Project 4 – FIR filter design

All characters and legal bodies appearing in this scenario are fictitious. Any resemblance to real persons, living or dead, or to existing corporations, is purely coincidental. (Having said that, I encourage attempts to draw parallels to existing scientific results)

1 Preamble

During the testing phase of the portable X-ray scanner (yes, that's a semester long project), you found out that the scanning beam produces high frequency noise which distorts the operation of electronic devices connected to the scanner. Some of these devices are highly sensitive to high frequency noise, while others not so much but need to work in real time. OptoCorp's chief engineer suggested that you add a digital low pass filter to your system. The filter should be easily adjustable to accommodate for the different devices. For example, there might be some device that requires very strict noise reduction, while another can cope with some noise but requires short delays between input and output.

2 More details

Digital Filtering is the processing of discrete time varying input signals to produce output signals. A **Linear Time Invariant (LTI) Filter** is a filter which satisfies both properties:

- Linearity - $h\{\alpha x_1[n] + \beta x_2[n]\} = \alpha h\{x_1[n]\} + \beta h\{x_2[n])\}$
- Time Invariance - if $y[n] = h\{x[n]\}$, then $y[n-N] = h\{x[n-N]\}$

An LTI filter is fully characterized by its **impulse response** which, by abuse of notation, we shall denote by $h[n] = h\{\delta[n]\}$. The application of $h[n]$ to an input signal $x[n]$ is the **convolution** between $x[n]$ and $h[n]$:

$$y[n] = h[n] * x[n] = \sum_{m=-\infty}^{\infty} h[m]x[n-m] = \sum_{m=-\infty}^{\infty} h[n-m]x[n] \tag{1}$$

By the **convolution theorem**, if we denote the **Discrete Time Fourier Transform (DTFT)** of $x[n]$ and $y[n]$ by $X(e^{j\omega})$ and $Y(e^{j\omega})$ respectively, then:

$$Y(e^{j\omega}) = H(e^{j\omega}) \cdot X(e^{j\omega}) \tag{2}$$

where $H(e^{j\omega})$ is the DTFT of $h[n]$, also called the **transfer function** or **frequency response** of the filter h. Note that the DTFT is a continuous function of ω, even though the signals are discrete in time! We distinguish between two types of filters:

- Finite Impulse Response (FIR) - an impulse response which has a finite number of taps. Assuming it has a duration of N and starts at $n = 0$, we can therefore represent it as a vector $\mathbf{h} = [h_0, ..., h_{N-1}]^T$.
- Infinite Impulse Response (IIR) - an impulse response which has an infinite number of taps.

The purpose of **Filter Design** is to find a realization of $h[n]$ that satisfies some predetermined design criteria. These criteria are usually given in the frequency domain as demands on the gain and phase of the system in different frequency bands. Whereas in some cases these demands are ideally realized by an IIR filter, the simplicity of the realization of FIR filters (which are always stable and causal), suggests that it may be advantageous to find an adequate approximation to the ideal filter $h_d[n]$ in terms of the coefficients of an FIR filter. Figure 1 shows an example of an ideal low-pass filter (LPF), and its approximation by an FIR filter. The frequency domain is divided into three bands:

- In the **passband** $\omega \leq \omega_p$, we allow a maximal error of $\pm\delta_p$ between the ideal and approximated filter.
- In the **stopband** $\omega \geq \omega_s$, we allow a maximal error of δ_s between the ideal and approximated filter.
- We impose no special requirement on the transition region (though we could).

