Structured FFT and TFT: symmetric and lattice polynomials

Joris van der Hoeven*
Laboratoire d'informatique
UMR 7161 CNRS
École polytechnique
91128 Palaiseau Cedex, France
vdhoeven@lix.polytechnique.fr

Romain Lebreton
LIRMM
UMR 5506 CNRS
Université de Montpellier II
Montpellier, France
lebreton@lirmm.fr

Éric Schost
Computer Science Department
Western University
London, Ontario
Canada
eschost@uwo.ca

ABSTRACT

In this paper, we consider the problem of efficient computations with structured polynomials. We provide complexity results for computing Fourier Transform and Truncated Fourier Transform of symmetric polynomials, and for multiplying polynomials supported on a lattice.

1. INTRODUCTION

Fast computations with multivariate polynomials and power series have been of fundamental importance since the early ages of computer algebra. The representation is an important issue which conditions the performance in an intrinsic way; see [23, 31, 8] for some historical references.

It is customary to distinguish three main types of representations: dense, sparse, and functional. A *dense representation* is made of a compact description of the support of the polynomial and the sequence of its coefficients. The main example concerns *block supports* – it suffices to store the coordinates of two opposite vertices. In a dense representation all the coefficients of the considered support are stored, even if they are zero. If a polynomial has only a few non-zero terms in its bounding block, we shall prefer to use a *sparse representation* which stores only the sequence of the non-zero terms as pairs of monomials and coefficients. Finally, a *functional representation* stores a function that can produce values of the polynomials at any given point. This can be a pure blackbox (which means that its internal structure is not supposed to be known) or a specific data structure such as *straight-line programs* (see Chapter 4 of [4], for instance, and [10] for a concrete library for the manipulation of polynomials with a functional representation).

For dense representations with block supports, it is classical that the algorithms used for the univariate case can be naturally extended: the naive algorithm, Karatsuba's algorithm, and even Fast Fourier Transforms [7, 30, 6, 28, 16] can be applied recursively in each variable, with good performance. Another classical approach is the Kronecker substitution which reduces the multivariate product to one variable only; for all these questions, we refer the reader to classical books such as [29, 14]. When the number of variables is fixed and the partial degrees tend to infinity, these techniques lead to softly linear costs.

After the discovery of sparse interpolation [2, 24, 25, 26, 11], probabilistic algorithms with a quasi-linear complexity have been developed for sparse polynomial multiplication [5]. It has recently be shown that such asymptotically fast algorithms may indeed become more efficient than naive sparse multiplication [19].

In practice however, it frequently happens that multivariate polynomials with a dense flavor do not admit a block support. For instance, it is common to consider polynomials of a given total degree. In a recent series of works [15, 17, 18, 21, 20], we have studied the complexity of polynomial multiplication in this "semi-dense" setting. In the case when the supports of the polynomials are initial segments for the partial ordering on $\mathbb{N}^n$, the truncated Fourier transform is a useful device for the design of efficient algorithms.

Besides polynomials with supports of a special kind, we may also consider what will call "structured polynomials". By analogy with linear algebra, such polynomials carry a special structure which might be exploited for the design of more efficient algorithms. In this paper, we turn our attention to a first important example of this kind: polynomials which are invariant under the action of certain matrix groups. We consider only two special cases: finite subgroups of $\mathfrak{S}_n$ and finite groups of diagonal matrices. These cases are already sufficient to address questions raised *e.g.* in celestial mechanics [12]; it is hoped that more general groups can be dealt with using similar ideas.

In the limited scope of this paper, our main objective is to prove complexity results that demonstrate the savings induced by a proper use of the symmetries. Our complexity analyses take into account the number of arithmetic operations in the base field, and as often, we consider that operations with groups and lattices take a constant number of operations. A serious implementation of our algorithms would require an improved study of these aspects.

Of course, there already exists an abundant body of work on some of these questions. *Crystallographic* FFT algorithms date back to [32], with contributions as recent as [27], but are dedicated to crystallographic symmetries. A more general framework due to [1] was recently revisited under the point of view of high-performance code generation [22]; our treatment of permutation groups is in a similar spirit, but to our understanding, these previous papers do not prove results such as those we give below (and they only consider the FFT, not its truncated version).

Similarly, our results on diagonal groups, which fit in the general context of FFTs over lattices, use similar techniques as in a series of papers initiated by [13] and continued as recently as [33, 3], but the actual results we prove are not in those references.

*. This work has been partly supported by the Digiteo 2009-36HD grant of the Rgion Ile-de-France, NSERC and the CRC program.

2. THE CLASSICAL FFT

2.1 Notation

We will work over an effective base field (or ring) $\mathbb{K}$ with sufficiently many roots of unity; the main objects are polynomials $\mathbb{K}[\boldsymbol{x}] := \mathbb{K}[x_1, ..., x_n]$ in n variables over $\mathbb{K}$. For any $\boldsymbol{k} = (\boldsymbol{k}_1, ..., \boldsymbol{k}_n) \in \mathbb{N}^n$ and $P \in \mathbb{K}[\boldsymbol{x}]$, we denote by $\boldsymbol{x}^{\boldsymbol{k}}$ the monomial $x_1^{\boldsymbol{k}_1} \cdots x_n^{\boldsymbol{k}_n}$ and by $P_{\boldsymbol{k}}$ the coefficient of P in $\boldsymbol{x}^{\boldsymbol{k}}$. The support $\mathrm{supp}\,(P)$ of a polynomial $P \in \mathbb{K}[\boldsymbol{x}]$ is the set of exponents $\boldsymbol{k} \in \mathbb{N}^n$ such that $P_{\boldsymbol{k}} \neq 0$.

For any subset S of $\mathbb{N}^n$, we define $\mathbb{K}[\boldsymbol{x}]_S$ as the polynomials with support included in S. As an important special case, when $S = \mathbb{N}_d^n := \{0, ..., d-1\}^n$, we denote by $\mathbb{K}[\boldsymbol{x}]_d := \mathbb{K}[\boldsymbol{x}]_S$ the set of polynomials with partial degree less than d in all variables.

Let $\omega \in \mathbb{K}$ be a primitive dth root of unity (in Section 4, it will be convenient to write this root $\mathrm{e}^{2\pi \mathrm{i}/d}$). For any $\boldsymbol{k} = (\boldsymbol{k}_1, ..., \boldsymbol{k}_n) \in \mathbb{N}^n$, we define $\omega^{\boldsymbol{i}} = (\omega^{\boldsymbol{k}_1}, ..., \omega_d^{\boldsymbol{k}_n})$. One of our aims is to compute efficiently the map

$$\mathrm{FFT}_\omega : \left\{ \begin{array}{ccc} \mathbb{K}[\boldsymbol{x}]_d & \longrightarrow & \mathbb{K}^{\mathbb{N}_d^n} \\ P & \longmapsto & (P(\omega^{\boldsymbol{i}}))_{\boldsymbol{i} \in \mathbb{N}_d^n} \end{array} \right.$$

when P is a "structured polynomial". Here, $\mathbb{K}^{\mathbb{N}_d^n}$ denotes the set of vectors with entries in $\mathbb{K}$ indexed by $\mathbb{N}_d^n$; in other words, $\boldsymbol{v} \in \mathbb{K}^{\mathbb{N}_d^n}$ implies that $\boldsymbol{v}_{\boldsymbol{i}} \in \mathbb{K}$ for all $\boldsymbol{i} \in \mathbb{N}_d^n$. Likewise, $\mathbb{K}^{\mathbb{N}_d^n \times \mathbb{N}_d^n}$ denotes the set of matrices with indices in $\mathbb{N}_d^n$, that is, $\mathbf{M} \in \mathbb{K}^{\mathbb{N}_d^n \times \mathbb{N}_d^n}$ implies that $\mathbf{M}_{\boldsymbol{i},\boldsymbol{j}} \in \mathbb{K}$ for all $\boldsymbol{i}, \boldsymbol{j} \in \mathbb{N}_d^n$.

Most of the time, we will take $d = 2^\ell$ with $\ell \in \mathbb{N}$, although we will also need more general d of mixed radices $d = p_1 \cdots p_\ell$.

We denote by $\boldsymbol{i} \cdot \boldsymbol{j}$ the inner product of two vectors in $\mathbb{N}^n$. We also let $\boldsymbol{e}_k := (0, \overset{k-1}{\ldots}, 0, 1, 0, ..., 0)$, for $1 \leqslant k \leqslant n$, be the kth element of the canonical basis of $\mathbb{K}^n$. At last, for $\ell \in \mathbb{N}$ and $k \in \mathbb{N}_{2^\ell}$ we denote by $[k]_\ell$ the bit reversal of k in length n and we extend this notation to vectors by $[\boldsymbol{k}]_s = ([\boldsymbol{k}_1]_s, ..., [\boldsymbol{k}_n]_s)$.

2.2 The classical multivariate FFT

Let us first consider the in-place computation of a full n-dimensional FFT of length $d = 2^\ell$ in each variable. We first recall the notations and operations of the decimation in time variant of the FFT, and we refer the reader to [9] for more details. In what follows, ω is a dth primitive root of unity.

We start with the FFT of a univariate polynomial $P \in \mathbb{K}[x]_d$. Decimation in time amounts to decomposing P into its even and odd parts, and proceeding recursively, by means of ℓ decimations applied to the variable x. For $0 \leqslant k < d$, we will write $\boldsymbol{c}_k^0 = P_k$ for the input and denote by $\boldsymbol{c}^s = (\boldsymbol{c}_k^s)_{0 \leqslant k < d}$ the result after s decimation steps, for $s \in \{1, ..., \ell\}$.

At stage s, the decimation is computed using butterflies of span $\delta := 2^{\ell - s}$. If $i \in \mathbb{N}_{2^s}$ is even and j belongs to $\mathbb{N}_\delta$ then these butterflies are given by

$$\begin{pmatrix} \boldsymbol{c}_{i\delta+j}^s \\ \boldsymbol{c}_{(i+1)\delta+j}^s \end{pmatrix} = \begin{pmatrix} 1 & \omega^{[i]_s \delta} \\ 1 & -\omega^{[i]_s \delta} \end{pmatrix} \begin{pmatrix} \boldsymbol{c}_{i\delta+j}^{s-1} \\ \boldsymbol{c}_{(i+1)\delta+j}^{s-1} \end{pmatrix}.$$

Putting the coefficients of all these linear relations in a matrix $\mathbf{B}^s \in \mathbb{K}^{\mathbb{N}_d \times \mathbb{N}_d}$, we get $\boldsymbol{c}^s = \mathbf{B}^s \, \boldsymbol{c}^{s-1}$. The matrix $\mathbf{B}^s$ is sparse, with at most two non-zero coefficients on each row and each column; up to permutation, it is block-diagonal, with blocks of size 2. After ℓ stages, we get the evaluations of P in the bit reversed order:

$$\boldsymbol{c}_k^\ell = P(\omega^{[k]_\ell}).$$

We now adapt this notation to the multivariate case. The computation is still divided in ℓ stages, each stage doing one decimation in every variable x_1, ..., x_n. Therefore, we will denote by $\boldsymbol{c}_{\boldsymbol{k}}^0 := P_{\boldsymbol{k}}$ the coefficients of the input for $\boldsymbol{k} \in \mathbb{N}_d$ and by $\boldsymbol{c}_{\boldsymbol{k}}^{s,t}$ the coefficients obtained during the sth stage, after the decimations in $x_1, ..., x_t$ are done, with $s \in \{1, ..., \ell\}$ and $t \in \{0, ..., n\}$, so that $\boldsymbol{c}_{\boldsymbol{k}}^{s-1,n} = \boldsymbol{c}_{\boldsymbol{k}}^{s,0}$. We abbreviate $\boldsymbol{c}_{\boldsymbol{k}}^{s-1} := \boldsymbol{c}_{\boldsymbol{k}}^{s,0}$ for the coefficients after $s-1$ stages.

