(q - 1)^t.$$

EXERCICE 10.4. [A] — Pourquoi ?

EXERCICE 10.5. [A] — Qu'obtient-on pour  $t = n$  ?

Si  $C$  est  $t$ -correcteur, on a donc

$$1 + n(q - 1) + \binom{n}{2}(q - 1)^2 + \cdots + \binom{n}{t}(q - 1)^t \leq q^{n-k},$$

avec égalité si  $C$  est parfait.

EXERCICE 10.6. [B] — Le code construit dans l'exercice 10.2 est 1-correcteur et parfait.

REMARQUE 10.3. — Les calculs qui précédent ne supposent pas que le code soit linéaire, mais seulement qu'il ait  $q^k$  éléments.

#### 10.1.4. Codes sur $\mathbf{F}_q$ et codes sur $\mathbf{F}_p$

Soit  $p$  la caractéristique de  $\mathbf{F}_q$ , de sorte que  $q = p^s$ . Comme on peut identifier additivement  $\mathbf{F}_q$  à  $\mathbf{F}_p^s$ , un code sur  $\mathbf{F}_q$  donne naissance à un code sur  $\mathbf{F}_p$ . Expliquons-le dans le cas essentiel, celui où  $p = 2$ .

Soit donc  $C$  un code linéaire, de paramètres  $(q = 2^s; n, k, d)$ . Choisissons une base du  $\mathbf{F}_2$ -espace vectoriel  $\mathbf{F}_q$ . On peut alors représenter tout élément, disons  $x$ , de  $\mathbf{F}_q$  par une suite  $(x_0, \dots, x_{s-1})$  de  $s$  bits. De cette façon, tout élément

$$\mathbf{m} = (m_0, \dots, m_{n-1}) \in \mathbf{F}_q^n$$

s'écrit comme un mot de  $ns$  bits

$$(\mathbf{m})_{(2)} = (m_{0,0}, \dots, m_{0,s-1}, m_{1,0}, \dots, m_{n-1,s-1}) \in \mathbf{F}_2^{sn},$$

et  $C$  s'interprète comme un code linéaire *binaire* de longueur  $sn$ . Il est clair que la dimension de  $C$  comme espace vectoriel sur  $\mathbf{F}_2$  est  $ks$ .

On appelle souvent *démultiplication* l'opération précédente, qui transforme un code linéaire de paramètres  $(2^s; n, k, d)$  en un code binaire de paramètres  $(ns, ks, ?)$ .

Que dire de la distance minimale de ce code binaire ? Elle est évidemment au moins égale à la distance minimale  $d$  du code initial, puisque le nombre de bits non nuls de  $\mathbf{m}_{(2)}$  est au moins égal au nombre de composantes non nulles de  $\mathbf{m}$ . Elle est aussi au plus égale à  $ds$  (prendre pour  $\mathbf{m}$  un mot de poids  $d$ ).

EXERCICE 10.7. [B] — Elle est en fait au plus égale à  $(d - 1)s + 1$ .

Mais deux choses sont à signaler ici. D'abord, il y arrive souvent que cette distance soit largement supérieure à  $d$ . Mais, même si ce n'est pas le cas, le code binaire C a un excellent comportement vis-à-vis des *bouffées d'erreurs*. Expliquons cela. Fixons  $t$  avec  $2t < d$ , de sorte que le code initial est  $t$ -correcteur. On peut donc corriger un ensemble d'erreurs de transmission qui ne modifient qu'au plus  $t$  des  $n$  coefficients initiaux (dans  $\mathbf{F}_q$ ), autrement dit qui se répartissent dans au plus  $t$  des  $n$  tranches de  $s$  bits du code démultiplié. C'est le cas par exemple si ces erreurs tiennent dans un intervalle de longueur totale d'au plus  $(t - 1)s + 1$  bits. Ce nombre est nettement plus grand que  $t$  (et d'autant plus que  $s$  est plus grand), alors que, pour des erreurs réparties aléatoirement, on ne peut a priori dépasser le nombre de  $t$  erreurs, chacune d'elle pouvant tomber dans une tranche différente.

Ainsi, lorsqu'on cherche à se prémunir contre des erreurs arrivant par bouffées plutôt qu'aléatoirement, il est tout naturel de choisir un code binaire obtenu par démultiplication à partir d'un code sur un corps  $\mathbf{F}_{2^s}$ , avec  $s$  grand. Par exemple, des codes binaires obtenus par démultiplication de codes sur  $\mathbf{F}_{2^8} = \mathbf{F}_{256}$  interviennent dans le codage des disques compacts (voir 12.2.2).

### 10.1.5. Un exemple

Prenons le corps non premier le plus simple, c'est-à-dire  $\mathbf{F}_4$ . Il est formé des quatre éléments 0, 1,  $\alpha$  et  $\beta$ , avec  $\alpha + \beta = \alpha\beta = 1$ ,  $\alpha^2 = \beta$  et  $\beta^2 = \alpha$ . Considérons dans l'espace vectoriel  $\mathbf{F}_4^3$  les deux vecteurs  $u = (\alpha, 1, 0)$  et  $v = (0, \alpha, 1)$ . Ils engendrent un code linéaire sur  $\mathbf{F}_4$ , de dimension 2, de longueur 3, formé des seize éléments  $au + bv$ , où  $a$  et  $b$  parcouruent  $\mathbf{F}_4$ , soient

$$\begin{array}{cccc} (0, 0, 0) & (\alpha, 1, 0) & (\beta, \alpha, 0) & (1, \beta, 0) \\ (0, \alpha, 1) & (\alpha, \beta, 1) & (\beta, 0, 1) & (1, 1, 1) \\ (0, \beta, \alpha) & (\alpha, \alpha, \alpha) & (\beta, 1, \alpha) & (1, 0, \alpha) \\ (0, 1, \beta) & (\alpha, 0, \beta) & (\beta, \beta, \beta) & (1, \alpha, \beta). \end{array}$$

Comme on le voit, la distance minimale de ce code est 2. Il est donc de paramètres  $(4; 3, 2, 2)$ . Par démultiplication, prenant  $\alpha$  et  $\beta$  comme éléments de base, 0 devient  $(0, 0)$ ,  $\alpha$  devient  $(1, 0)$ ,  $\beta$  devient  $(0, 1)$  et 1 devient  $(1, 1)$ . On obtient ainsi le code binaire linéaire de longueur 6 et de dimension 4 formé des seize éléments

$$\begin{array}{cccc} (0, 0, 0, 0, 0, 0) & (1, 0, 1, 1, 0, 0) & (0, 1, 1, 0, 0, 0) & (1, 1, 0, 1, 0, 0) \\ (0, 0, 1, 0, 1, 1) & (1, 0, 0, 1, 1, 1) & (0, 1, 0, 0, 1, 1) & (1, 1, 1, 1, 1, 1) \\ (0, 0, 0, 1, 1, 0) & (1, 0, 1, 0, 1, 0) & (0, 1, 1, 1, 1, 0) & (1, 1, 0, 0, 1, 0) \\ (0, 0, 1, 1, 0, 1) & (1, 0, 0, 0, 0, 1) & (0, 1, 0, 1, 0, 1) & (1, 1, 1, 0, 0, 1). \end{array}$$

Comme on le constate, la distance minimale n'a pas changé, et ce code est de paramètres  $(6, 4, 2)$ .

### 10.1.6. Extension paire

On généralise sans peine la notion de parité. On note  $\varepsilon : \mathbf{F}_q^n \rightarrow \mathbf{F}_q$  la forme linéaire « somme des coordonnées ». Un code linéaire  $C$  sur  $\mathbf{F}_q$  est dit *pair* si  $\varepsilon(\mathbf{m}) = 0$  pour tout  $\mathbf{m} \in C$ . L'extension paire d'un code  $C$  s'obtient à adjoignant à chaque mot de code  $\mathbf{m} \in C$  le coefficient supplémentaire  $-\varepsilon(\mathbf{m})$ .

**REMARQUE 10.4.** — Attention au piège. On a  $\varepsilon(m) = w(m) \bmod q$  pour  $q = 2$ . En revanche, dès qu'on a  $q > 2$ , il n'y a aucun rapport entre  $\varepsilon(\mathbf{m})$  et  $w(\mathbf{m})$ .

Par exemple, l'extension paire du code sur  $\mathbf{F}_4$  introduit précédemment est

$$\begin{array}{cccc} (0, 0, 0, 0) & (\alpha, 1, 0, \beta) & (\beta, \alpha, 0, 1) & (1, \beta, 0, \alpha) \\ (0, \alpha, 1, \beta) & (\alpha, \beta, 1, 0) & (\beta, 0, 1, \alpha) & (1, 1, 1, 1) \\ (0, \beta, \alpha, 1) & (\alpha, \alpha, \alpha, \alpha) & (\beta, 1, \alpha, 0) & (1, 0, \alpha, \beta) \\ (0, 1, \beta, \alpha) & (\alpha, 0, \beta, 1) & (\beta, \beta, \beta, \beta) & (1, \alpha, \beta, 0). \end{array}$$

Ce code est de paramètres  $(4; 4, 2, 3)$ . Le code démultiplié est de paramètres  $(8, 4, 4)$ .

**EXERCICE 10.8. [A]** — Vérifier.

**EXERCICE 10.9. [B]** — Ce code démultiplié est optimal.

### 10.1.7. Orthogonal

On étend de même la définition du code orthogonal  $C^\perp$  du code  $C$ , à partir du produit scalaire

$$\langle \mathbf{m} | \mathbf{n} \rangle = \sum_{i=1}^n m_i n_i \in \mathbf{F}_q = \varepsilon(\mathbf{m} \cdot \mathbf{n}),$$

où  $\mathbf{m} \cdot \mathbf{n}$  est le produit « coefficient par coefficient » de  $\mathbf{m}$  et  $\mathbf{n}$ . On a par exemple  $\langle \mathbf{1} | \mathbf{m} \rangle = \varepsilon(\mathbf{m})$ .

**EXERCICE 10.10. [A]** — Pour  $q > 2$ , on n'a plus identiquement  $\mathbf{m} \cdot \mathbf{m} = \mathbf{m}$ .

**EXERCICE 10.11. [B]** — Pour  $q = 2$  et  $q = 3$ , on a  $\langle \mathbf{m} | \mathbf{m} \rangle = w(\mathbf{m}) \bmod q$ .

## § 10.2. Codes de type MDS

### 10.2.1. La majoration de Singleton

Donnons une majoration élémentaire de la distance minimale, que nous avons rencontrée précédemment dans le cas binaire :

**PROPOSITION 10.5** (Majoration de Singleton). — Soit  $C$  un code linéaire non nul sur  $\mathbf{F}_q$ , de paramètres  $(q; n, k, d)$ . On a  $k + d \leq n + 1$ .

*Démonstration.* Considérons le sous-espace  $E = \mathbf{F}_q^{n-k+1}$  de  $\mathbf{F}_q^n$  formé des mots dont les  $k - 1$  dernières composantes sont nulles. Puisque  $\dim(C) + \dim(E) = n + 1 > n$ , il existe un élément non nul dans  $C \cap E$ . C'est un élément non nul  $\mathbf{m}$  de  $C$  tel que  $w(\mathbf{m}) \leq n + 1 - k$ , et on a par définition  $d \leq w(\mathbf{m})$ .  $\square$

**DÉFINITION 10.6.** — *On dit qu'un code linéaire est de type MDS<sup>1</sup> s'il est non nul et si l'égalité est atteinte dans la majoration de Singleton.*

Les paramètres d'un code de type MDS sont donc par définition de la forme  $(q; k + d - 1, k, d)$ . Par exemple, le code linéaire sur  $\mathbf{F}_4$  explicité dans 10.1.5 est de type MDS, de même que son extension paire. On notera au passage que le code démultiplié n'est pas de type MDS. D'ailleurs, il n'y a comme codes binaires de type MDS que les codes triviaux, comme le montrent les exercices 8.7 ou 10.14.

Au sens de la définition donnée pour les codes binaires en 8.1.1 et qui se généralise directement, les codes de type MDS sont évidemment optimaux.

### 10.2.2. Codes triviaux

Donnons comme exemples les codes dits « triviaux », qui sont de paramètres  $(q; n, n, 1)$ ,  $(q; n, n - 1, 2)$  ou  $(q; n, 1, n)$ , donc de type MDS.

Si on prend  $k = n$ , on a  $C = \mathbf{F}_q^n$  et  $d = 1$ ; les paramètres sont  $(q; n, n, 1)$ , il n'y a aucune redondance et aucune possibilité de corriger ou simplement de détecter les erreurs. C'est le code « plein ».

Les codes de paramètres  $(q; n, n - 1, 2)$  généralisent les codes de parité. Ce sont les *hyperplans* formé des  $\mathbf{m} \in \mathbf{F}_q^n$  tels que  $\sum \alpha_i m_i = 0$ , où les  $\alpha_i$  sont des scalaires fixés, *tous non nuls*. Ces codes sont de distance minimale égale à 2, ils permettent de détecter une erreur (mais pas de la corriger), ou de corriger un effacement.

À l'autre extrémité, si l'on prend  $k = 1$  et  $d = n$ , donc les paramètres  $(q; n, 1, n)$ , on obtient l'analogue des codes de répétition pure ; ce sont les codes de dimension 1, formé des multiples d'un même élément dont les composantes sont *toutes non nulles*.

### 10.2.3. Raccourcissement

Soit  $C \subset \mathbf{F}_q^n$  un code linéaire de paramètres  $(q; n, k, d)$ . Soit  $n'$  un entier avec  $1 \leq n' \leq n$ . Considérons le sous-espace de  $\mathbf{F}_q^n$  formé des mots dont toutes les composantes strictement au delà de la  $n'$ -ième sont nulles, sous-espace qui s'identifie naturellement à  $\mathbf{F}_q^{n'}$ . Alors  $C' = C \cap \mathbf{F}_q^{n'}$  est un code de longueur  $n'$  que l'on dit obtenu par *raccourcissement* de  $C$ . Explicitement,

---

1. Cet acronyme vient de l'anglais « *maximum distance separable* » ; on pourrait aussi l'interpréter en français comme « *majoration de Singleton* ».

il est formé des mots de  $C$  dont les  $n - n'$  derniers bits sont nuls, et que l'on a débarrassés de ces bits.

**PROPOSITION 10.7.** — Soit  $C$  un code linéaire sur  $\mathbf{F}_q$  de type MDS, donc de paramètres  $(q; n, k, d)$ , avec  $k + d = n + 1$ . Soit  $n'$  un entier avec  $d \leq n' \leq n$ . Notons  $C'$  le code de longueur  $n'$  obtenu par raccourcissement de  $C$ . Alors  $C'$  est de type MDS et de distance minimale  $d$ , donc de paramètres  $(q; n', k', d)$  avec  $k' = k + n' - n = n' - d + 1 \geq 1$ .

*Démonstration.* Notons  $k'$  la dimension de  $C'$ . Puisque la codimension de  $C'$  dans  $C$  est au plus égale à la codimension de  $\mathbf{F}_q^{n'}$  dans  $\mathbf{F}_q^n$ , on a  $k - k' \leq n - n'$ , soit  $k' \geq k + n' - n = n' - d + 1$ . On en particulier  $k' \geq 1$ , ce qui signifie que  $C'$  n'est pas nul. Tout élément de  $C'$  étant aussi un élément de  $C$ , la distance minimale  $d'$  de  $C'$  est  $\geq d$ . On a donc  $k' \geq n' - d + 1$  et  $d' \geq d$ . La majoration de Singleton donne une troisième inégalité  $k' + d' \leq n' + 1$ . Par conséquent, ces trois inégalités sont des égalités, ce qui implique la proposition.  $\square$

**COROLLAIRE 10.8.** — Soit  $C$  un code de type MDS, de longueur  $n$  et de distance minimale  $d$ . Pour tout choix de  $d$  des  $n$  coordonnées, il existe un mot de  $C$  dont les composantes non nulles sont exactement celles relatives à ces coordonnées.

*Démonstration.* Comme l'ordre des  $n$  coordonnées ne joue aucun rôle (un code de type MDS reste de type MDS par permutation des coordonnées), il suffit de faire la démonstration lorsqu'on choisit les  $d$  premières. Considérons alors le code raccourci à ces  $d$  coordonnées. D'après la proposition, il est de type  $(q; d, 1, d)$  et il suffit de considérer un mot non nul de ce code.  $\square$

**EXERCICE 10.12. [A]** — Vérifier dans les exemples donnés ci-dessus.

**EXERCICE 10.13. [C]** — Quel est le nombre des mots de  $C$  de poids  $d$  ?

**EXERCICE 10.14. [B]** — Déduire du corollaire que si  $q = 2$  alors le code est trivial (ce qu'on a déjà vu directement).

**EXERCICE 10.15. [B]** — Démontrer la réciproque du corollaire 10.8, à savoir qu'un code de distance minimale  $d$  qui satisfait à la conclusion est de type MDS.

**COROLLAIRE 10.9.** — L'orthogonal d'un code de type MDS est de type MDS.

*Démonstration.* Soit  $C$  un code de type MDS de paramètres  $(q; n, k, d)$ , avec  $d = n - k + 1$ . Son orthogonal  $C^\perp$  est de paramètres  $(q; n, n - k, \delta)$ . Il faut prouver qu'on a  $\delta = k + 1$  ou simplement, puisque  $\delta \leq n - (n - k) + 1 = k + 1$  d'après l'inégalité de Singleton, qu'on a  $\delta > k$ , autrement dit que  $C^\perp$  ne contient aucun mot non nul de poids  $\leq k$ .

L'ordre des coordonnées étant arbitraire, on peut supposer un tel mot de la forme  $\mathbf{r} = (r_0, \dots, r_{n-1})$ , avec  $r_k = \dots = r_{n-1} = 0$ . Mais on a  $d = n - k + 1$ , et d'après le corollaire 10.8, on peut trouver des mots de  $C$  de la forme  $\mathbf{m} = (m_0, \dots, m_{n-1})$  avec une seule des  $k$  premières composantes  $m_0, \dots, m_{k-1}$  non nulle, choisie de plus à la

place que l'on désire, disons  $i$ . On a alors  $0 = \langle \mathbf{m} | \mathbf{r} \rangle = m_i r_i$ , donc  $r_i = 0$ . Ainsi  $\mathbf{r}$  est nul, contrairement à l'hypothèse.  $\square$

**PROPOSITION 10.10.** — *Soit  $C$  un code de type MDS, de paramètres  $(q; n, k, d)$ . Si on a  $k \geq 2$ , alors  $d \leq q$ . Si l'on a  $k \leq n - 2$ , alors  $k + 1 \leq q$ .*

*Démonstration.* La seconde assertion se déduit de la première appliquée au code orthogonal (corollaire 10.9). Il suffit donc de démontrer la première. Supposons  $k \geq 2$ . Par raccourcissement (proposition 10.7),  $C$  donne naissance à un code  $C'$  de dimension 2, toujours de type MDS, donc de paramètres  $(q; d + 1, 2, d)$ . D'après le corollaire 10.8, il existe dans  $C'$  des mots

$$\mathbf{m} = (0, \alpha_1, \dots, \alpha_d), \quad \mathbf{n} = (\beta_0, 0, \beta_2, \dots, \beta_d).$$

Pour tout  $\lambda \in \mathbf{F}_q$ , le mot  $\mathbf{m} - \lambda \mathbf{n}$  doit être de poids  $\geq d$ , ce qui signifie qu'au plus *un seul* des coefficients  $\alpha_i - \lambda \beta_i$  peut être nul. Les  $d + 1$  rapports  $\alpha_i / \beta_i$  (éléments de  $\mathbf{F}_q$ , ou symbole  $+\infty$ ) doivent donc être deux à deux distincts. Ce qui impose  $q + 1 \leq d + 1$ , ce qu'on voulait démontrer.  $\square$

Le corps  $\mathbf{F}_q$  étant fixé, considérons des entiers  $n, k$  et  $d$  avec  $1 < k < n - 1$  (pour ne pas avoir de codes triviaux) et  $k + d = n + 1$ . Une question naturelle est la suivante : existe-t-il des codes de type MDS sur  $\mathbf{F}_q$  de paramètres  $(q; n, k, d)$  ?

Nous verrons ci-dessous (corollaire 10.12) qu'il en existe dès que  $n \leq q - 1$ . L'existence de codes MDS de longueur élevée reste une des grandes questions ouvertes. On sait construire des exemples avec  $n = q$  ou  $n = q + 1$ . *On ne connaît aucun code MDS de longueur  $n > q + 2$* . En fait, on ne connaît même que deux exemples de longueur  $q + 2$ , tous deux lorsque  $q$  est une puissance de 2, l'un de dimension 3, donc de paramètres  $(q; q + 2, 3, q)$ , et l'autre son orthogonal, donc de paramètres  $(q; q + 2, q - 1, 4)$ . On conjecture que, hormis ces exemples<sup>2</sup>, on a toujours  $n \leq q + 1$ .

#### 10.2.4. Première construction des codes de Reed-Solomon

Nous allons donner un exemple fondamental de code de type MDS. La définition en sera un peu abstraite. Nous reviendrons plus tard sur cet exemple et en donnerons une construction plus explicite.

Fixons le corps de base  $\mathbf{F}_q$  et numérotons-en de façon arbitraire les éléments non nuls, soient  $x_i$ , pour  $i = 0, \dots, q - 2$ . À tout polynôme  $u \in \mathbf{F}_q[X]$ , associons le mot

$$\mathbf{u} = (u(x_0), \dots, u(x_{q-2})) \in \mathbf{F}_q^{q-1}.$$

On obtient ainsi une application linéaire de  $\mathbf{F}_q[X]$  dans  $\mathbf{F}_q^{q-1}$ . Notons que  $w(\mathbf{u}) = q - 1 - z$ , où  $z$  est le nombre de racines de  $u$  dans  $\mathbf{F}_q^*$ . Si  $u$  n'est pas

2. Que l'on pourra trouver dans [MacWilliams, Sloane], ch. 11, fin du §5.

nul et est de degré  $< q - 1$ , alors on a  $z \leq \deg(u) < q - 1$ , donc  $\mathbf{u}$  n'est pas nul et on a

$$w(\mathbf{u}) \geq q - 1 - \deg(u).$$

Fixons un entier  $k$  avec  $0 < k < q$ . Soit  $C \subset \mathbb{F}_q^{q-1}$  le sous-ensemble formé des  $\mathbf{u}$ , lorsque  $u$  parcourt les polynômes de degré  $< k$ .

**PROPOSITION 10.11.** —  $C$  est un code linéaire sur  $\mathbb{F}_q$ , de paramètres  $(q; q - 1, k, q - k)$ , donc de type MDS.

*Démonstration.* Il est clair que  $C$  est un sous-espace vectoriel de  $\mathbb{F}_q^{q-1}$ . Comme  $\mathbf{u}$  ne peut être nul que si  $u$  est nul,  $C$  est bien de dimension  $k$ . On vient de voir que tout mot non nul est de poids  $> q - 1 - k$ . La distance minimale de  $d$  de  $C$  est donc  $\geq q - k$ , et on a  $k + d \geq q$ . Mais la majoration de Singleton s'écrit  $k + d \leq (q - 1) + 1 = q$ .  $\square$

On dit que  $C$  est le (un) *code de Reed-Solomon* de dimension  $k$  sur  $\mathbb{F}_q$ .

L'exemple le plus simple est celui du code de paramètres  $(4; 3, 2, 2)$ . Numérotons les trois éléments non nuls de  $\mathbb{F}_4$  dans l'ordre  $(1, \alpha, \beta)$ . Prenant  $u(X) = 1$ , on a  $\mathbf{u} = (1, 1, 1)$ ; pour  $u(X) = X$ , on obtient  $\mathbf{u} = (1, \alpha, \beta)$ . Le code est donc engendré par ces deux vecteurs et on retrouve l'exemple donné en 10.1.5.

Il existe donc des codes de type MDS de paramètres  $(q; q - 1, k, q - k)$ . Appliquant la proposition 10.7, on obtient aussitôt :

**COROLLAIRE 10.12.** — *On peut trouver des codes (de type MDS) avec comme paramètres  $(q; n, k, n - k + 1)$  dès que  $1 \leq k \leq n \leq q - 1$ .*

**EXERCICE 10.16. [B]** — Au lieu d'énumérer par les  $x_i$  la totalité de  $\mathbb{F}_q^*$ , on n'en prend qu'une partie, à  $n$  éléments. Montrer que l'on obtient ainsi un code de type MDS, de paramètres  $(q; n, k, n - k + 1)$ .

### § 10.3. Codes cycliques

Les plus importants des codes linéaires sont les codes *cycliques*.

#### 10.3.1. Définitions

Pour tout mot  $\mathbf{m} = (m_0, \dots, m_{n-1}) \in \mathbb{F}_q^n$ , notons

$$\sigma(\mathbf{m}) = (m_{n-1}, m_0, \dots, m_{n-2})$$

le mot déduit de  $\mathbf{m}$  par décalage circulaire à droite. Par itération de cette opération, on obtient les mots décalés successifs à droite depuis  $\sigma^2(\mathbf{m})$  jusqu'à  $\sigma^{n-2}(\mathbf{m})$ , et enfin  $\sigma^{n-1}(\mathbf{m}) = (m_1, \dots, m_{n-1}, m_0)$  qui est le mot décalé circulairement à gauche de  $\mathbf{m}$ . On a  $\sigma^n(\mathbf{m}) = \mathbf{m}$ .

**DÉFINITION 10.13.** — *On dit que le code linéaire  $C \subset \mathbf{F}_q^n$  de longueur  $n$  est cyclique si pour tout mot  $\mathbf{m}$  du code  $C$ , le mot  $\sigma(\mathbf{m})$  déduit par décalage circulaire est encore un mot du code.*

Par itération, on voit que  $C$  contient tous les décalés circulaires de  $\mathbf{m}$ . Un code linéaire cyclique est donc un code linéaire stable par tous les décalages circulaires (ou en d'autres termes dont le groupe des automorphismes contient les décalages circulaires).

Pour tout mot  $\mathbf{m} \in \mathbf{F}_q^n$ , il existe un plus petit code linéaire cyclique qui le contient : c'est le sous-espace vectoriel  $C$  engendré par les  $n$  décalés circulaires de  $\mathbf{m}$ , qui est formé par définition des combinaisons linéaires  $\sum_{0 \leq i < n} \lambda_i \sigma^i(\mathbf{m})$ . On dit que  $C$  est le code linéaire cyclique *engendré* par  $\mathbf{m}$ , ou que  $\mathbf{m}$  est un *générateur* du code cyclique  $C$ .

**EXERCICE 10.17. [B]** — Parmi les codes triviaux introduits en 10.2.2, lesquels sont cycliques ?

### 10.3.2. Représentation des mots par des polynômes

Il sera essentiel de passer à une autre représentation des mots. L'intérêt des codes linéaires cycliques est en effet qu'ils bénéficient d'une description algébrique particulièrement simple en termes de polynômes. Faisons correspondre à chaque mot  $\mathbf{m} = (m_0, \dots, m_{n-1})$  de  $\mathbf{F}_q^n$  le polynôme  $m(X) = m_0 + m_1 X + \dots + m_{n-1} X^{n-1} \in \mathbf{F}_q[X]$ . Cette opération est évidemment linéaire : elle respecte l'addition et la multiplication par un scalaire. Notons  $m^\sigma(X)$  le polynôme correspondant au vecteur décalé  $\sigma(\mathbf{m})$ . Un calcul immédiat donne  $Xm(X) = m_{n-1}(X^n - 1) + m^\sigma(X)$ . On reconnaît là une division euclidienne par  $X^n - 1$ , de quotient  $m_{n-1}$ , de reste  $m^\sigma(X)$ . En termes de polynômes, le décalage s'exprime donc par

$$m^\sigma(X) \equiv Xm(X) [\text{mod } (X^n - 1)].$$

Si on calcule « modulo  $X^n - 1$  », le décalage des mots correspond ainsi à la multiplication des polynômes par  $X$ .

**REMARQUE 10.14.** — On notera au passage qu'un décalage « ordinaire », sans report à gauche du bit qui « sort à droite » correspondrait à un calcul modulo  $X^n$ . De même, si le bit qui « sort à droite » était reporté à gauche après multiplication par  $\alpha \in \mathbf{F}_q$ , on aurait un calcul modulo  $X^n - \alpha$ .

**REMARQUE 10.15.** — On voit ici apparaître l'identification naturelle entre le  $\mathbf{F}_q$ -espace vectoriel  $\mathbf{F}_q^n$  et l'*anneau quotient*  $\mathbf{F}_q[X]/(X^n - 1)$ . Dans cette identification, le décalage circulaire correspond à la multiplication par  $X$ , les codes linéaires cycliques de longueur  $n$  correspondent aux sous-espaces stables par la multiplication par  $X$ , c'est-à-dire aux *idéaux* de l'anneau quotient. Or ces derniers correspondent par image réciproque aux idéaux de l'anneau  $\mathbf{F}_q[X]$  contenant  $X^n - 1$ . Et, comme on le sait en

algèbre, ceux-ci correspondent à leur tour aux diviseurs (unitaires) de  $X^n - 1$ . En définitive, les codes linéaires cycliques de longueur  $n$  sur  $\mathbf{F}_q$  correspondent bijectivement aux polynômes unitaires à coefficients dans  $\mathbf{F}_q$  qui divisent  $X^n - 1$ . Vu l'importance de cette correspondance, nous en donnons ci-dessous une construction directe.

### 10.3.3. Générateurs minimaux des codes cycliques

THÉORÈME 10.16. — a) Soit

$$g(X) = a_0 + \cdots + a_{n-k}X^{n-k}, \quad a_{n-k} = 1,$$

un diviseur unitaire de  $X^n - 1$  dans  $\mathbf{F}_q[X]$  et soit

$$\mathbf{m} = (a_0, \dots, a_{n-k-1}, 1, 0, \dots, 0) \in \mathbf{F}_q^n$$

le mot correspondant. Alors les  $k$  mots de longueur  $n$

$$\mathbf{m} = (a_0, \dots, a_{n-k-1}, 1, 0, \dots, 0),$$

$$\sigma(\mathbf{m}) = (0, a_0, \dots, a_{n-k-1}, 1, 0, \dots, 0),$$

...

$$\sigma^{k-1}(\mathbf{m}) = (0, \dots, 0, a_0, \dots, a_{n-k-1}, 1)$$

forment une base d'un code cyclique sur  $\mathbf{F}_q$  de dimension  $k$ .

b) Tout code cyclique  $C$  de longueur  $n$  sur  $\mathbf{F}_q$  s'obtient par la construction précédente, et le polynôme  $g$  est uniquement déterminé par  $C$ .

*Démonstration.* Considérons d'abord a). Les  $k$  mots considérés sont linéairement indépendants, puisque la matrice de leurs coefficients contient une matrice triangulaire d'ordre  $k$  à diagonale unité. Il s'agit de prouver que le sous-espace  $C$  qu'ils engendrent est stable par le décalage  $\sigma$ . Mais, puisque

$$\sigma(c_0\mathbf{m} + \cdots + c_{k-1}\sigma^{k-1}(\mathbf{m})) = c_0\sigma(\mathbf{m}) + \cdots + c_{k-1}\sigma^k(\mathbf{m}),$$

il suffit de prouver qu'on a  $\sigma^k(\mathbf{m}) \in C$ . Or, puisque  $g$  divise  $X^n - 1$ , on peut écrire

$$X^n - 1 = h(X)g(X) = (b_0 + \cdots + b_{k-1}X^{k-1} + X^k)g(X),$$

ce qui donne

$$X^k g(X) \equiv -b_0 g(X) - \cdots - b_{k-1} X^{k-1} g(X) \pmod{(X^n - 1)},$$

soit encore

$$\sigma^k(\mathbf{m}) = -b_0\mathbf{m} - \cdots - b_{k-1}\sigma^{k-1}(\mathbf{m}) \in C.$$

Ainsi,  $C$  est bien cyclique et on a prouvé a).

Soit inversement  $C$  un code cyclique. Parmi tous les mots de  $C$ , choisissons-en un qui ait le nombre maximum de zéros consécutifs « à la fin », soit

$$\mathbf{m} = (a_0, \dots, a_{n-k}, 0, \dots, 0) \text{ avec } a_{n-k} \neq 0.$$

Quitte à multiplier  $\mathbf{m}$  par l'inverse de  $a_{n-k}$ , on peut supposer que  $a_{n-k} = 1$ . On peut noter au passage que ce mot  $\mathbf{m}$  est uniquement déterminé, car la différence de deux tels mots aurait au moins un zéro de plus à droite. Puisque le code est cyclique, les  $k-1$  mots déduits de  $\mathbf{m}$  par décalages successifs à droite appartiennent eux aussi à  $C$ . Les  $k$  mots ainsi construits sont évidemment linéairement indépendants. Il reste à prouver, d'une part qu'ils engendrent bien  $C$ , et d'autre part que le polynôme  $g(X)$  correspondant à  $\mathbf{m}$  divise  $X^n - 1$ .

Soit  $\mathbf{n}$  un mot quelconque, correspondant au polynôme  $n(X)$ . Par division euclidienne dans  $\mathbb{F}_q[X]$ , on peut écrire

$$n(X) = (b_0 + \dots + b_{k-1}X^{k-1})g(X) + r(X)$$

où le reste  $r(X)$  est de degré  $< n-k$ . Cela donne une relation  $\mathbf{n} = (b_0\mathbf{m} + b_1\sigma(\mathbf{m}) + \dots) + \mathbf{r}$ . Si  $\mathbf{n}$  appartient à  $C$ , on voit par différence que  $\mathbf{r}$  appartient à  $C$ . Mais cela implique  $\mathbf{r} = 0$ , puisqu'il a trop de zéros à droite. Les mots proposés engendrent donc bien  $C$ . Effectuons maintenant un décalage à droite supplémentaire, ce qui donne le mot  $(a_{n-k}, 0, \dots, 0, a_0, \dots)$ , qui appartient à  $C$  puisque celui-ci est supposé être cyclique. Le polynôme correspondant vaut  $X^k g(X) - (X^n - 1)$  (ne pas oublier que  $a_{n-k} = 1$ ). Lui appliquant ce qui précède, on trouve une relation  $X^k g(X) - (X^n - 1) = (b_0 + \dots + b_{k-1}X^{k-1})g(X)$ , soit

$$X^n - 1 = (-b_0 - \dots - b_{k-1}X^{k-1} + X^k)g(X) = h(X)g(X).$$

Cela démontre que  $g(x)$  divise bien  $X^n - 1$  et achève la démonstration.  $\square$

**DÉFINITION 10.17.** — *Avec les notations précédentes, on dit que le polynôme  $g$  est le générateur unitaire minimal — ou simplement, s'il n'y a pas d'ambiguïté possible, le générateur — du code cyclique  $C$ .*

Fixons une notation provisoire. Pour chaque entier  $r$ , notons  $E_r$  le sous-espace vectoriel de dimension  $r$  de  $\mathbb{F}_q[X]$ , formé des polynômes de degré  $< r$ , que l'on munit de sa base naturelle formée des monômes et que l'on identifie ainsi à  $\mathbb{F}_q^r$ . Les éléments de  $E_r$  sont donc les polynômes  $a_0 + a_1X + \dots + a_{r-1}X^{r-1}$ , identifiés aux mots  $(a_0, \dots, a_{r-1}) \in \mathbb{F}_q^r$ .

Nous allons donc construire le code comme sous-espace vectoriel de  $E_n$ . On a

$$X^n - 1 = g(X)h(X), \quad \deg(g) = n - k, \quad \deg(h) = k.$$

L'application  $a(X) \mapsto a(X)g(X)$  de  $E_k$  dans  $E_n$  est injective. Son image est le code cyclique  $C$  considéré. Notons au passage que  $C$  est aussi le noyau de l'application  $m(X) \mapsto m(X)h(X) \bmod (X^n - 1)$ . On retrouve des définitions en termes de matrices génératrice ou vérificatrice.

**EXERCICE 10.18. [B]** — Montrer que les codes binaires suivants sont cycliques et en donner les polynômes générateurs : code de parité, code de répétition, code de Hamming  $H_3$ , code simplexe  $S_3$ .

### 10.3.4. Codages pour un code cyclique

Gardons les notations précédentes. Une application de codage pour  $C$  est par définition une application linéaire de  $\mathbf{F}_q^k$  dans  $\mathbf{F}_q^n$  dont l'image est  $C$  ou encore, avec la représentation polynomiale introduite, une application linéaire de  $E_k$  dans  $E_n$  d'image  $C$ . Une solution naïve est de prendre l'application  $a(X) \mapsto a(X)g(X)$  introduite ci-dessus, mais elle implique d'effectuer une division pour décoder, même en l'absence d'erreur.

Il y a plus astucieux. Pour chaque polynôme  $a(X) \in E_k$ , effectuons la division euclidienne de  $X^{n-k}a(X)$  par  $g(X)$  :

$$X^{n-k}a(X) = u(X)g(X) + r(X), \quad \deg(r) < n - k.$$

Alors  $-r(X) + X^{n-k}a(X)$  est multiple de  $g(X)$ , donc appartient à  $C$ . Dans la représentation initiale en termes de mots, c'est le résultat  $(-r, a)$  de l'adjonction du contrôle  $-r$  en tête du message initial  $a$ .

### 10.3.5. Constructions élémentaires

Montrons d'abord comment s'interprètent pour les codes cycliques trois constructions élémentaires. On garde les notations précédentes.

