The Parks McClellan algorithm: a scalable approach for designing FIR filters

Silviu Filip
under the supervision of N. Brisebarre and G. Hanrot
(AriC, LIP, ENS Lyon)

Rencontres Arithmétiques de l'Informatique Mathématique (RAIM)
Rennes, April 7-9, 2015

Digital Signal Processing

- became increasingly relevant over the past 4 decades:

$$\text{ANALOG} \rightarrow \text{DIGITAL}$$

Digital Signal Processing

- became increasingly relevant over the past 4 decades:

$$\text{ANALOG} \rightarrow \text{DIGITAL}$$

- think of:
  - data communications (ex: Internet, HD TV and digital radio)
  - audio and video systems (ex: CD, DVD, BD players)
  - many more

Digital Signal Processing

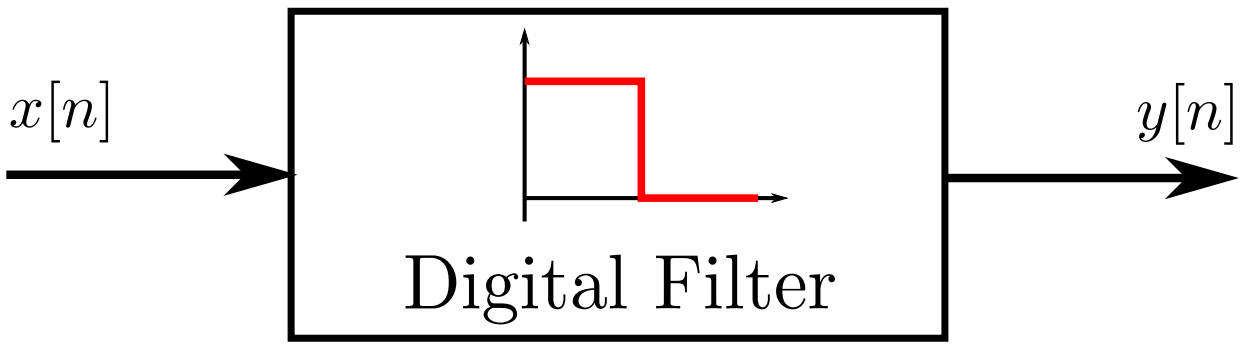
- became increasingly relevant over the past 4 decades:

$$\text{ANALOG} \rightarrow \text{DIGITAL}$$

- think of:
  - data communications (ex: Internet, HD TV and digital radio)
  - audio and video systems (ex: CD, DVD, BD players)
  - many more

What are the 'engines' powering all these?

Digital filters

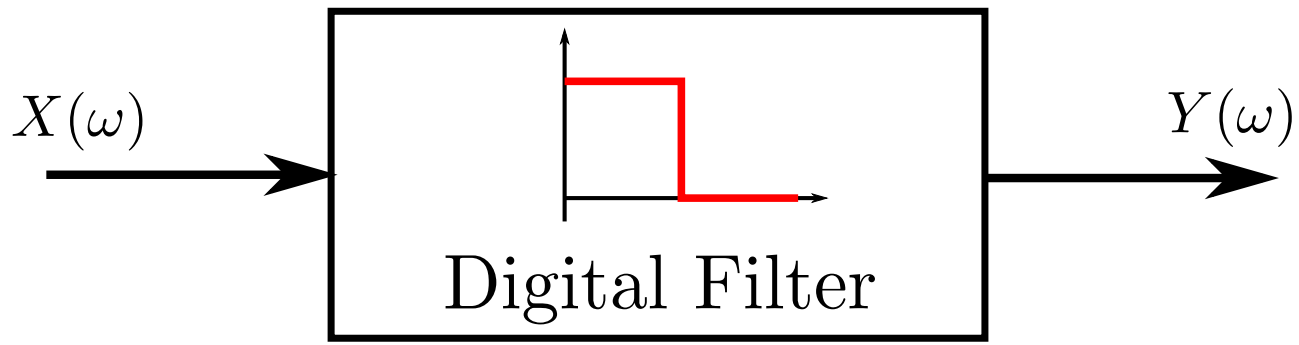


$$y[n] = \sum_{k=0}^{N} b_k x[n-k] - \sum_{k=1}^{M} a_k y[n-k], a_k, b_k \in \mathbb{R}$$

$\rightarrow$ we get two categories of filters

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

Digital filters



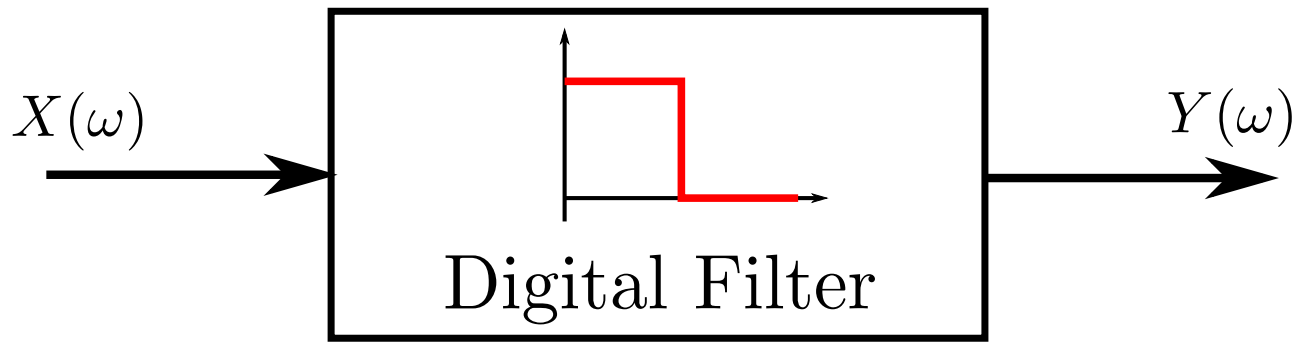
$$Y(\omega) = X(\omega)H(\omega), \omega \in [0, \pi]$$

$\rightarrow$ we get two categories of filters

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

$\rightarrow$ natural to work in the **frequency** domain

Digital filters



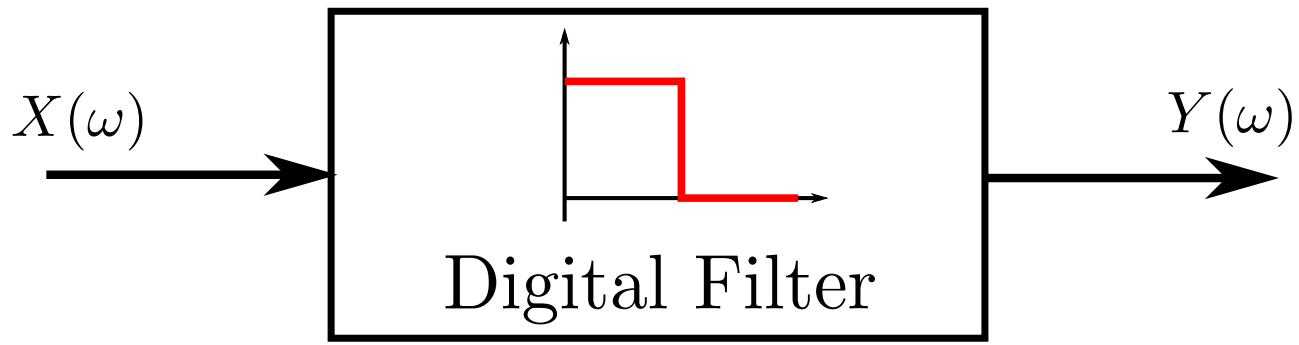
$$Y(\omega) = X(\omega)H(\omega), \omega \in [0, \pi]$$

$\rightarrow$ we get two categories of filters

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

$\rightarrow$ natural to work in the **frequency** domain
H is the **transfer function** of the filter

Digital filters



$$Y(\omega) = X(\omega)H(\omega), \omega \in [0, \pi]$$

$\rightarrow$ we get two categories of filters

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

$\rightarrow$ natural to work in the **frequency** domain
H is the **transfer function** of the filter

The filtering framework

Steps:

1. **derive a concrete mathematical representation of the filter**
   $\rightarrow$ use theory of minimax approximation

The filtering framework

Steps:

1. **derive a concrete mathematical representation of the filter**
   $\rightarrow$ use theory of minimax approximation

2. **quantization of the filter coefficients using fixed-point or floating-point formats**
   $\rightarrow$ use tools from algorithmic number theory (euclidean lattices)

The filtering framework

Steps:

1. **derive a concrete mathematical representation of the filter**
   $\rightarrow$ use theory of minimax approximation

2. **quantization of the filter coefficients using fixed-point or floating-point formats**
   $\rightarrow$ use tools from algorithmic number theory (euclidean lattices)

3. **hardware synthesis of the filter**

The filtering framework

Steps:

1. **derive a concrete mathematical representation of the filter**
   $\rightarrow$ use theory of minimax approximation
2. **quantization of the filter coefficients using fixed-point or floating-point formats**
   $\rightarrow$ use tools from algorithmic number theory (euclidean lattices)
3. **hardware synthesis of the filter**