The intermediate coefficients $\boldsymbol{c}_{\boldsymbol{k}}^s$ can be seen as evaluations of intermediate polynomials: for every $s \in \{0, ..., \ell\}$, $\boldsymbol{i} \in \mathbb{N}_{2^s}^n$ and $\boldsymbol{j} \in \mathbb{N}_\delta^n$, one has

$$\boldsymbol{c}_{\boldsymbol{i}\delta+\boldsymbol{j}}^s \;=\; P_{\boldsymbol{j}}^s((\omega^\delta)^{[\boldsymbol{i}]_s}) \tag{1}$$

where $P_{\boldsymbol{j}}^s = \sum_{\boldsymbol{i}' \in \mathbb{N}_{2^s}^n} P_{\boldsymbol{i}'\delta+\boldsymbol{j}} \, \boldsymbol{x}^{\boldsymbol{i}'}$ are obtained through an s-fold decimation of P. Equivalently, the coefficients $\boldsymbol{c}_{\boldsymbol{k}}^s$ satisfy

$$\boldsymbol{c}_{\boldsymbol{i}\delta+\boldsymbol{j}}^s \;=\; \sum_{\boldsymbol{i}' \in \mathbb{N}_{2^s}^n} P_{\boldsymbol{i}'\delta+\boldsymbol{j}} \, \omega^{[\boldsymbol{i}]_s \cdot \boldsymbol{i}'\delta}. \tag{2}$$

Thus, $\boldsymbol{c}_{\boldsymbol{j}}^\ell = P(\omega^{[\boldsymbol{j}]_\ell})$ yields $\mathrm{FFT}_\omega(P)$ in bit reversed order.

For the concrete computation of the coefficients $\boldsymbol{c}_{\boldsymbol{k}}^s$ at stage s from the coefficients $\boldsymbol{c}_{\boldsymbol{k}}^{s-1}$ at stage $s-1$, we use n so called "elementary transforms" with respect to each of the variables x_1, ..., x_n. For any $t \in \{1, ..., n\}$, the coefficients $\boldsymbol{c}_{\boldsymbol{k}}^{s,t}$ are obtained from the coefficients $\boldsymbol{c}_{\boldsymbol{k}}^{s,t-1}$ through butterflies of span $\delta = 2^{\ell-s}$ with respect to the variable x_t; this can be rewritten by means of the formula

$$\begin{pmatrix} \boldsymbol{c}_{\boldsymbol{i}\delta+\boldsymbol{j}}^{s,t} \\ \boldsymbol{c}_{(\boldsymbol{i}+\boldsymbol{e}_t)\delta+\boldsymbol{j}}^{s,t} \end{pmatrix} = \begin{pmatrix} 1 & \omega^{[\boldsymbol{i}_t]_s \delta} \\ 1 & -\omega^{[\boldsymbol{i}_t]_s \delta} \end{pmatrix} \begin{pmatrix} \boldsymbol{c}_{\boldsymbol{i}\delta+\boldsymbol{j}}^{s,t-1} \\ \boldsymbol{c}_{(\boldsymbol{i}+\boldsymbol{e}_t)\delta+\boldsymbol{j}}^{s,t-1} \end{pmatrix} \tag{3}$$

for any $\boldsymbol{i} \in \mathbb{N}_{2^s}^n$ with even t-th coordinate i_t and $\boldsymbol{j} \in \mathbb{N}_\delta^n$. Equation (3) can be rewritten more compactly in matrix form $\boldsymbol{c}^{s,t} = \mathbf{B}^{s,t} \, \boldsymbol{c}^{s,t-1}$ where $\mathbf{B}^{s,t}$ is a sparse matrix in $\mathbb{K}^{\mathbb{N}_d^n \times \mathbb{N}_d^n}$ which is naturally indexed by pairs of multi-indices in $\mathbb{N}_d^n$. Setting $\mathbf{B}^s = \mathbf{B}^{s,n} \cdots \mathbf{B}^{s,1} \in \mathbb{K}^{\mathbb{N}_d^n \times \mathbb{N}_d^n}$, we also obtain a short formula for $\boldsymbol{c}^s$ as a function of $\boldsymbol{c}^{s-1}$:

$$\boldsymbol{c}^s \;=\; \mathbf{B}^s \, \boldsymbol{c}^{s-1} \tag{4}$$

Remark that the matrices $\mathbf{B}^{s,1}, ..., \mathbf{B}^{s,n}$ commute pairwise. Notice also that each of the rows and columns of $\mathbf{B}^{s,t}$ has at most 2 non zero entries and consequently those of $\mathbf{B}^s$ contains at most 2^n non zero entries. For this reason, we can apply the matrices $\mathbf{B}^{s,t}$ (resp. $\mathbf{B}^s$) within $\mathcal{O}(|\mathbb{N}_d^n|) = \mathcal{O}(d^n)$ (resp. $\mathcal{O}(n\, d^n)$) operations in $\mathbb{K}$. Finally, the full n-dimensional FFT of length $d = 2^\ell$ costs $\mathsf{F}(d, n) := 3/2 \, \ell \, n \, d^n = 3/2 \, d^n \log(d^n)$ operations in $\mathbb{K}$ (see [14, Section 8.2]).

Example 1. Let us make these matrices explicit for polynomials in $n = 2$ variables and degree less than $d = 4$, so that $\ell = 2$. We start with the first decimation in x_1 whose butterflies of span δ are captured by $\mathbf{B}^{s,t}$ with $\delta = 2^{\ell-s} = 2^1$, $s = 1$ and $t = 1$. It takes as input $\boldsymbol{c}_{\boldsymbol{k}}^0 = \boldsymbol{c}_{\boldsymbol{k}}^{1,0} := P_{\boldsymbol{k}}$ and outputs $\boldsymbol{c}_{\boldsymbol{k}}^{1,1}$ for $\boldsymbol{k} \in \mathbb{N}_4^2 = \{0, ..., 3\}^2$. For any $\boldsymbol{j}_1 \in \mathbb{N}_2 = \{0, 1\}$ and $\boldsymbol{k}_2 \in \{0, ..., 3\}$, we let

$$\begin{pmatrix} \boldsymbol{c}_{(\boldsymbol{j}_1, \boldsymbol{k}_2)}^{1,1} \\ \boldsymbol{c}_{(2+\boldsymbol{j}_1, \boldsymbol{k}_2)}^{1,1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} \boldsymbol{c}_{(\boldsymbol{j}_1, \boldsymbol{k}_2)}^{1,0} \\ \boldsymbol{c}_{(2+\boldsymbol{j}_1, \boldsymbol{k}_2)}^{1,0} \end{pmatrix}.$$

So, $\mathbf{B}^{1,1}$ is a 16 by 16 matrix, which is made of diagonal blocks of $\left(\begin{smallmatrix}1 & 1\\ 1 & -1\end{smallmatrix}\right)$ in a suitable basis. The decimation in x_2 during stage 1 is similar :

$$\begin{pmatrix} \boldsymbol{c}^{1,2}_{(\boldsymbol{k}_1,\boldsymbol{j}_2)} \\ \boldsymbol{c}^{1,2}_{(\boldsymbol{k}_1,2+\boldsymbol{j}_2)} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} \boldsymbol{c}^{1,1}_{(\boldsymbol{k}_1,\boldsymbol{j}_2)} \\ \boldsymbol{c}^{1,1}_{(\boldsymbol{k}_1,2+\boldsymbol{j}_2)} \end{pmatrix}$$

for $\boldsymbol{j}_2 \in \mathbb{N}_2$ and $\boldsymbol{k}_1 \in \mathbb{N}_4$. Consequently, $\mathbf{B}^{1,2}$ is made of diagonal blocks $\left(\begin{smallmatrix}1 & 1\\ 1 & -1\end{smallmatrix}\right)$ (in another basis than $\mathbf{B}^{1,1}$). Their product $\mathbf{B}^1 := \mathbf{B}^{1,2}\,\mathbf{B}^{1,1}$ corresponds to the operations

$$\begin{pmatrix} \boldsymbol{c}^{1}_{(\boldsymbol{j}_1,\boldsymbol{j}_2)} \\ \boldsymbol{c}^{1}_{(2+\boldsymbol{j}_1,\boldsymbol{j}_2)} \\ \boldsymbol{c}^{1}_{(\boldsymbol{j}_1,2+\boldsymbol{j}_2)} \\ \boldsymbol{c}^{1}_{(2+\boldsymbol{j}_1,2+\boldsymbol{j}_2)} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{c}^{0}_{(\boldsymbol{j}_1,\boldsymbol{j}_2)} \\ \boldsymbol{c}^{0}_{(2+\boldsymbol{j}_1,\boldsymbol{j}_2)} \\ \boldsymbol{c}^{0}_{(\boldsymbol{j}_1,2+\boldsymbol{j}_2)} \\ \boldsymbol{c}^{0}_{(2+\boldsymbol{j}_1,2+\boldsymbol{j}_2)} \end{pmatrix}$$

for $\boldsymbol{j}_1, \boldsymbol{j}_2 \in \mathbb{N}_2$. Thus $\mathbf{B}^1$ is made of such 4 by 4 matrices on the diagonal (in yet another basis). Note that in general, the matrices $\mathbf{B}^{s,t}$ and $\mathbf{B}^s$ are still made of diagonal blocks, but these blocks vary along the diagonal.

We can sum it all up in the two following algorithms.

Algorithm Butterfly
INPUT: n, degree d, stage s, index $(2\,\boldsymbol{i}', \boldsymbol{j})$ of the butterfly with $\boldsymbol{i}' \in \mathbb{N}^n_{2^{s-1}}$, $\boldsymbol{j} \in \mathbb{N}^n_\delta$, root of unity ω and coefficients $\boldsymbol{c} \in \mathbb{K}^{\mathbb{N}^n_2}$ for the butterfly ($\boldsymbol{c}_{\boldsymbol{b}} := \boldsymbol{c}^{s-1}_{(2\boldsymbol{i}'+\boldsymbol{b})\delta+\boldsymbol{j}}$ for $\boldsymbol{b} \in \mathbb{N}^n_2$).
OUTPUT: the output of the butterfly $\boldsymbol{d} \in \mathbb{K}^{\mathbb{N}^n_2}$

For $t = 1, ..., n$ do //decimation in x_t
  For $\boldsymbol{b} \in \mathbb{N}^n_2 = \{0,1\}^n$ such that $\boldsymbol{b}_t = 0$ do
    $r = \omega^{[2\boldsymbol{i}'_t]_s \delta}\, \boldsymbol{c}_{\boldsymbol{b}+\boldsymbol{e}_t}$ //butterfly in one variable
    $\boldsymbol{d}_{\boldsymbol{b}} = \boldsymbol{c}_{\boldsymbol{b}} + r$
    $\boldsymbol{d}_{\boldsymbol{b}+\boldsymbol{e}_t} = \boldsymbol{c}_{\boldsymbol{b}} - r$
Return $\boldsymbol{d}_{\boldsymbol{b}}$

Algorithm FFT
INPUT: n, degree d, ω and coefficients $\boldsymbol{c}^0 \in \mathbb{K}^{\mathbb{N}^n_d}$ of P
OUTPUT: coefficients $\boldsymbol{c}^\ell \in \mathbb{K}^{\mathbb{N}^n_d}$ in bit reversed order