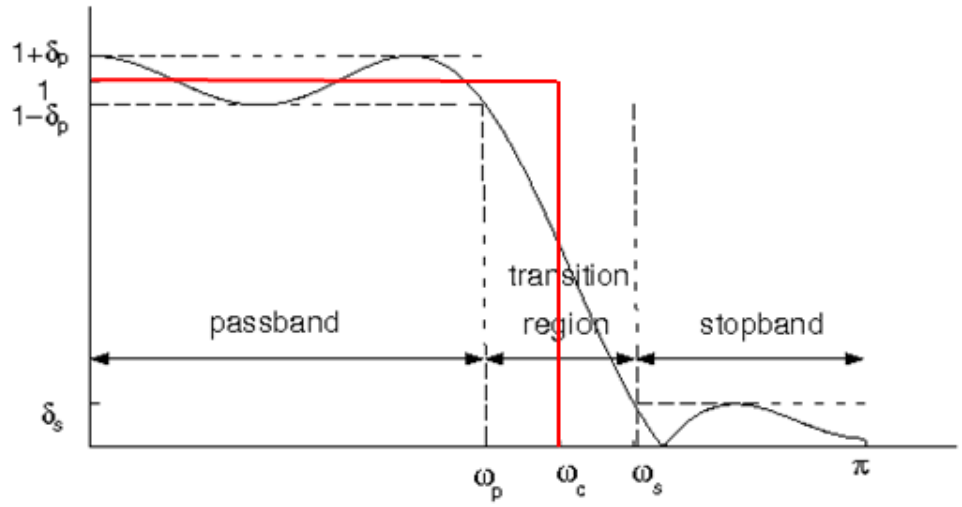

Figure 1: In red - frequency response (gain only) of the ideal LPF (which is IIR). In **black** - the approximated filter (FIR) and design considerations.

3 Problem statement

3.1 Min-max design criterion

As opposed to **inference** problems (e.g. project 1), where we want our cost function to be insensitive to measurements that are incompatible with the model (outliers), in **design** problems the situation is reversed: we need our cost function to be extremely sensitive to violation of design constraints. In other words, we want our design to work reliably under worst case scenarios. If we denote by $\varphi(\mathbf{x}, \mathbf{x_0})$ a function that measures the deviation of our variable $\mathbf{x}$ from $\mathbf{x_0}$ - the ideal solution satisfying all design constraints, we can try to minimize the maximal violation of constraints:

$$\min_{\mathbf{x}} \max \varphi(\mathbf{x}, \mathbf{x_0}) \tag{3}$$

3.2 Min-max design of linear phase filters

We shall consider designing a type I FIR filter subject to constraints on its frequency response.

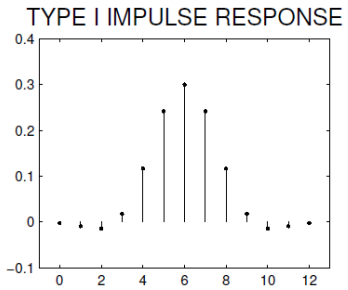

Figure 2: Type I FIR filter. By definition, if the order of the filter is $N-1$, it has N taps $[0, \ldots, N-1]$. The center of symmetry being tap number $(N-1)/2$.

Definition 1 (Type I linear phase filter)**.** *A linear phase filter has a phase response which is a linear function of the frequency : $\angle H(e^{-j\omega}) = K\omega$. A Type I linear phase filter is a symmetric filter with N odd (fig 2): $h[n] = h[N-1-n]$.*

✍ **1.** Show that the gain of a type I FIR filter of length N with real coefficients is given by

$$\begin{aligned} |H(e^{j\omega})| &= \sum_{n=0}^{M} a[n] \cdot cos(n\omega) \\ a[n] &= \begin{cases} h[M], & n = 0 \\ 2h[M-n], & n > 0 \end{cases} \\ M &= (N-1)/2 \end{aligned} \tag{4}$$

Since we are dealing only with linear phase filters with real coefficients, we shall (by abuse of terminology) refer to the gain of a filter by "frequency response" and omit the absolute value from now on. Denoting the frequency response of the ideal filter by $H_d(e^{j\omega})$ and our desired filter's frequency response by $H(e^{j\omega})$, we can weight the difference between them to get the **weighted error function**:

$$E(\omega) = W(\omega)\left(H(e^{j\omega}) - H_d(e^{j\omega})\right) \tag{5}$$

where $W(\omega)$ is a weighting function that allows the relative error of approximation between different bands to be defined. This function essentially encodes the design specifications (see figure 3).

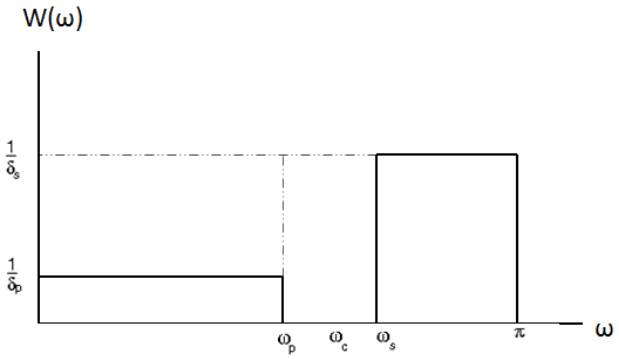

Figure 3: Weighting function. Usually $\delta_s < \delta_p$.

