A short introduction to finite fields and their arithmetic

Pierre Karpman

April 26, 2022

Contents

1 Prime fields (and more)

This section introduces the simplest examples of finite fields, *i.e.* *prime fields*, and some important definitions and first results about finite fields.

1.1 Fermat-based approach

We start by recalling Fermat's "little" theorem:

Theorem 1 (Fermat's little theorem (FLT))**.** *Let p be a prime, $a \in [\![1, p-1]\!]$, then $a^{p-1} \equiv 1 \mod p$.*

Proof. Following the strategy of Fermat, the proof is admitted. See however Corollary 11. □

This immediately leads to the fundamental result:

Theorem 2. *The ring $\mathbb{Z}/n\mathbb{Z}$ is a field iff. n is prime.*

Proof. If $n > 1$ is not prime, then $\exists\, a, b \in [\![2, n-1]\!]$ s.t. $n = ab$ so a is a zero divisor modulo n and hence not invertible.[*]

If n is prime, then by Theorem 1 every $a \in [\![1, n-1]\!]$ is invertible modulo n. □

[*] A ring with such non-trivial divisors of zero is called *non-integral*.

Note that the above proof is essentially algorithmic in nature in that it provides a way to compute inverses modulo p and then to divide in $\mathbb{Z}/p\mathbb{Z}$, which is the only one of the basic operations $(+,-,\times,\div)$ that is non trivial. In all of the following we will use the notation $\mathbb{F}_p$ to denote the *prime field* $\mathbb{Z}/p\mathbb{Z}$, and $[n]x$ where n is an integer and x the element of an additive group to denote the *scalar multiplication* of x by n, *i.e.* the sum of n copies of x together, or $\sum_{i=1}^{n} x$.

We already introduce the following definition, which will however mostly be useful in the next section.

Definition 3 (Characteristic)**.** The *characteristic* of a ring $\mathbb{A}$, denoted $\mathrm{char}(\mathbb{A})$, is the least positive integer n s.t. $\forall\, x \in \mathbb{A}$, $[n]x = 0$, if it exists, and zero otherwise.

An equivalent definition for fields is to only consider the multiplicative identity 1 instead of an arbitrary x in the above statement; indeed since every non-zero $x \in \mathbb{A}$ is by definition equal to $x \times 1$, then by distributivity one has that $[n]x = [n](x \times 1) = x \times ([n]1)$ which is zero iff. $[n]1 = 0$.

Exercise 1.

1. Why does the above equivalent definition fails for rings in general (Hint: Consider for instance a non-integral ring.)
2. Show that $\mathrm{char}(\mathbb{F}_p) = p$.

Exercise 2 (Prime fields arithmetic)**.** Let p be a prime. We write R the cost in elementary bit operations of an operation `RED` that "positively" reduces $x \in [\![-p^2, p^2]\!]$ modulo p (by that we mean that it computes the positive remainder of the division of x by p).

1. How many bits are necessary and sufficient to represent all the elements of $\mathbb{F}_p$?
2. Give an algorithm that computes the sum or the subtraction of two elements of $\mathbb{F}_p$. What is its cost in elementary bit operations?
3. Same question for the product, assuming that schoolbook integer multiplication is used.
4. Same question for computing the inverse of an element using the FLT. Give the cost first in the *algebraic model*, that is in function of the number of products in $\mathbb{F}_p$, then in elementary bit operations.

In the last question, one may rely on a "fast exponentiation" algorithm, for which we give an example in pseudo-code in Fig. 1.

Remark 4. The above exercise considers the (rather usual) case where the class modulo p of an element x is represented by a non-negative integer in $[\![0, p-1]\!]$. A possible alternative for $p > 2$ is to instead consider a *balanced* representation in $[\![-(p-1)/2, (p-1)/2]\!]$, which slightly decreases the size necessary to store a not-yet-reduced product. Consider indeed x, $y \in [\![0, p-1]\!]$, then xy is at most $(p-1)^2$; however for x, $y \in [\![-(p-1)/2, (p-1)/2]\!]$, $|xy|$ is at most $((p-1)/2)^2 = (p-1)^2/4$ and, adding a bit for the sign, one expects to save one bit over the non-balanced representation. This saving may even be increased to two bits when the physical representation already includes a sign bit even when it would not be needed (this is for instance the case for IEEE754 floats).

```
/* Input:
   - x, an element of a group G (with multiplicative notation)
   - b, an integer > 1
   Output:
   - x**b in G
*/
fastexp(x, b)
{
    res = 1;
    acc = x;
    while(b > 0)
    {
        if (b % 2 == 1)
            res = res * acc; // product in G
        acc = acc * acc;     // squaring in G
        b = b / 2;           // integer division
    }
    return res;
}
```

Figure 1: Fast exponentiation in a group.

1.2 Euclid-based approach

Given the importance of finite fields and of the ring $\mathbb{Z}/n\mathbb{Z}$ in general, we wish to reprove some of the above results using Euclid's extended algorithm as a basis instead of the FLT.

Theorem 5 (Extended Greatest Common Divisor (XGCD))**.** *Let $a, b \in \mathbb{Z}\backslash\{0\}$, then $\exists\, u, v, d \in \mathbb{Z}$ s.t. $ua + vb = d = \gcd(a, b)$, where d is the* greatest common divisor *of a and b,* i.e. *the largest positive integer that divides both a and b.*

Proof. We first show that if u, v, d exist s.t. $ua + vb = d$ and d divides both a and b, then d is indeed $\gcd(a, b)$. Assume instead that $d' > d$ divides both $a = a'd'$ and $b = b'd'$, then dividing both sides by d' we get $ua' + vb' = d/d'$ where the left-hand side is integral and the right-hand side is not, which is a contradiction, so such a d' cannot exist.

It remains to show that such a triplet exists, which we do algorithmically thanks to Euclid's algorithm. We start by proving the "standard" non-extended algorithm that only computes d and address the computation of u and v next.

W.l.o.g. we assume that $a \geqslant b$ and use the fact that $d = \gcd(a, b) = \gcd(b, a \mod b)$, where $x \mod y$ denotes here the positive remainder of the division of x by y. Indeed one has that $a = a'd$ and $b = b'd$ for some a', b', hence letting $a = qb + r$ we have $r = d(a' - qb')$ so $d \,|\, r = a \mod b$; conversely if there were $d' > d$ that divided both r and b, then one would have $a = d'(qb'' + r'')$ for some b'', r'', so d' would also divide a which contradicts the fact that $d = \gcd(a, b)$. We then use this equality repeatedly to compute the sequence $r_0 := a, r_1 := b, r_2 := r_0 \mod r_1, \ldots, r_i := r_{i-2} \mod r_{i-1}, \ldots$. Since the r_i's are positive and decreasing there is an index $k+1$ s.t. $r_{k+1} = 0$, which is equivalent to the fact that $r_k \,|\, r_{k-1}$ which itself implies that $\gcd(r_0, r_1) = \gcd(r_{k-1}, r_k) = r_k$.

We now conclude by showing how to compute u and v. Let $\boldsymbol{T}_1 = \begin{pmatrix} 0 & 1 \\ 1 & -r_0 \div r_1 \end{pmatrix}$, where $x \div y$ denotes here the quotient in the division of x by y, then we have that $\boldsymbol{T}_1 \begin{pmatrix} r_0 & r_1 \end{pmatrix}^t = \begin{pmatrix} r_1 & r_2 \end{pmatrix}^t$. By defining $\boldsymbol{T}_2, \ldots, \boldsymbol{T}_k$ similarly and letting $\boldsymbol{M} = \boldsymbol{T}_k \cdots \boldsymbol{T}_1$ it follows that $\boldsymbol{M} \begin{pmatrix} r_0 & r_1 \end{pmatrix}^t = \begin{pmatrix} r_k & 0 \end{pmatrix}^t$ and the first row of $\boldsymbol{M}$ gives the desired relation. □

Remark 6. The above algorithm (and the notion of GCD) is not restricted to integers and may also for instance be applied to polynomials. There also exist many variants of Euclid's algorithm, some being asymptotically faster than the one presented above.

We may now reprove the second part of Theorem 2 by observing that one may recover an inverse of any $a \in [\![1, p-1]\!]$ modulo p from $ua + vp = \gcd(a, p) = 1$ as (the possibly reduced) u, since $ua \equiv 1 \mod p$. It is also worth noting that this approach is not restricted to finite fields as it allows to compute the inverse of any element of (say) an arbitrary ring $\mathbb{Z}/n\mathbb{Z}$ as long as it is coprime with n, those elements being indeed exactly the invertible ones.

1.3 Structure of the multiplicative group of a finite field

While it is clear that the additive group of $\mathbb{F}_p$ is cyclic, this is quite less so for its multiplicative group. It is in fact the case that the multiplicative group of any finite field —and prime fields in particular— is cyclic. We state this in the general case of a finite field $\mathbb{F}_q$ with q (not necessarily a prime number) elements as:

Theorem 7. *The multiplicative group $\mathbb{F}_q^\times$ of $\mathbb{F}_q$ is cyclic. Any of its generators is called a* primitive *element of $\mathbb{F}_q$.*

Before (partially) proving this theorem, we recall the following:

Definition 8 (Order of an element, order of a group)**.** Let $\alpha \in \mathbb{G}$ be an element of a finite group (written multiplicatively). The *order* of α, written $\mathrm{ord}(\alpha)$, is the least positive integer t s.t. $\alpha^t = 1$, where 1 denotes the identity in $\mathbb{G}$.

The order of a finite group $\mathbb{G}$, written $\mathrm{ord}(\mathbb{G})$, is the number of elements (or cardinality) of $\mathbb{G}$.

From the above definitions, if we write $\langle\alpha\rangle$ the subgroup of $\mathbb{G}$ generated by α, we have $\mathrm{ord}(\langle\alpha\rangle) = \mathrm{ord}(\alpha)$. We also have the following classical theorem, whose proof we will admit:

Theorem 9. *Let $\alpha \in \mathbb{G}$ be an element of a finite group, then $\mathrm{ord}(\langle\alpha\rangle) \mid \mathrm{ord}(\mathbb{G})$.*

An immediate application of Theorem 9 to $\mathbb{F}_q^\times$ leads to the fact that the ("multiplicative") order of any non-zero element of $\mathbb{F}_q$ divides $q-1$.

Let now $\alpha \in \mathbb{F}_q^\times$ be an element of order t; there are t distinct elements in the subgroup it generates, *viz.* $\mathcal{A} := \{1, \alpha, \ldots, \alpha^{t-1}\}$. An important fact is then that $\mathcal{A}$ is *exactly* the set of the roots in $\mathbb{F}_q$ of the polynomial $P := X^t - 1$; indeed all the elements of $\mathcal{A}$ are indeed roots of P and by the following Proposition 10 there are at most t of them in $\mathbb{F}_q$.

Proposition 10. *Let $P \in \mathbb{F}_q[X]$ be a polynomial of degree d, then it has at most d roots in $\mathbb{F}_q$.*

Proof. Left as an exercise. HINT: Use Euclidean division and an induction on the degree. □

Note that Proposition 10 is not true over rings in general; for instance, $2X \in \mathbb{Z}/6\mathbb{Z}[X]$ has more than one root, *viz.* 0 and 3.

Corollary 11. *Let $\alpha \in \mathbb{F}_q^\times$, then $\alpha^{q-1} = 1$. It follows that all the elements of $\mathbb{F}_q^\times$ are exactly the roots of $X^{q-1} - 1$, and all the elements of $\mathbb{F}_q$ are exactly the roots of $X^q - X$.*

Proof. The first property is a consequence of Theorem 9, and the second of Proposition 10. □

Note that this corollary may be seen as a generalisation of Theorem 1 to not-necessarily-prime finite fields, and thusly provides a general inversion algorithm in finite fields.