For $s = 1, ..., \ell$ do //stage s
  $\delta := d/2^s$
  For $\boldsymbol{i}' \in \mathbb{N}^n_{2^{s-1}}$, $\boldsymbol{j} \in \mathbb{N}^n_{2^{\ell-s}}$ do //pick a butterfly
    For $\boldsymbol{b} \in \mathbb{N}^n_2$ do $\boldsymbol{c}_{\boldsymbol{b}} := \boldsymbol{c}^{s-1}_{(2\boldsymbol{i}'+\boldsymbol{b})\delta+\boldsymbol{j}}$
    $\boldsymbol{d}_{\boldsymbol{b}} = \mathbf{Butterfly}(n, d, s, \boldsymbol{i}', \boldsymbol{j}, \boldsymbol{c}_{\boldsymbol{b}})$
    For $\boldsymbol{b} \in \mathbb{N}^n_2$ do $\boldsymbol{c}^s_{(2\boldsymbol{i}'+\boldsymbol{b})\delta+\boldsymbol{j}} := \boldsymbol{d}_{\boldsymbol{b}}$
Return $\boldsymbol{c}^\ell$

In the last section, we will also use the ring isomorphism

$$\begin{array}{ccc} \mathbb{K}[\boldsymbol{x}]_d/(x_1^d - 1, ..., x_n^d - 1) & \to & [\mathbb{K}[\boldsymbol{x}]_\delta/(x_1^\delta - 1, ..., x_n^\delta - 1)]^{2^{ns}} \\ P & \mapsto & Q^s = (Q^s_{\boldsymbol{i}}(\boldsymbol{x}))_{\boldsymbol{i} \in \mathbb{N}^n_{2^s}} \end{array}$$

with $Q^s_{\boldsymbol{i}}(\boldsymbol{x}) = \sum_{\boldsymbol{j} \in \mathbb{N}^n_\delta} \left(\sum_{\boldsymbol{i}' \in \mathbb{N}^n_{2^s}} P_{\boldsymbol{i}'\delta+\boldsymbol{j}}\, \boldsymbol{\omega}^{\boldsymbol{i} \cdot \boldsymbol{i}'\delta} \right) (\boldsymbol{\omega}^{\boldsymbol{i}}\, \boldsymbol{x})^{\boldsymbol{j}}$ for $\boldsymbol{i} \in \mathbb{N}^n_{2^s}$, and $(\boldsymbol{\omega}^{\boldsymbol{i}}\, \boldsymbol{x})^{\boldsymbol{j}} = (\omega^{\boldsymbol{i}_1}\, x_1)^{\boldsymbol{j}_1} \cdots (\omega^{\boldsymbol{i}_n}\, x_n)^{\boldsymbol{j}_n}$. These polynomials generalize the decomposition of a univariate polynomials P into $P \bmod (x^{d/2} - 1)$ and $P \bmod (x^{d/2} + 1)$. We could obtain them through a decimation in frequency, but it is enough to remark that we can reconstruct them from the coefficients $\boldsymbol{c}^s$ thanks to the formula

$$Q^s_{\boldsymbol{i}}(\boldsymbol{x}) = \sum_{\boldsymbol{j} \in \mathbb{N}^n_\delta} \boldsymbol{c}^s_{[\boldsymbol{i}]_s \delta + \boldsymbol{j}}\, (\boldsymbol{\omega}^{\boldsymbol{i}}\, \boldsymbol{x})^{\boldsymbol{j}}. \tag{5}$$

3. THE SYMMETRIC FFT

In this section, we let $G \subseteq \mathfrak{S}_n$ be a permutation group. The group G acts on the polynomials $\mathbb{K}[\boldsymbol{x}]$ *via* the map

$$\varphi_g : \begin{cases} \mathbb{K}[\boldsymbol{x}] & \longrightarrow & \mathbb{K}[\boldsymbol{x}] \\ P(x_1, ..., x_n) & \longmapsto & P^g(\boldsymbol{x}) := P(x_{g(1)}, ..., x_{g(n)}) \end{cases}.$$

We denote by $\mathbb{K}[\boldsymbol{x}]^G := \mathrm{stab}_G\, \mathbb{K}[\boldsymbol{x}]$ the set of polynomials invariant under the action of G. Our main result here is that one can save a factor (roughly) $|G|$ when computing the FFT of an invariant polynomial. Our approach is in the same spirit as the one in [1], where similar statements on the savings induced by (very general) symmetries can be found. However, we are not aware of published results similar to Theorem 7 below.

3.1 The bivariate case

Let $P \in \mathbb{K}[x_1, x_2]$ be a symmetric bivariate polynomial of partial degrees less than $d = 8$, so that $\ell = 3$; let also $\omega \in \mathbb{K}$ be a primitive eighth root of unity. We detail on this easy example the methods used to exploit the symmetries to decrease the cost of the FFT.

The coefficients of P are placed on a 8×8 grid. The bivariate classical FFT consists in the application of butterflies of size $\delta := 2^{\ell-s}$ for s from 1 to 3, as in Figure 1. When some butterflies overlap, we draw only the shades of all but one of them. The result of stage $s = 3$ is the set of evaluations $(P(\omega^{\boldsymbol{i}_1}, \omega^{\boldsymbol{i}_2}))_{\boldsymbol{i} \in \mathbb{N}^2_8}$ in bit reversed order.

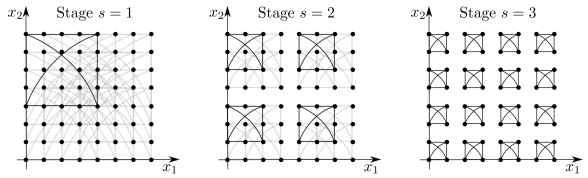

Figure 1. Classical bivariate FFT

We will remark below that each stage s preserves the symmetry of the input. In particular, since our polynomial P is symmetric, the coefficients $\boldsymbol{c}^1_{\boldsymbol{k}}$ at stage 1 are symmetric too, so we only need to compute the coefficients $\boldsymbol{c}^1_{(\boldsymbol{k}_1,\boldsymbol{k}_2)}$ for $(\boldsymbol{k}_1, \boldsymbol{k}_2)$ in the *fundamental domain* (sometimes called asymmetric unit) $0 \leqslant k_2 \leqslant k_1 < 8$. We choose to compute at stage s only the butterflies which involves at least an element of F; the set of indices that are included in a butterfly of span $\delta = 2^{\ell-s}$ that meets F will be called the extension F_δ of F.

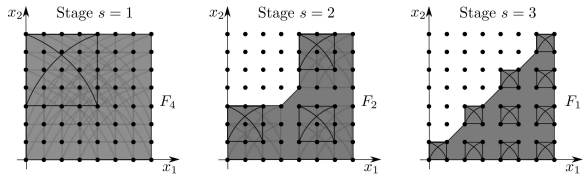

Figure 2. Symmetric bivariate FFT

Every butterfly of stage 1 meets the fundamental domain, so we do not save anything there. However, we save 4 out of 16 butterflies at stage 2 and 6 out of 16 butterflies at the third stage. Asymptotically in d, we gain a factor two in the number of arithmetic operations in $\mathbb{K}$ compared to the classical FFT; this corresponds to the order of the group.

3.2 Notation

The group G also acts on $\mathbb{N}^n$ with $g(\boldsymbol{i}) = (\boldsymbol{i}_{g(1)}, ..., \boldsymbol{i}_{g(n)})$ for any $\boldsymbol{i} \in \mathbb{N}^n$. This action is consistent with the action on polynomials since $\varphi_g(\boldsymbol{x}^{\boldsymbol{k}}) = \boldsymbol{x}^{g^{-1}(\boldsymbol{k})}$.

Because G acts on polynomials, it acts on the vector of its coefficients. More generally, if $I \subseteq \mathbb{N}^n$ and $\boldsymbol{v} \in \mathbb{K}^{G(I)}$, we denote by $\boldsymbol{v}^g \in \mathbb{K}^I$ the vector defined by $\boldsymbol{v}^g_{\boldsymbol{i}} = \boldsymbol{v}_{g(\boldsymbol{i})}$ for all $\boldsymbol{i} \in I$. If $I \subseteq \mathbb{N}^n$ is stable by G, *i.e.* $G(I) \subseteq I$, we define the set of invariant vectors $\mathbb{K}^{I,G}$ by $\mathbb{K}^{I,G} := \{\boldsymbol{v} \in \mathbb{K}^I \colon \forall g \in G, \boldsymbol{v}^g = \boldsymbol{v}\}$.

For any two sets I, J satisfying $J \subseteq I \subseteq \mathbb{N}^n$, we define the restriction $\boldsymbol{v}_J \in \mathbb{K}^J$ of $\boldsymbol{v} \in \mathbb{K}^I$ by $(\boldsymbol{v}_J)_{\boldsymbol{j}} = \boldsymbol{v}_{\boldsymbol{j}}$ for all $\boldsymbol{j} \in J$. Recall the definition of the lexicographical order $\leqslant_{\text{lex}}$: for all $\boldsymbol{i}, \boldsymbol{j} \in \mathbb{N}^n$, $\boldsymbol{i} <_{\text{lex}} \boldsymbol{j}$ if there exists $k \in \{0, ..., n-1\}$ such that $\boldsymbol{i}_t = \boldsymbol{j}_t$ for any $1 \leqslant t \leqslant k$ and $\boldsymbol{i}_{k+1} < \boldsymbol{j}_{k+1}$. Finally for any subset $S \subseteq \mathbb{N}^n$ stable by G, we define a fundamental domain S^G of the action of G on S by $S^G := \{\boldsymbol{i} \in S \colon \forall g \in G, \boldsymbol{i} \geqslant_{\text{lex}} g(\boldsymbol{i})\}$ together with the projection $\pi^G \colon \mathbb{N}^n \to \mathbb{N}^n$ such as $\{\pi^G(\boldsymbol{i})\} := G(\{\boldsymbol{i}\}) \cap (\mathbb{N}^n)^G$.

3.3 Fundamental domains

Let $F = F^{[d]} := (\mathbb{N}^n_d)^G$ be the fundamental domain associated to the action of G on $\mathbb{N}^n_d$. Any G-symmetric vector $\boldsymbol{c} \in \mathbb{K}^{\mathbb{N}^n_d}$ can be reconstructed from its restriction $\boldsymbol{c}_F \in \mathbb{K}^F$ using the formula $\boldsymbol{c}_{\boldsymbol{i}} = (\boldsymbol{c}_F)_{\pi^G(\boldsymbol{i})}$. As it turns out, the in-place FFT algorithm from Section 2.2 admits the important property that the coefficients at all stage are still G-symmetric.

