Ellipsoid Method and the Amazing Oracles

May 30, 2019

Abstract

Ellipsoid method is revisited. Besides that, three separation oracles are investigated for applications. They are robust optimization, semidefinite programming, and network optimization. Discuss the stability issue. Finally, the parallel cut is described.

1 Introduction

The reputation of the ellipsoid method is bad. And it is unfair. It is commonly believed that the method is slow for large-scale convex problems [**???**]. However, the ellipsoid method works very differently compared with the popular interior point method.

Firstly, the ellipsoid method does not require evaluation of all the constraint functions explicitly at each iteration. All it needs is a *separation oracle* that provides a *cutting-plane*. This can make the method attractive for certain problems in which the number of constraints is large or infinite.

Secondly, although the ellipsoid method cannot exploit the sparsity of the problem, the separation oracle can exploit certain types of structure in large and complex problems.

It desires to be known.

In Section 3.2, problems involving matrix inequalities are discussed. Recall that the checking of the positive definiteness of a symmetry matrix can be done efficiently using Cholesky or more precisely LDL^T factorization*. Moreover, if the matrix is not positive definite, a vector v can be found as a cutting plane during the factorization process. Thus, it can be used for efficient oracle implementation.

In Section 3.3, we show the network parametric problem, where the cut can be obtained by finding the negative cycle of a directed graph. Efficient algorithms exists, in which the locality of network and the associativity of data are exploited.

The implementation of the ellipsoid method is described in Section 4. Where the search space is an ellispoid, usually specfied as:

$$\{x \mid (x - x_c)P^{-1}(x - x_c) \leq 1\},$$

where x_c is the center of the ellipsoid, and P is a symmetric positive matrix. At each iteration, x_c and P are updated according to the cut. Although the updating of the ellipsoid is simple and the implementation has been found for decades, we show that one can reduce n^2 flops (floating point operations) by splitting P into α times Q.

Furthermore, the use of parallel cuts is discussed in Section 4.2. When a pair of parallel inequalities, one of which is violated, it is possible to use both constraints simultaneously to update the ellipsoid. Some papers reported that this technique did not provided signifancant improvement. We show that, for some cases when the upper and lower bounds of the constraints are tight, such as those for the filter designs, the use of parallel cuts could in fact improve the run time dramatically.

In many practice engineering problems, some design variables may be restricted in discrete forms. Since all the cutting-plane method request is separation oracle, it works for discretized problems also.

2 Cutting-plane Method Revisited

2.1 Convex Feasibility Problem

Let $\mathcal{K} \subseteq \mathbb{R}^n$ be a convex set. Consider the feasibility problem:

1. Find a point $x^* \in \mathbb{R}^n$ in $\mathcal{K}$, or
2. Determine that $\mathcal{K}$ is empty (i.e., no feasible solution)

When a *separation oracle* Ω is *queried* at x_0, it either

1. Asserts that $x_0 \in \mathcal{K}$, or
2. Returns a separating hyperplane between x_0 and $\mathcal{K}$:

$$g^\mathsf{T}(x - x_0) + \beta \leq 0, \beta \geq 0, g \neq 0, \ \forall x \in \mathcal{K}. \tag{1}$$

The pair (g, β) is called a *cutting-plane*, since it eliminates the halfspace $\{x \mid g^\mathsf{T}(x - x_0) + \beta > 0\}$ from our search. If $\beta = 0$ (x_0 is on the boundary of halfspace that is cut), cutting-plane is called *neutral cut*. If $\beta > 0$ (x_0 lies in the interior of halfspace that is cut), cutting-plane is called *deep cut*. If $\beta < 0$ (x_0 lies in the exterior of halfspace that is cut), cutting-plane is called *shadow cut*.

The $\mathcal{K}$ is usually given by a set of inequalities $f_j(x) \leq 0$ or $f_j(x) < 0$ for $j = 1 \cdots m$, where $f_j(x)$ is a convex function. A vector $g \equiv \partial f(x_0)$ is called a

subgradient of a convex function f at x_0 if $f(z) \geq f(x_0) + g^{\mathrm{T}}(z - x_0)$. Hence, the cut (g, β) is given by $(\partial f(x_0), f(x_0))$.

Note that if $f(x)$ is differentiable, we can simply take $\partial f(x_0) = \nabla f(x_0)$. Cutting-plane method consists of two key components: separation oracle Ω and a search space $\mathcal{S}$ initially big enough to cover $\mathcal{K}$. For example,

- Polyhedron $\mathcal{P} = \{z \mid Cz \preceq d\}$.
- Interval $\mathcal{I} = [l, u]$ (for one-dimensional problem).
- Ellipsoid $\mathcal{E} = \{z \mid (z - x_c)P^{-1}(z - x_c) \leq 1\}$.

Generic Cutting-plane method:

- **Given** initial $\mathcal{S}$ known to contain $\mathcal{K}$.
- **Repeat**
  1. Choose a point x_0 in $\mathcal{S}$.
  2. Query the separation oracle at x_0.
  3. **If** $x_0 \in \mathcal{K}$, quit.
  4. **Else**, update $\mathcal{S}$ to a smaller set that covers:
  $$\mathcal{S}^+ = \mathcal{S} \cap \{z \mid g^{\mathsf{T}}(z - x_0) + \beta \leq 0\}.$$
  5. **If** $\mathcal{S}^+ = \emptyset$ or it is small enough, quit.

Todo: What if the seach space is large enough?

2.2 From Feasibility to Optimization

Consider:

$$\begin{array}{ll} \text{minimize} & f_0(x), \\ \text{subject to} & x \in \mathcal{K}. \end{array} \tag{2}$$

The optimization problem is treated as a feasibility problem with an additional constraint $f_0(x) < t$. Here, $f_0(x)$ could be a convex function or a quasiconvex function. t is the best-so-far value of $f_0(x)$. The problem can be reformulated as:

$$\begin{array}{ll} \text{minimize} & t, \\ \text{subject to} & \Phi(x, t) < 0, \\ & x \in \mathcal{K}, \end{array} \tag{3}$$

where $\Phi(x, t) < 0$ is the t-sublevel set of $f_0(x)$.

For each x, $\Phi(x, t)$ is a nonincreasing function of t, *i.e.*, $\Phi(x, t') \leq \Phi(x, t) whenever t' \geq t$. Note that $\mathcal{K}_t \subseteq \mathcal{K}_u$ if and only if $t \leq u$ (monotonicity). One easy way to solve the optimization problem is to apply the binary search on t.

