Probabilistic Error Free Design of Long Fixed-Point Polynomial FIR Predictors

Jarno M. A. Tanskanen†, Jari A. K. Hyttinen†, and Vassil S. Dimitrov²

1 Tampere University of Technology
Ragnar Granit Institute
P.O. Box 692, FI-33101 Tampere, FINLAND
†Tel. +358 3 3116 4016, +358 50 3871 347, †Fax: +358 3 3116 4013
E-mail: tanskanen@ieee.org, jari.hyttinen@tut.fi
URL: http://www.rgi.tut.fi/~tanskanen/
²University of Calgary
Department of Electrical and Computer Engineering
2500 University Drive NW
Calgary, Alberta T2N 1N4, CANADA
E-mail: dimitrov@vlsi.enel.ucalgary.ca
URL: http://www.atips.ca/people/dimitrov/

ABSTRACT

In this paper, a method for designing long fixed-point polynomial FIR predictors (FPFPs) is proposed. Our method yields filters that perform exact prediction even with short coefficient word lengths. Under ordinary coefficient quantization, prediction capabilities degrade, or may be totally lost. Here, the filters are designed so that the prediction properties are exactly preserved in fixed-point implementations. The algorithm is derived for second degree polynomial prediction using any filter length and integer prediction step. Also some non-integer prediction steps are possible.

1. INTRODUCTION

Polynomial predictive filtering theory has been well established [1,2,3,4] but applicability of polynomial FIR predictors (PFPs) has suffered from the practical constraint of finite coefficient precision, which may cause severe degradation of filter characteristics. The coefficient quantization problem has been demonstrated in [5], and in [6] it has been shown possible to design coefficient quantization error free (CQEF) FPFPs, but since the design method employed was based on exhaustive search over a limited quantized coefficient space, the method was limited for design of fairly short CQEF FPFPs, e.g. up to the length \( N = 32 \), depending on the computational resources. From the practical application point of view, this length should be adequate most of the time, but should a lower noise gain (NG) of a long FPFP be desirable, one should resort to the algorithm proposed in this paper. The principal of the proposed algorithm has been presented in [7] where the algorithm was derived for fixed-point predictive polynomial FIR differentiators with one-step-ahead predictive differentiation as an example. In this paper, the algorithm is derived for FPFPs with an arbitrary integer prediction step while also some non-integer prediction steps are possible.

In this section, PFPs are shortly reviewed. In section 2, linear Diophantine equation formulation of the fixed-point filter design task is presented. The proposed algorithm for solving the Diophantine equations is given in section 3, and design examples in section 4. Section 5 concludes the paper.

1.1 Polynomial FIR Predictors

PFPs, derived in [1], assume a low-degree polynomial input signal contaminated by white Gaussian noise. Filter output at a discrete time instant \( n \), \( x(n) \), is defined to be a \((p+1)\)-steps-ahead (e.g., \( p = 0 \) yields one-step-ahead prediction) predicted input,

\[
x(n + p) = \sum_{k=1}^{N} h(k) x(n - k)
\]

where \( h(k) \) are filter coefficients, and \( N \) is filter length. After providing for exact prediction with piecewise polynomial input signal, the rest of the degrees of freedom are used to minimize the white NG,

\[
NG(h) = \sum_{k=1}^{N} |h(k)|^2 , \quad h = [h(1), \ldots, h(N)].
\]

A set of linear constraints can be derived from the definition of the filter output (1) and the piecewise polynomial signal model assumption. Considering polynomial degrees up to two, the constraints are given by

\[
g_0 = \sum_{k=1}^{N} h(k) - 1 = 0 , \quad g_p = \sum_{k=1}^{N} kh(k) + p = 0 , \quad (3,4)\]

and

\[
g_p = \sum_{k=1}^{N} k^2 h(k) - p^2 = 0 . \quad (5)
\]