LEMMA 2. *Let $P \in \mathbb{K}[\boldsymbol{x}]_d$ and the vectors $\boldsymbol{c}^0, ..., \boldsymbol{c}^\ell \in \mathbb{K}^{\mathbb{N}^n_d}$ be as in Section 2.2. Then if $P \in \mathbb{K}[\boldsymbol{x}]^G_d$, $\boldsymbol{c}^0, ..., \boldsymbol{c}^\ell \in \mathbb{K}^{\mathbb{N}^n_d, G}$.*

PROOF. Given $P \in \mathbb{K}[\boldsymbol{x}]^G_d$, $\boldsymbol{k} \in \mathbb{N}^n_d$ and $g \in G$, we clearly have $\boldsymbol{c}^0_{g(\boldsymbol{k})} = \boldsymbol{c}^0_{\boldsymbol{k}}$. For any $s \in \{0, ..., \ell\}$, $\boldsymbol{i}, \boldsymbol{i}' \in \mathbb{N}^n_{2^s}$, $\boldsymbol{j} \in \mathbb{N}^n_{2^{\ell-s}}$ and $g \in G \subseteq \mathfrak{S}_n$, we also notice that

$$\begin{aligned} g(\boldsymbol{i}\, 2^{\ell-s} + \boldsymbol{j}) &= g(\boldsymbol{i})\, 2^{\ell-s} + g(\boldsymbol{j}) \\ [g(\boldsymbol{i})]_s &= g([\boldsymbol{i}]_s) \\ g(\boldsymbol{i}) \cdot g(\boldsymbol{i}') &= \boldsymbol{i} \cdot \boldsymbol{i}'. \end{aligned}$$

Hence, using Equation (2), we get

$$\begin{aligned} \boldsymbol{c}^s_{g(\boldsymbol{i}2^{\ell-s}+\boldsymbol{j})} &= \boldsymbol{c}^s_{g(\boldsymbol{i})2^{\ell-s}+g(\boldsymbol{j})} \\ &= \sum_{\boldsymbol{i}' \in \mathbb{N}^n_{2^s}} P_{\boldsymbol{i}'2^{\ell-s}+g(\boldsymbol{j})}\, \omega^{[g(\boldsymbol{i})]_s \cdot \boldsymbol{i}' 2^{\ell-s}} \\ &= \sum_{\boldsymbol{i}' \in \mathbb{N}^n_{2^s}} P_{g(\boldsymbol{i}')2^{\ell-s}+g(\boldsymbol{j})}\, \omega^{[g(\boldsymbol{i})]_s \cdot g(\boldsymbol{i}') 2^{\ell-s}} \\ &= \sum_{\boldsymbol{i}' \in \mathbb{N}^n_{2^s}} P_{g(\boldsymbol{i}'2^{\ell-s}+\boldsymbol{j})}\, \omega^{g([\boldsymbol{i}]_s) \cdot g(\boldsymbol{i}') 2^{\ell-s}} \\ &= \sum_{\boldsymbol{i}' \in \mathbb{N}^n_{2^s}} P_{\boldsymbol{i}'2^{\ell-s}+\boldsymbol{j}}\, \omega^{[\boldsymbol{i}]_s \cdot \boldsymbol{i}' 2^{\ell-s}} \\ &= \boldsymbol{c}^s_{\boldsymbol{i}2^{\ell-s}+\boldsymbol{j}}. \end{aligned}$$

Thus, $\boldsymbol{c}^s_{g(\boldsymbol{k})} = \boldsymbol{c}^s_{\boldsymbol{k}}$ for all $\boldsymbol{k} \in \mathbb{N}^n_d$, whence $\boldsymbol{c}^0, ..., \boldsymbol{c}^\ell \in \mathbb{K}^{\mathbb{N}^n_d, G}$. □

This lemma implies that for the computation of the FFT of a G-symmetric polynomial P, it suffices to compute the projections $\boldsymbol{c}^0_F, ..., \boldsymbol{c}^\ell_F$.

In order to apply formula (4) for the computation of $\boldsymbol{c}^s_F$ as a function of $\boldsymbol{c}^{s-1}_F$, it is not necessary to completely reconstruct $\boldsymbol{c}^{s-1}$, due to the sparsity of the matrix $\mathbf{B}^s$. Instead, we define the *δ-expansion* of the set F by

$$F_\delta \;=\; \{\boldsymbol{k} \boxplus \boldsymbol{\epsilon}\, \delta \colon \boldsymbol{k} \in F, \boldsymbol{\epsilon} \in \mathbb{N}^n_2\},$$

where $\boldsymbol{i} \boxplus \boldsymbol{j}$ stands for the "bitwise exclusive or" of $\boldsymbol{i}$ and $\boldsymbol{j}$. For any $\boldsymbol{k} \in \mathbb{N}^n_d$, the set $\{\boldsymbol{k} \boxplus \boldsymbol{\epsilon}\, \delta \colon \boldsymbol{\epsilon} \in \mathbb{N}^n_2\}$ describes the vertices of the butterfly of span δ that includes the point $\boldsymbol{k}$. Thus, F_δ is indeed the set of indices of $\boldsymbol{c}^{s-1}$ that are involved in the computation of $\boldsymbol{c}^s$ *via* the formula $\boldsymbol{c}^s = \mathbf{B}^s\, \boldsymbol{c}^{s-1}$.

LEMMA 3. *For any $\delta \in \{1, 2, 4, ..., 2^{\ell-1}\}$, let*

$$F_{[\delta]} \;=\; \{\boldsymbol{k} \boxplus \boldsymbol{\epsilon} \colon \boldsymbol{k} \in F, \boldsymbol{\epsilon} \in \mathbb{N}^n_\delta\},$$

so that $F_\delta \subseteq F_{[2\delta]}$. Then we have $F_{[\delta]} = \delta\, F^{[d/\delta]} + \mathbb{N}^n_\delta$.

PROOF. Assume that $\boldsymbol{i} \in F^{[d/\delta]}$. Then clearly, $\delta\, \boldsymbol{i} \in F$, whence $\delta\, \boldsymbol{i} + \mathbb{N}^n_\delta \subseteq F_{[\delta]}$. Conversely, if $\boldsymbol{i} \in F_{[\delta]}$, then there exists a $\boldsymbol{j} \in F$ with $\boldsymbol{j} = \boldsymbol{i} \boxplus \boldsymbol{\epsilon}$ and $\boldsymbol{\epsilon} \in \mathbb{N}^n_\delta$. Let $\boldsymbol{k} = \lfloor \boldsymbol{j}/\delta \rfloor$. For any $g \in G$, we have $g(\boldsymbol{k}) = \lfloor g(\boldsymbol{j})/\delta \rfloor \leqslant_{\text{lex}} \lfloor \boldsymbol{j}/\delta \rfloor = \boldsymbol{k}$. Consequently, $\lfloor \boldsymbol{j}/\delta \rfloor \in F^{[d/\delta]}$, whence $\boldsymbol{i} \in \delta\, \lfloor \boldsymbol{j}/\delta \rfloor + \mathbb{N}^n_\delta$. □

For the proof of the next lemma, for $(j, k) \in \{1, ..., n\}^2$, define $\Delta_{j,k} = \{\boldsymbol{i} \in \mathbb{N}^n \colon \boldsymbol{i}_j = \boldsymbol{i}_k\}$ and $\Upsilon = \mathbb{N}^n_d \setminus \bigcup_{k \neq j} \Delta_{j,k}$.

LEMMA 4. *There exists a constant C (depending on n) such that*

$$\left| |F^{[d]}| - \frac{d^n}{|G|} \right| \;\leqslant\; C\, d^{n-1}.$$

PROOF. With the notation above, we have $|\Delta_{j,k} \cap \mathbb{N}^n_d| = d^{n-1}$. Taking $C = \binom{n}{2}$, it follows that $\left| \left(\bigcup_{k \neq j} \Delta_{j,k} \right) \cap \mathbb{N}^n_d \right| \leqslant C\, d^{n-1}$. On the other hand, the orbit of any element in Υ under G contains exactly $|G|$ elements and one element in F. In other words, $|F \cap \Upsilon| = |\Upsilon|/|G|$ and so $0 \leqslant |F| - |\Upsilon|/|G| \leqslant C\, d^{n-1}$. Finally, $0 \leqslant (d^n - |\Upsilon|)/|G| \leqslant C/|G|\ d^{n-1}$ which implies $-C/|G|\, d^{n-1} \leqslant |F| - d^n/|G| \leqslant C\, d^{n-1}$. □

3.4 The symmetric FFT

Let $\boldsymbol{c}^{s-1}_{F_\delta}$ be the restriction of $\boldsymbol{c}^{s-1}$ to F_δ and notice that $\mathbb{K}^{F_\delta}$ is stable under $\mathbf{B}^s$, *i.e.* $\mathbf{B}^s(\mathbb{K}^{F_\delta}) \subseteq \mathbb{K}^{F_\delta}$. Therefore the restriction $\mathbf{B}^s_{\mathbb{K}^{F_\delta}}$ of the map $\mathbf{B}^s \colon \mathbb{K}^{\mathbb{N}^n_d} \to \mathbb{K}^{\mathbb{N}^n_d}$ to vectors in $\mathbb{K}^{F_\delta}$ is a $\mathbb{K}$-algebra morphism from $\mathbb{K}^{F_\delta}$ to $\mathbb{K}^{F_\delta}$. By construction, we now have $\boldsymbol{c}^s_{F_\delta} = \mathbf{B}^s_{\mathbb{K}^{F_\delta}}\, \boldsymbol{c}^{s-1}_{F_\delta}$. This allows us to compute $\boldsymbol{c}^s_F$ as a function of $\boldsymbol{c}^{s-1}_F$ using

$$\boldsymbol{c}^s_F \;=\; (\mathbf{B}^s_{\mathbb{K}^{F_\delta}} \xi_\delta(\boldsymbol{c}^{s-1}_F))_F \tag{6}$$

where $\xi_\delta(\boldsymbol{c}^{s-1}_F)$ denotes the δ-expansion $\boldsymbol{c}^{s-1}_{F_\delta}$ of $\boldsymbol{c}^{s-1}_F$.

Remark 5. We could have saved a few more operations by only computing the coefficients $\boldsymbol{c}^s_F$. To do so, we would have performed just the operations of a butterfly corresponding to vertices inside F. However, the potential gain of complexity is only linear in the numbers of monomials d^n.

The formula (6) yields a straightforward way to compute the direct FFT of a G-symmetric polynomial:

Algorithm Symmetric-FFT
INPUT: n, d and coefficients $\boldsymbol{c}_F^0 \in \mathbb{K}^{\mathbb{N}_d^n}$ of $P \in \mathbb{K}[\boldsymbol{x}]_d^G$ in F
OUTPUT: $\boldsymbol{c}_F^\ell \in \mathbb{K}^{\mathbb{N}_d^n}$ in bit reversed order

For $s = 1, \ldots, \ell$ do
  $\delta := d/2^s$
  For $\boldsymbol{i}' \in \mathbb{N}_{2^{s-1}}^n$, $\boldsymbol{j} \in \mathbb{N}_{2^{\ell-s}}^n$ s.t. $2\,\boldsymbol{i}'\,\delta + \boldsymbol{j} \in F_\delta$ do
    For $\boldsymbol{b} \in \mathbb{N}_2^n$ do $\boldsymbol{c}_{\boldsymbol{b}} := \boldsymbol{c}^{s-1}_{\pi^G((2\boldsymbol{i}'+\boldsymbol{b})\delta+\boldsymbol{j})}$
    $\boldsymbol{d}_{\boldsymbol{b}} = \mathbf{Butterfly}(n, d, s, \boldsymbol{i}', \boldsymbol{j}, \boldsymbol{c}_{\boldsymbol{b}})$
    For $\boldsymbol{b} \in \mathbb{N}_2^n$ do $\boldsymbol{c}^s_{\pi^G((2\boldsymbol{i}'+\boldsymbol{b})\delta+\boldsymbol{j})} := \boldsymbol{d}_{\boldsymbol{b}}$
Return $\boldsymbol{c}^\ell$

Remark 6. The main challenge for an actual implementation of our algorithms consists in finding a way to iterate on sets like F_δ without too much overhead. We give a hint on how to iterate over F in Lemma 8 (see also [22] for similar considerations).

The inverse FFT can be computed classically by unrolling the loops in the inverse order and inverting the operations of the butterflies. The main result of this section is then the following theorem, where $\mathsf{F}(d, n)$ denotes the cost of a full n-dimensional FFT of multi-degree d.

