LINEAR-PHASE FIR FILTER DESIGN BY LINEAR PROGRAMMING

Often it is desirable that an FIR filter be designed to minimize the Chebyshev error subject to linear constraints that the Parks-McClellan algorithm does not allow. An example described by Rabiner includes time domain constraints. Another example comes from a communication application [1] — given $h_1(n)$, design $h_2(n)$ so that $h(n) = h_1(n) * h_2(n)$ is an Nyquist-M filter (i.e. $h(Mn) = 0$ for all $n \neq 0$). Such constraints are linear in $h_2(n)$. (In the special case that $h_1(n) = \delta(n)$, $h_2(n)$ is itself a Nyquist-M filter, and is often used for interpolation).

Linear programming formulations of approximation problems (and optimization problems in general) are very attractive because well developed algorithms exist (namely the simplex algorithm and more recently, interior point methods) for solving such problems. Although linear programming requires significantly more computation than the Remez algorithm, for many problems it is a very viable technique. Furthermore, this approach is very flexible — it allows arbitrary linear equality and *inequality* constraints.

A linear program is not a type of computer program — the term *linear program* is from optimization theory. A linear program is an optimization problem having a certain form. Specifically, the cost function and the constraints are both linear functions of the independent variables.

A linear program has the following form.

$$\left\{ \begin{array}{ll} \text{Minimize} & c_1\,x_1 + c_2\,x_2 + \cdots + c_n\,x_n \\ \text{over} & x_k \;:\; 1 \le k \le n \\ \text{such that} & A\,x \le b \end{array} \right\} \tag{1}$$

The cost function is:

$$f(x) = c_1\,x_1 + c_2\,x_2 + \cdots + c_n\,x_n.$$

The variables are:

$$x_1,\; x_2,\; \ldots,\; x_n$$

The constraints are:

$$A\,x \le b$$

where A is a matrix of size $m \times n$,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

$A\,x \le b$ means that each components of the vector $A\;x$ is less than the corresponding component of the vector b. Here m is the number of constraints.

Note that

$$f(x) = c_1\, x_1 + c_2\, x_2 + \cdots + c_n\, x_n.$$

can be written in vector form as

$$f(x) = c^t \cdot x$$

where

$$c^t = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix}$$

and

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

With this vector notation, the linear program can be written in the following notation.

$$\left\{ \begin{array}{rl} \text{Minimize} & c^t \cdot x \\ \text{over} & x_k \ : \ 1 \leq k \leq n \\ \text{such that} & A\, x \leq b \end{array} \right\} \tag{2}$$

THE MATLAB `lp` COMMAND

The Matlab command `lp` in the Optimization Toolbox can be used to solve a linear program. Here is the command description.

```
>> help lp

 LP     Linear programming.
    X=LP(f,A,b) solves the linear programming problem:

              min f'x    subject to:   Ax <= b
               x

    X=LP(f,A,b,VLB,VUB) defines a set of lower and upper
    bounds on the design variables, X, so that the solution is always in
    the range VLB <= X <= VUB.

    X=LP(f,A,b,VLB,VUB,X0) sets the initial starting point to X0.

    X=LP(f,A,b,VLB,VUB,X0,N) indicates that the first N constraints defined
    by A and b are equality constraints.

    X=LP(f,A,b,VLB,VUB,X0,N,DISPLAY) controls the level of warning
    messages displayed.  Warning messages can be turned off with
    DISPLAY = -1.

    [x,LAMBDA]=LP(f,A,b) returns the set of Lagrangian multipliers,
    LAMBDA, at the solution.

    [X,LAMBDA,HOW] = LP(f,A,b) also returns a string how that indicates
    error conditions at the final iteration.

    LP produces warning messages when the solution is either unbounded
    or infeasible.
```

EXAMPLE

For example, suppose we have four variables x_1, x_2, x_3, x_4 and we want to minimize

$$3\,x_1 + 2\,x_2 - x_3 + 5\,x_4$$

subject to the constraints

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 &\leq 5 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_3 &\geq 0 \\ x_4 &\geq 0. \end{aligned}$$