Today's focus: **first step** for FIR filters

Finite Impulse Response (FIR) filters

- large class of filters, with a lot of desirable properties

Usual representation: $H(\omega) = \sum_{k=0}^{n} a_k \cos(\omega k)$

Finite Impulse Response (FIR) filters

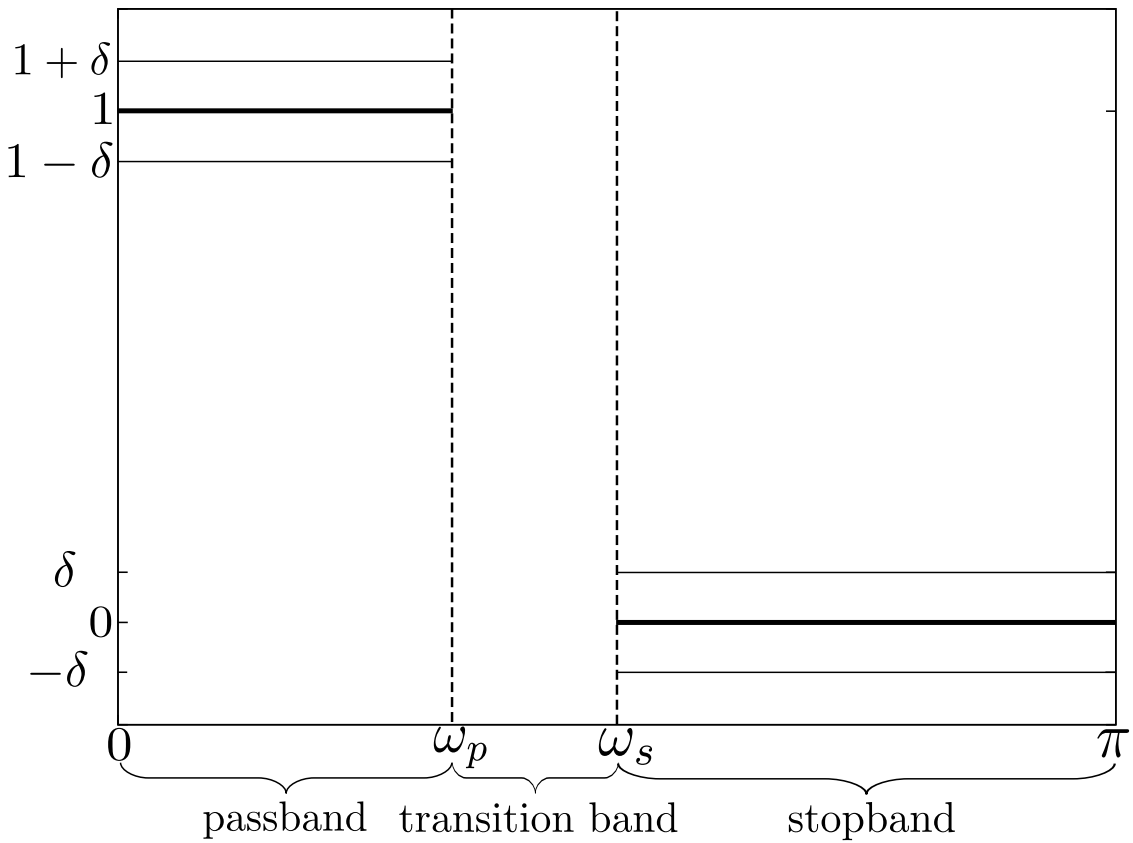
- large class of filters, with a lot of desirable properties

Usual representation: $H(\omega) = \sum_{k=0}^{n} a_k \cos(\omega k) = \sum_{k=0}^{n} a_k T_k(\cos(\omega))$
$\rightarrow$ if $x = \cos(\omega)$, view H in the basis of Chebyshev polynomials

Finite Impulse Response (FIR) filters

- large class of filters, with a lot of desirable properties

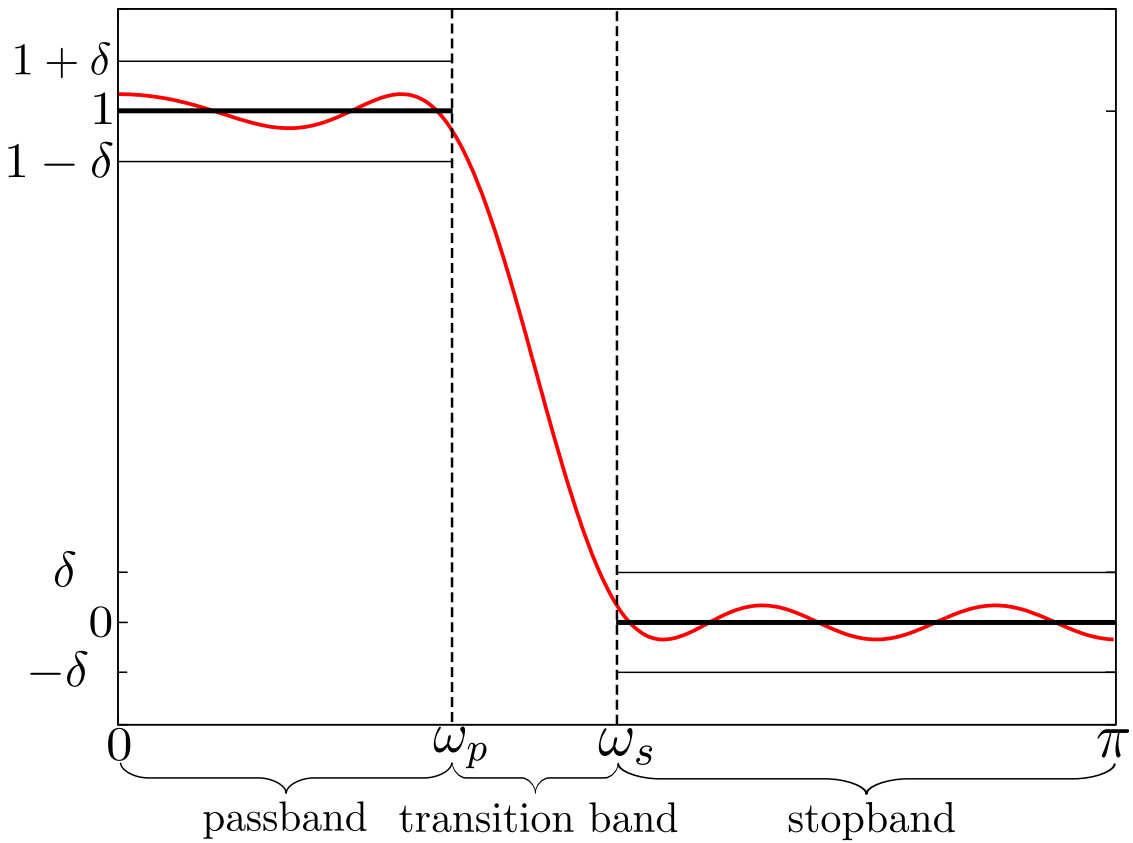
Usual representation: $H(\omega) = \sum_{k=0}^{n} a_k \cos(\omega k) = \sum_{k=0}^{n} a_k T_k(\cos(\omega))$
$\rightarrow$ if $x = \cos(\omega)$, view H in the basis of Chebyshev polynomials
Specification:



Finite Impulse Response (FIR) filters

- large class of filters, with a lot of desirable properties

Usual representation: $H(\omega) = \sum_{k=0}^{n} a_k \cos(\omega k) = \sum_{k=0}^{n} a_k T_k(\cos(\omega))$
$\rightarrow$ if $x = \cos(\omega)$, view H in the basis of Chebyshev polynomials
Specification:



$$H(\omega) = \sum_{k=0}^{8} a_k \cos(\omega k)$$

Optimal FIR design with real coefficients

The problem: Given a closed real set F, find an approximation $H(\omega) = \sum_{k=0}^{n} a_k \cos(\omega k)$ of degree n for a continuous function $D(\omega), \omega \in F$ such that

$$\delta = \|E(\omega)\|_{\infty,F} = \max_{\omega \in F} |H(\omega) - D(\omega)|$$

is **minimal**.

Optimal FIR design with real coefficients

The solution: characterized by the **Alternation Theorem**

Theorem

*The **unique** solution* $H(\omega) = \sum_{k=0}^{n} a_k \cos(\omega k)$ *has an **error function*** $E(\omega)$*, for which there exist* $n+2$ *values* $\omega_0 < \omega_1 < \cdots < \omega_{n+1}$*, belonging to* F*, such that*

$$E(\omega_i) = -E(\omega_{i+1}) = \pm\delta,$$

for $i = 0, \ldots, n$.

Optimal FIR design with real coefficients

The solution: characterized by the **Alternation Theorem**

Theorem

*The **unique** solution* $H(\omega) = \sum_{k=0}^{n} a_k \cos(\omega k)$ *has an **error function*** $E(\omega)$*, for which there exist* $n+2$ *values* $\omega_0 < \omega_1 < \cdots < \omega_{n+1}$*, belonging to* F*, such that*

$$E(\omega_i) = -E(\omega_{i+1}) = \pm\delta,$$

for $i = 0, \ldots, n$.

