A lattice basis reduction approach for the design of finite wordlength FIR filters

Silviu Filip
Mathematical Institute, Numerical Analysis Group, University of Oxford
joint work with Nicolas Brisebarre and Guillaume Hanrot

Rencontres Arithmétiques de l'Informatique Mathématique (RAIM)
Lyon, October 24-26, 2017
Digital filter design chain: outline

**Step 0**
Infinite precision design (i.e., find suitable $p^*$)

**Step 1**
Fixed-point
$x = s \cdot m \cdot 2^e$
$s \in \{\pm 1\}, e \text{ fixed}$

**Step 2**
Coefficient quantization (i.e., format constraints on the coefficients of $p^*$)

**Step 3**
Floating-point
$x = s \cdot m \cdot 2^e$
$s \in \{\pm 1\}, e \text{ variable}$

Hardware/software filter synthesis
Digital signal processing = the study of discrete-time signals

Usual notation: \( x[n] , n \in \mathbb{Z} \)

Example:

→ measured data = signals:
  - temperature readings;
  - content of data packets (in network transmissions);
  - stock price changes;
  - etc.
How do we extract information from signals?

Examples:

1. Periodicity?

2. Noise?
we get two categories of linear filters

1. Finite Impulse Response (FIR) filters
   - $H$ is a polynomial

2. Infinite Impulse Response (IIR) filters
   - $H$ is a rational function

→ convenient to work in the frequency domain
→ focus on FIR filters (with linear phase)
we get two categories of linear filters
- finite impulse response (FIR) filters

\[ y[n] = b_0 x[n] + \sum_{k=1}^{N} b_k x[n - k] \]
Digital filters

We get two categories of linear filters:

- **Finite impulse response (FIR) filters**
  - $H$ is a polynomial

- **Infinite impulse response (IIR) filters**
  - $H$ is a rational function

It's convenient to work in the frequency domain, so we focus on FIR filters (with linear phase).

The equation for the digital filter is:

$$y[n] = b_0 x[n] + \sum_{k=1}^{N} b_k x[n - k] - \sum_{k=1}^{M} a_k y[n - k]$$

We get two categories of linear filters:

- finite impulse response (FIR) filters
- infinite impulse response (IIR) filters
we get two categories of linear filters

- finite impulse response (FIR) filters
  \( H \) is a polynomial

- infinite impulse response (IIR) filters
  \( H \) is a rational function

→ convenient to work in the **frequency** domain

→ **focus** on FIR filters (with linear phase)
FIR filters: Chebyshev approximation

Input:
- degree $n$
- approximation bands $\Omega \subseteq [0, \pi]$
- ideal response $D(\omega)$, $\omega \in \Omega$

![Diagram showing passband, transition band, and stopband]
FIR filters: Chebyshev approximation

Input:
- degree $n$
- approximation bands $\Omega \subseteq [0, \pi]$
- ideal response $D(\omega)$, $\omega \in \Omega$

Output:

$$H(e^{i\omega}) = \sum_{k=0}^{n} h_k \cos(k\omega) = \sum_{k=0}^{n} h_k T_k(\cos(\omega)),$$

s.t.

$$\delta = \max_{\omega \in \Omega} |D(\omega) - H(e^{i\omega})|$$

is minimal
FIR filters: Chebyshev approximation

Input:
- degree $n$
- approximation bands $\Omega \subseteq [0, \pi]$
- ideal response $D(\omega)$, $\omega \in \Omega$

Output:

$$H(e^{i\omega}) = \sum_{k=0}^{n} h_k \cos(k\omega) = \sum_{k=0}^{n} h_k T_k(\cos(\omega)),$$

s.t.

$$\delta = \max_{\omega \in \Omega} |D(\omega) - H(e^{i\omega})|$$

is minimal
FIR filters: Chebyshev approximation

Input:
- degree $n$
- approximation bands $\Omega \subseteq [0, \pi]$
- ideal response $D(\omega)$, $\omega \in \Omega$

Output:

$$H(e^{i\omega}) = \sum_{k=0}^{n} h_k \cos(k\omega) = \sum_{k=0}^{n} h_k T_k(\cos(\omega)),$$

s.t.

$$\delta = \max_{\omega \in \Omega} \max_{x \in X} |D(\omega) - f(x) - p^*(x)|$$

is minimal

Where is this filter useful?
Floating point arithmetic is sometimes expensive (i.e., in embedded systems)

→ use fixed-point arithmetic (scaled integers in a certain range)