THEOREM 7. *For fixed n and G, and for $d \to \infty$, the direct (resp. inverse) symmetric FFT can be computed in time*

$$\mathsf{T}(d, n) \;=\; \tfrac{1}{|G|}\,\mathsf{F}(d, n) + \mathcal{O}(d^n).$$

PROOF. At stage s of the computation, the ratio between the computation of $\boldsymbol{c}_F^s$ as a function of $\boldsymbol{c}_F^{s-1}$ and $\boldsymbol{c}^s$ as a function of $\boldsymbol{c}^{s-1}$ is $|F_{2^{\ell-s}}|/|\mathbb{N}_d^n|$, so that

$$\frac{\mathsf{T}(d, n)}{\mathsf{F}(d, n)} \;\leqslant\; \frac{|F_{2^{\ell-1}}| + |F_{2^{\ell-2}}| + \cdots + |F_2|}{\ell\,|\mathbb{N}_d^n|}.$$

Lemmas 3 and 4 thus imply

$$|F_\delta| \leqslant |F_{[2\delta]}| \leqslant |F^{[d/(2\delta)]}|\,(2\,\delta)^n \leqslant d^n/|G| + 2\,C\,\delta\,d^{n-1},$$

whence

$$\frac{\mathsf{T}(d, n)}{\mathsf{F}(d, n)} \;\leqslant\; \frac{\ell\,d^n/|G| + 2\,C\,d^n}{\ell\,d^n} = \frac{1}{|G|} + \frac{2\,C}{\ell}.$$

The result follows for $d \to \infty$. □

Finally, the following lemma gives a description of an "open" subset of F, characterized only by simple inequalities. Although we do not describe implementation questions in detail here, we point out that this simple characterization would naturally be used in order to iterate over the sets F_δ, possibly using tree-based techniques as in [17].

LEMMA 8. *Let Π be the set of $(j, k) \in \{1, \ldots, n\}^2$ such that $j < k$ and $\exists g \in G$ such that $g(i) = i$ for $i \leqslant j$ and $g(j) = k$. For $(j, k) \in \{1, \ldots, n\}^2$, define $H_{j,k} = \{\boldsymbol{i} \in \mathbb{N}^n : \boldsymbol{i}_j > \boldsymbol{i}_k\}$. Then*

$$F \cap \Upsilon \;=\; \bigcap_{(j,k)\in\Pi} H_{j,k} \cap \mathbb{N}_d^n.$$

PROOF. Assume that $\boldsymbol{i} \in \Upsilon$ does not lie in F. Then $g(\boldsymbol{i}) >_{\text{lex}} \boldsymbol{i}$ for some $g \in G$. Let $j \in \{1, \ldots, n\}$ be minimal with $k = g(j) \neq j$, whence $k > j$ and $(j, k) \in \Pi$. Since $\boldsymbol{i}_j \neq \boldsymbol{i}_k$, it follows that $\boldsymbol{i}_k > \boldsymbol{i}_j$, so $\boldsymbol{i} \notin H_{j,k}$. Inversely, assume that $\boldsymbol{i} \in \mathbb{N}_d^n$ does not lie in $\bigcap_{(j,k)\in\Pi} H_{j,k}$, so that there exists $(j, k) \in \Pi$ with $\boldsymbol{i} \notin H_{j,k}$. Let $g \in G$ be such that $g(i) = i$ for $i \leqslant j$ and $g(j) = k$. Then, $g(\boldsymbol{i}) >_{\text{lex}} \boldsymbol{i}$. □

4. THE LATTICE FFT

In this section, we deal with polynomials supported on lattices; our main result is an algorithm for the multiplication of such polynomials using Fourier transforms.

The first subsection introduces the main objects we will need (a lattice Λ and its dual Γ); then, we will give an algorithm, based on Smith normal form computation, for the FFT on what we will call a *basic domain*, and we will finally deduce a multiplication algorithm for the general case.

The techniques we use, based on Smith normal form computation, can be found in several papers, originating from [13]; in particular, the results in Section 4.2 are essentially in [33] (in a more general form).

4.1 Lattice polynomials

Assume that $\mathbb{K}$ admits a primitive kth root of unity for any order $k > 1$, which we will denote by $\mathrm{e}^{2\pi\mathrm{i}/k}$. Let Λ be a free $\mathbb{Z}$-submodule of $\mathbb{Z}^n$ of rank n, generated by the vectors $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_n \in \mathbb{N}^n$. Then

$$\mathbb{K}[\boldsymbol{x}]_\Lambda \;=\; \{P \in \mathbb{K}[\boldsymbol{x}] : \operatorname{supp} P \subseteq \Lambda\}$$

is a subring of $\mathbb{K}[\boldsymbol{x}]$, and we will call elements of $\mathbb{K}[\boldsymbol{x}]_\Lambda$ *lattice polynomials*. First, we show that these polynomials are the invariant polynomial ring for a diagonal matrix group.

The set $\mathbb{Q}^n$ acts on $\mathbb{K}[\boldsymbol{x}]$ *via*, for any $\boldsymbol{\gamma} = (\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_n) \in \mathbb{Q}^n$, the isomorphism $\varphi_{\boldsymbol{\gamma}}$ of $\mathbb{K}[\boldsymbol{x}]$ given by

$$\varphi_{\boldsymbol{\gamma}} : P(x_1, \ldots, x_n) \;\mapsto\; P(\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\gamma}_1} x_1, \ldots, \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\gamma}_n} x_n).$$

Note that $\varphi_{\boldsymbol{\gamma}+\boldsymbol{v}} = \varphi_{\boldsymbol{\gamma}}$ for any $\boldsymbol{v} \in \mathbb{Z}^n$. The action of $\varphi_{\boldsymbol{\gamma}}$ on the monomial $\boldsymbol{x}^{\boldsymbol{\lambda}_i}$ is given by $\varphi_{\boldsymbol{\gamma}}(\boldsymbol{x}^{\boldsymbol{\lambda}_i}) = \mathrm{e}^{2\pi\mathrm{i}(\boldsymbol{\lambda}_i \cdot \boldsymbol{\gamma})}\,\boldsymbol{x}^{\boldsymbol{\lambda}_i}$.

In particular, all elements of $\mathbb{K}[\boldsymbol{x}]_{\boldsymbol{\Lambda}}$ are invariants under the action of $\varphi_{\boldsymbol{\gamma}}$ if and only if

$$^t\boldsymbol{\Lambda}\,\boldsymbol{\gamma} \;\in\; \mathbb{Z}^n, \tag{7}$$

where $\boldsymbol{\Lambda} \in \mathbb{N}^{n\times n}$ is the matrix with columns $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_n$, that is $\boldsymbol{\Lambda}_{i,j} = (\boldsymbol{\lambda}_j)_i$. Let Γ be the dual (or reciprocal) lattice of Λ, that is the set of all $\boldsymbol{\gamma} \in \mathbb{Q}^n$ satisfying Equation (7). A basis of Γ is given by the columns of $\boldsymbol{\Gamma} = {}^t\boldsymbol{\Lambda}^{-1} \in \mathbb{Q}^{n\times n}$.

Let G be the group of actions $\{\varphi_{\boldsymbol{\gamma}} : \boldsymbol{\gamma} \in \boldsymbol{\Gamma}\}$. It is generated by $\{\varphi_{\boldsymbol{\gamma}_i}\}_{1\leqslant i\leqslant n}$, where $\boldsymbol{\gamma}_i$ are the columns of $\boldsymbol{\Gamma}$. From Equation (7), we deduce that G is the diagonal matrix group of all actions $\varphi_{\boldsymbol{\gamma}}$ which leave $\mathbb{K}[\boldsymbol{x}]_\Lambda$ invariant. Conversely, because monomials are mapped to monomials by elements of G, the ring of invariants is spanned by the monomials $x^{\boldsymbol{\lambda}}$ with ${}^t\boldsymbol{\lambda}\,\boldsymbol{\Gamma} \in \mathbb{Z}^n$, *i.e.* $\boldsymbol{\lambda}$ belongs to the dual of Γ. Since the dual of the dual of a lattice is the lattice itself, we deduce that $\boldsymbol{\lambda} \in \Lambda$ and $\mathbb{K}[\boldsymbol{x}]_\Lambda = \mathbb{K}[\boldsymbol{x}]^G$. Note that only $\boldsymbol{\Gamma}$ modulo $\mathbb{Z}^{n\times n}$ matters to determine the group G.

Example 9. Consider the lattice Λ generated by $\boldsymbol{\lambda}_1 = (2,\ 0)$ and $\boldsymbol{\lambda}_2 = (1,\ 1)$, and let $\boldsymbol{\Lambda} = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$. The lattice polynomials $\mathbb{K}[\boldsymbol{x}]_\Lambda$ are the polynomials $\mathbb{K}[x_1^2,\ x_1\ x_2,\ x_2^2]$. We have $\boldsymbol{\Gamma} = \begin{pmatrix} 1/2 & 0 \\ -1/2 & 1 \end{pmatrix}$, so G is the group generated by $\varphi_{\boldsymbol{\gamma}_1}$: $P(x_1,\ x_2) \mapsto P(-x_1,\ -x_2)$ and $\varphi_{\boldsymbol{\gamma}_2} = \mathrm{Id}$. The lattice polynomials $\mathbb{K}[\boldsymbol{x}]_\Lambda = \mathbb{K}[x_1^2, x_1\,x_2, x_2^2]$ are those polynomials invariant under the symmetry $P(x_1, x_2) \mapsto P(-x_1, -x_2)$.

4.2 The lattice FFT on a basic domain

Given $\boldsymbol{d} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n)$, we define

$$\begin{array}{rcl} \mathbb{K}[\boldsymbol{x}]_{\boldsymbol{d}} & = & \{P \in \mathbb{K}[\boldsymbol{x}] : \deg_{x_1}(P) < \boldsymbol{d}_1, \ldots, \deg_{x_n}(P) < \boldsymbol{d}_n\} \\ \mathbb{K}[\boldsymbol{x}]_{\Lambda,\boldsymbol{d}} & = & \mathbb{K}[\boldsymbol{x}]_\Lambda \cap \mathbb{K}[\boldsymbol{x}]_{\boldsymbol{d}}. \end{array}$$

For $i \in \{1, \ldots, n\}$, let $p_i > 0$ be minimal such that $x_i^{p_i} \in \mathbb{K}[\boldsymbol{x}]_\Lambda$. We call the block $\mathbb{N}_{\boldsymbol{p}} := \mathbb{N}_{p_1} \times \cdots \times \mathbb{N}_{p_n}$ a *basic domain* for Λ. In this subsection, we consider the computation of the FFT at order $\boldsymbol{p}$ of a polynomial $P \in \mathbb{K}[\boldsymbol{x}]_{\Lambda,\boldsymbol{p}}$.

In what follows, we set $\omega_i = \mathrm{e}^{2\pi \mathrm{i}/p_i}$ for each i, so that we have to show how to compute the set of evaluations $P(\boldsymbol{\omega}^{\boldsymbol{k}}) = P(\omega_1^{\boldsymbol{k}_1}, \ldots, \omega_n^{\boldsymbol{k}_n})$ for $\boldsymbol{k} \in \mathbb{N}_{\boldsymbol{p}}$. If we proceeded directly, we would compute more evaluations that the number of monomials of P, which is $|\Lambda \cap \mathbb{N}_{\boldsymbol{p}}|$. We show how to reduce the problem to computing exactly $|\Lambda \cap \mathbb{N}_{\boldsymbol{p}}|$ evaluations.