This can be written as

$$\left\{ \begin{array}{lll} \text{Minimize} & c^t \cdot x & \\ \text{over} & x_k & : \; 1 \leq k \leq 4 \\ \text{such that} & A\,x \leq b & \end{array} \right\} \tag{3}$$

where

$$c^t = \begin{bmatrix} 3 & 2 & -1 & 5 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

EXAMPLE

We can find the values of x_k solving this constrained optimization problem using the Matlab `lp` command as follows.

```
>> c = [3 2 -1 5]';

>> A = [1 1 1 1;
     -1 0 0 0;
     0 -1 0 0;
     0 0 -1 0;
     0 0 0 -1];

>> b = [5; 0; 0; 0; 0];

>> x = lp(c,A,b)

x =
     0
     0
     5
     0
```

The optimal solution is therefore

$$x_1 = 0,\ x_2 = 0,\ x_3 = 5,\ x_4 = 0.$$

EXAMPLE 2

Here is an example with an *equality* constraint.

Suppose we have four variables x_1, x_2, x_3, x_4 and we want to minimize

$$3\,x_1 + 2\,x_2 - x_3 + 5\,x_4$$

subject to the constraints

$$\begin{aligned} 2\,x_1 - x_3 &= 3 \\ x_1 + x_2 + x_3 + x_4 &\leq 5 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_3 &\geq 0 \\ x_4 &\geq 0. \end{aligned}$$

We can also find the values of x_k solving this constrained optimization using the Matlab command `lp`.

The syntax

```
x = lp(c,A,b,[],[],[],N);
```

indicates that the first N constraints defined by A and b are *equality* constraints.

The Matlab code for solving this problem is shown on the following page.

EXAMPLE 2

```
>> c = [3 2 -1 5];

>> A = [2 0 -1 0;
     1 1 1 1;
     -1 0 0 0;
     0 -1 0 0;
     0 0 -1 0;
     0 0 0 -1];
>> b = [3; 5; 0; 0; 0; 0];

>> x = lp(c,A,b,[],[],[],1)

x =

    1.5000
         0
         0
         0
```

The optimal solution is therefore

$$x_1 = 1.5,\ x_2 = 0,\ x_3 = 0,\ x_4 = 0.$$

To understand how the problem of minimizing the weighted error

$$E(\omega) = W(\omega)\,(A(\omega) - D(\omega)) \tag{4}$$

where (assuming a type I FIR filter)

$$A(\omega) = \sum_{n=0}^{M} a(n)\cos(n\omega) \tag{5}$$

can be formulated as a linear program we give a chain of equivalent problems, beginning with the following one.

Find $a(n)$ such that

$$|E(\omega)| \leq \delta, \quad \text{for} \quad 0 \leq \omega \leq \pi \tag{6}$$

for the smallest possible value of δ.

Equivalently, we have the following statement of the problem which treats δ as variable. We have the $M+1$ cosine coefficients $a(n)$ and the variable δ, so the total number of variables is $M+2$.

$$\left\{ \begin{array}{lll} \text{Minimize} & \delta & \\ \quad \text{over} & a(n),\ \delta & \\ \text{such that} & |E(\omega)| \leq \delta \quad \text{for} & \quad 0 \leq \omega \leq \pi \end{array} \right\} \tag{7}$$

We can remove the absolute value:

$$\left\{\begin{array}{rlll} \text{Minimize} & \delta & & \\ \text{over} & a(n),\ \delta & & \\ \text{such that} & E(\omega) \leq \delta & \text{for} & 0 \leq \omega \leq \pi \\ \text{and} & -\delta \leq E(\omega) & \text{for} & 0 \leq \omega \leq \pi \end{array}\right\} \tag{8}$$

Expanding $E(\omega)$:

$$\left\{\begin{array}{rlll} \text{Minimize} & \delta & & \\ \text{over} & a(n),\ \delta & & \\ \text{such that} & W(\omega)\left(A(\omega) - D(\omega)\right) \leq \delta & \text{for} & 0 \leq \omega \leq \pi \\ \text{and} & -\delta \leq W(\omega)\left(A(\omega) - D(\omega)\right) & \text{for} & 0 \leq \omega \leq \pi \end{array}\right\} \tag{9}$$