Another possible way is, to update the best-so-far t whenever a feasible solution x_0 is found such that $\Phi(x_0, t) = 0$. We assume that the oracle takes responsibility for that.

Generic Cutting-plane method (Optim)

- **Given** initial $\mathcal{S}$ known to contain $\mathcal{K}_t$.
- **Repeat**
  1. Choose a point x_0 in $\mathcal{S}$
  2. Query the separation oracle at x_0
  3. **If** $x_0 \in \mathcal{K}_t$, update t such that $\Phi(x_0, t) = 0$.
  4. Update $\mathcal{S}$ to a smaller set that covers:

$$\mathcal{S}^+ = \mathcal{S} \cap \{z \mid g^\mathsf{T}(z - x_0) + \beta \leq 0\}$$

  5. **If** $\mathcal{S}^+ = \emptyset$ or it is small enough, quit.

2.3 Example: Profit Maximization

This example is taken from [1]. Consider the following short-run profit maximization problem:

$$\begin{array}{ll} \text{maximize} & p(Ax_1^\alpha x_2^\beta) - v_1 x_1 - v_2 x_2, \\ \text{subject to} & x_1 \leq k, \\ & x_1, x_2 positive, \end{array} \tag{4}$$

where p is the market price per unit, A is the scale of production, $\alpha and \beta$ are the output elasticities, x_i and v_i are ith input quantity and output price, $Ax_1^\alpha x_2^\beta)$ is the Cobb-Douglas production function, and k is a given constant that restricts the quantity of x_1.

The above formulation is not in a convex form. Firstly, we rewrite the problem as follows:

$$\begin{array}{ll} \text{maximize} & t, \\ \text{subject to} & t + v_1 x_1 + v_2 x_2 \leq pAx_1^\alpha x_2^\beta, \\ & x_1 \leq k, \\ & x_1, x_2 positive. \end{array}$$

By the change of variables, we have the following convex form of **???**:

$$\begin{array}{ll} \text{maximize} & t, \\ \text{subject to} & \log(t + v_1 e^{y_1} + v_2 e^{y_2}) - (\alpha y_1 + \beta y_2) \leq \log(p\,A), \\ & y_1 \leq \log k, \end{array} \tag{5}$$

where $y_1 = \log x_1$ and $y_2 = \log x_2$.

Some readers may recognize that the problem can also be written in a geometric program by introducing one additional variable [1].

3 Area of Applications

- Robust convex optimization
  - oracle technique: affine arithmetic
- Parametric network potential problem
  - oracle technique: negative cycle detection
- Semidefinite programming
  - oracle technique: Cholesky factorization

3.1 Robust Convex Optimization

Consider:

$$\begin{array}{ll} \text{minimize} & \sup_{q\in\mathcal{Q}} f_0(x,q), \\ \text{subject to} & f_j(x,q)\le 0,\ \forall q\in\mathcal{Q},\ j=1,2,\cdots,m, \end{array} \tag{6}$$

where q represents a set of varying parameters. The problem can be reformulated as:

$$\begin{array}{ll} \text{minimize} & t, \\ \text{subject to} & f_0(x,q)<t, \\ & f_j(x,q)\le 0,\ \forall q\in\mathcal{Q},\ j=1,2,\cdots,m. \end{array}$$

3.1.1 Algorithm

The oracle only needs to determine:

- If $f_j(x_0,q)>0$ for some j and $q=q_0$, then
- the cut $(g,\beta)=(\partial f_j(x_0,q_0), f_j(x_0,q_0))$
- If $f_0(x_0,q)\ge t$ for some $q=q_0$, then
- the cut $(g,\beta)=(\partial f_0(x_0,q_0), f_0(x_0,q_0)-t)$
- Otherwise, x_0 is feasible, then
- Let $q_{\max}=\text{argmax}_{q\in\mathcal{Q}} f_0(x_0,q)$.
- $t:=f_0(x_0,q_{\max})$.
- The cut $(g,\beta)=(\partial f_0(x_0,q_{\max}),0)$

Random sampling trick.

3.1.2 Example: Robust Profit Maximization

Consider again the profit maximization problem in § 2.3. Now suppose the parameters $\{\alpha, \beta, p, v_1, v_2, k\}$ are subject to interval uncertainties:

$$\begin{array}{rcl} \alpha - \varepsilon_1 \leq & \hat{\alpha} & \leq \alpha + \varepsilon_1 \\ \beta - \varepsilon_2 \leq & \hat{\beta} & \leq \beta + \varepsilon_2 \\ p - \varepsilon_3 \leq & \hat{p} & \leq p + \varepsilon_3 \\ v_1 - \varepsilon_4 \leq & \hat{v}_1 & \leq v_1 + \varepsilon_4 \\ v_2 - \varepsilon_5 \leq & \hat{v}_2 & \leq v_2 + \varepsilon_5 \\ k - \varepsilon_6 \leq & \hat{k} & \leq k + \varepsilon_6 \end{array}$$

The robust counterpart considering the worst-case scenario is given by:

$$\begin{array}{ll} \max & t \\ \text{s.t.} & \log(t + \hat{v}_1 e^{y_1} + \hat{v}_2 e^{y_2}) - (\hat{\alpha} y_1 + \hat{\beta} y_2) \leq \log(\hat{p}\, A) \\ & y_1 \leq \log \hat{k}. \end{array}$$

In [1], the aurthors proposed a *piecewise linear approximation* approach. It involves a lot of programming effort but the result is inaccurate. However, this can easily be done using the cutting-plane method. Note that in this simple example, the worst case happens when:

- $\hat{p} = p + e_3$, $k = \bar{k} + e_3$
- $v_1 = \bar{v}_1 - e_3$, $v_2 = \bar{v}_2 - e_3$,
- if $y_1 > 0$, $\alpha = \bar{\alpha} - e_1$, else $\alpha = \bar{\alpha} + e_1$
- if $y_2 > 0$, $\beta = \bar{\beta} - e_2$, else $\beta = \bar{\beta} + e_2$

We can even reuse the original oracle to compose the robust counterpart.

```
class profit_rb_oracle:
    def __init__(self, params, a, v, vparams):
        p, A, k = params
        e1, e2, e3, e4, e5 = vparams
        params_rb = p - e3, A, k - e4
        self.a = a
        self.e = [e1, e2]
        self.P = profit_oracle(params_rb, a, v + e5)

    def __call__(self, y, t):
        a_rb = self.a.copy()
        for i in [0, 1]:
            if y[i] <= 0:
                a_rb[i] += self.e[i]
            else:
                a_rb[i] -= self.e[i]
```

```
        self.P.a = a_rb
        return self.P(y, t)
```

Note that the 'argmax' in general can be non-convex and hence difficult to solved. For more complicated problems, affine arithmetic could be used [2].