Constraints (3)–(5) yield prediction of each of the polynomial degrees 0, 1, and 2, respectively. From (3)–(5) can closed form solutions for the FIR coefficients for polynomial input signals up to degree two be solved by the method of Lagrange multipliers [1,8]. The closed form solutions for PFP coefficients for the first, second, and third degree polynomial input signals can be found in [1] for \( p = 0 \). Filter coefficients for the first and second de-
pose that all the coefficients formulated as an integer programming (IP) problem. The optimization problem that has to be solved can be reformulated as an integer programming problem. Suppose that all the coefficients \( h(k) \), \( k = 1, \ldots, N \), of the filter are multiplied by \( 2^n \) where \( n \) is the number of bits available, not considering the sign bit, and rounded to the nearest integer. In the sequel, superscript \( * \) denotes an integer quantity.

\[
h(k) = \frac{4N^2 + 6N(1-k-p) + 12kp - 6p - 6k + 2}{(N + 1)N(N - 1)}
\]

(6)

Instructions for deriving the coefficients using Mathematica are given in [10].

2. LINEAR DIOPHANTINE EQUATION FORMULATION OF THE FILTER DESIGN PROBLEM

The optimization problem that has to be solved can be reformulated as an integer programming (IP) problem. Suppose that all the coefficients \( h(k) \), \( k = 1, \ldots, N \), of the filter are multiplied by \( 2^n \) where \( n \) is the number of bits available, not considering the sign bit, and rounded to the nearest integer. In the sequel, superscript \( * \) denotes an integer quantity.

The CQEF FPFP design task can be defined as an algorithm with the following input and output:

**Input:** Function \( F(h(1), h(2), \ldots, h(N)) = \sum_{k=1}^{N} h^2(k) \), with integer variables \( h^*(k) \), that is to be minimized while having the constraints

\[
g_1^* = \sum_{k=1}^{N} h^*(k) - 2^n = 0, \quad (8)
\]

\[
g_2^* = \sum_{k=1}^{N} kh^*(k) + 2^n p = 0, \quad (9)
\]

\[
g_3^* = \sum_{k=1}^{N} k^2 h^*(k) - 2^n p^2 = 0, \quad (10)
\]

on the variables. The constraints (8)–(10) are the integer versions of the constraints (3)–(5) scaled up by \( 2^n \).

**Output:** An integer vector \( h^* = [h^*(1), h^*(2), \ldots, h^*(N)] \) that minimizes \( F \) and satisfies exactly the constrains (8)–(10).

The solution we offer is based on the following considerations:

1. As the filter coefficients are to be presented with short word-length fixed-point numbers, the task in hand is a quadratic integer programming problem, which is well-known to be an NP-complete problem; therefore it is unrealistic to find the best solution in a reasonable amount of time, especially for long filters. Designing these filters with floating-point coefficients would present us with a quadratic real programming problem, which was solvable in polynomial time.

2. Without restricting the variables to be integers, we have closed form solutions of the problem, which are given for polynomial degrees one and two by (6) and (7), respectively. Although the values computed by these formulas are not integers, these expressions give us good initial approximations.

3. To make sure that the conditions (3)–(5), or (8)–(10), are met exactly, one has to solve a desired system above in integers. This problem has been a subject of deep investigations in number theory and the theory of Diophantine equations. By eliminating the variables, one can reduce the problem to a single linear equation of the form:

\[
A_1x_1 + A_2x_2 + \cdots + A_x = B,
\]

(11)

with integers \( A_1, A_2, \ldots, A_x, B \). Equations of the form (11) are called Diophantine equations.

3. PROBABILISTIC ALGORITHM FOR SOLVING THE SYSTEM OF LINEAR DIOPHANTINE EQUATIONS

If the filter is long, say, longer than \( N = 32 \), an exhaustive search even over a limited quantized coefficient space is no longer feasible because of the computer time required. Here, a fast probabilistic algorithm is proposed to yield solutions of (3)–(5), or (8)–(10), for large \( N \), and arbitrary \( p \). The algorithm can as well be applied to other filter families whose design constraints can be formulated as linear constraints on the filter coefficients \( h(k) \). A simple algorithm for designing first degree CQEF FPFPs has been given in [7].