$\rightarrow$ well studied in Digital Signal Processing literature

1972: Parks and McClellan

$\rightarrow$ based on a powerful iterative approach from Approximation Theory:

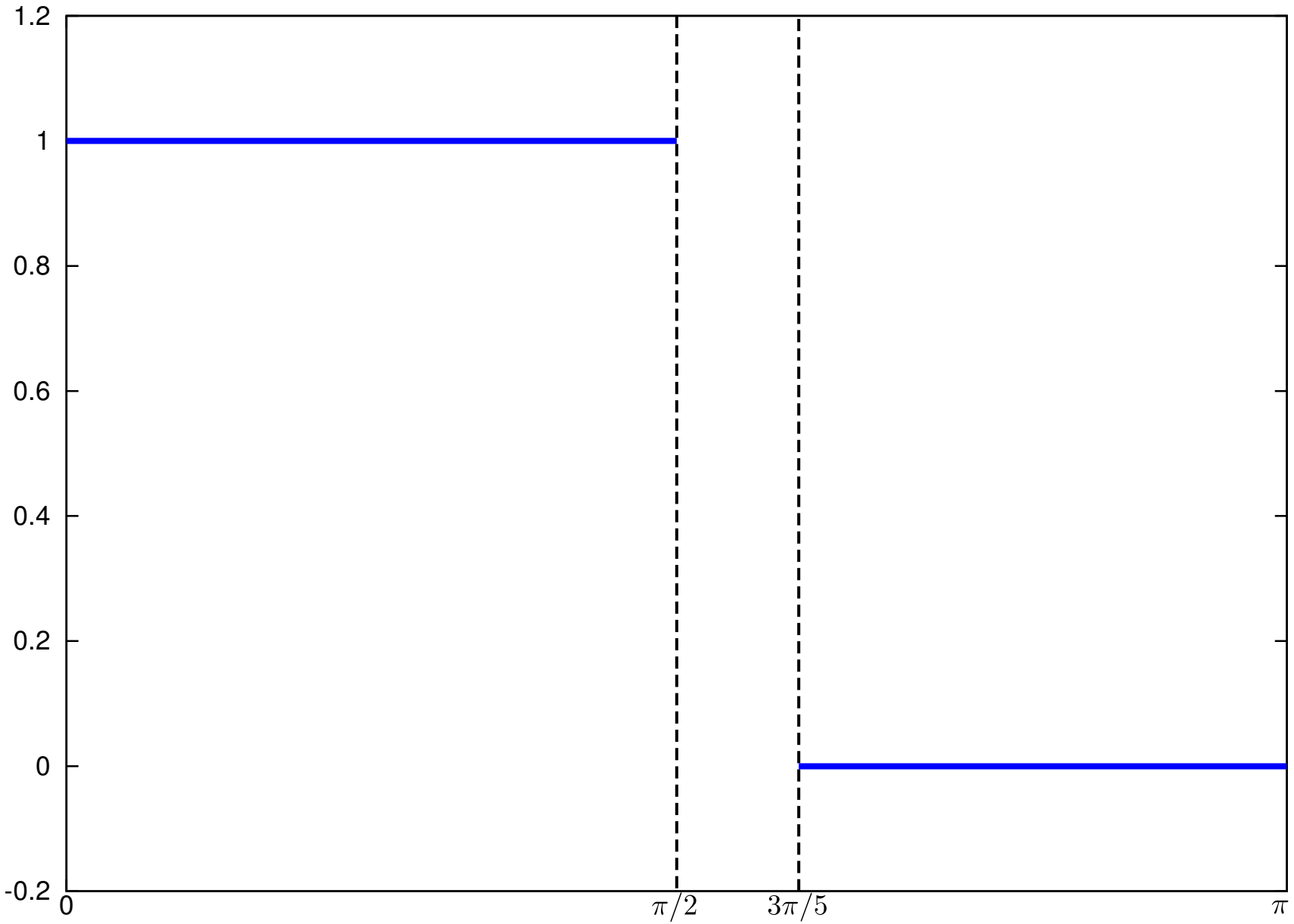
1934: Remez

The Parks-McClellan design method: Motivation

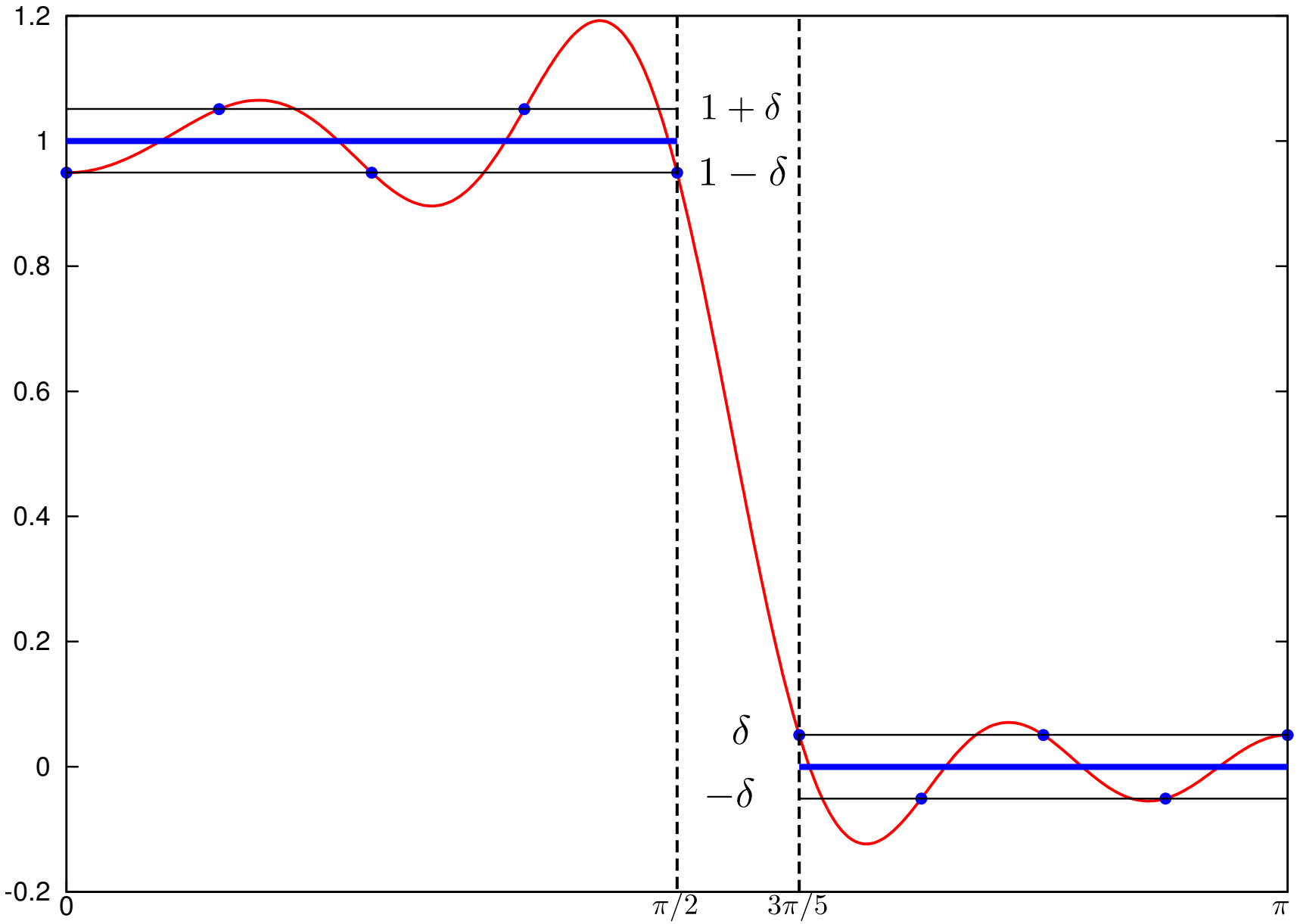
Why work on such a problem?