It is enlightening to further study the structure of the elements of $\mathcal{A}$ (or equivalently in our case, of the roots of $X^t - 1$). While $\operatorname{ord}(\alpha) = t$ by definition, it is clear that this is not the case for all the other powers of α since for instance $\alpha^0 = \alpha^t = 1$ has order 1. More generally, if s is not coprime with t, *i.e.* if $\gcd(s,t) =: d > 1$, then writing $s = s'd$, $t = t'd$, $(\alpha^s)^{t'} = \alpha^{s'dt} = (\alpha^{s'})^t = 1$; we have thus proven $\operatorname{ord}(\alpha^s) = t \Rightarrow \gcd(s,t) = 1$.

This latter condition is in fact also sufficient. From $\gcd(s,t) = 1$ and Theorem 5, $\exists\, u, v \in \mathbb{Z}$ s.t. $us + vt = 1$. Suppose now that $\exists\, u' \in [\![1, t-1]\!]$ s.t. $(\alpha^s)^{u'} = 1$, then we must have $u's = kt$ for some positive k, and so $u' = u'(us + vt) = uu's + u'vt = ukt + u'vt = t(uk + u'v)$, so t must divide u', which is a contradiction.

We have in fact proven:

Proposition 12. *Let* $\alpha \in \mathbb{G}$, $\operatorname{ord}(\alpha) = t$, *then for* $s \in [\![1, t-1]\!]$, $\operatorname{ord}(\alpha^s) = t$ *iff.* $\gcd(s,t) = 1$.

Combining Propositions 10 and 12 leads to the following weaker form of Theorem 7:

Proposition 13. *Let t be a positive integer dividing $q-1$, then either there are no elements of order t in $\mathbb{F}_q^\times$ or there are $\phi(t)$ of them, where $\phi : t \mapsto \#\{1 \leqslant s \leqslant t, \gcd(s,t) = 1\}$ is the* totient *function.*

Proposition 13 illustrates a situation that often arises in the study of finite fields: because of their strong algebraic structure, “asking” that a finite field includes an element of order t “automatically” adds $t-1$ other roots of $X^t - 1$, of which $\phi(t) - 1$ are also of order t.

To conclude the proof of Theorem 7, we will use the following important property of the totient:

Proposition 14. *Let n be a positive integer, then* $\sum_{t\,|\,n} \phi(t) = n$.

Proof. We will use a counting argument applied to the cyclic group of order n. Let g_n be a generator of this group, then for all positive t that divide n, $g_t := g_n^{n \div t}$ generates a subgroup of order t which by Proposition 12 has $\phi(t)$ generators. Conversely if h_t is an element of order t and since the group is cyclic, $h_t = g_n^x$ for some $x \in [\![1, n-1]\!]$ satisfying $xt = kn$, $k > 0$. It follows that $xt = kt(n \div t)$ and then $x = k(n \div t)$, so $h_t \in \langle g_t \rangle$; consequently there are exactly $\phi(t)$ elements of order t in the full group. We conclude by noting that all elements of $\langle g_n \rangle$ generate a subgroup, whose order must divide n by Theorem 9. ☐

Since the elements of $\mathbb{F}_q^\times$ have an order that must divide $q-1$, a counting argument applied to Propositions 13 and 14 shows that there must exist elements of order $q-1$, and hence $\mathbb{F}_q^\times$ is cyclic as needed to be shown.

Remark 15. Although we have (mostly) proven with Theorem 7 that the multiplicative group of a finite field is cyclic, we have not given an efficient algorithm to find a generator of this group.

Remark 16. The multiplicative group of a finite fields possesses much more structure than “just” being cyclic. It is for instance much easier to compute discrete logarithms therein than in “black-box” cyclic groups.

2 Extensions

In the previous section we have seen how to build and compute in finite fields $\mathbb{F}_p$ with a prime number of elements. We now wish to consider *extensions* of such fields whose number of elements is not necessarily prime any more.

2.1 Cardinality, uniqueness

Before building extension fields, we will state (and partially prove) two important results regarding finite fields in general.

Theorem 17 (Cardinality)**.** *Every finite-field $\mathbb{F}$ has cardinality $q = p^k$ where p is prime and k is a non-zero positive integer.*

Proof. We first show that $\mathrm{char}(\mathbb{F})$ is prime. Since q is finite $\exists\, n \in \mathbb{N}\backslash\{0\}$ s.t. $[n]1 = 0$, so $c := \mathrm{char}(\mathbb{F}) > 0$. Now assume by contradiction that $c = ab$ for some $a, b \in [\![2, c-1]\!]$. We then have $[ab]1 = 0 = [a]([b]1)$; let $\beta = [b]1 \neq 0$, since we are in a field this latter element must have a multiplicative inverse β^{-1} and it follows that $([a]\beta)\beta^{-1} = [a](\beta\beta^{-1}) = [a]1 = 0$ with $a < c$, which is a contradiction. So c must be a prime p.

To prove the remainder of the statement we will show (the stronger property) that $\mathbb{F}$ must have the structure of a finite-dimensional $\mathbb{F}_p$-vector space, which will allow us to conclude.

First let us remark that equipped with the same laws as $\mathbb{F}$, $\mathcal{F}_1 := \{1, [2]1, \ldots, [p-1]1\} \cong \mathbb{F}_p$: indeed it is clear that the $[i]1$, $1 \leqslant i < p$ are distinct, and that $[a]1 \times [b]1 = [ab]1 = [ab \bmod p]1$ follows from distributivity and the fact that $\mathrm{char}(\mathbb{F}) = p$. We thereafter identify the *subfield* $\mathcal{F}_1$ of $\mathbb{F}$ with $\mathbb{F}_p$.

Now either $\mathbb{F}_p = \mathbb{F}$ and we are done, or there exists $\beta_2 \in \mathbb{F}$ that is $\mathbb{F}_p$-linearly-independent from $\beta_1 := 1$, *i.e.* that cannot be written as $\lambda_1\beta_1$ for any $\lambda_1 \in \mathbb{F}_p$. In that latter case we consider the set $\mathcal{F}_2$ of linear combinations $\{\lambda_1\beta_1 + \lambda_2\beta_2\}$ with $\lambda_1, \lambda_2 \in \mathbb{F}_p$, which is of size p^2;[†] now either $\mathcal{F}_2 = \mathbb{F}$ and we are done, or we again pick $\beta_3 \in \mathbb{F}$ not of the form $\lambda_1\beta_1 + \lambda_2\beta_2$ to build $\mathcal{F}_3$ of size p^3. It is clear that we can iterate this process until $\mathcal{F}_k = \mathbb{F}$, which must be true for some k since q is finite. At that point one has that every element of $\mathbb{F}$ can be written as the $\mathbb{F}_p$-linear combination of $\{\beta_1, \ldots, \beta_k\}$, that two elements can be added "component-wise" in this representation, and that one can rescale an element by an arbitrary scalar in $\lambda \in \mathbb{F}_p$ by multiplying every component; we leave the proof that $\mathbb{F}$ satisfies the remaining axioms of an $\mathbb{F}_p$-vector space to the reader. $\square$

Remark 18. From the preceding proof, when $q = p^k$ with $k > 1$, the additive group of $\mathbb{F}_q$ is *not* cyclic. This is to be contrasted with the cyclicity of the additive group of prime fields and the one of the multiplicative group of any finite field.

Remark 19. The set $\{\beta_1, \ldots, \beta_k\}$ as above is said to form an $\mathbb{F}_p$*-basis* of $\mathbb{F}$, or *a basis of* $\mathbb{F}$ *over* $\mathbb{F}_p$. Note that in general such a basis is not unique.

We only state and do not prove the following fundamental result.

Theorem 20 (Unicity)**.** *Let $\mathbb{F}$ be a finite field with q elements, then every other field with q elements (if any) is isomorphic to $\mathbb{F}$.*

From now on we will denote the unique (up to isomorphism) field with q elements by $\mathbb{F}_q$.

Remark 21. Let $\mathbb{F}$, $\mathbb{F}'$ be two finite fields with q elements, although from Theorem 20 one has that $\mathbb{F} \cong \mathbb{F}'$, $\mathbb{F}$ and $\mathbb{F}'$ may still have different *representations* for their elements and their arithmetic. From a practical point of view this means that even if $\mathbb{F}$ and $\mathbb{F}'$ fundamentally possess the same structure, it might be easier to implement operations in $\mathbb{F}$ than in $\mathbb{F}'$.

[†] Suppose that there were $\lambda_1, \lambda_1' \neq \lambda_1, \lambda_2, \lambda_2' \neq \lambda_2 \in \mathbb{F}_p$ s.t. $\lambda_1\beta_1 + \lambda_2\beta_2 = \lambda_1'\beta_1 + \lambda_2\beta_2$. Then $(\lambda_1 - \lambda_1')\beta_1 = (\lambda_2' - \lambda_2)\beta_2$ (with $\beta_1 \neq 0 \neq \beta_2$). By definition of β_2, $\forall \lambda_1 \in \mathbb{F}_p$, $\lambda_1\beta_1 \neq \beta_2$. Now if there were $\lambda_1, \lambda_2 \in \mathbb{F}_p^\times$ with $\lambda_1\beta_1 = \lambda_2\beta_2$, one would have $\lambda_1\lambda_2^{-1}\beta_1 = \beta_2$ with $\lambda_1\lambda_2^{-1} \in \mathbb{F}_p$, leading to a contradiction.

2.2 Building extensions from polynomials

The basic idea used to build *extensions* of a prime field such as $\mathbb{F}_2$ is to observe that one may easily define polynomials over any field; for instance polynomials in one indeterminate X form the ring $\mathbb{F}_2[X]$.

Exercise 3. Let $P = X^2 + X + 1, Q = X^4 + X + 1 \in \mathbb{F}_2[X]$, compute $P + P$, $P + Q$, PQ.

There are two obstacles that prevent $\mathbb{F}_2[X]$ from being a finite field: it is infinite, and not all of its non-zero elements are invertible. One may remark that these are also what prevents $\mathbb{Z}$ from being a finite field and thus may try to adapt the strategy yielding the finite fields $\mathbb{Z}/p\mathbb{Z}$ to polynomials.

Fact 22. *Let $P \in \mathbb{F}_q[X]$ be a polynomial of degree $d \geqslant 1$, then $\mathbb{F}_q[X]/\langle P\rangle$ is a finite ring of cardinality q^d.*

Exercise 4. Let $P = X^2 + X + 1, Q = X^4 \in \mathbb{F}_2[X]$, compute the Euclidean division of Q by P. Let $R = X, S = X + 1 \in \mathbb{F}_2[X]$, compute (the class of) RS in $\mathbb{F}_2[X]/\langle P\rangle$, that is, the unique polynomial of degree less than two equal to the remainder of the division of RS by P. Same question where the product is computed in $\mathbb{F}_2[X]/\langle X^2 + X\rangle$; is this latter ring a field?

The last exercise shows that similarly to the fact that $\mathbb{Z}/n\mathbb{Z}$ is a field iff. n is prime, some constraint on P is also necessary for $\mathbb{F}_q[X]/\langle P\rangle$ to possibly be one. This constraint is in fact the same as in the integral case, that is we require P not to have any non-trivial factorisation over $\mathbb{F}_q$.

Definition 23 (Irreducible polynomial)**.** Let P be a polynomial of degree $d \geqslant 1$ with coefficients in some field $\mathbb{K}$, we say that P is *irreducible over* $\mathbb{K}' \supseteq \mathbb{K}$ iff. it cannot be written as the product of two non-constant polynomials with coefficients in $\mathbb{K}'$.

