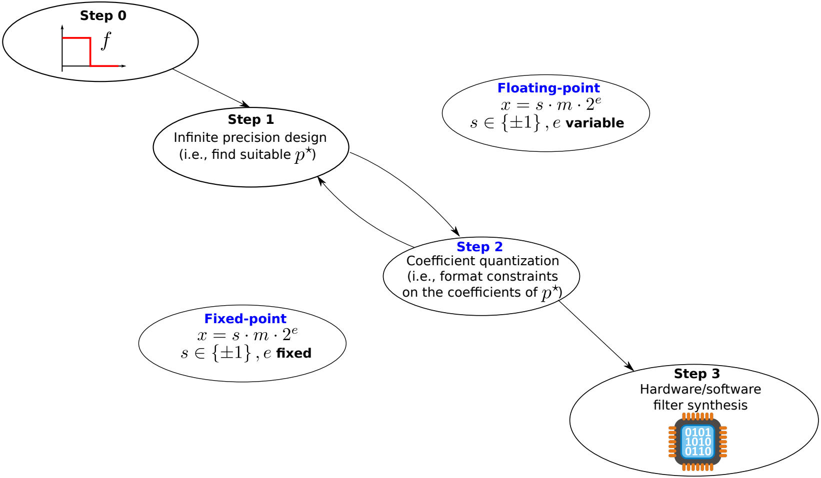
A lattice basis reduction approach for the design of finite wordlength FIR filters

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

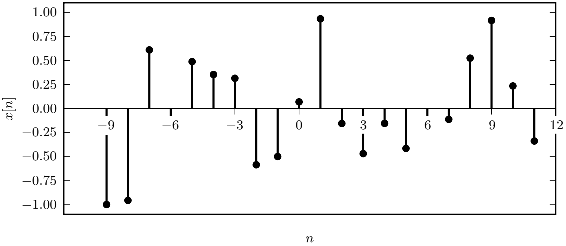
Digital filter design chain: outline



Digital signal processing = the study of **discrete-time signals**

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

Example:



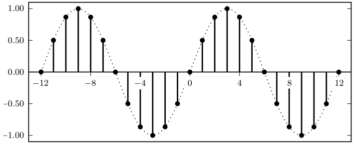
$\rightarrow$ measured data = signals:

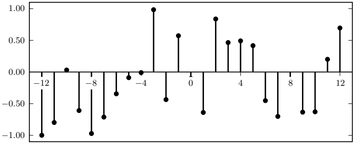
- temperature readings;
- content of data packets (in network transmissions);
- stock price changes;
- etc.

How do we extract information from signals?

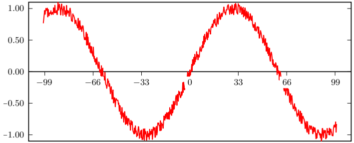
Examples:

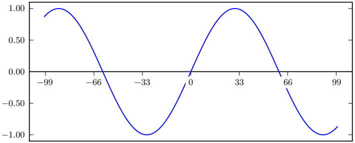
1. Periodicity?

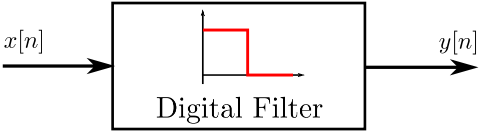




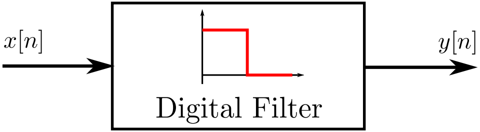
2. Noise?







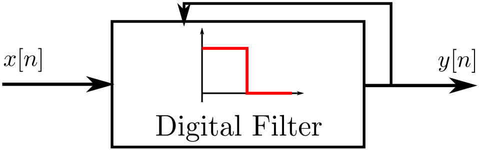
$\rightarrow$ we get two categories of linear filters



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

$\rightarrow$ we get two categories of linear filters

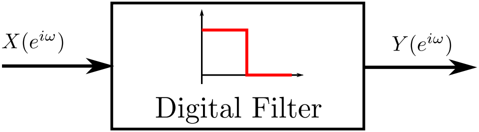
- finite impulse response (**FIR**) filters