- one of the most well-known filter design methods
- no concrete study about its numerical behavior in practice
- need for high degree ($n > 500$) filters + existing implementations not able to provide them (e.g. MATLAB, SciPy, GNURadio)
- useful for attacking the coefficient quantization problem

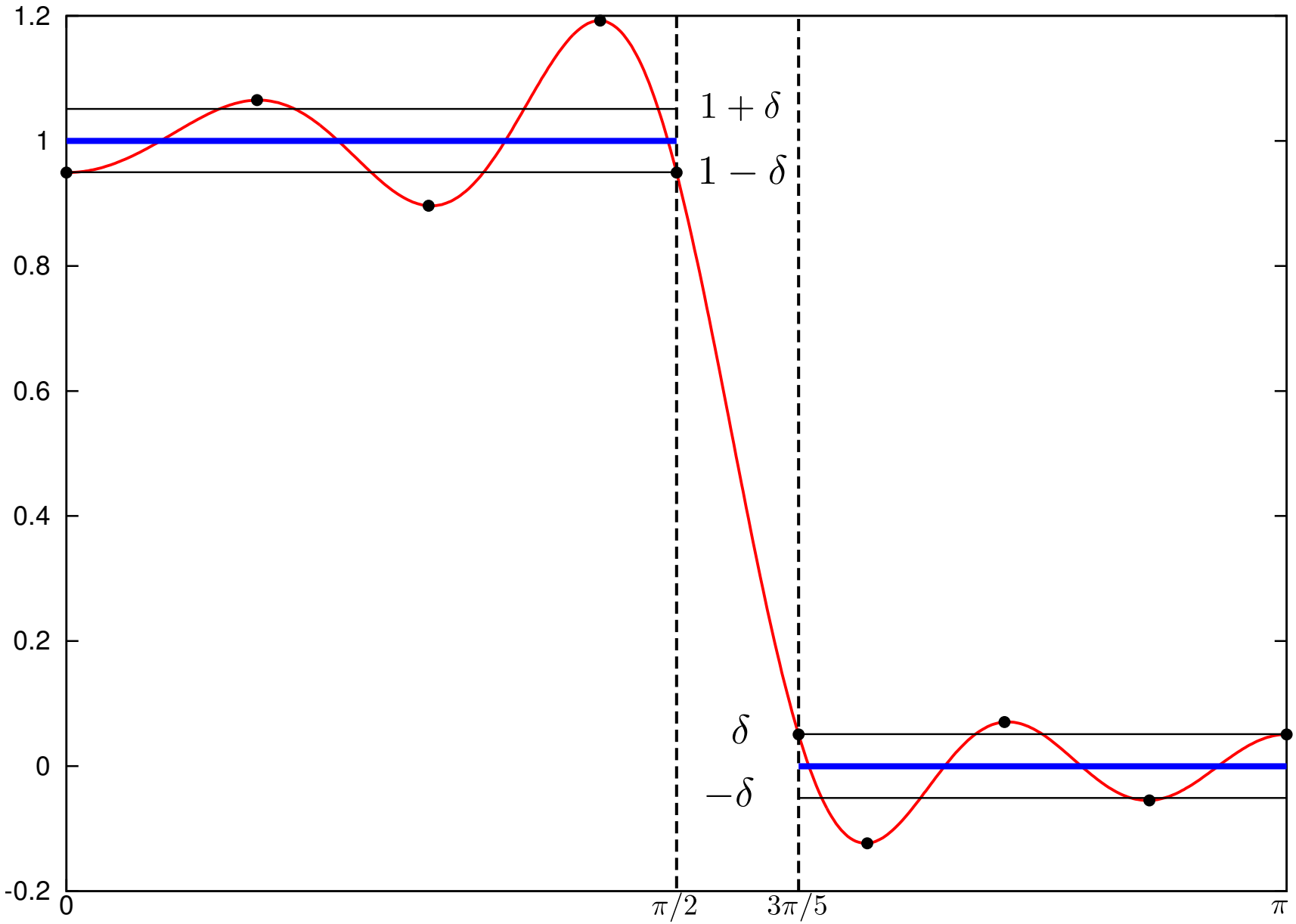
The Parks-McClellan design method: Example



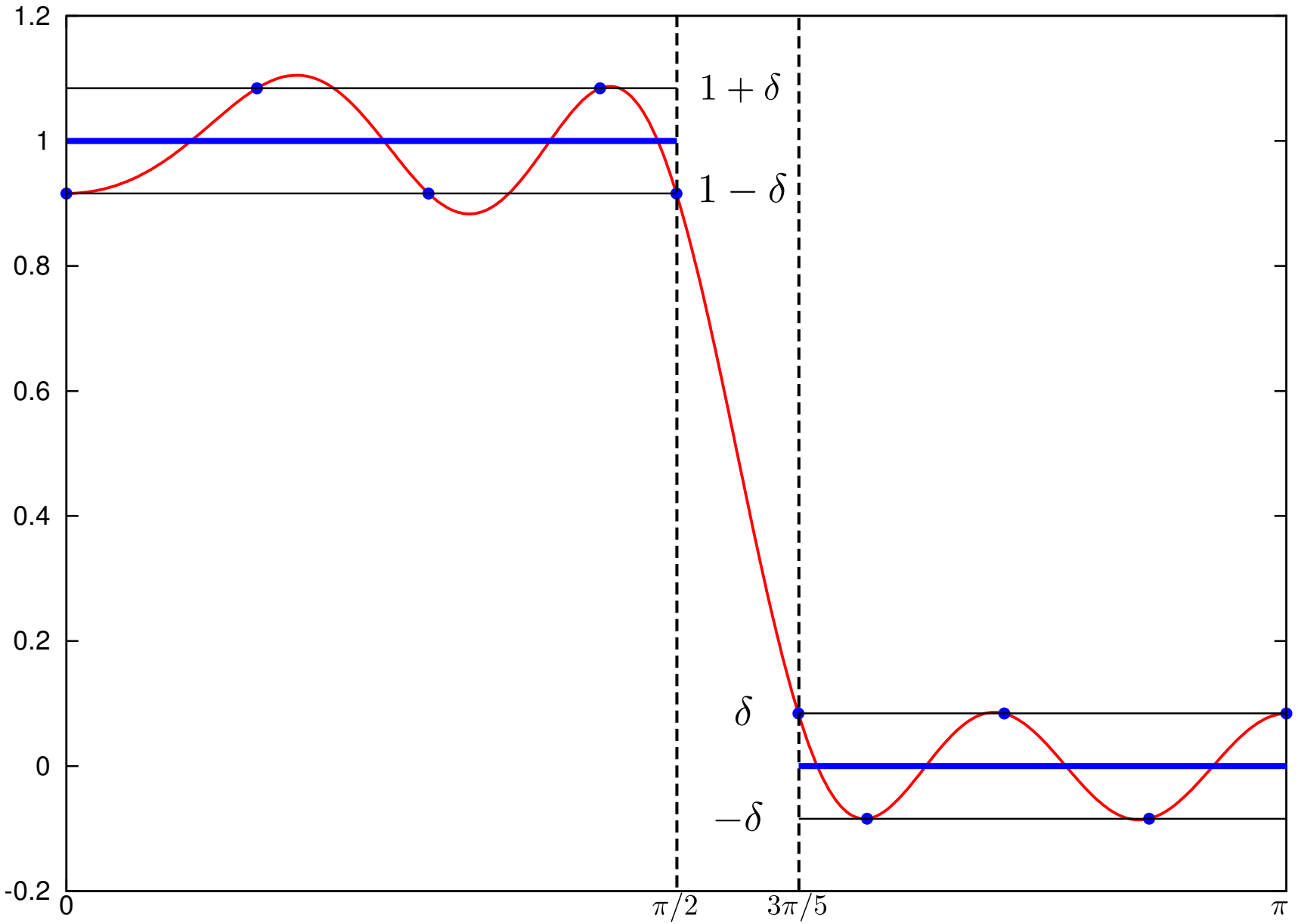
The Parks-McClellan design method: Example



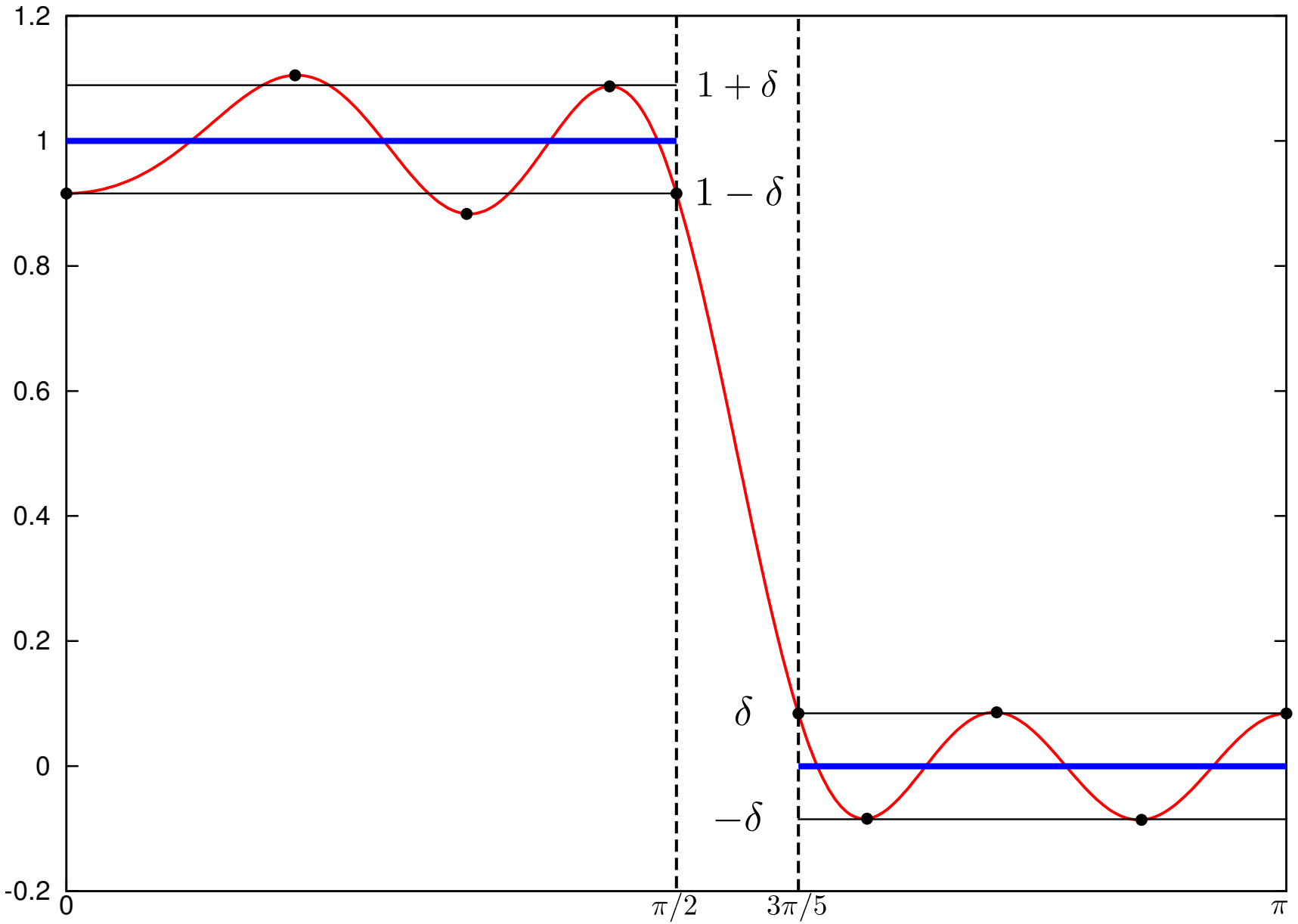
The Parks-McClellan design method: Example