3.2 Multi-parameter Network Problems

Given a network represented by a directed graph $G = (V, E)$. Consider :

$$\begin{array}{ll} \text{minimize} & t, \\ \text{subject to} & u_i - u_j \leq h_{ij}(x,t),\ \forall (i,j) \in E, \\ \text{variables} & x, u, \end{array}$$

where $h_{ij}(x,t)$ is the weight function of edge (i,j).

Assume that the network is large but the number of parameters is small. Given x and t, the problem has a feasible solution if and only if G contains no negative cycle. Let $\mathcal{C}$ be a set of all cycles of G. The problem can be formulated as:

$$\begin{array}{ll} \text{minimize} & t, \\ \text{subject to} & W_k(x,t) \geq 0, \forall C_k \in C, \\ \text{variables} & x, \end{array}$$

where C_k is a cycle of G:

$$W_k(x,t) = \sum_{(i,j)\in C_k} h_{ij}(x,t).$$

3.2.1 Negative Cycle Detection Algorithm

The negative cycle detection is the most time-consuming part of the proposed method, so it is very important to choose the proper negative cycle detection algorithm. There are lots of methods to detect negative cycles in a weighted graph [3], in which Tarjan's algorithm [4] is one of the fastest algorithms in practice [3,5].

The separation oracle only needs to determine:

- If there exists a negative cycle C_k under x_0, then
- the cut $(g, \beta) = (-\partial W_k(x_0), -W_k(x_0))$
- If $f_0(x_0) \geq t$, then the cut $(g, \beta) = (\partial f_0(x_0), f_0(x_0) - t)$.
- Otherwise, x_0 is feasible, then
  - $t := f_0(x_0)$.
  - The cut $(g, \beta) = (\partial f_0(x_0), 0)$

3.2.2 Example: Optimal Matrix Scaling

This example is taken from [6]. Given a matrix $A = [a_{ij}] \in \mathbb{R}^{N \times N}$. A *symmetric scaling* of A is a matrix B of the form UAU^{-1} where U is a nonnegative diagonal matrix having the same dimension. Under the *min-max criterion*, the objective is to minimize the largest absolute value of the element of B. The symmetric scaling problem under such a criterion can be reformulated as a single-parameter monotropic linear network optimization problem.

Another possible criterion is to minimize the ratio of largest absolute value of the element B to the smallest. The motivation for using this criterion is that high ratios cause difficulties in performing the simplex method. Under this *min-max-ratio* criterion, the symmetric scaling problem can be formulated as:

$$\begin{array}{ll} \text{minimize} & \pi/\psi \\ \text{subject to} & \psi \leq u_i |a_{ij}| u_j^{-1} \leq \pi, \ \forall a_{ij} \neq 0, \\ & \pi, \ \psi, u\text{positive} \\ \text{variables} & \pi, \psi, u \, . \end{array}$$

Let k' denotes $\log(|k|)$. By taking the logarithms of variables, the above programming can be transformed into a two-parameter network optimization problem:

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \pi' - \psi' \leq t \\ & u'_i - u'_j \leq \pi' - a'_{ij}, \ \forall a_{ij} \neq 0 \, , \\ & u'_j - u'_i \leq a'_{ij} - \psi', \ \forall a_{ij} \neq 0 \, , \\ \text{variables} & \pi', \psi', u' \, . \end{array}$$

where $x = (\pi', \psi')^{\mathsf{T}}$.

The authors of [6] claimed that efficient algorithms had been developed for solving such multi-parameter problem, but we cannot find any follow-up publications about this.

Interestingly, by using the cutting-plane method, one can easily extend the single-parameter network algorithm to the multi-parameter one.

In this application, $h_{ij}(x)$ is:

$$h_{ij}(x) = \begin{cases} -\pi' + a'_{ij}, & \forall a_{ij} \neq 0 \, , \\ \psi' - a'_{ji}, & \forall a_{ji} \neq 0 \, , \end{cases}$$

Fast algorithms for finding negative cycle can be found in [7,8]. More application in clock skew scheduling can be found in [9].

3.3 Problems Involving Matrix Inequalities

Consider the following problem:

$$\begin{array}{ll} \text{minimize} & t, \\ \text{subject to} & F(x,t) \succeq 0, \end{array}$$

where $F(x,t)$ is a matrix-valued function, $A \succeq 0$ denotes A is positive semidefinite. Recall that a matrix A is positive semidefinite if and only if $v^{\mathsf{T}} A v \geq 0$ for all $v \in \mathbb{R}^N$. The problem can be transformed into:

$$\begin{array}{ll} \text{minimize} & t, \\ \text{subject to} & v^{\mathsf{T}} F(x,t) v \geq 0, \ \forall v \in \mathbb{R}^N. \end{array}$$

Consider $v^{\mathsf{T}} F(x,t) v$ is concave for all $v \in \mathbb{R}^N$ w.r.t. x, then the above problem is a convex programming. Reduce to *semidefinite programming* if $F(x,t)$ is linear w.r.t. x, i.e., $F(x) = F_0 + x_1 F_1 + \cdots + x_n F_n$.

3.3.1 Cholesky Factorization Algorithm

An alternative form, eliminating the need to take square roots, is the symmetric indefinite factorization:

$$\begin{aligned} \mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}} &= \begin{pmatrix} 1 & 0 & 0 \\ L_{21} & 1 & 0 \\ L_{31} & L_{32} & 1 \end{pmatrix} \begin{pmatrix} D_1 & 0 & 0 \\ 0 & D_2 & 0 \\ 0 & 0 & D_3 \end{pmatrix} \begin{pmatrix} 1 & L_{21} & L_{31} \\ 0 & 1 & L_{32} \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} D_1 & & \text{(symmetric)} \\ L_{21}D_1 & L_{21}^2 D_1 + D_2 & \\ L_{31}D_1 & L_{31}L_{21}D_1 + L_{32}D_2 & L_{31}^2 D_1 + L_{32}^2 D_2 + D_3. \end{pmatrix}. \end{aligned}$$

If A is real, the following recursive relations apply for the entries of D and L:

$$D_j = A_{jj} - \sum_{k=1}^{j-1} L_{jk} L_{jk}^* D_k,$$

$$L_{ij} = \frac{1}{D_j} \left(A_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk}^* D_k \right) \quad \text{for } i > j.$$

Again, the pattern of access allows the entire computation to be performed in-place if desired.