#### Raccourcissement

Soit  $n'$  un entier avec  $1 \leq n' \leq n$ . Par définition même de l'opération de raccourcissement (10.2.3), le code raccourci de longueur  $n'$  est formé des polynômes de degré  $< n'$  multiples de  $g$ , c'est-à-dire des polynômes de la forme  $a(X)g(X)$ , où  $a(X)$  parcourt l'espace  $E_{k-n+n'}$  des polynômes de degré  $< k - n + n'$ . Ce code est donc nul si  $n' < n - k$  et de dimension  $k - n + n'$  sinon. On notera qu'il n'a aucune raison d'être cyclique. On notera aussi que les deux applications de codage introduites ci-dessus fonctionnent également pour les codes raccourcis.

**EXERCICE 10.19. [B] — Vérifier.**

#### Extension paire

La somme des coefficients du mot correspondant au polynôme  $m(X)$  est tout simplement  $m(1) = a(1)g(1)$ . Par exemple, dire que  $C$  est pair signifie que  $g(1) = 0$ . De manière générale, l'extension paire  $\bar{C}$  est formée des couples  $(g(X)a(X), -a(1)g(1)) \in \mathbf{F}_q^n \times \mathbf{F}_q = \mathbf{F}_q^{n+1}$ .

#### Orthogonal

Écrivons  $X^n - 1 = gh$ , de sorte que  $\deg h = k$ . Considérons le polynôme  $\tilde{h}$  réciproque de  $h$  :

$$\tilde{h}(X) = X^k h(1/X)$$

(on renverse simplement la suite des coefficients).

**PROPOSITION 10.18.** — *Le code orthogonal de C est cyclique, et possède comme générateur le polynôme  $\tilde{h}$  réciproque de h.*

*Démonstration.* Notons  $C'$  le code de générateur  $\tilde{h}$ . Comme

$$\dim C + \dim C' = k + (n - \deg \tilde{h}) = k + (n - \deg h) = k + (n - k) = n,$$

il suffit de prouver que tout mot de C est orthogonal à tout mot de  $C'$ . Considérons donc deux tels mots, soient  $m(X) = a(X)g(X) \in C$  et  $r(X) = b(X)\tilde{h}(X) \in C'$ , où  $a$  est de degré  $\leq k - 1$  et  $b$  de degré  $\leq n - k - 1$ . On a par définition  $\langle m|r \rangle = \sum_{0 \leq i \leq n-1} m_i r_i$ . Introduisons les polynômes réciproques  $\tilde{r}$  et  $\tilde{b}$ , définis par

$$\begin{aligned}\tilde{r}(X) &= X^{n-1}r(1/X), \\ \tilde{b}(X) &= X^{n-k-1}b(1/X),\end{aligned}$$

de sorte qu'on a  $\tilde{r}(X) = \tilde{b}(X)h(X)$ . On a

$$m(X)\tilde{r}(X) = \left(\sum_{0 \leq i \leq n-1} m_i X^i\right)\left(\sum_{0 \leq j \leq n-1} r_j X^{n-1-j}\right) = \sum_{0 \leq i, j \leq n-1} m_i r_j X^{n-1+i-j},$$

de sorte que la somme  $\sum_{0 \leq i \leq n-1} m_i r_i$  que l'on doit calculer est justement le coefficient de  $X^{n-1}$  dans le produit  $m(X)\tilde{r}(X)$ . Mais, comme on a

$$m(X)\tilde{r}(X) = a(X)g(X)\tilde{b}(X)h(X) = a(X)\tilde{b}(X)(X^n - 1),$$

avec  $a(X)\tilde{b}(X)$  de degré  $\leq n - 2$ , le terme de degré  $n - 1$  de ce produit est nul. On a donc  $\langle m|r \rangle = 0$ , ce qu'on voulait démontrer.  $\square$

#### § 10.4. Classes cyclotomiques ( $n$ premier à $q$ )

Dans toute ce paragraphe, nous faisons l'hypothèse supplémentaire que  $n$  et  $q$  sont premiers entre eux. Par exemple, dans le cas des codes sur les corps  $\mathbf{F}_{2^s}$ , cela signifie que nous supposons que la longueur du code considéré est impaire.

On fixe le corps  $\mathbf{F}_q$ , l'entier  $n$  et une racine primitive  $n$ -ième de l'unité  $\alpha$  dans un corps fini  $K = \mathbf{F}_{q^r}$  contenant  $\mathbf{F}_q$ . Pour qu'un tel  $\alpha$  existe, il faut et il suffit que  $n$  divise  $q^r - 1$ , donc qu'on ait  $q^r \equiv 1 \pmod{n}$ . On peut prendre pour  $r$  le plus petit entier ayant cette propriété, c'est-à-dire l'ordre de l'élément  $q$  dans le groupe multiplicatif  $(\mathbf{Z}/n\mathbf{Z})^*$ . On sait alors que  $\alpha$  est de degré  $r$  sur  $\mathbf{F}_q$  (proposition 9.28), et on note  $g_1$  son polynôme minimal.

Concrètement, cela signifie qu'on choisit un facteur irréductible  $g_1$  du polynôme  $\Phi_n$  dans  $\mathbf{F}_q[X]$  et qu'on définit  $K = \mathbf{F}_{q^r}$  en adjoignant à  $\mathbf{F}_q$  une racine  $\alpha$  de  $g_1$ .

### 10.4.1. Diviseurs de $X^n - 1$

Nous nous proposons maintenant de déterminer les diviseurs de  $X^n - 1$  dans l'anneau  $\mathbf{F}_q[X]$ . On peut écrire dans  $K[X]$  :

$$X^n - 1 = \prod_{i=0}^{n-1} (X - \alpha^i) = \prod_{i \in \mathbf{Z}/n\mathbf{Z}} (X - \alpha^i).$$

Tout diviseur de  $X^n - 1$  dans  $K[X]$  s'écrira donc sous la forme

$$g_\Sigma(X) = \prod_{i \in \Sigma} (X - \alpha^i),$$

où  $\Sigma$  est une partie convenable de  $\mathbf{Z}/n\mathbf{Z}$ .

**PROPOSITION 10.19.** — *Pour que  $g_\Sigma$  soit à coefficients dans  $\mathbf{F}_q$ , il faut et il suffit que  $\Sigma$  soit stable par multiplication par  $q$  (modulo  $n$ ).*

*Démonstration.* Écrivons pour simplifier  $g$  au lieu de  $g_\Sigma$ . Si  $g$  est à coefficients dans  $\mathbf{F}_q$ , chaque coefficient de  $g$  est stable par élévation à la puissance  $q$ -ième. Puisque cette opération respecte aussi l'addition, on voit que  $g(X^q) = (g(X))^q$ ; par conséquent, l'ensemble des racines de  $g$  est stable par passage à la puissance  $q$ -ième, ce qui signifie que  $\Sigma$  est stable par multiplication par  $q$ . Inversement, si  $\Sigma$  est stable par multiplication par  $q$ , on vérifie aussitôt que  $g(X^q) = (g(X))^q$ , et cela implique que les coefficients de  $g$  appartiennent à  $\mathbf{F}_q$ .  $\square$

**EXERCICE 10.20. [B]** — Compléter les détails de cette démonstration.

### 10.4.2. Classes cyclotomiques

Il s'agit donc maintenant de déterminer les parties  $\Sigma$  de  $\mathbf{Z}/n\mathbf{Z}$  stables par multiplication par  $q$  modulo  $n$ . Or la relation « il existe  $i$  avec  $q^i j \equiv j'$  » entre éléments  $j$  et  $j'$  de  $\mathbf{Z}/n\mathbf{Z}$  est une relation d'équivalence.

**EXERCICE 10.21. [A]** — Montrer qu'il s'agit bien d'une relation d'équivalence.

Les classes d'équivalence sont appelées *classes cyclotomiques*. Comme on dit en théorie des groupes, les classes cyclotomiques sont donc les *orbites* de la multiplication par  $q$  dans  $\mathbf{Z}/n\mathbf{Z}$ . Pour chaque élément  $j \bmod n$  de  $\mathbf{Z}/n\mathbf{Z}$ , notons  $\Sigma_j$  sa classe cyclotomique. Elle s'obtient comme suit : on considère le plus petit entier  $s > 0$  tel que  $q^s j \equiv j \pmod{n}$ , et on prend  $\Sigma_j = \{j, qj, \dots, q^{s-1}j\}$ . On a  $\#\Sigma_j = s$ . Notons que  $s$  est au plus égal à l'ordre  $r$  de  $q$  modulo  $n$ , ordre qui est aussi, comme on l'a vu plus haut, le degré sur  $\mathbf{F}_q$  de l'élément  $\alpha$ .

**EXERCICE 10.22. [B]** — L'entier  $s$  divise  $r$ .

D'après la proposition 9.7 (étendue au cas du corps  $\mathbf{F}_q$ ), le polynôme correspondant à la partie  $\Sigma_j$  est le polynôme minimal  $g_j$  de la puissance  $\alpha^j$  de  $\alpha$ . Les différents  $g_j$  sont les facteurs irréductibles de  $X^n - 1$  dans  $\mathbf{F}_q[X]$ , et la décomposition de  $\mathbf{Z}/n\mathbf{Z}$  comme réunion des classes cyclotomiques correspond à la décomposition de  $X^n - 1$  en facteurs irréductibles dans  $\mathbf{F}_q[X]$ . Les parties stables  $\Sigma$  sont exactement les réunions de classes cyclotomiques, de la même façon que les diviseurs de  $X^n - 1$  sont exactement les produits des facteurs irréductibles.

### Racines d'un code cyclique

Revenons aux codes. La détermination des codes cycliques de longueur  $n$  sur  $\mathbf{F}_q$ , que nous avons ramenée dans un premier temps à celle des facteurs de  $X^n - 1$  dans  $\mathbf{F}_q[X]$ , revient donc en définitive (dans le cas où  $n$  est premier à  $q$ , et lorsqu'on a fixé une racine primitive de l'unité) à celle des parties stables par multiplication par  $q$  dans  $\mathbf{Z}/n\mathbf{Z}$ , parties qui sont exactement les réunions de classes cyclotomiques. Si on a choisi une telle partie stable  $\Sigma$ , le code correspondant est le sous-espace vectoriel de l'espace des polynômes de  $\mathbf{F}_q[X]$  de degré  $< n$ , formé des polynômes  $m$  qui sont multiples de  $g_\Sigma$ , c'est-à-dire qui sont tels que  $m(\alpha^i) = 0$  pour tout  $i \in \Sigma$ . Ces  $\alpha^i$ , qui sont les racines communes de tous les polynômes éléments du code, s'appellent les *racines du code*.

Notons que pour un polynôme  $m \in \mathbf{F}_q[X]$ , il est équivalent de dire qu'il est multiple de  $g_j$ , ou que  $m(\alpha^j) = 0$ , ou que  $m(\alpha^i) = 0$  pour tout  $i \in \Sigma_j$ . Il suffit donc de prendre *une* condition  $m(\alpha^j) = 0$  par classe cyclotomique contenue dans  $\mathbf{Z}$ .

**REMARQUE 10.20.** — On pourra s'étonner qu'on puisse remplacer plusieurs conditions linéaires par une seule. Mais la condition  $m(\alpha^j) = 0$  est à coefficients dans  $K$  et non dans  $\mathbf{F}_q$ , donc s'exprime par  $r$  conditions à coefficients dans  $\mathbf{F}_q$ , et en fait exactement par  $\#\Sigma_j$  conditions linéairement indépendantes. On ne viole donc pas l'algèbre linéaire sur  $\mathbf{F}_q$ .

#### 10.4.3. Exemples

Les quatre parties stables évidentes (vide, pleine, réduite à 0, complémentaire de la précédente) donnent des codes triviaux (voir 10.2.2).

**EXERCICE 10.23. [B]** — Vérifier.

**EXERCICE 10.24. [B]** — On prend  $q = 2$  et  $n = 5$ . Que trouve-t-on comme codes possibles ?

Donnons un exemple simple, où nous retrouverons le code binaire de Hamming de longueur 7. On prend  $q = 2$  et  $n = 2^3 - 1 = 7$ . On peut donc prendre pour  $\alpha$  un générateur de  $\mathbf{F}_8^*$ . Par multiplication par 2 modulo 7, on

obtient les cycles  $1 \mapsto 2 \mapsto 4 \mapsto 1$  et  $3 \mapsto 6 \mapsto 5 \mapsto 3$ , ce qui donne les deux classes cyclotomiques  $\{1, 2, 4\}$  et  $\{3, 5, 6\}$  auxquelles il faut adjoindre  $\{0\}$ . On obtient ainsi la décomposition  $X^7 - 1 = (X - 1)g(X)h(X)$  avec

$$g(X) = (X - \alpha)(X - \alpha^2)(X - \alpha^4), \quad h(X) = (X - \alpha^3)(X - \alpha^5)(X - \alpha^6).$$

En fait,  $g$  et  $h$  sont les deux polynômes irréductibles primitifs de degré 3 sur  $\mathbf{F}_2$ . L'un est  $1 + X + X^3$ , l'autre  $1 + X^2 + X^3$ , correspondant aux vecteurs  $[1101000]$  et  $[1011000]$ . Le premier donne le code défini dans 6.5, le second donne un code isomorphe.

**EXERCICE 10.25. [C]** — Donner un isomorphisme.

**EXERCICE 10.26. [C]** — Comme on le laisse entendre ci-dessus, on ne sait pas lequel des deux polynômes  $g$  et  $h$  est  $1 + X + X^3$ , et lequel est l'autre. À quoi est due cette incertitude ?

**EXERCICE 10.27. [B]** — Si  $C$  est défini par la partie  $\Sigma$ , à quelle partie correspond le code orthogonal ?

#### 10.4.4. Distance minimale des codes cycliques

Supposons toujours  $n$  premier à  $q$  et fixons une racine primitive  $n$ -ième de l'unité  $\alpha$  dans un corps  $K$  convenable. Si l'on se donne une partie  $\Sigma$  de  $\mathbf{Z}/n\mathbf{Z}$ , réunion de classes cyclotomiques, le code  $C$  correspondant est formé par définition des polynômes  $m \in \mathbf{F}_q[X]$ , de degré  $< n$ , qui possèdent comme racines tous les  $\alpha^i$ , pour  $i \in \Sigma$ .

Dire que la distance minimale de ce code est  $\geq d$ , c'est dire que si un tel polynôme n'est pas nul, il possède au moins  $d$  coefficients non nuls. Cette distance minimale ne dépend que de la donnée des entiers  $q$  et  $n$  et de la partie  $\Sigma$ . Mais on ne connaît pas d'algorithme général pour la calculer et sa détermination dans les cas concrets peut être extrêmement difficile. On dispose cependant d'une minoration élémentaire :

**PROPOSITION 10.21.** — *S'il existe des entiers  $t$  et  $s > 0$  tels que  $\Sigma$  contienne les  $s$  entiers consécutifs  $t, t + 1, \dots, t + s - 1$ , alors la distance minimale du code associé est  $\geq s + 1$ .*

*Démonstration.* Il s'agit de démontrer qu'un polynôme  $m \in \mathbf{F}_q[X]$ , de degré  $< n$ , qui a au plus  $s$  termes non nuls et qui satisfait aux conditions  $m(\alpha^{t+i}) = 0$  pour  $0 \leq i < s$ , est identiquement nul. On est ainsi ramené au lemme suivant :  $\square$

**LEMME 10.22.** — *Soient  $K$  un corps,  $n > 1$  un entier,  $\alpha$  une racine primitive  $n$ -ième de l'unité dans  $K$  et  $m \in K[X]$  un polynôme de degré  $< n$ . Soient  $s > 0$  et  $t$  deux entiers tels que  $m(\alpha^{t+i}) = 0$  pour  $0 \leq i < s$ . Alors si  $m$  n'est pas nul, il possède au moins  $s + 1$  termes non nuls.*

*Démonstration.* Supposons que  $m$  possède au plus  $s$  termes non nuls. Soient  $d_1, \dots, d_s$  des entiers avec  $0 \leq d_1 < \dots < d_s < n$ , parmi lesquels se trouvent tous les degrés des termes non nuls de  $m$ . Il existe donc  $\lambda_1, \dots, \lambda_s$  dans  $K$  avec  $m(X) = \sum_{1 \leq j \leq s} \lambda_j X^{d_j}$ . Posons  $a_j = \alpha^{d_j}$ . Alors  $m(\alpha^{t+i})$  peut s'écrire sous la forme

$$\sum_{1 \leq j \leq s} \lambda_j (a_j)^{t+i},$$

de sorte que les  $\mu_j = (a_j)^t \lambda_j$  forment une solution du système linéaire

$$\sum_{1 \leq j \leq s} \mu_j (a_j)^i = 0.$$

Or, la matrice carrée formée des  $(a_j)^i$  est une matrice de Vandermonde déduite des éléments  $a_j$ . Puisque  $\alpha$  est une racine primitive  $n$ -ième de l'unité, les  $a_j$  sont deux à deux distincts, et la matrice est inversible. Par conséquent, les  $\mu_j$ , donc aussi les  $\lambda_j$  sont tous nuls, et le polynôme  $m$  est nul.  $\square$

Ainsi par exemple, le code de Hamming précédent est de distance minimale  $> 2$ , puisque la partie  $\Sigma$  correspondante contient  $\{1, 2\}$  (ou  $\{5, 6\}$ ).

#### 10.4.5. Idempotents des codes cycliques

Revenons un peu en arrière, à la notion de générateur. Dire qu'un polynôme  $f(X)$  est un générateur du code cyclique  $C$  engendré par  $g$  signifie, d'abord que  $f$  appartient à  $C$ , donc peut s'écrire  $f(X) = a(X)g(X)$  (avec si l'on veut  $\deg(a) < k$ ), et d'autre part que tout élément de  $C$  — et il suffit de le vérifier pour  $g$  — est somme de translatés circulaires de  $f$ . Mais cela signifie que l'on peut écrire  $g(X) \equiv c(X)f(X) \pmod{X^n - 1}$  pour un polynôme  $c$  convenable.

Reprenons maintenant notre situation de base, avec toujours  $n$  premier à  $q$ . Alors les racines de  $X^n - 1$  sont simples, ce qui implique que l'on a  $X^n - 1 = g(X)h(X)$ , avec  $h$  premier à  $g$ . On peut donc trouver (identité de Bézout, proposition 1.44) des polynômes  $c(X)$  et  $d(X)$  avec  $cg + dh = 1$ ,  $\deg(c) < \deg(h)$ ,  $\deg(d) < \deg(g)$ . Considérons alors le polynôme  $e = cg$ . Il appartient au code. Puisqu'on a par définition  $gh \equiv 0 \pmod{X^n - 1}$ , on a  $eh \equiv 0$ , donc  $e = e(cg + dh) \equiv ecg = e^2$ , ce qu'on exprime en disant que  $e$  est *idempotent* (égal à son carré) modulo  $X^n - 1$ .

De plus,  $g$  est multiple de  $e$  modulo  $X^n - 1$  puisqu'on a  $g = g(cg + dh) \equiv gcg = ge$ . Ainsi  $e$  est aussi un générateur de  $C$ . On peut donc remplacer le générateur minimal  $g$  par le générateur idempotent  $e$ . On gagne l'avantage d'avoir  $e^2 \equiv e \pmod{X^n - 1}$ , mais on perd celui que  $g$  divise  $X^n - 1$ .

**EXERCICE 10.28. [B]** — Montrer qu'on peut caractériser les éléments  $m$  de  $C$  par la propriété  $m \equiv me \pmod{X^n - 1}$ .

EXERCICE 10.29. [B] — Montrer que  $e$  est uniquement déterminé, modulo  $X^n - 1$ .

On notera que si l'on connaît  $g$ , on calcule  $e = cg$  par l'algorithme d'Euclide étendu (voir 1.5.3). Inversement, si on connaît  $e$ , on peut récupérer  $g$  par la relation

$$g = \text{pgcd}(e, X^n - 1),$$

qui résulte de ce que  $e$  est premier à  $h$  (puisque  $e + dh = 1$ ).

EXERCICE 10.30. [B] — Si  $e$  est le générateur idempotent de  $C$ , quel est celui de  $C^\perp$  ?



# Chapitre II

## Codes BCH

Nous avons décrit au chapitre précédent la construction générale des codes cycliques de longueur  $n$  sur un corps  $\mathbf{F}_q$ , si l'on suppose  $n$  premier à  $q$ . Rappelons qu'elle fait intervenir les racines  $n$ -ièmes de l'unité sur  $\mathbf{F}_q$ , et que celles-ci engendrent le corps  $\mathbf{R}_n(\mathbf{F}_q) = \mathbf{F}_{q^r}$ , où  $r$  est l'ordre de  $q$  modulo  $n$ , donc le plus petit entier tel que  $q^r \equiv 1 \pmod{n}$ .

La situation particulière dans chaque cas dépend essentiellement des propriétés de la multiplication par  $q$  modulo  $n$  et d'abord de son ordre  $r$ . Cet ordre est déterminé par  $q$  et  $n$ . Inversement, si l'on se donne  $q$  et  $r$ , les valeurs possibles pour  $n$  sont les diviseurs de  $q^r - 1$ .

Les cas les plus simples s'obtiennent lorsqu'on prend  $n$  maximal pour  $q$  et  $r$  donnés, soit  $n = q^r - 1$ . Si on prend de plus  $q = 2$ , on obtient ainsi les codes BCH binaires (11.2), dont les codes binaires de Hamming déjà rencontrés sont des cas particuliers (11.1.1). Si l'on prend  $q > 2$ , mais  $r = 1$ , on obtient les codes de Reed-Solomon (11.1.2). Le cas des codes BCH généraux ( $q$  et  $r$  quelconques) recouvre les deux précédents (11.1.5).

À l'inverse, on peut utiliser les diviseurs exceptionnels des  $q^r - 1$ . On a ainsi  $2^{11} - 1 = 23 \times 89$  et on peut prendre  $q = 2$ ,  $n = 23$  et  $r = 11$ ; c'est le cas du code binaire de Golay  $G_{23}$  (13.2). On a de même  $3^5 - 1 = 2 \times 11^2$  et on peut prendre  $q = 3$ ,  $n = 11$  et  $r = 5$ ; c'est le cas du code ternaire de Golay  $G_{11}$  (13.1.1).

### § 11.1. Codes cycliques usuels

#### 11.1.1. Codes de Hamming binaires

Prenons  $q = 2$ , fixons un entier  $r > 1$  et prenons  $n = 2^r - 1$ . Alors l'ordre de 2 modulo  $n$  est égal à  $r$  et le corps  $K = \mathbf{R}_n(\mathbf{F}_2)$  a  $2^r$  éléments. Fixons une racine primitive  $n$ -ième de l'unité dans ce corps, soit  $\alpha$ , de sorte que  $K = \mathbf{F}_2(\alpha)$ . Le polynôme minimal  $g$  de  $\alpha$  sur  $\mathbf{F}_2$  est un facteur irréductible du polynôme cyclotomique  $\Phi_n$  dans  $\mathbf{F}_2[X]$ . On a

$$g(X) = \prod_{i=0}^{r-1} (X - \alpha^{2^i}),$$

et  $g(X)$  correspond par définition à la classe cyclotomique

$$\Sigma = \{1, 2, \dots, 2^{r-1}\}.$$

Le code cyclique  $C$  correspondant est de degré  $n = 2^r - 1$  et de dimension  $n - \deg(g) = 2^r - 1 - r$ . Nous allons voir ci-dessous que c'est un code de Hamming. Notons déjà :

**PROPOSITION 11.1.** — *Le code binaire cyclique de générateur  $g$  est de type  $(2^r - 1, 2^r - r - 1, 3)$ . Il est 1-correcteur et parfait.*

*Démonstration.* Puisque  $\Sigma$  contient 1 et 2, la distance minimale de  $C$  est au moins 3 (proposition 10.21), donc  $C$  est 1-correcteur. On applique alors la proposition 6.8.  $\square$

Montrons maintenant comment on retrouve la description des codes de Hamming donnée en 6.5. Le code que nous venons de construire est par définition le sous-espace vectoriel de  $\mathbf{F}_2^n$  formé des mots  $\mathbf{m} = (m_0, \dots, m_{n-1})$  tels que le polynôme  $m(X) = m_0 + \dots + m_{n-1}X^{n-1}$  ait  $\alpha$  comme racine, ou encore que  $u(\mathbf{m}) = \sum_{i=0}^{n-1} m_i \alpha^i = 0$ . Mais les  $\alpha^i$  sont par hypothèse *tous les éléments non nuls* du  $\mathbf{F}_2$ -espace vectoriel  $K$ , de dimension  $r$ . La matrice de l'application linéaire  $u : \mathbf{F}_2^n \rightarrow \mathbf{F}_2^r$ , à  $r$  lignes et  $n$  colonnes, possède donc exactement comme colonnes les  $n = 2^r - 1$  vecteurs non nuls, chacun une fois. On retrouve bien la construction directe donnée précédemment.

### 11.1.2. Les codes de Reed-Solomon comme codes cycliques

#### Un lemme

Nous utiliserons le lemme suivant (valable dans un corps  $K$  quelconque et dans lequel  $n$  est quelconque) :

**LEMME 11.2.** — *Soient  $K$  un corps et  $n$  un entier  $> 0$ . Soit  $\alpha$  une racine primitive  $n$ -ième de l'unité dans  $K$ . Entre deux polynômes  $m(X)$  et  $u(X)$  de degré  $< n$  à coefficients dans  $K$ , les deux relations suivantes sont équivalentes :*

- (i) *on a  $n.m(X) = \sum_{0 \leq i < n} u(\alpha^i)X^i$ ,*
- (ii) *on a  $u(Z) = \sum_{0 \leq j < n} m(\alpha^{-j})Z^j$ .*

*Démonstration.* Considérons l'application qui, à un tel polynôme  $m$ , associe le polynôme

$$\phi_m(Z) = \sum_{0 \leq j < n} m(\alpha^{-j})Z^j.$$

Cette application est linéaire sur  $K$ , et il suffit de la calculer dans le cas d'un monôme  $m = X^i$ . On a alors  $\phi_m(Z) = \sum_{0 \leq j < n} \alpha^{-ij} Z^j = \sum (\alpha^{-i} Z)^j$ . En particulier, fixons un entier  $r$  avec  $0 \leq r < n$ . On a  $\phi_m(\alpha^r) = \sum_{0 \leq j < n} (\alpha^{r-i})^j$ . Il s'agit d'une progression géométrique de longueur  $n$ , dont la raison  $\alpha^{r-i}$  est de puissance  $n$ -ième égale à l'unité. Ainsi, de deux choses l'une : ou bien  $r = i$  et la somme vaut  $n$  ; ou bien  $r \neq i$  et la somme est nulle. Si l'on revient maintenant au cas général pour  $m$  et qu'on

somme les résultats obtenus, on voit que si  $m = \sum_i m_i X^i$ , on a  $\phi_m(\alpha^i) = nm_i$ , donc  $\sum_i \phi_m(\alpha^i)X^i = nm(X)$ . Cela prouve que (ii) implique (i).

La réciproque s'obtient en changeant  $\alpha$  en son inverse, et en échangeant les rôles de  $u$  et de  $n.m$  (on notera que  $n.1_K$  est inversible, comme il résulte par exemple de l'exercice 3.6).  $\square$

## Définition

Revenons à la situation décrite dans 10.4.2. Prenons maintenant  $q = p^s$  quelconque, mais  $n = q - 1$ , de sorte que  $r = 1$  et que  $K = \mathbf{F}_q$ . Notons  $\alpha$  un générateur de  $\mathbf{F}_q^*$ , donc une racine primitive  $(q - 1)$ -ième de l'unité. Le polynôme minimal de  $\alpha^i$  est simplement  $X - \alpha^i$ . Fixons un entier  $d$  avec  $1 < d < q$  et posons

$$g = \prod_{i=1}^{d-1} (X - \alpha^i) \in \mathbf{F}_q[X].$$

Alors  $g$  est le générateur d'un code cyclique  $C$  sur  $\mathbf{F}_q$ , de longueur  $q - 1$ , de dimension  $k = q - d$ , qu'on appelle *code de Reed-Solomon*, ou code de type RS. D'après la proposition 10.21, la distance minimale de  $C$  est  $\geq d$ . Mais elle est par ailleurs  $\leq q - k = d$ , soit parce que le polynôme  $g$  lui-même est de degré  $d - 1$  et donc de poids  $\leq d$ , soit à cause de la majoration de Singleton (proposition 10.5). Ainsi, les paramètres de  $C$  sont  $(q; n, k, d)$ , avec  $n = q - 1$  et  $k + d = q = n + 1$ . Par conséquent les codes de Reed-Solomon sont de type MDS.

**EXERCICE 11.1. [B]** — Décrire l'orthogonal d'un code RS.

On prend le plus souvent pour  $q$  une puissance de 2, soit  $q = 2^s$ . On représente alors les éléments de  $\mathbf{F}_q$  par des suites de  $s$  bits. De cette façon, on obtient par démultiplication (voir 10.1.4) un code *binaire* de longueur  $sn = s(q - 1) = s(2^s - 1)$  et de dimension  $s(q - d)$ . Quand à sa distance minimale, elle est au moins égale à la distance minimale  $d$  du code initial, et souvent strictement supérieure.

Donnons l'exemple le plus simple, celui du code de paramètres  $(4; 3, 2, 2)$ . On considère donc le corps  $\mathbf{F}_4$ , d'éléments 0, 1,  $\alpha$  et  $\beta$ . On prend  $g(X) = X - \alpha$ . Alors le code est engendré par  $g(X) = \alpha + X = (\alpha, 1, 1)$  et  $Xg(X) = \alpha X + X^2 = (0, \alpha, 1)$  et on retrouve l'exemple donné en 10.1.5.

### 11.1.3. L'autre description des codes de Reed-Solomon

Fixons  $d$ , donc  $k = q - d$ . Dire que le polynôme  $m$  est un mot de code, c'est-à-dire que  $m(\alpha^i) = 0$  pour  $0 < i < d$ , signifie aussi que  $m(\alpha^{-j}) = 0$  pour  $q - d - 1 < j < q - 1$ , c'est-à-dire  $k \leq j < n - 1$ . Si on applique le lemme 11.1.2, cela signifie que le polynôme  $u$ , qui correspond au polynôme  $m$ , est de degré  $< k$ . Par conséquent :

**PROPOSITION 11.3.** — *Le code de Reed-Solomon  $C \subset \mathbf{F}_q^{q-1}$  de dimension  $k$  sur  $\mathbf{F}_q$  est formé des vecteurs de la forme  $(u(\alpha^i))$ , avec  $i = 0, \dots, q-2$ , où  $u$  parcourt l'ensemble des polynômes de  $\mathbf{F}_q[X]$  de degré <  $k$ .*

On retrouve ainsi la construction donnée dans 10.2.4 : les  $\alpha^i$  sont simplement tous les éléments de  $\mathbf{F}_q^*$ , pris dans un ordre particulier qui manifeste le caractère cyclique du code.

On obtient ainsi une nouvelle application de *codage* pour C. Traduisons ce qui précède, en supprimant le passage intermédiaire par les polynômes : on associe au message initial  $\mathbf{u} = (u_0, \dots, u_{k-1}) \in \mathbf{F}_q^k$  le mot de code  $\mathbf{m} = (m_0, \dots, m_{q-2}) \in \mathbf{F}_q^{q-1}$  tel que

$$m_i = \sum_{j=0}^{k-1} u_j \alpha^{ij}, \quad i = 0, \dots, q-2.$$

On voit ainsi apparaître une matrice génératrice du type Vandermonde. On notera au passage que, contrairement au codage explicité en 10.3.4, le message initial n'apparaît pas comme partie du mot de code.

**REMARQUE 11.4.** — Les spécialistes de la théorie des codes disent d'une application de codage dans laquelle le message initial apparaît comme une partie du mot de code, qu'elle est *systématique*.

#### 11.1.4. Codes de Reed-Solomon raccourcis

Comme on l'a vu, le fait que les codes de Reed-Solomon atteignent la borne de Singleton permet de les « raccourcir » en conservant leur caractère MDS (proposition 10.7).

Soit  $C \subset \mathbf{F}_q^{q-1}$  un code de Reed-Solomon de paramètres  $(q; q-1, q-d, d)$ . Fixons un entier  $n$  avec  $0 \leq n \leq q-1$ . Considérons le sous-espace de  $\mathbf{F}_q^{q-1}$  formé des mots dont toutes les composantes strictement au-delà de la  $n$ -ième sont nulles, sous-espace qui s'identifie naturellement à  $\mathbf{F}_q^n$ . Alors  $C' = C \cap \mathbf{F}_q^n$  est un code de longueur  $n$  obtenu par *raccourcissement* de C, donc encore de type MDS. Ses paramètres sont  $(q; n, n-d+1, d)$ .

Par exemple, du code de Reed-Solomon de paramètres  $(256; 255, 251, 5)$ , on déduit par raccourcissement à 32 et 28 bits deux codes, de paramètres respectifs  $(256; 32, 28, 5)$  et  $(256; 28, 24, 5)$ , tous deux 2-correcteurs sur  $\mathbf{F}_{256}$ , qui sont utilisés dans les disques compacts audio (voir 12.2.2).

#### 11.1.5. Codes BCH

Les codes BCH sont ainsi nommés d'après les initiales de leurs inventeurs : R.C. BOSE, D.K. RAY-CHAUDHURI et A. HOCQUENGHEM.

Rappelons une fois de plus la situation générale de 10.4.2. On se donne un corps  $\mathbf{F}_q$ , un entier  $n$  premier à  $q$ , et on note  $r$  l'ordre de  $q$  modulo  $n$ , de sorte que, avec les abus de notations usuels, on a  $\mathbf{R}_n(\mathbf{F}_q) = \mathbf{F}_{q^r}$ . On choisit une racine primitive  $n$ -ième de l'unité  $\alpha$  dans le corps  $\mathbf{F}_{q^r}$ . Le choix de  $\alpha$  revient essentiellement, comme on l'a vu, à celui d'un facteur irréductible  $g_1$  de  $\Phi_n$  sur  $\mathbf{F}_q$ . Pour chaque entier  $j$  avec  $0 < j < n$ , on note  $g_j$  le polynôme minimal de  $\alpha^j$  sur  $\mathbf{F}_q$ . On a

$$g_j(X) = \prod_{i \in \Sigma_j} (X - \alpha^i) \in \mathbf{F}_q[X],$$

où  $\Sigma_j$  est la classe cyclotomique engendrée par  $j$ , donc la plus petite partie de  $\mathbf{Z}/n\mathbf{Z}$  qui contienne  $j$  et soit stable par multiplication par  $q$ . On a par exemple  $\Sigma_1 = \{1, q, q^2, \dots, q^{r-1}\}$  et ces éléments sont tous distincts, de sorte que  $\deg g_1 = \#\Sigma_1 = r$ , comme il convient. De même, on a  $\Sigma_j = \{j, qj, q^2j, \dots, q^{r-1}j\}$ , de sorte que  $\deg(g_j) = \#\Sigma_j \leq r$ . En fait  $\deg(g_j)$  est un diviseur de  $r$  (voir l'exercice 10.22).

Soit  $\delta$  un entier avec  $1 \leq \delta \leq n$ .

**DÉFINITION 11.5.** — *On appelle code BCH de longueur  $n$  et de distance assignée  $\delta$ , le code cyclique de longueur  $n$  sur  $\mathbf{F}_q$ , dont le générateur  $g$  est le plus petit multiple commun des  $g_j$  pour  $0 < j < \delta$ .*

Par exemple, pour  $n = q - 1$ , on obtient les codes de Reed-Solomon. Pour  $q = 2$  et  $n = 2^r - 1$ , on obtient les codes de Hamming binaires.

Pour  $q$  et  $r$  donnés, la valeur maximale possible pour  $n$  est  $q^r - 1$ . Lorsqu'on a  $n = q^r - 1$ , on dit avoir affaire à un code BCH strict. Dans le cas général ( $n$  est donc seulement un diviseur de  $q^r - 1$ ), on parle de code BCH généralisé.

Revenons à la construction précédente. Le code est formé des polynômes  $m \in \mathbf{F}_q[X]$ , de degré  $< n$ , et tels que  $m(\alpha^j) = 0$  pour  $j = 1, \dots, \delta - 1$ . En termes de classes cyclotomiques, ce code est défini par la réunion  $\Sigma$  des classes  $\Sigma_j$  pour  $0 < j < \delta$ . Par conséquent :

**LEMME 11.6.** — *Le générateur minimal  $g$  du code est égal à*

$$g(X) = \prod_{i \in \Sigma} (X - \alpha^i) \in \mathbf{F}_q[X],$$

où  $\Sigma$  est la plus petite partie de l'ensemble  $\mathbf{Z}/n\mathbf{Z}$  qui contienne  $\{1, \dots, \delta - 1\}$  et qui soit stable par multiplication par  $q$ .

Le code obtenu est de dimension  $k = n - \deg(g) = n - \#\Sigma$ . On a d'ailleurs  $\#\Sigma \leq r(\delta - 1)$ , donc

$$k \geq n - r(\delta - 1).$$

On déduit directement de la proposition 10.21 :

**PROPOSITION 11.7.** — *La distance minimale du code construit précédemment est au moins égale à la distance assignée  $\delta$ .*

**REMARQUE 11.8.** — Si  $d$  est la distance minimale du code, on a donc  $\delta \leq d$ . On peut avoir  $\delta < d$ . C'est par exemple le cas si  $\delta$  se trouve appartenir à  $\Sigma$ , de sorte qu'on aurait obtenu la même partie  $\Sigma$  en assignant la distance  $\delta + 1$ . On appelle *distance de Bose* du code le plus grand entier  $d_B$  tel que  $\Sigma$  contienne  $1, 2, \dots, d_B - 1$ . On a  $\delta \leq d_B \leq d$ . La distance de Bose  $d_B$  ne dépend que de la partie  $\Sigma$ ; sa détermination est immédiate. Celle de la vraie distance  $d$  est le plus souvent extrêmement difficile. On peut très bien avoir  $d_B < d$ : le premier exemple dans le cas binaire strict ( $q = 2$  et  $n = 2^r - 1$ ) se produit pour  $n = 127$  et  $d_B = 29$ , pour lequel on a  $d = 31$  (et  $k = 43$ ). La situation analogue pour les codes BCH généralisés est fréquente. On rencontrera par exemple plus loin le code binaire de Golay de distance minimale 7, mais de distance de Bose 5.