Exercise 5. Is $X^2 + 1$ irreducible over $\mathbb{R}$? What about over $\mathbb{C}$? And over $\mathbb{F}_2$?

We now state the "equivalent" of Theorem 2 for polynomials over finite fields.

Theorem 24. *Let $P \in \mathbb{F}_q[X]$ be a monic polynomial of degree $d \geqslant 1$, then $\mathbb{F}_q[X]/\langle P\rangle$ is a finite field with q^d elements iff. P is irreducible over $\mathbb{F}_q$.*

Proof. If P is not irreducible, then its factors divide zero in $\mathbb{F}_q[X]/\langle P\rangle$ and thus do not admit any inverse in this ring. In the converse case, all of the $q^d - 1$ non-zero polynomials P' of $\mathbb{F}_q[X]$ of degree less than d are coprime with P and thus using Theorem 5 for polynomials one may compute P'^{-1} s.t. $P'P'^{-1} + QP = 1$. □

In the above, we say that a field $\mathbb{F}_{q^d} := \mathbb{F}_q[X]/\langle P\rangle$ is an *extension* of $\mathbb{F}_q$ *of degree* d. One may remark that by construction $\mathbb{F}_{q^d}$ has the structure of a d-dimensional $\mathbb{F}_q$-vector space and that its characteristic is also equal to the one of $\mathbb{F}_q$. More generally given two finite fields $\mathbb{F}$ and $\mathbb{F}'$, we may write $\mathbb{F}'/\mathbb{F}$ to denote the fact that $\mathbb{F}'$ is an extension of $\mathbb{F}$ — conversely this means that $\mathbb{F}$ is a *subfield* of $\mathbb{F}$ — and $[\mathbb{F}' : \mathbb{F}]$ to denote the degree of the extension.

We admit the following result.

Theorem 25. *Let $\mathbb{F}$ be an arbitrary finite field, then for every $d \geqslant 1$ there exists at least one irreducible polynomial of degree d in $\mathbb{F}[X]$.*

To this we add that it is computationally "efficient" to test if a given polynomial is irreducible, and that irreducible polynomials form a dense subset of all the polynomials. It then follows that there exists an efficient randomised procedure that returns an irreducible polynomial of arbitrary degree over any finite field.

We also have the immediate corollary.

Corollary 26. *There is a finite field with q elements iff. $q = p^k$, where p is prime and k is a non-zero positive integer.*

Proof. One direction is given by Theorem 17, the other by Theorems 24 and 25. □

Finally since it is efficient to find an irreducible polynomial of arbitrary degree, it is similarly efficient to compute a *representation* of an arbitrary finite field. Since there are also in general several irreducible polynomials of the same degree, one may define several fields with the same number of elements as, say, $\mathbb{F}_q[X]/\langle P\rangle$, $\mathbb{F}_q[X]/\langle Q\rangle$ with distinct P and Q both of degree d and irreducible over $\mathbb{F}_q$. However by Theorem 20 those two fields would in fact be isomorphic (written $\mathbb{F}_q[X]/\langle P\rangle \cong \mathbb{F}_q[X]/\langle Q\rangle$).

We now give a few examples in the binary case.

Example 27. One may check that $P := X^2 + X + 1$ is irreducible over $\mathbb{F}_2$. Consequently $\mathbb{F}_2[X]/\langle P\rangle$ is a degree-2 extension over $\mathbb{F}_2$, *i.e.* a finite field with 4 elements. Those elements may be represented as polynomials as 0, 1 (the elements of the subfield $\mathbb{F}_2$), X and $X + 1$. The addition between the elements is the addition for polynomials and the multiplication is done modulo P, *i.e.* $X \times X = X + 1$, $X(X + 1) = 1$, $(X + 1)(X + 1) = X$.

Example 28. One may check that $P := X^4 + X + 1$ and $Q := X^4 + X^3 + 1$ are both irreducible over $\mathbb{F}_2$, so one may represent the field $\mathbb{F}_{2^4}$ both as $\mathbb{F}_2[X]/\langle P\rangle$ or $\mathbb{F}_2[X]/\langle Q\rangle$.

Example 29. Let P be as in Example 27, in the ring $(\mathbb{F}_2[X]/\langle P\rangle)[Y]$, the polynomial $Y^2 + Y + 1$ is *not* irreducible over $\mathbb{F}_2[X]/\langle P\rangle$ since it factors as $(Y + X)(Y + X + 1) = Y^2 + XY + Y + XY + X^2 + X = Y^2 + Y + X^2 + X = Y^2 + Y + 1$. One may however check that $Q := Y^2 + XY + 1$ is irreducible over the same field, so one may build a degree-2 field extension $(\mathbb{F}_2[X]/\langle P\rangle)[Y]/\langle Q\rangle$. Taken together, the extensions over $\mathbb{F}_2$ and $\mathbb{F}_2[X]/\langle P\rangle$ form an *extension tower* of two extensions of degree 2, which yields an extension of degree $4 = 2 \times 2$ over the base field $\mathbb{F}_2$, *i.e.* a representation of $\mathbb{F}_{2^4}$; in other words, $\mathbb{F}_{2^4} \cong \mathbb{F}_2[X, Y]/\langle P, Q\rangle$.

Example 30. Let P, Q be as in Example 28; we wish to explicit an isomorphism φ between $\mathbb{F} := \mathbb{F}_2[X]/\langle P\rangle$ and $\mathbb{F}' := \mathbb{F}_2[Y]/\langle Q\rangle$[‡] whose existence is guaranteed by Theorem 20. That is we want to determine a bijective map $\varphi : \mathbb{F} \to \mathbb{F}'$ s.t. $\forall\, a, b \in \mathbb{F}$ one has $\varphi(a + b) = \varphi(a) + \varphi(b)$ and $\varphi(ab) = \varphi(a)\,\varphi(b)$.

From the definition, it is clear that we need to have $\varphi(0) = 0$ and $\varphi(1) = 1$, where the right-hand side of both equalities "live in $\mathbb{F}'$". To determine the image of φ for the remaining elements we may try to use the fact that the multiplicative group of a finite field is cyclic. Observing that $\langle X\rangle = \mathbb{F}^\times$ and $\langle Y\rangle = \mathbb{F}'^\times$, we could try to extend φ by taking $\alpha = X$, $\beta = Y$, and $\varphi(\alpha^i) = \beta^i$ for all i. For any non-zero a, b one gets $\varphi(ab) = \varphi(\alpha^i\alpha^j) = \varphi(\alpha^{i+j}) = \beta^{i+j} = \beta^i\beta^j = \varphi(a)\,\varphi(b)$ for some i, j. Defined thusly φ would indeed be a group homomorphism $\mathbb{F}^\times \to \mathbb{F}'^\times$, however it would *not* be one for the additive group; this is not surprising since here φ would essentially be the "identity" mapping, and $\mathbb{F}$ and $\mathbb{F}'$ *do* use a different representation: in particular one requires $\varphi(X^4 + X + 1) = \varphi(X^4) + \varphi(X) + \varphi(1) = 0$ and taking $\varphi(X) = Y$ yields $Y^4 + Y + 1 = Y^3 + Y \neq 0$. More generally, if we define the *minimal polynomial* $\min_{\mathbb{K}'}(a)$ over $\mathbb{K}'$ of an element $a \in \mathbb{K}$ to be the monic polynomial $\sum_{i=0}^{d} \pi_i Z^i \in \mathbb{K}'[Z]$ of least degree that annihilates a (i.e. s.t. $\sum_{i=0}^{d} \pi_i a^i = 0$) we have here the necessary condition that $\forall a, \min_{\mathbb{F}_2}(\varphi(a)) = \min_{\mathbb{F}_2}(a)$.

To show that adding this condition is also sufficient we first remark that if $\deg(\min(a)) = k$[§] then k is also the rank of $\langle a^i\rangle_{i\in\mathbb{N}}$ as an $\mathbb{F}_2$-vector space. In our particular case the dimension of $\mathbb{F}$ over $\mathbb{F}_2$ is 4, so we have that if $k = 4$ then every element of $\mathbb{F}$ can be written as an

[‡]We use two names for the indeterminate so as to make it simpler to determine in which representation an element is living.

[§]We drop the index $\mathbb{F}_2$ from "$\min_{\mathbb{F}_2}$" in the following for the sake of conciseness.

$\mathbb{F}_2$-linear combination of $1 = a^0$, a, a^2 and a^3. Suppose now that we have $\varphi(\alpha) = \beta$ with primitive α, β and where $\min(\alpha) = \min(\beta)$ a polynomial of degree 4, then $\langle \alpha^i \rangle_{0 \leqslant i < 4}$ (resp. $\langle \beta^i \rangle_{0 \leqslant i < 4}$) is a basis of $\mathbb{F}$ (resp. $\mathbb{F}'$) over $\mathbb{F}_2$. Moreover writing $\sum_{i=0}^{4} \pi_i Z^i$ for $\min(\alpha)$ one has that $\alpha^4 = \sum_{i=0}^{3} \pi_i \alpha^i$ and $\beta^4 = \sum_{i=0}^{3} \pi_i \beta^i$; $\alpha^5 = \sum_{i=0}^{3} \pi_i \alpha^{i+1} = \sum_{i=0}^{2} \pi_i \alpha^{i+1} + \pi_3 \sum_{i=0}^{3} \pi_i \alpha^i$ and $\beta^5 = \sum_{i=0}^{3} \pi_i \beta^{i+1} = \sum_{i=0}^{2} \pi_i \beta^{i+1} + \pi_3 \sum_{i=0}^{3} \pi_i \beta^i$; more generally if $a = \alpha^j = \sum_{i=0}^{3} a_i \alpha^i$ then $\varphi(a) = \beta^j = \sum_{i=0}^{3} a_i \beta^i$ and so φ is a homomorphism for the addition.

To conclude the construction of φ, it is now sufficient to find a suitable primitive element $\beta \in \mathbb{F}'$ whose minimal polynomial equals $\min(\alpha)$ for some primitive $\alpha \in \mathbb{F}$. In our example one can check that $\min(Y^7) = Z^4 + Z + 1$ and since $7 \nmid 15$ its multiplicative order is 15; we then finally define φ as $X \mapsto Y^7$, and all the other points are obtained by using the fact that φ is an isomorphism.

Remark 31. In the above we have used the fact that in an extension $\mathbb{F}_{q^d}$ of $\mathbb{F}_q$, an element with minimal polynomial (over $\mathbb{F}_q$) of maximal degree d allows to additively generate $\mathbb{F}_{q^d}$ as the linear combinations of $a^0, \ldots, a^{d-1}$. If a had a minimal polynomial of degree $d' < d$, then $a^0, \ldots, a^{d-1}$ would only generate a vector space of dimension d', and thus not the full field. Thus, in the same way as an element "encodes" the full multiplicative group $\mathbb{F}_{q^d}^\times$ iff. it is of multiplicative order $q^d - 1$, it encodes the full additive group iff. its minimal polynomial over the base field is of maximal degree.

The above Example 30 goes a fair way to actually proving Theorem 20, as a full proof would only need to show in the last step that a suitable primitive element with a prescribed minimal polynomial always exists. We do not provide such a proof but sketch the main steps and arguments.