The following is the algorithm written in Python:

```
def factor(self, getA):
    T = self.T
    for i in range(self.n):  # from 0 to n-1
        for j in range(i+1): # from 0 to i
            d = getA(i, j) - np.dot(T[:j, i], T[j, :j])
            T[i, j] = d
            if i != j:
                T[j, i] = d / T[j, j]
        if d <= 0.:  # strictly positive
            self.p = i
            return
    self.p = self.n
```

The vector v can be found. The following is the algorithm written in Python:

```
def witness(self):
    p = self.p
    n = p + 1
    v = np.zeros(n)
    v[p] = 1
    for i in range(p, 0, -1): # backward substitution
        v[i-1] = -np.dot(self.T[i-1, i:n], v[i:n])
    return v, -self.T[p, p]
```

The oracle only needs to:

- Perform a *row-based* Cholesky factorization such that $F(x_0, t) = R^\mathsf{T} R$.
- Let $A_{:p,:p}$ denotes a submatrix $A(1:p, 1:p) \in \mathbb{R}^{p \times p}$.
- If Cholesky factorization fails at row p,
  - there exists a vector $e_p = (0, 0, \cdots, 0, 1)^\mathsf{T} \in \mathbb{R}^p$, such that
    - $v = R^{-1}_{:p,:p} e_p$, and
    - $v^\mathsf{T} F_{:p,:p}(x_0) v < 0$.
  - The cut $(g, \beta) = (-v^\mathsf{T} \partial F_{:p,:p}(x_0) v, -v^\mathsf{T} F_{:p,:p}(x_0) v)$

3.3.2 Example: Matrix Norm Minimization

Let $A(x) = A_0 + x_1 A_1 + \cdots + x_n A_n$. Problem $\min_x \|A(x)\|$ can be reformulated as

$$\begin{array}{ll} \text{minimize} & t, \\ \text{subject to} & \begin{pmatrix} t\,I & A(x) \\ A^\mathsf{T}(x) & t\,I \end{pmatrix} \succeq 0. \end{array}$$

Binary search on t can be used for this problem.

3.3.3 Example: Estimation of Correlation Function

3.4 Random Field [10]

Random field, also known as *stochastic process*, can be regarded as an indexed family of random variables denoted as $\{Z(\mathbf{s}) : \mathbf{s} \in D\}$, where D is a subset of d-dimensional Euclidean space $\mathbb{R}^d$. To specify a stochastic process, the joint probability distribution function of any finite subset $(Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_n))$ must be given in a consistent way, which is called *distribution* of the process. For ease of analysis, a random field is often assumed to be with *Gaussian* distribution, and is called Gaussian random field.

A random field has several key properties that are useful in practical problems. The field is *stationary* under translations, or *homogeneous*, if the distribution is unchanged when the point set is translated. The field is *isotropic* if the distribution is invariant under any rotation of the whole points in the parameter space. We study the homogeneous isotropic field in this paper.

The *covariance* C and *correlation* R of a stochastic process are defined by

$$C(\mathbf{s}_i, \mathbf{s}_j) = \mathrm{cov}(Z(\mathbf{s}_i), Z(\mathbf{s}_j)) = \mathrm{E}[(Z(\mathbf{s}_i) - \mathrm{E}[Z(\mathbf{s}_i)])(Z(\mathbf{s}_j) - \mathrm{E}[Z(\mathbf{s}_j)])]$$

and

$$R(\mathbf{s}_i, \mathbf{s}_j) = C(\mathbf{s}_i, \mathbf{s}_j) / \sqrt{C(\mathbf{s}_i, \mathbf{s}_i) C(\mathbf{s}_j, \mathbf{s}_j)}$$

respectively for all $\mathbf{s}_i, \mathbf{s}_j \in D$, where $\mathrm{E}[Z(\mathbf{s})]$ denotes the expectation of $Z(\mathbf{s})$. Thus a process is homogeneous if C and R depend only on the separation vector $\mathbf{h} = \mathbf{s}_i - \mathbf{s}_j$. Furthermore, it is isotropic if C and R depend upon $\mathbf{h}$ only through its length h, i.e.,

$$C(\mathbf{s}_i, \mathbf{s}_j) = C(\mathbf{h}) = C(h),$$

$$R(\mathbf{s}_i, \mathbf{s}_j) = R(\mathbf{h}) = R(h) = C(h)/C(0). \tag{7}$$

If we denote $C(0)$, the variance of $Z(\mathbf{s})$, as σ^2, then the relationship between covariance and correlation is $C(h) = \sigma^2 R(h)$.

When the two components are considered, the measurement data can still be regarded as a Gaussian random field, but the correlation function will have a discontinuity at the origin. This phenomenon is called “nugget effect" [11].

$$\begin{array}{ll} \min_{\kappa,p} & \|\Omega(p) + \kappa I - Y\| \\ \text{s. t.} & \Omega(p) \succcurlyeq 0, \kappa \geq 0 \ . \end{array}$$

Let $\rho(h) = \sum_i^n p_i \Psi_i(h)$, where p_i's are the unknown coefficients to be fitted Ψ_i's are a family of basis functions. The covariance matrix $\Omega(p)$ can be recast as:

$$\Omega(p) = p_1 F_1 + \cdots + p_n F_n,$$

where $\{F_k\}_{i,j} = \Psi_k(\|s_j - s_i\|_2)$.

4 Ellipsoid Method Revisited

Some History of Ellipsoid Method [12]. Introduced by Shor and Yudin and Nemirovskii in 1976. Used to show that linear programming (LP) is polynomial-time solvable (Kachiyan 1979), settled the long-standing problem of determining the theoretical complexity of LP. In practice, however, the simplex method runs much faster than the method, although its worst-case complexity is exponential.