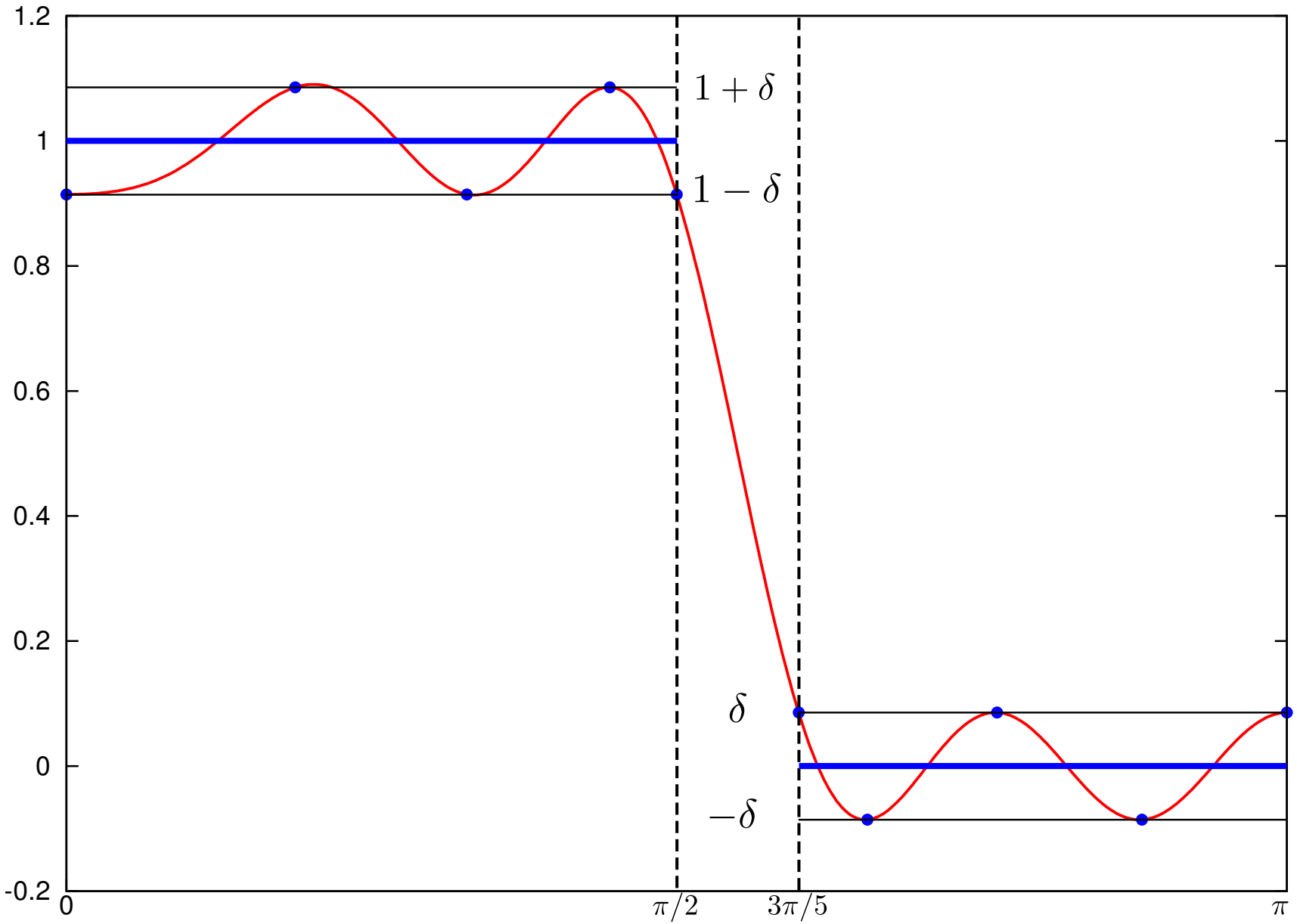
The Parks-McClellan design method: Example



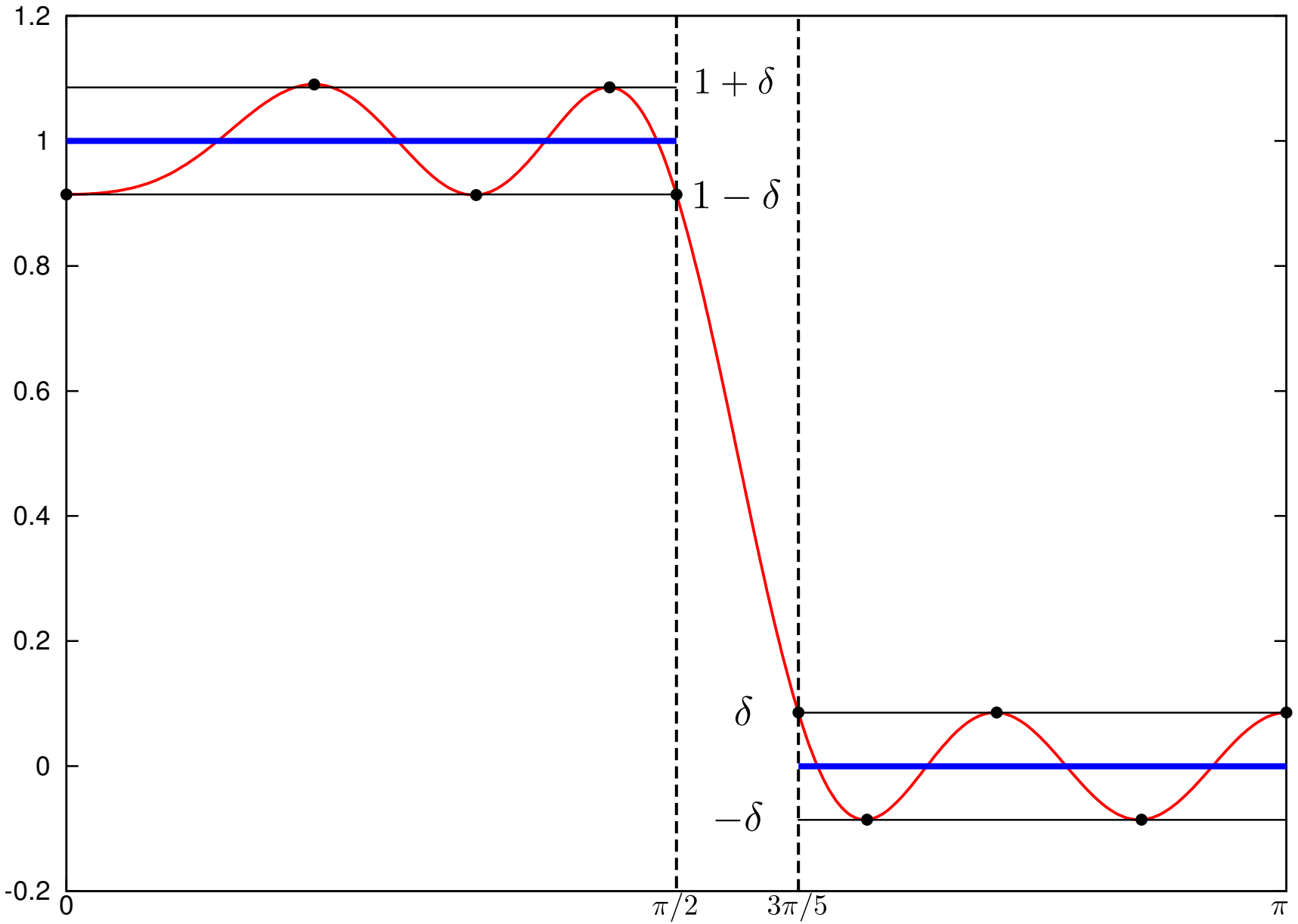
The Parks-McClellan design method: Example