Consider $\mathbb{F}/\mathbb{F}_q$, $[\mathbb{F} : \mathbb{F}_q] = d$, then from Corollary 11 every element of $\mathbb{F}$ is a root of $P := X^{q^d} - X$, *i.e.* P *splits* over $\mathbb{F}$. On the other hand one may show that over $\mathbb{F}_q$ the decomposition of P into irreducible factors contains *all* the monic irreducible polynomials of degree dividing d and it follows that those all split over $\mathbb{F}$. To build an isomorphism between $\mathbb{F}$ and another field $\mathbb{F}'$ of the same cardinality, it is sufficient to map a primitive element of $\mathbb{F}$ to one of $\mathbb{F}'$ with the same minimal polynomial Q, which is akin to finding a root of Q in $\mathbb{F}'$. One concludes by observing that the minimal polynomial is always monic and irreducible, and always of degree d if the associated element is primitive. Furthermore, the primitivity of an element is entirely determined by its minimal polynomial in the sense that all the primitive elements are exactly the roots of certain *primitive* irreducible polynomials of maximum degree d. Note however that not all irreducible polynomials are primitive, which means that in general not all elements with minimal polynomial of degree d are primitive.

We now give an alternative description of the process used to build extension fields in Theorem 24: one may build a degree-d extension $\mathbb{F}_{q^d}$ of $\mathbb{F}_q$ by adding a root α of a degree-d polynomial P that is irreducible in $\mathbb{F}_q$. If we write $\mathbb{F}_q[\alpha]$ for such a construction, then α corresponds to (the class of) X in $\mathbb{F}_q[X]/\langle P \rangle$; indeed in the latter case the minimal polynomial of X is P, *i.e.* it is one of its roots.

In the case of finite fields the choice of the root used to build the extension $\mathbb{F}_q[\alpha]$ does not matter in the sense that adding *one* root of P to $\mathbb{F}_q$ is equivalent to adding *all* of them; we say that $\mathbb{F}_q[\alpha]$ is the *splitting field* of P.

Example 32. Let $P := X^2 + 1$ and denote one of its roots by i, then the field $\mathbb{C}$ of complex numbers can be built as $\mathbb{R}[i]$, or equivalently $\mathbb{R}[X]/\langle P \rangle$.

In the above Example 32, the resulting field $\mathbb{C}$ is *algebraically closed*, or equivalently it is the *algebraic closure* $\overline{\mathbb{R}}$ of $\mathbb{R}$: every polynomial of $\mathbb{C}[X]$ can be decomposed into linear factors; in particular this means that one cannot build further extensions of $\mathbb{C}$ by adding more roots of irreducible polynomials. In the case of finite fields, we know from Theorem 25 that the algebraic closure $\overline{\mathbb{F}_q}$ of any finite field $\mathbb{F}_q$ is not finite. A more direct (albeit non constructive) proof of this fact is that assuming that $\overline{\mathbb{F}_q}$ is finite, then $1 + \prod_{a \in \overline{\mathbb{F}_q}}(X - a)$ would have no root in $\overline{\mathbb{F}_q}$, which is a contradiction.

3 Arithmetic in binary fields

Our goal is now to discuss some implementation aspects for extension fields, and binary fields in particular. We will mostly focus on computing products, since additions are trivial and inversion can be implemented with an exponentiation or a GCD algorithm, which both only require addition and multiplication for their arithmetic.

3.1 Data representation and basic operations

Before describing algorithms and their implementation, we need to decide on a suitable way to represent our field elements. Since we know that a binary field $\mathbb{F}_{2^n}$ has the structure of a vector space $\mathbb{F}_2^n$ and that elements of $\mathbb{F}_2$ can be represented by 0 and 1, a natural choice is to represent $x \in \mathbb{F}_{2^n}$ as a vector of $\mathbb{F}_2^n$, itself represented by a binary string of length n. In a programming language, a binary string is often conveniently manipulated as the (unsigned) integer that it represents; that is, one uses the canonical embedding $\{0,1\}^* \hookrightarrow \mathbb{N}$, $b_k \cdots b_1 b_0 \mapsto \sum_{i=0}^{k} b_i 2^i$. We will use a similar convention in our case and often denote elements of $\mathbb{F}_{2^n}$ by an integer $[\![0, 2^n - 1]\!]$; one must be cautious however not to interpret this as the (wrong) fact that $\mathbb{F}_{2^n} \cong \mathbb{Z}/2^n\mathbb{Z}$. More generally, this convention will be used for any element of an $\mathbb{F}_2$-vector space, not necessarily a finite field.

Example 33. Let $\mathbb{F}_{2^4} \cong \mathbb{F}_2[X]/\langle X^4 + X + 1\rangle$, then the element $X^3 + X + 1$ is represented by the binary string 1011, written `0xB`. The irreducible polynomial $X^4 + X + 1$ used in this representation of $\mathbb{F}_{2^4}$ is represented by the binary string 10011, written `0x13`; since this defining polynomial is usually taken to be monic, if its degree is known from the context, a compressed representation as 0011, written `0x3` may be used instead.

The representation of vectors of $\mathbb{F}_2^n$ as unsigned integers goes further than being a compact encoding; basic operations in this representation are also readily available in many instruction sets, and accessible through any decent programming language. We illustrate this in `C` in the case of elements represented in a single 64-bit machine word, but extension to larger or smaller inputs are straightforward.

— The addition of two vectors coincides with the bitwise XOR: `u ^ v;`.

— The pointwise product coincides with the bitwise AND: `u & v;`.

— The Hamming weight of a vector (*i.e.* the number of coordinates where it is non-zero) coincides with the population count. On `x86` processors with the `POPCNT` instruction set extension, this can be computed using the `popcnt` instruction, accessible for instance through the intrinsic `_mm_popcnt_u64`.

— In the more specific case of polynomials of $\mathbb{F}_2[X]$, multiplication by X^i coincides with the left logical shift, though one must be careful to take possible overflows into account (alternatively, one could say that such a bit shift for registers of size n implement multiplication in $\mathbb{F}_2[X]/\langle X^n\rangle$).

Similarly, the quotient (resp. remainder) in the division by X^i coincides with the right logical shift: `p >> i;` (resp. bitwise AND with the string whose i lowest bits are 1: `p & ((1 << i) - 1)`).

More generally, the product coincides with the carryless multiplication in base two. On x86 processors with the PCLMULQDQ instruction set extension, this can be computed (up to possible overflows) using the pclmulqdq instruction, accessible for instance through the intrinsic _mm_clmulepi64_si128.

— In the more specific case of polynomials of $\mathbb{F}_2[X]/\langle X^n - 1\rangle$, multiplication by X^i coincides with the left circular shift (or rotation) on n-bit words.

3.2 Product in modular polynomial rings: an xtime approach

For a field $\mathbb{F}_{2^n}$ represented as $\mathbb{F}_2[X]/\langle P\rangle$ with P some irreducible polynomial of degree n, the product of two field elements may be implemented by exactly using this representation, *i.e.* from the product of two polynomials modulo P. Note also that the same would be true even if P were not irreducible, in the same way that arithmetic in $\mathbb{Z}/p\mathbb{Z}$ is in large part a special case of the one in $\mathbb{Z}/n\mathbb{Z}$. In this section we describe how to implement the product by using an "xtime" primitive operation; we will describe an alternative approach in Section 3.4 that uses a different primitive operation.

We start with the useful subcase where one of the operands in the product is the monomial X, resulting in an elementary operation customarily called xtime. We take $Q \in \mathbb{F}_2[X]$ of degree at most $n - 1$ and wish to compute $Q \times X \mod P$, *i.e.* the unique polynomial U s.t. $\deg(U) < n$, $Q \times X = V \times P + U$. This can be done easily by observing that: 1) if $\deg(Q \times X) < n$ then $U = Q \times X$; 2) if $\deg(Q \times X) \geqslant n$ then it is exactly n so $\deg(V) = 0$; but V is also non-zero and since we are working in $\mathbb{F}_2[X]$ it must be 1, so then $U = Q \times X - P$.

If polynomials are represented as in Section 3.1, the above operation can be implemented efficiently as in the example of Fig. 2.

```
/* Input:
   - Q, a binary polynomial of degree < 4
   Output:
   - Q*X mod X**4+X+1
*/
uint8_t xtime(uint8_t Q)
{
    if (Q < 8) // the polynomial represented by Q has deg < 3
        return Q << 1;
    else
        return (Q << 1) ^ 0x13;
}
```

Figure 2: Multiplication of Q by X modulo $X^4 + X + 1$.

If the underlying architecture uses two's complement representation for signed integers, the branchless alternative of Fig. 3 may be preferred.

Remark 34. The above xtime operation is sometimes described to correspond to the clocking of a *Linear-Feedback Shift Register* (LFSR) in what is sometimes referred to as a *Galois configuration.* This is following the view that in xtime, one *shifts* bits in a register and *linearly* updates the state of the register with the leftmost, thrown-out bit. We defer our discussion of LFSRs to Section 4.

We are now ready to address the general case, using xtime as a subroutine. This essentially follows from distributivity, and using a process similar to the one of Fig. 1.

```c
/* Input:
   - Q, a binary polynomial of degree < 4
   Output:
   - Q*X mod X**4+X+1
*/
uint8_t xtime_bl(uint8_t Q)
{
    uint8_t m = -(Q >> 3); // ~0 if Q >> 3 == 1, 0 otherwise
    return (Q << 1) ^ (m & 0x13);
}
```

Figure 3: Multiplication of Q by X modulo $X^4 + X + 1$, branchless.

The idea is to remark that one can write the product $Q \times R \mod P$ as $Q \sum_{i=0}^{n-1} R_i X^i \mod P$, where $R = \sum_{i=0}^{n-1} R_i X^i$. Then since the R_i's belong to $\{0, 1\}$ one only needs to compute $Q \times X^i$ up to $i = \deg(R)$ using repeated calls to `xtime` and sum those terms for which $R_i = 1$. This is detailed in Fig. 4.

```c
/* Input:
   - Q, R binary polynomials of degree < 4
   Output:
   - Q*R mod X**4+X+1
*/
uint8_t gf16mul(uint8_t Q, uint8_t R)
{
    uint8_t res = 0;
    uint8_t acc = Q;
    while (R > 0)
    {
        if (R & 1)
            res ^= acc;
        acc = xtime(acc);
        R >>= 1;
    }
    return res;
}
```

Figure 4: Multiplication of Q by R modulo $X^4 + X + 1$.

3.3 Multiplication by a constant

One may sometimes be interested solely in the multiplication by a fixed (but arbitrary) constant $\alpha \in \mathbb{F}_{2^n} \cong \mathbb{F}_2[X]/\langle P \rangle$ rather than by an arbitrary element. Although it is of course possible to still use a general multiplication algorithm to do so, this specialisation opens the way to a more dedicated approach. The idea is to observe that multiplication by a constant is a linear operation, and since $\mathbb{F}_{2^n}$ has the structure of an $\mathbb{F}_2$-vector space, for every α there must exist a matrix $\boldsymbol{M}_\alpha \in \mathbb{F}_2^{n \times n}$ implementing $x \mapsto x\alpha$ as $\boldsymbol{x} \mapsto \boldsymbol{x}\boldsymbol{M}_\alpha$, writing $\boldsymbol{x}$ for x to emphasise its structure as a vector and using the right product for $\boldsymbol{M}_\alpha$ (making $\boldsymbol{x}$ a *row* vector).

We first describe $\boldsymbol{M}_X$ before generalising to arbitrary constants. In this case, $\boldsymbol{M}_X$ in fact implements `xtime`; taking the (somewhat arbitrary) convention that the polynomial $\sum_{i=0}^{n-1} Q_i X^i$ be represented as the vector $\begin{pmatrix} Q_0 & \cdots & Q_{n-1} \end{pmatrix}$, then one has:

$$\boldsymbol{M}_X = \begin{pmatrix} \boldsymbol{0}_{n-1} & \boldsymbol{I}_{n-1} \\ & \boldsymbol{p} \end{pmatrix},$$

with $\boldsymbol{0}_{n-1} \in \mathbb{F}_2^{(n-1)\times 1}$ the all-zero column vector in dimension $n-1$, $\boldsymbol{I}_{n-1} \in \mathbb{F}_2^{(n-1)\times(n-1)}$ the identity matrix, and $\boldsymbol{p} = \begin{pmatrix} P_0 & \cdots & P_{n-1} \end{pmatrix}$ an encoding of the defining polynomial P minus its degree-n coefficient. This matrix is in fact the *companion matrix* of P,[¶] a canonical representative of all matrices with minimal polynomial equal to P.