A filter $h[n]$ that satisfies all the design specifications, is the solution of

$$\min_{\mathbf{h}\in\mathbb{R}^{\mathbf{N}}} \max_{\omega\in[0,\pi]} E(\omega) \quad s.t \quad |E(\omega)| \leq 1 \tag{6}$$

or equivalently,

$$\min_{\mathbf{h},\delta} \delta \tag{7a}$$

$$\text{s.t}$$

$$|E(\omega)| \leq \delta \quad \forall\omega \in [0,\pi] \tag{7b}$$

$$\delta \leq 1 \tag{7c}$$

However, since the length of the filter N is given in advance, we cannot be sure that such a filter exists. For fixed N, we can instead try to minimize the maximal value of $|E(\omega)|$ by relaxing the $\delta \leq 1$ constraint, and if the result doesn't satisfy us, increase the length of the filter and try again.

$$\min_{\mathbf{h}\in\mathbb{R}^{\mathbf{N}},\delta} \delta \tag{8a}$$

$$\text{s.t}$$

$$|E(\omega)| \leq \delta \quad \forall\omega \in [0,\pi] \tag{8b}$$

Since problem (8) includes an infinite number of constraints, it is called a **semi-infinite program**. (In fact, the term **semi-infinite** comes from the fact that the constraints are only on a part of the real line : $\omega \in [0,\pi]$).

4 Min-max design of type I FIR filters with Linear Programming

4.1 Constraints in the frequency domain

One way to solve problem (8) is to discretize ω : $\omega_k = \frac{\pi}{L}(k-1) \quad k = 1\ldots L$, to get the following problem:

$$\min_{\mathbf{h},\delta} \delta \tag{9a}$$

$$\text{s.t}$$

$$|E(\omega_k)| \leq \delta \quad k = 1..L \tag{9b}$$

This is essentially a **Linear Program**, but a little work is needed to put it in canonical form:

$$\begin{aligned} \min_{x} \ & \mathbf{c}^T\mathbf{x} \\ s.t \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{10}$$

✍ **2.** Show that problem (9) can be written in canonical form as:

$$\begin{gathered} \min_{\mathbf{a}\in\mathbb{R}^{M+1},\delta} \delta \\ s.t \\ \begin{bmatrix} S\cdot C & -\mathbf{1} \\ -S\cdot C & -\mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \delta \end{bmatrix} \leq \begin{bmatrix} S\cdot\mathbf{d} \\ -S\cdot\mathbf{d} \end{bmatrix} \end{gathered} \tag{11}$$

With:

$$C = \begin{bmatrix} 1 & cos(\omega_1) & \dots & cos(M\omega_1) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & cos(\omega_k) & \dots & cos(M\omega_k) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & cos(\omega_L) & \dots & cos(M\omega_L) \end{bmatrix}$$

$$S = \begin{bmatrix} W(\omega_1) & & \\ & \ddots & \\ & & W(\omega_L) \end{bmatrix}$$

$$d = \begin{bmatrix} H_d(e^{j\omega_1}) & H_d(e^{j\omega_2}) & \dots & H_d(e^{j\omega_L}) \end{bmatrix}^T$$

$$\mathbf{a} = \begin{bmatrix} a[0] & a[1] & \dots & a[M] \end{bmatrix}^T$$

Identify and write down the terms $\mathbf{c}, A, \mathbf{b}, \mathbf{x}$ in (10).

Guidance:

- Start with splitting $|E(\omega)| \leq \delta$ into $E(\omega) \leq \delta$ and $E(\omega) \geq -\delta$.
- Plug in $E(\omega)$ from (5).
- Use (4) to represent $\{H(e^{j\omega_k})\}_{k=1}^{L}$ in terms of $\mathbf{a}$.

