A Lattice Basis Reduction Approach for the Design of Finite Wordlength FIR Filters

Nicolas Brisebarre, Silviu-Ioan Filip, Guillaume Hanrot

A Lattice Basis Reduction Approach for the Design of Finite Wordlength FIR Filters

Nicolas Brisebarre, Silviu-Ioan Filip and Guillaume Hanrot

***Abstract*—Many applications of finite impulse response (FIR) digital filters impose strict format constraints for the filter coefficients. Such requirements increase the complexity of determining optimal designs for the problem at hand. We introduce a fast and efficient method, based on the computation of good nodes for polynomial interpolation and Euclidean lattice basis reduction. Experiments show that it returns quasi-optimal finite wordlength FIR filters; compared to previous approaches it also scales remarkably well (length 125 filters are treated in < 9s). It also proves useful for accelerating the determination of optimal finite wordlength FIR filters.**

***Index Terms*—BKZ algorithm, Euclidean Lattice, finite wordlength, FIR coefficient quantization, FIR digital filters, HKZ algorithm, lattice basis reduction, LLL algorithm, minimax approximation**

I. Introduction

THE efficient design of FIR filters has been an active research topic in Digital Signal Processing (DSP) for several decades now. An important class of practical problems is related to the discrete nature of the representation space used for storing the filter coefficients (usually fixed-point or floating-point formats).

Typical constraints are that these coefficients be representable as signed b-bit integer values (up to a scaling factor) or that they have low complexity, such as them being sums of only a small number of power-of-two terms. While the first requirement is useful for implementation on fixed-point DSP processors, the second one occurs when one wants to construct multiplierless filtering circuits. It is addressed, for instance, in [1]–[11]. Our focus will be on the first requirement, in the case of direct-form FIR filters. This coefficient quantization issue has been considered as early as [12] (see also [13], the introduction of which gives an account on related work from the 1970s).

To better illustrate the problem, consider the following simple toy example. We use it for pedagogical purposes only, as in this case, an elementary analysis of the situation suffices to determine an optimal solution. We hope it allows the reader to quickly grasp the context of our study.

We want to compute a length 5 linear phase lowpass FIR filter with 5-bit wide fixed-point coefficients and uniformly weighted passband $[0, 0.4\pi]$ and stopband $[0.5\pi, \pi]$. If we let $D(\omega) = 1$ for $\omega \in [0, 0.4\pi]$ and $D(\omega) = 0$ if $\omega \in [0.5\pi, \pi]$, we want a_0, a_1 and $a_2 \in \{-2^4, \ldots, 2^4 - 1\}$ such that the approximation error

$$\max_{\omega\in[0,0.4\pi]\cup[0.5\pi,\pi]} \left| \frac{a_0}{2^4} + \frac{a_1}{2^3}\cos(\omega) + \frac{a_2}{2^3}\cos(2\omega) - D(\omega) \right|$$

is as small as possible.

The Parks-McClellan algorithm [14], for instance, outputs an optimal infinite precision frequency response $H^*(\omega) = 0.430\cdots + 0.860\cdots\cos(\omega) + 0.069\cdots\cos(2\omega)$ and corresponding approximation error $E^* = 0.360\cdots$. If we round each coefficient towards the nearest number of the form $k/2^3$ (or $k/2^4$ for the constant coefficient), we obtain $H_{\text{naive}}(\omega) = \frac{7}{2^4} + \frac{7}{2^3}\cos(\omega) + \frac{1}{2^3}\cos(2\omega)$, and a corresponding error $E_{\text{naive}} = 0.4375$. On the other hand, the approach we introduce in the subsequent sections returns $H(\omega) = \frac{8}{2^4} + \frac{6}{2^3}\cos(\omega) + \frac{1}{2^3}\cos(2\omega)$, which improves the error to $E = 0.375$. Using, for instance, a mixed integer linear programming (MILP) routine, we obtain that H is actually an optimal 5-bit coefficient frequency response H_{opt} and E is the optimal approximation error E_{opt}.

This example displays two interesting facts:

- The straightforward approach, which we call "naive rounding", consisting in rounding each coefficient of the infinite precision response to the nearest coefficient fitting the imposed constraints, may, even in very simple cases, yield results which are far from optimal (for instance, [15] mentions an example where a 30 dB improvement is observed).
- Another natural approach is to consider all the possible responses that we get by rounding up or down the coefficients of the infinite precision response. A first major drawback of this idea is that the number of possibilities is exponential in the degree of the filter. Moreover, our toy example shows that it is possible that none of them yields an optimal response, as was already noticed in [4].

In general, finding an optimal finite wordlength direct form FIR filter proves to be computationally expensive as the degree increases, with exact solvers making use of MILP techniques, sometimes combined with clever branch-and-bound strategies [1]–[3], [5], [13], [15]–[19]. To successfully cope with larger degrees, faster heuristics have also been proposed. They produce results that are, in many cases, quasi-optimal and can help speed up exact solvers. For instance, the approach introduced in [20], based on error spectrum shaping techniques, is routinely used inside MATLAB™ for FIR coefficient

N. Brisebarre and G. Hanrot are with CNRS, ÉNS de Lyon, Inria, Université Claude Bernard Lyon 1, Laboratoire LIP (UMR 5668), 15 parvis René-Descartes, F-69007 Lyon, France.

E-mail: nicolas.brisebarre@ens-lyon.fr,guillaume.hanrot@ens-lyon.fr

S-I. Filip is with CNRS, ÉNS de Lyon, Inria, Université Claude Bernard Lyon 1, Laboratoire LIP (UMR 5668), Lyon, France and Mathematical Institute, University of Oxford, Oxford, UK.

E-mail: silviu.filip@inria.fr

This work was partly supported by the FastRelax project of the French *Agence Nationale de la Recherche*.

quantization. The methods developed in [4] (see also [21] for a variant) and [22] also produce very good results.

The rest of the paper introduces a novel and efficient method for designing quasi-optimal direct form FIR filters with fixed-point and/or floating-point coefficients. It is based on previous work for doing machine-efficient polynomial approximation of functions [23] combined with techniques that yield families of very good interpolation nodes [24]–[29]. As the filter degree gets higher, our approach improves significantly on the state of the art:

- As illustrated by Section VI, it returns results that are most of the time better than those computed by competing methods;
- When compared to exact MILP-based approaches (which are executed to completion or only for a limited time, depending on the degree), we often find the same result or a better one[1]: this happens for 8 out of the 15 examples discussed in Section VI-A;
- Further, our method is very robust and fast, especially when dealing with high degree designs.

Thanks to these features, we believe that it will prove very useful as a fast preliminary accelerator for exact solvers based on MILP. Finally, our tool is available as an open source C++ implementation[2].

A. Sketch of our approach and outline of the paper

As we will show in Section II, the problem that we address can be formulated as an MILP question, constructed from a sufficiently dense discretization of the target filter bands. The first key aspect of our work is the modeling of the MILP instance as a Closest Vector Problem (CVP), a fundamental question in the study of Euclidean lattices[3] [30]. In particular, we take advantage of very efficient algorithms for computing approximate (and yet, at least in the filter design setting, often close to optimal) solutions to a CVP.

The CVP instance that we use is also constructed from a discretization of the filter frequency bands. For it to be effective, the following constraints are important:

- the number of discretization points should be as small as possible, both to produce a tractable CVP question and for a technical reason to be made clear in Section V;
- the discretization points should be chosen so that the CVP instance models as faithfully as possible the initial MILP problem.

As a second key idea, we propose several methods to compute such a discretization. These points are analogues to the so-called Chebyshev nodes on an interval and give rise to excellent polynomial approximations on the bands of the filter.

Then, we use an idea due to R. Kannan [31] to compute a good approximate solution to this CVP. A simple additional trick from [23] applied to this solution yields a certain frequency response H: in practice, its coefficients satisfy the requested constraints and offer a very good approximation to the ideal frequency response.

After giving a precise statement of the problem that we want to solve in Section II, Section III presents the discretizations that we use, while Section IV recalls the algorithmic questions attached to Euclidean lattices that we face in our approach and some state-of-the art algorithms to address them. They are later used for detailing our method in Section V. Examples and a comparison to other approaches are discussed in Section VI, followed by concluding remarks in Section VII.

II. The finite wordlength setting

We first recall some well-known ideas related to linear-phase FIR filter design [32]–[34][4]. Such a process is typically carried out in the frequency domain. In case of an N-tap causal filter with real valued impulse response $\{h[n]\}$, the corresponding frequency response we want to optimize is

$$\begin{aligned} H(\omega) &= \sum_{k=0}^{N-1} h[k] e^{-i\omega k} \\ &= G(\omega) e^{i\left(\frac{L\pi}{2} - \frac{N-1}{2}\omega\right)}, \end{aligned}$$

where G is a real valued function and L is 0 or 1. Traditionally, there are four such types of FIR filters considered in practice, labeled from I through IV; they depend on the parity of the filter length N and on the symmetry of h (positive for $L = 0$ and negative for $L = 1$). We can express $G(\omega)$ as $Q(\omega)P(\omega)$, where $P(\omega)$ is of the form:

$$P(\omega) = \sum_{k=0}^{n} p_k \cos(k\omega).$$

Depending on the filter type, $Q(\omega)$ is either $1, \cos(\omega/2), \sin(\omega)$ or $\sin(\omega/2)$. We have $n = \lfloor N/2 \rfloor$ and there are simple formulas linking the $h[k]$'s and the p_k's (see for instance [33, Ch. 15.8–15.10]).