4.1 Basic Ellipsoid Method

An ellipsoid $\mathcal{E}(x_c, P)$ is specified as a set

$$\{x \mid (x - x_c)P^{-1}(x - x_c) \leq 1\},$$

where x_c is the center of the ellipsoid.

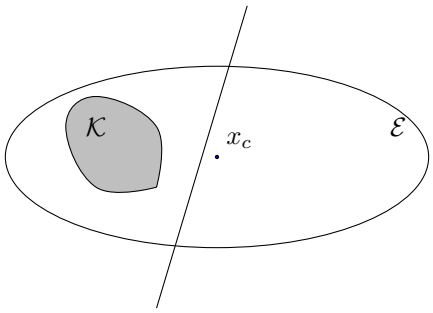

Figure 1: Ellipsoid, feasible region, and cut

Updating the ellipsoid (deep-cut)

Calculation of minimum volume ellipsoid covering:

$$\mathcal{E} \cap \{z \mid g^{\mathsf{T}}(z - x_c) + \beta \leq 0\}$$

Let $\tilde{g} = P\,g$, $\tau^2 = g^{\mathsf{T}} P g$.

- If $n \cdot \beta < -\tau$ (shallow cut), no smaller ellipsoid can be found.
- If $\beta > \tau$, intersection is empty.

Otherwise,

$$x_c^+ = x_c - \frac{\rho}{\tau^2}\tilde{g}, \qquad P^+ = \delta \cdot \left(P - \frac{\sigma}{\tau^2}\tilde{g}\tilde{g}^{\mathsf{T}}\right)$$

where

$$\rho = \frac{\tau + nh}{n+1}, \qquad \sigma = \frac{2\rho}{\tau + \beta}, \qquad \delta = \frac{n^2(\tau^2 - \beta^2)}{(n^2-1)\tau^2}$$

Even better, split P into two variables $\kappa \cdot Q$. Let $\tilde{g} = Q \cdot g$, $\omega = g^\mathsf{T}\tilde{g}$, $\tau = \sqrt{\kappa \cdot \omega}$.

$$x_c^+ = x_c - \frac{\rho}{\omega}\tilde{g}, \qquad Q^+ = Q - \frac{\sigma}{\omega}\tilde{g}\tilde{g}^\mathsf{T}, \qquad \kappa^+ = \delta \cdot \kappa$$

Reduce n^2 multiplications per iteration. Note that:

- The determinant of Q decreases monotonically.
- The range of δ is $(0, \frac{n^2}{n^2-1})$

4.2 Central Cut

A Special case of deep cut when $\beta = 0$. Deserve a separate implement because it is much simplier. Let $\tilde{g} = Q\,g$, $\tau = \sqrt{\kappa \cdot \omega}$,

$$\rho = \frac{\tau}{n+1}, \qquad \sigma = \frac{2}{n+1}, \qquad \delta = \frac{n^2}{n^2-1}.$$

4.3 Parallel Cuts

Oracle returns a pair of cuts instead of just one. The pair of cuts is given by g and (β_1, β_2) such that:

$$g^\mathsf{T}(x - x_c) + \beta_1 \le 0,$$
$$g^\mathsf{T}(x - x_c) + \beta_2 \ge 0,$$

for all $x \in \mathcal{K}$.

Only linear inequality constraint can produce such parallel cut:

$$l \le a^\mathsf{T}x + b \le u, \qquad L \preceq F(x) \preceq U.$$

Usually, provide faster convergence.

Updating the ellipsoid.

Let $\tilde{g} = Q\,g$, $\tau^2 = \kappa \cdot \omega$.

- If $\beta_1 > \beta_2$, intersection is empty.
- If $\beta_1\beta_2 < -\tau^2/n$, no smaller ellipsoid can be found.
- If $\beta_2^2 > \tau^2$, it reduces to deep-cut with $\alpha = \alpha_1$.

Otherwise,

$$x_c^+ = x_c - \frac{\rho}{\omega}\tilde{g}, \qquad Q^+ = Q - \frac{\sigma}{\omega}\tilde{g}\tilde{g}^\mathsf{T}, \qquad \kappa^+ = \delta\kappa.$$

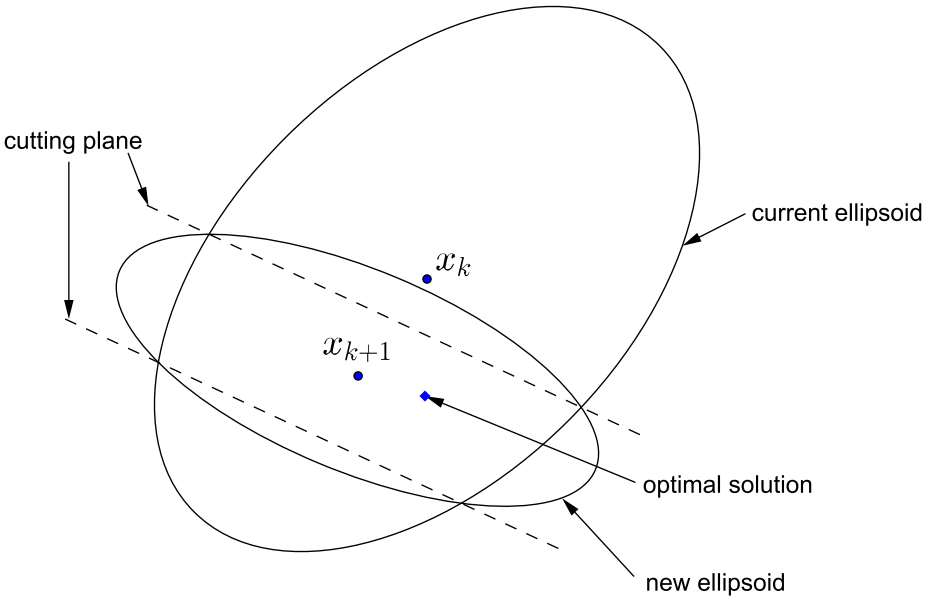

Figure 2: Parallel cuts

where

$$\begin{aligned}
\bar{\beta} &= (\beta_1 + \beta_2)/2 \\
\xi^2 &= (\tau^2 - \beta_1^2)(\tau^2 - \beta_2^2) + (n(\beta_2 - \beta_1)\bar{\beta})^2, \\
\sigma &= (n + (\tau^2 - \beta_1\beta_2 - \xi)/(2\bar{\beta}^2))/(n+1), \\
\rho &= \bar{\beta} \cdot \sigma, \\
\delta &= (n^2/(n^2-1))(\tau^2 - (\beta_1^2 + \beta_2^2)/2 + \xi/n)/\tau^2.
\end{aligned}$$

4.3.1 Example: FIR filter design

A typical structure of digital Finite Impulse Response (FIR) filter is shown in Fig. 3, where the coefficients $h[0], h[1], \ldots, h[n-1]$ must be determined to meet given specifications. Usually, they can be manually designed using windowing or frequency-sampling techniques [13].