✍ **3.** Transform problem (11) into standard form by introducing slack variables $\mathbf{z}$, and splitting each unconstrained variable x_i into the difference of two non-negative variables $x_i^+ - x_i^-$. This would change your linear program to:

$$\begin{aligned} \min_{x^+, x^-, z} \quad & \begin{bmatrix} \mathbf{c}^T & -\mathbf{c}^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \\ \mathbf{z} \end{bmatrix} \\ s.t \quad & \begin{bmatrix} A & -A & I \end{bmatrix} \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \\ \mathbf{z} \end{bmatrix} = \mathbf{b} \\ & \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \\ \mathbf{z} \end{bmatrix} \geq \mathbf{0} \end{aligned} \tag{12}$$

✍ **4.** Write down an auxiliary problem in order to find an initial basic feasible point for problem (12).

⌨ **5.** Use Matlab's `linprog` with the `simplex` algorithm to design a low-pass filter with:

$$\omega_p = 0.26\pi,\ \omega_s = 0.34\pi,\ \delta_p = 0.1,\ \delta_s = 0.001$$

Where

$$H_d(e^{-j\omega}) = \begin{cases} 1, & \omega \leq \omega_c \\ 0, & \omega \geq \omega_c \end{cases}, \quad \omega_c = 0.3\pi$$

(there's no need to solve the auxiliary problem from previous question. Matlab takes care of the initialization automatically). Find the smallest duration N that satisfies the design constraints by increasing N and repeatedly solving the problem (you can only use odd N due to the filter being Type I). Use $L = 500$. Show that the frequency response of the filter you got indeed satisfies the constraints. Plot the impulse response.

Tip: It's enough to compute the terms in (11) for $\omega_k : W(\omega_k) \neq 0$.

4.2 Constraints in the time domain

One of the benefits of using LP to design a filter (as opposed to other methods) is that you can include (linear) constraints on the filter in the time domain. For example, one can impose constraints on the step response of $h[n]$:

$$s[n] = \sum_{k=0}^{n} h[k] \tag{13}$$

⌨ **6.** Plot the step response of the filter you designed in the previous question. Find the region $0 \leq n \leq N_1 < M$ where the step response oscillates the most around zero.

We can now write a new problem (compare to (8)):

$$\min_{\mathbf{h}\in\mathbb{R}^{\mathbf{N}},\delta} \ \delta \tag{14a}$$

$$\text{s.t}$$

$$|E(\omega)| \leq \delta \quad \forall\omega \in [0,\pi] \tag{14b}$$

$$|s[n]| \leq \delta_t \quad 0 \leq n \leq N_1 \tag{14c}$$

✍ **7.** Show that after discretization of ω and rearranging the terms, you get the following linear program:

$$\min_{\mathbf{a}\in\mathbb{R}^{M+1},\delta} \delta$$

$$s.t$$

$$\begin{bmatrix} S\cdot C & -\mathbf{1} \\ -S\cdot C & -\mathbf{1} \\ T & \mathbf{0} \\ -T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \delta \end{bmatrix} \leq \begin{bmatrix} S\cdot \mathbf{d} \\ -S\cdot \mathbf{d} \\ \mathbf{v} \\ \mathbf{v} \end{bmatrix} \tag{15}$$

With

$$\mathbf{v} = 2\delta_t \cdot \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T$$

$$T = \begin{bmatrix} 0 & \dots & 0 & 0 & 1 \\ 0 & \dots & 0 & 1 & 1 \\ 0 & \dots & 1 & 1 & 1 \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

(You need to figure out how exactly T is constructed)

⌨ **8.** Use Matlab's `linprog` with the `simplex` algorithm to design a low-pass filter of duration N (same duration you found in question 5) with the same constraints as before on its frequency response, and additional constraints on its step response:

$$\delta_t = 0.05, \quad N_1 = \text{ what you found in previous question.}$$

Plot the impulse response, step response and frequency response of the filter you got. Compare the results with the ones obtained in question 5. Specifically:

- Gain in the pass and stop bands.
- Oscillations of the step response between $0 \leq n \leq N_1$.

If the new filter violates the demands on the frequency response, decrease N_1 to the maximal value for which the demands on the frequency response are not compromised.

5 Worst case complexity of the simplex method

In the simplex method, we travel between basic feasible points (which are vertices of the feasible polytope) until the optimal solution is found. For many years, researchers wondered whether the simplex method is a polynomial time algorithm, or there exist some example

that would require visiting all the vertices of the feasible polytope before the optimal point is found. In 1972 Victor Klee and George J. Minty found the **klee-minty cube**[1] - a unit cube whose corners have been slightly perturbed, which demonstrates that Dantzig's simplex algorithm has an exponential worst case complexity.