Remark 1. *If we consider the change of variable $x = \cos(\omega)$, $P(\omega) = \sum_{k=0}^{n} p_k \cos(k\omega)$ is in fact a polynomial of degree at most n in $\cos(\omega)$ expressed in the basis of Chebyshev polynomials $(T_k)_{0 \leqslant k \leqslant n}$ of the first kind,* i.e.,

$$P(\omega) = \sum_{k=0}^{n} p_k T_k(\cos(\omega)) = \sum_{k=0}^{n} p_k T_k(x).$$

It will sometimes be convenient to see our problems in the language of polynomial approximation. When this is the case, x will be the approximation variable and $X \subseteq [-1, 1]$ the transformed domain associated to Ω i.e., $\cos \Omega$.

While the focus of our presentation is on type I filters (hence $Q \equiv 1$), adapting it to the other three cases is straightforward.

The optimal length N frequency response with infinite precision coefficients is the solution of the following:

Problem 1 (Equiripple (or minimax) FIR filter design)**.** *Let Ω be a compact subset of $[0, \pi]$ and $D(\omega)$ an ideal frequency response, continuous on Ω. For a given filter degree $n \in \mathbb{N}$,*

[1] in the time limited case.

[2] See https://github.com/sfilip/fquantizer.

[3] Note that there is no link between Euclidean lattices and lattice wave digital filters, as the one used in [7] for instance.

[4] A part of this section follows Section II of [19].

we want to determine $P^*(\omega) = \sum_{k=0}^{n} p_k^* \cos(\omega k)$ *such that the weighted error function* $E^*(\omega) = W(\omega)\left(P^*(\omega) - D(\omega)\right)$ *has* ***minimum*** *uniform norm*

$$\|E^*\|_{\infty,\Omega} = \sup_{\omega\in\Omega} |E^*(\omega)|,$$

with the weight function W *continuous and positive over* Ω.

There exists a unique solution to this problem [35, Ch. 3.4]:

Theorem 1 (Alternation theorem)**.** *A necessary and sufficient condition for* $P(\omega)$ *to be the* ***unique*** *transfer function of degree at most* n *that minimizes the weighted approximation error* $\delta_\Omega = \|E(\omega)\|_{\infty,\Omega}$ *is that* $E(\omega) = W(\omega)\left(P(\omega) - D(\omega)\right)$ *has* ***at least*** $n+2$ *equioscillating extremal frequencies over* Ω*;* i.e.*, there exist at least* $n+2$ *values* ω_k *in* Ω *such that* $\omega_0 < \omega_1 < \ldots < \omega_{n+1}$ *and*

$$E(\omega_k) = -E(\omega_{k+1}) = \lambda(-1)^k \delta_\Omega, \qquad k = 0, \ldots, n,$$

where $\lambda \in \{\pm 1\}$ *is fixed.*

The usual way to solve Problem 1 is to use, as already mentioned in Section I, the Parks-McClellan version of the Remez algorithm.

The finite wordlength version of Problem 1 that is of interest here, and also in practice, proves to be much harder to solve in general. For fixed-point coefficient quantization we can basically consider the impulse response coefficients to be scaled integers. If we want to use b-bit fixed-point values, we view them under the form m/s, where m is an integer in $I_b = \{-2^{b-1}, \ldots, -1, 0, 1, \ldots, 2^{b-1}-1\}$ and s is a *fixed* scaling factor, which, in many cases, is a power of 2. Let $\varphi_0(\omega) = 1/s$ and $\varphi_k(\omega) = 2\cos(k\omega)/s$ for $k = 1, \ldots, n$. We then have:

Problem 2 (Finite wordlength minimax approximation)**.** *Consider* Ω, D, n *and* W *defined as in Problem 1. Determine* $P(\omega) = \sum_{k=0}^{n} m_k \varphi_k(\omega)$*, where* $m_k \in I_b$ *for* $k = 0, \ldots, n$*, such that the error*

$$\sup_{\omega\in\Omega} |W(\omega)\left(P(\omega) - D(\omega)\right)| \tag{1}$$

is ***minimal****.*

This problem always has a (not necessarily unique) solution. Expressed as an optimization question, it becomes:

$$\begin{aligned} \text{minimize} \quad & \delta \\ \text{subject to} \quad & W(\omega)\left(\sum_{k=0}^{n} m_k\varphi_k(\omega) - D(\omega)\right) \leqslant \delta,\ \omega\in\Omega, \\ & W(\omega)\left(D(\omega) - \sum_{k=0}^{n} m_k\varphi_k(\omega)\right) \leqslant \delta,\ \omega\in\Omega, \\ & m_k \in I_b, k = 0, \ldots, n. \end{aligned} \tag{2}$$

In practice, Ω is generally replaced with a finite discretization Ω_d, giving rise to a MILP instance. A number of points equal to $16n$ usually suffices, according to [36].

The scaling factor s, interpreted as the filter gain, can also be a design parameter. Optimizing for s as well can improve the quality of the quantization error (1) by a small factor. The caveat is that finding the best s is nontrivial [15], [36], [37].

More general quantization problems where coefficients have different formats from one another (be it fixed-point of floating-point) can also be expressed using the framework of Problem 2, with the difference being that each coefficient will have its own power of 2 scaling factor $s_k, k = 0, \ldots, n$. Nevertheless, for simplicity, in the sequel we will only consider fixed-point quantization examples with a uniform scaling factor s.

III. Discretization of the problem

The first step of our approach is to discretize Problem 2 with (almost) as few points as possible, in order to speed up computation and also because our lattice-based approach requires a coarse discretization in order to return relevant results (we will make this point clear at the beginning of Section V). We pick ℓ points $\omega_0, \ldots, \omega_{\ell-1}$ in Ω, with $\ell = n+1$ or slightly larger than $n+1$, and we want to determine a $P(\omega) = \sum_{k=0}^{n} m_k\varphi_k(\omega)$, where $m_k \in I_b$ for $k = 0, \ldots, n$, such that the error

$$\max_{j=0,\ldots,\ell-1} |W(\omega_j)\left(P(\omega_j) - D(\omega_j)\right)|$$

is as small as possible. A question immediately arises: how can we force this discretized version to be as faithful as possible to the initial Problem 2?

Choosing appropriate points ω_i (or $x_i = \cos(\omega_i)$) is indeed critical for obtaining very good results [23]. We propose three complementary alternatives that work well together in our practical experimentations. The idea underlying the first two choices is to force the finite wordlength transfer function to mimic the minimax approximation P^*, whereas the third one corresponds to a relevant choice for the initialization of the Parks-McClellan algorithm to determine P^*.

A. Alternating extrema of the minimax error function

Inspired by the characterization provided by Theorem 1, the discretization Ω_e we suggest is a set of $n+2$ frequencies that yield $n+2$ alternating extrema for the error function E^*.

There are some cases where the number of frequencies corresponding to these alternating extrema is larger than $n+2$. Therefore, we need an effective way to select exactly $n+2$ among them: the Parks-McClellan algorithm iteratively constructs such a list.

Remark 2. *In practice, we use the implementation of the Parks-McClellan algorithm presented in [38] and available from https://github.com/sfilip/firpm: it computes* P^* *and a list of* $n+2$ *nodes where* E^* *equialternates, our* Ω_e*.*

B. "Zeros" of the minimax error function

This strategy is the analogue of one of the choices from [23]. And yet, there is a significant difference in our setting. Actually, in the case of a single interval $[a, b]$, Theorem 1 and the intermediate value theorem ensure that there exist, at least, $n+1$ points $x_0, \ldots, x_n$ in $[a, b]$ such that $E^*(x_i) = 0$ for $i = 0, \ldots, n$. Unfortunately, we do not have any such guarantee

in the present multi-interval setting: the number of zeros might be less than $n+1$ so we complete the list by picking points that are half-way the endpoints of consecutive bands. Note that the latter points do not belong to Ω and we will explain in Section V (see discussion after Remark 7) how we use them. We execute the following:

Algorithm 1 "Zeros" of the minimax error discretization

Input: the minimax approximation P^*, ideal response D, weight W, transformed domain X

Output: an $\ell \geqslant n+1$-element discretization Ω_z of $[0,\pi]$ made of zeros of E^* and, if necessary, points that are half-way the extremities of the bands making up $X = \cos\Omega$

// Take the zeros of E^* over X

1: $X_z \leftarrow \text{roots}(E^*, X)$

// if the number of zeros is less than $n+1$,

// take the midpoints of the transition bands

// making up X (*i.e.,* $X = \dot{\cup}_{j=1}^{k}[x_0^{(j)}, x_1^{(j)}]$)

// with $x_1^{(j)} < x_0^{(j+1)}$ for $j = 0, \ldots, k-1$

2: $j \leftarrow 1$

3: **while** $|X_z| < n+1$ **do**

4: $\quad X_z \leftarrow X_z \cup \left\{(x_1^{(j)} + x_0^{(j+1)})/2\right\}$

5: $\quad j \leftarrow j+1$

6: **end while**

7: $\Omega_z \leftarrow \arccos(X_z)$

Remark 3. *One can prove that if a filter has p bands then the output discretization has at least $n+1$ points and no more than $n+p-1$ points. In the practical cases presented in Section VI, this discretization has exactly $n+1$ points (2 band case) or at most $n+2$ points (3 band case).*

C. Approximate Fekete points

The second author has previously used this last approach as a numerically robust way of initializing the Parks-McClellan algorithm [38, Sec. 4.1–4.2]. We now briefly recall it.

