Handling Nonnegative Constraints in Spectral Estimation

Brien Alkire and Lieven Vandenberghe

Electrical Engineering Department
University of California, Los Angeles
(brien@alkires.com, vandenbe@ee.ucla.edu)

Abstract

We consider convex optimization problems with the constraint that the variables form a finite autocorrelation sequence, or equivalently, that the corresponding power spectral density is nonnegative. This constraint is often approximated by sampling the power spectral density, which results in a set of linear inequalities. It can also be cast as a linear matrix inequality via the positive-real lemma. The linear matrix inequality formulation is exact, and results in convex optimization problems that can be solved using interior-point methods for semidefinite programming. However, these methods require $O(n^6)$ floating point operations per iteration, if a general-purpose implementation is used. We introduce a much more efficient method with a complexity of $O(n^3)$ flops per iteration.

1 Introduction

The following problem arises in MA and ARMA estimation [6]. Given a vector $\bar{x} \in \mathbf{R}^{n+1}$, and a positive definite matrix $Q = Q^T \in \mathbf{R}^{(n+1)\times(n+1)}$, solve the optimization problem

$$\begin{array}{ll} \text{minimize} & (x-\bar{x})^T Q (x-\bar{x}) \\ \text{subject to} & x \in \mathcal{C}^{n+1} \end{array} \qquad (1)$$

Research supported by the NSF under grant ECS-9733450.

where $\mathcal{C}^{n+1}$ is the set of *finite autocorrelation sequences* in $\mathbf{R}^{n+1}$, *i.e.*, $x \in \mathcal{C}^{n+1}$ if and only if

$$x_i = \sum_{k=0}^{n-i} y_k y_{k+i}, \quad i = 0, \ldots, n \qquad (2)$$

for some vector $y \in \mathbf{R}^{n+1}$. The variable in problem (1) is $x \in \mathbf{R}^{n+1}$.

It is well known that $x \in \mathcal{C}^{n+1}$ if and only if

$$X(\omega) \geq 0, \quad 0 \leq \omega \leq \pi \qquad (3)$$

where X is the Fourier transform of x, defined as

$$X(\omega) = x_0 + 2\sum_{k=1}^{n} x_k \cos(k\omega).$$

This characterizes $\mathcal{C}^{n+1}$ via an infinite set of linear inequalities in x; one for each value of ω. Researchers often handle constraints of the form $x \in \mathcal{C}^{n+1}$ by sampling the frequency response [8, 9]. For example, we can approximate problem (1) as

$$\begin{array}{ll} \text{minimize} & (x-\bar{x})^T Q (x-\bar{x}) \\ \text{subject to} & X(\omega_k) \geq 0, \quad k = 1, \ldots, m. \end{array}$$

where $\omega_1, \ldots, \omega_m \in [0, \pi]$. This is a quadratic programming problem in the variable x. Although it is quite efficient, the drawback of this method is that it is not exact.

An alternative representation, which does not involve any approximation, is based on the positive-real lemma and linear matrix inequalities (LMIs). It can be shown that $x \in \mathcal{C}^{n+1}$ if and only if there exists a matrix $P = P^T \in \mathbf{R}^{n\times n}$ such that

$$G(x, P) \equiv \begin{bmatrix} P & \hat{x} \\ \hat{x}^T & x_0 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & P \end{bmatrix} \succeq 0 \qquad (4)$$

where $\hat{x}^T = [x_1\, x_2 \ldots x_n]$, and the inequality $\succeq$ denotes matrix inequality, *i.e.*, $G(x, P)$ is positive semidefinite. Applying this result to (1), we obtain

$$\begin{array}{ll} \text{minimize} & (x-\bar{x})^T Q (x-\bar{x}) \\ \text{subject to} & G(x, P) \succeq 0, \end{array} \qquad (5)$$

which is a convex optimization problem in the variables $P = P^T \in \mathbf{R}^{n\times n}$ and $x \in \mathbf{R}^{n+1}$. The algorithm used in [6] is based on solving this problem using general-purpose interior-point methods for semidefinite programming (SDP). The drawback of this approach is that it introduces $n(n+1)/2$ auxiliary variables (the elements of P). The cost of one iteration of an interior-point method typically grows as the cube of the number of variables. The cost of solving (5) using standard SDP software is therefore at least $O(n^6)$ flops per iteration.