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

→ we get two categories of linear filters

- finite impulse response (**FIR**) filters
- infinite impulse response (**IIR**) filters



$$Y(e^{i\omega}) = X(e^{i\omega})H(e^{i\omega})$$

→ we get two categories of linear filters

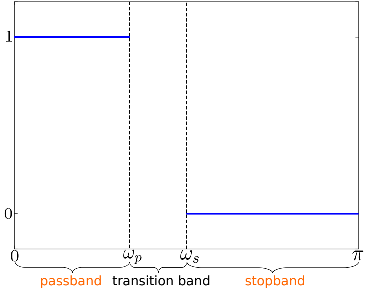
- finite impulse response (**FIR**) filters
  H is a polynomial
- infinite impulse response (**IIR**) filters
  H is a rational function

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

→ **focus** on FIR filters (with linear phase)

Input:

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



Input:

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

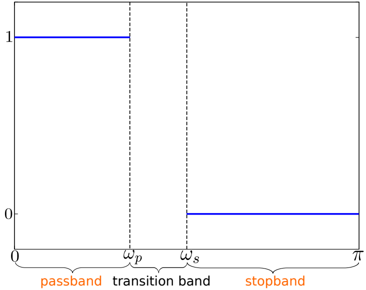
Output:

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

s.t.

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

is **minimal**



Input:

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

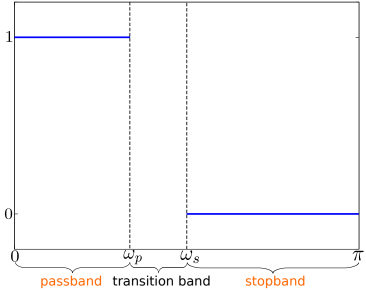
Output:

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

s.t.

$$\delta = \underbrace{\max_{\omega \in \Omega}}_{x \in X} |\underbrace{D(\omega)}_{f(x)} - \underbrace{H(e^{i\omega})}_{p^\star(x)}|$$

is **minimal**



Input:

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

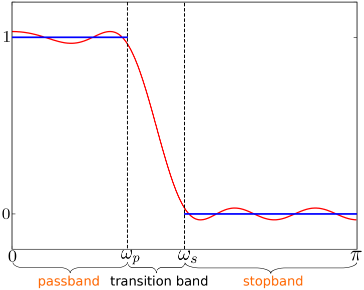
Output:

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

s.t.

$$\delta = \underbrace{\max_{\omega \in \Omega}}_{x \in X} |\underbrace{D(\omega)}_{f(x)} - \underbrace{H(e^{i\omega})}_{p^\star(x)}|$$

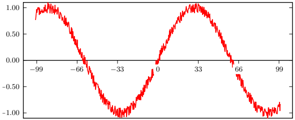
is **minimal**

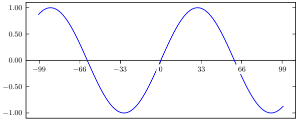


Example:
$\to$ degree $n = 8$

$$p^\star(x) = \sum_{k=0}^{8} h_k T_k(x)$$

Where is this filter useful?

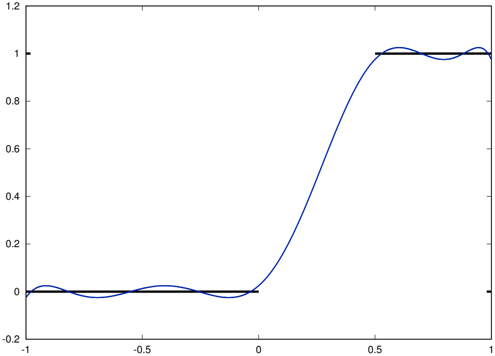




Floating point arithmetic is sometimes expensive (*i.e.*, in embedded systems)

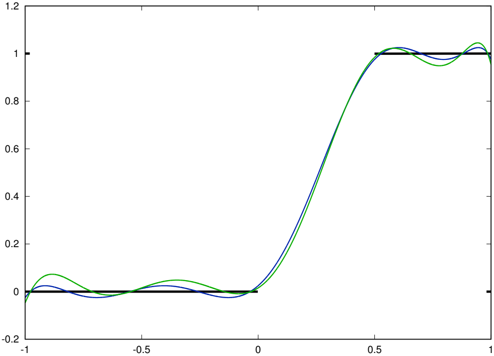
$\rightarrow$ use fixed-point arithmetic (scaled integers in a certain range)

Two main reasons: small price + fast hardware
$\rightarrow$ frequent for signal processing applications