## § 11.2. Codes BCH binaires stricts

### 11.2.1. Construction des codes BCH binaires stricts

Nous nous plaçons maintenant dans le cas binaire ( $q = 2$ ) strict ( $n$  est de la forme  $2^r - 1$ ). La distance assignée  $\delta$  étant donnée, on considère donc la partie  $\Sigma$  de  $\mathbf{Z}/n\mathbf{Z}$ , contenant  $\{1, \dots, \delta - 1\}$ , stable par multiplication par 2, et minimale pour ces propriétés.

Notons d'abord que la distance de Bose est impaire, donc que l'on peut supposer que  $\delta$  est impaire. En effet, si l'on assigne une distance  $\delta$  paire, on a  $\delta/2 < \delta$ , donc  $\delta/2 \in \Sigma$ , donc aussi  $\delta \in \Sigma$  et on aurait obtenu la même partie  $\Sigma$  en assignant la distance impaire  $\delta + 1$ . Posons donc  $\delta = 2t + 1$ . Le code obtenu est au moins  $t$ -correcteur. Avec les notations introduites plus haut, on a toujours  $\Sigma_{2i} = \Sigma_i$ , de sorte que  $\Sigma$  est la réunion des  $\Sigma_i$  avec  $i < d$  impair, soit  $i = 1, 3, \dots, 2t - 1$ .

Résumons :

**PROPOSITION 11.9.** — *Le code BCH binaire (strict) de longueur  $n = 2^r - 1$  et de distance assignée  $2t + 1$  est formé, une racine primitive  $n$ -ième de l'unité  $\alpha$  étant choisie, des polynômes  $m \in \mathbf{F}_2[X]$  qui sont de degré  $< n$  et qui satisfont aux conditions  $m(\alpha^j) = 0$  pour  $1 \leq j \leq 2t$  (il suffit de vérifier pour  $j$  impair). Les paramètres de ce code sont  $(n, k, d)$  avec*

$$k \geq 2^r - 1 - rt, \quad d \geq 2t + 1.$$

Notons que la détermination explicite de  $k$  et de  $d$  ne va pas de soi. On peut cependant trouver des estimations suffisamment fortes pour obtenir

l'énoncé suivant, que nous donnons sans démonstration<sup>1</sup>, et qui montre que la famille des codes BCH binaires stricts est « mauvaise » au sens de 8.3.5 :

**PROPOSITION 11.10.** — *Pour tout  $\varepsilon > 0$  donné, il n'existe qu'un nombre fini de codes BCH binaires stricts avec  $k/n \geq \varepsilon$  et  $d/n \geq \varepsilon$ .*

### 11.2.2. Exemple : les codes BCH binaires de longueur 15

Prenons par exemple  $r = 4$ , donc  $n = 15$  et reprenons les notations de 9.4.2. On choisit donc pour  $\alpha$  une racine du polynôme  $g_1 = 1 + X + X^4$ . Déterminons les ensembles  $\Sigma$  correspondants aux différentes valeurs de  $\delta$ . Par multiplications successives par 2 modulo 15, on obtient les chaînes  $1 \mapsto 2 \mapsto 4 \mapsto 8 \mapsto 1$ ,  $3 \mapsto 6 \mapsto 12 \mapsto 9 \mapsto 3$ ,  $5 \mapsto 10 \mapsto 5$  et  $7 \mapsto 14 \mapsto 13 \mapsto 11 \mapsto 7$ , ce qui donne les quatre classes cyclotomiques non-triviales

$$\Sigma_1 = \{1, 2, 4, 8\}, \quad \Sigma_3 = \{3, 6, 9, 12\}, \quad \Sigma_5 = \{5, 10\}, \quad \Sigma_7 = \{7, 11, 13, 14\}.$$

On en déduit le tableau suivant où l'on donne successivement : la distance assignée  $\delta$  (avec la distance de Bose indiquée en gras, qui se trouve être à chaque fois la distance minimale  $d$ ), la partie  $\Sigma$  obtenue, la dimension  $k$  du code correspondant, le nombre d'erreurs  $t$  qu'il peut corriger (on a  $d = 2t + 1$ ) et le polynôme générateur  $g$  (voir après le tableau pour les valeurs des  $g_i$  et quelques explications).

$\delta$	$\Sigma$	$k$	$t$	$g$
<b>2, 3</b>	{1, 2, 4, 8}	11	1	$g_1$
<b>4, 5</b>	{1, 2, 3, 4, 6, 8, 9, 12}	7	2	$g_1 g_3$
<b>6, 7</b>	{1, 2, 3, 4, 5, 6, 8, 9, 10, 12}	5	3	$g_1 g_3 g_5$
<b>8, ..., 15</b>	{1, ..., 14}	1	7	$1 + \cdots + X^{14}$

Le dernier code de la liste est le code de répétition pure. Les polynômes  $g$  sont les suivants. D'abord  $g_1$ , polynôme minimal de  $\alpha$ , est  $1 + X + X^4$ ; ensuite, puisque  $\alpha^3$  est une racine primitive 5-ième de l'unité et que  $\Phi_5$  est irréductible sur  $\mathbf{F}_2$ , on a  $g_3 = \Phi_5 = 1 + X + X^2 + X^3 + X^4$ ; on a de même  $g_5 = \Phi_3 = 1 + X + X^2$ . Enfin, tous calculs faits, on obtient

$$g_1 g_3 = 1 + X^4 + X^6 + X^7 + X^8, \quad g_1 g_3 g_5 = 1 + X + X^2 + X^4 + X^5 + X^8 + X^{10}.$$

On notera d'ailleurs les relations

$$w(g_1) = 3, \quad w(g_1 g_3) = 5, \quad w(g_1 g_3 g_5) = 7, \quad w(1 + \cdots + X^{14}) = 15,$$

qui impliquent dans chaque cas que la distance minimale  $d$  coïncide bien avec la distance de Bose  $d_B$ .

<sup>1</sup>. Voir [MacWilliams, Sloane], ch. 9, theorem 13.

**EXERCICE 11.2. [B]** — On prend  $n = 2^r - 1$  avec  $r \geq 3$ . Pour  $\delta = 3$ , montrer que  $g = g_1$ ,  $d = 3$  et  $k = 2^r - 1 - r$ . Pour  $\delta = 5$ , montrer que  $g = g_1g_3$ ,  $d \geq 5$  et  $k \geq 2^r - 1 - 2r$ , avec égalité si  $r$  est impair.

### 11.2.3. Une autre définition des codes BCH binaires stricts

Considérons toujours le cas binaire strict ( $q = 2$ ,  $r$  quelconque,  $n = 2^r - 1$ ) et posons comme plus haut  $K = \mathbf{F}_{2^r} = \mathbf{F}_2(\alpha)$ , avec  $\alpha^n = 1$ . Le code BCH binaire  $C$  de longueur  $n$  et de distance assignée  $\delta$  est formé des polynômes

$$m(X) = m_0 + m_1X + \cdots + m_{n-1}X^{n-1} \in \mathbf{F}_2[X]$$

tels que  $m(\alpha^j) = 0$  pour  $0 < j < \delta$ .

Nous allons maintenant identifier les indices  $0, \dots, n - 1$  des coefficients du polynôme  $m$  (autrement dit des bits du mot  $m$ ) aux éléments non nuls de  $K$ , à l'indice  $i \in \{0, \dots, n - 1\}$  correspondant l'élément  $\alpha^i \in K^*$ . De cette façon, les mots du code  $C$  s'identifient simplement à des parties de  $K^*$ .

On notera que le décalage  $i \mapsto i + 1 \bmod n$  qui sous-tend la cyclicité de  $C$  devient dans ce modèle la permutation  $x \mapsto \alpha x$  de  $K^*$ , qui est bien un cycle d'ordre  $n$ , puisque  $\alpha$  est une racine primitive d'ordre  $n$ .

**PROPOSITION 11.11.** — *Le code  $C$  est formé des parties  $A$  de  $K^*$  telles que  $\sum_{x \in A} x^j = 0$  pour  $0 < j < \delta$ .*

*Démonstration.* Soit  $A$  une partie de  $K^*$  et soit  $m(X) \in \mathbf{F}_2[X]$  le polynôme correspondant. On a par définition

$$m(X) = \sum_{\alpha^i \in A} X^i,$$

et donc

$$m(\alpha^j) = \sum_{\alpha^i \in A} (\alpha^j)^i = \sum_{\alpha^i \in A} (\alpha^i)^j = \sum_{x \in A} x^j,$$

d'où la conclusion. □

Par exemple, le code de Hamming binaire de longueur  $n$  est formé des parties de  $K^*$  de somme nulle. Ainsi, pour  $n = 7$ , avec la relation de définition  $1 + \alpha + \alpha^3 = 0$ , on a  $\alpha^i + \alpha^{i+1} + \alpha^{i+3} = 0$  pour tout  $i$ , ce qui donne les sept droites du plan de Fano (7.4.2) :  $\{0, 1, 3\}$ ,  $\{1, 2, 4\}$  et ainsi de suite par décalage modulo 7.

Considérons maintenant le *code BCH étendu*  $\bar{C}$  obtenu par adjonction à  $C$  d'un bit de parité et identifions l'indice supplémentaire de ce bit à l'élément nul de  $K$ , de sorte que les mots de  $\bar{C}$  s'identifient à des parties de  $K$  (qui ont toutes par construction un nombre pair d'éléments).

**COROLLAIRE 11.12.** — *Le code étendu  $\overline{C}$  est formé des parties B de K telles que  $\sum_{x \in B} x^j = 0$  pour  $0 \leq j < \delta$ .*

*Démonstration.* Soit en effet B une partie de K. Posons  $A = B \cap K^*$ . On a alors  $\sum_{x \in B} x^j = \sum_{x \in A} x^j$  pour  $j \neq 0$ , tandis que  $\sum_{x \in B} x^0 = \#B \bmod 2$ . Les conditions de l'énoncé signifient donc que l'on a  $A \in C$  et que  $\#B$  est pair, c'est-à-dire que l'on a  $B \in \overline{C}$ .  $\square$

**EXERCICE 11.3. [A]** — Pourquoi les définitions ci-dessus donnent-elles des codes linéaires ?

**REMARQUE 11.13.** — On notera que les définitions précédentes de C et  $\overline{C}$  ne dépendent plus du choix de la racine primitive  $\alpha$ . Ce sont donc les « bonnes » définitions. On notera aussi que cette interprétation en termes de parties de  $K^*$  s'étend aux codes cycliques, non nécessairement BCH, dont la longueur est de la forme  $2^r - 1$ .

#### 11.2.4. Automorphismes d'un code BCH binaire strict étendu

**PROPOSITION 11.14.** — *Soient  $u \in K^*$  et  $v \in K$ . Alors, dans l'identification précédente, l'application  $x \mapsto ux + v$  de K dans K est un automorphisme de  $\overline{C}$ .*

*Démonstration.* Soit en effet B  $\subset K$  et soit B'  $\subset K$  la partie parcourue par  $ux + v$  lorsque x parcourt B. Soit  $j$  un entier  $\geq 0$ . On a  $(ux + v)^j = \sum_{0 \leq s \leq j} \binom{j}{s} u^s x^s v^{n-s}$ , ce qui donne

$$\sum_{y \in B'} y^j = \sum_{x \in B} (ux + v)^j = \sum_{x \in B} \sum_{0 \leq s \leq j} \binom{j}{s} u^s v^{n-s} x^s = \sum_{0 \leq s \leq j} \binom{j}{s} u^s v^{n-s} \sum_{x \in B} x^s.$$

Il suffit alors d'appliquer le corollaire 11.12 pour voir que la condition  $B \in \overline{C}$  implique  $B' \in \overline{C}$ .  $\square$

**COROLLAIRE 11.15.** — *La distance minimale d'un code BCH binaire strict est impaire.*

*Démonstration.* Il suffit en effet d'appliquer le lemme 7.22.  $\square$

#### § 11.3. L'algorithme de décodage des codes BCH binaires

Ce qui est intéressant dans les codes BCH, ce n'est pas tant leur capacité de correction, qui est loin de l'optimum, que le fait qu'ils bénéficient d'algorithmes de décodage simples et rapides. Celui que nous présentons maintenant s'applique en fait à une classe de codes plus généraux que l'on appelle les *codes alternants*.

### II.3.1. Position du problème

Prenons toujours  $q = 2$  et considérons un code BCH binaire (généralisé) de longueur  $n$  et de distance assignée  $2t + 1$ . Rappelons les notations : on désigne par  $r$  l'ordre de 2 modulo  $n$  et on choisit une racine primitive  $n$ -ième de l'unité  $\alpha$  dans le corps  $K = \mathbf{F}_{2^r}$ .

Le code  $C$  considéré est le sous-espace vectoriel de l'espace  $E_n$  des polynômes de degré  $< n$  de  $\mathbf{F}_2[X]$ , formé des polynômes  $m$  tels que  $m(\alpha^j) = 0$  pour  $1 \leq j \leq 2t$ . Comme on l'a vu, ce code est  $t$ -correcteur.

Le problème posé est le suivant : le mot de code inconnu  $m \in C \subset E_n$  est perturbé par l'erreur inconnue  $e \in E_n$ , donnant ainsi le mot erroné  $m' = m + e \in E_n$ ; connaissant  $m'$  et sachant qu'on a  $w(e) \leq t$ , il faut retrouver  $m$  (ou  $e$ , ce qui revient au même).

### II.3.2. Syndrome

Par définition, on a  $m(\alpha^i) = 0$  pour  $1 \leq i \leq 2t$ . On appelle *syndrome* de  $m'$  la suite  $(s_1, \dots, s_{2t})$  d'éléments de  $K$  donnée par

$$s_j = m'(\alpha^j) = e(\alpha^j) \in K, \quad j = 1, \dots, 2t.$$

Par ailleurs, l'erreur  $e$  a par hypothèse au plus  $t$  coefficients non nuls (donc égaux à 1) et s'écrit

$$e(X) = X^{r_1} + \cdots + X^{r_f}$$

avec  $0 \leq r_1 < \cdots < r_f < n$  et  $0 \leq f \leq t$ . Reprenons l'interprétation des indices par les éléments de  $K$  et posons  $x_i = \alpha^{r_i} \in K$  pour  $i = 1, \dots, f$ . On a

$$s_j = e(\alpha^j) = \sum_{i=1}^f \alpha^{jr_i} = \sum_{i=1}^f (x_i)^j.$$

On introduit une variable auxiliaire, notée  $Z$ , et on pose

$$S(Z) = \sum_{j=1}^{2t} s_j Z^{j-1} \in K[Z]$$

(c'est le « polynôme syndrome ») et

$$\sigma(Z) = \prod_{i=1}^f (1 - x_i Z) = \prod_{i=1}^f (1 - \alpha^{r_i} Z) \in K[Z]$$

(c'est le « polynôme de localisation »). Le polynôme  $\sigma(Z)$  est de degré  $f \leq t$ . Les inverses de ses racines sont les  $x_i = \alpha^{r_i}$ ; la connaissance de ces éléments du corps  $K$  permet de déterminer les  $r_i$ , donc de reconstituer  $e$  et en définitive  $m = m' - e$ .

### 11.3.3. L'équation-clé

L'opération de décodage peut se décomposer en trois étapes : on calcule d'abord les  $s_j = m'(\alpha^j)$  (donc le polynôme-syndrome), puis on déduit du polynôme-syndrome le polynôme de localisation  $\sigma(Z)$ , et on calcule enfin les racines de ce polynôme dans  $K$  (simplement en essayant les divers éléments). Les deux opérations extrêmes étant directes et rapides, concentrons-nous sur le pas intermédiaire : comment déduire  $\sigma(Z)$  de  $S(Z)$ .

Ce calcul est basé sur l'*équation-clé* :

**PROPOSITION 11.16.** — *Il existe un polynôme  $\omega(Z) \in K[Z]$  de degré  $< t$ , tel que  $S(Z)\sigma(Z) \equiv \omega(Z) \pmod{Z^{2t}}$ . Les polynômes  $\sigma$  et  $\omega$  sont premiers entre eux.*

*Démonstration.* On a

$$S(Z) = \sum_{j=1}^{2t} \left( \sum_{i=1}^f x_i^j \right) Z^{j-1} = \sum_{i=1}^f x_i \left( \sum_{j=1}^{2t} (x_i Z)^{j-1} \right) = \sum_{i=1}^f x_i \frac{1 - x_i^{2t} Z^{2t}}{1 - x_i Z},$$

donc

$$S(Z)\sigma(Z) = \sum_{i=1}^f \left( x_i (1 - x_i^{2t} Z^{2t}) \prod_{k \neq i} (1 - x_k Z) \right).$$

Cela implique le résultat annoncé, avec

$$\omega(Z) = \sum_{i=1}^f x_i \prod_{k \neq i} (1 - x_k Z) = \sigma(Z) \sum_{i=1}^f \frac{x_i}{1 - x_i Z}.$$

On notera qu'on a bien  $\deg(\omega) \leq f - 1 < t$ . Par ailleurs, puisque qu'aucune des racines  $x_j^{-1}$  de  $\sigma$  n'est racine de  $\omega$ , les polynômes  $\sigma$  et  $\omega$  sont premiers entre eux.  $\square$

**LEMME 11.17.** — *Soient  $\sigma'$  et  $\omega'$  deux polynômes de  $K[Z]$  avec  $\deg(\sigma') \leq t$ ,  $\deg(\omega') < t$  et  $S(Z)\sigma'(Z) \equiv \omega'(Z) \pmod{Z^{2t}}$ . Il existe un polynôme  $C \in K[Z]$  avec  $\sigma'(Z) = C(Z)\sigma(Z)$  et  $\omega'(Z) = C(Z)\omega(Z)$ .*

*Démonstration.* On a, modulo  $Z^{2t}$ ,

$$\omega(Z)\sigma'(Z) \equiv S(Z)\sigma(Z)\sigma'(Z) \equiv \omega'(Z)\sigma(Z),$$

donc le polynôme  $P(Z) = \omega(Z)\sigma'(Z) - \omega'(Z)\sigma(Z)$  est divisible par  $Z^{2t}$ . Mais comme  $P$  est de degré  $< 2t$ , il doit être nul, ce qui implique la conclusion annoncée puisque  $\sigma$  et  $\omega$  sont premiers entre eux.  $\square$

### 11.3.4. Résolution de l'équation-clé

Posons  $P_0 = Z^{2t}$  et  $P_1 = S$ . On a  $\deg(P_1) = 2t - 1 < 2t = \deg(P_0)$ . Exécutons l'algorithme d'Euclide étendu à partir de ces deux polynômes

(voir 1.5.3). On construit donc par divisions euclidiennes successives deux suites de polynômes  $P_2, \dots, P_{n+1}$  et  $Q_1, \dots, Q_n$ , de façon que

$$P_{i-1} = P_i Q_i + P_{i+1}, \quad \deg(P_{i+1}) < \deg(P_i).$$

opération que l'on prolonge tant que  $P_i \neq 0$ . On calcule parallèlement les deux autres suites  $A_0, \dots, A_{n+1}$  et  $B_0, \dots, B_{n+1}$  en partant de

$$A_0 = B_1 = 1, \quad A_1 = B_0 = 0,$$

et en posant

$$A_{i+1} = A_{i-1} - Q_i A_i, \quad B_{i+1} = B_{i-1} - Q_i B_i.$$

et on a pour tout  $i$  la relation  $P_i = A_i P_0 + B_i P_1$  et par conséquent la congruence  $S B_i \equiv P_i \pmod{Z^{2t}}$ .

Avec ces notations, la solution de l'équation-clé s'obtient directement :

**PROPOSITION 11.18.** — Soit  $i$  l'unique indice avec  $\deg(P_{i-1}) \geq t$  et  $\deg(P_i) < t$ .  
On a alors

$$B_i(0) \neq 0, \quad \sigma(Z) = B_i(Z)/B_i(0), \quad \omega(Z) = P_i(Z)/B_i(0).$$

*Démonstration.* D'après le lemme 1.41, on a  $\deg(B_i) = \deg(P_0) - \deg(P_{i-1}) \leq 2t - t = t$ . Par conséquent, les polynômes  $\sigma' = B_i$  et  $\omega' = P_i$  satisfont aux hypothèses du lemme précédent. Il existe donc un polynôme  $C(Z) \in K[Z]$  avec

$$B_i = C\sigma, \quad P_i = C\omega.$$

Mais il existe un polynôme  $A(Z)$  tel que  $\omega = S\sigma + AZ^{2t}$ , donc  $P_i = SB_i + CAZ^{2t}$ . Comparant avec la relation liant  $P_i$ ,  $B_i$  et  $A_i$ , on obtient  $A_i = CA$ . Ainsi, le polynôme  $C$  divise  $A_i$  et  $B_i$ ; mais ceux-ci sont premiers entre eux (c'est une des vertus de l'algorithme d'Euclide étendu). Par conséquent, le polynôme  $C$  est une constante non nulle. Puisque  $\sigma(0) = 1$  par construction, on a  $C = B_i(0)$ .  $\square$

**REMARQUE 11.19.** — On peut toujours prendre pour  $\delta = 2t + 1$  la distance de Bose du code. Alors la distance minimale du code est  $\geq 2t + 1$ , le code est  $t$ -correcteur et l'algorithme de décodage que l'on vient de voir permet bien de corriger  $t$  erreurs. Il se peut cependant que la distance minimale du code soit  $\geq 2t + 3$ ; en ce cas le code est même  $t + 1$ -correcteur, mais l'algorithme donné ne permet pas de corriger  $t + 1$  erreurs.

#### § 11.4. L'algorithme de décodage des codes BCH généraux

Passons au cas général, où  $q$  n'est plus nécessairement égal à 2. Récapitulons les notations. On note  $K$  le corps  $R_n(\mathbf{F}_q) = \mathbf{F}_{q^n}$  et on fixe une racine primitive  $n$ -ième de l'unité dans  $K$ , soit  $\alpha$ . On fixe un entier  $\delta$  avec  $1 \leq \delta < n$ . Le code  $C$  considéré est formé des polynômes  $m \in \mathbf{F}_q[X]$ , de degré  $< n$ , tels que  $m(\alpha^j) = 0$  pour  $1 \leq j < \delta$ .

Le mot de code  $m$  inconnu est perturbé par l'erreur  $e$ , donnant le mot reçu  $m' = m + e$ . La suite des syndromes  $s_j \in K$  est donnée par

$$s_j = m'(\alpha^j) = e(\alpha^j), \quad j = 1, \dots, \delta - 1.$$

Nous allons traiter successivement les deux problèmes de corrections : erreurs et effacements.

#### 11.4.1. Correction de $t$ erreurs, $t < \delta/2$

On suppose donné un entier  $t > 0$ , avec  $2t < \delta$ , de sorte que  $C$  est  $t$ -correcteur. On ignore la valeur de l'erreur  $e$ , mais on suppose qu'on a  $w(e) \leq t$ . On peut écrire

$$e(X) = a_1 X^{r_1} + \cdots + a_f X^{r_f},$$

avec  $f = w(e) \leq t$  et, pour tout  $i$ ,  $r_i \in [0, n - 1]$  et  $a_i \in \mathbb{F}_q^*$ . On posera encore  $x_i = \alpha^{r_i} \in K$ . Il s'agit donc de déterminer  $f$ , les  $a_i$  et les  $x_i$ .

On introduit  $S(Z)$  et  $\sigma(Z)$  par les mêmes relations que dans le cas  $q = 2$  :

$$\begin{aligned} S(Z) &= \sum_{j=1}^{2t} s_j Z^{j-1} \in K[Z], \\ \sigma(Z) &= \prod_{i=1}^f (1 - x_i Z) = \prod_{i=1}^f (1 - \alpha^{r_i} Z) \in K[Z]. \end{aligned}$$

On pose également

$$\omega(Z) = \sum_{i=1}^f a_i x_i \prod_{k \neq i} (1 - x_k Z) = \sigma(Z) \sum_{i=1}^f a_i \frac{x_i}{1 - x_i Z},$$

et on a comme ci-dessus l'*équation-clé* :

$$S(Z)\sigma(Z) \equiv \omega(Z) [\text{mod } Z^{2t}].$$

**EXERCICE 11.4. [A] —** Vérifier l'équation-clé.

La même utilisation de l'algorithme d'Euclide que ci-dessus permet de déterminer  $\sigma$  (et aussi  $\omega$ ). La connaissance de  $\sigma$  permet de déterminer les  $x_i$ , donc les positions  $r_i$  des erreurs. Pour déterminer les  $a_i$ , on peut, soit utiliser la connaissance de  $\omega$ , soit se ramener au problème d'effacement correspondant (voir l'algorithme ci-dessous).

#### 11.4.2. Correction de $f$ effacements, $f < \delta$

Supposons maintenant connues les positions  $r_i$  des erreurs possibles. Il s'agit donc de résoudre le problème des effacements. On suppose que ces effacements sont en nombre  $< \delta$ , de sorte que  $w(e) < \delta$ .

On a donc comme ci-dessus

$$e(\mathbf{X}) = a_1 \mathbf{X}^{r_1} + \cdots + a_f \mathbf{X}^{r_f},$$

où cette fois-ci les  $r_i$  sont connus, donc aussi les  $x_i = \alpha^{r_i}$ , et il s'agit de déterminer les  $a_i \in \mathbb{F}_q$ . Mais on a aussitôt

$$s_j = e(\alpha^j) = \sum_{1 \leq i \leq f} a_i (x_i)^j.$$

Les  $s_j$ , pour  $1 \leq j \leq f$ , et les  $x_i$ , pour  $1 \leq i \leq f$ , étant connus, on obtient un système linéaire carré du type Vandermonde, que l'on résout sans difficultés.

# Chapitre 12

## Le codage des disques compacts

Ce chapitre est consacré au codage correcteur dit CIRC utilisé dans les disques compacts audio. Il nous permettra de voir comment, dans un cas concret, les codes construits par les méthodes théoriques que nous avons expliquées (en ce cas les codes de Reed-Solomon) servent de « pièces détachées » pour construire un mécanisme de codage plus complexe, approprié aux contraintes précises du problème posé.

Je me suis directement inspiré d'un exposé de Nicolas SENDRIER.

### § 12.1. Position du problème

#### 12.1.1. La chaîne de codage

Entre le son qu'il s'agit d'enregistrer ou de reproduire et ce qui est gravé sur le disque compact, s'interposent une suite de codages de diverses natures. Ces divers codages répondent à des contraintes différentes. Il s'agit successivement : de numériser le signal audio ; d'appliquer un code correcteur ; de répondre aux contraintes de la gravure et de la lecture. C'est le deuxième pas qui nous intéresse ici, mais il n'est pas inutile de présenter brièvement l'ensemble du processus, ne serait-ce que pour donner une idée des quantités et des flux d'information concernés.

Le son (rappelons qu'il s'agit simplement, pour chaque voie, d'une fonction scalaire du temps) est d'abord échantillonné à 44,1 kHz sur  $2^{16}$  niveaux sur chaque voie : chaque échantillon stéréo contient 2 enregistrements de 2 octets, donc 4 octets. Une seconde de son donne ainsi naissance à  $44\,100 \times 4 = 196\,400$  octets.

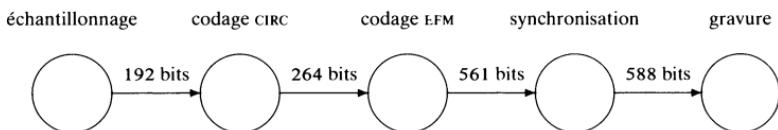
**REMARQUE 12.1.** — Le choix d'une fréquence d'échantillonnage supérieure à 40 kHz permet de reproduire des sons jusqu'à une fréquence de 20 kHz, sans bruit parasite autre que le bruit de quantification (théorème d'échantillonnage de Shannon). La valeur précise de 44 100 Hz provient du standard de télévision PAL : chacune des 625 lignes, sauf 37 d'entre elles, peut porter trois enregistrements audio de 16 bits ce qui, à 25 images par seconde, fait  $(625 - 37) \times 3 \times 25 = 44\,100$  enregistrements par seconde.

En fait, on regroupe les échantillons stéréo en *trames* de 6 échantillons stéréo, soit 24 octets ou 192 bits. Ce qui nous fait un débit de  $44\,100/6 = 7\,300$  trames par seconde. Le pas suivant, que nous détaillerons ci-dessous,

transforme par deux phases successives de codage correcteur, ces trames de 24 octets en trames de 28, puis de 32 octets. Précisons déjà que ce codage n'agit pas trame par trame, mais qu'il est combiné à un *entrelacement*.

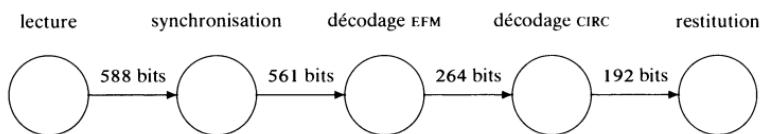
À la suite de ce codage on ajoute à chaque trame de 32 octets un octet supplémentaire de contrôle. Nous sommes alors en présence de trames de 33 octets, soit 264 bits.

Les contraintes technologiques de la gravure et de lecture<sup>1</sup> imposent ensuite un codage supplémentaire dit « eight to fourteen modulation » ou EFM qui, comme son nom le suggère, transforme chaque octet en une suite de 14 bits. On ajoute des suites de liaison de 3 bits (toujours pour répondre à ces contraintes, et à d'autres<sup>2</sup>) et on obtient maintenant  $33 \times 17 = 561$  bits, auquel on adjoint enfin une information de synchronisation de 27 bits, ce qui nous amène enfin à 588 bits, que l'on grave sur le disque. En résumé, voici la suite des transformations effectuées.



En définitive, pour chaque trame initiale de 192 bits est gravée sur le disque une suite de 588 bits, ce qui, pour chaque seconde de son enregistré, nous donne  $588 \times 7300 = 4\,321\,800$  bits. La distance entre deux bits gravés consécutifs étant de  $0,3 \mu\text{m}$ , une trame occupe après codage environ  $180 \mu\text{m}$  linéaires. La longueur totale de la spirale gravée est d'environ 5,7 km, ce qui permet d'enregistrer plus de  $3\,10^7$  trames, soit environ 74 minutes de musique<sup>3</sup>.

Naturellement, à la lecture, on suivra le chemin inverse :



- 
1. On veut interdire les suites trop longues de bits identiques qui se traduirait par des sillons « homogènes » trop longs et qui pourraient induire ainsi à la lecture une erreur sur le nombre de ces bits.
  2. Il s'agit en plus d'éliminer des quasi-périodicités à basse fréquence qui risqueraient de provoquer des résonances du mécanisme de lecture.
  3. On dit que cette durée maximale a été définie en fonction du plus long enregistrement existant sur microsillon d'une symphonie de Beethoven.

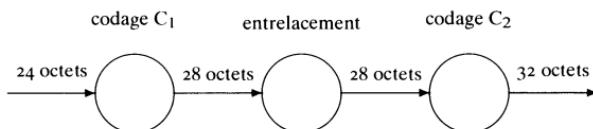
### 12.1.2. Les contraintes du codage correcteur

Nous allons maintenant nous intéresser spécifiquement à la partie « code correcteur » de l'opération. Il s'agit donc de transformer une suite de trames de 24 octets en une suite de trames de 32 octets (« codage ») et inversement d'exhiber un algorithme permettant de reconstituer la suite initiale à partir de la suite codée, modifiée éventuellement par des erreurs de lecture (« décodage »).

On veut faire face simultanément à deux types d'erreurs : des erreurs « microscopiques » portant sur quelques bits (poussières, interférences, défauts de métallisation) et des « bouffées d'erreurs » portant sur un grand nombre de bits successifs. Songeons par exemple qu'une rayure d'un millimètre détruit plus de 5 trames, soit un millier de bits d'information successifs. Le codage que nous allons présenter permet de corriger l'effacement total de 16 trames consécutives, soit environ trois millimètres.

## § 12.2. Le code CIRC

Le codage CIRC comporte trois phases, résumées par le schéma suivant :



Expliquons d'abord ce qu'est un entrelacement.

### 12.2.1. Entrelacement

Pour corriger des bouffées particulièrement longues, on utilise une technique dite *d'entrelacement*. Montrons-en d'abord le principe d'une façon un peu plus générale. On fixe un ensemble de symboles, disons  $S$  (ce sera dans notre cas l'ensemble  $\mathbf{F}_2^8$  des octets binaires) et un entier  $n > 0$  (ici, on aura  $n = 28$ ). On considère l'ensemble  $S^n$  des mots formés de  $n$  symboles de  $S$ , dont on notera un élément typique  $\mathbf{m} = (m_0, \dots, m_{n-1})$ , et on suppose avoir à transmettre une suite  $\sigma$  de tels mots, soit  $\mathbf{m}^{(0)}, \mathbf{m}^{(1)}, \dots$ .

On fixe un entier  $r \geq 0$  (ici, ce sera  $r = 4$ ) et on va définir ce qu'est l'*entrelacement à retard r* de cette suite. Pour  $r = 0$ , c'est tout simplement la suite de ces mots mis bout à bout,

$$E_0(\sigma) = (m_0^{(0)}, m_1^{(0)}, \dots, m_{n-1}^{(0)}, m_0^{(1)}, \dots, m_{n-1}^{(1)}, m_0^{(2)}, \dots, m_j^{(i)}, \dots)$$

que l'on peut imaginer obtenue en écrivant les  $\mathbf{m}^{(i)}$  en colonnes les uns à la suite des autres, de façon à former une matrice à  $n$  lignes, par exemple pour  $n = 3$ ,

$$\begin{bmatrix} m_0^{(0)} & m_0^{(1)} & m_0^{(2)} & m_0^{(3)} & m_0^{(4)} & m_0^{(5)} & \dots \\ m_1^{(0)} & m_1^{(1)} & m_1^{(2)} & m_1^{(3)} & m_1^{(4)} & m_1^{(5)} & \dots \\ m_2^{(0)} & m_2^{(1)} & m_2^{(2)} & m_2^{(3)} & m_2^{(4)} & m_2^{(5)} & \dots \end{bmatrix}$$

et en la lisant colonne par colonne. Pour construire l'entrelacement à retard  $r$ , on décale chaque ligne de la matrice de  $r$  cases à droite relativement à la ligne précédente, on remplit les cases laissées vides par un symbole fixé, disons  $*$  ∈ S, et on lit la matrice colonne par colonne. Par exemple, si  $n = 3$  et  $r = 2$ , on obtient la matrice

$$\begin{bmatrix} m_0^{(0)} & m_0^{(1)} & m_0^{(2)} & m_0^{(3)} & m_0^{(4)} & m_0^{(5)} & \dots \\ * & * & m_1^{(0)} & m_1^{(1)} & m_1^{(2)} & m_1^{(3)} & \dots \\ * & * & * & * & m_2^{(0)} & m_2^{(1)} & \dots \end{bmatrix}$$

et donc la suite

$$\begin{aligned} E_2(\sigma) = (m_0^{(0)}, *, *, m_0^{(1)}, *, *, m_0^{(2)}, m_1^{(0)}, *, \\ m_0^{(3)}, m_1^{(1)}, *, m_0^{(4)}, m_1^{(2)}, m_2^{(0)}, \dots) \end{aligned}$$

Une façon plus arithmétique de faire est de considérer l'application  $e_r : \mathbf{Z} \times [0, n - 1] \rightarrow \mathbf{Z}$  définie par  $e_r(i, j) = ni + (nr + 1)j$ .

**LEMME 12.2.** — *L'application  $e_r$  est bijective.*

**EXERCICE 12.1. [B]** — Démontrer le lemme.

On pose alors  $E_r(\sigma)_{e_r(i,j)} = m_j^{(i)}$  en convenant que  $m_j^{(i)} = *$  pour  $i < 0$ .

Quoi qu'il en soit, la vertu essentielle de la construction précédente est que les composantes d'un même mot initial  $\mathbf{m}^{(i)}$  apparaissent maintenant de  $rn + 1$  en  $rn + 1$  dans la suite entrelacée. Reprenons l'exemple précédent, où l'on a  $rn + 1 = 7$  :

$$\begin{aligned} E_2(\sigma) = (\underbrace{m_0^{(0)}, *, *, m_0^{(1)}, *, *, m_0^{(2)},}_{m_1^{(0)}, *}, \\ m_0^{(3)}, m_1^{(1)}, *, m_0^{(4)}, m_1^{(2)}, \underbrace{m_2^{(0)}, \dots}) \end{aligned}$$

On voit maintenant clairement l'intérêt : une bouffée d'erreurs de longueur  $t(rn + 1)$  dans la suite entrelacée n'atteindra que  $t$  composantes dans chaque mot initial.