The purpose of this paper is to describe a more efficient interior-point method for handling the constraint $x \in \mathcal{C}^{n+1}$, with a computational complexity of only $O(n^3)$ per iteration.

We also point out that the constraint $x \in \mathcal{C}^{n+1}$ occurs in many other signal processing problems, such as spectral estimation and FIR filter design [1, 8, 9, 3, 7]. The techniques described in this paper are also applicable to problems in those fields. In this paper we will concentrate on problem (1) as a simple representative example, and refer to [1] for details on other applications.

An outline of this paper is as follows. In §2 we show that the set $\mathcal{C}^{n+1}$ is a convex cone and derive its dual cone. In §3 we introduce the dual optimization problem corresponding to (1) and show how to obtain a solution to (1) from the solution to the dual problem. In §4 we discuss a numerical method based on the dual problem, and show that it is much more efficient than the method based on the positive-real lemma. We give some numerical results in §5, and summarize our findings in §6.

2 The dual cone

The frequency-domain characterization of $\mathcal{C}^{n+1}$ given in (3) has several important consequences. First, it immediately implies that $\mathcal{C}^{n+1}$ is a *cone*: if $x \in \mathcal{C}^{n+1}$, then obviously $tx \in \mathcal{C}^{n+1}$ for all $t \geq 0$. Secondly, we note that for fixed ω, (3) is a linear inequality in x. Therefore $\mathcal{C}^{n+1}$ is the intersection of infinitely many halfspaces, parameterized by ω. As a consequence, $\mathcal{C}^{n+1}$ is a closed convex cone.

The dual cone of $\mathcal{C}^{n+1}$ is defined as

$$\mathcal{C}_D^{n+1} = \left\{ z \in \mathbf{R}^{n+1} | z^T x \geq 0 \; \forall x \in \mathcal{C}^{n+1} \right\}. \qquad (6)$$

(From this point forward we will also refer to $\mathcal{C}^{n+1}$ as the *primal cone*.) Using the definition of $\mathcal{C}^{n+1}$ from (2) we see that $z \in \mathcal{C}_D^{n+1}$ if and only if

$$\sum_{i=0}^{n} z_i \sum_{k=0}^{n-i} y_k y_{k+i} = \frac{1}{2} y^T F(z) y \geq 0$$

for all $y \in \mathbf{R}^{n+1}$ where $F(z)$ is the Toeplitz matrix given by

$$F(z) = \begin{bmatrix} 2z_0 & z_1 & z_2 & \cdots & z_n \\ z_1 & 2z_0 & z_1 & \cdots & z_{n-1} \\ z_2 & z_1 & 2z_0 & \cdots & z_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z_n & z_{n-1} & z_{n-2} & \cdots & 2z_0 \end{bmatrix}. \qquad (7)$$

In other words, $z \in \mathcal{C}_D^{n+1}$ if and only if z satisifies the LMI $F(z) \succeq 0$.

3 Lagrange dual problem

The Lagrangian function L associated with (1) is defined as

$$L(x,z) = (x-\bar{x})^T Q (x-\bar{x}) - x^T z, \qquad (8)$$

where $z \in \mathbf{R}^{n+1}$ is the dual variable or Lagrange multiplier associated with the 'generalized inequality' $x \in \mathcal{C}^{n+1}$ (see [2] for details). The Lagrange dual problem is defined as

$$\begin{array}{ll} \text{maximize} & \inf_x L(x,z) \\ \text{subject to} & z \in \mathcal{C}_D^{n+1}, \end{array}$$

or more explicitly,

$$\begin{array}{ll} \text{maximize} & -\frac{1}{4} z^T Q^{-1} z - \bar{x}^T z \\ \text{subject to} & z \in \mathcal{C}_D^{n+1}. \end{array} \qquad (9)$$

It can be shown that the optimal values of the primal problem (1) and the dual problem (9) are equal, and that both optimal values are attained, *i.e.*, there exists a pair of (unique) optimal solutions x^* and z^*. Moreover, the optimal solutions satisfy

$$x^* = \frac{1}{2} Q^{-1} z^* + \bar{x}. \qquad (10)$$

We can therefore solve the primal problem (1) by solving the dual problem, and then calculating x^* from z^* using (10).