Moving the variables to the left-hand side:

$$\left\{\begin{array}{rll} \text{Minimize} & \delta & \\ \text{over} & a(n),\ \delta & \\ \text{such that} & A(\omega) - \dfrac{\delta}{W(\omega)} \leq D(\omega) & \text{for} \quad \omega \in B \\ \text{and} & -A(\omega) - \dfrac{\delta}{W(\omega)} \leq -D(\omega) & \text{for} \quad \omega \in B \\ \text{where} & B = \{\omega \subset [0,\pi]; W(\omega) > 0\}. & \end{array}\right\} \quad (10)$$

The variables are $a(0), \ldots, a(M)$ and δ. The cost function and the constraints are linear functions of the variables, hence the formulation is that of a linear program. To be precise, this is a semi-infinite linear program because there is a continuum of constraints — one constraint for each $\omega \in [0, \pi]$ where $W(\omega) > 0$. To obtain a linear program with a finite number of constraints, a suitably dense grid of points distributed in $[0, \pi]$ is commonly used.

Discretize: Use a discretization of the frequency variable

$$\{\omega_k \ : \ 1 \leq k \leq L\}$$

so that $W(\omega_k) > 0$ for all $1 \leq k \leq L$. This discretization need not be uniform, and it will not be uniform in general because we need $W(\omega_k) > 0$.

$$\left\{ \begin{array}{rll} \text{Minimize} & \delta & \\ \text{over} & a(n),\ \delta & \\ \text{such that} & A_k - \dfrac{\delta}{W_k} \leq D_k & \text{for} \quad 1 \leq k \leq L \\ \text{and} & -A_k - \dfrac{\delta}{W_k} \leq -D_k & \text{for} \quad 1 \leq k \leq L \end{array} \right\} \tag{11}$$

where

$$A_k = A(\omega_k), \qquad D_k = D(\omega_k), \qquad W_k = W(\omega_k). \tag{12}$$

The inequalities

$$A_k - \frac{\delta}{W_k} \leq D_k \quad \text{for} \quad 1 \leq k \leq L \tag{13}$$

can be written in matrix form as

$$\begin{bmatrix} A_1 \\ \vdots \\ A_k \\ \vdots \\ A_L \end{bmatrix} - \begin{bmatrix} \frac{\delta}{W_1} \\ \vdots \\ \frac{\delta}{W_k} \\ \vdots \\ \frac{\delta}{W_L} \end{bmatrix} \leq \begin{bmatrix} D_1 \\ \vdots \\ D_k \\ \vdots \\ D_L \end{bmatrix}. \tag{14}$$

The vector of values A_k can be written as

$$\begin{bmatrix} A_1 \\ \vdots \\ A_k \\ \vdots \\ A_L \end{bmatrix} = \begin{bmatrix} 1 & \cos(w_1) & \cdots & \cos(Mw_1) \\ \vdots & & & \vdots \\ 1 & \cos(w_k) & \cdots & \cos(Mw_k) \\ \vdots & & & \vdots \\ 1 & \cos(w_L) & \cdots & \cos(Mw_L) \end{bmatrix} \begin{bmatrix} a(0) \\ a(1) \\ \vdots \\ a(M) \end{bmatrix} \quad (15)$$

or as

$$A = C\,a. \quad (16)$$

Therefore, the inequalities can be written as

$$C\,a - V\,\delta \leq d \quad (17)$$

where

$$V = \begin{bmatrix} \frac{1}{W_1} \\ \frac{1}{W_2} \\ \vdots \\ \frac{1}{W_L} \end{bmatrix}, \qquad d = \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_L \end{bmatrix}. \quad (18)$$

Similarly, the constraints

$$-A_k - \frac{\delta}{W_k} \leq -D_k \quad \text{for} \quad 1 \leq k \leq L \quad (19)$$

can be written as

$$-C\,a - V\,\delta \leq -d. \quad (20)$$

Combining the inequality constraints into one matrix:

$$\begin{bmatrix} C & -V \\ -C & -V \end{bmatrix} \begin{bmatrix} a \\ \delta \end{bmatrix} \leq \begin{bmatrix} d \\ -d \end{bmatrix}. \tag{21}$$