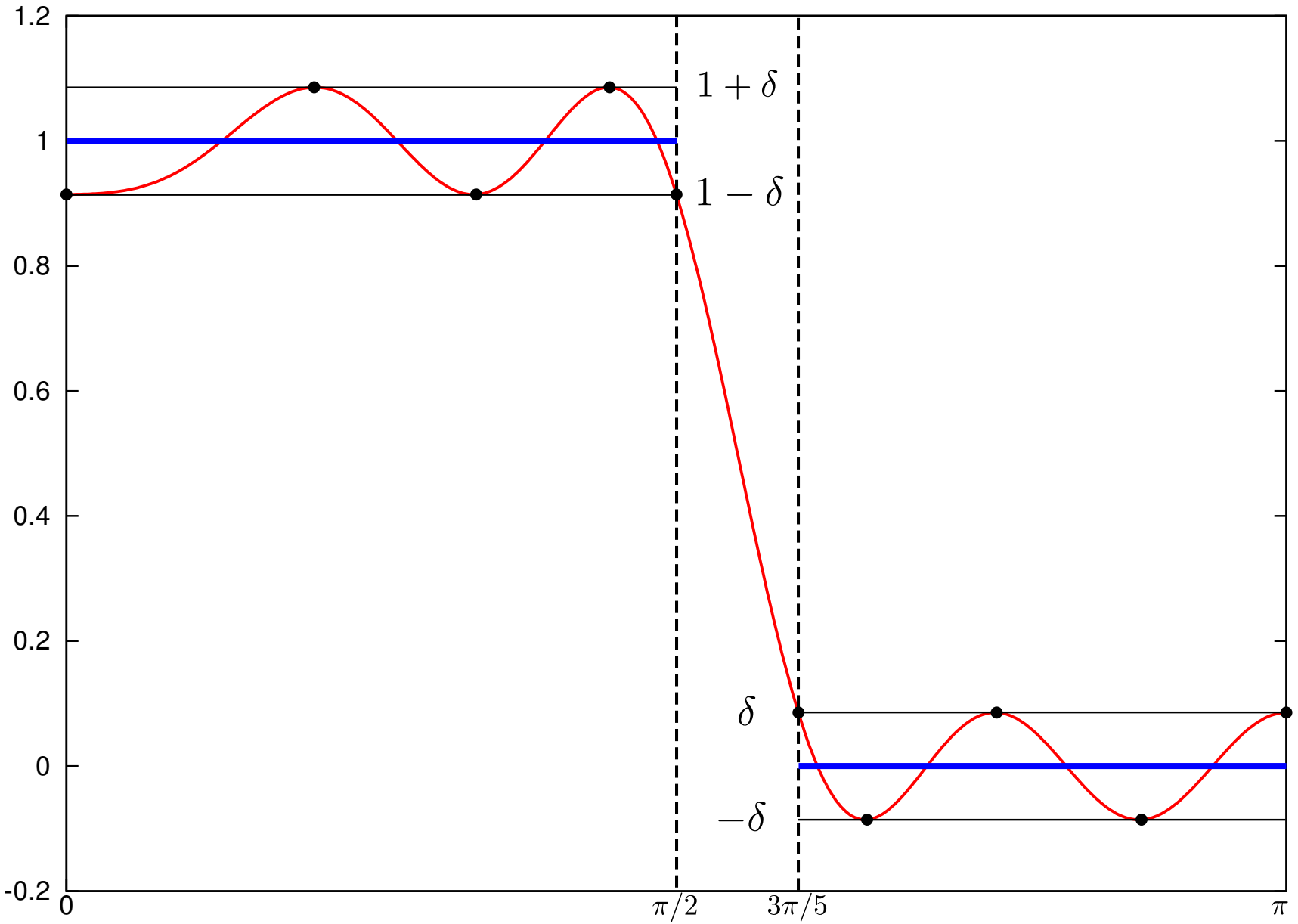
The Parks-McClellan design method: Example



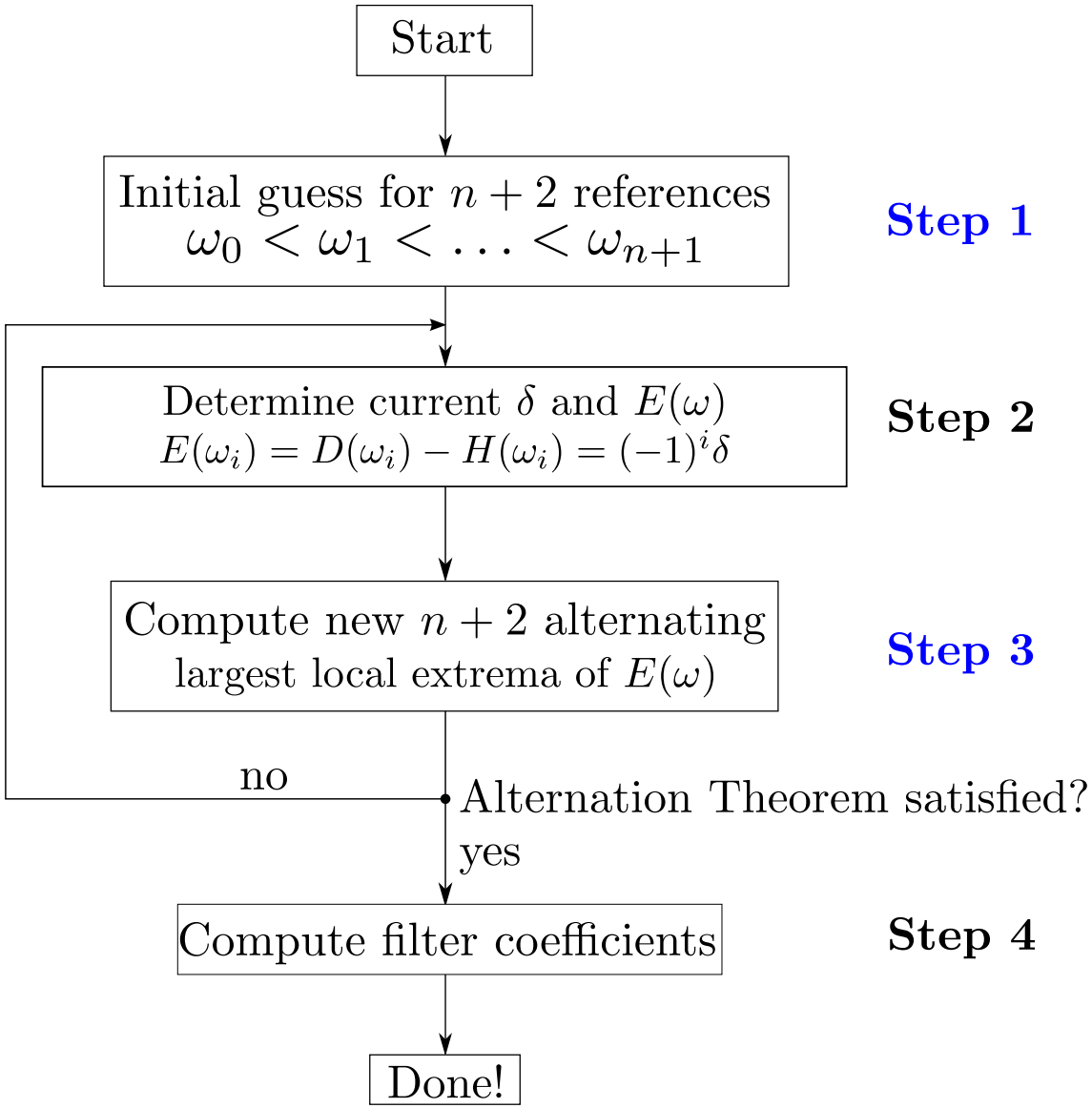
The Parks-McClellan design method: Example



The Parks-McClellan design method: Example



The Parks-McClellan design method: Steps



Step 1: Choosing the $n + 2$ initial references

Traditional approach: take the $n + 2$ references uniformly from F
$\rightarrow$ can lead to convergence problems

Step 1: Choosing the $n + 2$ initial references

Traditional approach: take the $n + 2$ references uniformly from F
$\rightarrow$ can lead to convergence problems