Using the characterization of the dual cone in terms of the Toeplitz matrix (7), we can re-write the dual problem (9) as

$$\begin{array}{ll} \text{maximize} & -\frac{1}{4} z^T Q^{-1} z - \bar{x}^T z \\ \text{subject to} & F(z) \succeq 0, \end{array} \qquad (11)$$

which is a convex optimization problem with a quadratic objective function and an LMI constraint. In the next section we derive the computational cost of solving (11).

4 Efficient solution of the dual problem

Suppose we solve problem (11) using a barrier method, such as SUMT [4, 2], which is based on minimizing the function

$$t\,(\frac{1}{4}z^TQ^{-1}z + \bar{x}^Tz) - \log\det F(z) \qquad (12)$$

for a sequence of increasing values of t. The cost of one iteration of this method is dominated by the cost of forming the gradient and Hessian of the barrier function

$$\phi(z) = -\log\det F(z),$$

and the cost of solving the Newton equation for (12), which is given by

$$(\frac{t}{2}Q^{-1} + \nabla^2\phi(z))v = -t(\frac{1}{2}Q^{-1}z + \bar{x}) - \nabla\phi(z).$$

The Newton equation can be solved in $O(n^3)$ flops, since we have $n+1$ variables. We now show that the gradient and Hessian of ϕ can also be evaluated in $O(n^3)$ flops.

The first and second derivatives of ϕ are given by

$$\begin{aligned}\nabla\phi(z)_j &= -\,\mathbf{Tr}(E^j + (E^j)^T)F(z)^{-1}\\ &= -2\,\mathbf{Tr}\,E^jF(z)^{-1},\end{aligned}$$

and

$$\begin{aligned}&\nabla^2\phi(z)_{ij} =\\ &= \mathbf{Tr}\,F(z)^{-1}(E^i + (E^i)^T)F(z)^{-1}(E^j + (E^j)^T)\\ &= 2\,\mathbf{Tr}\,E^iF(z)^{-1}E^jF(z)^{-1}\\ &\quad + 2\,\mathbf{Tr}\,E^iF(z)^{-1}{E^j}^TF(z)^{-1}\end{aligned}$$

for $i,j = 0,1,\ldots,n$, where $E \in \mathbf{R}^{(n+1)\times(n+1)}$ is the unit-shift matrix, defined as

$$E = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0\\ 1 & 0 & 0 & \cdots & 0 & 0\\ 0 & 1 & 0 & \cdots & 0 & 0\\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots\\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix},$$

and E^j denotes the jth power of E.

The gradient and Hessian can be efficiently evaluated as follows. We first factorize $F(z)^{-1}$ as

$$F(z)^{-1} = RR^T$$

where R is upper triangular. The Cholesky factor R can be obtained using the Levinson-Durbin algorithm at a cost of $O(n^2)$ flops. Let r_k be the kth column of R. The gradient and Hessian of ϕ can be written as

$$\nabla\phi(z)_j = -2\sum_{k=0}^{n} r_k^TE^jr_k$$

and

$$\nabla\phi(z)_{ij} = 2\sum_{k=0}^{n}\sum_{l=0}^{n}(r_k^TE^ir_l)(r_l^TE^jr_k + r_k^TE^jr_l)$$

for $i = 0,\ldots,n$ and $j = 0,\ldots,n$. More compactly, the gradient can be written as

$$\nabla\phi(z) = -2\sum_{k=0}^{n} c(k,k) \qquad (13)$$

and the Hessian as

$$\nabla^2\phi(z) = 2\sum_{k=0}^{n}\sum_{l=0}^{n} c(k,l)(c(l,k) + c(k,l))^T, \qquad (14)$$

where $c(k,l) \in \mathbf{R}^{n+1}$ denotes the *crosscorrelation* between the vectors r_k and r_l, *i.e.*,