The new constrained minimization problem is:

$$\left\{ \begin{array}{lll} \text{Minimize} & \delta & \\ \quad \text{over} & a(n),\ \delta & \\ \text{such that} & \begin{bmatrix} C & -V \\ -C & -V \end{bmatrix} \begin{bmatrix} a \\ \delta \end{bmatrix} \leq \begin{bmatrix} d \\ -d \end{bmatrix} \end{array} \right\} \tag{22}$$

This fits the standard linear program:

$$\left\{ \begin{array}{lll} \text{Minimize} & c \cdot x & \\ \quad \text{over} & x & \\ \text{such that} & Q\, x \leq b \end{array} \right\} \tag{23}$$

where we use

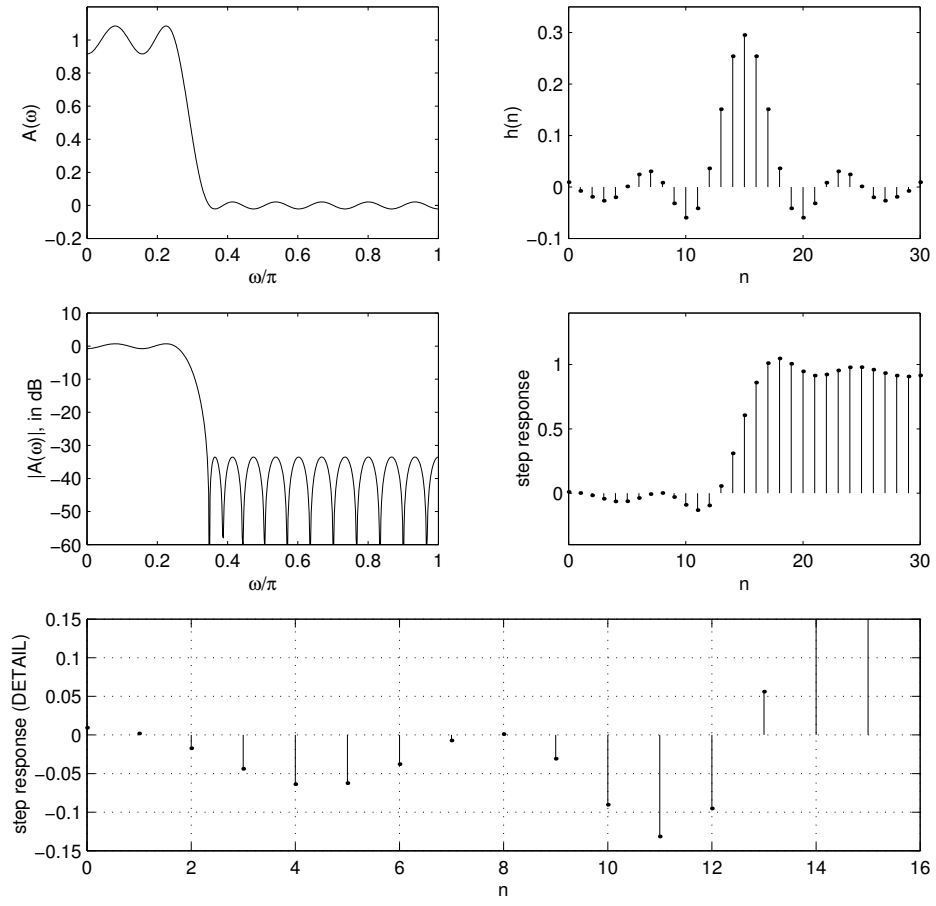
$$Q = \begin{bmatrix} C & -V \\ -C & -V \end{bmatrix}, \quad b = \begin{bmatrix} d \\ -d \end{bmatrix}, \qquad c = [0,\ 0,\ \cdots,\ 0,\ 1]. \tag{24}$$