For any $\boldsymbol{\gamma} = (\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_n) \in \Gamma$, notice that

$$\begin{array}{rcl} P(\boldsymbol{\omega}^{\boldsymbol{k}}) & = & P(\mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}_1/p_1}, \ldots, \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}_n/p_n}) \\ & = & P(\mathrm{e}^{2\pi \mathrm{i}(\boldsymbol{k}_1/p_1 + \boldsymbol{\gamma}_1)}, \ldots, \mathrm{e}^{2\pi \mathrm{i}(\boldsymbol{k}_n/p_n + \boldsymbol{\gamma}_n)}). \end{array}$$

Therefore we would only need to consider the evaluations with multi-indices in $(1/p_1\,\mathbb{Z}, \ldots, 1/p_n\,\mathbb{Z})$ modulo Γ. There are only $|\Lambda \cap \mathbb{N}_{\boldsymbol{p}}|$ such evaluations but, as in Section 3.4, we would have to expand the fundamental domain of evaluations at each stage of the FFT to compute the butterflies.

Instead, we propose a more direct method with no domain expansion. We show that, regarding the evaluations, polynomials P in $\mathbb{K}[\boldsymbol{x}]_\Lambda$ can be written $Q(\boldsymbol{x}^{\boldsymbol{\lambda}'_1}, \ldots, \boldsymbol{x}^{\boldsymbol{\lambda}'_n})$ where $\boldsymbol{\lambda}'_1, \ldots, \boldsymbol{\lambda}'_n$ is a basis of Λ and the evaluation can be done directly in this rewritten form.

We introduce the notation $\overline{k}^{\,p} \in \{0, \ldots, p-1\}$ for the remainder of $k \in \mathbb{Z}$ by $p \in \mathbb{N}^*$. If $\boldsymbol{k} = (\boldsymbol{k}_1, \ldots, \boldsymbol{k}_n) \in \mathbb{Z}^n$ and $\boldsymbol{p} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n) \in (\mathbb{N}^*)^n$, we let $\overline{\boldsymbol{k}}^{\,\boldsymbol{p}} := (\overline{\boldsymbol{k}_1}^{\,\boldsymbol{p}_1}, \ldots, \overline{\boldsymbol{k}_n}^{\,\boldsymbol{p}_n})$. This notation is motivated by the remark that $\boldsymbol{x}^{\boldsymbol{\lambda}}(\boldsymbol{\omega}^{\boldsymbol{k}})$ depends only on the class of $\boldsymbol{\lambda}$ and $\boldsymbol{k}$ modulo $(p_1\,\mathbb{Z}, \ldots, p_n\,\mathbb{Z})$.

LEMMA 10. *There exists a basis $(\boldsymbol{\lambda}'_1, \ldots, \boldsymbol{\lambda}'_n)$ of Λ and a basis $(\boldsymbol{k}_1, \ldots, \boldsymbol{k}_n)$ of $\mathbb{Z}^n$ such that*

$$\boldsymbol{y}_i(\boldsymbol{\omega}^{\boldsymbol{k}_j}) \;=\; \mathrm{e}^{2\pi \mathrm{i} \delta_{i,j}/q_i} \tag{8}$$

where $\boldsymbol{y}_i = \boldsymbol{x}^{\overline{\boldsymbol{\lambda}'_i}^{\,\boldsymbol{p}}}$ and the q_i's are positive integers satisfying $q_1 \,|\, q_2 \,|\, \cdots \,|\, q_n$.

PROOF. Let us consider the lattice of the exponents of $(\boldsymbol{x}^{\boldsymbol{\lambda}_i}(\boldsymbol{\omega}^{\boldsymbol{k}}))_{1 \leqslant i \leqslant n}$ for $\boldsymbol{k} \in \mathbb{N}^n$. If $\boldsymbol{k} = (\boldsymbol{k}_1, \ldots, \boldsymbol{k}_n) \in \mathbb{N}^n$, then

$$\boldsymbol{x}^{\boldsymbol{\lambda}_i}(\boldsymbol{\omega}^{\boldsymbol{k}}) = \mathrm{e}^{2\pi \mathrm{i}[(\boldsymbol{k}_1/p_1, \ldots, \boldsymbol{k}_n/p_n) \cdot \boldsymbol{\lambda}_i]} = \mathrm{e}^{2\pi \mathrm{i}[\boldsymbol{k} \cdot ((\boldsymbol{\lambda}_i)_1/p_1, \ldots, (\boldsymbol{\lambda}_i)_n/p_n)]}.$$

We define the lattice Δ spanned by the columns of $\boldsymbol{\Delta} = {}^t\boldsymbol{\Lambda} \begin{pmatrix} 1/p_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 1/p_n \end{pmatrix}$, that is $\boldsymbol{\Delta}_{i,j} = (\boldsymbol{\lambda}_i)_j/p_j$. We want to take the Smith normal form of $\boldsymbol{\Delta}$ but the coefficients of $\boldsymbol{\Delta}$ are not necessarily integers. So we multiply the matrix by $\ell := \mathrm{LCM}(p_1, \ldots, p_n)$, take the Smith normal form and divide by ℓ. Therefore there exists ${}^t\boldsymbol{H}, \boldsymbol{K} \in \mathrm{GL}_n(\mathbb{Z})$ and integers $d_n \,|\, \ldots \,|\, d_2 \,|\, d_1$ such that

$${}^t\boldsymbol{H}\,\boldsymbol{\Delta}\,\boldsymbol{K} = \begin{pmatrix} d_1/\ell & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & d_n/\ell \end{pmatrix}. \tag{9}$$

Let us prove that $\mathbb{Z}^n \subseteq \Delta$. By definition of $p_1, \ldots, p_n$, there exists $\boldsymbol{S} \in \mathbb{Z}^{n \times n}$ such that $\boldsymbol{\Lambda}\,\boldsymbol{S} = \begin{pmatrix} p_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & p_n \end{pmatrix}$. Thus we have ${}^t\boldsymbol{S}\,{}^t\boldsymbol{\Lambda} \begin{pmatrix} 1/p_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 1/p_n \end{pmatrix} = \mathrm{Id}$, implying that $\boldsymbol{\Delta}\,{}^t\boldsymbol{S} = \mathrm{Id}$ and our result. Because $\mathbb{Z}^n \subseteq \Delta$, we have $d_i \mid \ell$ and by setting $q_i := \ell/d_i \in \mathbb{N}^*$, we get $q_1 \,|\, q_2 \,|\, \ldots \,|\, q_n$.

With a geometrical point of view, the equality of Equation (9) gives the existence of the two required bases. The columns of $\boldsymbol{K}$ give the basis $(\boldsymbol{k}_1, \ldots, \boldsymbol{k}_n)$ of $\mathbb{Z}^n$ and the columns of $\boldsymbol{\Lambda}' := \boldsymbol{\Lambda}\,\boldsymbol{H}$ give the basis $(\boldsymbol{\lambda}'_1, \ldots, \boldsymbol{\lambda}'_n)$ of Λ. To sum up, we have

$${}^t\boldsymbol{\Lambda}' \begin{pmatrix} 1/p_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 1/p_n \end{pmatrix} \boldsymbol{K} = \begin{pmatrix} 1/q_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & 1/q_n \end{pmatrix} \tag{10}$$

which is the matricial form of Equation (8). □

PROPOSITION 11. *The following ring morphism Φ*

$$\begin{array}{ccc} \mathbb{K}[\boldsymbol{y}]/(y_1^{q_1} - 1, \ldots, y_n^{q_n} - 1) & \longrightarrow & \mathbb{K}[\boldsymbol{x}]/(x_1^{p_1} - 1, \ldots, x_n^{p_n} - 1) \\ y_i & \longmapsto & \boldsymbol{x}^{\overline{\boldsymbol{\lambda}'_i}^{\,\boldsymbol{p}}} \end{array}$$

is well-defined, injective and its image is $\mathbb{K}[\boldsymbol{x}]_{\Lambda,\boldsymbol{p}}$. Moreover, if $P = \Phi(Q)$ then

$$P(\boldsymbol{\omega}^{\ell_1 \boldsymbol{k}_1 + \cdots + \ell_n \boldsymbol{k}_n}) \;=\; Q(\mathrm{e}^{2\pi \mathrm{i} \ell_1/q_1}, \ldots, \mathrm{e}^{2\pi \mathrm{i} \ell_n/q_n}). \tag{11}$$

PROOF. First, we prove that Φ is well defined. For this matter we have to check that $\Phi(y_i^{q_i} - 1) = \boldsymbol{x}^{q_i \overline{\boldsymbol{\lambda}'_i}^{\,\boldsymbol{p}}} - 1 = 0$ modulo $(x_1^{p_1} - 1, \ldots, x_n^{p_n} - 1)$. It is sufficient to prove that $q_i\,\overline{\boldsymbol{\lambda}'_i}^{\,\boldsymbol{p}} \in (p_1\,\mathbb{Z}, \ldots, p_n\,\mathbb{Z})$, which follows from Equation (10).

Then we prove Equation (11). Let $Q \in \mathbb{K}[\boldsymbol{y}]$ and $P(\boldsymbol{x}) := \Phi(Q)(\boldsymbol{x})$. As a consequence of Lemma 10, one has

$$P(\boldsymbol{\omega}^{\ell_1 \boldsymbol{k}_1 + \cdots + \ell_n \boldsymbol{k}_n}) \;=\; Q(\mathrm{e}^{2\pi \mathrm{i} \ell_1/q_1}, \ldots, \mathrm{e}^{2\pi \mathrm{i} \ell_n/q_n}).$$

Now, we prove that Φ is injective. Indeed if $Q \in \mathbb{K}[\boldsymbol{y}]$ satisfies $\Phi(Q) = 0$ then for all $\boldsymbol{k} \in \mathbb{N}^n$, one has $\Phi(Q)(\boldsymbol{\omega}^{\boldsymbol{k}}) = 0$. Using Equation (11), we get that for all $\ell_1, \ldots, \ell_n \in \mathbb{Z}$, $Q(\mathrm{e}^{2\pi \mathrm{i} \ell_1/q_1}, \ldots, \mathrm{e}^{2\pi \mathrm{i} \ell_n/q_n}) = 0$. As a result, Q belongs to the ideal generated by $(y_1^{q_1} - 1, \ldots, y_n^{q_n} - 1)$.

Finally it is trivial to see that the image of Φ is included in $\mathbb{K}[\boldsymbol{x}]_{\Lambda,\boldsymbol{p}}$. Reciprocally, let $\boldsymbol{x}^{\boldsymbol{\lambda}} \in \mathbb{K}[\boldsymbol{x}]_{\Lambda,\boldsymbol{p}}$. We write $\boldsymbol{\lambda} = \sum_{i=1}^n \ell_i\,\overline{\boldsymbol{\lambda}'_i}^{\,\boldsymbol{p}}$ with $\ell_i \in \mathbb{Z}$ and define $\overline{\boldsymbol{\lambda}} = \sum_{i=1}^n \overline{\ell_i}^{\,q_i}\,\overline{\boldsymbol{\lambda}'_i}^{\,\boldsymbol{p}}$. Now because the lattice $q_1\,\boldsymbol{\lambda}'_1\,\mathbb{Z} \oplus \cdots \oplus q_n\,\boldsymbol{\lambda}'_n\,\mathbb{Z}$ is included in $(p_1\,\mathbb{Z}, \ldots, p_n\,\mathbb{Z})$, we have that $\Phi\big(\boldsymbol{y}^{(\overline{\ell_1}^{\,q_1}, \ldots, \overline{\ell_n}^{\,q_n})}\big) = x^{\overline{\boldsymbol{\lambda}}} = x^{\boldsymbol{\lambda}}$ modulo $(x_1^{p_1} - 1, \ldots, x_n^{p_n} - 1)$. □

In particular, the FFT of P is uniquely determined by its restriction to evaluations of the form $P(\boldsymbol{\omega}^{\boldsymbol{k}})$ with $\boldsymbol{k} = \ell_1 \boldsymbol{k}_1 + \cdots + \ell_n \boldsymbol{k}_n$ and $\boldsymbol{\ell} = (\ell_1, ..., \ell_n) \in \mathbb{N}_{\boldsymbol{q}}$. Notice that Proposition 11 implies that the number $q_1 \cdots q_n$ of evaluations equals the numbers $|\Lambda \cap \mathbb{N}_{\boldsymbol{p}}|$ of monomials in $\mathbb{K}[\boldsymbol{x}]_{\Lambda,\boldsymbol{p}}$.