### 12.2.2. Le codage

Revenons à notre problème de codage. On va utiliser des codes de Reed-Solomon sur le corps à 256 éléments. On se place donc dans le corps  $K = \mathbf{F}_{256}$

dans lequel on a choisi une racine primitive 255-ième de l'unité, notée  $\alpha$ . Les éléments de  $K$  sont donc, outre 0, les 255 puissances  $\alpha^i$ , l'entier  $i$  variant modulo 255. Par ailleurs, on choisit une base de l'espace vectoriel  $K$  sur  $\mathbf{F}_2$ , par exemple les  $\alpha^i$  pour  $0 \leq i \leq 7$ , ce qui permet d'identifier  $K$  à  $\mathbf{F}_2^8$ , donc chaque élément de  $K$  à un octet binaire.

### Le premier codage

On part donc de trames de 24 octets, considérés comme autant d'éléments du corps  $\mathbf{F}_{256}$ . On utilise le code RS raccourci  $C_1$  de paramètres  $(256; 28, 24, 5)$  introduit dans 11.1.4, et on obtient des trames de 28 octets. Le code  $C_1$  étant de distance minimale 5, la suite obtenue peut supporter 2 erreurs par trame de 28 octets, ou plutôt 4 effacements puisque, comme nous allons le voir, ce premier codage ne sera utilisé qu'en correction d'effacements.

### L'entrelacement

Ensuite, on entrelace ces trames de 28 octets avec un retard 4. On a donc  $S = \mathbf{F}_{256}$  (on prend  $* = 0$  par exemple),  $n = 28$ ,  $r = 4$ ,  $rn + 1 = 113$ . Ainsi, au terme de cette opération, on obtient une nouvelle suite de trames de 28 octets (ce sont les colonnes de la matrice considérée ci-dessus), qui peut maintenant supporter une bouffée d'effacements de  $4 \times 113 = 452$  octets successifs, soit au moins 16 trames consécutives (soit 2,88 mm linéaires).

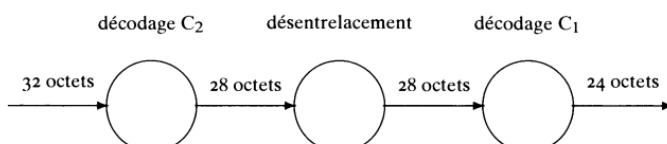
### Le deuxième codage

On code alors la suite précédente à l'aide du deuxième code RS raccourci  $C_2$  de paramètres  $(256; 32, 28, 5)$ , ce qui donne une suite de trames de 32 octets. Ce deuxième codage permet de détecter 4 erreurs par trame et d'en corriger 2.

L'opération combinée : codage par  $C_1$ , entrelacement, codage par  $C_2$ , porte le nom de *Cross Interleaved Reed-Solomon Code*, en abrégé *code CIRC*.

#### 12.2.3. Le décodage

Naturellement, le décodage est organisé en trois étapes, mais l'articulation des deux phases de décodage est assez subtile.



## Le décodage de C<sub>2</sub>

On décompose d'abord la chaîne lue en trames de 32 octets. Chacune de ces trames est de la forme  $\mathbf{m}' = \mathbf{m}_0 + \mathbf{e}_0$ , où  $\mathbf{m}_0$  est le mot du code C<sub>2</sub> qu'il s'agit de retrouver et  $\mathbf{e}_0$  l'erreur de lecture. Le code C<sub>2</sub>, de distance minimale 5, permettrait de corriger 2 erreurs, mais *on se contentera d'en corriger une seule*. Plus précisément, on applique l'algorithme de décodage expliqué en 11.4.1 et explicité plus loin, avec la stratégie suivante :

a) si  $\mathbf{m}'$  peut s'écrire  $\mathbf{m} + \mathbf{e}$ , avec  $\mathbf{m} \in C_2$  et  $w(\mathbf{e}) \leq 1$ , on décide que  $\mathbf{m}_0 = \mathbf{m}$ ;

b) sinon, on décide que  $\mathbf{m}_0$  est inconnu.

Notons déjà que cette réponse ne peut être inexacte que dans le cas où on aurait  $\mathbf{m}_0 + \mathbf{e}_0 = \mathbf{m} + \mathbf{e}$ , avec  $w(\mathbf{e}) \leq 1$ ,  $\mathbf{m} \in C_2$  et  $\mathbf{m} \neq \mathbf{m}_0$ . Mais cela impliquerait  $w(\mathbf{e} - \mathbf{e}_0) = w(\mathbf{m} - \mathbf{m}_0) \geq 5$ , donc  $w(\mathbf{e}_0) \geq 4$ .

Majorons grossièrement la probabilité d'une telle réponse inexacte en faisant l'hypothèse extraordinairement pessimiste qu'il y a tant d'erreurs élémentaires que tous les mots en deviennent équiprobables à la lecture. Le nombre de mots contenus dans une boule de rayon 1 autour d'un mot de code donné est  $1 + n(q - 1)$ ; le nombre total des mots de code est  $q^k$ , ce qui donne  $q^k(1 + n(q - 1))$  mots reconnus par l'algorithme. Le nombre total des mots est  $q^n$ . Au total, la probabilité de la réponse a) est  $q^{k-n}(1 + n(q - 1))$ . Pour  $q = 256$ ,  $n = 32$  et  $k = 28$ , on obtient  $1,9 \cdot 10^{-6}$ . La probabilité d'une réponse a) erronée est évidemment inférieure.

## Le décodage de C<sub>1</sub>

À la sortie de cette première étape, on obtient une suite de trames de 28 octets, dont certaines sont déclarées inconnues. On désentrelace alors cette suite de trames. Une trame déclarée comme inconnue au pas précédent donne 28 octets inconnus donc, après désentrelacement, des effacements dans 28 trames distinctes. On a récupéré ainsi une suite de trames de 28 octets partiellement effacées auxquelles on applique l'algorithme de décodage des effacements du code C<sub>1</sub>, vu en 11.4.2 et explicité plus loin, qui fonctionne tant que l'on ne dépasse pas 4 effacements par trame. C'est par exemple le cas si lors du premier pas, on a déclaré inconnues moins de 16 trames successives.

### 12.2.4. Les détails du codage et du décodage

#### Codage pour C<sub>1</sub> et C<sub>2</sub>

Le premier pas du codage consiste en la transformation des trames initiales de 24 octets en trames de 28 octets du code C<sub>1</sub>. Chaque trame initiale  $\mathbf{m} = (m_0, \dots, m_{23})$  est assimilée à une suite de 24 éléments du corps K, puis au polynôme

$$m(X) = m_0 + \cdots + m_{23}X^{23} \in \mathbf{F}_{256}[X].$$

Les mots du code  $C_1$  sont par définition les polynômes  $P$  de degré  $< 28$ , multiples de  $g(X) = (X - \alpha)(X - \alpha^2)(X - \alpha^3)(X - \alpha^4)$ , ou encore tels que  $P(\alpha) = P(\alpha^2) = P(\alpha^3) = P(\alpha^4) = 0$ . L'application de codage a été expliquée en 10.3.4. On écrit

$$X^4 m(X) = q(X)(X - \alpha)(X - \alpha^2)(X - \alpha^3)(X - \alpha^4) + r_0 + r_1 X + r_2 X^2 + r_3 X^3,$$

et on code  $m$  par  $(-r_0, -r_1, -r_2, -r_3, m)$ , où apparaissent en tête quatre octets de contrôle. On notera que la relation de division ci-dessus s'exprime simplement par un système d'équations linéaires entre les  $r_i$  :

$$\begin{aligned} r_0 + r_1 \alpha + r_2 \alpha^2 + r_3 \alpha^3 &= \alpha^4 m(\alpha) \\ r_0 + r_1 \alpha^2 + r_2 \alpha^4 + r_3 \alpha^6 &= \alpha^8 m(\alpha^2) \\ r_0 + r_1 \alpha^3 + r_2 \alpha^6 + r_3 \alpha^9 &= \alpha^{12} m(\alpha^3) \\ r_0 + r_1 \alpha^4 + r_2 \alpha^8 + r_3 \alpha^{12} &= \alpha^{16} m(\alpha^4). \end{aligned}$$

### EXERCICE 12.2. [A] — Pourquoi ?

Le codage pour  $C_2$  est similaire ; on remplace simplement les dimensions 24 et 28 par 28 et 32 respectivement.

### Décodage de $C_2$

On part donc d'un polynôme  $P \in \mathbf{F}_{256}[X]$  de degré  $< 32$ . Le code  $C_2$  est formé des polynômes de ce type qui s'annulent pour  $X = \alpha^i$  avec  $1 \leq i \leq 4$ . Il s'agit de décider dans lequel des cas suivants on se trouve :

- a) il n'y a aucune d'erreur : on a  $P \in C_2$  ;
- b) il y a une seule erreur : il existe  $j$  avec  $0 \leq j < 32$  et  $\lambda \in \mathbf{F}_{256}^*$  avec  $P - \lambda X^j \in C_2$  ;
- c) il y a plus qu'une erreur.

Dans les deux premiers cas, on obtiendra  $m$  en prenant les 28 derniers coefficients de  $P$  pour le cas a), de  $P - \lambda X^j$  pour le cas b). Dans le cas c), on considérera le polynôme  $m$  comme totalement inconnu.

L'algorithme de décision se base sur la considération des quatre *syndromes*

$$s_1 = P(\alpha), \quad s_2 = P(\alpha^2), \quad s_3 = P(\alpha^3), \quad s_4 = P(\alpha^4).$$

Dans le cas a), tous les  $s_i$  sont nuls. Dans le cas b), on a

$$s_1 = \lambda \alpha^j, \quad s_2 = \lambda \alpha^{2j}, \quad s_3 = \lambda \alpha^{3j}, \quad s_4 = \lambda \alpha^{4j},$$

de sorte qu'aucun des  $s_i$  n'est nul. L'algorithme de décision est le suivant :

- ▷ si les  $s_i$  sont tous nuls, on est dans le cas a) ;
- ▷ si  $s_1$  et  $s_2$  sont non nuls, et si l'on a  $s_2^2 = s_1 s_3$  et  $s_1 s_4 = s_2 s_3$ , on est dans le cas b), et on calcule  $\lambda$  et  $j$  par  $\lambda = s_1^2 / s_2$  et  $\alpha^j = s_2 / s_1$  ;
- ▷ dans les autres cas, on est dans le cas c).

### EXERCICE 12.3. [A] — Vérifier.

## Décodage de $C_1$

En ce qui concerne  $C_1$ , le problème de décodage est le suivant. On considère un polynôme  $P \in \mathbf{F}_{256}[X]$ , de degré  $< 28$ , supposé multiple du produit  $(X - \alpha)(X - \alpha^2)(X - \alpha^3)(X - \alpha^4)$ , dont certains coefficients, au nombre d'au plus quatre, sont inconnus et qu'il s'agit de déterminer. Écrivons donc

$$P(X) = P^{\text{connu}}(X) + \lambda_1 X^{n_1} + \cdots + \lambda_e X^{n_e},$$

avec  $e \leq 4$ . Il s'agit de déterminer les  $\lambda_i$ , pour  $1 \leq i \leq e$ . Mais on doit avoir  $P(\alpha^i) = 0$  pour  $1 \leq i \leq e$ , ce qui donne un système linéaire en les  $\lambda_i$ , dont on vérifie que le déterminant est un déterminant de Vandermonde non nul.

EXERCICE 12.4. [B] — Vérifier.

### § 12.3. Au delà du code CIRC

Tout cela s'appliquerait identiquement pour des informations purement numériques (ce qui est le cas des CD-ROM). Mais il s'agit ici de sons, de sorte que les échantillons successifs ne se succèdent pas arbitrairement (du moins en principe!). On tire profit de cela pour introduire une couche supplémentaire de sécurité de la façon suivante.

Rappelons qu'une trame initiale est formée de six couples d'enregistrements successifs, six à gauche, six à droite. Notons-les par exemple  $G_i$  et  $D_i$  pour  $i = 1, \dots, 6$ . Le premier codage ajoute à ces douze doubles octets quatre octets « de parité » pour former des trames de 28 octets. Notons  $P$  et  $P'$  les deux doubles octets de parité.

Au lieu de former cette trame dans l'ordre « naturel »

$$(P, P', G_1, D_1, \dots, G_6, D_6),$$

on entrelace les doubles octets comme suit :

$$(G_1, G_3, G_5, D_1, D_3, D_5, P, P', G_2, G_4, G_6, D_2, D_4, D_6).$$

De cette façon, deux enregistrements consécutifs de la même voie sont séparés d'au moins 12 octets. Après l'entrelacement à retard 4 du code CIRC, cela donne en eux un espace de 48 trames, soit 8,5 millimètres.

Il suffit alors de munir le lecteur de CD de la capacité d'interpoler un enregistrement manquant entre deux enregistrements connus (par exemple en prenant la moyenne, n'oublions pas qu'il s'agit d'un « trou » de moins de 1/40000 seconde) pour qu'il puisse ainsi reconstituer sans dégradation audible le son enregistré, malgré une destruction d'un secteur de près d'un centimètre de large sur le support. Et on voit bien qu'on pourrait aller beaucoup plus loin encore.

# Chapitre 13

## Codes de résidus quadratiques

Ce chapitre est consacré à une classe spéciale de codes linéaires cycliques, dont la longueur est un nombre premier, disons  $p$ , et dont la construction fait intervenir l’élévation au carré dans le corps fini  $\mathbf{F}_p$ . On prend  $p \equiv -1 \pmod{8}$  pour des raisons techniques. Pour  $p = 7$ , on retrouve sans surprise l’omniprésent code de Hamming.

Le cas suivant,  $p = 23$ , donne le *code de Golay* dont on a pu dire sans exagération qu’il est le plus bel objet combinatoire des mathématiques finies. Il permet en effet de construire nombre d’objets exceptionnels comme le système de Steiner de type  $(5, 8, 24)$  — et aussi les groupes de Mathieu  $M_{24}$  et  $M_{23}$  dont nous dirons quelques mots.

Dans le dernier paragraphe, nous abordons brièvement une connexion un peu surprenante entre la théorie des codes et celle des *réseaux* des espaces vectoriels.

### § 13.1. Codes de résidus quadratiques

#### 13.1.1. Définition générale

Ce sont des cas particulièrement agréables de codes linéaires cycliques sur un corps  $\mathbf{F}_q$ , dont la longueur est un nombre premier  $p$  premier à  $q$ . Par conséquent, *contrairement à un usage précédent*,  $q$  est une puissance d’un nombre premier *distinct* de  $p$ . Comme on l’a vu en 10.4, construire un code de longueur  $p$  sur  $\mathbf{F}_q$  revient à déterminer une partie  $\Sigma$  de  $\mathbf{Z}/p\mathbf{Z} = \mathbf{F}_p$ , stable par multiplication par  $q$ . Or, si  $q$  est un résidu quadratique modulo  $p$ , les résidus et les non-résidus modulo  $p$  forment deux telles parties, chacune de  $(p-1)/2$  éléments, ce qui donne dans  $\mathbf{F}_q[X]$  une décomposition de la forme

$$X^p - 1 = (X - 1)g(X)g'(X)$$

où  $g$  et  $g'$  sont tous deux de degré  $(p-1)/2$ . Rappelons que si  $\alpha$  est une racine primitive  $p$ -ième de l’unité sur  $\mathbf{F}_q$  — et appartient donc à  $\mathbf{F}_{qr}$  pour un  $r$  que l’on sait déterminer d’après la proposition 9.28 — on a

$$g(X) = \prod_{(\frac{i}{p})=1} (X - \alpha^i), \quad g'(X) = \prod_{(\frac{i}{p})=-1} (X - \alpha^i).$$

Lorsqu'on prend  $g$  comme générateur, on obtient un code linéaire  $C$  sur  $\mathbf{F}_q$  de longueur  $p$  et de dimension  $(p+1)/2$ , dit *code de résidus quadratiques*, ou en abrégé code QR.

Ce code est formé, rappelons-le, de tous les polynômes de  $\mathbf{F}_q[X]$  de degré  $\leq p-1$ , qui sont multiples de  $g$ , ou de manière équivalente, admettent comme racines tous les  $\alpha^i$ , où  $i$  est un résidu quadratique modulo  $p$ . Notons au passage que le polynôme qui a tous ses coefficients égaux à 1, soit  $1 + \cdots + X^{p-1} = gg'$ , appartient à  $C$ .

**REMARQUE 13.1.** — On peut échanger  $g$  et  $g'$  en modifiant le choix de  $\alpha$ . Il suffit de remplacer  $\alpha$  par  $\alpha^r$  où  $r$  n'est pas un résidu quadratique.

Lorsqu'on suppose en outre que  $-1$  n'est pas un résidu quadratique (ce qui signifie que  $p$  est congru à 3 modulo 4), les non-résidus sont les opposés modulo  $p$  des résidus ; par conséquent, les racines de  $g'$  sont les inverses des racines de  $g$ , ce qui signifie que les polynômes  $g$  et  $g'$  sont réciproques l'un de l'autre :

$$g'(X) = X^{\frac{p-1}{2}} g(1/X).$$

Prenons par exemple  $q = 3$  et  $p = 11$ . Les carrés modulo 11 sont 1, 3, 4, 5 et 9, les non-carrés sont leurs opposés, la décomposition de  $X^{11} - 1$  est la suivante :

$$X^{11} - 1 = (X - 1)(X^5 + X^4 - X^3 + X^2 - 1)(X^5 - X^3 + X^2 - X - 1) \in \mathbf{F}_3[X].$$

Le code correspondant est le *code ternaire de Golay*<sup>1</sup>  $G_{11}$ , de longueur 11, de dimension 6.

**EXERCICE 13.1. [N]** — Exhiber les  $3^6$  mots du code  $G_{11}$ . Vérifier que sa distance minimale est 5, donc que  $G_{11}$  est 2-correcteur, et qu'il est parfait.

En dehors du cas  $q = 3$ ,  $p = 11$  cité ci-dessus, les exemples les plus intéressants de la situation précédente sont ceux où  $q = 2$ , cas auquel nous nous limitons maintenant. On doit donc avoir  $(\frac{2}{p}) = 1$ , soit  $p \equiv \pm 1 \pmod{8}$  (ce qui donne  $p = 7, 17, 23, \dots$ ). Si on veut en outre que  $(\frac{-1}{p}) = -1$ , on doit prendre  $p \equiv -1 \pmod{8}$ .

*Nous supposons désormais*  $q = 2$  et  $p \equiv -1 \pmod{8}$ . Il s'agira donc de codes binaires.

---

1. Les codes binaires « de Golay », que nous analyserons en détail un peu plus loin, ont été effectivement découverts par Golay lui-même. Le code ternaire a une origine totalement différente. Il a été découvert par un Finlandais nommé Juhani VIRTAKALLIO, qui cherchait à optimiser des paris à trois valeurs (gagné, nul, perdu) pour des groupes de 11 matchs de football et a publié dans un magazine spécialisé la liste des  $3^6$  combinaisons. On pourra lire cette histoire dans l'article d'Alexander BARG « *At the Dawn of the Theory of Codes* » dans *The Mathematical Intelligencer*, vol. 15 (1993), No. 1, pp. 20–26.

Le premier cas est  $p = 7$ , pour lequel les résidus sont 1, 2 et 4, et on retrouve le code de Hamming de type (7, 4, 3). Le second cas est  $p = 23$ , qui donne le code de Golay  $G_{23}$  que nous étudierons en détail un peu plus loin.

### 13.1.2. Idempotents des codes QR binaires

Pour les codes binaires de résidus quadratiques, la construction des idempotents (voir 10.4.5) prend une importance particulière, car ceux-ci sont particulièrement faciles à calculer :

**PROPOSITION 13.2.** — *Prenons  $q = 2$  et supposons  $p \equiv -1 \pmod{8}$ . Définissons les polynômes  $e(X)$  et  $e'(X)$  de  $\mathbf{F}_2[X]$ , de degré  $\frac{p-1}{2}$ , par*

$$e(X) = \sum_{\left(\frac{i}{p}\right)=1} X^i, \quad e'(X) = \sum_{\left(\frac{i}{p}\right)=-1} X^i.$$

*Alors  $e$  et  $e'$  sont idempotents modulo  $X^p - 1$  et ils sont multiples, l'un de  $g$  et l'autre de  $g'$ . On peut choisir la racine primitive  $\alpha$  de façon que  $e$  soit multiple de  $g$  et  $e'$  de  $g'$ .*

*Démonstration.* Notons  $\Sigma$  l'ensemble des résidus quadratiques modulo  $p$ . Puisque que l'on est à coefficients modulo 2 et que l'ensemble  $\Sigma$  des exposants de  $e$  est stable par multiplication par 2 modulo  $p$ , on a bien  $e^2 \equiv e \pmod{(X^p - 1)}$  :

$$e(X)^2 = \sum_{i \in \Sigma} X^{2i} \equiv \sum_{j \in \Sigma} X^j = e(X).$$

Mais cela implique  $e(\alpha)^2 = e(\alpha)$ , donc  $e(\alpha) \in \mathbf{F}_2$ . On a de même  $e'^2 = e'$  et  $e'(\alpha) \in \mathbf{F}_2$ .

On a par construction  $1 + e(X) + e'(X) = \sum_{0 < i < p} X^i = (X^p - 1)/(X - 1)$ . Puisque les non-résidus sont les opposés modulo  $p$  des résidus, on a aussi  $e'(X) \equiv e(X^{-1}) \pmod{(X^p - 1)}$ . Puisque  $\alpha$  est une racine  $p$ -ième de l'unité distincte de 1, ces deux relations impliquent  $e(\alpha) + e(\alpha^{-1}) = 1$ .

Soit  $s \in \Sigma$ . Alors l'application  $i \mapsto is \bmod p$  est une permutation de  $\Sigma$  et on a :

$$e(\alpha^s) = \sum_{i \in \Sigma} \alpha^{is} = \sum_{j \in \Sigma} \alpha^j = e(\alpha).$$

Les valeurs des  $e(\alpha^s)$  pour  $s \in \Sigma$  sont donc toutes les mêmes. Un calcul analogue montre qu'il en est de même pour les  $e(\alpha^{-s})$  pour  $s \in \Sigma$ . Mais on a vu que ces deux valeurs communes sont des éléments de  $\mathbf{F}_2$  de somme 1. Il a donc deux cas possibles : ou bien  $e(\alpha^s) = 0$  et  $e(\alpha^{-s}) = 1$  pour tout  $s \in \Sigma$ , ou bien c'est l'inverse. Dans le premier cas,  $e$  est multiple de  $g$  (et donc  $e'$  multiple de  $g'$ ) ; dans le deuxième cas, c'est l'inverse. Mais en changeant  $\alpha$  en  $\alpha^{-1}$ , on peut échanger les rôles de  $g$  et  $g'$ .  $\square$

Nous supposerons désormais  $\alpha$  choisi de façon que  $e(X)$  appartienne au code ; il en est donc un générateur. Ainsi, le code QR binaire de longueur  $p$

peut être défini de la manière totalement explicite suivante : on considère le mot  $\mathbf{e}$  de longueur  $p$  dont les bits égaux à 1 sont ceux dont le numéro est un carré (non nul) modulo  $p$  ; alors les translatés cycliques de  $\mathbf{e}$  engendrent un code de longueur  $p$  de dimension  $\frac{p+1}{2}$ .

Prenons par exemple  $p = 7$ . On a alors  $e = X + X^2 + X^4 = X(1 + X + X^3)$  et  $e' = X^3 + X^5 + X^6 = X^3(1 + X^2 + X^3)$ , et on retrouve encore une fois le code de Hamming.

### 13.1.3. Codes QR binaires étendus

Supposons toujours  $p \equiv -1 \pmod{8}$ . Alors  $g$  est de degré  $(p-1)/2$ , donc impair, et le code  $C$  contient des mots de poids impair. On peut alors en prendre l'extension paire  $\bar{C}$ . On obtient ainsi le *code binaire étendu de résidus quadratiques*, qui est de longueur  $p+1$  (divisible par 8) et de dimension moitié  $(p+1)/2$ .

**PROPOSITION 13.3.** — Soit  $p$  un nombre premier avec  $p \equiv -1 \pmod{8}$  et soit  $\bar{C}$  le code QR binaire étendu de longueur  $p+1$ . Alors  $\bar{C}$  est auto-orthogonal.

*Démonstration.* Rappelons que l'on a  $X^p - 1 = (X-1)g(X)g'(X)$ , où  $g'$  est le polynôme réciproque de  $g$ . D'après 10.18, le code  $C^\perp$  orthogonal à  $C$  a comme générateur le polynôme réciproque de  $(X^n - 1)/g(X) = (X-1)g'(X)$ , c'est-à-dire le polynôme  $(X-1)g(X)$ . Mais cela signifie exactement que  $C^\perp$  est le sous-code pair de  $C$ . On applique alors le corollaire 6.15.  $\square$

**COROLLAIRE 13.4.** — Tous les mots de  $\bar{C}$  sont de poids multiples de 4.

*Démonstration.* Le code  $C$  est engendré par les translatés cycliques du mot associé à  $e$ , qui est de poids  $\frac{p-1}{2} \equiv -1 \pmod{4}$ . Par conséquent,  $\bar{C}$  est engendré par des mots dont le poids est multiple de 4. Il suffit donc d'appliquer le lemme suivant :  $\square$

**LEMME 13.5.** — Dans un code binaire auto-orthogonal engendré par des mots de poids multiples de 4, tous les mots sont de poids multiples de 4.

*Démonstration.* Soient en effet  $\mathbf{m}$  et  $\mathbf{n}$  deux mots du code. On a  $w(\mathbf{m} + \mathbf{n}) = w(\mathbf{m}) + w(\mathbf{n}) - 2w(\mathbf{m} \cdot \mathbf{n})$ , mais  $w(\mathbf{m} \cdot \mathbf{n})$  est pair, puisque  $\langle \mathbf{m} | \mathbf{n} \rangle = 0$ . Ainsi on a  $w(\mathbf{m} + \mathbf{n}) \equiv w(\mathbf{m}) + w(\mathbf{n}) \pmod{4}$ . L'ensemble des mots du code de poids multiple de 4 est donc stable par addition. S'il contient des générateurs, c'est nécessairement le code tout entier.  $\square$

**COROLLAIRE 13.6.** — Les poids des mots de  $C$  sont congrus à 0 ou  $-1$  modulo 4.

### 13.1.4. Automorphismes d'un code QR binaire étendu

Appliquant la construction expliquée en 7.1.1, on peut identifier chaque mot du code  $C$  à une partie de  $\mathbf{F}_p$ . Nous noterons  $\infty$  le bit de parité sup-

plémentaire de  $\overline{C}$ , identifiant ainsi les mots de  $\overline{C}$  à des parties de la *droite projective* sur  $\mathbf{F}_p$

$$\mathbf{P}^1(\mathbf{F}_p) = \mathbf{F}_p \cup \{\infty\}.$$

Nous noterons  $\oplus$  la différence symétrique des parties (donc l'addition des mots), réservant la notation  $+$  à l'addition dans  $\mathbf{F}_p$ . Notons  $\Sigma$  l'ensemble des résidus quadratiques, de sorte que  $-\Sigma$  est l'ensemble des non-résidus. Nous avons vu en 13.1.2 que  $\Sigma$  est un générateur de  $C$ . Les éléments de  $C$  s'obtiennent donc par combinaison linéaire (c'est-à-dire par différence symétrique) à partir des translatés cycliques de  $\Sigma$ , c'est-à-dire des  $p$  éléments  $m_r = \Sigma + r$ , où  $r$  parcourt  $\mathbf{F}_p$ . Tous les  $m_r$  étant impairs, on doit leur adjoindre  $\infty$  pour former les éléments correspondants de  $\overline{C}$ , parties paires de  $\mathbf{P}^1(\mathbf{F}_p)$ :

$$\overline{m}_r = (\Sigma + r) \cup \{\infty\} = (\Sigma + r) \oplus \{\infty\}, \quad r \in \mathbf{F}_p.$$

Notons au passage, ce qui nous servira, que la partie pleine de  $\mathbf{F}_p$  est un mot de  $C$ , d'ailleurs impair, et donc que la partie pleine de  $\mathbf{P}^1(\mathbf{F}_p)$  est un mot de  $\overline{C}$ .

Nous allons maintenant nous intéresser aux automorphismes de  $C$  et  $\overline{C}$ . Ce sont par définition les permutations de  $\mathbf{F}_p$  (respectivement de  $\mathbf{P}^1(\mathbf{F}_p)$ ) qui permutent entre elles les parties formant le code  $C$  (respectivement  $\overline{C}$ ).

On a d'abord comme automorphismes de  $C$  les translations  $i \mapsto i + s$ , qui appliquent chaque  $m_r$  sur  $m_{r+s}$  (ce qui signifie que l'on a bien affaire à un code cyclique), et aussi les homothéties dont le rapport est un carré  $i \mapsto t^2 i$ , qui appliquent chaque  $m_r$  sur  $m_{t^2 r}$ . Plus généralement, une transformation  $f : i \mapsto t^2 i + s$  applique  $m_r$  sur  $m_{f(r)}$ , donc est un automorphisme du code  $C$ . Son extension à  $\mathbf{P}^1(\mathbf{F}_p)$ , qui laisse fixe le point à l'infini, applique  $\overline{m}_r$  sur  $\overline{m}_{f(r)}$ , donc est un automorphisme de  $\overline{C}$ .

Nous allons maintenant considérer des automorphismes qui ne laissent pas fixe le point à l'infini. On note  $\mathbf{PSL}_2(\mathbf{F}_p)$  le groupe formé des transformations de  $\mathbf{P}^1(\mathbf{F}_p)$  de la forme

$$f(x) = \frac{ax + b}{cx + d}, \quad a, b, c, d \in \mathbf{F}_p, \quad ad - bc = 1.$$

Un élément typique en est  $T(x) = -\frac{1}{x}$  qui échange 0 et  $\infty$  et aussi échange carrés et non-carrés.

**EXERCICE 13.2. [A]** — Les  $f \in \mathbf{PSL}_2(\mathbf{F}_p)$  qui laissent fixe le point à l'infini, c'est-à-dire telles que  $c = 0$ , sont exactement les transformations considérées précédemment.

**EXERCICE 13.3. [B]** — L'application qui, à une matrice  $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$  de déterminant

inversible, associe la transformation  $f(x)$  ci-dessus est un homomorphisme de groupes. Quel est son noyau ?

**EXERCICE 13.4. [B]** — Pour qu'une transformation homographique  $x \mapsto (ax + b)/(cx + d)$ , avec  $ad - bc \neq 0$  soit de la forme précédente, il faut et il suffit que  $ad - bc$  soit un carré. En particulier  $x \mapsto 1/x$  n'appartient pas à  $\mathbf{PSL}_2(\mathbf{F}_p)$ .

**EXERCICE 13.5. [C]** — Combien le groupe  $\mathbf{PSL}_2(\mathbf{F}_p)$  a-t-il d'éléments ? Déterminer ce groupe pour  $p = 2$  et  $p = 3$ .

**THÉORÈME 13.7.** — *Les éléments de  $\mathbf{PSL}_2(\mathbf{F}_p)$  sont des automorphismes de  $\overline{\mathbb{C}}$ .*

*Démonstration.* Si  $c = 0$ , on a  $f(x) = a^2x + ab$  et  $f$  est bien un automorphisme. Si  $c \neq 0$ , on a

$$f(x) = \frac{a}{c} + \frac{1}{c^2} \frac{-1}{x + d/c} = \frac{a}{c} + c^{-2} T(x + d/c).$$

Ainsi,  $f$  s'exprime à l'aide de deux translations, d'une homothétie de rapport carré et de la transformation  $T(x) = -1/x$ . Il suffit donc de prouver que  $T$  laisse stable le code  $\overline{\mathbb{C}}$ , ce qui résulte de la proposition suivante.  $\square$

**PROPOSITION 13.8.** — *Notons  $\mathbf{1} \in \overline{\mathbb{C}}$  la partie pleine. On a*

- (i)  $T(\overline{m}_0) = \overline{m}_0 \oplus \mathbf{1}$ ,
- (ii)  $T(\overline{m}_r) = \overline{m}_0 \oplus \overline{m}_{T(r)}$  si  $r$  est un carré non nul,
- (iii)  $T(\overline{m}_r) = \overline{m}_0 \oplus \overline{m}_{T(r)} \oplus \mathbf{1}$  si  $r$  n'est pas un carré.

*Démonstration.* Notons d'abord que  $\mathbf{1} = \Sigma \oplus -\Sigma \oplus \{0\} \oplus \{\infty\}$ . On a  $\overline{m}_0 = \Sigma \oplus \{\infty\}$ , donc

$$T(\overline{m}_0) = T(\Sigma) \oplus \{T(\infty)\} = (-\Sigma) \oplus \{0\},$$

d'où (i) résulte immédiatement.

Démontrons (ii). Par construction  $\overline{m}_r$  est la partie formée de  $\infty$  et des  $r + i$  avec  $i \in \Sigma$ . Par conséquent  $T(\overline{m}_r)$  est la partie formée de  $0$  et des  $-1/(r + i)$  avec  $i \in \Sigma$ . Puisque  $r$  est un carré,  $r + i$  ne peut jamais être nul (car cela donnerait  $r = -i \in -\Sigma$ ) ; ainsi  $T(\overline{m}_r)$  est formée de  $0$  et des  $s \in \mathbf{F}_p^*$  avec  $-r - 1/s \in \Sigma$ , ce qui s'écrit aussi  $(rs + 1)/s \in -\Sigma$ . On peut distinguer deux cas suivant que  $s \in \Sigma$  ou  $s \in -\Sigma$ . Si  $s \in \Sigma$ , alors la condition signifie  $rs + 1 \in -\Sigma$  ; sinon elle signifie  $rs + 1 \in \Sigma$ . Par conséquent,  $T(\overline{m}_r)$  est la réunion de trois parties, à savoir  $\{0\}$ ,  $A = \{s \in \Sigma \mid rs + 1 \in -\Sigma\}$  et  $B = \{s \in -\Sigma \mid rs + 1 \in \Sigma\}$ . D'autre part,  $\overline{m}_{T(r)}$  est la partie formée de  $\infty$  et des  $s = j - 1/r$ , avec  $j \in \Sigma$ , c'est-à-dire des  $s \in \mathbf{F}_p$  avec  $s + 1/r \in \Sigma$  ; mais puisque  $r$  est un carré, cela s'écrit aussi  $rs + 1 \in \Sigma$ . Distinguant trois cas, suivant que  $s$  est nul, un carré ou un non-carré, on voit que  $\overline{m}_{T(r)}$  est formée de  $\infty$ , de  $0$ , de la partie  $B$  ci-dessus et de  $C = \{s \in \Sigma \mid rs + 1 \in \Sigma\}$ . En définitive, on obtient

$$\begin{aligned} T(\overline{m}_r) \oplus \overline{m}_{T(r)} &= (\{0\} \oplus A \oplus B) \oplus (\{0\} \oplus \{\infty\} \oplus B \oplus C) \\ &= A \oplus C \oplus \{\infty\} = \Sigma \oplus \{\infty\} = \overline{m}_0. \end{aligned}$$

La partie (iii) se démontre par la même méthode.  $\square$

## EXERCICE 13.6. [B] — Vérifier (iii).

REMARQUE 13.9. — Dans le cas du code de Hamming étendu ( $p = 7$ ), il y a d'autres automorphismes que les précédents (voir 7.1.2). Avec celui du code de Golay ( $p = 23$ , voir 13.2.2), ce sont les deux seuls cas connus de ce phénomène.

**13.1.5. Distance minimale d'un code QR binaire**

PROPOSITION 13.10. — Soit  $p$  un nombre premier avec  $p \equiv -1 \pmod{8}$ . La distance minimale  $d$  du code QR binaire de longueur  $p$  est impaire et satisfait à l'inégalité

$$d^2 - d + 1 \geq p.$$

*Démonstration.* Le fait que  $d$  soit impaire résulte du lemme 7.22. Soit  $a(X) = \sum a_i X^i$  un mot de code de poids  $d$ . C'est un multiple du polynôme générateur  $g(X)$ . Le polynôme réciproque  $a'(X) = \sum a_i X^{p-1-i}$  est un multiple de  $g'(X)$  et est aussi de poids  $d$ . Dans le produit  $a(X)a'(X)$ ,  $d$  des  $d^2$  produits de monômes sont égaux à 1, de sorte que  $aa'$  contient au plus  $d^2 - d + 1$  monômes. Par réduction modulo  $X^p - 1$ , le nombre de monômes ne peut que diminuer et l'unique polynôme  $b$  de degré  $< p$  congru à  $aa'$  modulo  $X^p - 1$  est donc de poids  $\leq d^2 - d + 1$ . Nous allons prouver que  $b = gg'$ , ce qui achèvera la démonstration, puisque  $gg' = X^{p-1} + \dots + 1$  est de poids  $p$ .

On a  $(X - 1)gg' = X^p - 1$ , ce qui entraîne  $Xgg' \equiv gg' \pmod{(X^p - 1)}$  et par conséquent  $cgg' \equiv c(1)gg' \pmod{(X^p - 1)}$  pour tout polynôme  $c$ . Ainsi, tout multiple  $m$  de  $gg'$  tel que  $m(1) = 1$  est congru à  $gg'$  modulo  $X^p - 1$ . Il nous suffit maintenant de remarquer que  $a$  et  $a'$  étant de poids *impairs*, on a  $a(1) = a'(1) = 1$ .  $\square$

On dispose ainsi des renseignements suivants sur les distances minimales  $d$  et  $\bar{d}$  de  $C$  et  $\bar{C}$  respectivement.

- ▷  $d$  est impaire et  $\bar{d} = d + 1$ ,
- ▷  $d$  est au plus égale à  $\frac{p-1}{2}$ ,
- ▷  $d$  est supérieure à  $\sqrt{p}$ ,
- ▷  $\bar{d}$  est divisible par 4.