For any compact set $X \subset \mathbb{R}$, a set of Fekete points of degree $n+2$ are the elements of a set $\{t_0, \ldots, t_{n+1}\} \subset X$ which maximize the absolute value of the Vandermonde-like determinant $|w(y_i)T_j(y_i)|_{0\leqslant i,j\leqslant n+1}$, with $w(x) = W(\arccos(x))$, where W is the weight function used in the statements of Problems 1 and 2 and the T_j's are the first $n+2$ Chebyshev polynomials of the first kind.

Unfortunately, computing them when X is not an interval is difficult in general. Still, *approximate* versions of such nodes, called Approximate Fekete Points (AFP), can be determined efficiently. The idea is to replace X by a suitable discretization $Y \subseteq X$ with $m+1 \geqslant n+2$ elements and extract Fekete points of Y. One good choice is that of so-called *weakly admissible meshes* [25]. The example we use consists of taking Y to be the union of the $(n+1)$-th order Chebyshev nodes of second kind

$$\nu_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{k\pi}{n+1}\right), k = 0, \ldots, n+1,$$

scaled to each interval that makes up X. Even in this case, obtaining Fekete points over Y is an NP-hard problem [39]. We can, however, use an effective greedy algorithm based on column pivoting QR [24], [26]–[28]:

Algorithm 2 Approximate Fekete points discretization

Input: discretized subset $Y = \{y_0, \ldots, y_m\} \subseteq X$

Output: an $n+2$ distinct element set $X_{\text{afp}} \subset Y$ s.t. the Vandermonde-like submatrix $\mathbf{V}(X_{\text{afp}})$ generated by the elements of X_{afp} has a large volume

// Initialization. $\mathbf{V}(Y)$ is the Vandermonde-like matrix

// $(w(y_i)T_j(y_i))_{0\leqslant i\leqslant m, 0\leqslant j\leqslant n+1}$

1: $\mathbf{A} \leftarrow \mathbf{V}(Y), b \in \mathbb{R}^{n+2}, b \leftarrow (1 \cdots 1)^t$

// QR-based Linear System Solver

2: using a column pivoting-based QR solver (via an equivalent of LAPACK's DGEQP3 routine [40]), find $\mathbf{w} \in \mathbb{R}^{m+1}$, a solution to the underdetermined system $\mathbf{b} = \mathbf{A}^t\mathbf{w}$.

// Subset Selection

3: take X_{afp} as the set of elements from Y whose corresponding terms inside $\mathbf{w}$ are different from zero, that is, if $y_i \in Y$ and $w_i \neq 0$, then $y_i \in X_{\text{afp}}$.

Similar to the previous two choices, we will in fact work with $\Omega_{\text{afp}} = \arccos(X_{\text{afp}})$.

Remark 4. *There are solid theoretical arguments for the AFP approach. They are based on the study of so-called Lebesgue constants and can be found in [38], [41].*

Remark 5. *Another alternative of good discretization grids is to take them according to the so-called equilibrium distribution for weighted minimax approximation on X [42]–[44]. We have not tested this idea here because it is more involved to set up than the approximate Fekete points method.*

IV. The closest vector problem and lattice basis reduction

Let $\|\cdot\|_2$ and $\|\cdot\|_\infty$ denote the usual Euclidean and supremum norms over $\mathbb{R}^\ell$ (i.e., $\|\mathbf{v}\|_2 = \left(\sum_{i=0}^{\ell-1} v_i^2\right)^{1/2}$ and $\|\mathbf{v}\|_\infty = \max_{i=0}^{\ell-1}|v_i|$).

The Euclidean lattice structure is a fundamental component of various problems in Mathematics, Computer Science or Crystallography [30], [45]–[49]:

Definition 1. A lattice $L \subset \mathbb{R}^\ell$ is the set of integer linear combinations of a family $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of $\mathbb{R}$-linearly independent vectors of $\mathbb{R}^\ell$. We shall then say that $(\mathbf{b}_i)_{1\leqslant i\leqslant d}$ is a **basis** of L, and that $d(\leqslant \ell)$ is the **dimension** of L.

Finally, we define one of the most important algorithmic problems concerning Euclidean lattices:

Problem 3 (Closest vector problem (CVP))**.** *Let $||\cdot||$ be a norm on $\mathbb{R}^\ell$. Given as input a basis of a lattice $L \subset \mathbb{R}^\ell$ of dimension d, $d \leqslant \ell$, and $x \in \mathbb{R}^\ell$, find $y \in L$ s.t. $||x-y|| = \min\{||x-v||, v \in L\}$.*

In order to find a good solution to Problem 2, we address, after a suitable discretization, System (2). Up to a very mild relaxation of the constraint $m_k \in I_b$ to $m_k \in \mathbb{Z}$, the latter is actually a CVP in the supremum (or $\|\cdot\|_\infty$) norm. Although

CVP_∞ can be attacked using MILP techniques, we propose to approximate it by the same CVP, but in the Euclidean (or $\|\cdot\|_2$) norm, a problem which we shall denote by CVP_2. Even though, from a complexity-theoretic point of view, both problems are NP-hard, this change of perspective is motivated by the existence of a wealth of practical (i.e., efficient and with good performances) approximation algorithms to deal with this kind of problems.

The algorithms addressing a CVP_2 usually rely on a preprocessing step called *lattice basis reduction*, a notion that we briefly discuss in Subsection IV-A, together with three main tools for performing it: the LLL, BKZ and HKZ algorithms.

An idea due to Kannan [31], [50] will then allow us to use lattice basis reduction to obtain good approximate solutions of CVP_2 in the setting under study.

We also point out that the LLL algorithm has already been used in the digital filter design context, but in a different way [19]: in order to accelerate the MILP search for an optimal filter, Kodek determines, thanks to LLL[5], a lower bound on the optimal approximation error, which is used as a first guess for δ in (2). Moreover, to the best of our knowledge, the present text discusses the first use of the BKZ, HKZ and Kannan embedding algorithms for a filter design purpose.

A. Lattice basis reduction

A lattice of dimension $d \geqslant 2$ has infinitely many bases. Among those bases, the ones which are made up of short and somewhat orthogonal vectors are usually considered good, whereas the other ones are deemed bad. Lattice basis reduction algorithms allow one to move from a bad basis to a good one, an important preconditioning step in solving lattice problems.

We shall consider three different lattice basis reduction algorithms, namely LLL, BKZ and HKZ. We shall not describe them nor their complete output – this is a rather technical topic and is out of the scope of this paper; we refer the reader to [30] for a detailed discussion of lattice basis reduction algorithms.

Regarding their respective performances, the bottom line is that in terms of quality, LLL $\leqslant$ BKZ $\leqslant$ HKZ, while in terms of efficiency, unsurprisingly, LLL $\geqslant$ BKZ $\geqslant$ HKZ.

Remark 6. *Actually, it seems that our input lattices and lattice bases are much better conditioned than expected, as all these algorithms seem to have a much better performance than expected on both quality and efficiency – HKZ-reducing a random lattice of dimension greater than 60 is, as of today, a considerable task, while we were able to do it on instances of degree 62 resulting from our problems in less than two minutes (see Section VI).*

B. The CVP problem & Kannan's embedding

When facing a CVP_2 instance, one may choose to resort to exact, exponential-time approaches [51]. Even though they are efficient enough to deal with moderate size problems, we have found them to provide results which are not superior to our more efficient method. This might be explained by two reasons :

- as already observed, the lattice problems raised by digital filter design seem to be very well-conditioned, so that approximation algorithms tend to give much better results than expected (actually, often, optimal results).
- since CVP_2 is already an approximate formulation of the problem to be solved, seeking an exact solution for it is not very meaningful. Actually, our main claim is that very good solutions to CVP_2 correspond to very good solutions to (2) (and Section VI provides evidence for that), whereas experience shows that optimal CVP_2 solutions do not, in general, correspond to optimal solutions of (2).

We use Kannan's embedding technique[6] [31] in the following way: given a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of L and the target vector $\mathbf{t}$, we form the $(d+1)$-dimensional lattice L' of $\mathbb{R}^{\ell+1}$ generated by the vectors $\mathbf{b}'_i := \begin{pmatrix} \mathbf{b}_i \\ 0 \end{pmatrix}$ and $\mathbf{t}' := \begin{pmatrix} \mathbf{t} \\ \gamma \end{pmatrix}$, where γ is a real parameter to be chosen later on. If $\mathbf{u}$ is a short vector of the lattice L', there exist $u_1, \ldots, u_d, u_{d+1}$ such that $\mathbf{u} = \sum_{k=1}^{d} u_k \mathbf{b}'_k + u_{d+1}\mathbf{t}'$; hence $\|\mathbf{u}\|_2^2 = \left\|\sum_{k=1}^{d} u_k \mathbf{b}_k + u_{d+1}\mathbf{t}\right\|_2^2 + \gamma^2 u_{d+1}^2$.

If γ is taken large enough and $\mathbf{u}$ is short, we must have have $u_{d+1} = \pm 1$, meaning that $\mp \sum_{k=1}^{d} u_k \mathbf{b}_k \in L$ is close to the vector $\mathbf{t}$. In practice, $\mathbf{u}$ is found as a vector of a reduced basis of L'. The (heuristic) choice $\gamma = \max_{k=1}^{d} \|\mathbf{b}_k\|_2$ always yielded a vector with $u_{d+1} = \pm 1$ in all experiments (see [41, Ch. 4.3.4] for a theoretical justification).