Then the variables $a(n)$ and δ can be directly obtained from the vector x

$$x = [a(0),\ a(1),\ \cdots,\ a(M),\ \delta]^t. \tag{25}$$

EXAMPLE

To show how to use the linear programming approach, we repeat the design of a length 31 FIR low-pass filter with $K_p = 1$, $K_s = 4$ and $\omega_p = 0.26\pi$, $\omega_s = 0.34\pi$. The filter minimizing the weighted Chebyshev error is illustrated in the figure. It is the same as the filter obtained with the Remez algorithm. Also in the figure, the step response of the filter is plotted. In the next example, constraints will be imposed upon the step response.



```
N = 31;
Kp = 1;
Ks = 4;
wp = 0.26*pi;
ws = 0.34*pi;
wo = 0.30*pi;
L = 500;
w = [0:L]*pi/L;
W = Kp*(w<=wp) + Ks*(w>=ws);
D = (w<=wo);

% remove points where W == 0
SN = 1e-8;
k = (W>SN);
w = w(k);
W = W(k);
D = D(k);

% construct matrices
M = (N-1)/2;
C = cos(w'*[0:M]);
V = 1./W';
Q = [C, -V; -C, -V];
b = [D'; -D'];
c = [zeros(M+1,1); 1];

% solve linear program
x   = lp(c,Q,b);
a   = x(1:M+1);
del = x(M+2);
h   = [a(M+1:-1:2); 2*a(1); a(2:M+1)]/2;
```

REMARKS

This approach is slower than the Remez algorithm because solving a linear program is in general a more difficult problem. In this example, we used a grid density of 500 instead of 1000. Because the required computation depends on both the number of variables and the number of constraints, for long filters (which would require more grid points) the discretization of the constraints may require more computation than is acceptable. This problem can be alleviated by a procedure that invokes a sequence of linear programs. By using a coarse grid density initially, and by appending to the existing set of constraints, new constraints where the constraint violation is greatest, a more accurate solution is obtained without the need to use an overly dense grid.

TIME-DOMAIN CONSTRAINTS

The usefulness of linear programming for filter design is well illustrated by an example described by Rabiner [2]. Using a linear program, he illustrates that some simultaneous restrictions on both the time and the frequency response of a filter can be readily imposed. In his example, the oscillatory behavior of the step response of a low-pass filter is included in the design formulation.

EXAMPLE

In the following design problem, the first set of constraints correspond to the frequency response, and the second set of constraints correspond to the time-domain. $s(n)$ is the step response

$$s(n) = \sum_{k=0}^{n} h(k). \tag{26}$$

N_1 is that region in which the step response oscillates around 0.

$$\left\{ \begin{array}{rll} \text{Minimize} & \delta & \\ \text{over} & a(n),\ \delta & \\ \text{such that} & |E(\omega)| \leq \delta \quad \text{for} & 0 \leq \omega \leq \pi \\ \text{and} & |s(n)| \leq \delta_t \quad \text{for} & 0 \leq n \leq N_1 \end{array} \right\} \tag{27}$$

After discretization, this can be written as the following linear program,

$$\left\{ \begin{array}{rl} \text{Minimize} & \delta \\ \text{over} & a(n),\ \delta \\ \text{such that} & \begin{bmatrix} C & -V \\ -C & -V \\ T & 0 \\ -T & 0 \end{bmatrix} \begin{bmatrix} a \\ \delta \end{bmatrix} \leq \begin{bmatrix} d \\ -d \\ v \\ v \end{bmatrix} \end{array} \right\} \tag{28}$$