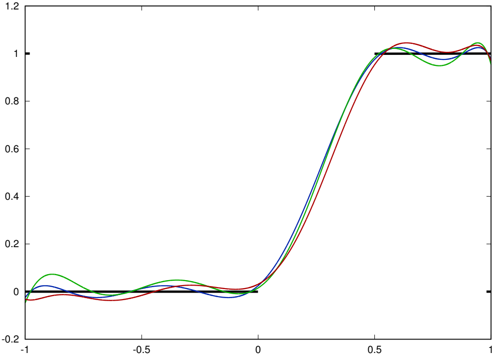
$$p^\star(x) = \frac{a_0}{2} T_0(x) + \sum_{k=1}^{9} a_k T_k(x), \delta \simeq 0.0249$$

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



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

$\rightarrow$ 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: $\delta \simeq 0.0731$



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

$\rightarrow$ 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 $\delta \simeq 0.0468$

$\rightarrow$ take $\varphi_0(x) = \frac{T_0(x)}{2^6}, \varphi_1(x) = \frac{T_1(x)}{2^5}, \ldots, \varphi_9(x) = \frac{T_9(x)}{2^5}$

$\rightarrow$ **want** to solve

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

$\rightarrow$ take $\varphi_0(x) = \frac{T_0(x)}{2^6}, \varphi_1(x) = \frac{T_1(x)}{2^5}, \dots, \varphi_9(x) = \frac{T_9(x)}{2^5}$

$\rightarrow$ **actually** solve

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

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

Other heuristic approaches:

$\rightarrow$ MATLAB uses a stochastic-based method

$\rightarrow$ with no format constraints, **interpolation** at well-placed nodes works well
Why not do something similar here?

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

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

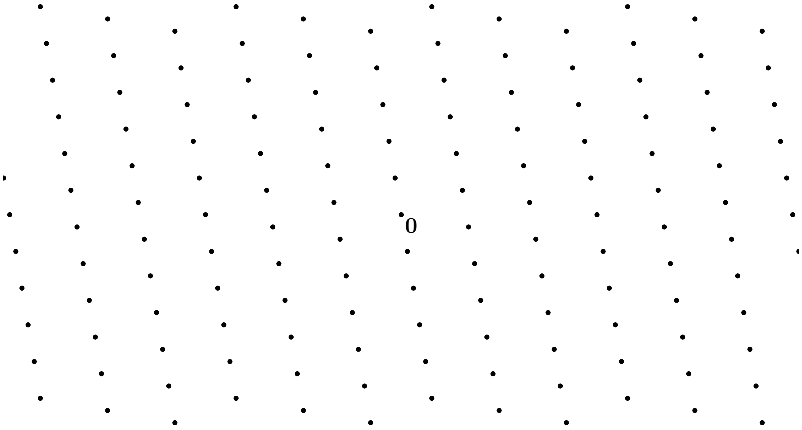
$\rightarrow$ with no format constraints, **interpolation** at well-placed nodes works well
Why not do something similar here?

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

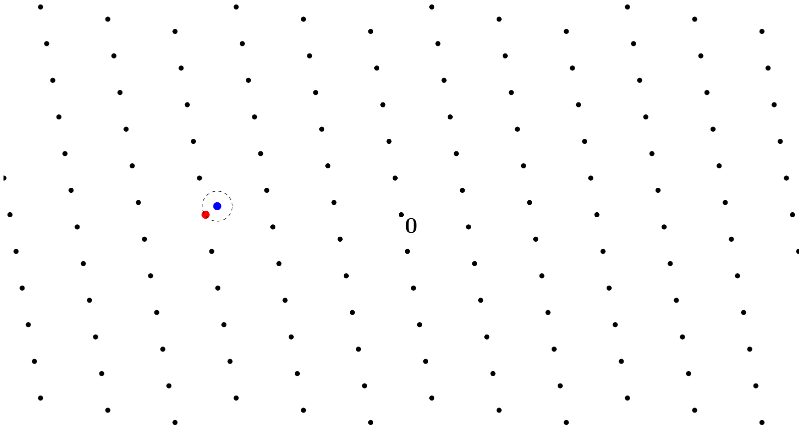
$$\sum_{i=0}^{n} m_k \begin{bmatrix} \varphi_k(x_0) \\ \varphi_k(x_1) \\ \vdots \\ \varphi_k(x_n) \end{bmatrix} \simeq \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}$$

How can we solve this pseudo interpolation problem?
$\rightarrow$ state it as an Euclidean lattice problem