The entire procedure is summarized in Algorithm 3.

V. Lattice-based filter design

We start by discretizing Problem 2: we pick $\ell (\geqslant n+1)$ points $\omega_0, \ldots, \omega_{\ell-1}$ in Ω and search for an approximation $P(\omega) = \sum_{k=0}^{n} m_k \varphi_k(\omega)$, where $m_k \in I_b$ for $k = 0, \ldots, n$, such that the vectors

$$\begin{pmatrix} W(\omega_0) \sum_{k=0}^{n} m_k \varphi_k(\omega_0) \\ W(\omega_1) \sum_{k=0}^{n} m_k \varphi_k(\omega_1) \\ \vdots \\ W(\omega_{\ell-1}) \sum_{k=0}^{n} m_k \varphi_k(\omega_{\ell-1}) \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} W(\omega_0) D(\omega_0) \\ W(\omega_1) D(\omega_1) \\ \vdots \\ W(\omega_{\ell-1}) D(\omega_{\ell-1}) \end{pmatrix}$$

are as close as possible with respect to $\|\cdot\|_\infty$. In other words, we need to find $(m_0, \cdots, m_n) \in {I_b}^{n+1}$ such that

$$m_0 \underbrace{\begin{pmatrix} W(\omega_0)\varphi_0(\omega_0) \\ W(\omega_1)\varphi_0(\omega_1) \\ \vdots \\ W(\omega_{\ell-1})\varphi_0(\omega_{\ell-1}) \end{pmatrix}}_{\mathbf{b}_0} + \cdots + m_n \underbrace{\begin{pmatrix} W(\omega_0)\varphi_n(\omega_0) \\ W(\omega_1)\varphi_n(\omega_1) \\ \vdots \\ W(\omega_{\ell-1})\varphi_n(\omega_{\ell-1}) \end{pmatrix}}_{\mathbf{b}_n}$$

and $\mathbf{t} = (W(\omega_0)D(\omega_0) \cdots W(\omega_{\ell-1})D(\omega_{\ell-1}))^t$ are as close as possible to each other, *i.e.,*

$$\text{minimize} \quad \|m_0\mathbf{b}_0 + \ldots + m_n\mathbf{b}_n - \mathbf{t}\|_\infty , \tag{3}$$

which is a CVP_∞ instance, the Euclidean lattice under consideration being $\mathbb{Z}\mathbf{b}_0 + \ldots + \mathbb{Z}\mathbf{b}_n \subset \mathbb{R}^\ell$. We approximately

[5] Kodek actually applies LLL to a lattice which is dual to ours – which is coherent with the use he makes of it.

[6] Classically [31], [50], [52] Kannan's idea is used in a much finer way, but this rough version proves to be sufficient for the problem at hand.

Algorithm 3 Kannan embedding

Input: basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of $L \subset \mathbb{R}^\ell$, target vector $\mathbf{t} \in \mathbb{R}^\ell$, a lattice basis reduction algorithm (LLL, BKZ or HKZ)
Output: $\mathbf{m} = (m_1 \cdots m_d)^t \in \mathbb{Z}^d$ s.t. $\sum_{k=1}^d m_k \mathbf{b}_k \approx \mathbf{t}$, change of basis matrix $\mathbf{U} \in \mathbb{Z}^{d \times d}$

$\gamma \leftarrow \max_{1 \leqslant k \leqslant d} \|\mathbf{b}_k\|_2$
// Construct the embedded basis
$\mathbf{B}' \leftarrow \begin{pmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_d & \mathbf{t} \\ 0 & 0 & \cdots & 0 & \gamma \end{pmatrix}$
// Perform basis reduction (LLL, BKZ or HKZ) on $\mathbf{B}'$
// $\mathbf{C}' = \left(c'_{i,j}\right) \in \mathbb{R}^{(\ell+1)\times(d+1)}$ is the reduced basis
// $\mathbf{U}' = \left(u'_{i,j}\right) \in \mathbb{Z}^{(d+1)\times(d+1)}$ is the change of basis
// matrix (*i.e.,* $\mathbf{C}' = \mathbf{B}'\mathbf{U}'$)
$(\mathbf{C}', \mathbf{U}') \leftarrow \text{LatticeReduce}(\mathbf{B}')$
// Get change of basis matrix for $\mathbf{B}$ (*i.e.,* first d
// rows and columns of $\mathbf{U}'$)
$\mathbf{U} \leftarrow \left(u'_{i,j}\right)_{1 \leqslant i \leqslant d, 1 \leqslant j \leqslant d}$
// Extract element from last line and column of $\mathbf{C}'$
$\gamma' \leftarrow c'_{\ell+1,d+1}$
// Construct the outputs in terms of γ' and the last
// column of $\mathbf{U}'$
$s \leftarrow 1$
if $\gamma' = -\gamma$ **then**
  $s \leftarrow -1$
end if
for $i = 1$ **to** d **do**
  $m_i \leftarrow s \cdot u'_{i,d+1}$
end for

solve (3) in two steps, that we will present in more details in Subsection V-A:

- the existence of well-tuned, practical approximation algorithms in the Euclidean setting leads us to (approximately) solve the $\|\cdot\|_2$ version of (3) instead:

$$\text{minimize } \|m_0\mathbf{b}_0 + \ldots + m_n\mathbf{b}_n - \mathbf{t}\|_2, \qquad (4)$$

- using combinations of the short vectors computed during the first step, we "turn" around the approximate solution of the CVP$_2$ instance (4) in order to improve the approximate solution of the CVP$_\infty$ instance (3). In fact, as the reader will see in V-A, we use this vicinity search to directly improve our solution to Problem 2.

Note that working with (4) instead of directly solving (3) has the following important consequence. Since in $\mathbb{R}^\ell$, $\|\cdot\|_\infty \leqslant \|\cdot\|_2 \leqslant \sqrt{\ell}\|\cdot\|_\infty$, we expect in general that small vectors with respect to the $\|\cdot\|_2$ norm are also small when considering $\|\cdot\|_\infty$. Taking ℓ close to the minimal value n hence helps [23].

Remark 7. *It also proves very useful to approximate P^*, the minimax approximation, instead of D: in this case, we consider* $\mathbf{t} = \left(W(\omega_0)P^*(\omega_0) \cdots W(\omega_{\ell-1})P^*(\omega_{\ell-1})\right)^t$ *in* (3) *and* (4).

We have presented in Section III three families of discretization points:

- the alternating extrema of the minimax error function, cf. III-A, and approximate Fekete points, cf. III-C. We will use them to approximate both D and P^*;
- the "zeros" of the minimax error function, cf. III-B. Some of these points may belong to $[0, \pi] \setminus \Omega$, hence we can use this family only to approximate P^*.

Eventually, we summarize our method in the form of a pseudo-code listing in Section V-B.

A. Solving the CVP problem and a refinement trick

We saw in Section IV that (4) is an NP-hard problem, but argued that we can obtain an approximate solution quite efficiently by using a version of Kannan's embedding technique. Applying Algorithm 3 to (4) determines:

- a lattice vector $\sum_{k=0}^n m_k\mathbf{b}_k$ that is close to $\mathbf{t}$ with respect to $\|\cdot\|_2$ and hopefully close to $\mathbf{t}$ with respect to $\|\cdot\|_\infty$,
- but also a reduced basis $(\mathbf{c}_0, \ldots, \mathbf{c}_n)$ of $(\mathbf{b}_0, \ldots, \mathbf{b}_n)$ (by applying the LLL, BKZ or HKZ algorithm and retrieving the change of basis matrix $\mathbf{U}$).

Since the $\mathbf{c}_i$ vectors are usually short with respect to the $\|\cdot\|_2$ norm (and consequently with the $\|\cdot\|_\infty$ norm as well), we can use them to search in the vicinity of $\sum_{k=0}^n m_k\mathbf{b}_k$ with the goal of potentially improving the quality of our result with respect to Problem 2.

In the function approximation setting, this idea is described in [53, p. 128]. Here, it translates to the following strategy:

Algorithm 4 Vicinity search

Input: vector of discretized coefficients $\mathbf{m} \in \mathbb{Z}^{n+1}$, ideal response D, weight W, basis functions $(\varphi_k)_{0 \leqslant k \leqslant n}$, change of basis matrix $\mathbf{U} \in \mathbb{Z}^{(n+1)\times(n+1)}$.
Output: vector of coefficients $\mathbf{m}' \in \mathbb{Z}^{n+1}$.

// Get initial approximation error & coefficients
$E_{\min} \leftarrow \|W(\omega)\left(\sum_{k=0}^n m_k\varphi_k(\omega) - D(\omega)\right)\|_{\Omega,\infty}$
$\mathbf{m}' \leftarrow \mathbf{m}$
// Try to improve this initial approximation by selecting
// two directions d_0 and d_1 in which to search
for $d_0 = 0$ **to** 8 **do**
  for $d_1 = d_0 + 1$ **to** n **do**
    $\mathbf{u} \leftarrow \mathbf{U}_{0:n,d_0}, \mathbf{v} \leftarrow \mathbf{U}_{0:n,d_1}$
    // Update coefficients in these directions if it leads
    // to smaller approximation errors
    for $\varepsilon_0, \varepsilon_1 \in \{0, \pm 1\}$ **do**
      $\mathbf{m}'' \leftarrow \mathbf{m}$
      for $i = 0$ **to** n **do**
        $m''_i \leftarrow m''_i + u_i\varepsilon_0 + v_i\varepsilon_1$
      end for
      $E \leftarrow \|W(\omega)\left(\sum_{k=0}^n m''_k\varphi_k(\omega) - D(\omega)\right)\|_{\Omega,\infty}$
      if $E < E_{\min}$ **then**
        $E_{\min} \leftarrow E$
        $\mathbf{m}' \leftarrow \mathbf{m}''$
      end if
    end for
  end for
end for

We limit the value of d_0 to a small constant (here to eight) in order to keep the execution time reasonable (line 11 takes $O(n^2)$ operations, and lines 3, 4 & 6 tell us that we execute it $O(n)$ times), but also because in practice we did not notice any significantly improved results by taking a larger search space. Increasing the number of search directions to three also helps, but we found the computational cost usually outweighs the improvements.