$$c_i(k,l) = r_k^TE^ir_l = \sum_{j=0}^{n-j} r_{k,j+i}\,r_{lj}.$$

The cost of a straightforward evaluation of the expressions (13) and (14) is $O(n^3)$ flops and $O(n^4)$ flops, respectively. It takes $O(n^3)$ flops to calculate the autocorrelation vectors $c(k,k)$ by working out the inner products in the definition, and the addition in (13) costs $O(n^2)$ flops. Evaluating all crosscorrelation vectors $c(k,l)$ would take $O(n^4)$ flops, and the sum in (14) requires another $O(n^4)$ flops.

A more efficient method is based on the discrete Fourier transform (DFT). We define a complex matrix $W \in \mathbf{C}^{N\times(n+1)}$ with elements

$$W_{ik} = e^{-ik(2\pi\sqrt{-1}/N)},$$

for $i = 0,\ldots,N-1$ and $k = 0,\ldots,n$ where N is the smallest power of two greater than or equal to $2(n+1)$. The DFT of a vector $x \in \mathbf{R}^{n+1}$ is the vector $X \in \mathbf{C}^N$, defined as

$$X = Wx.$$

where x is assumed to be zero-padded to length N. It is readily verified that $\frac{1}{N}W^*W = I$, so we can easily obtain the inverse DFT of a vector $X \in \mathbf{C}^N$, using

$$x = \frac{1}{N}W^*X.$$

We now return to the expressions for the gradient and Hessian in (13) and (14). Let $R_k = Wr_k$ and $C(k,l)$ be the DFTs of r_k and $c(k,l)$. The DFT $C(k,l)$ is readily computed from R_k and R_l using well known properties of the DFT [5, §8]:

$$C(k,l) = \mathbf{diag}(R_k)\overline{R}_l = \mathbf{diag}(\overline{R}_l)R_k$$

where $\overline{R}_l$ denotes the complex conjugate of R_l, and $\mathbf{diag}\, R_k$ is the diagonal matrix with R_k on its diagonal. In particular, we note that $C(k,l) = \overline{C}(l,k)$, and that $C(k,k)$ is real.

The previous expression (13) for the gradient can be written in terms of the vectors R_k as follows:

$$\begin{aligned}\nabla\phi(z) &= -\frac{2}{N}W^* \sum_{k=0}^{n} C(k,k) \\ &= -\frac{2}{N}W^* \sum_{k=0}^{n} \mathbf{diag}(R_k)\overline{R}_k. \qquad (15)\end{aligned}$$

In other words, the gradient is the inverse DFT of a vector with components

$$-2\sum_{k=0}^{n} R_{ki}\overline{R}_{ki} = -2\sum_{k=0}^{n} |R_{ki}|^2,$$

for $i = 0, \ldots, N$. The expression for the Hessian (14) is more complicated. We have

$$\begin{aligned}&\nabla^2\phi(z) \\ &= \frac{2}{N^2}W^* \left(\sum_{k=0}^{n}\sum_{l=0}^{n} C(k,l)(C(l,k) + C(k,l))^*\right) W \\ &= \frac{2}{N^2}W^* \left(\sum_{l=0}^{n} \mathbf{diag}(\overline{R}_l)\left(\left(\sum_{k=0}^{n} R_k R_k^*\right)\mathbf{diag}(R_l)\right.\right. \\ &\qquad \left.\left. + \left(\sum_{k=0}^{n} R_k R_k^T\right)\mathbf{diag}(\overline{R}_l)\right)\right) W. \qquad (16)\end{aligned}$$

The formulas (15) and (16) suggest a more efficient way of evaluating gradient and Hessian. Calculating the gradient from the vectors R_k requires only $O(n^2)$ flops, while calculating the Hessian via (16) takes $O(n^3)$ flops.