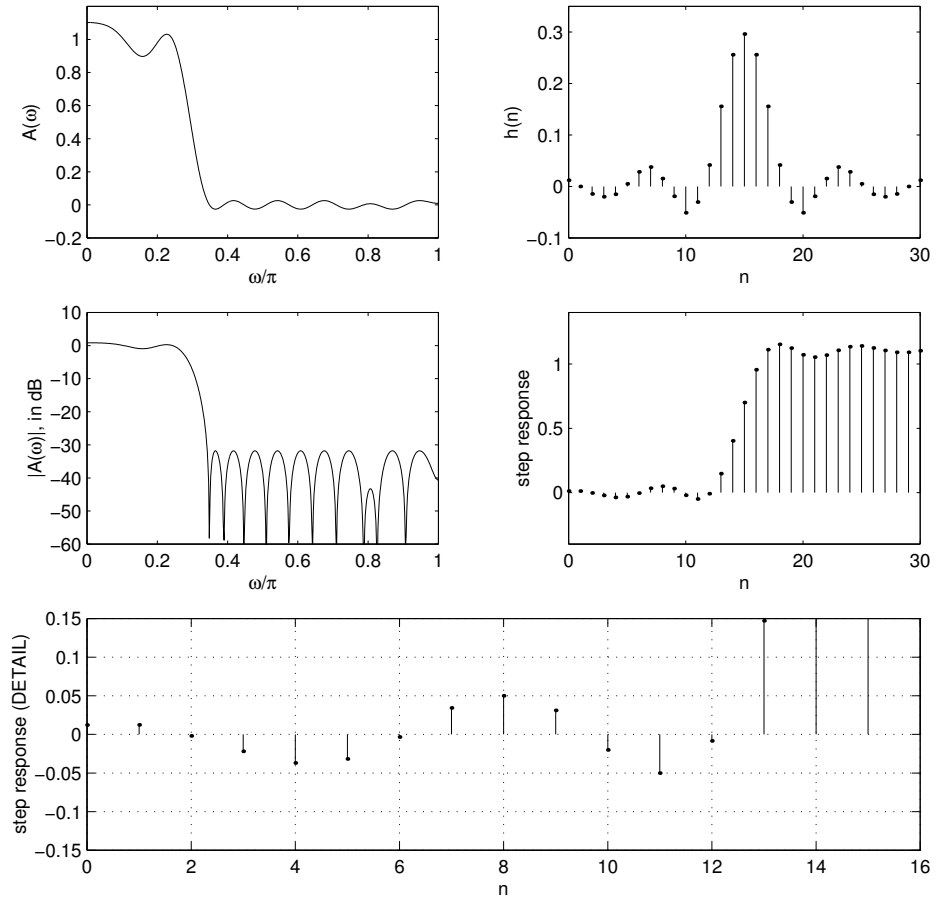
where the matrices T and v have the form

$$T = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ \vdots & & & & \end{bmatrix} \quad \text{and} \quad v = 2\,\delta_t \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \end{bmatrix}. \tag{29}$$

(Can you work out why?)

When the step response is not included in the problem formulation, the length 31 solution is shown in the previous figure. For that filter, the peak-error in the pass-band (δ) is 0.0844, but the oscillation of the step response around 0 is 0.1315. When δ_t is set equal to 0.05 and $N_1 = 12$, the solution is illustrated in following figure.

EXAMPLE



As is evident from the figures, the step response is indeed less oscillatory, however, the frequency response is not quite as good. For this filter the oscillation of the step response around 0 is decreased to 0.05, but the peak-error in the pass-band (δ) has increased to 0.1026. It is clear that the trade-off between time and frequency response specifications can be well managed with a linear program formulation. This filter was designed using the following Matlab code.

EXAMPLE

```
N = 31;
Kp = 1;
Ks = 4;
wp = 0.26*pi;
ws = 0.34*pi;
wo = (wp+ws)/2;
L = 500;
w = [0:L]*pi/L;
W = Kp*(w<=wp) + Ks*(w>=ws);
D = (w<=wo);

% remove points where W == 0
SN = 1e-8;
k = (W>SN);
w = w(k);
W = W(k);
D = D(k);

% construct matrices
M = (N-1)/2;
C = cos(w'*[0:M]);
V = 1./W';
Q = [C, -V; -C, -V];
b = [D'; -D'];
c = [zeros(M+1,1); 1];

% time-domain constraints
T = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
     0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
     0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0
     0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0
     0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0
     0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0
     0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0
     0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0
     0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0
     0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0
     0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0
     0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0
     0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0
];
v = 2*0.05*ones(13,1);

% append time-domain constraints to existing constraints
Q = [Q; -T; T];
b = [b;  v; v];

% solve linear program
```

```
x   = lp(c,Q,b);
a   = x(1:M+1);
del = x(M+2);
h   = [a(M+1:-1:2); 2*a(1); a(2:M+1)]/2;
```

PROS AND CONS

- Optimal with respect to chosen criteria.
- Easy to include arbitrary linear constraints — including *inequality* constraints.
- Criteria limited to linear programming formulation.
- Computational cost higher than Remez algorithm.

References

[1] F. M. de Saint-Martin and Pierre Siohan. Design of optimal linear-phase transmitter and receiver filters for digital systems. In *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, volume 2, pages 885–888, April 30-May 3 1995.

[2] L. R. Rabiner and B. Gold. *Theory and Application of Digital Signal Processing*. Prentice Hall, Englewood Cliffs, NJ, 1975.