Remark 8. *The lattice basis reduction approaches we use provide results that are hopefully close to the actual CVP solution, but they are not necessarily optimal. And yet, there are two quite encouraging points:*

- *the experiments in [41, §4.4.1] show that the output reduced basis is actually of excellent quality;*
- *the theoretical estimate in the same text [41, §4.4.3] supports our practical approach.*

B. A pseudo-code synthesis of our approach

We can synthesize our whole approach as follows:

Algorithm 5 Lattice-based finite wordlength coefficient design

Input: degree n, ℓ-point discretization $\{\omega_0, \ldots, \omega_{\ell-1}\}$ of Ω ($\ell \geqslant n+1$), scaling factor s, minimax response P^*, weight W, a lattice basis reduction algorithm LBR $\in$ {LLL, BKZ, HKZ}.

Output: fixed-point filter coefficients $h[k], k = 0, \ldots, 2n$.

// Construct the appropriate basis functions $\varphi_0, \ldots, \varphi_n$

1: $\varphi_0(\omega) \leftarrow \frac{1}{s}$
2: **for** $k = 1$ **to** n **do**
3: $\quad \varphi_k(\omega) \leftarrow \frac{2\cos(k\omega)}{s}$
4: **end for**

// Construct the lattice basis vectors

5: **for** $i = 0$ **to** n **do**
6: $\quad \mathbf{b}_i \leftarrow (W(\omega_0)\varphi_i(\omega_0) \cdots W(\omega_{\ell-1})\varphi_i(\omega_{\ell-1}))^t$
7: **end for**

// Construct our two possible target vectors

8: $\mathbf{t} \leftarrow (W(\omega_0)P^*(\omega_0) \cdots W(\omega_{\ell-1})P^*(\omega_{\ell-1}))^t$
9: $\mathbf{t}' \leftarrow (W(\omega_0)D(\omega_0) \cdots W(\omega_{\ell-1})D(\omega_{\ell-1}))^t$

// Find approximate CVP solutions using Algorithm 3

10: $(\mathbf{m}, \mathbf{U}) \leftarrow$ KannanEmbedding($\mathbf{B}, \mathbf{t}$, LBR)
11: $(\mathbf{m}', \mathbf{U}') \leftarrow$ KannanEmbedding($\mathbf{B}, \mathbf{t}'$, LBR)

// Use the vicinity search of Algorithm 4 to improve the
// quality of the solution

12: $\mathbf{m} \leftarrow \text{VicinitySearch}(\mathbf{m}, D, W, (\varphi_k)_{0\leqslant k\leqslant n}, \mathbf{U})$
13: $\mathbf{m}' \leftarrow \text{VicinitySearch}(\mathbf{m}', D, W, (\varphi_k)_{0\leqslant k\leqslant n}, \mathbf{U}')$
14: $E_1 \leftarrow \|W(\omega)\,(\sum_{k=0}^{n} m_k\varphi_k(\omega) - D(\omega))\|_{\Omega,\infty}$
15: $E_2 \leftarrow \|W(\omega)\,(\sum_{k=0}^{n} m'_k\varphi_k(\omega) - D(\omega))\|_{\Omega,\infty}$
16: **if** $E_2 < E_1$ **then**
17: $\quad \mathbf{m} \leftarrow \mathbf{m}'$
18: **end if**

// Retrieve the corresponding finite wordlength coefficients

19: $h[n] \leftarrow \frac{m_0}{s}$
20: **for** $k = 0$ **to** $n-1$ **do**
21: $\quad h[k] \leftarrow h[2n-k] \leftarrow \frac{m_{n-k}}{s}$
22: **end for**

TABLE I
FILTER SPECIFICATIONS CONSIDERED IN [22].

Filter	Bands	$D(\omega)$	$W(\omega)$
A	$[0, 0.4\pi]$	1	1
	$[0.5\pi, \pi]$	0	1
B	$[0, 0.4\pi]$	1	1
	$[0.5\pi, \pi]$	0	10
C	$[0, 0.24\pi]$	1	1
	$[0.4\pi, 0.68\pi]$	0	1
	$[0.84\pi, \pi]$	1	1
D	$[0, 0.24\pi]$	1	1
	$[0.4\pi, 0.68\pi]$	0	10
	$[0.84\pi, \pi]$	1	1
E	$[0.02\pi, 0.42\pi]$	1	1
	$[0.52\pi, 0.98\pi]$	0	1

Remark 9. *Lines 10 and 11 will generate the same reduced basis for* $\mathbf{B}$*, so in practice we combine them to avoid any re-computations.*

Remark 10. *We call Algorithm 5 with the two discretization choices III-A and III-C. Regarding the discretization choice III-B, we call a reduced version of Algorithm 5: we don't execute lines 9, 11, 13 and 15 that all deal with the ideal function D. We keep the best result among the results returned from these calls.*

VI. EXPERIMENTAL RESULTS

To illustrate the effectiveness of our method, we first compare it to the telescoping rounding approach introduced in [22], the error spectrum shaping method from [20] and the tree search approach of [4], based on least squares error optimization. To our knowledge, these[7] are the most efficient quasi-optimal methods that explicitly treat the finite wordlength direct-form FIR design problem. We conclude this section with an account on the practical computation time of our code.

The following batch of tests were executed on an Intel i7-3687U CPU with a 64-bit Linux-based system and a `g++` version 5.3.0 C++ compiler. As explained in Remark 9, all three discretization approaches presented in Section III are used, with the best result chosen in the end.

A. Kodek and Krisper's telescoping rounding [22]

We start with the type I FIR filter specifications given in [22, Table 1], which we reproduce for convenience, in Table I. When referring to a particular design instance, we give the specification letter, filter length and scaling factor. For example, A35/8 denotes a design problem adhering to specification A, of length $N = 35$ (meaning a degree $n = 17$ approximation), $b = 8$ bits used to store the filter coefficients (sign bit included) and an $s = 2^{b-1}$ scaling factor.

The results are highlighted in Table II:

- the first column lists the problem specification;
- the second one gives the minimax error E^* computed using the Parks-McClellan algorithm.

All remaining columns represent approximation errors of the transfer function for various coefficient quantization strategies:

[7] Unfortunately, we were not able to make an accurate comparison with the method from [21].

TABLE II
QUANTIZATION ERROR COMPARISON FOR THE FILTER SPECIFICATIONS GIVEN IN [22].

Filter	Minimax E^*	Best known finite wordlength E_{opt}	Naive rounding E_{naive}	Telescoping E_{tel}	LLL reduction E_{LLL}	BKZ reduction E_{BKZ}	HKZ reduction E_{HKZ}
A35/8	0.01595	0.02983 (M,L)	0.03266	0.03266	**0.02983**	**0.02983**	**0.02983**
A45/8	$7.132 \cdot 10^{-3}$	0.02962 (M,L)	0.03701	0.03186	**0.02962**	**0.02962**	**0.02962**
A125/21†	$8.055 \cdot 10^{-6}$	$1.077 \cdot 10^{-5}$ (M)	$1.620 \cdot 10^{-5}$	$1.179 \cdot 10^{-5}$	$\mathbf{1.161 \cdot 10^{-5}}$	$1.168 \cdot 10^{-5}$	$1.251 \cdot 10^{-5}$
B35/9	0.05275	0.07709 (M)	0.15879	**0.07854**	0.08205	0.08205	0.08205
B45/9	0.02111	0.05679 (M)	0.11719	0.06641	**0.06041**	**0.06041**	**0.06041**
B125/22†	$2.499 \cdot 10^{-5}$	$2.959 \cdot 10^{-5}$ (M)	$6.198 \cdot 10^{-5}$	$3.293 \cdot 10^{-5}$	$\mathbf{3.243 \cdot 10^{-5}}$	$3.344 \cdot 10^{-5}$	$3.344 \cdot 10^{-5}$
C35/8	$2.631 \cdot 10^{-3}$	0.01787 (M,T,L)	0.04687	**0.01787**	**0.01787**	0.01917	0.01917
C45/8	$6.709 \cdot 10^{-4}$	0.01609 (M,L)	0.03046	0.02103	**0.01609**	**0.01609**	0.02291
C125/21‡	$1.278 \cdot 10^{-8}$	$1.564 \cdot 10^{-6}$ (L)	$8.203 \cdot 10^{-6}$	$2.1 \cdot 10^{-6}$	$1.606 \cdot 10^{-6}$	$\mathbf{1.564 \cdot 10^{-6}}$	$\mathbf{1.564 \cdot 10^{-6}}$
D35/9	0.01044	0.03252 (M,L)	0.12189	0.03368	**0.03252**	0.03291	0.03291
D45/9	$2.239 \cdot 10^{-3}$	0.02612 (M)	0.10898	0.02859	**0.02706**	0.02805	**0.02706**
D125/22‡	$4.142 \cdot 10^{-8}$	$1.781 \cdot 10^{-6}$ (L)	$3.425 \cdot 10^{-5}$	$2.16 \cdot 10^{-6}$	$1.864 \cdot 10^{-6}$	$\mathbf{1.781 \cdot 10^{-6}}$	$1.826 \cdot 10^{-6}$
E35/8	0.01761	0.03299 (M)	0.04692	0.03404	**0.03349**	**0.03349**	**0.03349**
E45/8	$6.543 \cdot 10^{-3}$	0.02887 (M,L)	0.03571	0.03403	0.03167	0.03094	**0.02887**
E125/21†	$7.889 \cdot 10^{-6}$	$1.034 \cdot 10^{-5}$ (M)	$1.479 \cdot 10^{-5}$	$\mathbf{1.127 \cdot 10^{-5}}$	$1.215 \cdot 10^{-5}$	$1.245 \cdot 10^{-5}$	$1.162 \cdot 10^{-5}$