A Klee-Minty example:

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} 10^{n-j} x_j \\ s.t \quad & 2\sum_{j=1}^{i-1} 10^{i-j} x_j + x_i \leq 100^{i-1} \quad for \quad 1 \leq i \leq n \\ & x_j \geq 0 \quad \forall j \end{aligned} \tag{16}$$

Using the most negative entry pivoting rule, it requires $2^n - 1$ pivots to find an optimal solution (see `klee-minty.pdf`).

Even though there are some pathological examples like (16) that exhibit poor convergence, in practice the simplex algorithm converges very fast. In fact, this dissonance between the worst case complexity and what happens in practice has led to the introduction of **average-case complexity** as a more accurate measure for the complexity of algorithms. More recently, researchers were able to explain the observed fast convergence of the simplex algorithm using a new model of complexity called **smoothed complexity** [2].

6 Introduction to Interior Point Methods

The fact that there are some examples where the simplex algorithm converges poorly has prompted researchers to look for worst case polynomial time algorithms for linear programming. The issue was resolved by Leonid Khachiyan who presented the **ellipsoid algorithm**[3] which had worst case polynomial complexity. However, this algorithm tended to approach the worst case complexity on nearly all inputs, and so the simplex algorithm remained dominant in practice. In 1984, Narendra Karmarkar came up with the first polynomial time algorithm for linear programming that had the promising potential to compete with the simplex algorithm. Though eventually Karmarkar's **projection algorithm**[4] didn't live up to its name, it ignited the research on **interior point methods**, a special set of algorithms that approach the solution to the linear program (and other convex programs) from the *interior* of the feasible region, (as opposed to the simplex algorithm who approaches the solution by traveling along its boundary). Interior point methods have revolutionized the field of optimization during the 80's and 90's and allowed the solution (in polynomial time) of many medium scale convex optimization problems. The basic idea however dates back to the early 60's works of Anthony V. Fiacco, Garth P. McCormick and others, who proposed to use a **barrier function** to encode the information of the feasible set. We shall explore

the application of the simplest Barrier Method combined with Newton's method for solving the canonical form LP (10).

6.1 Primal Barrier Method

We shall use a logarithmic barrier function to encode the constraint set of the inequality form LP :

$$\begin{aligned} \min_x \ & \mathbf{c}^T\mathbf{x} \\ s.t \quad & A\mathbf{x} \leq \mathbf{b} \end{aligned} \tag{17}$$

where we have included the non-negativity constraints on $\mathbf{x}$ (if there are any) inside $A\mathbf{x} \leq \mathbf{b}$

✍ **9.** Compute the gradient and Hessian of the log-barrier function $\varphi(p) = -\log(-p)$ applied to the constraints $g_i(\mathbf{x}) = (\mathbf{a}_i^T\mathbf{x} - b_i) \leq 0$ $\left(i.e.,\ \phi(\mathbf{x}) = \sum_{i=1}^{m} -\log(b_i - \mathbf{a}_i^T\mathbf{x}) \right)$, where $\mathbf{a}_i$ is a column vector comprised of the elements of the i^{th} row of A.

✍ **10.** Write down the Lagrangian for problem (17). Show that

$$\lambda_i^\star(t) \equiv -\frac{1}{t g_i(\mathbf{x}^*(t))} \quad i = 1 \dots m$$

are dual feasible:

- $\boldsymbol{\lambda}^\star(t) \geq 0$
- $\nabla_x L(\mathbf{x}^*(t), \boldsymbol{\lambda}^\star(t)) = 0$

where $\mathbf{x}^*(t)$ is the solution to the *centering problem*:

$$\mathbf{x}^*(t) = \operatorname*{argmin}_x \ f_t(\mathbf{x}) \equiv t f(\mathbf{x}) + \phi(\mathbf{x})$$

✍ **11.** Show that

$$h(\boldsymbol{\lambda}^\star(t)) = f(\mathbf{x}^*(t)) - \frac{m}{t}$$

Where $h(\boldsymbol{\lambda}^\star(t))$ is the dual objective of (17) evaluated at $\boldsymbol{\lambda}^\star(t) = [\lambda_1^\star(t), \dots, \lambda_m^\star(t)]^T$.