We define the *lattice FFT* of P to be the vector of values $(P(\boldsymbol{\omega}^{\boldsymbol{\ell}_1 \boldsymbol{k}_1 + \cdots + \boldsymbol{\ell}_n \boldsymbol{k}_n}))_{\boldsymbol{\ell} \in \mathbb{N}_{\boldsymbol{q}}}$. We have thus shown that the computation of the lattice FFT of P reduces to the computation of an ordinary multivariate FFT of order $\boldsymbol{q}$.

4.3 The general lattice FFT multiplication

Assume that $\boldsymbol{d} \in (\mathbb{N}^*)^n$ is such that $p_i \mid d_i$ for each i and consider the computation of the FFT at order $\boldsymbol{d}$ of a lattice polynomial $P \in \mathbb{K}[\boldsymbol{x}]_{\Lambda,\boldsymbol{d}}$. In practice, one often has $\boldsymbol{d}_i = p_i\, 2^{\ell_i}$, and this is what we will assume from now on. For simplicity, we also suppose that $\ell_1 = \cdots = \ell_n$ and denote by ℓ this common value. The results are valid in general but we would have to do more decimations in some variables than in others, which would not match the notations of Section 2.

We give here an algorithm for the multiplication of two polynomials $P_1, P_2 \in \mathbb{K}[\boldsymbol{x}]_{\Lambda,\boldsymbol{d}/2}$.

We start by doing ℓ FFT stages. These stages preserve the lattice Λ, because the butterflies have a span which belongs to Λ. Therefore we compute only the butterflies whose vertices are in Λ.

After these ℓ stages, we twist the coefficients thanks to Formula 5 and obtain the polynomials $Q^\ell = (Q^\ell_{\boldsymbol{i}})$, for $\boldsymbol{i} \in \mathbb{N}^n_{2^\ell}$. As explained in Section 2.2, up to a minor modification (here, the partial degrees are not all the same), these polynomials reduce our multiplication to $2^{n\ell}$ multiplications in $\mathbb{K}[\boldsymbol{x}]_{\Lambda,\boldsymbol{p}}/(x_1^{p_1} - 1, ..., x_n^{p_n} - 1)$. Thanks to Proposition 11, we are able to perform these latter multiplications as multiplications in $\mathbb{K}[\boldsymbol{y}]_{\boldsymbol{q}}/(y_1^{q_1} - 1, ..., y_n^{q_n} - 1)$.

Algorithm Lattice-partial-FFT
INPUT: n, $\boldsymbol{d}$, ω and coefficients $\boldsymbol{c}^0 \in \mathbb{K}^{\mathbb{N}^n_d}$ of $P \in \mathbb{K}[\boldsymbol{x}]_{\Lambda,d}$
OUTPUT: the coefficients $\boldsymbol{c}^\ell \in \mathbb{K}^{\mathbb{N}^n_d}$

For $s = 1, ..., \ell$ do //ℓ decimation steps in each variable
  $\delta := d/2^s$
  For $\boldsymbol{i}' \in \mathbb{N}^n_{2^{s-1}}$, $\boldsymbol{j} \in \Lambda \cap \mathbb{N}^n_{d/2^s}$ do //a butterfly in Λ
    For $\boldsymbol{b} \in \mathbb{N}^n_2$ do $\boldsymbol{c}_{\boldsymbol{b}} := \boldsymbol{c}^{s-1}_{(2\boldsymbol{i}'+\boldsymbol{b})\delta+\boldsymbol{j}}$
    $\boldsymbol{d}_{\boldsymbol{b}} = \mathbf{Butterfly}(n, d, s, \boldsymbol{i}', \boldsymbol{j}, \boldsymbol{c}_{\boldsymbol{b}})$
    For $\boldsymbol{b} \in \mathbb{N}^n_2$ do $\boldsymbol{c}^s_{(2\boldsymbol{i}'+\boldsymbol{b})\delta+\boldsymbol{j}} := \boldsymbol{d}_{\boldsymbol{b}}$
Return $\boldsymbol{c}^s$

Algorithm Lattice-FFT-multiplication
INPUT: n, $\boldsymbol{d}$, ω and $P_1, P_2 \in \mathbb{K}[\boldsymbol{x}]_{\Lambda,d}$
OUTPUT: the coefficients product $P_1 P_2$

For $i = 1, 2$ do
  $\boldsymbol{c}^\ell = \mathbf{Lattice\text{-}partial\text{-}FFT}(n, 2\boldsymbol{d}, \omega, ((P_i)_{\boldsymbol{k}})_{\boldsymbol{k} \in \mathbb{N}^n_{2d}})$
  Let $Q_i := Q^\ell \in (\mathbb{K}[\boldsymbol{x}]_{\Lambda,\boldsymbol{p}})^{\mathbb{N}^n_{2^\ell}}$ obtained from $\boldsymbol{c}^\ell$
  Transform Q_i as a vector of $\mathbb{K}[\boldsymbol{y}]_{\boldsymbol{q}}$ using Proposition 11
Multiply $Q = Q_1\, Q_2$ in $(\mathbb{K}[\boldsymbol{y}]/(y_1^{q_1} - 1, ..., y_n^{q_n} - 1))^{\mathbb{N}^n_{2^\ell}}$
Transform Q as a vector of $\mathbb{K}[\boldsymbol{x}]_{\Lambda,\boldsymbol{p}}$ using Proposition 11
Recover $\boldsymbol{c}^\ell$ from $Q \in (\mathbb{K}[\boldsymbol{x}]_{\Lambda,\boldsymbol{p}})^{\mathbb{N}^n_{2^\ell}}$
$\boldsymbol{c}^0 = \mathbf{Lattice\text{-}partial\text{-}FFT}^{-1}(n, 2\boldsymbol{d}, \omega, (\boldsymbol{c}^\ell_{[\boldsymbol{k}]_\ell})_{\boldsymbol{k} \in \mathbb{N}^n_{2d}})$
Return $\boldsymbol{c}^0$

As for symmetric polynomials, the inverse algorithm **Lattice-partial-FFT**$^{-1}$ is obtained by reversing the loops and inverting the butterflies of **Lattice-partial-FFT**.

Let us analyze the cost of this algorithm. It starts with ℓ stages of classical n-dimensional FFT, computing only the butterflies whose vertices are in Λ. This amounts to $\frac{3}{2}\, n\, \ell\, |\Lambda \cap \mathbb{N}^n_d|$ arithmetic operations in $\mathbb{K}$. The second part is made of the transformations of Equation 5 and Proposition 11. Since we consider that operations with the lattice Λ take time $\mathcal{O}(1)$, these transformations take time $\mathcal{O}(|\Lambda \cap \mathbb{N}^n_d|)$. Finally, the componentwise multiplication of $(\mathbb{K}[\boldsymbol{y}]/(y_1^{q_1} - 1, ..., y_n^{q_n} - 1))^{\mathbb{N}^n_{2^\ell}}$ costs $2^{n\ell}\, \mathsf{M}(\boldsymbol{q}_1, ..., \boldsymbol{q}_n)$ where $\mathsf{M}(\boldsymbol{d})$ stands for the arithmetic complexity of polynomials multiplication in $\mathbb{K}[\boldsymbol{x}]_{\boldsymbol{d}}$. Altogether, our multiplication algorithms costs

$$\mathsf{T}(\boldsymbol{d}) := 3/2\, n\, \ell\, |\Lambda \cap \mathbb{N}^n_d| + 2^{n\ell}\, \mathsf{M}(\boldsymbol{q}) + \mathcal{O}(|\Lambda \cap \mathbb{N}^n_d|).$$

By contrast, a symmetry-oblivious approach would consist in doing ℓ stages of classical n-dimensional FFT and then $2^{n\ell}$ multiplications in $\mathbb{K}[\boldsymbol{x}]/(x_1^{p_1} - 1, ..., x_n^{p_n} - 1)$. The cost analysis is similar to the one before and the classical approach's cost is $\mathsf{F}(\boldsymbol{d}) := 3/2\, n\, \ell\, d^n + 2^{n\ell}\, \mathsf{M}(\boldsymbol{p}) + \mathcal{O}(d^n)$.

The ratio $d^n/|\Lambda \cap \mathbb{N}^n_d|$ is exactly the volume $\mathrm{vol}(\Lambda)$ of the lattice, defined by $\mathrm{vol}(\Lambda) := \det(\boldsymbol{\Lambda}) \in \mathbb{Z}$. Under the superlinearity assumption for the function M, we get that $\mathrm{vol}(\Lambda)\, \mathsf{M}(\boldsymbol{q}) \leqslant \mathsf{M}(\boldsymbol{p})$ and we deduce the following theorem.

THEOREM 12. *For a fixed lattice Λ and $d \to \infty$, the direct (resp. inverse) lattice FFT can be computed in time*

$$\mathsf{T}(\boldsymbol{d}) \;=\; \frac{1}{\mathrm{vol}(\Lambda)}\, \mathsf{F}(\boldsymbol{d}) + \mathcal{O}(d^n).$$

5. THE SYMMETRIC TFT

To conclude this paper, we briefly discuss the extensions of the previous results to the Truncated Fourier Transform (TFT). With the notation of Section 2, let $I \subseteq \mathbb{N}^n_d$ be an initial segment subset of $\mathbb{N}^n_d$: for $\boldsymbol{i}, \boldsymbol{j} \in \mathbb{N}^n_d$, if $\boldsymbol{i} \in I$ and $\boldsymbol{j} \leqslant \boldsymbol{i}$, then $\boldsymbol{j} \in I$. Given $P \in \mathbb{K}[\boldsymbol{x}]_I$, the TFT of P is defined to be the vector $\hat{P} \in \mathbb{K}^I$ defined by $\hat{P}_{\boldsymbol{i}} = P(\omega^{[\boldsymbol{i}]_\ell})$, where ω is a root of unity of order d.

In [16, 17], van der Hoeven described fast algorithms for computing the TFT and its inverse. For a fixed dimension n, the cost of these algorithms is bounded by $\mathcal{O}(|I| \log |I| + d^n)$ instead of $\mathcal{O}(d^n \log d^n)$. In this section we will outline how a further acceleration can be achieved for symmetric polynomials of the types studied in Sections 3 and 4.