- the third column presents optimal (or best known) finite wordlength errors E_{opt}: the errors for the length 125 filters are not proven to be optimal whereas the other ten errors are. Apart from the filters marked with the ‡, they are obtained using an MILP solver (with the values obtained in [22, Table 2]). In case of the problems marked with †, note that the exact solver was stopped after a certain time limit. We also indicate the methods attaining this optimal value, with M = (time-limited) MILP, L = lattice based (this paper), T = two coefficient telescoping rounding approach [22, Sec. 4].
- column four lists the errors obtained by simply rounding the real-valued minimax coefficients to their closest values in the imposed quantization format;
- the fifth column gives the best errors from using the two coefficient telescoping rounding approach of [22, Sec. 4];
- the last three columns show the lattice-based quantization errors when choosing the LLL, BKZ and HKZ basis reduction option, respectively, when applying Algorithm 5. A default block size of 8 was used when calling BKZ.

For the last four columns, values reported in bold are the best out of the telescoping rounding approach and our approach.

We notice that lattice-based quantization gives results which are optimal (or the best known ones) in eight cases out of fifteen and the other seven cases are very close to the optimal ones. It outperforms telescoping rounding in twelve cases out of fifteen and yields the same (optimal) result in a thirteenth case: the use of the LLL option gives a better result in twelve cases and an identical (optimal) result in a thirteenth case, the use of the BKZ option gives a better result in eleven cases and the use of the HKZ option in nine cases. We remark, in particular, the good behavior of the LLL algorithm in all 15 test cases, further emphasizing the idea that the lattice bases we use are close to being reduced. Our approach seems to work particularly well when the gap between the minimax error and the naive rounding error is significant, which is the case where one is most interested in improvements over the naive rounding filter. Eventually, note that, in the C125/21 and D125/22 cases, our approach returns (in less than 8 seconds, see Subsection VI-D) results that are better than the ones provided by (time-limited) MILP tools.

B. Nielsen's error spectrum shaping approach [20]

Consider now the finite wordlength low-pass filter specification [20, Sec. V] with passband $[0, 0.4\pi]$, stopband $[0.6\pi, \pi]$, stopband attenuation $\leqslant -90$ dB and passband peak to peak ripple $\leqslant 0.01994$ dB (passband attenuation -58.8 dB). The corresponding weighting function is

$$W(\omega) = \begin{cases} 1, & \omega \in [0, 0.4\pi], \\ 10^{\frac{90-58.8}{20}}, & \omega \in [0.6\pi, \pi]. \end{cases}$$

The error spectrum shaping approach introduced in [20] and used by MATLAB™'s fixed-point filter design tools takes such a specification and produces quasi-optimal fixed-point coefficient filters that satisfy it. For instance, the `constraincoeffwl` function takes the magnitude bounds and a maximum coefficient wordlength as inputs. It produces an FIR filter with a coefficient format upper bounded by the maximal parameter value. The length of the filter is also taken to be as small as possible.

For coefficients stored on at most 15 bits, we can call

```
d = fdesign.lowpass('Fp,Fst,Ap,Ast',.4,.6,.01994,90);
Hd = design(d,'equiripple');
Hq = constraincoeffwl(Hd,15,'NoiseShaping',true);
```

Running this stochastic routine 10 times, the best result we obtained had $N = 51$, 15-bit coefficients, -95.17 dB attenuation and a 0.0148 dB ripple. Our lattice based approach with the BKZ option was able to decrease the length to $N = 47$, 15-bit coefficients, -90.66 dB attenuation and 0.0192 dB ripple. See Figure 1 for its magnitude response and Table III for the coefficients.

C. Lim and Parker's LMS criterion-based tree search approach [4]

Similarly, we can take the high-pass specification from Example 1 in [4] consisting of a $[0, 0.74\pi]$ stopband, $[0.8\pi, \pi]$ passband, 17 bit wordlength (including sign bit) coefficients, stopband attenuation $\leqslant -80$ dB and passband peak to peak ripple $\leqslant 0.1$ dB (passband attenuation -44.79 dB). The result

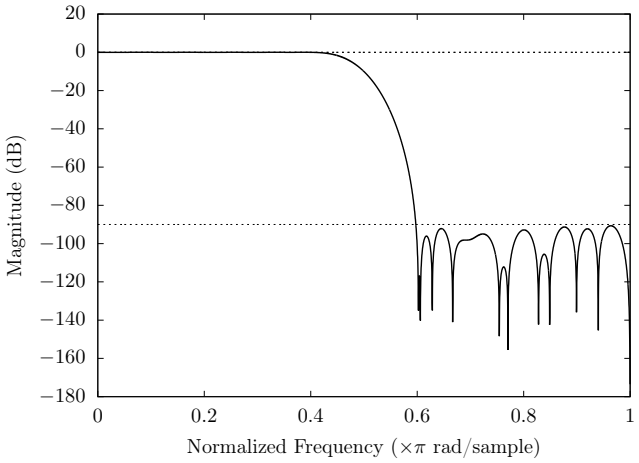

Fig. 1. The magnitude response of the finite wordlength filter produced by our lattice-based code on the example from Section VI-B.

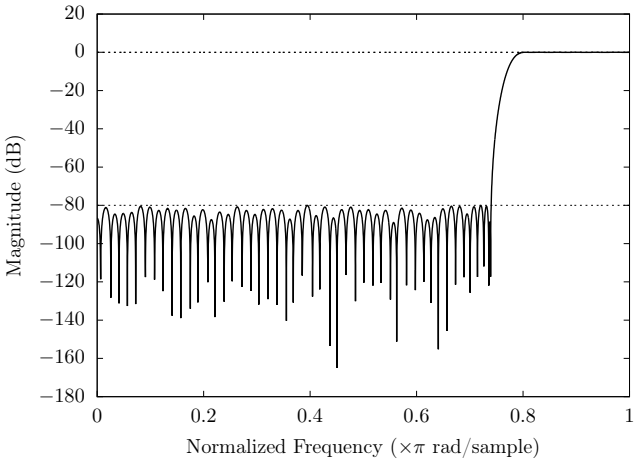

Fig. 2. The magnitude response of the finite wordlength filter produced by our lattice-based code on the example from Section VI-C.

TABLE III
IMPULSE RESPONSE FOR THE DISCRETE COEFFICIENT FILTER MEETING THE SPECIFICATIONS OF [20, SEC. 5]
$h[n] = h[46-n]$ FOR $23 \leqslant n \leqslant 46$.

Lowpass filter, $N = 47$		
Passband peak to peak ripple = 0.0192 dB		
Peak stopband attenuation = −90.66 dB		
Impulse response $\times 2^{15}$		
$h[23] = 15864$	$h[15] = -358$	$h[7] = -96$
$h[22] = 10356$	$h[14] = 628$	$h[6] = 44$
$h[21] = 508$	$h[13] = 284$	$h[5] = 56$
$h[20] = -3260$	$h[12] = -372$	$h[4] = -14$
$h[19] = -476$	$h[11] = -216$	$h[3] = -30$
$h[18] = 1738$	$h[10] = 204$	$h[2] = -2$
$h[17] = 420$	$h[9] = 150$	$h[1] = 10$
$h[16] = -1040$	$h[8] = -102$	$h[0] = 4$

they obtain has degree $n = 60$, passband ripple 0.08 dB and stopband attenuation −80.3 dB, leading to the weight function

$$W(\omega) = \begin{cases} 10^{\frac{80-44.79}{20}}, & \omega \in [0, 0.74\pi], \\ 1, & \omega \in [0.8\pi, \pi]. \end{cases}$$

By using LLL as the basis reduction algorithm, we obtain a similar filter of degree $n = 60$, with passband ripple 0.091 dB and stopband attenuation −80.03 dB in under 9 seconds; the resulting magnitude response is given in Figure 2, with coefficients in Table IV.

D. Computation time in practice

Analyzing the runtime of Algorithm 5 is dependent on the basis reduction approach used for the Kannan embedding portion of the computation (lines 10 and 11) and the short vector-based vicinity search (lines 12 and 13), which takes $O(n^3)$. Actually, the lattice bases that appear in our problems are usually quite close to being reduced, making the Kannan embedding portion of the code much faster than the vicinity search (at least when using LLL and BKZ reductions). This can be seen for instance in Table V, which breaks down the execution time of our code, applied to the three discretizations mentioned in Section III, on the examples of Table II:

TABLE IV
IMPULSE RESPONSE FOR THE DISCRETE COEFFICIENT FILTER MEETING THE SPECIFICATIONS OF [3, EXAMPLE 1]
$h[n] = h[120-n]$ FOR $61 \leqslant n \leqslant 120$.