Example 35. In the field $\mathbb{F}_{2^4} \cong \mathbb{F}_2[X]/\langle X^4 + X + 1\rangle$, one has:

$$\boldsymbol{M}_X = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}.$$

Now to define $\boldsymbol{M}_\alpha$ for an arbitrary element α one uses that the minimal polynomial of X is of degree n, which means that $\boldsymbol{M}_\alpha \in \langle \boldsymbol{M}_{X^i} = \boldsymbol{M}_X^i \rangle_{0 \leqslant i < n}$. Concretely, writing $\alpha = \sum_{i=0}^{n-1} \alpha_i X^i$, one simply has $\boldsymbol{M}_\alpha = \sum_{i=0}^{n-1} \alpha_i \boldsymbol{M}_{X^i}$.

Example 36. Continuing Example 35, take $\alpha = X^3 + X + 1$, then:

$$\boldsymbol{M}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \boldsymbol{M}_{X^3} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad \boldsymbol{M}_\alpha = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

From an implementation perspective, the vector-matrix product $\boldsymbol{x}\boldsymbol{M}_\alpha$ can be efficiently implemented in the binary case by using a *broadcast* approach. The idea is to remark that the result is the sum of the rows of $\boldsymbol{M}_\alpha$ for which the corresponding coordinate in $\boldsymbol{x}$ is non-zero, and this can be computed efficiently if $\boldsymbol{M}_\alpha$ is stored in row-major format. We illustrate this in Fig. 5, in dimension 64 (assuming a two's complement representation for signed integers).

We conclude with the remark that the matrix $\boldsymbol{M}_\alpha$ of the multiplication by α may also be used to reduce the computation of inverses in $\mathbb{F}_{2^n}$ to linear algebra in $\mathbb{F}_2^{n\times n}$. Indeed it is clear that $\boldsymbol{M}_\alpha^{-1} = \boldsymbol{M}_{\alpha^{-1}}$, and the only potential difficulty in using this approach as a general inversion algorithm would be to recover α^{-1} from $\boldsymbol{M}_\alpha^{-1}$ (or in fact α from $\boldsymbol{M}_\alpha$ in general). However this latter task is made rather easy by observing that due to the special structure of $\boldsymbol{M}_X$, the only non-zero coefficient in the first row of $\boldsymbol{M}_{X^i}$ is in the i^{th} coordinate (with indices starting from zero); it is thus straightforward to recover an expression of α in the basis $\langle X^i \rangle_{0 \leqslant i < n}$. In fact this observation tells us that if what we wish to compute is α^{-1} rather than its full multiplication matrix, it is in fact enough to compute only the first row of $\boldsymbol{M}_\alpha^{-1}$.

Without more refinements, this approach for computing the inverse is not competitive with for instance using a GCD algorithm, since it ignores most of the special structure of the input. It may however be improved, for instance by exploiting a tower structure (if present) and being made recursive.

Example 37. Continuing Example 36, we compute

$$\boldsymbol{M}_{\alpha^{-1}} = \boldsymbol{M}_\alpha^{-1} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix},$$

[¶]Or rather, one of the four possible definitions for the companion matrix, since this particular one reflects our ordering for the coefficients of polynomials and the fact that we multiply on the right.

```
/* Input:
   - x, a binary vector of dimension 64
   - M, a binary 64*64 matrix in row-major format
   Output:
   - x*M
*/
uint64_t mmul64(uint64_t x, uint64_t M[64])
{
    uint64_t res = 0;
    for (int i = 0; i < 64; i++)
    {
        uint64_t m = -(x & 1);
        res ^= m & M[i];
        x  >>= 1;
    }

    return res;
}
```

Figure 5: Vector-matrix multiplication in $\mathbb{F}_2^{64\times 64}$.

whose first row reveals that $\alpha^{-1} = X^2 + 1$.

3.4 Product in modular polynomial rings: a carryless approach

It is clear that to compute a product in a modular ring, one may compose the computation of the product in the corresponding "full" (not modular) ring with a reduction step. The goal of this section is to describe such a reduction algorithm for polynomials (adapted from Barrett's similar algorithm for integers), *i.e.* which on input P, Q computes the unique polynomial R s.t. $\deg(R) < \deg(P)$ and $R \equiv Q \mod P$, that uses full polynomial multiplication and addition as primitive operations; one also needs to compute quotients and remainders of divisions by monomials X^d, but those are easy since they only consist in keeping the monomials of degree more and less than d respectively, as already remarked in Section 3.1. This reduction algorithm is then especially useful if one is able to compute products efficiently, which is for instance the case in $\mathbb{F}_2[X]$ if an instruction such as `PCLMULQDQ` is available; in that case both the full product and the reduction will essentially rely on this instruction. We will also see that, justifying Remark 21, the cost of this reduction algorithm will depend on the "shape" of the reduction polynomial P: this means that for a fixed field (up to isomorphism), the implementation of arithmetic using this approach will be more efficient in some representations than in others.

The algorithm is defined in Fig. 6. To prove its cost and its correctness, we will use an invariant on `A` + `B` and a variant on `B`. Namely, we have that `A` + `B` is always congruent to Q modulo P, and that the degree of `B` decreases by at least $d - d_u \geqslant 1$ at every iteration (where $d_u := \deg(\mathtt{U})$) until its degree becomes less than d, at which point the algorithm terminates in the next iteration at the latest. It follows that the algorithm returns the correct result after at most $(\deg(Q) - d)/(d - d_u)$ iterations, and it will then be more efficient when d_u is small. Now to prove the invariant $\mathtt{A} + \mathtt{B} \equiv Q \mod P$ we observe that this is initially true; also $\deg(\mathtt{A}) < d$ since `A` is initially zero and one only adds to it polynomials of degree less than d. Then the previous equivalence can be written as $\mathtt{A} + \mathtt{B}_{lo} + \mathtt{B}_{hi} \equiv Q \mod P$, where $\mathtt{B}_{lo}$ (resp. $\mathtt{B}_{hi}$) denotes the terms of `B` of degree at most $d - 1$ (resp. degree at least d), *i.e.* $\mathtt{B}_{lo}$ = `B % X**d`, $\mathtt{B}_{hi}$ = `X**d(B / X**d)`, and the conclusion follows from the fact that `U` ≡ `X**d` mod P. Finally the variant on `B` comes from the fact that the degree of `B / X**d` is at least d less than the one of `B` when $\deg(\texttt{B}) \geqslant d$ and zero otherwise, and that the one of `U` is d_u by definition.

```
1           /* Input:
2              - P, Q; deg(P) = d
3              Output:
4              - R ~ Q mod P, deg(R) < d
5           */
6           REDBL(P, Q)
7           {
8               A = 0;
9               B = Q;
10              U = X**d % P;
11              while (B > 0)
12              {
13                  A += B % X**d;
14                  B  = (B / X**d)*U;
15              }
16              return A;
17          }
```

Figure 6: Reduction of Q modulo P, using a Barrett-like approach.

We now give in Fig. 7 an example of an implementation of the above algorithm in `C`, using `PCLMULQDQ` accessed from the `_mm_clmulepi64_si128` intrinsic. More precisely, this is a full implementation of the multiplication in $\mathbb{F}_{2^{64}} \cong \mathbb{F}_2[X]/\langle X^{64} + X^4 + X^3 + X + 1\rangle$ that uses the above for the reduction step. In some more details, lines 8 and 9 set the low 64 bits of two 128-bit registers to the binary representation of the operands Q and R respectively, while line 10 does it for the representation of X^{64} modulo the chosen irreducible polynomial, *i.e.* $X^4 + X^3 + X + 1$. Line 12 assigns to `b` the result of the carryless (*i.e.* "binary polynomial") multiplication of `q` and `r`. This is a polynomial of degree at most 126, which then fits into 128 bits. The last operand `0x00` means here that the two inputs of degree at most 63 are to be found in the 64 low bits of the first two operands. Line 15 performs a similar computation between `b` and `u`, but this time with last operand `0x01`, meaning that the first polynomial is defined by the 64 *high* bits of `b`, *i.e.* the terms of degree above 64 then divided by X^{64}; that is, this line corresponds to the computation of line 14 in Fig. 6. Note that at this point, the polynomial represented by `b` has degree at most $66 = 126 - (64 - 4)$. Line 16 adds the full result to `a` that already contains the previous value of `b` (however only the 64 low bits of this register, corresponding to the terms of degree less than 64, will eventually be used). Line 17 proceeds as line 15; at this point the polynomial represented by `b` has degree at most 6, so we know that another iteration would result in the zero polynomial and there would be nothing more to add to `a`. This means that the fully-reduced result is to be found in the 64 low bits of `a` after line 18, which are then returned in line 20. We conclude with two remarks: first the implementation in Fig. 7 runs in "constant time", in the sense that the amount of work does not depend on the input (in particular it always does the maximum work needed to fully reduce the product, which may be more than what is sometimes needed); this may be useful in contexts when one does not want the computation time to leak information about the operands. Second, we again emphasize that this running time was kept low by using an irreducible polynomial s.t. d_U is particularly small.

```
/* Input:
   - Q, R binary polynomials of degree < 64
   Output:
   - Q*R mod X**64 + X**4 + X**3 + X + 1
 */
uint64_t gf2_64mul(uint64_t Q, uint64_t R)
{
    __m128i q = _mm_set_epi64x(0, Q);
    __m128i r = _mm_set_epi64x(0, R);
    __m128i u = _mm_set_epi64x(0, 0x1B);
    __m128i a;
    __m128i b = _mm_clmulepi64_si128(q, r, 0x00);

    a = b;
    b = _mm_clmulepi64_si128(b, u, 0x01);
    a = _mm_xor_si128(a, b);
    b = _mm_clmulepi64_si128(b, u, 0x01);
    a = _mm_xor_si128(a, b);

    return _mm_extract_epi64(a, 0);
}
```

Figure 7: Multiplication of Q by R modulo $X^{64} + X^4 + X^3 + X + 1$.

4 Linearly-recurring sequences

In this section we will study two aspects of *linearly-recurring sequences*: we will first look at how sequences with a known recurrence can be efficiently computed, and then look at the opposite problem of computing the smallest linear recurrence that generates a known sequence. We will focus on sequences over finite fields (and with examples mostly drawn from $\mathbb{F}_2$) for simplicity, but the general theory (and the results we will present) similarly works (at least to some extent) over (not-necessarily finite) rings.

4.1 Definitions and first properties

We consider sequences (u_n) over some finite field $\mathbb{F}$, except if specified otherwise. We will use a few useful conventions:

— We allow negative indices, and for all such $n < 0$ define u_n to be 0.

— The only sequence for which $u_0 = 0$ is the all-zero sequence. That is, any sequence with at least one non-zero term has its index defined such that the first non-zero term is in position 0.

— If $P \in \mathbb{F}[X]$ is a polynomial, we will denote by (p_n) the sequence of its coefficients.

— We may use both terms "index" and "rank" to refer to n for a term $u_n \in (u_n)$. We tend to favour the former when talking about a specific term, and the latter when talking about a property that is true for all terms after a given one.