Pour des  $p$  petits, cela permet de déterminer  $d$  et  $\bar{d}$ . Ainsi, pour  $p = 7$ , on a nécessairement  $d = 3$ ; on a d'ailleurs affaire au code de Hamming de paramètres  $(7, 4, 3)$  et à son extension paire, de paramètres  $(8, 4, 4)$ . Pour  $p = 23$ , on a nécessairement  $d = 7$  ou  $d = 11$ ; mais le second cas est exclu (par exemple parce qu'on exhibera des mots de poids  $< 11$ , mais beaucoup plus simplement parce que les boules de rayon 5 ont beaucoup trop d'éléments pour pouvoir être disjointes) et on obtient les deux codes binaires de Golay de paramètres  $(23, 11, 7)$  et  $(24, 12, 8)$ , que nous allons examiner en détails.

### § 13.2. Les codes binaires de Golay $G_{23}$ et $G_{24}$

Le code de Golay  $G_{23}$  est le code binaire de résidus quadratiques de longueur 23. Explicitons sa construction. Si l'on prend  $n = 23$  dans la construction des codes cycliques, on a modulo  $n$  deux cycles de longueur 11, soit

$$1 \mapsto 2 \mapsto 4 \mapsto 8 \mapsto 16 \mapsto 9 \mapsto 18 \mapsto 13 \mapsto 3 \mapsto 6 \mapsto 12 \mapsto 1$$

et le cycle formé des opposés des précédents. Cela donne deux classes cyclotomiques

$$\begin{aligned}\Sigma &= \{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18\}, \\ \Sigma' &= \{5, 7, 10, 11, 14, 15, 17, 19, 20, 21, 22\},\end{aligned}$$

formées en fait respectivement des résidus quadratiques et des non-résidus modulo 23. On sait donc que  $X^{23} - 1$  se décompose dans  $\mathbf{F}_2[X]$  en produit de  $X - 1$  et de deux polynômes irréductibles de degré 11.

On a par ailleurs modulo  $X^{23} - 1$  les idempotents (voir la proposition 13.2)

$$\begin{aligned}e(X) &= X + X^2 + X^3 + X^4 + X^6 + X^8 + X^9 + X^{12} + X^{13} + X^{16} + X^{18}, \\ e'(X) &= X^5 + X^7 + X^{10} + X^{11} + X^{14} + X^{15} + X^{17} + X^{19} + X^{20} + X^{21} + X^{22}.\end{aligned}$$

Prenant les pgcd de  $e(X)$  et  $e'(X)$  avec  $X^{23} - 1$  (on écrit indifféremment + ou -), on obtient les décompositions  $e(X) = (X + X^3 + X^7)g(X)$  avec

$$g(X) = 1 + X + X^5 + X^6 + X^7 + X^9 + X^{11},$$

et  $e'(X) = (X^5 + X^9 + X^{11})g'(X)$  avec

$$g'(X) = 1 + X^2 + X^4 + X^5 + X^6 + X^{10} + X^{11},$$

et on a

$$X^{23} - 1 = (X - 1)g(X)g'(X) \in \mathbf{F}_2[X].$$

On a ainsi identifié la décomposition de  $X^{23} - 1$  en polynômes irréductibles sur  $\mathbf{F}_2$ . Notons  $\alpha$  une racine de  $g(X)$ ; elle engendre le corps  $R_{23}(\mathbf{F}_2) = \mathbf{F}_{2048}$ . D'après ce qu'on a vu précédemment, on a

$$g(X) = \prod_{i \in \Sigma} (X - \alpha^i), \quad g'(X) = \prod_{i \in \Sigma'} (X - \alpha^i).$$

Les racines de  $g'$  sont exactement les inverses des racines de  $g$ , de sorte que  $g$  et  $g'$  sont deux polynômes *réciproques*:  $g'(X) = X^{11}g(1/X)$ , ce qu'on vérifie sur les formes explicites.

**DÉFINITION 13.11.** — *Le code de Golay de longueur 23 est le code binaire cyclique  $G_{23}$  de générateur  $g$ .*

Il est de dimension 12. C'est par construction le code BCH binaire généralisé de longueur 23 et de distance assignée (et distance de Bose) 5. Rappelons en la définition. On considère les espaces vectoriels  $E_{12}$  et  $E_{23}$  des polynômes de degré au plus égal à 11 et 22 respectivement et, à chaque élément  $a(X) \in E_{12}$ , on associe  $a(X)g(X) \in E_{23}$ . Alors, les  $a(X)g(X)$  forment le code  $G_{23} \subset E_{23}$ . Autrement dit, en passant à la base des monômes,  $G_{23}$  est le sous-espace vectoriel de dimension 12 de  $\mathbf{F}_2^{23}$  dont une base est formée du vecteur  $[1100011101010000000000]$  et de ses onze translatés à droite.

On pourrait aussi prendre comme générateur le vecteur correspondant à l'idempotent  $e(X)$ . En termes plus « géométriques », cela donne la définition équivalente :

**DÉFINITION 13.12.** — *Le code de Golay  $G_{23}$  est le code linéaire de parties de  $\mathbf{F}_{23}$  engendré par la partie  $\Sigma$  formée des résidus quadratiques et toutes ses translatées  $\Sigma + i$ ,  $i \in \mathbf{F}_{23}$ .*

Le code  $G_{23}$  est impair, car engendré par des mots de poids 7 (ou 11 selon la définition choisie), et on peut considérer son extension paire.

**DÉFINITION 13.13.** — *Le code de Golay de longueur 24 est l'extension paire  $G_{24}$  du code  $G_{23}$ .*

La distance de Bose du code  $G_{23}$  est égale à 5, mais sa distance minimale vaut en fait 7. Plus précisément :

**THÉORÈME 13.14.** — *a) Le code de Golay  $G_{23}$  est de distance minimale 7. C'est un code 3-correcteur parfait, de paramètres  $(23, 12, 7)$ .*

*b) Le code  $G_{24}$  est auto-orthogonal et de paramètres  $(24, 12, 8)$ . Tous ses poids sont multiples de 4.*

*c) Les mots de poids 7 de  $G_{23}$  forment un système de Steiner (voir 7.5.1) de type  $(4, 7, 23)$ . Ils engendrent  $G_{23}$ .*

*d) Les mots de poids 8 de  $G_{24}$  forment un système de Steiner de type  $(5, 8, 24)$ . Ils engendrent  $G_{24}$ .*

*Démonstration.* On a vu que  $G_{24}$  est auto-orthogonal (proposition 13.3) et que les poids de ses mots sont multiples de 4 (corollaire 13.4). Les poids des mots de  $G_{23}$  sont donc de la forme  $4r - 1$  ou  $4r$ , et sa distance minimale  $d$  est donc 3, 4 ou  $\geq 7$ .

Mais on a  $d > 4$ . Cela résulte par exemple de la proposition 10.21 puisque  $\Sigma$  contient la suite  $\{1, 2, 3, 4\}$ ; cela résulte aussi de la minoration  $d \geq \sqrt{23}$ . On a par conséquent  $d \geq 7$  et  $G_{23}$  est 3-correcteur. Or le nombre des éléments d'une boule de rayon 3 dans  $\mathbf{F}_2^{23}$  est

$$1 + \binom{23}{1} + \binom{23}{2} + \binom{23}{3} = 1 + 23 + 253 + 1771 = 2048 = 2^{11}.$$

Comme  $G_{23}$  possède  $2^{12}$  éléments, et l'espace ambiant  $2^{23}$  éléments, le code est 3-correcteur parfait. Cela achève de prouver *a)* et *b)*. Les assertion *c)* et *d)* résultent alors de la proposition 7.20.  $\square$

**REMARQUE 13.15.** — Les codes  $G_{23}$  et  $G_{24}$  ont été découverts par Golay en 1949. Le code  $G_{24}$  a été utilisé lors des missions Voyager 1 et 2 en 1979 et 1980 pour transmettre les images en couleurs de Jupiter et de Saturne.

**PROPOSITION 13.16.** — *La distribution des poids des mots du code  $G_{23}$  est donnée par le tableau suivant :*

poids	0	7	8	11	12	15	16	23
nombre de mots	1	253	506	1288	1288	506	253	1

*La distribution des poids des mots du code  $G_{24}$  est donnée par le tableau suivant :*

poids	0	8	12	16	24
nombre de mots	1	759	2576	759	1

*Démonstration.* Notons  $a_i$  (resp.  $b_i$ ) le nombre de mots de poids  $i$  dans  $G_{23}$  (resp.  $G_{24}$ ). On sait déjà que  $b_i$  est nul si  $i$  n'est pas multiple de 4, que  $b_0 = 1$  et  $b_4 = 0$ . On sait aussi que  $a_{4j-1} + a_{4j} = b_{4j}$  et que la somme des  $a_i$ , comme celle des  $b_i$ , est égale à  $2^{12} = 4096$ .

Par ailleurs, notons  $\mathbf{1}_{23}$  le mot de  $G_{23}$  dont tous les bits sont égaux à 1. Pour tout mot  $\mathbf{m}$  de  $G_{23}$ ,  $\mathbf{1}_{23} + \mathbf{m}$  est un mot de  $G_{23}$  et on a  $w(\mathbf{1}_{23} + \mathbf{m}) = 23 - w(\mathbf{m})$ . On en tire aussitôt  $a_{23-i} = a_i$ . De même, on a  $b_{24-i} = b_i$ .

Enfin, d'après le corollaire 7.21, on a

$$a_7 = \frac{23 \times 22 \times 21 \times 20}{7 \times 6 \times 5 \times 4} = 253, \quad b_8 = \frac{24 \times 23 \times 22 \times 21 \times 20}{8 \times 7 \times 6 \times 5 \times 4} = 759.$$

On conclut alors sans difficultés.  $\square$

**EXERCICE 13.7. [A]** — Vérifier.

**EXERCICE 13.8. [N]** — Vérifier les résultats précédents par énumération explicite.

### 13.2.1. Détermination des codes parfaits

Le théorème qui va suivre porte sur tous les codes, non nécessairement linéaires, ce qui nécessite un tout petit préliminaire.

Considérons un code  $C \subset \mathbb{F}_q^n$  et un mot  $\mathbf{a} \in \mathbb{F}_q^n$ . Le *code translaté* de  $C$  par  $\mathbf{a}$  est l'ensemble  $C + \mathbf{a}$  des mots  $\mathbf{m} + \mathbf{a}$ , pour  $\mathbf{m} \in C$ . Mais la translation préserve les distances, puisque

$$d(\mathbf{m} + \mathbf{a}, \mathbf{m}' + \mathbf{a}) = w((\mathbf{m} + \mathbf{a}) - (\mathbf{m}' + \mathbf{a})) = w(\mathbf{m} - \mathbf{m}') = d(\mathbf{m}, \mathbf{m}').$$

Le code translaté  $C + \mathbf{a}$  a donc exactement les mêmes paramètres que  $C$ , sera parfait si  $C$  est parfait, etc.

Si  $C$  est linéaire, il contient le mot nul. Dans le cas général, on peut toujours translater  $C$  par l'opposé d'un de ses éléments et obtenir un code qui contient le mot nul.

**EXERCICE 13.9. [A]** — Si  $C$  est linéaire,  $C + \mathbf{a}$  ne contient le mot nul que si  $\mathbf{a} \in C$  et on a alors  $C + \mathbf{a} = C$ .

Cela étant, les codes parfaits sont donnés par :

**THÉORÈME 13.17.** — Soit  $C$  un code  $t$ -correcteur parfait sur  $\mathbf{F}_q$ , non nécessairement linéaire, qui contient le mot nul. Notons  $n$  sa dimension. Alors, de quatre choses l'une :

- (i)  $C$  est nul, ou est plein, ou est un code binaire de répétition pure de longueur impaire ;
- (ii) on a  $t = 1$ ,  $n = (q^m - 1)/(q - 1)$  et  $C$  possède  $q^{n-m}$  éléments ;
- (iii) on a  $t = 2$ ,  $q = 3$  et  $C$  est équivalent au code ternaire de Golay  $G_{11}$  ;
- (iv) on a  $t = 3$ ,  $q = 2$  et  $C$  est équivalent au code binaire de Golay  $G_{23}$ .

Nous ne donnons ci-dessous que quelques très brèves indications sur la démonstration (difficile) de ce théorème, car nous ne disposons pas ici des outils techniques suffisants. Faisons d'abord quelques commentaires. On ne suppose pas dans l'énoncé que le code est linéaire, bien qu'il le soit automatiquement dans les cas (i), (iii) et (iv). Ensuite, les paramètres des codes du cas (ii) sont ceux des codes de Hamming, mais il y a des codes non linéaires qui ont les mêmes paramètres que les codes de Hamming.

La démonstration se décompose en deux.

Le difficile théorème de Tietäväinen-Van Lint<sup>2</sup> montre que les paramètres du code sont nécessairement, hormis les cas triviaux de (i), ceux donnés dans (ii), (iii) et (iv). Ensuite, on démontre que les codes de Golay sont les seuls qui possèdent les paramètres donnés dans (iii) et (iv). De façon plus précise, dans le cas de  $G_{23}$ , on a :

**THÉORÈME 13.18.** — Soit  $C$  un code binaire de longueur 23 et de distance minimum 7, contenant le mot nul. Alors  $C$  a au plus  $2^{12}$  éléments. Si  $C$  a  $2^{12}$  éléments, il est équivalent à  $G_{23}$ .

Ajoutant à  $C$  un bit de parité, il suffit de prouver l'énoncé analogue pour  $G_{24}$  :

**THÉORÈME 13.19.** — Soit  $C$  est un code binaire de longueur 24 et de distance minimum 8, contenant le mot nul. Alors  $C$  a au plus  $2^{12}$  éléments. Si  $C$  a  $2^{12}$  éléments, il est équivalent à  $G_{24}$ .

---

2. Voir [MacWilliams, Sloane], chapitre 6, §10, theorem 33.

*Démonstration.* Soit donc  $C$  un code binaire de longueur 24 et de distance minimale 8, à  $N$  éléments. Projetant  $C$  dans  $\mathbf{F}_2^{23}$ , on obtient un code de distance minimale 7, donc 3-correcteur. On a déjà remarqué (voir la démonstration du théorème 13.14) que les boules de rayon 3 dans  $\mathbf{F}_2^{23}$  ont  $2^{11}$  éléments. Donc ce code a au plus  $2^{23}/2^{11} = 2^{12}$  éléments, et on a  $N \leq 2^{12}$ .

Supposons que la borne soit atteinte (le code projeté est alors parfait). On peut prouver<sup>3</sup> que deux mots quelconques de  $C$  sont orthogonaux. Il en résulte que  $C$  est contenu dans son orthogonal  $C^\perp$ . Ce dernier a donc au moins  $2^{12}$  éléments. Mais cet orthogonal est par construction un code linéaire, qui est donc de dimension  $\geq 12$ . Ainsi,  $C$ , qui est orthogonal à  $C^\perp$ , est contenu dans l'orthogonal de ce dernier qui est linéaire de dimension  $\leq 24 - 12 = 12$ , donc a au plus  $2^{12}$  éléments. Il lui est donc égal et on a prouvé que  $C$  est linéaire.

On peut alors appliquer la proposition 7.20 : l'espace vectoriel  $C$  est engendré par ses mots de poids 8, qui forment un système de Steiner de type  $(5, 8, 24)$ . Le théorème résulte alors de l'unicité de ces systèmes :  $\square$

**THÉORÈME 13.20** (Witt, 1938). — *Tous les systèmes de Steiner de type  $(5, 8, 24)$  sont équivalents : soient  $I$  et  $I'$  deux ensembles à 24 éléments et soient  $S$  et  $S'$  des systèmes de Steiner de type  $(5, 8, 24)$  de parties de  $I$  et  $I'$  respectivement. Il existe une bijection de  $I$  sur  $I'$  qui transforme  $S$  en  $S'$ .*

La démonstration en est longue et de nature combinatoire.

### 13.2.2. Automorphismes du code de Golay : le groupe $M_{24}$

Les codes de Golay sont des objets combinatoires d'une richesse extrême. À titre d'échantillon, concluons sur un très (trop) bref exemple qui nous amène aux frontières de la théorie des groupes finis.

Rappelons que l'ensemble des bits du code de Golay  $G_{23}$  est naturellement  $\mathbf{Z}/23\mathbf{Z} = \mathbf{F}_{23}$  et qu'on a identifié le bit de parité au point à l'infini de  $\mathbf{P}^1(\mathbf{F}_{23})$ , de sorte que le code de Golay  $G_{24}$  est un code de parties de la droite projective  $\mathbf{P}^1(\mathbf{F}_{23})$ . Soit  $S$  le système de Steiner formé de ses mots de poids 8.

Pour une permutation  $\sigma$  de l'ensemble  $\mathbf{P}^1(\mathbf{F}_{23})$ , il est équivalent de dire qu'elle transforme  $G_{24}$  en lui-même, ou qu'elle transforme  $S$  en lui-même. Ces permutations forment le *groupe de Mathieu  $M_{24}$* . On a vu (théorème 13.7) que ce groupe contient le groupe  $\mathbf{PSL}_2(\mathbf{F}_{23})$ , formé des transformations

$$f(x) = \frac{ax + b}{cx + d}, \quad a, b, c, d \in \mathbf{F}_{23}, \quad ad - bc = 1.$$

Il est en fait beaucoup plus gros. En effet :

**THÉORÈME 13.21.** — a) *Le groupe  $M_{24}$  est d'ordre  $24 \cdot 23 \cdot 22 \cdot 21 \cdot 20 \cdot 48 = 244\,823\,040$ .*

3. Par une technique dont nous n'avons pas du tout parlé, voir [MacWilliams, Sloane], ch. 20, §6.

b) Le groupe  $M_{24}$  est 5 fois transitif : si on se donne deux familles formées chacune de cinq éléments distincts de  $\mathbf{P}^1(\mathbf{F}_{23})$ , soit  $(x_1, \dots, x_5)$  et  $(y_1, \dots, y_5)$ , il existe un élément  $\sigma \in M_{24}$  tel que  $\sigma(x_i) = y_i$  pour tout  $i$ .

EXERCICE 13.10. [C] — Combien y a-t-il de  $\sigma$  possibles dans b) ?

EXERCICE 13.11. [B] — Le groupe  $\mathbf{PSL}_2(\mathbf{F}_{23})$  est deux fois transitif.

La démonstration du théorème est trop compliquée pour que nous la donnions ici. Disons simplement qu'on commence par exhiber un automorphisme  $\tau$  de  $M_{24}$  qui n'appartient pas à  $\mathbf{PSL}_2(\mathbf{F}_{23})$  et qu'on utilise cet élément et les éléments de  $\mathbf{PSL}_2(\mathbf{F}_{23})$  pour prouver d'abord b).

Cet élément  $\tau$  est le suivant. Il laisse fixe le point à l'infini et agit séparément sur les carrés et les non-carrés :

$$\begin{aligned}\tau(x) &= x^4 \text{ si } x \text{ est un carré,} \\ \tau(x) &= 7x^4 = -(2x)^4 \text{ si } x \text{ n'est pas un carré.}\end{aligned}$$

Comme  $x^{11}$  vaut 1 ou  $-1$  selon que  $x$  est ou non un carré, on peut aussi écrire  $\tau(x) = 4x^4 - 3x^{15}$ . La permutation  $\tau$  est d'ordre 5, a quatre points fixes (0, 1, 5 et  $\infty$ ) et quatre cycles d'ordre 5.

EXERCICE 13.12. [C-N] — Vérifier. Montrer que  $\tau$  respecte bien le code  $G_{24}$ .

### 13.2.3. Les groupes de Mathieu

Les groupes de permutations qui sont transitifs ou 2-fois transitifs abondent. Ceux qui sont  $t$ -fois transitifs pour  $t \geq 3$  sont beaucoup moins nombreux. C'est d'ailleurs en cherchant des groupes multiplement transitifs que Émile MATHIEU a découvert (entre 1864 et 1873) les cinq groupes qui portent son nom.

On déduit de  $M_{24}$  deux autres groupes de Mathieu. Tout automorphisme de  $G_{23}$  s'étend en un automorphisme de  $G_{24}$  qui laisse fixe le point à l'infini. Le groupe d'automorphismes de  $G_{23}$  est donc le sous-groupe  $M_{23}$  de  $M_{24}$  formé des transformations qui laissent fixe ce point (d'ailleurs, choisir un autre point donnerait un groupe isomorphe). Il résulte de ce qui précède que  $M_{23}$  est 4-fois transitif et que son ordre est  $23 \cdot 22 \cdot 21 \cdot 20 \cdot 48 = 10\,200\,960$ . Recommençant avec un autre point, on obtient le groupe  $M_{22}$  qui est 3-fois transitif et d'ordre  $22 \cdot 21 \cdot 20 \cdot 48 = 443\,520$ .

Le code ternaire de Golay  $G_{11}$  sur le corps à 3 éléments permet de construire de la manière analogue deux autres groupes de Mathieu  $M_{12}$  et  $M_{11}$  qui sont respectivement 5-fois et 4-fois transitifs.

On a alors le théorème suivant, extrêmement difficile<sup>4</sup> :

---

4. Il résulte en effet de la classification des groupes finis simples.

**THÉORÈME 13.22.** — *Soit  $G$  un groupe de permutation de  $n$  lettres qui est  $t$  fois transitif, avec  $t \geq 4$ . Alors on est dans l'un des cas suivants :*

- a) *on a  $t = n$  et  $G$  est le groupe de toutes les permutations ;*
- b) *on a  $t = n - 2$  et  $G$  est le groupe de toutes les permutations paires ;*
- c) *on a  $t = 5$  et, soit  $n = 24$  et  $G = M_{24}$ , soit  $n = 12$  et  $G = M_{12}$  ;*
- d) *on a  $t = 4$  et, soit  $n = 23$  et  $G = M_{23}$ , soit  $n = 11$  et  $G = M_{11}$ .*

### § 13.3. Codes et réseaux

On fixe un entier naturel  $n$  et on munit l'espace vectoriel  $\mathbf{R}^n$  du produit scalaire  $\langle\langle \mathbf{m}|\mathbf{n} \rangle\rangle = 2\langle \mathbf{m}|\mathbf{n} \rangle$  double du produit usuel, donc tel que

$$\langle\langle (m_i)|(n_i) \rangle\rangle = 2 \sum_{i=0}^{n-1} m_i n_i.$$

Pour ce produit scalaire, les vecteurs de la base canonique sont donc par construction deux à deux orthogonaux et de carré scalaire égal à 2 (*et non pas 1, c'est là le point-clé*).

À chaque base  $(e_i)$  de  $\mathbf{R}^n$ , on associe la *base duale* formée des vecteurs  $e'_i$  définis comme suit : le produit scalaire  $\langle\langle e'_i|e \rangle\rangle_j$  est nul si  $j \neq i$  et égal à 1 si  $i = j$ . Par exemple, la base duale de la base canonique de  $\mathbf{R}^n$  est formée des moitiés des vecteurs de cette base.

#### 13.3.1. Réseaux

On appelle *réseaux* dans  $\mathbf{R}^n$  les sous-groupes additifs  $M$  de la forme suivante : on prend une base quelconque  $(e_i)$  de  $\mathbf{R}^n$  et  $M$  est l'ensemble des combinaisons linéaires à coefficients *entiers* des  $e_i$ . On dit alors que  $(e_i)$  est une base du réseau  $M$ . Par exemple  $\mathbf{Z}^n$  est un réseau de  $\mathbf{R}^n$ , qui possède la base canonique de  $\mathbf{R}^n$  comme base.

Nous utiliserons ci-dessous la proposition suivante, donnée ici sans démonstration :

**PROPOSITION 13.23.** — *Soient  $L_1$  et  $L_2$  deux réseaux de  $\mathbf{R}^n$  avec  $L_1 \subset L_2$ , et soit  $M$  un sous-groupe additif de  $\mathbf{R}^n$  avec  $L_1 \subset M \subset L_2$ . Alors  $M$  est un réseau.*

**REMARQUE 13.24.** — En fait, on peut prouver qu'un sous-groupe est un réseau si et seulement si il satisfait aux deux conditions suivantes :

- a) il engendre l'espace vectoriel  $\mathbf{R}^n$  ;
- b) il est un sous-espace discret au sens topologique.

Or l'inclusion  $L_1 \subset M$  implique a), et l'inclusion  $M \subset L_2$  implique b).

Si  $M$  est un réseau de  $\mathbf{R}^n$ , on appelle réseau *associé* de  $M$  l'ensemble  $M^\#$  des  $\mathbf{m} \in \mathbf{R}^n$  tels que  $\langle\langle \mathbf{m}|\mathbf{n} \rangle\rangle \in \mathbf{Z}$  pour tout  $\mathbf{n} \in M$ . C'est bien un réseau : en effet, si on suppose  $M$  construit comme ci-dessus à l'aide de la base  $(e_i)$

et si on désigne par  $(e'_i)$  la base duale,  $M^\#$  est l'ensemble des combinaisons linéaires à coefficients entiers des  $(e'_i)$ .

EXERCICE 13.13. [A] — Vérifier ce point.

Il résulte de ce qu'on vient de dire que le réseau associé à  $M^\#$  est  $M$ . En formules, on a  $(M^\#)^\# = M$ .

On démontre aussitôt les propriétés suivantes :

a) Le réseau associé à  $\mathbf{Z}^n$  est le réseau  $\frac{1}{2}\mathbf{Z}^n$  formé des vecteurs dont les coordonnées sont des entiers ou des demi-entiers.

b) Dire que le produit scalaire  $\langle\langle \mathbf{m} | \mathbf{n} \rangle\rangle$  de deux vecteurs quelconques  $\mathbf{m}$  et  $\mathbf{n}$  du réseau  $M$  est entier signifie que l'on a  $M \subset M^\#$ .

c) Si  $M$  et  $N$  sont deux réseaux tels que  $M \subset N$ , alors  $N^\# \subset M^\#$ .

EXERCICE 13.14. [A] — Démontrer ces différents points.

DÉFINITION 13.25. — *Un réseau contenu dans son associé est dit entier. Un réseau égal à son associé est dit unimodulaire. Un réseau  $M$  est dit pair si  $\langle\langle \mathbf{m} | \mathbf{m} \rangle\rangle$  est un entier pair pour tout élément  $\mathbf{m} \in M$ .*

En vertu de la relation

$$2\langle\langle \mathbf{m} | \mathbf{n} \rangle\rangle = \langle\langle \mathbf{m} + \mathbf{n} | \mathbf{m} + \mathbf{n} \rangle\rangle - \langle\langle \mathbf{m} | \mathbf{m} \rangle\rangle - \langle\langle \mathbf{n} | \mathbf{n} \rangle\rangle,$$

un réseau pair est automatiquement entier.

### 13.3.2. Réseau déduit d'un code binaire

Nous allons dans la suite considérer le réseau  $\frac{1}{2}\mathbf{Z}^n$ , son associé  $\mathbf{Z}^n$  et les réseaux intermédiaires

$$\mathbf{Z}^n \subset M \subset \frac{1}{2}\mathbf{Z}^n.$$

On notera qu'on a aussi, par passage aux associés,  $\mathbf{Z}^n \subset M^\# \subset \frac{1}{2}\mathbf{Z}^n$ .

Un élément  $\mathbf{m}$  de  $\frac{1}{2}\mathbf{Z}^n$  peut s'écrire  $\frac{1}{2}(m_i)$  où les  $m_i$  sont entiers. On notera alors  $\bar{\mathbf{m}}$  l'élément de  $\mathbf{F}_2^n$  défini par  $\bar{\mathbf{m}} = (m_i \bmod 2)$ . La condition  $\bar{\mathbf{m}} = 0$  équivaut donc à  $\mathbf{m} \in \mathbf{Z}^n$ .

L'application  $\mathbf{m} \mapsto \bar{\mathbf{m}}$  est compatible avec l'addition, de sorte que l'ensemble  $\bar{M}$  des  $\bar{\mathbf{m}}$  où  $\mathbf{m}$  parcourt  $M$  est stable par addition, donc est un code linéaire binaire de longueur  $n$ .

Inversement, si on se donne un code linéaire binaire  $C \subset \mathbf{F}_2^n$ , l'ensemble  $M$  des  $\mathbf{m} \in \frac{1}{2}\mathbf{Z}^n$  tels que  $\bar{\mathbf{m}} \in C$ , est un sous-groupe intermédiaire entre  $\mathbf{Z}^n$  et  $\frac{1}{2}\mathbf{Z}^n$ . C'est donc un réseau en vertu de la proposition précédente. On dira que  $M$  est le *réseau déduit du code  $C$* . Par exemple, les deux cas extrêmes sont  $M = \mathbf{Z}^n$  pour  $C = \{0\}$  et  $M = \frac{1}{2}\mathbf{Z}^n$  pour  $C = \mathbf{F}_2^n$ .

LEMME 13.26. — *Soient  $C \in \mathbf{F}_2^n$  un code linéaire,  $C^\perp$  le code orthogonal et  $M \subset \frac{1}{2}\mathbf{Z}^n$  le réseau déduit de  $C$ .*

- a) *Le réseau déduit du code  $C^\perp$  est  $M^\sharp$ .*  
 b) *Pour que  $M$  soit entier, il faut et il suffit qu'on ait  $C \subset C^\perp$ . Pour que  $M$  soit unimodulaire, il faut et il suffit qu'on ait  $C = C^\perp$ .*  
 c) *Pour que  $M$  soit pair, il faut et il suffit que tous les mots de  $C$  soient de poids multiples de 4.*

*Démonstration.* Soient  $\mathbf{m} = \frac{1}{2}(m_i)$  et  $\mathbf{n} = \frac{1}{2}(n_i)$  deux éléments de  $\frac{1}{2}\mathbf{Z}^n$ . On a

$$\langle\langle \mathbf{m} | \mathbf{n} \rangle\rangle = \frac{1}{2} \sum m_i n_i,$$

de sorte que dire que  $\langle\langle \mathbf{m} | \mathbf{n} \rangle\rangle$  est entier équivaut à ce que  $\sum m_i n_i$  soit pair, ou encore que  $\langle \overline{\mathbf{m}} | \overline{\mathbf{n}} \rangle$  soit nul. Dire qu'on a  $\langle\langle \mathbf{m} | \mathbf{n} \rangle\rangle \in \mathbf{Z}$  pour tout  $\mathbf{m} \in M$  équivaut donc à ce qu'on ait  $\langle \overline{\mathbf{m}} | \overline{\mathbf{n}} \rangle = 0$  pour tout  $\overline{\mathbf{m}} \in C$ . Cela implique a) et donc b). Prenant  $\mathbf{m} = \mathbf{n}$  dans la formule précédente, on voit que  $\langle\langle \mathbf{m} | \mathbf{m} \rangle\rangle = \frac{1}{2} \sum m_i^2$ . Mais on a  $m_i^2 \equiv 0 \pmod{4}$  si  $\overline{m}_i = 0$ , et  $m_i^2 \equiv 1 \pmod{4}$  si  $\overline{m}_i = 1$ . On en déduit aussitôt c).  $\square$

On déduit aussitôt du lemme :

**PROPOSITION 13.27.** — *Soit  $C \subset \mathbf{F}_2^n$  un code linéaire de longueur  $n$  dont les poids sont multiples de 4 et qui est son propre orthogonal. Soit  $M \subset \frac{1}{2}\mathbf{Z}^n$  le réseau déduit de  $C$ . Alors le réseau  $M$  est unimodulaire et pair.*

**COROLLAIRE 13.28.** — *Soit  $n$  un entier divisible par 8, tel que  $p = n - 1$  soit premier. Soit  $C \subset \mathbf{F}_2^n$  un code binaire étendu de résidus quadratiques et soit  $M \subset \frac{1}{2}\mathbf{Z}^n$  le réseau déduit de  $C$ . Alors le réseau  $M$  est unimodulaire et pair.*

*Démonstration.* Les hypothèses de la proposition sont en effet satisfaites en vertu de la proposition 13.3 et de son corollaire 13.4.  $\square$

Le premier cas est  $n = 8$ ,  $p = 7$ , avec pour  $C$  le code de Hamming étendu, le second cas est  $n = 24$ ,  $p = 23$ , avec pour  $C$  le code de Golay  $G_{24}$ .

### 13.3.3. Exemple : le réseau $E_8$

Traitons en détail le premier cas. Prenons donc  $n = 8$  et  $C = \overline{H}_3$ . Soit  $M \subset \frac{1}{2}\mathbf{Z}^8$  le réseau déduit de  $C$ . Notons  $e_i$  les vecteurs de base de  $\mathbf{Z}^8$ , de sorte que les  $\frac{1}{2}e_i$  forment une base de  $\frac{1}{2}\mathbf{Z}^8$ . Posons

$$\begin{aligned} f_0 &= \frac{1}{2}(e_0 + e_1 + e_3 + e_7), \\ f_1 &= \frac{1}{2}(e_1 + e_2 + e_4 + e_7), \\ f_2 &= \frac{1}{2}(e_2 + e_3 + e_5 + e_7), \\ f_3 &= \frac{1}{2}(e_3 + e_4 + e_6 + e_7). \end{aligned}$$

**PROPOSITION 13.29.** *Les vecteurs  $(f_0, \dots, f_3, e_4, \dots, e_7)$  forment une base du réseau M.*

*Démonstration.* Notons d'abord que les  $\bar{f}_i \in \mathbf{F}_2^8$  forment une base de C : c'est la base même par laquelle nous avons défini  $\bar{H}_3$  dans 6.3.2. Par ailleurs les  $f_i$  pour  $i = 0, 1, 2, 3$  et les  $\frac{1}{2}e_j$  pour  $j = 4, 5, 6, 7$  forment une base de  $\frac{1}{2}\mathbf{Z}^8$  : elle se déduit de la base des  $\frac{1}{2}e_i$  par une matrice triangulaire de diagonale unité. Les éléments  $\mathbf{m}$  de  $\frac{1}{2}\mathbf{Z}^8$  sont donc les combinaisons linéaires

$$a_0 f_0 + \cdots + a_3 f_3 + (a_4/2) e_4 + \cdots + (a_7/2) e_7,$$

où les  $a_i$  sont entiers. L'image  $\bar{\mathbf{m}}$  d'un tel élément dans  $\mathbf{F}_2^8$  est la somme d'un élément de C et de  $(0, 0, 0, 0, \varepsilon_4, \varepsilon_5, \varepsilon_6, \varepsilon_7)$  où  $\varepsilon_j$  est la classe de  $a_j$  modulo 2. Dire qu'il appartient à C, c'est dire que tous les  $\varepsilon_j$  sont nuls, donc que tous les  $a_j$  sont pairs.  $\square$

Ce réseau est appelé le réseau E<sub>8</sub>. Il a comme base les huit lignes de la matrice

$$\left[ \begin{array}{cccccccc} \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right],$$

déduite directement en fin de compte de la matrice génératrice  $\bar{G}$  de C donnée en 6.4.4. Notons d'ailleurs qu'on pourrait obtenir d'autres bases en remplaçant indépendamment un nombre quelconque des coefficients par leurs opposés, la démonstration restant identique.

### 13.3.4. Vecteurs de carré scalaire 2

Soit  $\mathbf{m} \in \frac{1}{2}\mathbf{Z}^n$ . Écrivons  $\mathbf{m} = \frac{1}{2} \sum_{i=0}^{n-1} a_i e_i$  où les  $a_i$  sont entiers. On a  $\langle\langle \mathbf{m}| \mathbf{m} \rangle\rangle = \frac{1}{2} \sum a_i^2$ . Il en résulte que les vecteurs  $\mathbf{m} \in \frac{1}{2}\mathbf{Z}^n$  avec  $\langle\langle \mathbf{m}| \mathbf{m} \rangle\rangle = 1$  sont les  $\frac{1}{2}(\pm e_i \pm e_j)$ , où  $i$  et  $j$  sont deux indices distincts. De même, les vecteurs  $\mathbf{m} \in \frac{1}{2}\mathbf{Z}^n$  avec  $\langle\langle \mathbf{m}| \mathbf{m} \rangle\rangle = 2$  sont, d'une part les  $\pm e_i$ , et d'autre part les  $\frac{1}{2}(\pm e_i \pm e_j \pm e_k \pm e_l)$ , où  $i, j, k, l$  sont quatre indices distincts. Par conséquent :

**PROPOSITION 13.30.** — *Soient C ⊂ F<sub>2</sub><sup>n</sup> un code linéaire de longueur n et M ⊂ 1/2Z<sup>n</sup> le réseau déduit de C.*

a) *Le nombre des vecteurs  $\mathbf{m} \in M$  tels que  $\langle\langle \mathbf{m}| \mathbf{m} \rangle\rangle = 1$  est  $4n_2$ , où  $n_2$  est le nombre de mots de poids 2 du code C.*

b) *Le nombre des vecteurs  $\mathbf{m} \in M$  tels que  $\langle\langle \mathbf{m}| \mathbf{m} \rangle\rangle = 2$  est  $2n + 16n_4$ , où  $n_4$  est le nombre de mots de poids 4 du code C.*

Par exemple, le réseau  $E_8$  déduit du code de Hamming étendu ne possède aucun vecteur de carré scalaire 1 et possède  $2 \times 8 + 16 \times 14 = 240$  vecteurs de carré scalaire 2.

**REMARQUE 13.31.** — On notera que b) implique que le nombre de vecteurs de carré scalaire 2 est au moins égal à  $2n$ . Par exemple, si on part du code de Golay  $G_{24}$ , on trouvera un réseau unimodulaire pair de dimension 24 ayant  $2n = 48$  vecteurs de carré scalaire 2. Une construction un peu plus compliquée, utilisant la réduction modulo 4 et non la réduction modulo 2, permet de construire un réseau unimodulaire pair de dimension 24 sans aucun vecteur de carré scalaire 2. C'est le *réseau de Leech*<sup>5</sup>; il est caractérisé par les propriétés précédentes.