5.1 The symmetric TFT

For each $\delta \in \{1, ..., 2^\ell\}$, let $I_{[\delta]} = \{\boldsymbol{i} \boxplus \boldsymbol{\epsilon} \colon \boldsymbol{i} \in I, \boldsymbol{\epsilon} \in \mathbb{N}^n_\delta\}$. The entire computation of the TFT and its inverse can be represented schematically by a graph Γ. The vertices of the graph are pairs $(s, \boldsymbol{i})$ with $s \in \{0, ..., \ell\}$ and $\boldsymbol{i} \in I_{[2^{\ell-s}]}$. The edges are between vertices $(s, \boldsymbol{i})$ and $(s+1, \boldsymbol{j})$ with $\boldsymbol{i} = \boldsymbol{j} \boxplus (2^{\ell-s-1}\, \boldsymbol{\epsilon})$ and $\boldsymbol{\epsilon} \in \mathbb{N}^n_2$. The edge is labeled by a constant that we will simply write $c_{s,\boldsymbol{i},\boldsymbol{j}}$ such that

$$P^{s+1}_{\boldsymbol{j}} \;=\; \sum_{\boldsymbol{i}} c_{s,\boldsymbol{i},\boldsymbol{j}}\, P^s_{\boldsymbol{i}}. \tag{12}$$

For a direct TFT, we are given $P^0_{\boldsymbol{i}}$ on input and "fill out" the remaining values $P^s_{\boldsymbol{i}}$ for increasing values of s using (12). In the case of an inverse TFT, we are given the $P^\ell_{\boldsymbol{i}}$ with $\boldsymbol{i} \in I$ on input, as well as the coefficients $P^0_{\boldsymbol{i}} = 0$ with $\boldsymbol{i} \in \mathbb{N}^n_d \setminus I$. We next "fill out" the remaining values $P^s_{\boldsymbol{i}}$ using the special algorithm described in [17].

Now let G be as in Section 3 and assume that I is stable under G. Then each of the $I_{[\delta]}$ is also stable under G. Furthermore, any $\boldsymbol{i} \in I$ lies in the orbit of some element of $I \cap F$ under the action of G. Let $(I \cap F)_{[\delta]} = \{\boldsymbol{i} \boxplus \boldsymbol{\epsilon}: \boldsymbol{i} \in I \cap F, \boldsymbol{\epsilon} \in \mathbb{N}_\delta^n\}$. Given a G-symmetric input polynomial on input, the idea behind the symmetric TFT is to use the restriction Γ' of the above graph Γ by keeping only those vertices $(s, \boldsymbol{i})$ such that $\boldsymbol{i} \in (F \cap I)_{[2^{\ell-s}]}$. The symmetric TFT and its inverse can then be computed by intertwining the above "filling out" process with steps in which we compute $P^s_{g(\boldsymbol{i})} = P^s_{\boldsymbol{i}}$ for all $\boldsymbol{i} \in \mathbb{N}_d^n$ and $g \in G$ such that $P^s_{\boldsymbol{i}}$ is known but not $P^s_{g(\boldsymbol{i})}$.

The computational complexities of the symmetric TFT and its inverse are both proportional to the size $|\Gamma'|$ of Γ'. For many initial domains I of interest, it can be shown that $|\Gamma'| \sim |\Gamma|/|G|$, as soon as d gets large. This is for instance the case for $I = \Sigma_k = \{\boldsymbol{i} \in \mathbb{N}_d^n: i_1 + \cdots + i_n \leqslant k\}$, when $k \to \infty$, and where $\ell = \lceil \log_2 k \rceil$. Indeed, using similar techniques as in the proof of Theorem 7, we first show that $|(\Sigma_k \cap F)_{[\delta]} - (\Sigma_k)_{[\delta]}/|G|| = \mathcal{O}(\delta\, d^{n-1})$, and then conclude in a similar way.

5.2 The lattice TFT

Let us now consider the case of a polynomial $P \in \mathbb{K}[\boldsymbol{x}]_{\Lambda \cap I}$, with Λ as in Section 4. In order to design a fast "lattice TFT", the idea is simply to replace I by a slightly larger set J which preserves the fundamental domains. More precisely, with $p_1, \ldots, p_n$ as in Section 4, we take

$$\begin{aligned} J &= \{(p_1\, i_1 + j_1, \ldots, p_n\, i_n + j_n): \boldsymbol{i} \in K, \boldsymbol{j} \in \mathbb{N}_{\boldsymbol{p}}\} \\ K &= \{(\lfloor i_1/p_1 \rfloor, \ldots, \lfloor i_n/p_n \rfloor): \boldsymbol{i} \in I\}. \end{aligned}$$

A lattice TFT of order $(p_1\ 2^{\ell_1},\ \ldots,\ p_n\ 2^{\ell_n})$ can then be regarded as $|\Lambda \cap \mathbb{N}_{\boldsymbol{p}}|$ TFTs at order $(2^{\ell_1}, \ldots, 2^{\ell_n})$ and initial segment K, followed by $|K|$ TFTs lattice FFTs on a fundamental domain. Asymptotically speaking, we thus gain a factor $|\mathbb{N}_{\boldsymbol{p}}|/|\Lambda \cap \mathbb{N}_{\boldsymbol{p}}|$ with respect to a usual TFT with initial segment J. In the case when $I = \Sigma_k = \{\boldsymbol{i} \in \mathbb{N}_d^n: i_1 + \cdots + i_n \leqslant k\}$, we finally notice that $|J| \sim |I|$ for $k \to \infty$.

6. CONCLUSION

Let us quickly outline possible generalizations of the results of this paper.

It seems that the most general kinds of finite groups for which the techniques in this paper work are finite subgroups G of $\Pi_n\, U_n$, where Π_n is the group of $n \times n$ permutation matrices and U_n the group of diagonal matrices whose entries are all roots of unity. Indeed, any such group G both acts on the sets $\mathbb{K}[\boldsymbol{x}]$ and on the torus $\mathbb{U}^n$, where $\mathbb{U} = \{e^{2\pi\, i/n}: i, n \in \mathbb{N}\}$. Even more generally, the results may still hold for closed algebraic subgroups G generated by an infinite number of elements of $\Pi_n\, U_n$.

Many other interesting groups can be obtained as conjugates $G' = T^{-1}\, G\, T$ of groups G of the above kind. For certain applications, such as the integration of dynamical systems, the change of coordinates T can be done "once and for all" on the initial differential equations, after which the results of this paper again apply.

It is classical that the FFT of a polynomial with real coefficients can be computed twice as fast (roughly speaking) as a polynomial with complex coefficient. Real polynomials can be considered as symmetric polynomials for complex conjugation: $P(\bar{z}) = \overline{P(z)}$. Some further extensions of our setting are possible by including this kind of symmetries.

On some simple examples, we have verified that the ideas of this paper generalize to other evaluation-interpolation models for polynomial multiplication, such as Karatsuba multiplication and Toom-Cook multiplication.

We intend to study the above generalizations in more detail in a forthcoming paper.

7. REFERENCES

[1] L. Auslander, J. R. Johnson and R. W. Johnson. An equivariant Fast Fourier Transform algorithm. Drexel University Technical Report DU-MCS-96-02, 1996.

[2] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 301–309. New York, NY, USA, 1988. ACM Press.

[3] R. Bergmann. The fast Fourier transform and fast wavelet transform for patterns on the torus. *Applied and Computational Harmonic Analysis*, , 2012. In Press.

[4] P. Bürgisser, M. Clausen and M. A. Shokrollahi. *Algebraic complexity theory*. Springer-Verlag, 1997.

[5] J. Canny, E. Kaltofen and Y. Lakshman. Solving systems of nonlinear polynomial equations faster. In *Proc. ISSAC '89*, pages 121–128. Portland, Oregon, A.C.M., New York, 1989. ACM Press.

[6] D.G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28:693–701, 1991.

[7] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Computat.*, 19:297–301, 1965.

[8] S. Czapor, K. Geddes and G. Labahn. *Algorithms for Computer Algebra*. Kluwer Academic Publishers, 1992.

[9] P. Duhamel and M. Vetterli. Fast Fourier transforms: a tutorial review and a state of the art. *Signal Process.*, 19(4):259–299, apr 1990.

[10] T. S. Freeman, G. M. Imirzian, E. Kaltofen and Y. Lakshman. DAGWOOD a system for manipulating polynomials given bystraight-line programs. *ACM Trans. Math. Software*, 14:218–240, 1988.

[11] S. Garg and É. Schost. Interpolation of polynomials given by straight-line programs. *Theoretical Computer Science*, 410(27-29):2659–2662, 2009.

[12] M. Gastineau and J. Laskar. Trip 1.2.26. TRIP Reference manual, IMCCE, 2012. `http://www.imcce.fr/trip/`.

[13] A. Guessoum and R. Mersereau. Fast algorithms for the multidimensional discrete Fourier transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(4):937–943, 1986.

[14] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2-nd edition, 2002.

[15] J. van der Hoeven. Relax, but don't be too lazy. *JSC*, 34:479–542, 2002.

[16] J. van der Hoeven. The truncated Fourier transform and applications. In J. Gutierrez, editor, *Proc. ISSAC 2004*, pages 290–296. Univ. of Cantabria, Santander, Spain, July 4–7 2004.

[17] J. van der Hoeven. Notes on the Truncated Fourier Transform. Technical Report 2005-5, Université Paris-Sud, Orsay, France, 2005.

[18] J. van der Hoeven. Newton's method and FFT trading. *JSC*, 45(8):857–878, 2010.

[19] J. van der Hoeven and G. Lecerf. On the bit-complexity of sparse polynomial multiplication. Technical Report, HAL, 2010. `Http://hal.archives-ouvertes.fr/hal-00476223`, accepted for publication in JSC.

[20] J. van der Hoeven and G. Lecerf. On the complexity of blockwise polynomial multiplication. In *Proc. ISSAC '12*, pages 211–218. Grenoble, France, July 2012.

[21] J. van der Hoeven and É. Schost. Multi-point evaluation in higher dimensions. Technical Report, HAL, 2010. `Http://hal.archives-ouvertes.fr/hal-00477658`, accepted for publication in AAECC.

[22] J. Johnson and X. Xu. Generating symmetric DFTs and equivariant FFT algorithms. In *ISSAC'07*, pages 195–202. ACM, 2007.

[23] S. C. Johnson. Sparse polynomial arithmetic. *SIGSAM Bull.*, 8(3):63–71, 1974.

[24] E. Kaltofen and Y. N. Lakshman. Improved sparse multivariate polynomial interpolation algorithms. In *ISSAC '88: Proceedings of the international symposium on Symbolic and algebraic computation*, pages 467–474. Springer Verlag, 1988.

[25] E. Kaltofen, Y. N. Lakshman and J.-M. Wiley. Modular rational sparse multivariate polynomial interpolation. In *ISSAC '90: Proceedings of the international symposium on Symbolic and algebraic computation*, pages 135–139. New York, NY, USA, 1990. ACM Press.

[26] E. Kaltofen, W. Lee and A. A. Lobo. Early termination in Ben-Or/Tiwari sparse interpolation and a hybrid of Zippel's algorithm. In *ISSAC '00: Proceedings of the 2000 international symposium on Symbolic and algebraic computation*, pages 192–201. New York, NY, USA, 2000. ACM Press.

[27] A Kudlicki, M. Rowicka and Z. Otwinowski. The crystallographic Fast Fourier Transform. recursive symmetry reduction. *Acta Cryst.*, A63:465–480, 2007.

[28] Victor Y. Pan. Simple multivariate polynomial multiplication. *JSC*, 18(3):183–186, 1994.

[29] V. Pan and D. Bini. *Polynomial and matrix computations*. Birkhauser, 1994.

[30] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.

[31] D. R. Stoutemyer. Which polynomial representation is best? In *Proceedings of the 1984 MACSYMA Users' Conference: Schenectady, New York, July 23–25, 1984*, pages 221–243. 1984.

[32] L. F. Ten Eyck. Crystallographic Fast Fourier Transform. *Acta Cryst.*, A29:183–191, 1973.

[33] A. Vince and X. Zheng. Computing the Discrete Fourier Transform on a hexagonal lattice. *Journal of Mathematical Imaging and Vision*, 28:125–133, 2007.