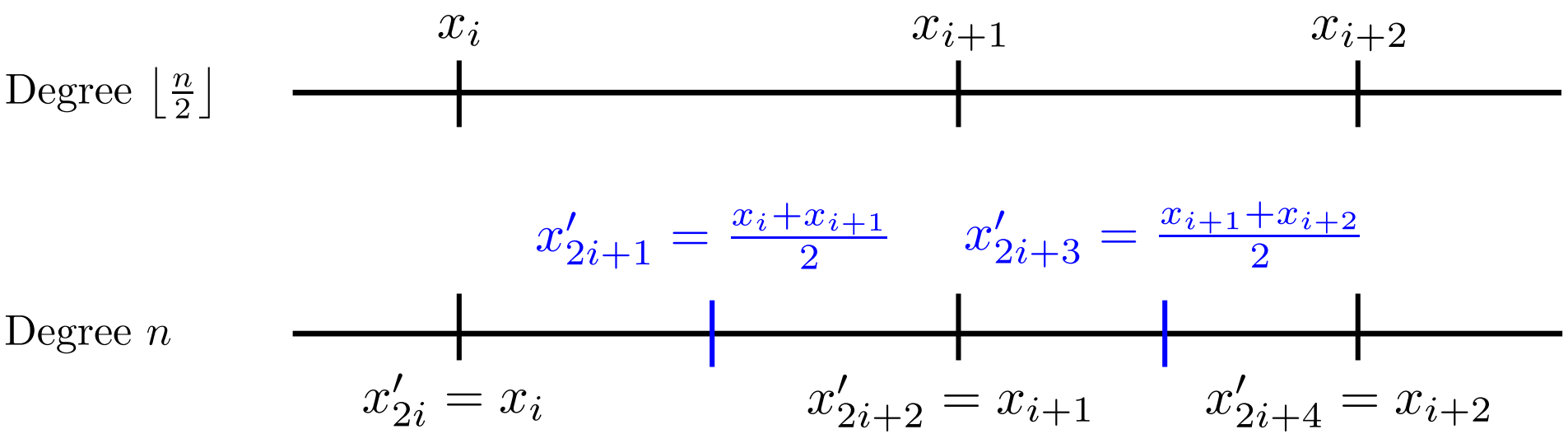
$\rightarrow$ **want to start** from better approximations
Existing approaches: most are not general enough and/or costly to execute

Step 1: Choosing the $n+2$ initial references

Traditional approach: take the $n+2$ references uniformly from F
$\rightarrow$ can lead to convergence problems

$\rightarrow$ **want to start** from better approximations
Existing approaches: most are not general enough and/or costly to execute

Our approach: extrema position extrapolation from smaller filters



$\rightarrow$ although empirical, this **reference scaling** idea is rather robust in practice

Step 2: Computing the current error function $E(\omega)$ and δ

Amounts to solving a linear system in $a_0, \ldots, a_n$ and δ.

$$\begin{bmatrix} 1 & \cos(\omega_0) & \cdots & \cos(n\omega_0) & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & \cos(\omega_n) & \cdots & \cos(n\omega_n) & (-1)^n \\ 1 & \cos(\omega_{n+1}) & \cdots & \cos(n\omega_{n+1}) & (-1)^{n+1} \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_n \\ \delta \end{bmatrix} = \begin{bmatrix} D(\omega_0) \\ \vdots \\ D(\omega_n) \\ D(\omega_{n+1}) \end{bmatrix}$$

$\rightarrow$ solving system directly: can be numerically unstable

Step 2: Computing the current error function $E(\omega)$ and δ

Amounts to solving a linear system in $a_0, \ldots, a_n$ and δ.

$$\begin{bmatrix} 1 & \cos(\omega_0) & \cdots & \cos(n\omega_0) & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & \cos(\omega_n) & \cdots & \cos(n\omega_n) & (-1)^n \\ 1 & \cos(\omega_{n+1}) & \cdots & \cos(n\omega_{n+1}) & (-1)^{n+1} \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_n \\ \delta \end{bmatrix} = \begin{bmatrix} D(\omega_0) \\ \vdots \\ D(\omega_n) \\ D(\omega_{n+1}) \end{bmatrix}$$

$\rightarrow$ solving system directly: can be numerically unstable

$\rightarrow$ use **barycentric** form of Lagrange interpolation [Berrut&Trefethen2004]

Barycentric Lagrange interpolation

Problem: p polynomial with $\deg p \leqslant n$ interpolates f at points x_j, i.e.,

$$p(x_j) = f_j, j = 0, \ldots, n$$

Barycentric Lagrange interpolation

Problem: p polynomial with $\deg p \leqslant n$ interpolates f at points x_j, i.e.,

$$p(x_j) = f_j, j = 0, \ldots, n$$

$\rightarrow$ the barycentric form of p is:

$$p(x) = \frac{\displaystyle\sum_{j=0}^{n} \frac{w_j}{x - x_j} f_j}{\displaystyle\sum_{j=0}^{n} \frac{w_j}{x - x_j}},$$

where $w_j = \dfrac{1}{\prod_{k \neq j}(x_j - x_k)}$.

Cost: $O(n^2)$ for computing *all* w_j, $O(n)$ for evaluating $p(x)$.

Barycentric Lagrange interpolation

Why should we use it?
$\rightarrow$ numerically stable **if the family of interpolation nodes used has a small Lebesgue constant** [Higham2004;Mascarenhas&Camargo2014]

The Lebesgue constant: specific for each grid of points; measures the quality of a polynomial interpolant with respect to the function to be approximated

Barycentric Lagrange interpolation

Why should we use it?
$\rightarrow$ numerically stable **if the family of interpolation nodes used has a small Lebesgue constant** [Higham2004;Mascarenhas&Camargo2014]

The Lebesgue constant: specific for each grid of points; measures the quality of a polynomial interpolant with respect to the function to be approximated

$\rightarrow$ from **empirical observation**, the families of points used inside the Parks-McClellan algorithm (Step 1 + Step 3) usually converge to sets of points with **small** Lebesgue constant

Step 3: Finding the local extrema of $E(\omega)$

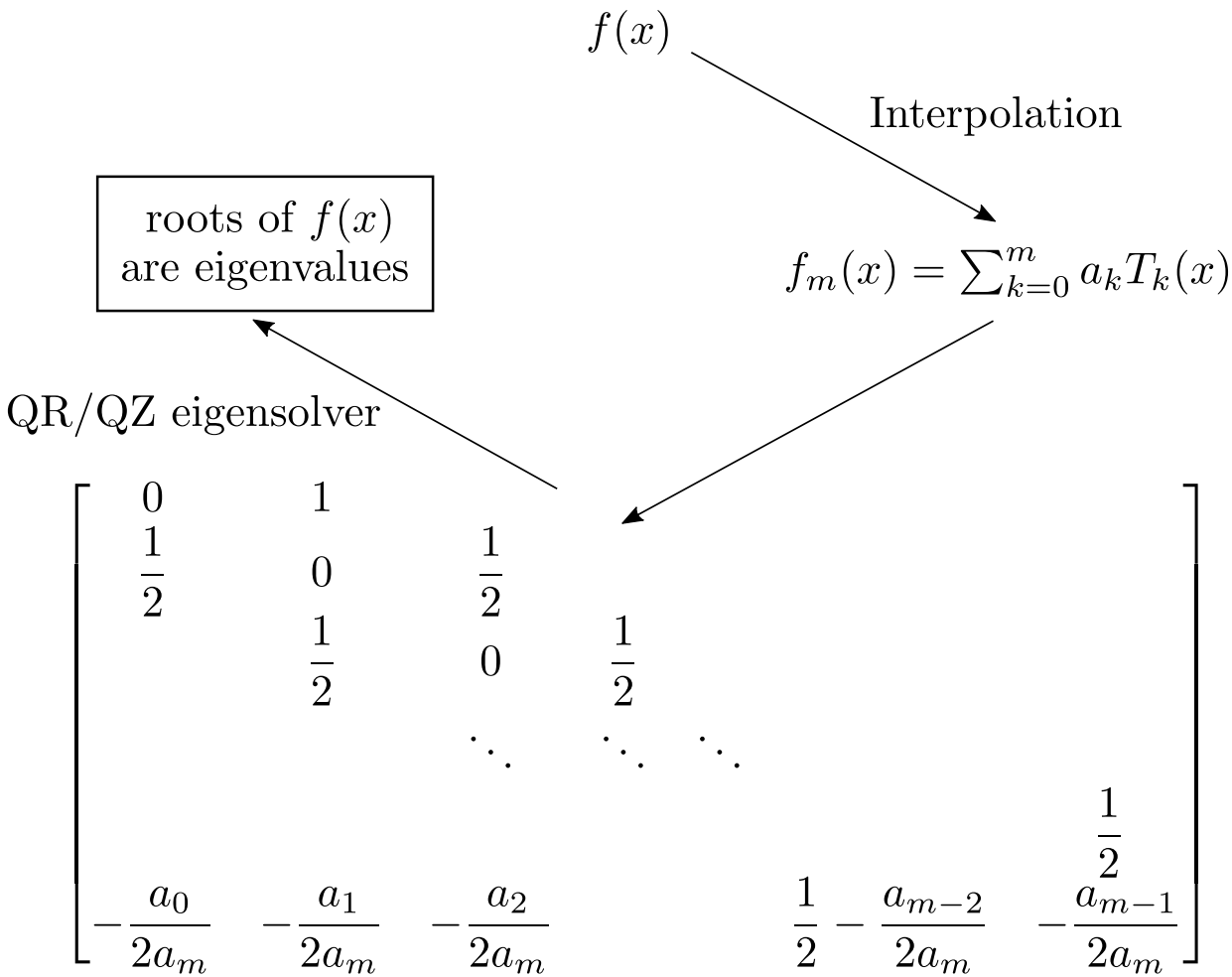
Traditional approach: evaluate $E(\omega)$ on a dense grid of uniformly distributed points (in practice it is usually $16n$)