### 13.3.5. Réseaux et matrices symétriques entières

Soit  $M \subset \mathbf{R}^n$  un réseau et soit  $(e_i)$  une base de ce réseau. Notons  $(e'_i)$  la base dual, qui est donc une base du réseau associé  $M^\sharp$ . Puisque  $(e_i)$  et  $(e'_i)$  sont deux bases de  $\mathbf{R}^n$ , on peut écrire

$$e_i = \sum_j a_{ij} e'_j,$$

où  $A = (a_{ij})$  est une matrice réelle inversible. Prenant le produit scalaire de la somme précédente avec  $e_j$ , on obtient par définition même de la base dual

$$a_{ij} = \langle\langle e_i | e_j \rangle\rangle.$$

La matrice A est donc symétrique.

**PROPOSITION 13.32.** — a) Pour que le réseau M soit entier, il faut et il suffit que la matrice A soit à coefficients entiers.

b) Pour que le réseau M soit pair, il faut et il suffit que les coefficients de la matrice soient entiers, les coefficients diagonaux étant pairs.

c) Pour que le réseau M soit unimodulaire, il faut et il suffit que la matrice A soit à coefficients entiers et de déterminant égal à 1.

*Démonstration.* Dire que le réseau M est entier, c'est dire que les  $e_i$  appartiennent à  $M^\sharp$ , c'est-à-dire que les  $a_{ij}$  sont entiers, ce qui prouve a). Si M est pair, il est entier, donc les  $a_{ij}$  sont entiers; de plus, les  $a_{ii} = \langle\langle e_i | e_j \rangle\rangle$  sont pairs. Inversement, si la matrice A est entière à diagonale paire, alors, pour tout  $\mathbf{m} = \sum n_i e_i \in M$ , on a

$$\langle\langle \mathbf{m} | \mathbf{m} \rangle\rangle = \sum_i a_{ii} n_i^2 + 2 \sum_{i < j} a_{ij} n_i n_j \in 2\mathbf{Z}.$$

On a ainsi prouvé b). Prouvons enfin c). On peut supposer M entier. Dire que M est unimodulaire, c'est dire que les  $e_i$  forment une base du réseau  $M^\sharp$ . Comme les  $e'_i$

---

5. Ce réseau a semble-t-il été découvert par Ernst WITT en 1940, mais ce dernier ne l'a pas publié. John LEECH l'a décrit en 1964.

forment aussi une base, cela signifie que la matrice A est inversible dans l'anneau des matrices à coefficients entiers, ou encore que son déterminant est inversible dans  $\mathbf{Z}$ . Mais les seuls éléments inversibles de  $\mathbf{Z}$  sont 1 et  $-1$ , et le déterminant de la matrice symétrique A est nécessairement  $> 0$ . Cela achève la démonstration.  $\square$

Ainsi, d'une base d'un réseau unimodulaire pair, on déduit une matrice symétrique entière à diagonale paire de déterminant 1. La construction directe de telles matrices est difficile ; on peut d'ailleurs démontrer que leur taille est nécessairement multiple de 8 (voir [Serre]).

Reprenons l'exemple de  $E_8$ . La matrice des produits scalaires de la base donnée est

$$\begin{bmatrix} 2 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 2 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 2 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 2 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 2 \end{bmatrix}.$$

### 13.3.6. Ceci n'est pas une conclusion

Comme on l'imaginera sans peine, le sujet est loin d'être épuisé par ce premier exemple. C'est même, comme aurait dit Kipling, le début d'une autre histoire, que l'on trouvera notamment dans [Conway].



## Pour aller plus loin sur les codes

La théorie des codes est très multiforme et se sépare assez nettement en branches. Comme nous l'avons vu en 8.4, elle a donné naissance dès sa création au sein des laboratoires Bell à la fin des années 40, à deux variantes, probabiliste (Shannon) et algébrique (Hamming et Golay), qui se sont développées ensuite de façon essentiellement indépendante.

Du côté algébrique se sont fait jour des liens avec beaucoup d'autres branches de l'algèbre, notamment la théorie des courbes algébriques, la théorie des groupes finis, la théorie des réseaux et des formes quadratiques entières.

Nous n'avons pas du tout parlé du premier thème (codes de Goppa) et très brièvement abordé les autres (13.3). Sur le modèle des liens entre code de Golay (1949), groupe de Mathieu (1873) et réseau de Leech (1964, Witt 1940, mais voir note p. 296), s'est créé tout un univers combinatoire jetant un nouvel éclairage sur la classification des groupes finis simples.

### Bibliographie

En ce qui concerne la théorie algébrique « classique », le traité de MAC-WILLIAMS et SLOANE (les références précises se trouvent en page 329) reste une référence essentielle, malgré son âge et le caractère ardu de sa lecture. On consultera aussi le *Handbook* [Handbook] et les traités de LIN et COSTELLO et de MORELOS-ZARAGOZA. Le site de ce dernier (<http://www.eccpage.com>) met en ligne des programmes de calcul sur les codes.

Pour la théorie probabiliste, on renvoie au traité de David MACKAY, disponible au téléchargement sur <http://www.inference.phy.cam.ac.uk/mackay/itila/book.html>. Citons aussi le livre écrit sous la direction de Claude BERROU, co-inventeur des turbo-codes.

Pour la relation entre codes correcteurs, réseaux et groupes sporadiques, on se reportera à [Conway].

On trouvera un catalogue de réseaux à l'URL :

<http://www.research.att.com/~njas/lattices/index.html>



# Glossaire d'algèbre

## Anneau

Un anneau<sup>1,2</sup> est muni de deux lois de composition : une loi de groupe commutatif, notée  $x + y$ , et une loi associative et commutative, notée  $xy$  ou  $x \cdot y$ , distributive par rapport à l'addition et possédant un élément unité, noté  $1_A$  ou  $1$ .

Les axiomes des anneaux sont donc les suivants :

- ▷  $(x + y) + z = x + (y + z)$
- ▷  $x + y = y + x$
- ▷  $x + 0 = x$
- ▷  $x + (-x) = 0$
- ▷  $(xy)z = x(yz)$
- ▷  $xy = yx$
- ▷  $1x = x$
- ▷  $x(y + z) = xy + xz$

On déduit de là toutes les règles de calcul habituelles. On a par exemple  $0x = 0$ , car  $x = 1x = (0 + 1)x = 0x + 1x = 0x + x$ , et par conséquent  $0x = x + (-x) = 0$ . On a de même  $(-1)x = -x$ .

Comme pour toutes les lois de composition commutatives, on utilise les notations habituelles telles que  $x - y$  pour  $x + (-y)$ ,  $2x$  ou  $2 \cdot x$  pour  $x + x$ ,  $3x$  pour  $x + x + x$ ,  $-2x$  pour  $(-x) + (-x)$ ,  $x^2$  pour  $xx$ , ... Dans un anneau A, on utilisera donc librement des expressions du type  $nx$  ou  $n \cdot x$  pour  $n \in \mathbf{Z}$  et  $x \in A$ ,  $x^n$  pour  $n \in \mathbf{N}$  et  $x \in A$  (on a  $x^0 = 1$ ). On prendra garde à la chose suivante : pour  $n \in \mathbf{Z}$  et  $x \in A$ , on a  $n \cdot x = (n \cdot 1_A)x$ , ce qui amène à confondre l'entier  $n \in \mathbf{Z}$  et l'élément  $n \cdot 1_A$  de A ; mais on peut avoir dans A l'égalité  $n \cdot 1_A = 0$  sans que l'entier  $n$  soit nul !

## Anneau intègre

On appelle anneau intègre<sup>1</sup> un anneau A qui n'est pas réduit à 0 et dont le seul diviseur de zéro est 0 (autrement dit dans lequel le produit de deux

1. On peut généraliser cette définition en n'obligeant pas la multiplication à être commutative et/ou en ne l'obligeant pas à avoir un élément unité. Les anneaux de ce livre sont commutatifs et unifères par convention.

2. *Ring* en anglais.

3. L'adjectif anglais « *integral* » signifie déjà « entier » et aussi « intégral ». Pour « anneau intègre », on ne dit pas « *integral ring* », mais « *integral domain* » ou plus simplement « *domain* ». Certains auteurs disent d'ailleurs en français « anneau d'intégrité » ou « domaine d'intégrité ».

éléments non nuls n'est jamais nul). Un corps est un anneau intègre. Un anneau de polynômes à coefficients dans un anneau intègre (et notamment à coefficients dans un corps) est intègre.

### Anneau nul

Anneau réduit à son élément nul. Si A est non nul, on a  $1_A \neq 0$ .

### Anneau-quotient

Anneau dont les éléments sont des classes de congruence.

### Caractéristique d'un corps

Nombre premier  $p$ , s'il existe, tel que  $p \cdot 1 = 0$ , sinon 0.

### Coefficients du binôme

Si  $n$  est un entier positif, les *coefficients du binôme* sont les entiers  $\binom{n}{i}$  donnés par

$$\binom{n}{i} = \begin{cases} 0 & \text{pour } i < 0 \text{ et } i > n \\ \frac{n(n-1)\cdots(n-i+1)}{i!} & \text{pour } 0 \leq i \leq n. \end{cases}$$

Ils satisfont à la relation de récurrence

$$\binom{n+1}{i} = \binom{n}{i} + \binom{n}{i-1}, \quad i \in \mathbf{Z}.$$

### Corps

On dit qu'un anneau A est un *corps*<sup>4</sup> si A n'est pas réduit à 0 et si tout élément non nul est inversible.

### Différence symétrique

La différence symétrique de deux ensembles A et B est  $A \cup B - A \cap B$ , complémentaire de leur intersection dans leur réunion. Cela correspond ou *ou exclusif* en logique.

### Diviseur de zéro dans un anneau

On dit qu'un élément  $x$  d'un anneau A est un *diviseur de zéro* s'il existe  $y \neq 0$  avec  $xy = 0$ .

### Élément inversible d'un anneau

Un élément  $x$  d'un anneau A est dit *inversible* s'il existe  $y$  avec  $xy = 1$ . L'élément  $y$  est uniquement déterminé : si  $xy = xy' = 1$ , alors  $y = 1y = (xy)y = (xy)y' = y'$ . On le note  $x^{-1}$  et on étend la notation  $x^n$  au cas où  $n \in \mathbf{Z}$ .

4. Le français « corps » (qui provient de l'usage du mot « *Körper* » dans les premiers articles allemands introduisant la théorie générale des corps) devient « *field* » en anglais. On trouve d'ailleurs dans certains textes français « champ » au lieu de corps, surtout dans l'expression « champ de Galois », pour désigner les corps finis.

**Élément nul d'un anneau**

Élément neutre de l'addition. On le note 0. On a  $0 + x = x$  pour tout  $x$ .

**Élément unité d'un anneau**

Élément neutre de la multiplication. On le note  $1_A$  ou 1. On a  $1_A \cdot x = x$  pour tout  $x \in A$ .

**Formule du binôme**

Si  $x$  et  $y$  sont deux éléments d'un anneau avec commutatif et  $n$  un entier  $\geq 0$ , on a

$$(x + y)^n = \sum_{i \in \mathbf{Z}} \binom{n}{i} x^i y^{n-i} = \sum_{i=0}^n \binom{n}{i} x^i y^{n-i}.$$

**Groupe multiplicatif d'un anneau**

Les éléments inversibles d'un anneau  $A$  forment un groupe pour la multiplication, d'élément unité  $1_A$ , noté  $A^*$ .

**Homomorphisme d'anneaux**

Application d'un anneau dans un autre qui respecte l'addition et la multiplication et envoie élément unité sur élément unité.

**Logarithme**

Par *logarithme* d'un nombre réel  $x > 0$ , on entendra toujours le logarithme à base 2, de sorte que  $x = 2^{\log x}$ .

**Plancher, plafond**

Pour tout nombre réel  $x$ , le *plancher*  $\lfloor x \rfloor$  et le *plafond*  $\lceil x \rceil$  sont les entiers caractérisés par les inégalités suivantes :

$$\lfloor x \rfloor \leq x < \lfloor x \rfloor + 1, \quad \lceil x \rceil - 1 < x \leq \lceil x \rceil.$$

Si  $x$  est entier, on a  $\lfloor x \rfloor = x = \lceil x \rceil$ . S'il ne l'est pas, on a  $\lfloor x \rfloor < x < \lceil x \rceil$  et  $\lceil x \rceil = \lfloor x \rfloor + 1$ .

**Sous-anneau**

Une partie d'un anneau  $A$  est appelée un sous-anneau si elle est stable par addition, passage à l'opposé et multiplication, *et*<sup>5</sup> si elle contient l'élément unité de  $A$ . C'est alors un anneau pour les opérations induites.

**Sous-corps**

Sous-anneau qui est un corps. Si  $k$  est un sous-corps de l'anneau  $A$ , alors  $A$  est naturellement un espace vectoriel sur  $k$ .

5. Cette condition ne résulte pas des autres : par exemple la partie réduite à 0 qui est en elle-même un anneau, n'est un sous-anneau que si  $A$  est lui-même réduit à 0.



# Solutions des exercices

1.1. — On prend  $a \bmod b$  comme invariant.

1.2. — On a toujours  $0 \leq r < 2^k b$  et  $2^k b q + r$  est invariant.

1.3. — Pour montrer que le nombre de pas est au moins  $n$  et pour déterminer la stratégie donnant  $n$  pas, on pourra considérer le nombre de  $a_i$  présents dans l'expression obtenue. Pour la borne supérieure, on pourra associer à toute expression  $u = \sum c_i x_i + \sum d_j a_j$  l'entier  $f(u) = \sum e_i 2^{i-1}$  où  $e_i = 1$  si  $c_i = 0$  et  $e_i = 0$  si  $c_i \neq 0$ . Dans ce dernier cas, par exemple, on note que  $f(u)$  diminue strictement à chaque réécriture, qu'il diminue exactement de 1 dans le seul cas où l'on prend la dernière règle applicable, qu'il part de  $2^n - 1$  et qu'il arrive à 0.

1.4. —  $n! f$ .

1.5. — Réécrire dans l'ensemble des triplets  $(m, f, f')$ , en partant de  $(1, 0, 1)$  pour aboutir à  $(n, F_{n-1}, F_n)$ .

1.6. — Réécrire dans l'espace des suites pour manipuler des expressions du type  $(n, f(n-k), \dots, f(n-1))$ .

1.7. — On trouve  $\lfloor \log m \rfloor + S(m) - 1$ , à savoir  $\lfloor \log m \rfloor$  élévations au carré et  $S(m) - 1$  autres multiplications.

1.8. — Calculer la puissance  $n$ -ième de la matrice.

1.9. — Le nombre de E est la somme  $S(m)$  des digits binaires de  $n$  (si l'on part de  $f(0)$ ), le nombre total de D et de E est  $\lfloor \log(m) \rfloor$ .

1.10. — On a  $F_{2n} = 2F_n F_{n+1} - F_n^2$ ,  $F_{2n+1} = F_{n+1}^2 + F_n^2$  et  $F_{2n+2} = F_{n+1}^2 + 2F_{n+1} F_n$ .

1.11. — Vérifier que  $\gamma(2n) < \gamma(2n+1) < \gamma(2n+2)$ .

1.12. — On applique l'algorithme d'Euclide soustractif à  $u$  et  $v$ , et on remarque que  $\text{pgcd}(f(u), f(v))$  reste invariant. On a par ailleurs  $a^m - b^m = a^{m-n}(a^n - b^n) + a^n(a^{m-n} - b^{m-n})$ .

1.14. — La validité est claire :  $\text{pgcd}(u, v)$  est invariant ainsi que l'imparité de  $v$ . Puisque le produit  $uv$  est au moins divisé par 2 à chaque pas, le nombre des pas de réécriture est au plus  $\log(u) + \log(v)$ .

1.16. — Récurrence sur  $n$  par exemple.

1.17. — Multiplier les matrices correspondant à  $n$  et  $m$ .

1.18. — Pour la première assertion, on utilise la relation donnant  $F_{n+m}$  en notant que  $F_m$  et  $F_{m-1}$  sont premiers entre eux (identité de Bézout). On suit alors l'algorithme d'Euclide soustractif.

1.19. — On a  $F_{-n} = F_n$  pour  $n$  impair et  $F_{-n} = -F_n$  pour  $n$  pair.

1.20. — La relation est vraie pour  $n = 0$  et pour  $n = 1$  et le second membre satisfait à la même relation de récurrence que les  $F_n$ .

1.21. — Il s'agit de prouver que l'on a  $F_{n+1} \geq 2^{\frac{2}{3}(n-1)}$ . C'est vrai pour  $n = -1$  et  $n = 0$ . On raisonne par récurrence en notant que  $1 + 2^{2/3} \geq 2^{4/3}$ .

1.22. — On a  $\sum \log(x_i) \leq c_1(\log v)^2$ .

1.23. — Les réécritures respectent les relations  $cx + dy = u$ ,  $ax + by = v$ , et conservent le pgcd de  $u$  et  $v$ . On peut remarquer d'ailleurs que  $ad - bc$  reste invariant au signe près, donc égal à  $\pm 1$ .

1.24. — Si  $m = \alpha x = \beta y$ , on a  $md = (\alpha\beta + \alpha b)xy$ .

1.26. —  $R = U(a)$ ; en dérivant, on obtient  $Q(a) = U'(a)$ .

1.27. — Modifier la démonstration en posant  $U' = bU - aX^{n-m}V$ .

1.28. — Le degré de  $R$  décroît;  $VQ + R$  est invariant.

1.29. — Référence sur  $i$ .

1.30. — Référence sur  $i$ .

1.32. — Tout au long de l'exécution, on a  $x = x_i$  et  $y = x_{2i}$ .

1.34. — L'entier  $r$  repart à 1 à chaque saut de  $i$ .

2.1. — La nécessité est claire. Inversement, on pose  $x = x_1 + z$ , et tout se divise par  $d$ .

2.2. — On note d'abord que le pgcd de  $x + 1$  et  $x - 1$  est 1 ou 2. Supposons d'abord  $n$  impair. Si  $n = \prod p_i^{n_i}$ , avec  $N$  nombres premiers distincts, chacun des  $N$  facteurs doit donc diviser  $x + 1$  ou  $x - 1$ , et il s'agit de résoudre l'un des  $2^N$  systèmes « chinois »  $x \equiv \pm 1 \pmod{p_i^{n_i}}$ . Il y a donc  $2^N$  solutions. Si maintenant  $n = 2^a n'$ , avec  $n'$  comme ci-dessus, il faut ajouter la condition relative à 2. Pour  $a = 1$ , c'est  $x \equiv 1 \pmod{2}$  et on a  $2^N$  solutions. Pour  $a = 2$ , c'est  $x \equiv \pm 1 \pmod{4}$ , et on a  $2^{N+1}$  solutions. Pour  $a > 2$ , on a quatre possibilités, à savoir  $x \equiv \pm 1 \pmod{2^a}$  et  $x \equiv \pm 1 + 2^{a-1} \pmod{2^a}$  et on a  $2^{N+2}$  solutions. Les seuls cas pour lesquels il y a exactement 2 solutions sont  $n = 4$ ,  $n = p^r$  et  $n = 2p^r$ , avec  $p \neq 2$  premier et  $r \geq 1$ .

2.3. — Dès que  $n$  est  $> 2$ , alors 1 et  $n - 1$  sont inversibles modulo  $n$  et non congrus modulo  $n$ , de sorte que  $\varphi(n) > 1$ .

2.4. — Si  $a$  est premier à  $n$ ,  $n - a$  l'est aussi, et il sont distincts, puisque sinon  $n = 2a$ . Dès que  $n \geq 5$ ,  $n$  est divisible soit par un nombre premier  $\geq 5$ , soit par 6, 8 ou 9, ce qui implique que  $\varphi(n)$  est  $> 2$ .

2.5. — Si  $n$  est impair, on a  $\varphi(2n) = \varphi(n)$ . Si  $n$  est pair, on a  $\varphi(2n) = 2\varphi(n)$ .

2.6. — Si  $p^r$  divise  $n$ , alors  $(p - 1)p^{r-1}$  divise  $a$ . Cela donne un nombre fini de  $p$  et de  $r$  possibles.

2.7. — Soit  $p_i$  le  $i$ -ième nombre premier. On considère la suite des  $n_j = p_1 \cdots p_j$ . Alors  $n_j/\varphi(n_j)$  est le produit  $(1 - 1/p_1)^{-1} \cdots (1 - 1/p_j)^{-1}$ . En développant en série et en multipliant, on voit que cette expression est la somme des  $1/m$  où  $m$  parcourt les entiers qui n'ont que  $p_1, \dots, p_j$  comme facteurs premiers. Comme la série de terme général  $1/m$  diverge, il en résulte que  $n_j/\varphi(n_j)$  augmente indéfiniment lorsque  $j$  tend vers l'infini.

2.8. — Si  $n$  est premier,  $\varphi(n)/n = 1 - 1/n$ .

2.9. — Si  $m$  et  $n$  sont premiers entre eux, les diviseurs de  $mn$  sont exactement les produits d'un diviseur de  $m$  et d'un diviseur de  $n$ .

2.10. — Les deux membres sont multiplicatifs. Il suffit donc de traiter le cas où  $n = p^r$ .

2.11. — L'ordre de  $g$  doit diviser  $m$ , mais ne peut diviser aucun diviseur propre de  $m$ .

2.12. — Si  $rk$  est multiple de  $m$ , alors  $r$  est multiple de  $m$ . Il y a par ailleurs  $\varphi(m)$  entiers premiers à  $m$  et compris entre 1 et  $m$ .

2.13. — Pour que  $nr$  soit un multiple de  $m$ , il faut il suffit que  $r$  soit un multiple de  $m/d$ .

2.14. — L'application  $h \mapsto gh$  est bijective, puisque  $g$  est inversible. Si  $g_1h_1 = g_2h_2$ , alors  $g_2 = g_1h_1h_2^{-1}$  et  $g_2h = g_1(h_1h_2^{-1}h)$ .

2.15. — Calculer le produit des  $ax$ , où  $x$  parcourt les éléments non nuls de  $K$ .

2.16. — Prendre le coefficient de  $X$ , en notant que, ou bien  $p$  est impair, ou bien  $p = 2$ . Inversement, cette relation implique que les entiers  $< p$  sont inversibles modulo  $p$ , donc que  $p$  est premier.

2.17. — Le produit de deux éléments inverses l'une de l'autre vaut 1. Dans le produit de tous les éléments non nuls ne restent ainsi que les éléments égaux à leur inverse. Mais l'équation  $x^{-1} = x$  a comme solutions  $x = 1$  et  $x = -1$ , ce dernier seulement si 2 n'est pas nul dans  $K$ . Dans ce deuxième cas, on a  $-1 = 1$ .

2.18. — Si  $n$  n'est pas le carré d'un nombre premier, il est le produit de deux facteurs distincts et  $< n$ . Si  $n = p^2$ , alors  $p$  et  $2p$  sont  $< n$ .

2.19. — On peut utiliser le théorème de Wilson lorsque  $n + 1$  est premier. Ne pas s'étonner si on a des difficultés pour  $n = 20$ .

2.20. — On a  $\binom{p}{i} = \frac{p(p-1)\cdots}{(i-1)\cdots}$ .

2.21. — On prend  $a = 1 \cdot 2 \cdots ((p-1)/2)$ .

2.22. — On trouve  $3 \times 37$ . On a bien  $37 \equiv 1 \pmod{6}$ , mais non pas  $3 \equiv 1 \pmod{6}$ . Cela est dû à ce que 3 est un facteur de  $10 - 1$ . Il ne faut donc faire attention au fait que des facteurs premiers des  $a^m \pm b^m$  peuvent intervenir dans  $a^n \pm b^n$  avec des multiplicités plus importantes que ce que l'on pense.

2.23. — Dans le groupe  $(\mathbf{Z}/(a^n-1)\mathbf{Z})^*$ , l'élément  $a$  est évidemment d'ordre  $n$ .

2.24. — L'ordre de  $a$  modulo  $n$  divise  $m$  par hypothèse. Puisque  $m$  est premier, cet ordre vaut donc 1 (auquel cas  $a$  est congru à 1 modulo  $n$ ) ou  $m$  (auquel cas  $m$  doit diviser l'ordre du groupe). Dans le cas du nombre de Mersenne, qui est impair,  $n$  doit en outre être impair.

2.25. — L'algorithme d'Euclide, qui a un coût en  $\log(n)^2$  devrait être plus rapide que l'autre méthode, qui a un coût en  $\log(n)^3$ .

2.26. —  $a = 2$  et  $n = 4$ .

2.27. — Si  $p = 3a + 2$  et  $q = 3b + 2$ , on a  $\varphi(pq) = 3(3ab + a + b) + 1$  et on prend  $s = 2(3ab + a + b) + 1$ .

2.28. — On  $2^q \equiv 2 \pmod{q}$ , donc  $2^n \equiv 2^p \pmod{q}$ . La condition  $2^{n-1} \equiv 1 \pmod{q}$  implique donc  $2^{p-1} \equiv 1 \pmod{q}$ . Comme on a aussi  $2^{p-1} \equiv 1 \pmod{p}$ , on a  $2^{p-1} \equiv 1 \pmod{n}$ . L'ordre de 2 modulo  $n$  doit ainsi diviser  $p-1$  et aussi  $q-1$ , donc aussi  $d$ , ce qui donne  $2^d \equiv 1 \pmod{n}$ .

2.29. — Faire 32 élévations au carré successives.

2.31. — Les  $a$  premiers à  $n$  tels que  $a^{n-1} \equiv 1 \pmod{n}$  forment un sous-groupe, donc soit forment le groupe entier, soit sont en nombre au plus  $\varphi(n)/2$ .

2.32. — On sait que  $n$  est sans facteur multiple, et il suffit de prouver qu'il ne peut pas s'écrire  $mm'$  avec  $m$  et  $m'$  premiers entre eux et  $> 1$ . D'après le théorème chinois, on peut trouver  $b$  avec  $b \equiv a \pmod{m}$  et  $b \equiv 1 \pmod{m'}$ . Alors  $b^{(n-1)/2}$  est congru à  $-1$  modulo  $m$  et à  $1$  modulo  $m'$ . Il ne peut donc pas être congru à  $1$ , ni à  $-1$  modulo  $n$ .

3.1. — Il y a un élément d'ordre 1 : l'identité ; trois éléments d'ordre 2 : les transpositions ; et deux éléments d'ordre 3 : les permutations circulaires.

3.2. — Notons  $d$  le pgcd de  $m$  et  $n$ . Il s'agit de construire un diviseur  $n'$  de  $n$  qui soit premier à  $m/d$ . Pour ce faire, on part de  $n' = n$  et on réécrit  $n'$  en  $n'/\text{pgcd}(n', m/d)$ , tant que ce pgcd est  $\neq 1$ .

3.3. — Pour  $(g, g') \in G \times G'$ , on a  $(g, g')^r = (g^r, g'^r)$ , de sorte que l'ordre de  $(g, g')$  est le ppcm des ordres de  $g$  et  $g'$ . Il divise donc le ppcm  $m$  de  $\omega(G)$  et  $\omega(G')$ . Inversement, en prenant  $g$  et  $g'$  d'ordre maximal, on voit que  $G \times G'$  contient un élément d'ordre  $m$ . Cela implique  $\omega(G \times G') = m$ . Pour que  $G \times G'$  soit cyclique, il faut et il suffit donc que  $\#G \times \#G' = \text{ppcm}(\omega(G), \omega(G'))$ ; mais cela signifie que  $\#G = \omega(G)$  et de même pour  $G'$ , et que ces deux entiers sont premiers entre eux.

3.4. — C'est l'énoncé sur les ordres des éléments dans un groupe commutatif.

3.5. — Il faut vérifier que  $\zeta^{2^n} = 1$  et  $\zeta^{2^{n-1}} \neq 1$ . Mais, l'anneau  $A$  étant intègre, les relations  $\alpha^2 = 1$  et  $\alpha \neq 1$  n'ont qu'une solution commune, à savoir  $-1_A$ , et ceci à condition que  $-1_A \neq 1_A$ .

3.6. — Les racines du polynôme  $X' - 1$  sont simples, et on calcule sa dérivée.

3.7. — Ce groupe est donc cyclique à  $r$  éléments, et formé des racines  $r$ -ièmes de l'unité. Donc  $r$  doit diviser  $n$ . Les trois conditions de b) équivalent alors au fait que  $r = n$ . Enfin c) traduit simplement le fait que le groupe est cyclique d'ordre  $n$ .

3.8. — Plus  $r$  est grand et plus la conclusion est forte. L'énoncé où l'on prend  $r$  maximal, donc  $m$  premier à  $q$ , implique les autres.

3.9. — On a  $n - 1 = 3 \times 2^{12}$  et on trouve  $3^{2^8} \equiv -1$ .

3.11. — Soit  $p$  un diviseur premier commun à  $\text{Fer}_n$  et  $\text{Fer}_m$ . Alors  $2^{2^m}$  et  $2^{2^n}$  sont tous deux congrus à  $-1$  modulo  $p$ , ce qui est impossible puisque le plus grand des deux est une puissance paire de l'autre.

3.12. — En effet  $2^{2^n}$  est congru à  $-1$  modulo  $p$ , voir l'exercice 3.5. Cet exercice est aussi un cas particulier de la proposition 2.16.

3.13. — 257 est un nombre de Fermat.

3.14. —  $1 - a^4b^4$  est divisible par  $1 + ab$ .

3.15. — S'il existe  $a$  avec  $a^{2^{r-1}} \equiv -1 \pmod{n}$ , alors  $n$  est premier (« critère de Proth »).

3.16. — On a  $2^{2^n} \equiv -1 \pmod{\text{Fer}_n}$ , et  $(\text{Fer}_n - 1)/2$  est une puissance paire de  $2^n$ .

3.17. — On a  $3^{32768} \equiv -1 \pmod{65537}$ .

3.19. —  $a' = a + 2c, b' = b - ac - c^2, a'^2 - 4b' = a^2 - 4b$ .

3.20. — Si  $a = x + x^{-1}$ , alors  $V_m(a) = y + y^{-1}$  avec  $y = x^m$ .

3.21. — On envoie  $B$  dans l'anneau proposé en posant  $a = x + y$  et  $X = x$ . L'application réciproque envoie  $x$  sur  $X$  et  $y$  sur  $a - X$ . En partant de

l'identité  $(x^n + y^n)(x + y) = x^{n+1} + y^{n+1} + xy(x^{n-1} + y^{n-1})$ , on obtient une relation de récurrence pour les  $W_n$ . Ceux-ci sont d'ailleurs symétriques et peuvent donc se mettre sous la forme  $Z_n(x + y, xy)$ . On obtient de même une relation de récurrence pour les  $Z_n$  à savoir

$$Z_{n+1}(s, p) = sZ_n(s, p) - pZ_{n-1}(s, p) - V_{n-1}(s).$$

3.22. — On pose  $n + 1 = uv$ , et on suppose qu'on a  $V_{n+1} \equiv 2 \pmod{n}$  et  $\text{pgcd}(V_{(n+1)/q} - 2, n) = 1$  pour tout facteur premier  $q$  de  $u$ . Alors tout facteur premier de  $n$  est congru à  $\pm 1$  modulo  $u$ . Si de plus  $v < u$ , alors  $n$  est premier.

3.23. — On trouve les  $2^s - 1$  pour les valeurs suivantes de  $s : 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, \dots$

3.24. — Il vaut 561, et 560 est divisible par 2, par 10 et par 16.

3.25. — Calquer la démonstration de la proposition précédente.

3.26. — En effet,  $(n - 1)/2$  vaut  $18m(36m^2 + 11m + 1)$ .

3.27. — Il y a 27 valeurs de  $m$  entre 1 et 1000, la 28-ième est 1025. Pour  $m = 6$  par exemple, on obtient le joli nombre  $331 \times 661 \times 991 = 216821881$ .

3.28. — On a  $qr - 1 = \alpha(p - 1)$  et  $pr - 1 = \beta(q - 1)$  où  $\alpha$  et  $\beta$  sont des entiers  $> 1$ . Donc  $\alpha\beta$  s'écrit  $\frac{(pr-1)(qr-1)}{(p-1)(q-1)}$ . On en tire par exemple  $r^2 < \alpha\beta \leqslant 2r^2 - r$  qui n'a qu'un nombre fini de solutions. Prenant alors  $r = 3$ , on trouve  $3 \times 11 \times 17 = 561$ . Prenant  $r = 5$ , on trouve  $5 \times 13 \times 17 = 1105$ ,  $5 \times 17 \times 29 = 2465$  et  $5 \times 29 \times 73 = 10585$ .

3.29. — Prendre  $a = -1$ .

3.30. —  $a = 2$  et  $n = 4$ .

3.31. — Dans a), pour que  $\binom{p}{2}$  soit divisible par  $p$ .

3.32. — Ces éléments sont les classes de 3, 5 et 7. Les carrés de ces trois nombres sont congrus à 1 modulo 8. On a enfin  $3 \times 5 \equiv 7 \pmod{8}$ .

3.33. — De  $(1 + 4y)^{2^{n-3}} = 1 + 2^{n-1}y + 2^nz$ , on déduit par élévation au carré  $(1 + 4y)^{2^{n-2}} = 1 + 2^ny + 2^{n+1}z'$ .

3.34. — C'est en effet une puissance de 5, qui doit être paire, puisque  $5^2$  est congru à 1 modulo 8, et que 5 ne l'est pas.

3.35. — Soit  $N$  le nombre de facteurs premiers impairs de  $n$  et soit  $2^r$  la puissance maximale de 2 qui divise  $n$ . Alors le nombre de groupes est  $N$  si  $r = 0$  ou  $r = 1$ ,  $N + 1$  si  $r = 2$ , et  $N + 2$  si  $r > 2$ .

3.36. — Le nombre d'éléments d'ordre 2 est  $2^m$ , où  $m$  est le nombre de facteurs cycliques trouvés ci-dessus. Puisqu'un produit de groupes cycliques sont les ordres ne sont pas premiers entre eux n'est pas cyclique, le groupe  $(\mathbf{Z}/n\mathbf{Z})^*$  ne peut être cyclique que lorsque  $m = 1$ .

3.37. — Cela résulte du calcul de  $\lambda(n)$ , ou directement des exercices précédents.

3.38. — Soit  $a$  un entier. Pour que  $a$  modulo  $p^r$  engende  $(\mathbf{Z}/p^r\mathbf{Z})^*$ , il faut et il suffit, d'une part que sa classe modulo  $p$  soit une racine primitive modulo  $p$  et d'autre part qu'on ait  $a^{p-1} \not\equiv 1 \pmod{p^2}$ . Cela se démontre à l'aide du lemme 3.39 en raisonnant comme dans la démonstration de la proposition 3.40. Enfin, supposant  $a$  impair, dire que sa classe modulo  $2p^r$  engendre  $(\mathbf{Z}/2p^r\mathbf{Z})^*$  signifie que sa classe modulo  $p^r$  engendre  $(\mathbf{Z}/p^r\mathbf{Z})^*$  et les critères sont les mêmes.

3.39. — On reprend la démonstration et on trouve deux familles : les nombres  $n = p_1 p_2$ , avec  $p_2 - 1 = 2(p_1 - 1)$  et  $p_1 \equiv 3 \pmod{4}$ ; et les nombres de Carmichael  $n = p_1 p_2 p_3$ , avec, pour tout  $i$ ,  $p_i \equiv 3 \pmod{4}$  (et  $p_i - 1$  divisant  $n - 1$ ). Le plus petit nombre de la première famille est  $3 \times 5 = 15$ , le plus petit nombre de la seconde est  $271 \times 811 \times 2971 = 652969351$ .

4.1. — Si  $\omega$  est une racine principale seconde de l'unité, on a  $(\omega - 1)(\omega + 1) = 0$  et  $\omega - 1$  n'est pas diviseur de 0 dans  $a$ . Donc  $\omega = -1_A$  et  $2 \cdot 1_A$  n'est pas diviseur de zéro. Inversement, si  $2 \cdot 1_A$  n'est pas diviseur de 0,  $\omega = -1_A$  est une racine principale seconde de l'unité.

4.2. — On a  $(\omega^{n'})^i = \omega^{n'i}$ .

4.3. — Calcul explicite.

4.4. — Les deux premières relations sont claires. Par conséquent  $\mathcal{F} \circ \mathcal{F} \circ \sigma = \mathcal{F} \circ \overline{\mathcal{F}} = n \text{Id}$ .

4.5. — On peut raisonner directement comme suit. Si  $\mathcal{F}a = 0$ , le polynôme  $P_a$  possède  $n$  racines dont les différences ne sont pas diviseur de zéro. Comme il est de degré  $< n$ , il est nul (reprendre la démonstration de la proposition 3.4). Ainsi  $\mathcal{F}$  (et aussi  $\overline{\mathcal{F}}$ ) sont injectives. On peut aussi noter que la multiplication par  $n$ , qui est leur composée, l'est.

4.6. — On a  $\mathcal{F}(\overline{\mathcal{F}}(a) * \overline{\mathcal{F}}(b)) = \mathcal{F}\overline{\mathcal{F}}(a) \cdot \mathcal{F}\overline{\mathcal{F}}(b) = n^2 a \cdot b$ . Appliquant  $\overline{\mathcal{F}}$  aux deux membres, on obtient l'une des relations cherchée, et on fait de même pour l'autre (ou bien on applique  $\sigma$ ).