Highpass filter, $N = 121$		
Passband peak to peak ripple = 0.091 dB		
Peak stopband attenuation = −80.03 dB		
Impulse response $\times 2^{16}$		
$h[60] = 14705$	$h[39] = -593$	$h[18] = -139$
$h[59] = -13508$	$h[38] = 148$	$h[17] = 131$
$h[58] = 10268$	$h[37] = 301$	$h[16] = -71$
$h[57] = -5914$	$h[36] = -555$	$h[15] = -6$
$h[56] = 1631$	$h[35] = 529$	$h[14] = 64$
$h[55] = 1536$	$h[34] = -274$	$h[13] = -81$
$h[54] = -3010$	$h[33] = -65$	$h[12] = 57$
$h[53] = 2817$	$h[32] = 325$	$h[11] = -9$
$h[52] = -1502$	$h[31] = -402$	$h[10] = -39$
$h[51] = -135$	$h[30] = 287$	$h[9] = 67$
$h[50] = 1358$	$h[29] = -58$	$h[8] = -68$
$h[49] = -1749$	$h[28] = -167$	$h[7] = 46$
$h[48] = 1308$	$h[27] = 289$	$h[6] = -14$
$h[47] = -377$	$h[26] = -269$	$h[5] = -15$
$h[46] = -559$	$h[25] = 136$	$h[4] = 32$
$h[45] = 1095$	$h[24] = 35$	$h[3] = -35$
$h[44] = -1063$	$h[23] = -163$	$h[2] = 28$
$h[43] = 566$	$h[22] = 199$	$h[1] = -17$
$h[42] = 107$	$h[21] = -141$	$h[0] = 8$
$h[41] = -634$	$h[20] = 29$	
$h[40] = 804$	$h[19] = 80$	

- the first column lists the problem specification;
- the second one is the time required for finding the three discretizations;
- the following three columns show the computation time of the Euclidean lattice part of the code (lattice basis reduction and approximate CVP solving, which correspond to lines 10 and 11 in Algorithm 5) when choosing the LLL, BKZ and HKZ basis reduction option, respectively;
- column six presents the vicinity search computation time, which correspond to lines 12 and 13 in Algorithm 5;
- the last column gives the total execution time of our code when choosing the LLL option. The remaining steps are executed fast enough so that the values in this column are just slightly larger than the sum of the values presented

TABLE V
TIMINGS (IN SECONDS) WHEN USING THE THREE DISCRETIZATIONS OF SECTION III. THE LAST COLUMN GIVES THE TOTAL RUNTIME WHEN LLL IS CHOSEN AS THE LATTICE BASIS REDUCTION ALGORITHM.

Filter	Point generation	LLL	BKZ	HKZ	Vicinity search	Total (LLL)
A35/8	0.048	0.026	0.028	0.028	0.231	0.325
A45/8	0.047	0.033	0.035	0.035	0.474	0.571
A125/21	0.112	0.291	0.348	0.471	7.686	8.185
B35/9	0.112	0.018	0.019	0.027	0.243	0.401
B45/9	0.102	0.033	0.035	0.045	0.468	0.676
B125/22	0.191	0.296	0.342	1.194	7.746	8.314
C35/8	0.094	0.022	0.021	0.021	0.225	0.396
C45/8	0.107	0.032	0.036	0.037	0.414	0.611
C125/21	0.319	0.455	0.671	303.98	6.366	7.278
D35/9	0.162	0.019	0.021	0.021	0.211	0.411
D45/9	0.096	0.031	0.037	0.037	0.411	0.595
D125/22	0.251	0.435	0.784	335.38	6.240	7.042
E35/8	0.063	0.017	0.018	0.021	0.211	0.301
E45/8	0.089	0.018	0.031	0.031	0.420	0.574
E125/21	0.236	0.261	0.312	1.935	1.184	7.712

TABLE VI
HIGH DEGREE TYPE I FIR SPECIFICATIONS

Filter	Bands	$D(\omega)$	$W(\omega)$
F	$[0, 0.2\pi]$	0	10
	$[0.205\pi, \pi]$	1	1
G	$[0, 0.4\pi]$	1	1
	$[0.405\pi, 0.7\pi]$	0	10
	$[0.705\pi, \pi]$	1	1
H	$[0, 0.45\pi]$	0	10
	$[0.47\pi, \pi]$	1	1

in columns 2, 3 and 6.

We used the `fplll` C++ library implementations [54] of the LLL, BKZ and HKZ algorithms.

One can note that our approach, when choosing the LLL option, is fast: for instance, it takes at most 8 seconds to compute a very good frequency response for the filters of length 125. Interestingly enough, the use of the BKZ option, which computes better reduced bases, leads to a very small computing time penalty. The use of the BKZ option allows us to compute the best, as of today, frequency responses for C125 and D125 in less than 8 seconds. Another remarkable fact is the excellent behavior of our approach when choosing the HKZ option: it is very fast in the cases of A125, B125 and E125 and offers a reasonable execution time in the cases of C125, D125 whereas the corresponding lattices have a dimension equal to, at least, 63, a size which usually means a very difficult challenge for an HKZ reduction attempt. In addition to the output quality, the efficiency of our approach strongly advocates for its use as a first accelerating step for exact MILP based approaches (where it would cause almost no time penalty).

TABLE VII
HIGH DEGREE QUANTIZATION RESULTS

Filter	Minimax $E^\star$	Naive rounding E_{naive}	LLL reduction E_{LLL}	Runtime in sec.
F1601/18	$8.64 \cdot 10^{-4}$	$3.41 \cdot 10^{-3}$	$\mathbf{1.20 \cdot 10^{-3}}$	846.85
G1201/16	$4.78 \cdot 10^{-3}$	$1.21 \cdot 10^{-2}$	$\mathbf{5.79 \cdot 10^{-3}}$	567.01
H501/16	$1.67 \cdot 10^{-4}$	$5.79 \cdot 10^{-3}$	$\mathbf{1.44 \cdot 10^{-3}}$	379.11

Remark 11. *To further illustrate its robustness, let's mention that our Euclidean lattice-based routine scales well to much larger degrees, as illustrated in Table VII, which adhere to the specifications from Table VI. The scaling factor used is again* $s = 2^{b-1}$. *Note that:*

- *since the exhaustive enumeration performed by the vicinity search would be quite costly on such examples, we use a simple stochastic approach: take uniform random values of* $d_0, d_1, \varepsilon_0, \varepsilon_1$ *inside Algorithm 4 for one minute, keeping the best result at the end;*
- *if the filter degree is too large, at some point the leading coefficients of the optimal quantization will become zero [55]. We have avoided this scenario by considering only very narrow transition bands, which correspond to a slow decrease in the size of the minimax filter coefficients.*

VII. CONCLUSION

We have developed a novel approach for designing machine-number coefficient FIR filters based on an idea previously introduced in [23], which transforms the design problem using the language of Euclidean lattices. The values obtained show that the method is extremely robust and competitive in practice. It frequently produces results which are close to optimal and/or beats other heuristic approaches.

There are several directions of research we are currently pursuing. The first is integrating this method into an FIR filter design suite for FPGA targets. The text [56] is a first attempt in this direction. IIR filter quantization using Euclidean lattices is another idea. There are several difficulties which have to be overcome in the rational IIR setting:

- nonlinearity of the transfer function;
- the approximation domain switches from Ω to a subset of the unit circle;
- ensuring stability of the transfer function (control the position of poles).

REFERENCES

[1] Y. C. Lim and A. Constantinides, "New integer programming scheme for nonrecursive digital filter design," *Electron. Lett.*, vol. 15, no. 25, pp. 812–813, Dec. 1979.

[2] Y. C. Lim, S. Parker, and A. Constantinides, "Finite Word Length FIR Filter Design using Integer Programming Over a Discrete Coefficient Space," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 30, no. 4, pp. 661–664, Aug. 1982.

[3] Y. C. Lim and S. Parker, "FIR Filter Design Over a Discrete Powers-of-Two Coefficient Space," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 31, no. 3, pp. 583–591, Jun. 1983.

[4] ——, "Discrete Coefficient FIR Digital Filter Design Based Upon an LMS Criteria," *IEEE Trans. Circuits Syst.*, vol. 30, no. 10, pp. 723–739, Oct. 1983.

[5] Y. C. Lim, R. Yang, D. Li, and J. Song, "Signed Power-of-Two Term Allocation Scheme for the Design of Digital Filters," *IEEE Trans. Circuits Syst. II*, vol. 46, no. 5, pp. 577–584, May 1999.

[6] J. Yli-Kaakinen and T. Saramäki, "A systematic algorithm for the design of multiplierless FIR filters," in *The 2001 IEEE International Symposium on Circuits and Systems, 2001. ISCAS 2001. Sydney, Australia, May 6–9*, vol. 2. IEEE, 2001, pp. 185–188.

[7] ——, "A Systematic Algorithm for the Design of Lattice Wave Digital Filters With Short-Coefficient Wordlength," *IEEE Trans. Circuits Syst.*, vol. 54, no. 8, pp. 1838–1851, Aug. 2007.

[8] J. Skaf and S. P. Boyd, "Filter Design With Low Complexity Coefficients," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3162–3169, Jul. 2008.

[9] Y. Cao, K. Wang, W. Pei, Y. Liu, and Y. Zhang, "Design of High-Order Extrapolated Impulse Response FIR Filters with Signed Powers-of-Two Coefficients," *Circuits Syst. Signal Process.*, vol. 30, no. 5, pp. 963–985, 2011.