However, the experience and knowledge of designers are highly demanded in this kind of design methods. Moreover, there is no guarantee about the design's quality. Therefore, the optimization-based techniques (e.g. [14], more reference) have attracted tons of research effort. In this kind of methods, facilitated with growing computing resource and efficient optimization algorithms, the solution space can be effectively explored.

In the area of optimization algorithms, what is particularly interesting is convex optimization. If a problem is in a convex form, it can be efficiently and optimally solved. Convex optimization techniques are also implementable in designing

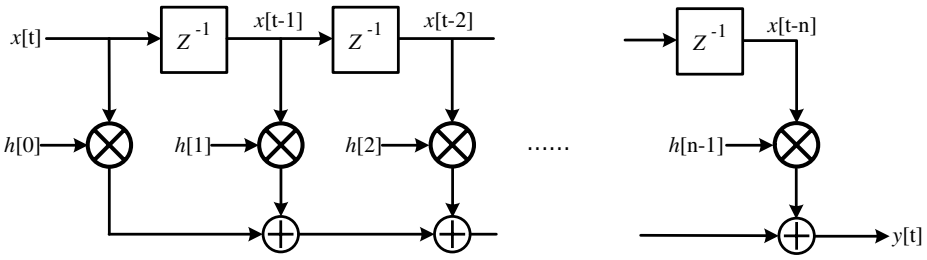

Figure 3: A typical structure of an FIR filter [15].

FIR filters, including Parks-McClellan algorithm [16], METEOR [17] and peak-constrained least-squares (PCLS) [18,19]. In the mentioned articles, with the help of exchange algorithms (e.g. Remez exchange algorithm), certain FIR filter design problems can be formed as linear or quadratic programs. They are two simple forms of convex optimization problems, which can be optimally solved with existing algorithms, such as the interior-point method [20]. Tempted by the optimality, more efforts were devoted to form the problem convex. Particularly, in [14], via spectral factorization [21], the problem of designing an FIR filter with magnitude constraints on frequency-domain is formulated as a convex optimization problem. More examples are provided in [22].

Its time response is

$$y[t] = \sum_{k=0}^{n-1} h[k]u[t-k] \tag{8}$$

where $\mathbf{h} = (h(0), h(1), ..., h(n-1))$ is the filter coefficients. Its frequency response $H : [0, \pi] \to \mathbb{C}$ is

$$H(\omega) = \sum_{m=0}^{n-1} h(m)e^{-jm\omega} \tag{9}$$

where $j = \sqrt{-1}$, n is the order of the filter. The design of a filter with magnitude constraints is often formulated as a constraint optimization problem as the form

$$\begin{aligned} &\min \gamma \\ &\text{s.t.} f(\mathbf{x}) \leq \gamma \\ &\quad g(\mathbf{x}) \leq 0. \end{aligned} \tag{10}$$

where $\mathbf{x}$ is the vector of design variables, $g(\mathbf{x})$ represents the characteristics of the desirable filter and $f(\mathbf{x})$ is the performance metric to be optimized. For example, the magnitude constraints on frequency domain are expressed as

$$L(\omega) \leq |H(\omega)| \leq U(\omega), \forall \omega \in (-\infty, +\infty) \tag{11}$$

where $L(\omega)$ and $U(\omega)$ are the lower and upper (nonnegative) bounds at frequency ω respectively. Note that $H(\omega)$ is 2π periodic and $H(\omega) = \overline{H(-\omega)}$. Therefore, we can only consider the magnitude constraint on $[0, \pi]$ [14].

Generally, the problem might be quite difficult to solve, since the global optimal solution can only be obtained with resource consuming methods, such as branch-and-bound [22]. However, the situation is totally different if the problem is convex, where $f(\mathbf{x})$ and $g(\mathbf{x})$ are convex functions. In such a case, the problem can be optimally solved with many efficient algorithms.

Attracted by the benefits, the authors of [14] transformed , originally non-convex, into a convex form via spectral factorization:

$$L^2(\omega) \leq R(\omega) \leq U^2(\omega), \forall \omega \in (0, \pi) \tag{12}$$

where $R(\omega) = \sum_{i=-n+1}^{n-1} r(t)e^{-j\omega t} = |H(\omega)|^2$ and $\mathbf{r} = (r(-n+1), r(-n+2), \ldots, r(n-1))$ are the autocorrelation coefficients. Especially, $\mathbf{r}$ can be determined by $\mathbf{h}$, with the following equation vice versa [14]:

$$r(t) = \sum_{i=-n+1}^{n-1} h(i)h(i+t), t \in \mathbb{Z}. \tag{13}$$

where $h(t) = 0$ for $t < 0$ or $t > n-1$.