**Two main reasons:** small price + fast hardware

→ frequent for signal processing applications
A toy example

\[ p^*(x) = \frac{a_0}{2} T_0(x) + \sum_{k=1}^{9} a_k T_k(x), \delta \simeq 0.0249 \]

→ 7-bit coefficients: \( a_k = \frac{m_k}{2^5}, m_k \in \mathbb{Z} \cap [-63, 63], k = 0, \ldots, 9. \)
\[ p^*(x) = \frac{a_0}{2} T_0(x) + \sum_{k=1}^{9} a_k T_k(x), \delta \simeq 0.0249 \]

→ 7-bit coefficients: \( a_k = \frac{m_k}{2^5}, m_k \in \mathbb{Z} \cap [-63, 63], k = 0, \ldots, 9. \)

**Naive approach** Simple rounding of the \( a_k \in \mathbb{R} \) result: \( \delta \simeq 0.0731 \)
A toy example

\[ p^*(x) = \frac{a_0}{2} T_0(x) + \sum_{k=1}^{9} a_k T_k(x), \delta \simeq 0.0249 \]

→ 7-bit coefficients: \( a_k = \frac{m_k}{2^5} \), \( m_k \in \mathbb{Z} \cap [-63, 63] \), \( k = 0, \ldots, 9 \).

**Naive approach** Simple rounding of the \( a_k \in \mathbb{R} \) result: \( \delta \simeq 0.0731 \)

**Optimal quantization** \( \delta \simeq 0.0468 \)
take \( \varphi_0(x) = \frac{T_0(x)}{2^6} \), \( \varphi_1(x) = \frac{T_1(x)}{2^5} \), \ldots, \( \varphi_9(x) = \frac{T_9(x)}{2^5} \)

want to solve

\[
\begin{align*}
\text{minimize} & \quad \delta \\
\text{subject to} & \quad \sum_{k=0}^{9} m_k \varphi_k(x) - f(x) \leq \delta, \quad x \in X, \\
& \quad f(x) - \sum_{k=0}^{9} m_k \varphi_k(x) \leq \delta, \quad x \in X, \\
& \quad \delta > 0, \quad m_k \in \mathbb{Z} \cap [-63, 63], \quad k = 0, \ldots, 9.
\end{align*}
\]
→ take $\varphi_0(x) = \frac{T_0(x)}{2^6}$, $\varphi_1(x) = \frac{T_1(x)}{2^5}$, \ldots, $\varphi_9(x) = \frac{T_9(x)}{2^5}$

→ actually solve

\[
\begin{align*}
\text{minimize} & \quad \delta \\
\text{subject to} & \quad \sum_{k=0}^{9} m_k \varphi_k(x) - f(x) \leq \delta, x \in X_d, \\
& \quad f(x) - \sum_{k=0}^{9} m_k \varphi_k(x) \leq \delta, x \in X_d, \\
& \quad \delta > 0, \quad m_k \in \mathbb{Z} \cap [-63, 63], k = 0, \ldots, 9.
\end{align*}
\]

$X \rightarrow X_d$ discrete: mixed-integer linear programming

Other heuristic approaches:
→ MATLAB uses a stochastic-based method
\[ \rightarrow \text{with no format constraints, interpolation at well-placed nodes works well} \]

\[ \text{Why not do something similar here?} \]

\[ \rightarrow \text{take } x_0 < x_1 < \cdots < x_n, \text{ all from } X \text{ and find appropriate } m_k \in \mathbb{Z} \text{ s.t.} \]

\[ \sum_{i=0}^{n} m_k \begin{bmatrix} \varphi_k(x_0) \\ \varphi_k(x_1) \\ \vdots \\ \varphi_k(x_n) \end{bmatrix} \approx \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \]
The idea

→ with no format constraints, **interpolation** at well-placed nodes works well.