Let us now consider the second degree CQEF FPFP for which we have the constraints (8)–(10). The simple algorithm [7] generally fails to find quantized coefficients fulfilling (8)–(10) for long FPFPs. From (8)–(10), the following formulas for the filter coefficients can be derived:

\[
h^*(N) = -\sum_{k=1}^{N-1} h^*(k) + 2^n,
\]

(12)

\[
h^*(N-1) = \sum_{k=1}^{N-2} (k - N)h^*(k) + 2^n(N + p),
\]

(13)

\[
A(1)h^*(1) + \cdots + A(N-2)h^*(N-2) = B,
\]

(14)

where

\[
A(k) = N^2 - N - 2Nk + k^2, \quad k = 1, \ldots, N - 2
\]

(15)

\[
B = 2^n(N^2 - N + 2Np - p + p^2),
\]

(16)

Here solving (14) presents the main computational problem. It is well known in computational number theory, that it is easy to find a solution of (14), but not so easy to
find one of small Euclidean length,
\[ \|h^*\| = \sqrt{h^2(1) + h^2(2) + \ldots + h^2(N-2)}, \quad (17) \]
c.f., NG (2), for the truncated coefficient vector \( h^* \) with the elements \( h^*(k), k = 1, \ldots, N-2 \). The problem has been a subject of extensive study last decades [8,11–14]. Without any specific information about the solution looked for, the problem is computationally challenging, especially if one wants to find the best solution, that is, the one that globally minimizes the Euclidean length (17) of the vector \( h^* \). However, in our particular case we may assume that, roughly speaking, a sufficiently good solution could be somewhere in the close vicinity of the real number form coefficient vector \( h \), computed by (7). Based on this, we offer the following algorithm for solving the system of linear Diophantine equations (8)–(10):

**Input:** Coefficients \( A(k) \) and \( B \) computed by (15) and (16), respectively.

**Output:** Vector \( h^* \) of integers \( h^*(k), k = 1, 2, \ldots, N \), satisfying the equations (8)–(10) and such that the NG (2) is small (23), and the coefficients \( h^*(k) \) satisfy a dynamic range requirement (22).

**Step 0:** Calculate the real solution \( h \) by formula (7).

**Step 1:** Multiply the coefficients by \( 2^n \) and find the two smallest values, denote their indices by \( k'_1 \) and \( k'_2 \), such that \( k'_1, k'_2 \neq N, N-1 \).

**Step 2:** Pick up a random vector \( r \) with elements \( r(k) = \{-1, +1\}, k = 1, \ldots, N-2, k \neq k'_1, k'_2 \).

**Step 3:** Approximate the integer coefficients \( h^*(k), k = 1, \ldots, N-2, k \neq k'_1, k'_2 \) according to
\[ h^*(k) = \lfloor h(k) \rfloor \text{ if } r(k) = +1, \quad (18) \]
\[ h^*(k) = \lceil h(k) \rceil \text{ if } r(k) = -1, \quad (19) \]

**Step 4:** Solve the Diophantine equation
\[ A(k'_1)h^*(k'_1) + A(k'_2)h^*(k'_2) = B', \quad (20) \]
\[ B' = 2^n(N^2 - N + 2Np - p + p^2) - \sum_{k=1, k \neq k'_1, k'_2}^{N-2} A(k)h^*(k), \quad (21) \]

to find the coefficients \( h^*(k'_1) \) and \( h^*(k'_2) \).

**Step 5:** Calculate \( h^*(N-1) \) and \( h^*(N) \) using (13) and (12), respectively.

**Step 6:** If the obtained coefficients satisfy dynamic range condition
\[ |h^*(k)| < 2^n, \quad (22) \]
and a possible NG condition with user selectable criterion \( a \)
\[ \text{then print them and stop; else } \]

**Step 7:** Go to Step 2.

Let us comment the algorithm shown above:

**Step 0** returns a real (infinite precision) coefficients that will serve as an initial starting point.

**Step 1** looks for and finds the smallest (in absolute value) coefficients.

**Step 2** and Step 3 are the randomization steps of the algorithm. They are aimed at random truncation of the coefficients obtained by formula (7).