$\rightarrow$ consider the case $n = 1$

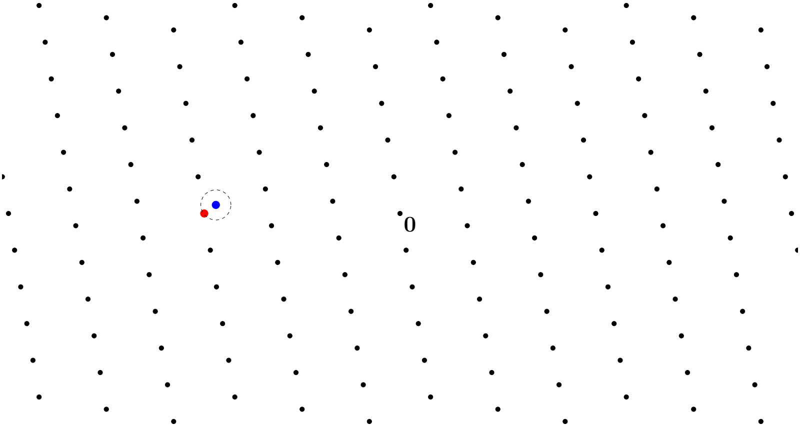


$\rightarrow$ consider the case $n = 1$



NP-hard problem in general.

$\rightarrow$ consider the case $n = 1$



NP-hard problem in general. Efficient algorithms: approximate solutions.

$\rightarrow$ problem in Euclidean lattice theory:

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

→ problem in Euclidean lattice theory:

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

→ use fast, approximate solvers:

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

→ on the toy example, we get the **optimal** result

What interpolation points do we use?

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

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

What interpolation points do we use?

$\boldsymbol{x} = \{x_0, \ldots, x_n\}$
$\mathbb{R}_n[x] = \mathrm{span}_{\mathbb{R}} \{\varphi_0, \ldots, \varphi_n\} \qquad (\varphi_k(x) = T_k(x))$

$\rightarrow$ Lagrange interpolation at x:

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

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

What interpolation points do we use?

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

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

$\rightarrow$ Lagrange interpolation at x:

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

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

$\rightarrow$ important quantity: Lebesgue constant

$$\Lambda_{X,\boldsymbol{x}} = \max_{x \in X} \sum_{i=0}^{n} |\ell_i(x)|$$

What interpolation points do we use?

$\boldsymbol{x} = \{x_0, \ldots, x_n\}$
$\mathbb{R}_n[x] = \mathrm{span}_{\mathbb{R}} \{\varphi_0, \ldots, \varphi_n\} \qquad (\varphi_k(x) = T_k(x))$

$\rightarrow$ Lagrange interpolation at x:

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

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

$\rightarrow$ important quantity: Lebesgue constant

$$\Lambda_{X,\boldsymbol{x}} = \max_{x \in X} \sum_{i=0}^{n} |\ell_i(x)|$$

$\rightarrow$ measures **quality** of x for doing interpolation:

$$\boxed{\forall f \in \mathcal{C}(X), \|f - \mathcal{L}_{\mathrm{x}} f\|_{\infty,X} \leqslant (1 + \Lambda_{X,\boldsymbol{x}}) \|f - p^\star\|_{\infty,X}}$$

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

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

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

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

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

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

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

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

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

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

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

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

$\rightarrow$ NP-hard problem [Çivril & Magdon-Ismail, 2009]
$\rightarrow$ greedy algorithm based on QR factorization [Sommariva & Vianello, 2010]

What is a suitable discretization of X?
$\rightarrow$ use the theory of weakly admissible meshes [Calvi & Levenberg, 2008]

What is a suitable discretization of X?
$\rightarrow$ use the theory of weakly admissible meshes [Calvi & Levenberg, 2008]

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

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

What is a suitable discretization of X?
$\rightarrow$ use the theory of weakly admissible meshes [Calvi & Levenberg, 2008]

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

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

$\rightarrow$ our finite wordlength polynomial p satisfies:

$$\boxed{||f - p||_{\infty,X} \leqslant (1 + \Lambda_{X,\mathbf{x}})||f - p_{opt}||_{\infty,X} + \Lambda_{X,\mathbf{x}}\delta}$$

where:

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

$\rightarrow$ specification:

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

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

$\rightarrow$ specification:

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

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

n/bit size	Minimax δ_{minimax}	Optimal δ_{opt}	Naive rounding δ_{naive}	Lattice-based δ_{lattice}
17/8	$2.631 \cdot 10^{-3}$	0.01787	0.04687	0.01787
22/8	$6.709 \cdot 10^{-4}$	0.01609	0.03046	0.01609
$62/21^{\dagger}$	$1.278 \cdot 10^{-8}$	*$1.564 \cdot 10^{-6}$*	$8.203 \cdot 10^{-6}$	$1.621 \cdot 10^{-6}$

- 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!