*Why not do something similar here?*

→ take \( x_0 < x_1 < \cdots < x_n \), all from \( X \) and find appropriate \( m_k \in \mathbb{Z} \) s.t.

\[
\sum_{i=0}^{n} m_k \begin{bmatrix}
\varphi_k(x_0) \\
\varphi_k(x_1) \\
\vdots \\
\varphi_k(x_n)
\end{bmatrix} \approx \begin{bmatrix}
f(x_0) \\
f(x_1) \\
\vdots \\
f(x_n)
\end{bmatrix}
\]

*How can we solve this pseudo interpolation problem?*

→ state it as an Euclidean lattice problem.
→ consider the case $n = 1$
consider the case $n = 1$

NP-hard problem in general.
→ consider the case $n = 1$

The closest vector problem

→ problem in Euclidean lattice theory:

\[
\sum_{k=0}^{n} m_k \begin{bmatrix}
\varphi_k(x_0) \\
\varphi_k(x_1) \\
\vdots \\
\varphi_k(x_n)
\end{bmatrix} \sim \begin{bmatrix}
f(x_0) \\
f(x_1) \\
\vdots \\
f(x_n)
\end{bmatrix}
\]

→ use fast, approximate solvers:

- LLL algorithm [Lenstra, Lenstra & Lovász 1982]
- Kannan embedding [Kannan 1987]

→ on the toy example, we get the optimal result.
The closest vector problem

→ problem in Euclidean lattice theory:

\[ \sum_{k=0}^{n} m_k \begin{bmatrix} \varphi_k(x_0) \\ \varphi_k(x_1) \\ \vdots \\ \varphi_k(x_n) \end{bmatrix} \succsim \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \]

→ use fast, approximate solvers:

- **LLL** algorithm [Lenstra, Lenstra & Lovász 1982]
- Kannan embedding [Kannan 1987]

→ on the toy example, we get the optimal result
Choosing good points

What interpolation points do we use?

$$x = \{x_0, \ldots, x_n\}$$

$$\mathbb{R}_n[x] = \text{span}_{\mathbb{R}} \{\varphi_0, \ldots, \varphi_n\} \quad (\varphi_k(x) = T_k(x))$$
Choosing good points

What interpolation points do we use?

\( x = \{x_0, \ldots, x_n\} \)

\( \mathbb{R}_n[x] = \text{span}_{\mathbb{R}} \{\varphi_0, \ldots, \varphi_n\} \quad (\varphi_k(x) = T_k(x)) \)

→ Lagrange interpolation at \( x \):

\[
\mathcal{L}_x f(x) = \sum_{i=0}^{n} f(x_i) \frac{\det V(x_0, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n)}{\ell_i(x)}
\]

where \( V(x_0, \ldots, x_n) = [v_{ij}] := [\varphi_j(x_i)] \).
Choosing good points: Lebesgue constants

What interpolation points do we use?
\( x = \{x_0, \ldots, x_n\} \)
\( \mathbb{R}_n[x] = \text{span}_\mathbb{R} \{\varphi_0, \ldots, \varphi_n\} \) \hspace{1cm} (\varphi_k(x) = T_k(x))

→ Lagrange interpolation at \( x \):

\[
\mathcal{L}_x f(x) = \sum_{i=0}^{n} f(x_i) \frac{\det V(x_0, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n)}{\det V(x_0, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \ell_i(x)}
\]

where \( V(x_0, \ldots, x_n) = [v_{ij}] := [\varphi_j(x_i)]. \)

→ important quantity: Lebesgue constant

\[
\Lambda_{X,x} = \max_{x \in X} \sum_{i=0}^{n} |\ell_i(x)|
\]
Choosing good points: Lebesgue constants

What interpolation points do we use?
\[ x = \{x_0, \ldots, x_n\} \]
\[ \mathbb{R}_n[x] = \text{span}_{\mathbb{R}} \{\varphi_0, \ldots, \varphi_n\} \quad (\varphi_k(x) = T_k(x)) \]

→ Lagrange interpolation at \( x \):
\[
L_x f(x) = \sum_{i=0}^{n} f(x_i) \frac{\det V(x_0, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n)}{\det V(x_0, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)} \ell_i(x)
\]

where \( V(x_0, \ldots, x_n) = [v_{ij}] := [\varphi_j(x_i)] \).

→ important quantity: Lebesgue constant
\[
\Lambda_{X,x} = \max_{x \in X} \sum_{i=0}^{n} |\ell_i(x)|
\]

→ measures quality of \( x \) for doing interpolation:
\[
\forall f \in C(X), \| f - L_x f \|_{\infty,X} \leq (1 + \Lambda_{X,x}) \| f - p^* \|_{\infty,X}
\]
An approximate Fekete points approach

→ if $X = [-1, 1]$, take Chebyshev nodes

$$x_k = \cos \left( \frac{(n - k)\pi}{n} \right), \quad k = 0, \ldots, n$$

$$\Lambda_{X,x} = \mathcal{O}(\log n)$$
An approximate Fekete points approach

→ if $X = [-1, 1]$, take Chebyshev nodes

$$x_k = \cos \left( \frac{(n - k)\pi}{n} \right), \quad k = 0, \ldots, n$$

$$\Lambda_{X,X} = O(\log n)$$

→ Fekete points: choose $x$ s.t. $|\det V(x)|$ is maximized ($\Lambda_{X,X} \leq n + 1$)
→ difficult to compute them in general
An approximate Fekete points approach