We now give:

Definition 38 (Convolution of two sequences)**.** Let (u_n), (v_n) be two sequences, its *convolution* $((u * v)_n) =: (w_n)$ is the sequence defined by $w_n = \sum_{i=0}^{\infty} u_i v_{n-i}$. Equivalently, $w_n = \sum_{i+j=n} u_i v_j$, which makes it clear that $((u * v)_n) = ((v * u)_n)$.

Note that if (p_n) and (q_n) are sequences representing the coefficients of polynomials P and Q, then $((p * q)_n)$ represents the coefficients of $P \times Q$. That is, the convolution product is the "extension" to sequences of the polynomial product.

We now give two definitions for linearly-recurring sequences, and will then prove their equivalence. The first "primal" characterisation defines a sequence from a linear recurrence that from finitely-many successive terms of the sequence specifies how to generate the following one, and the second "dual" characterisation defines a sequence from a linear relation that annihilates it after a given rank.

Definition 39 (Linearly-recurring sequence (primal characterisation))**.** A sequence (u_n) is *linearly-recurring* if $\exists\, k \in \mathbb{N}$, (g_n) over $\mathbb{F}$ s.t. $g_i = 0$ for all $i \geqslant k$, and for all $n+1 \geqslant k$ one has that $u_{n+1} = \sum_{i=0}^{k-1} g_i u_{n-i}$. Equivalently, for all $n+1 \geqslant k$ one has that $u_{n+1} = (g * u)_n$.

Example 40. Let $\mathbb{F} = \mathbb{F}_2$, $u_0 = u_1 = g_0 = g_1 = 1$, then the first 10 terms of (u_n) are [1, 1, 0, 1, 1, 0, 1, 1, 0, 1].

Let $\mathbb{F} = \mathbb{F}_2$, $u_0 = 1$, $u_1 = 0$, $u_2 = 1$, $u_3 = 0$, $g_0 = g_1 = 0$, $g_2 = g_3 = 1$, then the first 20 terms of (u_n) are [1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1].

The relation $u_{n+1} = \sum_{i=0}^{k-1} g_i u_{n-i}$ of Definition 39 may of course be "shifted" to express any term u_{n+j} in function of the u_{n+j-i}, $1 \leqslant i < k$, preceding ones. A popular way to do so is to define $u_{n+k} = \sum_{i=0}^{k-1} g'_i u_{n+i}$ which, since the index i is reversed, also changes the definition of (g_n) into (g'_n) as $g'_i = g_{(k-1)-i}$.

We may now observe that up to another simple change in the definition of (g_n), the relation $u_{n+1} = (g * u)_n$ implies $(\tilde{g} * u)_n = 0$ for some $(\tilde{g}_n)$. Indeed if $u_{n+1} = \sum_{i=0}^{k-1} g_i u_{n-i}$, then surely $u_{n+1} - \sum_{i=0}^{k-1} g_i u_{n-i} = 0$, and the left-hand term can be rewritten as $u_n - \sum_{i=1}^{k} g_{i-1} u_{n-i}$, or $\sum_{i=0}^{k} \tilde{g}_i u_{n-i}$ with $\tilde{g}_0 = 1$, $\tilde{g}_i = -g_{i-1}$ for $1 \leqslant i < k$. This leads to the second, dual characterisation of linearly-recurring sequences, which as we will argue is in fact the "right" one.

Definition 41 (Linearly-recurring sequence (dual characterisation))**.** A sequence (u_n) is *linearly-recurring* if $\exists\, k \in \mathbb{N}$, $A \neq 0 \in \mathbb{F}[X]$ a polynomial of degree k s.t. $\forall\, n \geqslant k$, $(a * u)_n = 0$. The polynomial A is said to *annihilate* (u_n) after the rank k, and is called an *annihilator* of the sequence.

It may seem arbitrary to specifically regard the above sequence (a_n) as a polynomial A, but we will later justify this by highlighting the relation between a linearly-recurring sequence annihilated by A and the ring $\mathbb{F}[X]/\langle A \rangle$.

Proposition 42. *The two Definitions 39 and 41 are equivalent.*

Proof. From the discussion preceding Definition 41. □

Example 43. An annihilator for the first sequence of Example 40 is $X^2 + X + 1$. An annihilator for the second sequence is $X^4 + X^3 + 1$.

Using the dual characterisation of a linearly-recurring sequence, it is easy to define the *linear complexity* of a sequence, which is the smallest degree for which an annihilator exists (if there is one). Formally:

Definition 44 (Linear complexity of a sequence)**.** The *linear complexity* of a sequence (u_n) is the smallest integer ℓ (if it exists) s.t. $\exists\, \ell' \in \mathbb{N}$, $A \neq 0 \in \mathbb{F}[X]$, $\deg(A) = \ell$ and $\forall\, n \geqslant \ell'$, $(u * a)_n = 0$. If there is no such ℓ, the linear complexity of (u_n) is defined to be infinite, noted ∞.

By definition, linearly-recurring sequences are exactly the sequences of finite linear complexity.

Definition 44 extends to finite (or *truncated*) sequences where one only considers a finite number of terms. There, except in the trivial case of a sequence with a single term, the linear complexity is always finite and at most the rank of the largest non-zero term. Indeed, either there are at least two non-zero terms and the last one can always be expressed as a linear combination of the previous ones, or there is a single non-zero term which by convention is at index 0 and the sequence is then annihilated by the constant polynomial 1. Yet, as it should become clear thanks to the following Example 46, the notion may become ambiguous, since there may for instance be cases where a polynomial of (essentially minimal) degree one annihilates the last term, but not any of the others. We will thus adopt the following:

Definition 45 (Linear complexity of a sequence (finite case))**.** Let (u_n) be a finite sequence, and ℓ' the smallest integer for which $\exists\, A \neq 0 \in \mathbb{F}[X]$ s.t. $\forall\, n \geqslant \ell'$, $(u * a)_n = 0$, then the linear complexity of (u_n) is the least degree ℓ of any such polynomial A.

We will later see in Section 4.3 how to compute an annihilator satisfying the condition of Definition 45 (and thus the linear complexity) of any finite sequence.

Example 46.

1. Let $\mathbb{F} = \mathbb{F}_2$, the finite sequence $[1, 1, 0, 1]$ is annihilated into $[1, 0, 0, 0]$ by $X^2 + X + 1$, which is the polynomial of least degree to do so, and this sequence has linear complexity 2.

2. Let $\mathbb{F} = \mathbb{F}_2$, the sequence $[1, 1, 0, 1, 1, 1, 1, \ldots]$ is annihilated into $[1, 0, 1, 1, 0, \ldots]$ by $X + 1$, which is the polynomial of least degree to do so, and this sequence has linear complexity 1.

3. Let $\mathbb{F} = \mathbb{F}_2$, the periodic sequence $[1, 0, 1, 0, 1, \ldots]$ is annihilated into $[1, 0, \ldots]$ by $X^2 + 1$, which is the polynomial of least degree to do so, and this sequence has linear complexity 2.

4. Let $\mathbb{F} = \mathbb{F}_2$, the periodic sequence $[1, 0, 1, 1, \ldots]$ is annihilated into $[1, 0, 1, 1, 0, \ldots]$ by $X^4 + 1$, which is the polynomial of least degree to do so, and this sequence has linear complexity 4.

5. Let $\mathbb{F} = \mathbb{F}_2$, the periodic sequence $[1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, \ldots]$ (the second sequence of Example 40) is annihilated into $[1, 0, 1, 1, 0, \ldots]$ by $X^4 + X^3 + 1$, which is the polynomial of least degree to do so, and this sequence has linear complexity 4.

6. Let $\mathbb{F} = \mathbb{F}_2$, the finite sequence $[1, 0, 1, 1]$ is annihilated into $[1, 1, 1, 0]$ by $X + 1$, $[1, 1, 0, 0]$ by $X^2 + X + 1$, and $[1, 0, 0, 0]$ by $X^3 + X^2 + 1$ which is the polynomial of least degree to do so, and this sequence has linear complexity 3.

The above examples illustrate some notable properties of the linear complexity. A first one is that from (1) and (2), it is possible for the linear complexity of a finite prefix of a sequence to be larger than the one of the full sequence. Indeed it is not hard to build such sequences, as is done in the above: simply start from a finite sequence of linear complexity ℓ and complete it with a sequence of strictly smaller complexity. In this case, an annihilator of least degree does not necessarily annihilate the sequence “early”, since it is only required to annihilate the suffix and there is no particular reason that it would also annihilate the independent prefix.

Another property illustrated by (3), (4) and (5) is that a periodic sequence of period T has linear complexity at most T since it is always annihilated by $-X^T + 1$ (or equivalently, $X^T - 1$). The linear complexity may however also be smaller.

This last property, together with the fact that defining the linear complexity in the finite case is rather cumbersome (see (6)), justifies why one may be tempted to analyse a finite sequence by making it periodic, rather than cutting it short. This results in the definition of the *cyclic* linear complexity, which simply consists in considering the "standard" linear complexity of Definition 44 for a finite sequence that is made infinite by repeating itself indefinitely. This results in:

Definition 47 (Cyclic linear complexity of a sequence)**.** Let (u_n) be a periodic sequence of period T, $U = \sum_{i=0}^{T-1} u_i X^i \in \mathbb{F}[X]$ the polynomial formed by its T first coefficients. The *cyclic linear complexity* of (u_n) is the degree of the least non-zero polynomial $A \in \mathbb{F}[X]$ s.t. $AU \equiv 0 \mod X^T - 1$.

This calls for several comments. First, the notion is well-defined since $X^T - 1$ itself always satisfies the condition; as can be seen from some of the cases of the above Example 46, it may be that this is also the solution of least degree. Second, the formulation of this condition purely in terms of polynomials is justified from the fact that annihilating (u_n) after a given rank (which is always possible, as per the previous remark) is equivalent to annihilating T consecutive terms "completely" using a *cyclic* convolution, which itself is equivalent to the polynomial product modulo $X^T - 1$. In more details, let $A \neq 0 \in \mathbb{F}[X]$ of degree $\ell \leqslant T$, ℓ' be such that for all $n \geqslant \ell'$ $(a * u)_n = 0$, that is $\sum_{i=0}^{\ell} a_i u_{n-i} = 0$. Since (u_n) is periodic of period T, $u_{n+T} = u_n$, and so the terms u_{n-i} in the previous equality may be "replaced" by $u_{(n-i)\%T}$ (where $x\%N$ denotes the non-negative remainder of the division of x by N) and it follows that for all $n \in [\![0, T-1]\!]$, $\sum_{i=0}^{\ell} a_i u_{(n-i)\%T} = 0$. This latter implication may then be expressed as $(a * u)^{\circlearrowleft T} = 0$, where the i^{th} term $(u * v)_i^{\circlearrowleft T}$ of the *cyclic convolution* of period T of the two finite sequences (u_n) and (v_n) is defined as $\sum_{i=0}^{T-1} u_i v_{(n-i)\%T}$. We leave it to the reader to check that this latter corresponds to the product in $\mathbb{F}[X]/\langle X^T - 1\rangle$.

Remark 48. An alternative, possibly cleaner, formulation of Definition 47 solely in terms of the structure $R := \mathbb{F}[X]/\langle X^T - 1\rangle$ would be to define the cyclic linear complexity as the least degree of a non-zero $A \in R$ that annihilates U if it exists, and T otherwise. Note that such an A exists iff. U is a non-trivial zero divisor in R, which may well happen since the latter is not an integral domain.

An important, rather immediate consequence of Definition 47 is that the cyclic linear complexity of the T-periodic sequence represented by the polynomial U is directly related to the common factors of U and $X^T - 1$. Indeed, the condition that AU be zero in $\mathbb{F}[X]/\langle X^T - 1\rangle$ is equivalent to requiring that $AU = (X^T - 1)B$ in $\mathbb{F}[X]$, for some other polynomial B, and the degree of A is minimised when B is constant. Thus writing U and $X^T - 1$ as the products of their irreducible factors over $\mathbb{F}$, it is clear that an annihilator A of least degree is given by the product of the factors of $X^T - 1$ not found in U.