4.7. — Calculs directs.

4.8. — Il y a des multiplications par 1 ou  $-1$ .

5.1. — Modulo 17, on a  $(\pm 1)^2 \equiv 1$ ,  $(\pm 2)^2 \equiv 4$ ,  $(\pm 3)^2 \equiv -8$ ,  $(\pm 4)^2 \equiv -1$ ,  $(\pm 5)^2 \equiv 8$ ,  $(\pm 6)^2 \equiv 2$ ,  $(\pm 7)^2 \equiv -2$ ,  $(\pm 8)^2 \equiv -4$ .

5.2. — La première assertion résulte du fait que  $K^*$  est cyclique. On peut alors décalquer la démonstration de la proposition.

5.3. — Modulo 4, on a  $0^2 \equiv 2^2 \equiv 0$  et  $1^2 \equiv 3^2 \equiv 1$ .

5.4. — Évident.

5.5. — Utilisant l'exercice précédent, il suffit de traiter le cas où  $m$  et  $m'$  sont premiers entre eux. On applique alors le théorème chinois.

5.6. — 1 est un carré et  $-1$  ne l'est pas.

5.7. — Si  $n$  était le carré de  $a$ , alors  $a$  serait une racine primitive  $2(p-1)$ -ième.

5.8. — Calculer  $N(x)^2 - n$  modulo  $p^{r+1}$ , sachant que  $x^2 - n$  est divisible par  $p^r$ .

5.9. — Calculer  $N(y) + 2N(y)^2 - b$  modulo  $4^{r+1}$ , sachant que  $y + 2y^2 - b$  est divisible par  $4^r$ .

5.10. — Ce groupe est nul pour  $r \leq 1$ , cyclique d'ordre pair pour  $r = 2$ , produit d'un groupe cyclique à deux éléments et d'un groupe cyclique d'ordre pair pour  $r \geq 3$ . Il y a donc 1, 2 ou 4 homomorphismes :  $n \mapsto 1$  dans tous les cas,  $\varepsilon$  si  $r \geq 2$ ,  $\omega$  et  $\varepsilon\omega$  si  $r \geq 3$ .

5.11. — La multiplicativité résulte de celle de  $\varepsilon$ .

5.12. — En effet,  $p$  est congru à 7 modulo 12 : il est congru à  $-1$  modulo 4 et à 1 modulo 3, car  $n$  est impair.

5.13. — Supposons  $p > 3$ . Les racines de  $X^3 - 1$  sont 1 et les racines du trinôme  $X^2 + X + 1$ , de déterminant  $-3$ . Il y a donc trois ou une racine selon que  $-3$  est un carré ou non. Il y a donc trois racines pour  $p \equiv 1 \pmod{3}$  et une racine pour  $p \equiv -1 \pmod{3}$ . L'application  $x \mapsto x^3$  est un homomorphisme de groupes, dont on connaît le noyau, ce qui permet de calculer le nombre d'éléments de l'image. Tout élément est donc un cube si  $p \equiv -1 \pmod{3}$ , et il y a  $(p-1)/3$  résidus cubiques non nuls si  $p \equiv 1 \pmod{3}$ .

5.14. — Comme on l'a déjà vu, l'ordre de 2 modulo  $p$  est  $2^{n+1}$ . On a  $(\frac{2}{p}) = 1$  d'après la proposition précédente. Comparant à la congruence d'Euler  $2^{(p-1)/2} \equiv 1 \pmod{p}$ , on voit que  $(p-1)/2$  est un multiple de  $2^{n+1}$ .

5.15. — La somme précédente est le double de la somme étendue aux résidus, mais la somme totale (résidus et non-résidus) est nulle.

5.16. — Puisque  $p$  est congru à 1 modulo 4, on a  $(\frac{q}{p}) = (\frac{p}{q})$ . Il s'agit donc de prouver que  $(\frac{p}{q}) = -1$  pour  $q = 3, q = 5$  et  $q = 7$ . Pour  $q = 3$ ,

c'est clair puisque  $4 \equiv 1 \pmod{3}$  donc  $p \equiv 1 + 1 \pmod{3}$ . Pour  $q = 5$ , on a de même  $16 \equiv 1 \pmod{5}$ , donc  $p \equiv 1 + 1 \pmod{5}$ . Pour  $q = 7$ , on a enfin  $256 \equiv 4 \pmod{7}$ , mais les puissances de 4 modulo 7 sont 2 et 4, ce qui implique, soit  $p \equiv 2 + 1 \pmod{7}$ , soit  $p \equiv 4 + 1 \pmod{7}$ .

5.17. — La première assertion est claire. La seconde en résulte via la loi de réciprocité.

5.18. — Cela résulte de la proposition 5.12 et du théorème chinois.

5.19. —  $v$  reste impair et  $(\frac{u}{v})_v$  est invariant.

5.20. — La multiplication  $b \times c$  se fait en temps polynomial.

5.21. — Vérifier que l'égalité d'un **n** donné et de l'un des éléments de la partie se fait en temps polynomial.

5.22. — Pour chaque opération, on fabrique un algorithme à partir des algorithmes donnés.

5.23. — Si  $E(\mathbf{n})$  est le temps de l'essai d'un témoin pour **n**, temps qui est polynomial en  $\|\mathbf{n}\|$ , le temps moyen sera  $E(\mathbf{n}) + (1-\alpha)(E(\mathbf{n}) + (1-\alpha)(\dots))$ , qui vaut  $E(\mathbf{n}) \sum_{i \geq 0} (1-\alpha)^i = \alpha^{-1} E(\mathbf{n})$ .

6.1. — On regarde les contributions bit à bit, et il y a quatre cas possibles.

6.2. — Il suffit de regarder des mots réduits à un bit.

6.3. — Avec les notations précédents, si  $\mathbf{m}_1 + \mathbf{e}_1$  et  $\mathbf{m}_2 + \mathbf{e}_2$  diffèrent au plus en  $e$  places, on a  $w(\mathbf{m}_1 - \mathbf{m}_2) \leq 2t + e$ .

6.4. — Les boules réduites à un point forment une partition.

6.5. — On a  $n = 2t + 1$ , et pour tout élément **m**, on a soit  $w(\mathbf{m}) \leq t$ , soit  $w(\mathbf{m}) > t$  et alors  $w(\mathbf{1} - \mathbf{m}) \leq t$ .

6.6. — Prendre un mot de code **m** et un mot **n** à distance  $t + 1$ . Alors **n** est à distance  $\leq t$  d'un mot de code **m'**. Alors **m** et **m'** sont distincts et à distance  $\leq 2t + 1$ .

6.7. — On a  $n = 3$  et le code est formé de deux éléments à distance 3, c'est-à-dire dont tous les bits diffèrent. Dans le cas linéaire, ce sont nécessairement  $(0, 0, 0)$  et  $(1, 1, 1)$ .

6.8. — On a  $N(1 + n + \dots + \binom{n}{t}) = 2^n$ .

6.9. — Les quatre premiers bits parcourent tous les mots de 4 bits.

6.10. — Le code de parité ; le code de répétition pure.

6.11. —  $C$  est orthogonal à  $C^\perp$  et a la bonne dimension.

6.12. — Calculs directs.

6.13. — Calculs directs.

6.14. — Vérification explicite.

6.15. — On vient de le dire pour  $P$  et il suffit d'appliquer à nouveau ce résultat à  $C^\perp$ .

6.16. — La première assertion est claire. La seconde s'en déduit en l'appliquant au code orthogonal ; on peut aussi remarquer que les lignes de  $P$  forment une base de  $C^\perp$  et qu'il s'agit de vérifier qu'elles sont dans le noyau de  $P$ .

6.17. — Le code contient les mots  $\gamma[1110] = [1000110]$ ,  $\gamma[0111] = [0100011]$ ,  $\gamma[0011] = [0010111]$  et  $\gamma[0001] = [0001101]$ . Ils forment une matrice génératrice dont la partie gauche est une matrice unité.

6.18. —  $P$  est de rang  $n - k$  et tout (donc au moins un) ensemble de  $d - 1$  colonnes est linéairement indépendant.

6.19. — C'est le code de répétition pure de longueur 3.

6.20. — Cela résulte par exemple de la proposition 6.16.

6.21. — Calcul explicite.

6.22. — Les trois lignes de  $P$  sont  $\gamma[1100]$ ,  $\gamma[0110]$  et  $\gamma[0011]$ , qui forment une base du sous-code pair.

6.23. — Le code de répétition pure de longueur 4.

6.24. — Calculant de façon analogue le noyau de  $P_{r+1}$ , on obtient que  $H_{r+1}$  est composé des triplets  $(\mathbf{a}, \varepsilon(\mathbf{a}), \mathbf{a} + \mathbf{a}')$ , où  $\mathbf{a}$  décrit  $\mathbb{F}_2^{2^r-1}$  et  $\mathbf{a}'$  décrit  $H_r$ .

7.1. — Prendre un code impair, puis son extension paire, puis une nouvelle extension paire, ce qui revient à ajouter un bit constamment nul.

7.2. — Cette permutation est une transformation affine inversible, qui envoie  $(x, y, z)$  sur  $(y + z + 1, y + 1, x + y + z + 1)$ .

7.3. — Le code plein de longueur 1, le code de parité de longueur 3, le sous-code pair du code de Hamming  $H_3$  de longueur 7.

7.4. — Le code plein de longueur 2, le code de parité de longueur 4, le code de Hamming étendu de longueur 8.

7.5. — Le groupe de toutes les permutations.

7.6. — Le même que celui du code initial.

7.7. — Les automorphismes du code sont les automorphismes de l'extension paire qui laissent fixe le bit de parité.

7.8. — Parce que, si  $C$  est pair, le dernier bit est toujours à 0, donc aussi tous les autres par l'hypothèse de transitivité.

7.9. — Même méthode : envoyer le bit choisi sur le bit de parité.

7.10. — Évident.

7.11. — Le décalage laisse le produit scalaire invariant.

7.12. — Pour se donner une transformation linéaire de  $\mathbf{F}_2^3$ , il faut se donner les images des trois vecteurs de base ; cela donne  $(8 - 1)(8 - 2)(8 - 4) = 168$  transformations. Pour obtenir le nombre des transformations affines, on doit encore multiplier par le nombre des translations. On trouve donc  $8(8 - 1)(8 - 2)(8 - 4) = 1344$ .

7.13. — On a affaire à des codes triviaux, stables par toutes les permutations.

8.1. — Le code obtenu par raccourcissement au complémentaire de  $J$ .

8.2. — Le code initial ; sa partie paire.

8.3. — On a  $N(k + 1, d) > N(k, d)$  et de même pour  $d + 1$ .

8.4. — On a  $N(k, d) \geq N(k', d')$ .

8.5. — Il suffit de traiter les cas  $(a, b) = (1, 0)$  et  $(a, b) = (0, 1)$ .

8.6. — En dehors du premier terme  $d$ , la somme contient  $k - 1$  termes, au moins égaux à 1.

8.7. — On doit avoir, soit  $k \leq 1$ , soit  $\lfloor d/2 \rfloor \leq 1$ , ce qui donne, en dehors du code nul, les paramètres  $(n, 1, n)$ ,  $(n, n, 1)$  et  $(n, n - 1, 2)$ , donc les codes triviaux de 6.2.1.

8.8. — La première assertion résulte directement de a). La seconde s'en déduit en l'appliquant à l'extension paire de  $C$ .

8.9. — Il suffit de grouper les éléments suivant leur distance au centre : on a un élément à distance 0 du centre,  $n$  à distance 1, etc.

8.10. —  $C(n, 0) = 2^n$ .

8.11. — Prendre des codes triviaux.

8.12. — Utiliser l'exercice 8.4.

8.13. — Montrer d'abord que, pour  $x$  fixé, l'ensemble des  $y$  tels que  $(x, y)$  appartienne au domaine des codes est un intervalle fermé  $\llbracket 0, f(x) \rrbracket$ . Puis, voir que  $f$  est décroissante.

8.14. — Si  $\pi(\mathbf{n}) = \pi(\mathbf{n}')$ , alors  $\mathbf{n} - \mathbf{n}' \in C$ , donc  $w(\mathbf{n} - \mathbf{n}') > 2t$ .

9.1. — Si  $Q(\alpha)R(\alpha) = 0$ , alors soit  $Q(\alpha) = 0$ , soit  $R(\alpha) = 0$ .

9.2. — Si  $(ab) \cdot 1_K = 0$ , alors soit  $a \cdot 1_K = 0$ , soit  $b \cdot 1_K = 0$ .

9.3. — L'application  $f : \alpha \mapsto \alpha^{p^r}$  est un automorphisme du corps K.

9.4. — Si  $n = qr + s$ , alors modulo  $p^r - 1$ , on a  $p^n \equiv p^s$ . C'est aussi un cas particulier de l'exercice 1.12.

9.5. — On a  $f(1)^2 = f(1)$ , donc  $f(1) = 1$  ou  $f(1) = 0$ . Mais dans ce dernier cas,  $f(x) = 0$  pour tout  $x$  est  $f$  n'est pas bijectif.

9.6. — Prendre  $x + y = 0$ , ou  $xy = 1$ .

9.7. — Soit  $f$  un automorphisme de  $\mathbf{R}$ . L'image par  $f$  d'un carré est un carré, donc  $f$  respecte l'ensemble des éléments positifs. Ainsi,  $f$  respecte la relation d'ordre, donc est continu. Or il vaut l'identité sur le sous-ensemble dense  $\mathbf{Q}$ .

9.8. — La valeur absolue de chacun des nombres complexes  $x - e^{2\pi i k/n}$  est  $> (x - 1)$ .

9.9. —  $\Phi_p(X) = X^{p-1} + \cdots + 1$ .

9.10. — Pour que  $\zeta$  soit une racine primitive  $(p^r)$ -ième de l'unité, il faut et il suffit que  $\zeta^{p^{r-1}}$  soit une racine primitive  $p$ -ième de l'unité.

9.11. — Il suffit de se demander ce que sont les racines primitives  $2n$ -ièmes de l'unité. On distingue deux cas suivant la parité de  $n$  et on obtient  $\Phi_{2n}(X) = \Phi_n(X^2)$  si  $n$  est pair, et  $\Phi_{2n}(X) = \Phi_n(-X)$  si  $n$  est impair.

9.12. — Non, voir l'exercice 2.22.

9.13. — Non, c'est faux pour  $n = 105$ .

9.14. — Pour 2), raisonner comme dans la démonstration de la proposition 9.5. Pour 3), noter que l'ensemble des  $y$  avec  $yxy^{-1} = x$  est un sous-corps de K contenant Z et appliquer 2). Par conséquent, ou bien  $x \in Z$ , ou bien  $\#C(x)$  est divisible par  $\Phi_n(q)$ . Puisque  $\#K^*$  est divisible par  $\Phi_n(q)$ , cela donne 4).

9.15. — Écrire  $X^n - 1 = \Phi_n(X)Q(X)$  et dériver.

9.16. — On choisit une racine primitive  $n$ -ième de l'unité, soit  $\zeta$ . Si  $g$  est un automorphisme de  $\mathbf{R}_n$ , alors  $g(\zeta)$  est aussi une racine primitive  $n$ -ième de l'unité et il existe  $m \in (\mathbf{Z}/n\mathbf{Z})^*$  tel que  $g(\zeta) = \zeta^m$ . Cette formule s'étend aussitôt à toutes les puissances de  $\zeta$ , donc à toutes les racines  $n$ -ièmes de l'unité. Cela prouve que  $m$  ne dépend pas du choix de  $\zeta$ , et détermine  $g$ . Inversement, si on se donne un tel  $m$ , alors  $\zeta^m$  est une racine primitive de l'unité, donc une racine de  $\Phi_n$  dans  $\mathbf{R}_n$ . Il existe donc un automorphisme de  $\mathbf{R}_n$  qui applique l'élément primitif  $\zeta$  sur  $\zeta^m$ . Cela prouve la bijectivité de l'application  $u$ . Le fait que c'est un homomorphisme de groupes est clair : si  $g(\zeta) = \zeta^m$  et  $g'(\zeta) = \zeta^{m'}$ , on a  $(g \circ g')(\zeta) = g(\zeta^{m'}) = g(\zeta)^{m'} = \zeta^{mm'}$ .

9.17. — Utiliser la multiplication par  $p$  modulo  $p^r - 1$ .

9.18. — La première assertion est claire. Par ailleurs, puisque  $2 \cdot 1_{\mathbf{F}_p}$  est inversible, la donnée du couple  $(s, t)$  équivaut à celle du couple  $(s, \delta)$ , et il y a  $(p-1)/2$  non-carrés.

9.19. — On obtient modulo  $P$ ,  $X^{p^i} = X + ia$ .

9.20. — Seul  $K$  est un corps, seul  $\mathbf{Z}/4\mathbf{Z}$  a des éléments d'ordre (additif) 4.

9.21. — Il y en a un autre, à savoir  $\mathbf{F}_2[X]/(X^2)$ .

9.22. —  $R_2(k) = k$ .

9.23. — Il s'agit par exemple de résoudre l'équation  $y^2 + y + 1 = 0$  dans le corps  $\mathbf{F}_q[X]/(X^2 + 1) = \mathbf{F}_q(i)$ , où  $i^2 = -1$ . Posant  $y = a + bi$ , on aboutit à l'extraction d'une racine carrée de 3. Si  $p$  est la caractéristique de  $k$ , donc si  $q = p^s$ , on a nécessairement  $p \equiv \pm 1 \pmod{12}$ ; sinon en effet, on aurait  $p \equiv \pm 3 \pmod{12}$ ; mais les puissances de  $\pm 3$  modulo 12 sont toutes égales à  $-3$ , ce qui contredit l'hypothèse faite sur  $q$ . Par conséquent, on a  $(\frac{3}{p}) = 1$ , et 3 est bien un résidu quadratique modulo  $p$ , ce qui achève la démonstration. Notons d'ailleurs que, puisque  $q \equiv -1 \pmod{12}$ , on a aussi  $R_{12}(\mathbf{F}_q) = \mathbf{F}_{q^2}$ , et on pourrait comparer séparément  $R_4(\mathbf{F}_q)$  et  $R_3(\mathbf{F}_q)$  à  $R_{12}(\mathbf{F}_q)$ .

9.24. — Courage !

9.25. — Courage !

9.27. — Si  $P$  a une racine multiple dans une extension de  $K$ , le carré du polynôme minimal de cette racine divise  $P$ . Donc (iii) implique (iv). Par ailleurs, la condition (i) passe automatiquement à toute extension de  $K$  (identité de Bézout).

9.28. — On décompose  $P$  en facteurs irréductibles.

9.29. — Notons  $b_n$  ce nombre. On considérant le degré de  $R$  dans l'expression  $P = Q^2R$ , prouver que  $p^n = b_n + pb^{n-2} + p^2b_{n-4} + \dots$  et résoudre.

9.30. —  $P$  est sans facteur multiple, et chacun de ses facteurs irréductibles divise un et un seul des  $Q - \alpha$ .

9.31. — Soient  $P_1$  et  $P_2$  deux facteurs irréductibles distincts de  $P$ . On note  $K_1$  et  $K_2$  les corps, chacun à  $p^r$  élément, quotients de  $\mathbf{F}_p[X]$  par  $P_1$  et  $P_2$  et on considère l'application de l'espace vectoriel des polynômes de degré  $< 2r$  dans le produit  $K_1 \times K_2$  obtenue par réduction modulo  $P_1$  et  $P_2$ . Montrer d'abord qu'elle est bijective (car injective entre deux espaces de même dimension). Si on note  $(q_1, q_2)$  l'image de  $Q$ , remarquer que dire que la décomposition proposée est triviale signifie que  $q_1$  et  $q_2$  sont tous deux nuls, ou tous deux résidus quadratiques, où tous deux non résidus. Conclure.

10.1. — Dire que cette distance minimale vaut  $d$ , c'est dire que toute famille de  $d - 1$  colonnes distinctes de  $P$  est linéairement indépendante, et qu'il existe une famille de  $d$  colonnes linéairement dépendante. Pour que  $d > 2$ , il faut et il suffit que les colonnes de  $P$  soient non nulles et deux à deux non proportionnelles.

10.2. — Il y en a  $n = (q^r - 1)/(q - 1)$ . On applique ensuite l'exercice 10.1.

10.3. — Avec les notations précédents, si  $\mathbf{m}_1 + \mathbf{e}_1$  et  $\mathbf{m}_2 + \mathbf{e}_2$  diffèrent au plus en  $f$  places connues, on a  $w(\mathbf{m}_1 - \mathbf{m}_2) \leq 2t + f$ .

10.4. — On a un élément à distance 0 du centre,  $n(q - 1)$  à distance 1, etc.

10.5. — On retrouve comme il se doit  $(1 + (q - 1))^n = q^n$ .

10.6. — On a en effet  $d \geq 3$  et  $n = (q^r - 1)/(q - 1)$  et  $k = n - r$ .

10.7. — Utiliser la multiplication par les scalaires.

10.8. — Examiner la liste.

10.9. — Utiliser la majoration de Griesmer.

10.10. — Évident.

10.11. — C'est clair, puisque  $\alpha \neq 0$  implique  $\alpha^2 = 1$ .

10.12. — Clair.

10.13. — On trouve  $(q - 1)\binom{n}{d}$ . Remarquer que deux mots de code de poids  $d$  et de même support sont nécessairement proportionnels, car sinon on pourrait fabriquer un mot plus court.

10.14. — La somme des vecteurs  $(1, 0, 1, 1, 1, 0, \dots)$  et  $(0, 1, 1, 1, 1, 0, \dots)$  vaut  $(1, 1, 0, 0, \dots)$ .

10.15. — En fixant  $d - 1$  composantes d'un mot de poids minimal, prouver que la dimension du code est au moins  $n - d + 1$ .

10.16. — La démonstration reste valable telle quelle.

10.17. — En codimension 1, on trouve certains des codes de parités. En dimension 1, on trouve certains des codes de répétition.

10.18. —  $1 + X$  pour le code de parité,  $1 + X + \dots + X^{n-1}$  pour le code de répétition pure,  $1 + X + X^3$  pour le code de Hamming,  $(1 + X)(1 + X + X^3)$  pour le code simplexe.

10.19. — On a  $\deg(ug) = \deg(X^{n-k}a) = (n - k) + \deg(a) < (n - k) + (k - n - n') = n'$ .

10.20. — Raisonner comme dans 9.4, en utilisant 9.5.

10.21. — Pour la symétrie, noter qu'il existe  $r$  avec  $q^r \equiv 1 \pmod{n}$ .

10.22. — L'ensemble des  $t \in \mathbf{Z}$  tels que  $q^t j \equiv j \pmod{n}$  est un sous-groupe de  $\mathbf{Z}$ , qui contient  $r$ . Il est de la forme  $s\mathbf{Z}$  où  $s$  divise  $r$ .

10.23. — On obtient respectivement : le code total, le code nul, le code de parité, le code de répétition pure.

10.24. — Par multiplication par 2, on obtient la chaîne  $1 \mapsto 2 \mapsto 4 \mapsto 3 \mapsto 1$ . Il n'y a donc que deux classes cyclotomiques, celle qui est réduite à 0 et son complémentaire. Les seuls codes possibles sont donc les quatre codes triviaux.

10.25. — On obtient le second code en renversant l'ordre des bits dans chacun des mots du premier code.

10.26. — Dans le corps  $R_7(\mathbf{F}_2) = \mathbf{F}_8$ , il y a deux types de racines primitives 7-ièmes de l'unité. Trois de ces racines ont  $1 + X + X^3$  comme polynôme minimal sur  $\mathbf{F}_2$ , les trois autres  $1 + X^2 + X^3$ . La valeur de  $g$  dépend du choix initial de  $\alpha$ .

10.27. — La partie formée des opposés des éléments qui n'appartiennent pas à  $\Sigma$ .

10.28. — Si  $m = ag$ , alors  $me = a(ge) \equiv ag = m$ . Si  $m \equiv me = (mc)g$ , on  $m \in C$ .

10.29. — Pour  $m \in C$  on a  $me \equiv m$ . Donc, raisonnant comme à l'habitude, on a  $e'e \equiv e$  et de même  $e'e \equiv e'$ , etc.

10.30. — C'est le polynôme réciproque de  $1 - e$ .

11.1. — Appliquer la proposition 10.18. On trouve la partie « paire » d'un code RS.

11.2. — Pour  $\delta = 3$ , on a un code de Hamming. Prenons  $\delta = 5$ . Alors  $\Sigma$  est la réunion des classes cyclotomiques  $\Sigma_1$  et  $\Sigma_3$ . Il s'agit de vérifier, d'une part que 3 n'est pas de la forme  $2^i$  modulo  $2^m - 1$ , et d'autre part que si  $r$  est impair, on a  $3 \cdot 2^i \not\equiv 3 \pmod{2^r - 1}$  pour  $0 < i < r$ .

11.3. — Parce que toute fonction de la forme  $\sum_{x \in A} \phi(x)$  est linéaire relativement à la partie A.

11.4. — Même calcul que dans le cas  $q = 2$ .

12.1. — On peut écrire  $e_r(i, j) = n(i + rj) + j$  et utiliser la division euclidienne par  $n$ .

12.2. — On remplace X par  $\alpha^i$ , pour  $i = 1, \dots, 4$ .

12.3. — Pas de difficulté particulière.

12.4. — Les coefficients sont les  $\alpha^{in_j}$ .

13.1. — Une boule de rayon 2 sur  $\mathbf{F}_3$  a  $1 + 2 \times 11 + 2^2 \times \binom{11}{2} = 243 = 3^5$  éléments.

13.2. — Si  $c = 0$ , on a  $ad = 1$ , donc  $a/d = a^2$ .

13.3. — Il est formé des matrices diagonales  $\begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix}$ . En effet, si  $ax + b/cx + d = a'x + b'/c'x + d'$  pour tout  $x$ , alors il existe  $\lambda \in \mathbf{F}_p$  avec  $a' = \lambda a$ ,  $b' = \lambda b$ , etc. (utiliser le fait qu'il y a au moins trois points dans  $\mathbf{P}^1(\mathbf{F}_p)$ ).

13.4. — Utiliser l'exercice précédent. Avec les notations ci-dessus, noter ensuite que  $a'd' - b'c' = \lambda^2(ad - bc)$ .

13.5. — On trouve 6 pour  $p = 2$  et  $\frac{(p-1)p(p+1)}{2}$  pour  $p > 2$ . Pour  $p = 2$ , c'est le groupe symétrique tout entier, pour  $p = 3$ , c'est le groupe alterné.

13.6. — Même méthode, avec les modifications qui s'imposent : remarquer notamment que  $m_r$  contient 0.

13.7. — On a  $4096 - 1 - 759 - 759 - 1 = 2576$ .

13.9. — Si  $-\mathbf{a} \in C$ , alors  $\mathbf{a} \in C$ .

13.10. — Il y en a 48 : compter le nombre de familles et comparer à  $a$ ).

13.11. — Il suffit de vérifier qu'étant donnés  $x$  et  $y$  distincts, on peut envoyer 0 sur  $x$  et  $\infty$  sur  $y$ , ce qui amène à résoudre  $b/d = x$ ,  $a/c = y$  et  $ad - bc = 1$ .

13.13. — Si  $\mathbf{m} = m_1 e'_1 + \cdots + m_n e'_n$ , on a  $\langle\!\langle \mathbf{m} | e_i \rangle\!\rangle = m_i$ .

13.14. — Aucune difficulté particulière.

# Table des matières

## Introduction

I

## I. Primalité

<b>Introduction à la première partie</b>	<b>9</b>
<b>Chapitre I. Trois algorithmes fondamentaux</b>	<b>15</b>
I.1. Réécriture . . . . .	15
I.1.1. Règles de réécriture . . . . .	15
I.1.2. Réécriture et déterminisme . . . . .	17
I.1.3. Traduction dans un langage de programmation . . . . .	19
I.1.4. Récursion et itération . . . . .	21
I.2. Calcul rapide des puissances . . . . .	23
I.2.1. Des poids faibles vers les poids forts . . . . .	24
I.2.2. Des poids forts vers les poids faibles . . . . .	25
I.3. Complexité des algorithmes arithmétiques . . . . .	27
I.3.1. Coût des opérations arithmétiques élémentaires . . . . .	27
I.3.2. Congruences dans $\mathbf{Z}$ . . . . .	28
I.3.3. Le coût du calcul modulaire . . . . .	30
I.3.4. Le coût de l'exponentielle . . . . .	31
I.4. Algorithmes d'Euclide . . . . .	33
I.4.1. L'algorithme de base . . . . .	33
I.4.2. L'algorithme d'Euclide binaire . . . . .	36
I.4.3. La suite de Fibonacci . . . . .	37
I.4.4. Le coût de l'algorithme d'Euclide . . . . .	38
I.4.5. L'algorithme d'Euclide étendu . . . . .	38
I.4.6. Le coût de l'algorithme d'Euclide étendu . . . . .	40
I.5. Algorithmes d'Euclide pour les polynômes . . . . .	41
I.5.1. Polynômes en une variable . . . . .	41
I.5.2. Division euclidienne des polynômes . . . . .	42
I.5.3. Algorithmes d'Euclide . . . . .	42
I.6. Algorithmes de recherche d'une période . . . . .	44
I.6.1. Exemple : écriture décimale d'un nombre rationnel . . . . .	44
I.6.2. Période d'une suite récurrente . . . . .	45
I.6.3. Algorithme de Floyd . . . . .	46

1.6.4. Algorithme de Brent . . . . .	47
1.6.5. La méthode de factorisation $\rho$ de Pollard . . . . .	48
<b>Chapitre 2. Théorème de Fermat et primalité</b>	<b>51</b>
2.1. Théorème chinois . . . . .	51
2.1.1. L'énoncé : forme classique . . . . .	51
2.1.2. L'énoncé : forme abstraite . . . . .	52
2.1.3. Les démonstrations du théorème chinois . . . . .	52
2.1.4. Un algorithme . . . . .	53
2.2. L'indicateur d'Euler . . . . .	54
2.2.1. Le groupe $(\mathbf{Z}/n\mathbf{Z})^*$ . . . . .	54
2.2.2. L'indicateur d'Euler . . . . .	55
2.3. Le petit théorème de Fermat . . . . .	58
2.3.1. Ordre d'un élément d'un groupe . . . . .	58
2.3.2. Le petit théorème de Fermat . . . . .	59
2.3.3. Une application du théorème de Fermat à la factorisation . . . . .	61
2.3.4. Le théorème d'Euler . . . . .	62
2.3.5. Cryptographie à clés publiques et nombres premiers : la méthode RSA . . . . .	63
2.3.6. Critères de non-primalité tirés du petit théorème de Fermat . . . . .	66
2.3.7. Le critère de Miller-Rabin . . . . .	68
<b>Chapitre 3. Racines primitives</b>	<b>71</b>
3.1. Structure du groupe $K^*$ . . . . .	71
3.1.1. Groupes cycliques . . . . .	71
3.1.2. Exposant d'un groupe commutatif fini . . . . .	71
3.1.3. Racines primitives de l'unité . . . . .	73
3.1.4. Racines primitives modulo $p$ . . . . .	74
3.1.5. Recherche des racines primitives . . . . .	75
3.2. Critères de primalité . . . . .	76
3.2.1. Critères de primalité « à la Lehmer » . . . . .	76
3.2.2. Certificats de primalité . . . . .	78
3.2.3. Les nombres de Fermat . . . . .	79
3.2.4. Nombres de Mersenne . . . . .	81
3.2.5. Suites de Lucas . . . . .	82
3.2.6. Construction d'anneaux par adjonction . . . . .	84
3.2.7. Le critère de primalité de Lucas-Lehmer . . . . .	85
3.2.8. Critère de primalité des nombres de Mersenne . . . . .	87
3.3. Indicateur et nombres de Carmichael . . . . .	89
3.3.1. Nombres de Carmichael . . . . .	89
3.3.2. L'indicateur de Carmichael . . . . .	90

3.3.3. Structure du groupe $(\mathbf{Z}/p^n\mathbf{Z})^*$ , $p$ premier impair . . . . .	91
3.3.4. Structure du groupe $(\mathbf{Z}/2^n\mathbf{Z})^*$ . . . . .	92
3.3.5. Calcul de l'indicateur de Carmichael . . . . .	93
3.3.6. Preuve du théorème de Rabin . . . . .	94
<b>Chapitre 4. Transformation de Fourier rapide</b>	<b>99</b>
4.1. Transformation de Fourier discrète . . . . .	99
4.1.1. Racines principales de l'unité . . . . .	99
4.1.2. L'anneau $\mathbf{A}[X]/(X^n - 1)$ . . . . .	100
4.1.3. Définition de la transformation de Fourier . . . . .	101
4.2. Transformation de Fourier rapide . . . . .	102
4.2.1. Le principe . . . . .	102
4.2.2. L'algorithme . . . . .	104
4.3. Applications . . . . .	105
4.3.1. Transformation de Fourier rapide modulo N . . . . .	105
4.3.2. Applications arithmétiques . . . . .	106
4.3.3. Multiplication des grands entiers . . . . .	107
4.3.4. La méthode de Pollard . . . . .	108
4.3.5. La méthode de Schönhage-Strassen . . . . .	109
<b>Chapitre 5. Résidus quadratiques et applications</b>	<b>111</b>
5.1. Résidus quadratiques . . . . .	111
5.1.1. Carrés dans un corps fini . . . . .	111
5.1.2. Le symbole de Legendre . . . . .	112
5.1.3. Calcul d'une racine carrée dans $\mathbf{Z}/p\mathbf{Z}$ . . . . .	115
5.1.4. Carrés dans $\mathbf{Z}/p^n\mathbf{Z}$ . . . . .	116
5.1.5. Les signes $\varepsilon(n)$ , $\omega(n)$ et $\theta(a, b)$ . . . . .	117
5.2. Réciprocité quadratique . . . . .	118
5.2.1. Deux exemples . . . . .	118
5.2.2. Sommes de Gauss . . . . .	120
5.2.3. La loi de réciprocité quadratique . . . . .	121
5.3. Symbole de Jacobi . . . . .	122
5.3.1. Définition et réciprocité . . . . .	122
5.3.2. Algorithmes de calcul du symbole de Jacobi . . . . .	124
5.3.3. Le critère de Solovay et Strassen . . . . .	126
5.3.4. Tests probabilistes de primalité . . . . .	127
5.3.5. Comparaison des algorithmes de Miller-Rabin et de Solovay-Strassen . . . . .	129
5.4. Algorithmes probabilistes . . . . .	130
5.4.1. Parties de type $\mathcal{P}$ . . . . .	130
5.4.2. Parties de type $\mathcal{NP}$ . . . . .	131
5.4.3. Parties de type $\mathcal{RP}$ . . . . .	133

5.4.4. Le cas des nombres premiers . . . . .	134
<b>Pour aller plus loin sur la primalité</b>	<b>137</b>

## II. Codes correcteurs

<b>Introduction à la deuxième partie</b>	<b>141</b>
<b>Chapitre 6. Codes binaires</b>	<b>147</b>
6.1. Définitions générales . . . . .	147
6.1.1. Le corps $\mathbf{F}_2$ . . . . .	147
6.1.2. Le modèle . . . . .	147
6.1.3. Le modèle d'erreur : le canal binaire symétrique . . . . .	149
6.1.4. Le décodage . . . . .	149
6.1.5. Codes linéaires . . . . .	151
6.2. Exemples . . . . .	151
6.2.1. Les codes triviaux : les cas $k = 0, 1, n - 1, n$ . . . . .	151
6.2.2. Codes $t$ -correcteurs, capacité de correction, codes parfaits . . . . .	152
6.2.3. Le code de Hamming $H_3$ de longueur 7 . . . . .	154
6.3. Outils de construction de codes . . . . .	155
6.3.1. Constructions élémentaires . . . . .	155
6.3.2. Extension paire . . . . .	156
6.3.3. Orthogonal d'un code . . . . .	157
6.4. Matrices génératrices et vérificatrices d'un code linéaire . . . . .	158
6.4.1. Matrices génératrices et vérificatrices . . . . .	159
6.4.2. Changements de base . . . . .	159
6.4.3. Conditions de parité généralisées . . . . .	160
6.4.4. Extension paire . . . . .	161
6.4.5. Matrice vérificatrice et syndrome . . . . .	161
6.5. Les codes binaires de Hamming . . . . .	162
6.5.1. Construction . . . . .	162
6.5.2. Les codes de Hamming étendus . . . . .	164
<b>Chapitre 7. Codes, combinatoire, géométrie</b>	<b>167</b>
7.1. Les codes binaires comme codes de parties . . . . .	167
7.1.1. Traductions . . . . .	167
7.1.2. Un exemple . . . . .	168
7.2. Codes d'hyperplans affines . . . . .	170
7.2.1. Hyperplans . . . . .	171
7.2.2. L'orthogonal du code de Hamming : le code simplexe . . . . .	171
7.2.3. L'orthogonal du code de Hamming étendu : le code $R_{1,m}$ . . . . .	172