**Step 4** determines the integer values of the two coefficients selected at Step 1. For short coefficient word lengths, e.g. up to \( n = 10 \), an exhaustive search over \( k'_1 \) and \( k'_2 \) is feasible, and employed in this paper. Thus, also the NG contributions of \( k'_1 \) and \( k'_2 \) can be minimized. It is to be noted that it is possible to try to design CQEF FPFPs with non-integer prediction steps, as long as \( B' \) (21) is an integer.

**Step 5** calculates the last two remaining coefficients so that the coefficients exactly fulfill the constraints (8)–(10).

**Step 6** terminates the algorithm if all the coefficients satisfy the dynamic range condition (22), and the NG is sufficiently low (23), otherwise it goes back to Step 2 and picks up another random vector \( r \).

### 4. DESIGN EXAMPLES

To demonstrate the proposed CQEF FPFP design method, we used the algorithm to find CQEF FPFPs of lengths \( N = 31, \ldots, 130 \), with prediction steps \( p = \{-0.5, 0, 0.5, 4, 9\} \), and coefficient word lengths \( n = \{3, 4, 6, 8, 10\} \) for second degree polynomial signals. For computer with a 3 GHz Pentium IV processor and 512 MB of memory it takes at maximum approximately 0.2 s to run 100 iterations of the proposed algorithm using Matlab. The results are summarized in Table 1, in which the counts of successfully found CQEF FPFPs are given, along with the counts of the CQEF FPFPs whose NGs were lower than those of the corresponding rounded coefficient FPFPs. From Table 1 it is seen that eight bits is sufficient for designing a CQEF FPFP of any length considered for the considered prediction steps, and that several CQEF FPFPs can be designed with three bit coefficients.

In Table 2, a few exemplary NG comparisons between the infinite precision FPFPs, CQEF FPFPs, and rounded coefficient FPFPs are given, along with the absolute magnitude response and group delay errors of the rounded coefficient FPFPs at zero frequency. In Table 2, the natural tendency of rounded coefficient FPFPs to maintain their NG while their magnitude response and group delay errors at zero frequency increase is seen, whereas CQEF FPFPs retain their exact prediction properties at zero frequency, and their NG generally increases as the coefficient word length is shortened. An exemplary FPFP and the corresponding FPFPs are shown in Fig. 1.
Table 1. Numbers of CQEF FPFPs found out of a hundred filters \((N = 31, \ldots, 130)\) during 100 runs of the algorithm for some coefficient word lengths and prediction steps \((p+1)\)-steps-ahead prediction), and the numbers of the CQEF FPFPs (in parentheses), which have lower NGs than their rounded coefficients counterparts.

<table>
<thead>
<tr>
<th>(p)</th>
<th>(-0.5)</th>
<th>0</th>
<th>0.5</th>
<th>4</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)</td>
<td>(37) (0)</td>
<td>37 (0)</td>
<td>36 (0)</td>
<td>41 (0)</td>
<td>46 (0)</td>
</tr>
<tr>
<td>(4)</td>
<td>79 (0)</td>
<td>81 (0)</td>
<td>80 (0)</td>
<td>81 (0)</td>
<td>82 (0)</td>
</tr>
<tr>
<td>(6)</td>
<td>100 (0)</td>
<td>98 (0)</td>
<td>100 (1)</td>
<td>100 (1)</td>
<td>100 (1)</td>
</tr>
<tr>
<td>(8)</td>
<td>100 (1)</td>
<td>100 (8)</td>
<td>100 (3)</td>
<td>100 (3)</td>
<td>100 (8)</td>
</tr>
<tr>
<td>(10)</td>
<td>100 (12)</td>
<td>100 (6)</td>
<td>100 (6)</td>
<td>100 (13)</td>
<td>100 (10)</td>
</tr>
</tbody>
</table>