In summary, the proposed algorithm for evaluating the barrier function $\phi(z)$, its gradient $\nabla\phi(z)$ and Hessian $\nabla^2\phi(z)$, proceeds as follows:

1. calculate the Cholesky factorization $F(z)^{-1} = RR^T$ via the Levinson-Durbin algorithm ($O(n^2)$ flops)

2. the value of the barrier function is given by $\phi(z) = 2\sum_{k=0}^{n} \log r_{kk}$, where r_{kk} is the kth diagonal element of R

3. choose an integer $N \geq 2(n+1)$ (for example, the smallest power of 2 greater than $2(n+1)$), and calculate the DFTs R_k of the columns of R ($O(n^2 \log n)$ flops)

4. evaluate the gradient via the expression (15) ($O(n^2)$ flops)

5. evaluate the Hessian via (16) ($O(n^3)$ flops).

The total cost is $O(n^3)$ flops. For comparison, one iteration of a barrier method applied to the primal problem (5) would have a complexity of at least $O(n^6)$ flops, since we have $O(n^2)$ variables.

$n+1$	time (sec.)
100	0.14
200	0.78
300	4.00
400	4.65
500	5.52
600	25.36

Table 1: CPU times for the Hessian and gradient.

$n+1$	time (sec.)	time/iter. (sec.)
100	5.3	0.16
200	34.7	0.89
300	252.7	4.37
400	312.1	5.10
500	324.7	6.13
600	2033.8	27.94

Table 2: CPU times for the example problem.

5 Numerical Results

Table 1 lists CPU times required for evaluation of the gradient and Hessian of $\phi(z)$ as a function of problem size. Notice the jump in CPU time that results when the problem size crosses a power of two. This is due to the change in the length of the FFT that is used. The code was written in C++. Calls were made to optimized BLAS, LAPACK and FFT libraries. Specifically, the multi-threaded Intel Math Kernel Library and Signal Processing Library were used. The code was executed on a dual 350MHz PII system.

Table 2 lists CPU times required for solving (9) with $Q = I$, and randomly generated vectors $\bar{x}$. The results were averaged over five instances for each problem size. The implementation is a very basic version of the SUMT method (typically requiring over

50 Newton iterations), with the optimized C++ code for evaluating gradients and Hessians. Note that for $n = 600$ the primal SDP formulation (5) would involve solving an SDP with about $180,000$ variables.

6 Conclusions

We considered efficient interior-point methods for convex optimization prolems involving finite autocorrelation sequences. Our approach is based on solving the dual problem, which has a smaller number of variables, and includes an LMI constraint with Toeplitz structure. By taking advantage of the Toeplitz structure, we reduce the cost to $O(n^3)$ flops per iteration. This is much lower than previously used methods based on the positive real lemma and general-purpose semidefinite programming solvers.

References

[1] B. Alkire and L. Vandenberghe. Convex optimization with constraints on the cone of finite autocorrelation sequences. Technical report, UCLA Electrical Engineering Department, 2000. In preparation.

[2] S. Boyd and L. Vandenberghe. Convex optimization. UCLA Academic Publishing, January 1999. Course Reader for EE236B: Nonlinear Programming.

[3] T. N. Davidson, Z.-Q. Luo, and K. M. Wong. Design of orthogonal pulse shapes for communications via semidefinite programming. *IEEE Transactions on Signal Processing*, 48(5):1433–1445, 2000.

[4] Y. Nesterov and A. Nemirovsky. *Interior-point polynomial methods in convex programming*, volume 13 of *Studies in Applied Mathematics.* SIAM, Philadelphia, PA, 1994.

[5] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing.* Prentice Hall, 1989.

[6] P. Stoica, T. McKelvey, and J. Mari. MA estimation in polynomial time. *IEEE Transactions on Signal Processing*, 48(7):1999–2012, July 2000.

[7] P. Stoica and R. Moses. *Introduction to Spectral Analysis.* Prentice Hall, 1997.

[8] S.-P. Wu, S. Boyd, and L. Vandenberghe. FIR filter design via semidefinite programming and spectral factorization. In *Proc. IEEE Conf. on Decision and Control*, pages 271–276, 1996.

[9] S.-P. Wu, S. Boyd, and L. Vandenberghe. FIR filter design via spectral factorization and convex optimization. In B. Datta, editor, *Applied and Computational Control, Signals and Circuits*, pages 219–250. Birkhauser, 1998.