In the special case where $(X^T - 1)$ splits in $\mathbb{F}$, one may reexpress this in terms of a T-adic "Fourier" transform. Recall from our study of the structure of $\mathbb{F}^\times$ that if $X^N - 1$ splits in $\mathbb{F}$ then its roots can be written as $\{1, \omega, \ldots, \omega^{N-1}\}$ for some $\omega \in \mathbb{F}$ of (multiplicative) order N, and the elements of order N in $\mathbb{F}$ are exactly $\{\omega^t \mid \gcd(t, N) = 1\}$. We then get:

Definition 49 (N-adic transform)**.** Let $P \in \mathbb{F}[X]$ be a polynomial of degree $N - 1$, $\omega \in \mathbb{F}$ be an element of order N. Its *N-adic transform* $\widehat{P} \in \mathbb{F}^N$ is given by $\big(\mathrm{eval}(P, \omega^i)\big)$, $0 \leqslant i < N$.

From the above discussion, the exact choice of element of order N for ω does not matter, since two different choices lead to identical transforms up to a permutation of coordinates.

We then have:

Theorem 50. *Let (u_n) be a periodic sequence of period T of cyclic linear complexity ℓ, represented by $U \in \mathbb{F}[X]$ of degree $T-1$. Then if $\exists\, \omega \in \mathbb{F}$ of order T, and denoting by $\widehat{U}$ the T-adic transform of U, one has the equality $\ell = wt(\widehat{U})$.*

Where in the above $wt(\cdot)$ denotes the number of non-zero coordinates, or *weight* of its argument.

Proof. From the existence of $\omega \in \mathbb{F}$ of order T, one has that $X^T - 1$ factors in $\mathbb{F}$ as $\prod_{i=0}^{T-1}(X - \omega^i)$, and a non-zero annihilator A of least degree ℓ is given by $\prod_{i=0}^{T-1}(X - \omega^i) \nmid U$. Since $(X - \omega^i) \nmid U$ iff. $\mathrm{eval}(U, \omega^i) \neq 0$, the number of terms in the latter product is equal to the number of ω^i's on which the evaluation of U is non-zero, *i.e.* $wt(\widehat{U})$. $\square$

Theorem 50 (and the more general characterisation given above) immediately suggest an algorithm to compute the cyclic linear complexity of a periodic sequence. If the T-adic transform can be computed in quasi-linear time, then it will not be possible to do (much) better in general. However the algorithm given in Section 4.3 (or similar techniques) may improve on this approach if a bound on ℓ is known (or assumed), since its cost will depend (at most quadratically) on this bound, and not on the possibly larger period.

4.2 Linear-feedback shift registers

We will now look at linearly-recurring sequences from a slightly different point of view, *viz.* the one of *linear-feedback shift registers* (or *LFSRs* for short). Over finite fields, this is most suited to analyse periodic sequences.

We start with a general definition of LFSRs:

Definition 51 (Linear-feedback shift register)**.** A *linear-feedback shift register* is a register $S^t = [S^t_{n-1}, \ldots, S^t_0]$ of *size* $n > 0$ over a finite field $\mathbb{F}$, indexed by a discrete "time" parameter $t \in \mathbb{N}$ s.t.:

1. At every t, an *output value* $s_t := S^t_{n-1}$ is produced.
2. The *state* S^{t+1} of the register S is given by $S^{t+1} = [S^t_{n-2}, \ldots, S^t_0, 0] + \Phi(S^t)$, where Φ is a linear *feedback* function that maps a register $[R_{n-1}, \ldots, R_0]$ to another register $[\sum_{i=0}^{n-1} \Phi_i^{n-1} R_i, \ldots, \sum_{i=0}^{n-1} \Phi_i^0 R_i]$ for some fixed coefficients $\Phi_i^j \in \mathbb{F}$,[‖] and where the addition between two registers is defined from their canonical embedding into $\mathbb{F}^n$.

From their definitions we immediately have the following properties for LFSRs (in fact also true of any sequence obtained similarly by iterating a non-necessarily linear function on a state S):

1. An LFSR generates an infinite sequence.
2. The sequence generated by an LFSR is always periodic after some rank n.
3. The sequence generated by an LFSR has a finite linear complexity.
4. The maximum (over S^0 and Φ) period of an LFSR of size n over $\mathbb{F}$ is at most $\#\mathbb{F}^n - 1$.

Exercise 6. Prove the above statements.

[‖] This could be summarised by simply saying that as a linear map, Φ can be represented by a matrix.

The general form of Definition 51 is in fact unnecessarily complex, and one usually favours one of two simpler variants where Φ is restricted to be s.t. $\Phi_i^{n-1}, \ldots, \Phi_i^1$ are zero for all i, or s.t. $\Phi_i^{n-1}, \ldots, \Phi_i^0$ are zero for all $i < n-1$. In the former case, the LFSRs obtained thusly closely follow the primal characterisation of linearly-recurring sequences of Definition 39 and are sometimes called "Fibonacci" LFSRs, while in the latter the LFSRs closely follow the dual characterisation of Definition 41 and are sometimes called "Galois" LFSRs. We will now make these remarks more concrete and show how, in the same way as Definitions 39 and 41 both define linearly-recurring sequences, these two special forms of LFSRs may be used interchangeably to generate the same sequences. We will do this by first connecting effectively the two forms with the Definition 39 for linearly-recurring sequences, and highlight the relation between the dual form and Definition 41 in a later step.

"Primal" LFSRs. Consider an LFSR of size n with some initial state S^0 and a feedback function Φ s.t. $\Phi_i^{n-1}, \ldots, \Phi_i^1$ are zero for all i. In other words, Φ is a map $[R_{n-1}, \ldots, R_0] \mapsto [0, \ldots, 0, \sum_{i=0}^{n-1} \Phi_i\, R_i]$ (where we drop the exponent '0' in Φ^0 for the sake of conciseness), and the state S^{t+1} is defined as $[S_{n-2}^t, \ldots, S_0^t, \sum_{i=0}^{n-1} \Phi_i\, S_i^t]$.

The sequence formed by the output values s_t is such that $s_t = S^0_{(n-1)-t}$ for $0 \leqslant t < n$, and more generally $s_t = S_i^{t-(n-1)+i}$ for $0 \leqslant i < n$, provided that $t-(n-1)+i \geqslant 0$. When $t \geqslant n$, one further gets more explicitly that $s_t = S_0^{t-(n-1)} = \sum_{i=0}^{n-1} \Phi_i\, S_i^{t-(n-1)-1} = \sum_{i=0}^{n-1} \Phi_i\, S_i^{t-n}$. Using $S_i^{t-n} = s_{t-(i+1)}$, this may then be rewritten solely in terms of s_t's as $s_t = \sum_{i=0}^{n-1} \Phi_i\, s_{t-(i+1)}$, or equivalently $s_{t+1} = \sum_{i=0}^{n-1} \Phi_i\, s_{t-i}$. Finally this last expression is immediately identifiable with the one for u_{n+1} in Definition 39, where in particular Φ exactly gives the non-zero terms of the sequence (g_n).

"Dual" LFSRs. Consider an LFSR of size n with some initial state S^0 and a feedback function Φ s.t. $\Phi_i^{n-1}, \ldots, \Phi_i^0$ are zero for all $i < n-1$. In other words, Φ is a map $[R_{n-1}, \ldots, R_0] \mapsto [\Phi^{n-1}\, R_{n-1}, \ldots, \Phi^0\, R_{n-1}]$ (where we drop the indices '$n-1$' in the Φ_{n-1}^i's for the sake of conciseness), and the state S^{t+1} is defined as $[S_{n-2}^t + \Phi^{n-1}\, S_{n-1}^t, \ldots, S_0^t + \Phi^1\, S_{n-1}^t, \Phi^0\, S_{n-1}^t]$.

The sequence formed by the output values s_t is such that $s_0 = S^0_{n-1}$, $s_1 = S^0_{n-2} + \Phi^{n-1}\, s_0$, and more generally for $0 \leqslant t < n$, $s_t = S^0_{(n-1)-t} + \sum_{i=0}^{t-1} \Phi^{(n-t)+i}\, s_i$; for $t \geqslant n$, $s_t = \sum_{i=t-n}^{t-1} \Phi^{(n-t)+i}\, s_i$. Starting the index i from zero, this latter expression becomes $s_t = \sum_{i=0}^{n-1} \Phi^i\, s_{t-n+i}$, and if one defines Φ' from $\Phi'^i = \Phi^{n-1-i}$ one may further rewrite it as $s_t = \sum_{i=0}^{n-1} \Phi'^i\, s_{t-1-i}$, or again $s_{t+1} = \sum_{i=0}^{n-1} \Phi'^i\, s_{t-i}$. This last expression may then again readily be identified with the one for u_{n+1} in Definition 39, where in particular Φ' exactly gives the non-zero terms of the sequence (g_n).

The equivalence of the two types of LFSRs in terms of generated sequence should now be apparent: given an LFSR in dual form, one may construct an equivalent primal description by "reversing" the coefficients of the feedback function and by using the first n output values for the initial state. Conversely, a primal description may be converted into a dual one by again reversing the feedback function and by "inverting" the sequence computation to obtain the initial state from the first n outputs. This last step is easy to perform by using the above expression $s_t = S^0_{(n-1)-t} + \sum_{i=0}^{t-1} \Phi^{(n-t)+i}\, s_i$, from which one gets $S^0_{(n-1)-t} = s_t - \sum_{i=0}^{t-1} \Phi^{(n-t)+i}\, s_i$, valid for $0 \leqslant t < n$.

Example 52. Let $\mathbb{F} = \mathbb{F}_2$, and consider the dual LFSR of size four with initial state $[S_3^0, \ldots, S_0^0] = [1, 0, 1, 1]$ and feedback function $\Phi^3 = \Phi^2 = 0$, $\Phi^1 = \Phi^0 = 1$. Then the first 20 terms $s_0, \ldots$ of the output sequence are: $[1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1]$. This same sequence may then be generated by a primal LFSR whose initial state is set to $[s_0, \ldots, s_3] = [1, 0, 1, 0]$ and with feedback function $\Phi^3 = \Phi^2 = 1$, $\Phi^1 = \Phi^0 = 0$, and indeed this is exactly how we have described it in the second part of Example 40.

Example 53. Let $\mathbb{F} = \mathbb{F}_2$, and consider the primal LFSR of size four with initial state $[S^0_3, \ldots, S^0_0] = [1, 1, 0, 0]$ and feedback function $\Phi^3 = \Phi^2 = \Phi^1 = \Phi^0 = 1$. Then the first 20 terms $s_0, \ldots$ of the output sequence are: $[1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0]$. This same sequence may then be generated by a dual LFSR with reversed (although in fact here identical) feedback function Φ and with initial state $S^0_3 = s_0 = 1$, $S^0_2 = s_0 + s_1 = 0$, $S^0_1 = s_0 + s_1 + s_2 = 0$, $S^0_0 = s_0 + s_1 + s_2 + s_3 = 0$.