7.3. Codes de Reed-Muller . . . . .	173
7.3.1. Fonctions booléennes et polynômes . . . . .	173
7.3.2. Définition des codes de Reed-Muller . . . . .	174
7.3.3. Description itérative des codes de Reed-Muller . . . . .	175
7.3.4. Mots de poids minimal du code $R_{r,m}$ . . . . .	176
7.4. Géométries finies . . . . .	178
7.4.1. Corps finis . . . . .	178
7.4.2. Espaces affines ou projectifs finis . . . . .	178
7.4.3. Plans projectifs « abstraits » . . . . .	180
7.5. Systèmes de Steiner . . . . .	181
7.5.1. Définition des systèmes de Steiner . . . . .	181
7.5.2. Exemples . . . . .	181
7.5.3. Codes et systèmes de Steiner . . . . .	182
7.6. Automorphismes . . . . .	184
7.6.1. Exemple : les codes cycliques . . . . .	185
7.6.2. Exemple : les transformations affines . . . . .	186
7.6.3. Codes et géométries finies . . . . .	187
<b>Chapitre 8. Majorations de la taille des codes</b>	<b>189</b>
8.1. Conditions nécessaires . . . . .	189
8.1.1. Majorations, codes optimaux . . . . .	189
8.1.2. Projections et raccourcissements d'un code . . . . .	190
8.1.3. Majoration de Griesmer . . . . .	191
8.1.4. Majorations de Plotkin . . . . .	192
8.1.5. Majoration de Hamming . . . . .	192
8.2. Conditions de Gilbert-Varshamov . . . . .	193
8.3. Formes asymptotiques . . . . .	194
8.3.1. Le diagramme $(d/n, k/n)$ . . . . .	194
8.3.2. Le domaine des codes . . . . .	195
8.3.3. Préliminaires . . . . .	196
8.3.4. Existence de bons codes . . . . .	198
8.3.5. Bonnes familles de codes . . . . .	200
8.4. Et Shannon dans tout cela ? . . . . .	201
8.4.1. L'approche probabiliste . . . . .	201
8.4.2. Le décodage total . . . . .	202
8.4.3. Le théorème de Shannon et sa réciproque . . . . .	203
8.4.4. Quel rapport avec ce qui précède ? . . . . .	204
8.4.5. Une idée de la démonstration . . . . .	204
8.4.6. Moralité . . . . .	206

<b>Chapitre 9. Les corps finis</b>	<b>207</b>
9.1. Structure des corps finis . . . . .	207
9.1.1. Éléments algébriques, polynômes minimaux . . . . .	207
9.1.2. Construction de corps par adjonction . . . . .	208
9.1.3. Corps premiers . . . . .	209
9.1.4. Structure des corps finis . . . . .	210
9.1.5. Polynômes minimaux sur $\mathbf{F}_p$ . . . . .	211
9.1.6. Automorphismes d'un corps fini . . . . .	212
9.2. Polynômes cyclotomiques . . . . .	212
9.2.1. Le polynôme $\Phi_n$ . . . . .	213
9.2.2. Racines des polynômes cyclotomiques . . . . .	214
9.2.3. Irréductibilité sur $\mathbf{Q}$ des polynômes cyclotomiques . . . . .	215
9.2.4. Corps cyclotomiques . . . . .	216
9.2.5. Décomposition des polynômes cyclotomiques dans un corps fini . . . . .	217
9.3. Construction des corps finis . . . . .	219
9.3.1. Polynômes irréductibles sur $\mathbf{F}_p$ . . . . .	219
9.3.2. Relation entre ordre et degré . . . . .	221
9.3.3. « Le » corps à $q$ éléments . . . . .	222
9.3.4. Racines de l'unité dans un corps fini . . . . .	224
9.4. Calculs explicites dans un corps fini . . . . .	225
9.4.1. Les corps à $2^m$ éléments . . . . .	225
9.4.2. Exemple : le corps $\mathbf{F}_{16}$ . . . . .	225
9.4.3. Le logarithme de Zech . . . . .	227
9.5. Démonstration du théorème AKS . . . . .	228
9.6. Décomposition des polynômes dans $\mathbf{F}_p[X]$ . . . . .	231
9.6.1. Polynômes sans facteur multiple . . . . .	231
9.6.2. L'algorithme de Berlekamp . . . . .	232
9.6.3. Une variante probabiliste . . . . .	233
9.6.4. L'algorithme de Cantor-Zassenhaus . . . . .	234
9.6.5. Décomposition des polynômes dans $\mathbf{Q}[X]$ . . . . .	235
<b>Chapitre 10. Codes linéaires cycliques</b>	<b>237</b>
10.1. Codes linéaires sur $\mathbf{F}_q$ . . . . .	237
10.1.1. Paramètres d'un code linéaire . . . . .	237
10.1.2. Décodage . . . . .	238
10.1.3. Codes parfaits . . . . .	238
10.1.4. Codes sur $\mathbf{F}_q$ et codes sur $\mathbf{F}_p$ . . . . .	239
10.1.5. Un exemple . . . . .	240
10.1.6. Extension paire . . . . .	241
10.1.7. Orthogonal . . . . .	241

10.2. Codes de type MDS . . . . .	241
10.2.1. La majoration de Singleton . . . . .	241
10.2.2. Codes triviaux . . . . .	242
10.2.3. Raccourcissement . . . . .	242
10.2.4. Première construction des codes de Reed-Solomon . . . . .	244
10.3. Codes cycliques . . . . .	245
10.3.1. Définitions . . . . .	245
10.3.2. Représentation des mots par des polynômes . . . . .	246
10.3.3. Générateurs minimaux des codes cycliques . . . . .	247
10.3.4. Codages pour un code cyclique . . . . .	249
10.3.5. Constructions élémentaires . . . . .	249
10.4. Classes cyclotomiques ( $n$ premier à $q$ ) . . . . .	250
10.4.1. Diviseurs de $X^n - 1$ . . . . .	251
10.4.2. Classes cyclotomiques . . . . .	251
10.4.3. Exemples . . . . .	252
10.4.4. Distance minimale des codes cycliques . . . . .	253
10.4.5. Idempotents des codes cycliques . . . . .	254
<b>Chapitre 11. Codes BCH</b>	<b>257</b>
11.1. Codes cycliques usuels . . . . .	257
11.1.1. Codes de Hamming binaires . . . . .	257
11.1.2. Les codes de Reed-Solomon comme codes cycliques . . . . .	258
11.1.3. L'autre description des codes de Reed-Solomon . . . . .	259
11.1.4. Codes de Reed-Solomon raccourcis . . . . .	260
11.1.5. Codes BCH . . . . .	260
11.2. Codes BCH binaires stricts . . . . .	262
11.2.1. Construction des codes BCH binaires stricts . . . . .	262
11.2.2. Exemple : les codes BCH binaires de longueur 15 . . . . .	263
11.2.3. Une autre définition des codes BCH binaires stricts . . . . .	264
11.2.4. Automorphismes d'un code BCH binaire strict étendu . . . . .	265
11.3. L'algorithme de décodage des codes BCH binaires . . . . .	265
11.3.1. Position du problème . . . . .	266
11.3.2. Syndrome . . . . .	266
11.3.3. L'équation-clé . . . . .	267
11.3.4. Résolution de l'équation-clé . . . . .	267
11.4. L'algorithme de décodage des codes BCH généraux . . . . .	268
11.4.1. Correction de $t$ erreurs, $t < \delta/2$ . . . . .	269
11.4.2. Correction de $f$ effacements, $f < \delta$ . . . . .	269

<b>Chapitre 12. Le codage des disques compacts</b>	<b>271</b>
12.1. Position du problème . . . . .	271
12.1.1. La chaîne de codage . . . . .	271
12.1.2. Les contraintes du codage correcteur . . . . .	273
12.2. Le code CIRC . . . . .	273
12.2.1. Entrelacement . . . . .	273
12.2.2. Le codage . . . . .	274
12.2.3. Le décodage . . . . .	275
12.2.4. Les détails du codage et du décodage . . . . .	276
12.3. Au delà du code CIRC . . . . .	278
<b>Chapitre 13. Codes de résidus quadratiques</b>	<b>279</b>
13.1. Codes de résidus quadratiques . . . . .	279
13.1.1. Définition générale . . . . .	279
13.1.2. Idempotents des codes QR binaires . . . . .	281
13.1.3. Codes QR binaires étendus . . . . .	282
13.1.4. Automorphismes d'un code QR binaire étendu . . . . .	282
13.1.5. Distance minimale d'un code QR binaire . . . . .	285
13.2. Les codes binaires de Golay $G_{23}$ et $G_{24}$ . . . . .	286
13.2.1. Détermination des codes parfaits . . . . .	288
13.2.2. Automorphismes du code de Golay : le groupe $M_{24}$ . . . . .	290
13.2.3. Les groupes de Mathieu . . . . .	291
13.3. Codes et réseaux . . . . .	292
13.3.1. Réseaux . . . . .	292
13.3.2. Réseau déduit d'un code binaire . . . . .	293
13.3.3. Exemple : le réseau $E_8$ . . . . .	294
13.3.4. Vecteurs de carré scalaire 2 . . . . .	295
13.3.5. Réseaux et matrices symétriques entières . . . . .	296
13.3.6. Ceci n'est pas une conclusion . . . . .	297
<b>Pour aller plus loin sur les codes</b>	<b>299</b>
<b>Glossaire d'algèbre</b>	<b>301</b>
<b>Solutions des exercices</b>	<b>305</b>
<b>Bibliographie</b>	<b>329</b>
<b>Index des notations</b>	<b>331</b>

# Bibliographie

- [Bach, Shallit] Eric Bach and Jeffrey Shallit. *Algorithmic number theory, volume I*. MIT Press, Cambridge, MA, USA, 1996.
- [Berrou] Claude Berrou. *Codes et turbocodes (sous la direction de Claude Berrou)*. Iris. Springer, Paris, 2007.
- [Cohen] Henri Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, 1993.
- [Conway] J.H. Conway and N.J.A. Sloane. *Sphere Packings, Lattices and Groups*, volume 290 of *Grundlehren der mathematischen Wissenschaften*. Springer, New York, U.S.A., 3rd edition, 1999.
- [Cooper] Peter Cooper. *Beginning Ruby*. APress, Berkeley, 2007.
- [Crandall, Pomerance] Richard Crandall and Carl Pomerance. *Prime numbers. A Computational Perspective*. Springer-Verlag, New York, 2nd edition, 2005.
- [Davis, Hersh] Philip J. Davis and Reuben Hersh. *L'Univers mathématique*. Gauthier-Villars, 1985.
- [Delahaye] Jean-Paul Delahaye. *Merveilleux nombres premiers*. Belin, Pour la Science, 2000.
- [Naudin, Quitté] Patrice Naudin et Claude Quitté. *Algorithmique algébrique*. Masson, 1992.
- [Fulton] Hal Fulton. *The Ruby Way*. Addison-Wesley, 2nd edition, 2007.
- [Garey, Johnson] M. Garey and D. Johnson. *Computers and Intractability*. Freeman and Co., New York, 1979.
- [Hardy, Wright] G.H. Hardy and E.W. Wright. *An Introduction to the theory of Numbers*. Oxford University Press, 1979.
- [Knuth 1] Donald Knuth. *The Art of Computer Programming, Vol. 1*. Addison-Wesley Professional, Reading, 3rd edition, 1997.
- [Knuth 2] Donald Knuth. *The Art of Computer Programming, Vol. 2*. Addison-Wesley Professional, Reading, 3rd edition, 1997.
- [Koblitz] Neal Koblitz. *A Course in Number Theory and Cryptography*, volume 114 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin-Heidelberg-New York, 2nd edition, 1994.
- [Lin, Costello] Shu Lin and Daniel J. Costello. *Error Control Coding*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2nd edition, 2004.
- [MacKay] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
- [MacWilliams, Sloane] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*, volume 16 of *North-Holland Mathematical Library*. North-Holland, 1977.

- [Menezes et al.] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996.
- [Morelos-Zaragoza] R.H. Morelos-Zaragoza. *The Art of Correcting Coding*. John Wiley and Sons Ltd, 2002.
- [Handbook] Vera Pless, Richard A. Brualdi, and W. C. Huffman. *Handbook of Coding Theory*. Elsevier Science Inc., New York, NY, USA, 1998.
- [Ribenboim] Paulo Ribenboim. *The New Book of Prime Number Records*. Springer, 1996.
- [Riesel] Hans Riesel. *Prime Numbers and Computer Methods for Factorization*, volume 126 of *Progress in Mathematics*. Birkhäuser, 1994.
- [Schneier] Bruce Schneier. *Applied cryptography : protocols, algorithms, and source code in C*. John Wiley and sons, New York, 2nd edition, 1996.
- [Schneier(fr)] Bruce Schneier. *Cryptographie appliquée*. Vuibert, 2-e édition, 2001.
- [Serre] Jean-Pierre Serre. *Cours d'Arithmétique*. Presses Universitaires de France (PUF), 1977.
- [Sipser] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.
- [Thomas, Hunt 1] Dave Thomas and Andy Hunt. *Programming Ruby : A Pragmatic Programmer's Guide*. Addison-Wesley, 2000.
- [Thomas, Hunt 2] Dave Thomas and Andy Hunt. *Programming Ruby : A Pragmatic Programmer's Guide*. Pragmatic Bookshelf, 2nd edition, 2004.

# Index des notations

- ( $\mathbf{Z}/n\mathbf{Z}$ )<sup>\*</sup>**, groupe multiplicatif des entiers modulo  $n$ , 54  
**( $n, k, d$ )**, paramètres d'un code binaire linéaire, 151  
**( $q; n, k, d$ )**, paramètres d'un code linéaire, 237  
 $(s)_2$ , entier d'écriture binaire  $s$ , 24  
**A[X]**, anneau des polynômes en la variable  $X$ , 41  
**A ⊕ B**, différence symétrique, 167, 283  
**A\***, groupe multiplicatif d'un anneau, 303  
**C<sup>⊥</sup>**, code orthogonal, 157, 241  
**C<sup>pair</sup>**, sous-code pair, 156  
**F<sub>n</sub>**, suite de Fibonacci, 23, 37  
**G<sub>11</sub>**, code ternaire de Golay, 280  
**G<sub>23</sub>, G<sub>24</sub>**, codes de Golay, 287  
**H(x)**, entropie, 197  
**H<sub>3</sub>**, code de Hamming de longueur 7, 154  
**̄H<sub>3</sub>**, code de Hamming étendu de longueur 8, 157  
**H<sub>r</sub>**, code de Hamming, 162  
**̄H<sub>r</sub>**, code de Hamming étendu, 164  
**M<sub>11</sub>, M<sub>12</sub>, M<sub>22</sub>, M<sub>23</sub>**, groupes de Mathieu, 291  
**M<sub>24</sub>**, groupe de Mathieu, 290  
**N(K, d)**, longueur minimale d'un code linéaire de paramètres ( $?, k, d$ ), 189  
**P<sub>MR</sub>**, Propriété de Miller-Rabin, 128  
**P<sub>SS</sub>**, Propriété de Solovay-Strassen, 128  
**R<sub>1,m</sub>**, code de Reed-Muller, 172  
**R<sub>n</sub>(Q)**, corps cyclotomique, 216  
**R<sub>n</sub>(k)**, extension cyclotomique, 224  
**R<sub>r,m</sub>**, code de Reed-Muller, 174  
**S<sub>m</sub>**, code simplexe, 171  
**V( $n, r$ )**, volume de la boule, 193  
 $[x]$ , plafond du nombre réel  $x$ , 303  
 $[x]$ , plancher du nombre réel  $x$ , 303  
**F<sub>2</sub>**, corps à deux éléments, 147  
**F<sub>p</sub>**, corps premier à  $p$  éléments, 209  
**F<sub>q</sub>**, corps fini à  $q$  éléments, 223  
**Fa**, transformée de Fourier, 101  
**̄Fa**, transformée de Fourier, 101  
**Fer<sub>m</sub>**, nombre de Fermat, 79  
**GA<sub>m</sub>(F<sub>2</sub>)**, groupe affine, 186  
**GL<sub>m</sub>(F<sub>2</sub>)**, groupe linéaire, 186  
 $(\frac{m}{n})$ , symbole de Jacobi, 123  
 $(\frac{n}{p})$ , symbole de Legendre, 113  
**N**, ensemble entiers naturels, 29  
**N $\mathcal{P}$** , partie de type  $N\mathcal{P}$ , 132  
**P<sub>m-1</sub>(K)**, espace projectif, 178  
**P<sup>1</sup>(F<sub>p</sub>)**, droite projective sur F<sub>p</sub>, 283  
**P**, partie de type P, 131  
**PSL<sub>2</sub>(F<sub>p</sub>)**, groupe spécial projectif, 283  
**P<sub>err</sub>(C, p)**, probabilité d'erreur d'un code, 203  
**Φ<sub>n</sub>(X)**, polynôme cyclotomique, 213  
**Q**, corps des nombres rationnels, 209  
**R $\mathcal{P}$** , partie de type R $\mathcal{P}$ , 133  
**Z**, anneau des entiers, 29  
**Z/NZ**, anneau des entiers modulo N, 29  
**Z $\mathcal{P}\mathcal{P}$** , partie de type Z $\mathcal{P}\mathcal{P}$ , 134  
 $\chi_A$ , fonction caractéristique d'une partie, 167  
**coN $\mathcal{P}$** , partie de type coN $\mathcal{P}$ , 132  
**deg(P)**, degré d'un polynôme, 41  
 $\varepsilon(\mathbf{m})$ , parité (généralisée), 241  
 $\varepsilon(\mathbf{m})$ , parité d'un mot, 152, 156  
 $\varepsilon(n)$ , signe, 117  
 $a \equiv b \pmod{N}$ , congruence modulo N, 29  
 $\lambda(n)$ , indicateur de Carmichael, 90  
**log**, logarithme à base 2, 303  
 $a \pmod{N}$ , classe de congruence, 30  
 $a \pmod{b}$ , reste de la division euclidienne de  $a$  par  $b$ , 15  
 $\omega(G)$ , exposant d'un groupe fini, 72  
 $\omega(n)$ , signe, 117  
**#E**, nombre des éléments de E, 55  
**pgcd**, plus grand commun diviseur de deux entiers, 33  
**pgcd**, plus grand commun diviseur de deux polynômes, 44  
 $\pi(x)$ , syndrome du mot  $x$ , 161  
**ppcm**, plus petit commun multiple de deux entiers, 40  
 $\langle \mathbf{m} | \mathbf{n} \rangle$ , produit scalaire, 157, 241, 292  
 $\sigma(\mathbf{m})$ , permutation circulaire, 245  
 $\|n\|$ , taille d'un entier, 131  
 $\langle\langle \mathbf{m} | \mathbf{n} \rangle\rangle$ , produit scalaire, 292  
 $\binom{n}{i}$ , coefficient du binôme, 302  
 $\theta(a, b)$ , signe, 117  
 $\tilde{\cdot}$ , transposée d'une matrice, 159  
**1**, mot plein, 147  
**1<sup>[n]</sup>**, repunit, 9

- $\varphi(n)$ , indicateur d'Euler, 55  
**0**, mot nul, 147  
 $a * b$ , produit de convolution, 101  
 $d(\mathbf{m}, \mathbf{m}')$ , distance de Hamming de  
deux mots, 150, 237  
 $k(\alpha)$ , sous corps engendré, 208  
 $nx$ , multiple dans un anneau, 301  
 $q = a \div b$ , quotient entier de  $a$  par  $b$ ,  
15  
 $w(\mathbf{m})$ , poids d'un mot, 148, 237  
 $x^{-1}$ , inverse d'un élément, 302  
 $x^n$ , puissance dans un anneau, 301  
 $z(i)$ , table de Zech, 227  
AND, et logique, 147  
XOR, ou exclusif, 147

# Index

- accumulation (point), 195  
adjonction (construction par), 84, 208  
Adleman, 64  
affine  
  groupe, 186  
  hyperplan, 171  
  plan, 170  
  sous-espace, 178  
  transformation, 186  
AKS  
  algorithme, 136  
  théorème, 135, 228  
algébrique (élément), 207  
algorithme  
  AKS, 136  
  d'Euclide, 33  
  binaire, 36  
  étendu, 38  
  pour les polynômes, 43  
  étendu pour les polynômes, 43  
de Berlekamp, 232  
de Brent, 47  
de Cantor-Zassenhaus, 234  
de factorisation de Pollard, 48  
de Floyd, 46  
de Shanks, 115  
anneau, 301  
  des entiers, 29  
  des polynômes, 41  
  intègre, 301  
  nul, 302  
  quotient, 30, 84, 302  
  sous-anneau, 303  
assignée (distance), 261  
associé (réseau), 293  
auto-orthogonal (code), 158  
automorphisme  
  d'un code, 184  
  d'un corps, 212
- Bachet de Méziriac, 39  
Berlekamp (algorithme), 232  
best match, 20  
Bézout (identité), 39  
  pour les polynômes, 44  
bilinéaire (coût), 28  
binaire  
  algorithme d'Euclide, 36  
  canal binaire symétrique, 149  
  code, 148, 237  
  division euclidienne, 17
- binôme  
  coefficients, 302  
  formule, 303  
bit, 147  
bloc (code de blocs), 143  
bon  
  bonne famille de codes, 200  
  code, 198  
booléenne (fonction), 173  
Bose  
  code BCH, 260  
  distance de Bose d'un code BCH, 262  
bouffées d'erreurs, 240, 273  
boule (de Hamming), 152, 239  
Brent (algorithme), 47
- canal  
  binaire symétrique, 149  
  capacité, 203  
  de transmission, 148  
Cantor-Zassenhaus (algorithme), 234  
capacité  
  d'un canal, 203  
  de correction, 152  
caractéristique  
  d'un corps, 209, 302  
  fonction, 167, 173  
Carmichael  
  indicateur, 90  
  nombre, 89  
centre (d'un corps non commutatif), 214  
certificat court de non-primalité, 11  
certificat de primalité, 79  
Chaudhuri (code BCH), 260  
chiffrement, 63  
chinois (théorème), 51  
Church (thèse), 131  
CIRC (code), 275  
classe  
  de congruence, 29  
  cyclotomique, 251  
clé (publique), 63  
codage, 147  
  du code CIRC, 274  
  systématique, 260

- code  
*t*-correcteur, 152, 238  
 auto-orthogonal, 158  
 BCH, 261  
   étendu, 264  
   généralisé, 261  
   strict, 261  
 binaire, 148, 237  
 bon, 198  
 cyclique, 185, 246  
 de blocs, 143  
 de Golay  
    $G_{11}$ , 280  
    $G_{23}$ , 287  
    $G_{24}$ , 287  
 de Hamming, 154, 162  
   sur  $\mathbf{F}_q$ , 238  
   étendu, 157, 164  
 de parité, 152  
 de parties d'un ensemble fini, 168  
 de Reed-Muller, 174  
 de Reed-Salomon, 245, 259  
 de répétition, 151  
 de résidus quadratiques, 280  
 de type MDS, 242  
 de type RS, 259  
 de type RM, 174  
 domaine des codes, 195  
 dual, 157  
 impair, 156  
 linéaire binaire, 151  
 linéaire sur  $\mathbf{F}_q$ , 237  
 nul, 152  
 optimal, 189  
 orthogonal, 157, 241  
 pair, 156, 241  
 parfait, 153, 239  
 plein, 151, 242  
 projeté, 190  
 QR, 280  
   étendu, 282  
 raccourci, 190, 243  
 simplexe, 171  
 ternaire, 237  
   de Golay, 280  
 trivial, 152, 242  
 codimension d'un code linéaire, 151  
 coefficients (du binôme), 302  
 composé (nombre), 9  
 concaténation de deux mots, 155  
 congru (entiers), 29  
 congruence  
   classe, 29  
   dans  $\mathbf{Z}$ , 29  
   pour les polynômes, 84  
 convolution (produit), 101  
 corps, 59, 207, 302  
   cyclotomique, 216  
   fini, 30, 60, 178, 207  
   premier, 209  
   sous-corps, 303  
 correcteur (code *t*-correcteur), 152, 238  
 correction  
   capacité, 152  
   d'un algorithme, 16  
   d'une erreur, 150, 238  
 courbe de Gilbert-Varshamov, 198  
 coût  
   modèle à coûts bilinéaires, 28  
   modèle à coûts fixes, 27  
 critère  
   de Lehmer, 77  
   de Lehmer-Pocklington, 78  
   de Lucas-Lehmer, 86  
   de Miller-Rabin, 68  
   de Pépin, 80  
   de Pocklington, 77  
   de Solovay-Strassen, 126  
 croissance polynomiale, 131  
 cyclique  
   code, 185, 246  
   groupe, 71  
 cyclotomique  
   classe, 251  
   corps, 216  
   extension, 224  
   polynôme, 213  
 déchiffrement, 63  
 décodage, 149, 238  
   du code CIRC, 275  
   total, 202  
 décomposition  
   d'un polynôme en produit, 44  
   suite, 78  
 déduit (réseau - d'un code), 293  
 degré  
   d'un polynôme, 41  
   d'un élément algébrique, 208  
 démultiplication d'un code sur  $\mathbf{F}_{2^s}$ , 239  
 détection d'une erreur, 150, 238  
 diagonale (construction), 155  
 différence symétrique, 302

- Diffie, 63  
dimension d'un code linéaire, 151, 237  
directe (somme), 155  
distance  
  assignée d'un code BCH, 261  
  de Bose d'un code BCH, 262  
  de Hamming, 150, 237  
  minimale, 150, 237  
diviseur de zéro, 302  
division euclidienne, 15  
  des polynômes, 42  
domaine des codes, 195  
dominant (coefficient), 41  
droite  
  dans un plan projectif abstrait, 180  
  projective, 178, 179  
  sur  $\mathbf{F}_p$ , 283  
dual (code), 157  
dualité des plans projectifs, 180
- effacement, 150  
EFM (modulation), 272  
égyptien (algorithme), 25  
élément  
  inversible, 302  
  nul, 303  
  primitif, 208  
  unité, 303  
engendré  
  code cyclique engendré, 246  
  sous-groupe engendré, 58  
entier  
  anneau des entiers, 29  
  réseau, 293  
  sans facteur multiple, 62  
entrée dans la période, 45  
entrelacement, 273  
entropie, 197  
énumération d'un ensemble fini, 168  
équation-clé, 267, 269  
équivalent (code), 156  
erreur  
  de transmission, 148  
  probabilité, 149, 203  
espace projectif, 178  
étendu  
  algorithme d'Euclide, 38  
  pour les polynômes, 43  
code  
  BCH, 264  
  binaire de Hamming, 164  
  QR, 282
- Euclide (algorithme), 33  
  binaire, 36  
  étendu, 38  
  étendu pour les polynômes, 43  
  pour les polynômes, 43  
euclidienne (division), 15  
  des polynômes, 42
- Euler  
  formule, 113  
  indicateur, 55  
  pseudoprime, 128  
  théorème, 62  
exposant d'un groupe commutatif, 72  
extension  
  cyclotomique, 224  
  paire d'un code, 156, 241
- Fano (plan), 179  
Fermat  
  nombre, 80  
  témoin, 67  
  théorème, 59  
FFT, 103  
Fibonacci (suite), 23, 37  
fini (corps), 30, 60, 178, 207  
fixe (coût), 27  
Floyd (algorithme), 46  
fonction  
  booléenne, 173  
  caractéristique, 167, 173  
formule  
  d'Euler, 113  
  du binôme, 303  
Fourier  
  transformation, 101  
  transformation rapide, 103
- Gauss  
  lemme, 215  
  somme, 120  
générateur  
  d'un code cyclique, 246  
  d'un groupe cyclique, 71  
  minimal d'un code cyclique, 248  
génératrice (matrice), 159  
Germain (nombre de Sophie Germain), 82  
Gilbert-Varshamov  
  conditions, 193  
  courbe, 198  
GIMPS, 82

- Golay  
 code binaire  $G_{23}$ , 287  
 code binaire  $G_{24}$ , 287  
 code ternaire  $G_{11}$ , 280
- Griesmer (majoration), 191
- groupe  
 affine, 186  
 cyclique, 71  
 de Mathieu  
 $M_{11}, M_{12}, M_{22}, M_{23}$ , 291  
 $M_{24}$ , 290  
 linéaire, 186  
 multiplement transitif, 291  
 multiplicatif d'un anneau, 303  
 transitif, 185
- Hamming  
 boule, 152  
 code, 162  
 $H_3$ , 154  
 $\bar{H}_3$ , 157  
 binaire étendu, 164  
 sur  $\mathbf{F}_q$ , 238  
 distance, 150, 237  
 majoration, 193
- Hellman, 63
- Hocquenghem (code BCH), 260
- homomorphisme d'anneaux, 303
- hypercube, 170
- hyperplan  
 affine, 171  
 vectoriel, 171
- hypothèse de Riemann généralisée, 75
- idempotent d'un code cyclique, 254
- identité de Bézout, 39  
 pour les polynômes, 44
- impair  
 code, 156  
 mot, 156
- indéterminisme, 18
- indicateur  
 d'Euler, 55  
 de Carmichael, 90
- information (taux), 148
- intègre (anneau), 301
- invariant d'un algorithme, 16
- inversible (élément), 302
- irréductible (polynôme), 44
- Jacobi (symbole), 123
- Jiuzhang suanshu, 51
- Lagrange (théorème), 58
- Leech (réseau), 296
- Legendre  
 symbole, 113  
 théorème, 61
- Lehmer (critère), 77, 78, 86
- lemme de Gauss, 215
- Leonelli (table de), 227
- linéaire  
 code, 237  
 code linéaire binaire, 151  
 groupe, 186  
 localisation (polynôme), 266  
 logarithme, 303  
 de Zech, 227  
 loi de réciprocité quadratique, 123  
 longueur d'un code, 148, 237
- Lucas  
 critère, 86  
 suite, 83
- majoration, 190  
 de Griesmer, 191  
 de Hamming, 193  
 de Plotkin, 192  
 de Singleton, 162, 191, 242
- Mathieu  
 groupe  $M_{24}$ , 290  
 groupes  $M_{11}, M_{12}, M_{22}, M_{23}$ , 291
- matrice  
 de parité, 160  
 génératrice, 159  
 vérificatrice, 159
- mémoire (canal sans -), 149
- Mersenne (nombre), 81
- Miller  
 critère de Miller-Rabin, 68  
 témoin, 68  
 test, 128
- minimal  
 distance minimale, 150, 237  
 générateur d'un code cyclique, 248  
 polynôme, 208
- mot  
 impair, 156  
 nul, 147  
 pair, 156  
 plein, 147

- multiple
  - entier sans facteur multiple, 62
  - polynôme sans facteur multiple, 232
- multiplicatif
  - fonction multiplicative, 56
  - groupe, 303
- multiplication
  - égyptienne, 25
  - méthode de Pollard, 108
  - méthode de Schönhage-Strassen, 109
- NIR (numéro), 142
- niveau
  - d'un corps cyclotomique, 216
  - d'une extension cyclotomique, 224
- nombre
  - composé, 9
  - de Carmichael, 89
  - de Fermat, 80
  - de Mersenne, 81
  - de Sophie Germain, 82
  - premier, 9
- non-résidu, 112
- nul
  - anneau, 302
  - code, 152
  - élément, 303
  - mot, 147
- optimal (code), 189
- ordre
  - d'un élément d'un groupe, 58
  - d'un groupe fini, 58
  - d'un plan projectif, 180
- orthogonal
  - code, 157, 241
  - codes orthogonaux, 157
- pair
  - code, 156, 241
  - extension paire, 156, 168, 241
  - mot, 156
  - réseau, 293
  - sous-code, 156
- paramètres
  - d'un code binaire linéaire, 151
  - d'un code linéaire, 237
- parfait
  - code binaire, 153
  - code sur  $\mathbb{F}_q$ , 239
- PARI, 20
- parité
  - code, 152
  - d'un entier, 147
  - matrice, 160
- partie
  - code de parties, 168
  - de type  $\text{coNP}$ , 132
  - de type  $\mathcal{NP}$ , 132
  - de type  $\mathcal{P}$ , 131
  - de type  $\mathcal{RP}$ , 133
  - de type  $\mathcal{ZPP}$ , 134
  - récursive, 131
- Pépin (critère), 80
- période d'une suite, 45
- périodique (suite ultimement -), 45
- pgcd
  - de deux entiers, 33
  - de deux polynômes, 44
- plafond d'un nombre réel, 303
- plan
  - affine, 170
  - de Fano, 179
  - projectif, 180
- plancher d'un nombre réel, 303
- plein
  - code, 151, 242
  - mot, 147
- Plotkin (majoration), 192
- Pocklington (critère), 77, 78
- poids d'un mot, 148, 237
- point d'accumulation, 195
- Pollard
  - algorithme de factorisation, 48
  - méthode de multiplication, 108
- polynôme
  - anneau des polynômes, 41
  - cyclotomique, 213
  - de localisation, 266
  - générateur d'un code cyclique, 248
  - minimal, 208
  - primitif, 219
  - réciproque, 249
  - sans facteur multiple, 232
  - syndrome, 266
- polynomiale (croissance), 131
- premier
  - corps, 209
  - nombre, 9
  - polynômes premiers entre eux, 44
  - sous-corps, 209

- primalité
  - certificat, 79
  - témoin, 66
- primitif
  - polynôme, 219
  - racine primitive, 73
  - élément, 208
- principale (racine), 99, 106
- probabilité
  - d'erreur d'un code, 203
  - d'une erreur, 149
- produit
  - de convolution, 101
  - scalaire, 157
- programme d'Erlangen, 187
- projectif
  - droite projective, 178, 179
  - droite projective sur  $\mathbb{F}_p$ , 283
  - espace, 178
  - plan projectif abstrait, 180
- projection d'un code, 190
- projété (code), 190
  
- quadratique
  - réciprocité, 122, 123
  - résidu, 112
- quotient (anneau), 30, 84, 302
  
- Rabin
  - critère de Miller-Rabin, 68
  - test, 128
  - théorème, 69
- raccourci (code), 190
- raccourcissement d'un code, 190, 243
- racine
  - d'un code cyclique, 252
  - de l'unité, 73
  - primitive, 73
  - principale, 99
  - principale modulo N, 106
- rapide
  - calcul des puissances, 23
  - transformation de Fourier, 103
- réciprocité quadratique, 122, 123
- réciproque (polynôme), 249
- récurrente (suite), 45
- récursion terminale, 21
- réursive (partie), 131
- réécriture, 15
- Reed-Muller (code), 174
- Reed-Salomon (code), 245, 259
  
- règle de réécriture, 15
- rendement d'un code, 148
- renumérotation, 156
- répétition (code), 151
- repunit, 9
- réseau, 292
  - associé, 293
  - de Leech, 296
  - déduit d'un code, 293
  - $E_8$ , 296
  - entier, 293
  - pair, 293
  - unimodulaire, 293
- résidu
  - codes de résidus quadratiques, 280
  - quadratique, 112
- Riemann (hypothèse), 75
- Rivest, 64
- RSA (méthode), 64
  
- scalaire (produit), 157
- Schönhage (méthode de multiplication), 109
- Sendrier, 271
- Shamir, 64
- Shanks (algorithme), 115
- Shannon (théorème), 203
- simplexe (code), 171
- Singleton (majoration), 162, 191, 242
- Solovay
  - critère, 126
  - témoin, 127
  - test, 128
- somme
  - de Gauss, 120
  - directe de codes, 155
- sous-anneau, 303
- sous-corps, 303
- sous-espace affine, 176, 178
- sous-groupe engendré, 58
- Steiner (système), 181
- Strassen
  - critère, 126
  - méthode de multiplication, 109
  - test, 128
- strong pseudoprime, 128
- suite
  - de décomposition, 78
  - de Fibonacci, 23, 37
  - de Lucas, 83
  - récurrente, 45
- Sun Tzu Suan Ching, 51

- support
  - d'un mot, 167
  - d'une fonction, 173
- symbole
  - de Jacobi, 123
  - de Legendre, 113
- symétrique
  - canal binaire symétrique, 149
  - différence symétrique, 302
- syndrome, 161, 266
  - polynôme, 266
- systématique (codage), 260
- système de Steiner, 181
- table
  - de vérité, 173
  - de Zech, 227
- taille d'un entier, 27
- taux d'information d'un code, 148
- témoin
  - de Fermat, 67
  - de Miller, 68
  - de primalité, 66
  - de Solovay, 127
- terminaison d'un algorithme, 16
- terminale (récursion), 21
- ternaire (code), 237
  - de Golay, 280
- test
  - de Miller-Rabin, 128
  - de Solovay-Strassen, 128
- théorème
  - AKS, 135, 228
  - d'Euler, 62
  - de Fermat, 59
  - de Lagrange, 58
  - de Legendre, 61
  - de Rabin, 69
  - de réciprocité quadratique, 122
  - de Shannon, 203
  - de Tietäväinen-Van Lint, 289
  - de Wedderburn, 207, 214
  - de Wilson, 60
- thèse de Church, 131
- Tietäväinen-Van Lint (théorème), 289
- total (décodage), 202
- trames, 271
- transformation
  - affine, 186
  - de Fourier, 101
  - de Fourier rapide, 103
- transitif
  - groupe de permutations, 185
  - groupe multiplement transitif, 291
- transmission
  - canal, 148
  - erreur, 148
- trivial (code), 152, 242
- turbo-codes, 201
- témoin de non-primalité, 11
- unimodulaire (réseau), 293
- unitaire (polynôme), 41
- unité (élément), 303
- vectoriel (hyperplan), 171
- vérificatrice (matrice), 159
- vérité (table), 173
- Wedderburn (théorème), 207, 214
- Wilson (théorème), 60
- Zech (table de), 227
- zéro (diviseur), 302

IMPRIMÉ EN GRANDE-BRETAGNE  
PAR CAMBRIDGE UNIVERSITY PRESS  
DÉPÔT LÉGAL DÉCEMBRE 2008

Disciplines plus que bimillénaires, l'algèbre et l'arithmétique ont connu récemment des applications aussi spectaculaires qu'inattendues. Comment décomposer un nombre en facteurs premiers, comment reconnaître si un nombre est premier : ces questions, revivifiées par l'existence des moyens