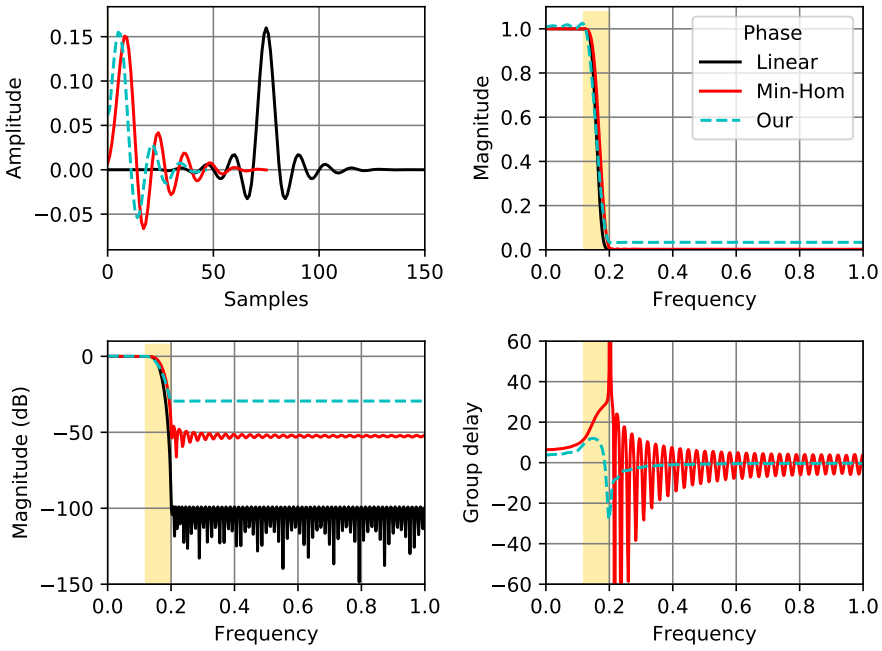

Figure 4: Result

4.3.2 Example: Maximum Likelihood estimation

Consider

$$\begin{array}{ll} \min_{\kappa,p} & \log\det(\Omega(p)+\kappa\cdot I)+\mathrm{Tr}((\Omega(p)+\kappa\cdot I)^{-1}Y), \\ \text{s.t.} & \Omega(p)\succeq 0, \kappa\geq 0. \end{array}$$

Note that the 1st term is concave, the 2nd term is convex. However, if there are enough samples such that Y is a positive definite matrix, then the function is convex within $[0, 2Y]$. Therefore, the following problem is convex:

$$\begin{array}{ll} \min_{\kappa,p} & \log\det V(p)+\mathrm{Tr}(V(p)^{-1}Y), \\ \text{s.t.} & \Omega(p)+\kappa\cdot I = V(p) \\ & 0\preceq V(p)\preceq 2Y, \kappa{>}0. \end{array}$$

4.4 Discrete Optimization

Many engineering problems can be formulated as a convex/geometric programming, e.g. digital circuit sizing. Yet in an ASIC design, often there is only a limited set of choices from the cell library. In other words, some design variables are discrete. The discrete version can be formulated as Mixed-Integer Convex programming (MICP) by mapping the design variables to integers.

What's wrong with the existing methods? Mostly based on relaxation. Then use the relaxed solution as a lower bound and use the branch–and–bound method for the discrete optimal solution. Note that the branch-and-bound method does not utilize the convexity of the problem. What if I can only evaluate constraints on discrete data?

Usually a relaxed optimal solution (convex) is obtained first. Then the optimized discrete solution is obtained by searching exhaustively the neigbourhood. However, sometimes the constraints are tight so that the relaxed continuous optimal solution is far from the discrete one. Enumeration of the discrete domain is difficult.

Consider:

$$\begin{array}{ll} \text{minimize} & f_0(x), \\ \text{subject to} & f_j(x)\leq 0,\ \forall j=1,2,\ldots, \\ & x\in\mathbb{D}, \end{array}$$

where $f_0(x)$ and $f_j(x)$ are "convex". Note that some design variables are discrete. The oracle looks for the nearby discrete solution x_d of x_c with the cutting-plane:

$$g^{\mathsf{T}}(x-x_d)+\beta\leq 0, \beta\geq 0, g\neq 0.$$

Note that the cut may be a shallow cut. Suggestion: use different cuts as possible for each iteration (e.g. round-robin the evaluation of constraints).

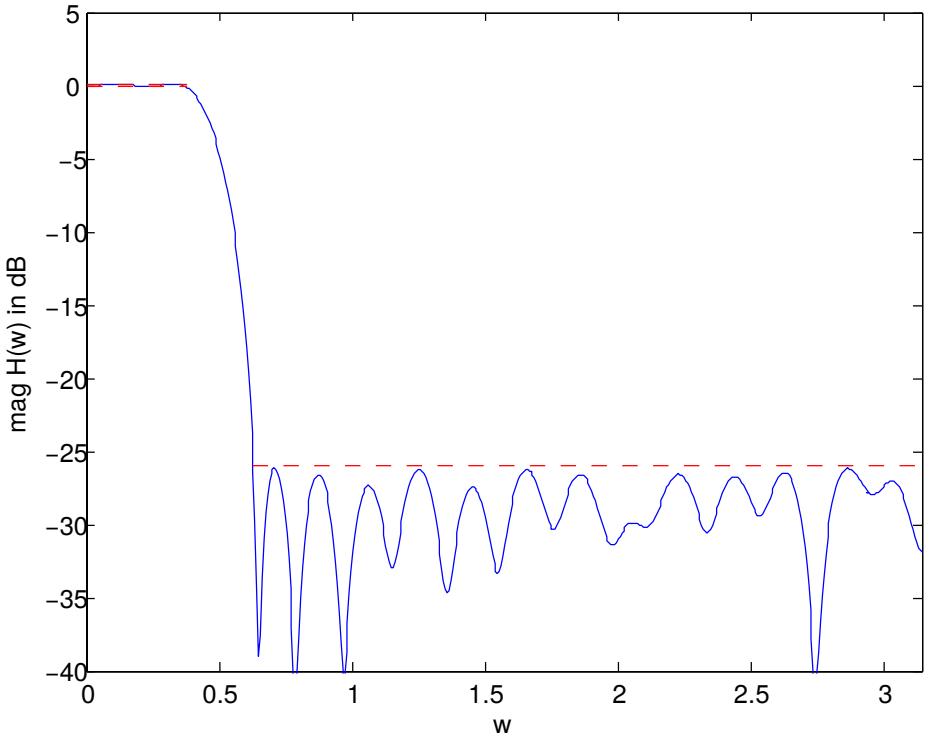

Figure 5: Lowpass ripple

4.4.1 Example: Multiplierless FIR Filter Design

However, there are still many filter design problems that are non-convex, such as multiplierless FIR filter design problems. Note that in Fig. 3, each coefficient associated with a multiplier unit makes the filter power hungry, especially in Application Specific Integrated Circuits (ASIC). Fortunately, it can be implemented multiplierless if each coefficient is quantized and represented as a sum of Singed Power-of-Two (SPT). Such a coefficient can be uniquely represented by a Canonic Signed-Digit (CSD) code [23] with the smallest number of non-zero digits. In such a case, the multiplication is confined to add and shift operations. An example is shown in **??**. A coefficient $0.65625 = 21/32$ can be written as $2^{-1} + 2^{-3} + 2^{-5}$. Consequently, as shown in **??**, the multiplier can be replaced with three shifters and two adders which are with much lower cost. However, the coefficient quantization constraint, which is non-convex, makes the convex optimization algorithm cannot be directly applied. A similar scenario is considering the finite wordlength effect [24].