[10] D. Wei, "Design of discrete-time filters for efficient implementation," Ph.D. dissertation, Massachusetts Institute of Technology, 2011. [Online]. Available: https://dspace.mit.edu/handle/1721.1/66470

[11] E. A. da Silva, L. Lovisolo, A. J. Dutra, and P. S. Diniz, "FIR Filter Design Based on Successive Approximation of Vectors," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3833–3848, 2014.

[12] J. Knowles and E. Olcayto, "Coefficient Accuracy and Digital Filter Response," *IEEE Trans. Circuit Theory*, vol. 15, no. 1, pp. 31–41, 1968.

[13] D. M. Kodek, "Design of Optimal Finite Wordlength FIR Digital Filters Using Integer Programming Techniques," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 28, no. 3, pp. 304–308, Jun. 1980.

[14] T. W. Parks and J. H. McClellan, "Chebyshev Approximation for Nonrecursive Digital Filters with Linear Phase," *IEEE Trans. Circuit Theory*, vol. 19, no. 2, pp. 189–194, March 1972.

[15] D. M. Kodek, "Performance limit of finite wordlength FIR digital filters," *IEEE Trans. Signal Process.*, vol. 53, no. 7, pp. 2462–2469, Jul. 2005.

[16] D. M. Kodek and K. Steiglitz, "Comparison of Optimal and Local Search Methods for Designing Finite Wordlength FIR Digital Filters," *IEEE Trans. Circuits Syst.*, vol. 28, no. 1, pp. 28–32, 1981.

[17] O. Gustafsson and L. Wanhammar, "Design of linear-phase FIR filters combining subexpression sharing with MILP," in *The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002.*, vol. 3, Aug. 2002, pp. 9–12.

[18] D. Shi and Y. J. Yu, "Design of Linear Phase FIR Filters With High Probability of Achieving Minimum Number of Adders," *IEEE Trans. Circuits Syst. I*, vol. 58, no. 1, pp. 126–136, Jan. 2011.

[19] D. M. Kodek, "LLL Algorithm and the Optimal Finite Wordlength FIR Design," *IEEE Trans. Signal Process.*, vol. 60, no. 3, pp. 1493–1498, Mar. 2012.

[20] J. Nielsen, "Design of Linear-Phase Direct-Form FIR Digital Filters with Quantized Coefficients Using Error Spectrum Shaping," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 7, pp. 1020–1026, Jul. 1989.

[21] G. Evangelista, "Design of optimum high-order finite-wordlength digital FIR filters with linear phase," *Signal Process.*, vol. 82, no. 2, pp. 187–194, 2002.

[22] D. M. Kodek and M. Krisper, "Telescoping rounding for suboptimal finite wordlength FIR digital filter design," *Digit. Signal Process.*, vol. 15, no. 6, pp. 522–535, 2005.

[23] N. Brisebarre and S. Chevillard, "Efficient polynomial L^∞-approximations," in *18th IEEE Symposium on Computer Arithmetic, 2007. ARITH '07*, Jun. 2007, pp. 169–176.

[24] L. P. Bos and N. Levenberg, "On the calculation of approximate Fekete points: the univariate case," *Electron. Trans. Numer. Anal.*, vol. 30, pp. 377–397, 2008.

[25] J.-P. Calvi and N. Levenberg, "Uniform approximation by discrete least squares polynomials," *J. Approx. Theory*, vol. 152, no. 1, pp. 82–100, May 2008.

[26] A. Sommariva and M. Vianello, "Computing approximate Fekete points by QR factorizations of Vandermonde matrices," *Comput. Math. Appl.*, vol. 57, no. 8, pp. 1324–1336, Apr. 2009.

[27] ——, "Approximate Fekete points for weighted polynomial interpolation," *Electron. Trans. Numer. Anal.*, vol. 37, pp. 1–22, 2010.

[28] L. P. Bos, J.-P. Calvi, N. Levenberg, A. Sommariva, and M. Vianello, "Geometric weakly admissible meshes, discrete least squares approximations and approximate Fekete points," *Math. Comp*, vol. 80, no. 275, pp. 1623–1638, 2011.

[29] S. De Marchi, F. Piazzon, A. Sommariva, and M. Vianello, "Polynomial Meshes: Computation and Approximation," in *Proceedings of the 15th International Conference on Computational and Mathematical Methods in Science and Engineering*, 2015, pp. 414–425.

[30] P. Q. Nguyen and B. Vallée, Eds., *The LLL Algorithm - Survey and Applications*, ser. Information Security and Cryptography. Springer, 2010.

[31] R. Kannan, "Minkowski's convex body theorem and integer programming," *Math. Oper. Res.*, vol. 12, no. 3, pp. 415–440, Aug. 1987.

[32] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, ser. Prentice-Hall Signal Processing Series. Prentice Hall, 2010.

[33] A. Antoniou, *Digital Signal Processing: Signals, Systems, and Filters*. McGraw-Hill Education, 2005.

[34] P. Prandoni and M. Vetterli, *Signal Processing for Communications*. Taylor & Francis, 2008. [Online]. Available: http://www.sp4comm.org/

[35] E. W. Cheney, *Introduction to Approximation Theory*, ser. AMS Chelsea Publishing Series. AMS Chelsea Pub., 1982.

[36] D. M. Kodek, "Design of optimal finite wordlength FIR digital filters," in *Proc. of the European Conference on Circuit Theory and Design, ECCTD '99.*, vol. 1, Aug. 1999, pp. 401–404.

[37] Y. Lim, "Design of Discrete-Coefficient-Value Linear Phase FIR Filters with Optimum Normalized Peak Ripple Magnitude," *IEEE Trans. Circuits Syst.*, vol. 37, no. 12, pp. 1480–1486, Dec. 1990.

[38] S.-I. Filip, "A robust and scalable implementation of the Parks-McClellan algorithm for designing FIR filters," *ACM Trans. Math. Software*, vol. 43, no. 1, Aug. 2016.

[39] A. Çivril and M. Magdon-Ismail, "On selecting a maximum volume sub-matrix of a matrix and related problems," *Theoret. Comput. Sci.*, vol. 410, no. 47-49, pp. 4801–4811, Nov. 2009.

[40] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' guide*. SIAM, 1999, vol. 9.

[41] S.-I. Filip, "Robust tools for weighted Chebyshev approximation and applications to digital filter design," Ph.D. dissertation, École Normale Supérieure de Lyon, 2016. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01447081/

[42] M. Embree and L. N. Trefethen, "Green's Functions for Multiply Connected Domains via Conformal Mapping," *SIAM Review*, vol. 41, no. 4, pp. 745–761, Dec. 1999.

[43] J. Shen and G. Strang, "The Asymptotics of Optimal (Equiripple) Filters," *IEEE Trans. Signal Process.*, vol. 47, pp. 1087–1098, 1999.

[44] J. Shen, G. Strang, and A. J. Wathen, "The Potential Theory of Several Intervals and Its Applications," *Appl. Math. Opt.*, vol. 44, pp. 67–85, 2001.

[45] P. M. Gruber and C. G. Lekkerkerker, *Geometry of numbers*, 2nd ed., ser. North-Holland Mathematical Library. Amsterdam: North-Holland Publishing Co., 1987, vol. 37.

[46] L. Lovász, *An algorithmic theory of numbers, graphs and convexity*, ser. CBMS-NSF Regional Conference Series in Applied Mathematics. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1986, vol. 50.

[47] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*. New York: Springer-Verlag, 1988.

[48] D. Micciancio and S. Goldwasser, *Complexity of Lattice Problems: a cryptographic perspective*, ser. The Kluwer International Series in Engineering and Computer Science. Boston, Massachusetts: Kluwer Academic Publishers, Mar. 2002, vol. 671.

[49] T. H. I. U. of Crystallography, *Space-group symmetry*, ser. International Tables for Crystallography. Springer Netherlands, 2002, vol. A.

[50] R. Kannan, "Algorithmic geometry of numbers," *Annual Review of Computer Science*, vol. 2, pp. 231–267, 1987.

[51] D. Aggarwal, D. Dadush, and N. Stephens-Davidowitz, "Solving the Closest Vector Problem in 2^n Time - The Discrete Gaussian Strikes Again!" in *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA*, V. Guruswami, Ed. IEEE Computer Society, 2015, pp. 563–582.

[52] C. Dubey and T. Holenstein, "Approximating the closest vector problem using an approximate shortest vector oracle," in *Approximation, randomization, and combinatorial optimization*, ser. Lecture Notes in Comput. Sci. Springer, Heidelberg, 2011, vol. 6845, pp. 184–193.

[53] S. Chevillard, "Évaluation efficace de fonctions numériques. outils et exemples," Ph.D. dissertation, École Normale Supérieure de Lyon, 2009. [Online]. Available: http://www-sop.inria.fr/members/Sylvain.Chevillard/

[54] The FPLLL development team, "fplll, a lattice reduction library," 2016. [Online]. Available: https://github.com/fplll/fplll

[55] D. M. Kodek, "Length limit of optimal finite wordlength FIR filters," *Digit. Signal Process.*, vol. 25, no. 5, pp. 1798–1805, Sep. 2013.

[56] N. Brisebarre, F. de Dinechin, S.-I. Filip, and M. Istoan, "Automatic generation of hardware FIR filters from a frequency domain specification," 2016. [Online]. Available: https://hal.inria.fr/hal-01308377