$\rightarrow$ can sometimes fail to find all the extrema

$\rightarrow$ need for a more robust alternative

Chebyshev-proxy rootfinders

Some promoters: I.J. Good, J.P. Boyd, L.N. Trefethen



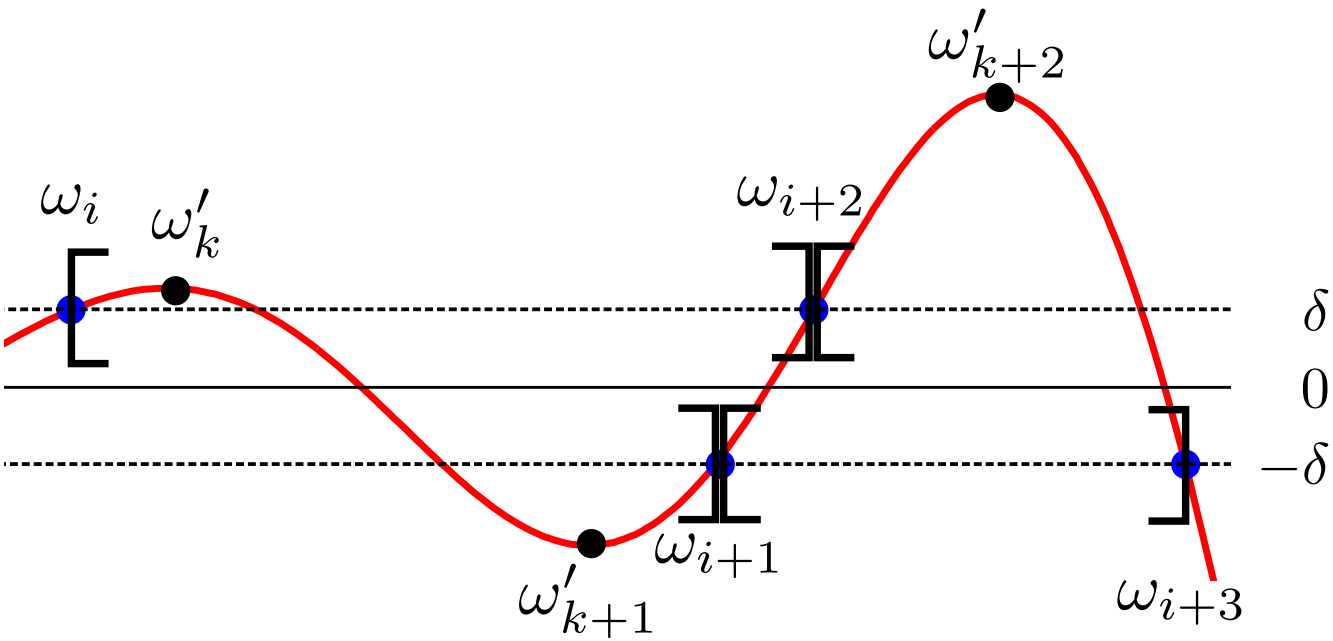
→ [Pachon&Trefethen2009] use it to implement the Remez algorithm

Step 3: Finding the local extrema of $E(\omega)$

$\rightarrow$ **local extrema** of $E(\omega) \rightarrow$ **roots** of $E'(\omega)$

$\rightarrow$ at each iteration, we know:

- $E(\omega)$ usually has *very close* to $n+2$ local extrema inside F
- placement information for the local extrema

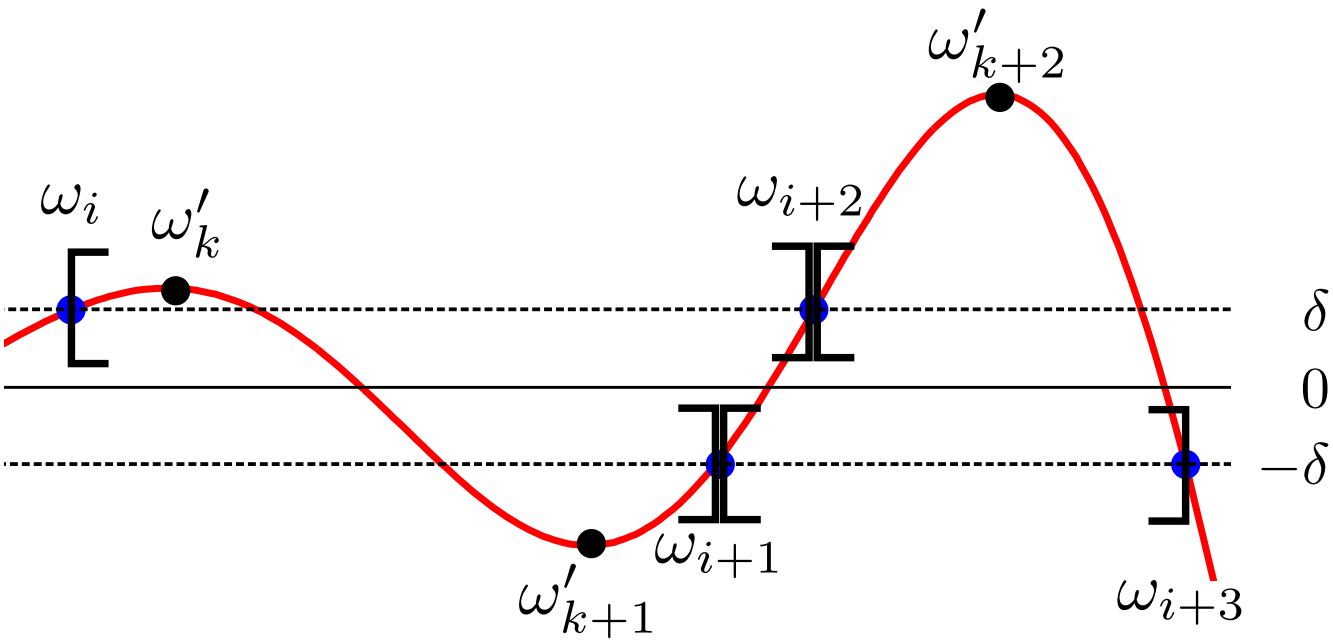


Step 3: Finding the local extrema of $E(\omega)$

$\rightarrow$ **local extrema** of $E(\omega)$ $\rightarrow$ **roots** of $E'(\omega)$

$\rightarrow$ at each iteration, we know:

- $E(\omega)$ usually has *very close* to $n+2$ local extrema inside F
- placement information for the local extrema



$\rightarrow$ Chebyshev-proxy rootfinders + interval subdivision

Cost: $O(n^2)$ operations

Step 4: Recover coefficients of $H(\omega)$ upon convergence

$\rightarrow$ can use the *Inverse Discrete Fourier Transform*

Step 4: Recover coefficients of $H(\omega)$ upon convergence

$\rightarrow$ can use the *Inverse Discrete Fourier Transform*

$\rightarrow$ implement it using Clenshaw's algorithm for computing linear combinations of Chebyshev polynomials (numerically robust approach)

Cost: $O(n^2)$ arithmetic operations

Results

Examples:

1. degree $n = 100$ filter with passband $[0, 0.4\pi]$ and stopband $[0.5\pi, \pi]$
2. degree $n = 100$ filter with passbands $[0, 0.2\pi], [0.6\pi, \pi]$ and stopband $[0.3\pi, 0.5\pi]$
3. degree $n = 520$ filter with passband $[0, 0.99\pi]$ and stopband centered at π
4. degree $n = 53248$ lowpass filter with passband $\left[0, \frac{1}{8192}\pi\right]$ and stopband $\left[\frac{3}{8192}\pi, \pi\right]$

Results

$\rightarrow$ running times (in seconds) on a 3.6 GHz 64-bit Intel Xeon(R) E5-1620

Problem	Uniform (sequential)	GNURadio	MATLAB	SciPy
Example 1 ($n = 100$)	0.0112	NC	0.1491	0.3511
Example 2 ($n = 100$)	0.0395	NC	NC	NC
Example 3 ($n = 520$)	0.3519	NC	NC	NC
Example 4 ($n = 53248$)	NC	NC	NC	NC

Example (degree)	Uniform (sequential)	Uniform (parallel)	Scaling (sequential)	Scaling (parallel)
Example 1 ($n = 100$)	0.0112	0.0073	0.0147	0.011
Example 2 ($n = 100$)	0.0395	0.0274	0.0339	0.0275
Example 3 ($n = 520$)	0.3519	0.2251	0.0982	0.0716
Example 4 ($n = 53248$)	NC	NC	537.8	162.6

Perspectives

Conclusion:

- improved the practical behavior of a well known polynomial approximation algorithm for filter design
  $\rightarrow$ use numerically stable barycentric Lagrange interpolation + rootfinders without sacrifices in efficiency
- this approach can take huge advantage of parallel architectures

Future work:

- provide a complete toolchain for constructing FIR filters (approximation + quantization + hardware synthesis)
- tackle the IIR filter setting (rational fraction)
  - non-linear problem
  - constraints: poles located inside the unit circle