→ if $X = [-1, 1]$, take Chebyshev nodes

$$x_k = \cos \left( \frac{(n - k)\pi}{n} \right), \quad k = 0, \ldots, n$$

$$\Lambda_{X,x} = \mathcal{O}(\log n)$$

→ Fekete points: choose $x$ s.t. $|\det V(x)|$ is maximized ($\Lambda_{X,x} \leq n + 1$)
→ difficult to compute them in general

**Approach:**
→ replace $X$ with a **suitable** discretization $X_n$
→ $x \subset X_n$ generates max volume $(n + 1) \times (n + 1)$ submatrix of $V(X_n)$

→ NP-hard problem [Çivril & Magdon-Ismail, 2009]
→ greedy algorithm based on QR factorization [Sommariva & Vianello, 2010]
What is a suitable discretization of $X$?
→ use the theory of weakly admissible meshes [Calvi & Levenberg, 2008]
What is a suitable discretization of $X$?
→ use the theory of weakly admissible meshes [Calvi & Levenberg, 2008]

We have:
→ $X_n = \text{union of } n\text{-th order Chebyshev nodes scaled to each interval of } X$

$$\Lambda_{X,x} = O_X(n \log n)$$
What is a suitable discretization of $X$?

→ use the theory of weakly admissible meshes [Calvi & Levenberg, 2008]

We have:

→ $X_n = \text{union of } n\text{-th order Chebyshev nodes scaled to each interval of } X$

\[ \Lambda_{X,x} = \mathcal{O}_X(n \log n) \]

→ our finite wordlength polynomial $p$ satisfies:

\[ ||f - p||_{\infty,X} \leq (1 + \Lambda_{X,x})||f - p_{opt}||_{\infty,X} + \Lambda_{X,x}\delta \]

where:

- $\delta = \max_{0 \leq k \leq n} |p(z_k) - f(z_k)|$
- $p_{opt}$ is an optimal solution
→ specification:

\[ X = [-1, \cos(0.84\pi)] \cup [\cos(0.68\pi), \cos(0.4\pi)] \cup [\cos(0.24\pi), 1], \]

\[ f(x) = \begin{cases} 
1, & x \in [-1, \cos(0.84\pi)] \cup [\cos(0.24\pi), 1] \\
0, & x \in [\cos(0.68\pi), \cos(0.4\pi)] 
\end{cases} \]
Another example

→ specification:

\[ X = [-1, \cos(0.84\pi)] \cup [\cos(0.68\pi), \cos(0.4\pi)] \cup [\cos(0.24\pi), 1], \]

\[ f(x) = \begin{cases} 
1, & x \in [-1, \cos(0.84\pi)] \cup [\cos(0.24\pi), 1] \\
0, & x \in [\cos(0.68\pi), \cos(0.4\pi)]
\end{cases} \]

<table>
<thead>
<tr>
<th>( n/\text{bit size} )</th>
<th>( \delta_{\text{minimax}} )</th>
<th>( \delta_{\text{opt}} )</th>
<th>( \delta_{\text{naive}} )</th>
<th>( \delta_{\text{lattice}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>17/8</td>
<td>( 2.631 \cdot 10^{-3} )</td>
<td>0.01787</td>
<td>0.04687</td>
<td>0.01787</td>
</tr>
<tr>
<td>22/8</td>
<td>( 6.709 \cdot 10^{-4} )</td>
<td>0.01609</td>
<td>0.03046</td>
<td>0.01609</td>
</tr>
<tr>
<td>62/21†</td>
<td>( 1.278 \cdot 10^{-8} )</td>
<td>( 1.564 \cdot 10^{-6} )</td>
<td>( 8.203 \cdot 10^{-6} )</td>
<td>( 1.621 \cdot 10^{-6} )</td>
</tr>
</tbody>
</table>
• efficient method for obtaining quasi-optimal fixed point FIR filters
• very scalable \((n = 100)\) problems usually take < 10 seconds
• available as an open source C++ library:
  https://github.com/sfilip/fquantizer

Future work:
• low-complexity coefficients
• IIR filters with fixed-point coefficients

Thank you!