Therefor, from weak duality ($h \leq f^*$) we get that $f(\mathbf{x}^*(t))$ is $\frac{m}{t}$ sub-optimal:

$$f(\mathbf{x}^*(t)) - f^* \leq \frac{m}{t}$$

input : function f_t, initial strictly feasible point $\mathbf{x}_0$,$t_0 > 0$, $\mu > 1$,ϵ
output: (approximate) minimizer $\mathbf{x}^*$ of f
Start with a strictly feasible point $\mathbf{x}_0$ (*i.e* $A\mathbf{x}_0 < \mathbf{b}$)
for $k = 1, 2, \ldots,$ *until* $(m/t < \epsilon)$ **do**
  Solve *centering problem* $\mathbf{x}^*(t_k) = \underset{x}{\operatorname{argmin}}\ t_k f(\mathbf{x}) + \phi(\mathbf{x})$ starting at $\mathbf{x}^*(t_{k-1})$
  Update $t_{k+1} = \mu t_k$
end
Return $\mathbf{x}^\star = \mathbf{x}^*(t_k)$

Algorithm 1: Barrier Method

⌨ **12.** Implement the barrier method for minimizing the inequality form LP (17).

- For initialization you need a **strictly** feasible point (see next question).
- Use Newton's method with inexact line search (Armijo rule) for the minimization of $f_t(\mathbf{x})$. Use $\sigma = 0.01,\ \beta = 0.5,\ \alpha_0 = 1$ (remember that in Newton's method, when you are close enough to the minimum you should have $\alpha = 1$ to get the quadratic rate of convergence).
- Don't set t_0 too large\small. try $t_0 \in [100, 1000]$.

⌨ **13.** Solve the above FIR design problems with your interior-point solver. For the initialization, solve a Phase I problem ("auxiliary problem") with the solver you built:

$$\begin{aligned} &\min_{x,\gamma} \quad \gamma \\ s.t \quad &A\mathbf{x} - \mathbf{b} \leq \gamma \cdot \mathbf{1} \end{aligned} \tag{18}$$

The Phase I problem also needs to be initialized with a strictly feasible point, but this is easy. Find a suitable initialization for (18) and report it. How can you get an initial strictly feasible point for Phase II (the original problem) by solving the Phase I problem? Actually, you don't need to solve the Phase I problem exactly, as long as you end up with a strictly feasible point. Explain how you would stop the solver beforehand.

Summary : the above algorithm works pretty well in practice. However it depends on multiple parameters that have to be set by hand. The theory of **self-concordant functions** allows us to construct interior point methods which are parameter free. These are some of the best methods available in the market today for the solution of linear programs and a bunch of other convex optimization problems.

7 Encore - FIR filter Design : Can we do better?

LP offered us great versatility in designing our filter, but it didn't come without limits:

- The original problem (8) was **semi-infinite**. In order to solve it, we had to discretize the frequency range, so we only solved an approximate problem.
- The LP formulation above only allowed us to design **linear phase** filters (= symmetric) with real coefficients. A **non-linear phase** filter might get the same approximation error with shorter duration (thus, introducing less delay). Though linear phase is usually a desirable property, we don't really need linear phase in the stopband.

These limitations can be overcome by modeling the FIR filter design problem as a **semidefinite program (SDP)** (see [5]).

8 Tips

- You may find the following Matlab commands useful (for comparison and display): `linprog, optimset, optimoptions, stem, cumsum`

References

[1] Klee, Victor; Minty, George J. (1972). *How Good is the Simplex Algorithm?* In O. Shisha, editor, Inequalities, III, pages 159-175. Academic Press, New York, NY, 1972.

[2] Spielman, Daniel A., and Shang-Hua Teng. *Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time.* Journal of the ACM (JACM) 51.3 (2004): 385-463.

[3] L. G. Khachian (1979) Doklady Akademi Nauk S.S.S.R. (Vol 244 pp 1093–1096)

[4] Narendra Karmarkar (1984). *"A New Polynomial Time Algorithm for Linear Programming"*, Combinatorica, Vol 4, nr. 4, p. 373395.

[5] Masaaki Nagahara (2011). *Min-Max Design of FIR Digital Filters by Semidefinite Programming, Applications of Digital Signal Processing*, Dr. Christian Cuadrado-Laborde (Ed.), ISBN: 978-953-307-406-1, InTech, DOI: 10.5772/25413. Available from here.