Table 2. Exemplary NGs (rounded to three decimals) of infinite precision FPFPs \((h)\), CQEF FPFPs \((h^*)\), and rounded coefficient FPFPs \((h^r)\), along with the absolute values of their magnitude errors \((e_{\text{r,mag}})\) and group delay errors \((e_{\text{r,grpd}})\) of the rounded coefficient FPFPs at zero frequency, for a few filter lengths, prediction steps, and coefficient word lengths.

<table>
<thead>
<tr>
<th>(N, p, n)</th>
<th>NG ((h))</th>
<th>NG ((h^*))</th>
<th>NG ((h^r))</th>
<th>(e_{\text{r,mag}})</th>
<th>(e_{\text{r,grpd}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>33, 0, 3</td>
<td>0.3087</td>
<td>1.4375</td>
<td>0.2969</td>
<td>0.625</td>
<td>8.231</td>
</tr>
<tr>
<td>33, 0, 10</td>
<td>0.3087</td>
<td>0.3087</td>
<td>0.3081</td>
<td>0.002</td>
<td>0.024</td>
</tr>
<tr>
<td>33, 9, 3</td>
<td>1.839</td>
<td>2.438</td>
<td>1.953</td>
<td>0.125</td>
<td>1.571</td>
</tr>
<tr>
<td>33, 9, 10</td>
<td>1.839</td>
<td>1.839</td>
<td>1.840</td>
<td>0.002</td>
<td>0.046</td>
</tr>
<tr>
<td>90, 0.5, 4</td>
<td>0.109</td>
<td>172.438</td>
<td>0.106</td>
<td>0.438</td>
<td>16.674</td>
</tr>
<tr>
<td>90, 0.5, 6</td>
<td>0.109</td>
<td>57.409</td>
<td>0.108</td>
<td>0.094</td>
<td>3.914</td>
</tr>
<tr>
<td>90, 0.5, 8</td>
<td>0.109</td>
<td>1.215</td>
<td>0.109</td>
<td>0.000</td>
<td>0.086</td>
</tr>
<tr>
<td>90, 0.5, 10</td>
<td>0.109</td>
<td>0.109</td>
<td>0.109</td>
<td>0.004</td>
<td>0.222</td>
</tr>
</tbody>
</table>

1 NG of the CQEF FPFP whose NG is closest to that of its infinite precision coefficient counterpart.
2 Magnitude response and group delay errors may allow lower NGs in ordinary coefficient quantization than achievable with exact infinite precision coefficient FPFPs.

Fig. 1. An exemplary set of second degree polynomial FIR predictors with \(N = 90\), \(p = 0.5\), and \(n = 10\): infinite precision FPFP (solid), CQEF FPFP (dotted), and rounded coefficient FPFP (dash-dot).

5. CONCLUSIONS
A probabilistic technique for error free digital FPFP coefficient quantization has been proposed. The method uses number-theoretic tools, and is applicable for designing long CQEF FPFPs. The filter design constraints, giving FPFPs their polynomial signal prediction properties, can be exactly satisfied for most of the considered filter lengths. In these cases the influence of the coefficient round-off errors is eliminated. Although the algorithm does not yield globally minimized NG, many of the produced filters are sufficient for practical purposes. The proposed integer programming method for fixed-point filter design is well suited for all filter design tasks in which the design criteria can be formulated in a form of linear constraints on the filter coefficients.

ACKNOWLEDGMENT
The work of J. M. A. Tanskanen was funded by the Academy of Finland grant number 206996.

REFERENCES
[11] C. G. J. Jacobi, “Über die auflösung der gleichung \(a_1 \alpha_1 + a_2 \alpha_2 + \ldots + a_n \alpha_n = f - u\),” J. Reine Angew. Math., vol. 69, pp. 1–28, 1868.