To see why dual LFSRs are strongly connected to Definition 41, it is useful to first make the important observation that the process of computing S^{t+1} from S^t exactly corresponds to a multiplication by a constant in some modular polynomial ring. Interpreting S^t as a polynomial $\sum_{i=0}^{n-1} S^t_i X^i$ of $\mathbb{F}[X]$, one indeed has that $\Phi(S^t) = S^t_{n-1} \sum_{i=0}^{n-1} \Phi^i\, X^i$ is congruent to $S^t_{n-1}X^n$ modulo $X^n - \sum_{i=0}^{n-1} \Phi^i\, X^i$, and it follows that $S^tX = S^t_{n-1}X^n + \sum_{i=0}^{n-2} S^t_i X^{i+1}$ is congruent to $\Phi(S^t) + \sum_{i=0}^{n-2} S^t_i X^{i+1} = S^{t+1}$ modulo $X^n - \sum_{i=0}^{n-1} \Phi^i\, X^i$. In other words, letting $\Psi := X^n - \sum_{i=0}^{n-1} \Phi^i\, X^i$, one may identify S^t with $\varsigma\xi^t \in \mathbb{F}[X]/\langle\Psi\rangle$, for some initial state $\varsigma \cong S^0$, and where ξ is a root of Ψ and denotes the class of X in $\mathbb{F}[X]/\langle\Psi\rangle$. This finally makes the connection between LFSRs and Remark 34 which me made when studying the "`xtime`" operation that can be used to implement the arithmetic of extension fields.

The connection with Definition 41 now follows from the two easy facts that: 1) the minimal polynomial (over $\mathbb{F}$) of $\xi \in \mathbb{F}[X]/\langle\Psi\rangle$ divides Ψ, hence $S^{t+n} - \sum_{i=0}^{n-1} \Phi^i\, S^{t+i} \cong \varsigma\xi^t(\xi^n - \sum_{i=0}^{n-1} \Phi^i\, \xi^i) = 0$; 2) the previous equality carries over the output sequence, since its terms are obtained by applying an $\mathbb{F}$-linear form to the states S. In other words, Ψ annihilates (s_t); this leads to the following:

Proposition 54. *The linear complexity of the sequence generated by an LFSR of size n is at most n.*

We conclude this short presentation of LFSRs by discussing two points that highly benefit from the above interpretation of (dual) LFSRs.

Computing far-away terms of the output sequence. To compute an arbitrary term s_t of the output sequence, it is sufficient to compute the state S^t from which it is defined. Since the latter may be identified with $\varsigma\xi^t \in \mathbb{F}[X]/\langle\Psi\rangle$, one may compute it from one exponentiation and one multiplication in $\mathbb{F}[X]/\langle\Psi\rangle$, and this can be done efficiently using any suitable fast exponentiation algorithm.

Generating sequences of maximum period. Since the state S of an LFSR is finite, the output sequence (s_t) it generates is necessarily periodic after a finite rank, and the maximum period is equal to the number of non-zero states, *i.e.* $\#\mathbb{F}^n - 1$. Since reaching this maximum period means that all such states need to be "visited", one gets that the initial state S^0 does not influence the ability of an LFSR to generate such a sequence, apart from the obvious condition that it be not all zero. Now identifying again S_t with $\varsigma\xi^t \in \mathbb{F}[X]/\langle\Psi\rangle$, it is clear that a necessary condition for an LFSR to produce a sequence of maximum period is that ξ has multiplicative order $\#\mathbb{F}^n - 1$ in $\mathbb{F}[X]/\langle\Psi\rangle$. This is also in fact sufficient (as long as $\varsigma \neq 0$): we have seen above that one could recover the state of an LFSR from n consecutive terms of the output sequence, and it follows that the latter cannot have a period smaller than the one of the state, and thence the order of ξ.

4.3 Computing an annihilator of a finite sequence

We have just used in the previous section the fact that given a few consecutive output terms of the sequence generated by an LFSR, it is easy to recover its state. Consequently, if one additionally knows the feedback function, it is enough to observe a few terms to be able to generate all the following ones. In this section we will see that this requirement is in fact not necessary by showing that the feedback function can itself be reconstructed from sufficiently many terms of the sequence. This result has many applications, one of the most immediate ones being the computation of minimal polynomials: given for instance an element $\xi \in \mathbb{F}'$, its minimal polynomial over a subfield $\mathbb{F}$ of $\mathbb{F}'$ can be found as the feedback polynomial of the sequence (u_n), $u_n := \mathrm{Tr}_{\mathbb{F}'/\mathbb{F}}(\xi^n)$, where the *trace* function $\mathrm{Tr}_{\mathbb{F}'/\mathbb{F}}$ of $\mathbb{F}'$ onto $\mathbb{F}$ is (among other things) a morphism for the addition.[♯]

Given (s_t) the sequence generated by an LFSR of known size n, a simple approach to recover the feedback function Φ w.r.t. the primal definition is to solve the obvious underlying linear system. Indeed, considering $2n$ consecutive terms (w.l.o.g. $s_0, \ldots, s_{2n-1}$), one may write n equations $s_{i+n} = \sum_{j=0}^{n-1} \Phi_j\, s_{i+j}$, which is enough to recover the n unknown coefficients of Φ provided that the rank of the system is n.[♭] This can be summarised in matrix form by saying that all one needs to do is solving the system:

$$\begin{pmatrix}\Phi_{n-1} & \cdots & \Phi_0\end{pmatrix} \times \begin{pmatrix} s_{2n-2} & \cdots & s_{n-1} \\ \vdots & \ddots & \vdots \\ s_{n-1} & \cdots & s_0 \end{pmatrix} = \begin{pmatrix} s_{2n-1} & \cdots & s_n\end{pmatrix}.$$

While any general-purpose algorithm may be used for that, writing the system in this form however makes it obvious that it is highly structured. For instance, only $2n$ scalars are sufficient to describe it instead of $n^2 + n$ in general, and $n-1$ coefficients of one row of the matrix may be obtained from the row above (resp. below) by simply shifting it to the left (resp. right). It is thus reasonable to expect that such a system may be solved more efficiently than a generic one, and we will indeed sketch one algorithm that does so.

We now briefly describe the "Berlekamp-Massey" algorithm for computing an annihilator of a finite sequence. If applied to sufficiently many terms of the output sequence of an LFSR, the returned annihilator will then correspond to the feedback function w.r.t. the *dual* definition and then solve the above problem. Yet the algorithm is more general, in that it *always* returns an annihilator even when the sequence was not produced in such a way.

The algorithm starts from the observation that given a sequence (u_n), $u_0 \neq 0$, one may iteratively compute an annihilator A by "using" u_0 to cancel later non-zero terms.[♮] Suppose that u_{i_1} is the first non-zero term of (u_n) that follows u_0, then $u_{i_1} - (u_{i_1}u_0^{-1})u_0 = 0$, and so $-u_{i_1}u_0^{-1}X^{i_1} + 1$ annihilates (u_n) at rank i_1; more generally, the first $i_1 + 1$ terms of $(a_1 * u)$ are necessarily $u_0, 0, \ldots, 0$ by construction and the assumption on i_1.

If all the terms after index i_1 are also zero, A_1 annihilates the full sequence (except, unavoidably, the first term) and we are done. Otherwise let $(a_1 * u)_{i_2}$ be the first non-zero term, the objective is to find A_2 s.t. $(a_2 * u)_{i_2} = 0$. In that case this can be done effectively without "breaking" the annihilation of previous terms $< i_2$ by exploiting the linearity of the convolution and the fact that u_0 is still "available" to cancel further terms. For the sake of simplicity and w.l.o.g., we will now assume that the sequence has been normalised so that $u_0 = 1$; letting $A_1' = -(a * u)_{i_2}X^{i_2}$, $A_2 = A_1 + A_1'$, then $(a_2 * u) = (a_1' * u) + (a_1 * u)$ and by construction $(a_1 * u)_{0<i<i_2} = 0$, $(a_1' * u)_{<i_2} = 0$,[•] $(a_1' * u)_{i_2} = -(a * u)_{i_2}$ and so

[♯]If $[\mathbb{F}' : \mathbb{F}] = d \in \mathbb{N}$, a possible definition for $\mathrm{Tr}_{\mathbb{F}'/\mathbb{F}}$ is as $\mathrm{Tr}(\boldsymbol{M}_\xi^{\mathbb{F}})$, where $\boldsymbol{M}_\xi^{\mathbb{F}}$ is the matrix of multiplication by ξ over $\mathbb{F}$ w.r.t to any representation and basis, and Tr is here the usual trace function for matrices.

[♭]We will always make this assumption in the following, since otherwise (s_t) is all-zero after some rank, and then not very interesting.

[♮]If there are no such terms, the sequence is trivially annihilated by 1 after rank 1 and we are done.

[•]This is because $(a_1' * u)_i$ for $i < i_2$ is simply equal to a multiple of u_{i-i_2}, which is zero as per our convention.

$(a_2 * u)_{0<i\leqslant i_2} = 0$ as desired. Now again if $(a_2 * u)$ annihilates the full remaining sequence, we are done, otherwise one iterates the same process, with a twist.

The key observation, is that while it is always possible to use u_0 in the same fashion to annihilate every term except the first one, this may result in a trivial annihilator that is as long as the sequence itself. However, if one focuses only on annihilating terms "as far as possible" by using an annihilator "as short as possible" (*i.e.* of degree as small as possible), one may use a previous non-trivial (hopefully short) annihilator for part of the sequence and "shift" it so that it annihilates one more term. The general process is then the following: suppose that one has A_j, A_k s.t. $(a_j * u)_{i_j} = 0$, $(a_j * u)_{i_j+1} =: x_j \neq 0$, $(a_k * u)_{i_k} = 0$, $(a_k * u)_{i_k+1} =: x_k \neq 0$, $i_j < i_k$, $\deg(A_j) < \deg(A_k)$ (we know from the above that one can always use u_0 at various offsets to obtain such annihilators in the non-trivial cases). The goal is to find A_{k+1} of minimal degree s.t. $(a_{k+1} * u)_{i_{k+1}} = 0$. Then letting $A'_k := -x_k x_j^{-1} X^{k-j} A_j$, $A_{k+1} := A_k + A'_k$, by linearity of the convolution $(a_{k+1} * u)_{i_{k+1}} = 0$; one may then iterate this process until an annihilator for the full sequence is found.

For the annihilator produced by the following process to be the shortest possible (or *minimal*), one needs to specify how the "auxiliary" annihilator (A_i in the above) is updated through the execution of the algorithm. If it is never changed this corresponds to always using u_0 to construct the annihilator, and as we already mentioned this may result into annihilators as long as the sequence even when shorter ones may exist. One may show (but *we* will admit) that a change is necessary and sufficient every time there is a jump in the degree of the current annihilator; that is, if (using the above's notation) $\deg(A_{k+1} > \deg(A_k)$, A_k becomes the "new A_j", otherwise the latter is left unchanged. A possible proof is by induction, both on the degree of the thusly obtained annihilator and on a lower bound on the linear complexity of the sequence and showing their equality.

We conclude by briefly mentioning the cost of this algorithm. Every step of the above annihilates at least one term of the sequence. If ℓ is its (possibly unknown) linear complexity, a minimal annihilator will be found after annihilating at most 2ℓ terms,[**] so after at most 2ℓ steps. Since every step costs $O(\ell)$ arithmetic operations in $\mathbb{F}$, the total cost is $O(\ell^2)$, which is then indeed better than a general-purpose resolution of a linear system. There is also an added benefit over the latter when (as often in applications) only an upper-bound n on ℓ is known: here the cost depends quadratically only on (the possibly smaller) ℓ, plus an unavoidable term linear in n to check that the annihilator is indeed one for $2n$ terms, whereas general-purpose linear algebra would either need to solve one size-n system with cost $O(n^\omega)$ (where ω is a feasible exponent for matrix multiplication), or ℓ systems up to size ℓ (and up to ℓ additional verifications) for a cost of $O(\ell^{\omega+1} + \ell n)$.

[**]This is in the case where the sequence was produced by an LFSR of size ℓ; more terms may be necessary in the presence of an arbitrary prefix.