Attracted by the benefits of the multiplierlessness, many efforts have been devoted to its design techniques. For its general problems, integer programming (e.g. [24–27]) can be implemented to achieve the optimal solution. However, it demands excessive computing resources. Other heuristic techniques, such as genetic algorithm [28] and dynamic-programming-like method [29], are also with low efficiency. Furthermore, if the quantization constraint is the only non-convex constraint in the design problem, a lower bound can be efficiently obtained by solving the relaxed problem [22]. Then to make the solution feasible, it can be either rounded to the nearest CSD codes or treated as a starting point of a local search algorithm for a better solution [30]. However, both of the methods can not guarantee the feasibility of the final solution. Besides, the local search problem is still non-convex. Therefore, the adopted algorithm could also be inefficient, such as branch-and-bound in [30].

References

[1] H. Aliabadi, M. Salahi, Robust geometric programming approach to profit maximization with interval uncertainty, Computer Science Journal of Moldova. 21 (2013) 86–96.

[2] X. Liu, W.-S. Luk, Y. Song, P. Tang, X. Zeng, Robust analog circuit sizing using ellipsoid method and affine arithmetic, in: Proceedings of the 2007 Asia and South Pacific Design Automation Conference, IEEE Computer Society, 2007: pp. 203–208.

[3] B.V. Cherkassky, A.V. Goldberg, Negative-cycle detection algorithms, Mathematical Programming. 85 (1999) 277–311.

[4] R. Tarjan, Shortest paths, AT&T Bell Laboratories, 1981.

[5] A. Dasdan, Experimental analysis of the fastest optimum cycle ratio and mean algorithms, ACM Transactions on Design Automation of Electronic Systems. 9 (2004) 385–418.

[6] J.B. Orlin, U.G. Rothblum, Computing optimal scalings by parametric network algorithms, Mathematical Programming. 32 (1985) 1–10.

[7] A. Dasdan, R.K. Gupta, Faster maximum and minimum mean cycle algorithms for system-performance analysis, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 17 (1998) 889–899.

[8] A. Dasdan, Experimental analysis of the fastest optimum cycle ratio and mean algorithms, ACM Transactions on Design Automation of Electronic Systems (TODAES). 9 (2004) 385–418.

[9] X. Zhou, W.-S. Luk, H. Zhou, F. Yang, C. Yan, X. Zeng, Multi-parameter clock skew scheduling, Integration, the VLSI Journal. 48 (2015) 129–137.

[10] O. Schabenberger, C.A. Gotway, Statistical Methods for Spatial Data Analysis, Chapman & Hall/CRC, Florida, 2005.

[11] P.J. Diggle, P.J.R. Jr., Model-based Geostatistics, Springer, New York, 2007.

[12] R.G. Bland, D. Goldfarb, M.J. Todd, The ellipsoid method: A survey, Operations Research. 29 (1981) 1039–1091.

[13] A.V. Oppenheim, R.W. Schafer, J.R. Buck, others, Discrete-time signal processing, Prentice hall Englewood Cliffs, NJ, 1989.

[14] S.-P. Wu, S. Boyd, L. Vandenberghe, FIR filter design via spectral factorization and convex optimization, in: Applied and Computational Control, Signals, and Circuits, Springer, 1999: pp. 215–245.

[15] S.K. Mitra, Y. Kuo, Digital signal processing: A computer-based approach, McGraw-Hill New York, 2006.

[16] T.W. Parks, J.H. McClellan, Chebyshev approximation for nonrecursive digital filters with linear phase, Circuit Theory, IEEE Transactions on. CT-19 (1972) 189–194.

[17] K. Steiglitz, T.W. Parks, J.F. Kaiser, METEOR: A constraint-based FIR filter design program, Signal Processing, IEEE Transactions on. 40 (1992) 1901–1909.

[18] I.W. Selesnick, M. Lang, C.S. Burrus, Constrained least square design for FIR filters without specified transition bands, Signal Processing, IEEE Transactions on. 44 (1996) 1879–1892.

[19] J.W. Adams, J.L. Sullivan, Peak-constrained least-squares optimization, Signal Processing, IEEE Transactions on. 46 (1998) 306–321.

[20] S. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2009.

[21] T.N.T. Goodman, C.A. Micchelli, G. Rodriguez, and S. Seatzu, Spectral factorization of laurent polynomials, Advances Comput. Math. 7 (1997) 429–454.

[22] T.N. Davidson, Enriching the art of FIR filter design via convex optimization, Signal Processing Magazine, IEEE. 27 (2010) 89–101.

[23] G.W. Reitwiesner, Binary Arithmetic, Advances in Computers. 1 (1960) 231–308.

[24] Y.C. Lim, S. Parker, A. Constantinides, Finite word length FIR filter design using integer programming over a discrete coefficient space, Acoustics, Speech and Signal Processing, IEEE Transactions on. 30 (1982) 661–664.

[25] D.M. Kodek, Design of optimal finite wordlength FIR digital filters using integer programming techniques, Acoustics, Speech and Signal Processing, IEEE Transactions on. 28 (1980) 304–308.

[26] Y. Lim, S. Parker, FIR filter design over a discrete powers-of-two coefficient space, Acoustics, Speech and Signal Processing, IEEE Transactions on. 31 (1983) 583–591.

[27] Y.C. Lim, R. Yang, D. Li, J. Song, Signed power-of-two term allocation scheme for the design of digital filters, Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on. 46 (1999) 577–584.

[28] D.J. Xu, M.L. Daley, Design of optimal digital filter using a parallel genetic algorithm, Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on. 42 (1995) 673–675.

[29] C.-L. Chen, A.N. Willson Jr, A trellis search algorithm for the design of FIR filters with signed-powers-of-two coefficients, Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on. 46 (1999) 29–39.

[30] D. Kodek, K. Steiglitz, Comparison of optimal and local search methods for designing finite wordlength FIR digital filters, Circuits and Systems, IEEE Transactions on. 28 (1981) 28–32.