Penalty/Barrier Multiplier Algorithm for Semidefinite Programming

Leonid Mosheyev[1]
mosheyev@ie.technion.ac.il

Michael Zibulevsky[2]
michael@cs.unm.edu

October 1999

Abstract

We present a generalization of the Penalty/Barrier Multiplier algorithm for the semidefinite programming, based on a matrix form of Lagrange multipliers. Our approach allows to use among others *logarithmic, shifted logarithmic, exponential* and a very effective *quadratic-logarithmic* penalty/barrier functions. We present dual analysis of the method, based on its correspondence to a proximal point algorithm with nonquadratic distance-like function. We give computationally tractable dual bounds, which are produced by the Legendre transformation of the penalty function. Numerical results for large-scale problems from robust control, robust truss topology design and free material design demonstrate high efficiency of the algorithm.

[1]The research was supported in part by the grant # 9636-1-96 of the Israeli Ministry of Sciences and The Center for Absorption in Science, Ministry of Immigrant Absorption State of Israel

[2]The research was supported in part by the Optimization Laboratory in the Technion - Israel Institute of Technology and by Albuquerque High Performance Computing Center

1 Introduction

Optimization under linear matrix inequality (LMI) constraints, or *semidefinite programming* (SDP), is a very attractive field for the application of modern interior-point methods. Numerous problems in control and systems theory [9], combinatorics [1], structural design, robust optimization and many others (see for example [4]) can be expressed in terms of SDP. Various extensions to semidefinite programming can be found in [1, 8, 15, 19, 21] and references therein.

Methods of multipliers, involving *nonquadratic augmented Lagrangians* [6, 7, 10, 22, 26, 29] successfully compete with the interior-point methods in non-linear programming. Especially efficient they are when a very high accuracy of solution is required. This success is partially explained by the fact, that due to iterative update of multipliers, the penalty parameter does not need to become extremely small in the neighborhood of solution. Additional improvement of numerical efficiency was obtained by using a "soft" *quadratic-logarithmic* penalty function, introduced in [7, 29]. This function has a bounded third derivative and hence is well matched to the Newton method.

In this paper we present an extension of the Penalty/Barrier Multiplier (PBM) algorithm [7, 29] to the semidefinite programming, based on a matrix form of Lagrange multipliers. This method inherits abovementioned advantages of nonquadratic augmented Lagrangian algorithms. A close related approach with exponential penalty function was also studied in [11].

The paper is organized as follows. Section 3 describes the primal and Lagrange dual semidefinite programming problems and optimality conditions. We present the PBM algorithm in Section 4, and in Section 5 give dual analysis of the method, based on its correspondence to a proximal-point algorithm with nonquadratic distance-like function[3]. Section 6 gives computationally tractable dual bounds for the objective function, which are produced by the Legendre transformation of the penalty function. Section 7 discusses implementation aspects, in particular, the use of the Newton method with *frozen Hessian* for unconstrained minimization. Numerical results for large-scale problems from robust control, robust truss topology design and free material design demonstrate the high efficiency of the algorithm.

2 Notation and Preliminaries

The following notation is used in the sequel.

• S^m — the space of symmetric $m \times m$ matrices of a given block-diagonal structure with real entries. As usual, S^m is provided with the scalar product

$$\langle A, B \rangle = Tr\{AB\}.$$

The corresponding induced norm is Frobenius norm

$$\|A\|_F = (Tr\{AA\})^{1/2}.$$

[3]Results in sections 4 and 5 are mainly based on [28].

• S^m_{++} (S^m_+)	— the cone of positive definite (semidefinite) matrices from S^m.
• $\mathbb{R}^n$	— the n-dimensional Euclidean space.
• $\mathbb{R}$ $(\mathbb{R}_{--})$	— the real (negative real) line.
• I	— the identity matrix.
• $x = (x_1, ..., x_n)$	— a vector of real scalar variables.
• x^*	— a vector of the optimal solution.
• $\| \cdot \|$	— the Euclidean norm.
• $Tr\{\cdot\}$	— trace of a matrix.
• $\succ$ $(\succeq)$	— positive definite (semidefinite).
• $A_0, A_1, ..., A_n$	— given symmetric $m \times m$ matrices from S^m.
• $\mathcal{A}$	— a linear transformation from $\mathbb{R}^n$ to S^m, where

$$\mathcal{A}x = \sum_{i=1}^{n} x_i A_i. \tag{1}$$

- $\mathcal{A}^*$ — the operator adjoint to $\mathcal{A}$, it denotes a linear transformation from S^m to $\mathbb{R}^n$, where

$$\mathcal{A}^* B = (Tr\{A_1 B\},\ \dots\ , Tr\{A_n B\})^T. \tag{2}$$

3 Primal and Dual Problem Formulations

We consider the following problem of minimizing a convex function under LMI constraint:

$$(SDP) \quad Minimize \;\; f(x) \;\; s.t. \;\; A(x) \equiv \mathcal{A}x + A_0 \succeq 0,$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a twice differentiable convex function, defined over the entire space $\mathbb{R}^n$; $A(x)$ is a symmetric matrix that depends affinely on the entries of the vector $x \in \mathbb{R}^n$. The problem is defined by the function f and $n+1$ symmetric matrices $A_0, A_1,\ \dots\ , A_n \in S^m$. This problem is called a semidefinite program, following Nesterov and Nemirovski [19].

Problem (SDP) is a convex optimization problem since its objective and LMI constraint are convex. We make the following assumptions about (SDP):

(a) Problem (SDP) is strictly feasible, i.e. there exists a vector $\widehat{x}$, which satisfies $A(\widehat{x}) \succ 0$ (Slater condition).

(b) The optimal solution set X^* for (SDP) is nonempty and compact.

(c) $\mathcal{A}x$ is nonzero whenever x is nonzero (non-degeneracy assumption), i.e. the matrices A_i $(i = 1, ..., n)$ are linearly independent.

Associated with the primal problem (SDP) is its Lagrangian

$$L(x, U) = f(x) - \langle A(x), U \rangle, \tag{3}$$

where U is a symmetric matrix of Lagrange multipliers from S^m_+. Theorems about optimality conditions for (SDP) and a saddle point of the Lagrangian may be found in Appendix A.

Let $G(U)$ be the dual functional associated with (SDP) given by

$$G(U) = \min\{L(x,U) : x \in I\!R^n\}. \tag{4}$$

The matrix function G is the objective function of the dual concave problem

$$(DSDP) \quad Maximize \;\; G(U) \;\; s.t. \;\; U \succeq 0. \tag{5}$$

Duality Theorem (follows directly from the saddle point theorem in appendix A)
(1) The optimal value of (DSDP) $G(U^*)$ is equal to the optimal value of (SDP) $f(x^*)$
(2) The value $G(U)$ of any dual feasible matrix U is less than or equal to the objective $f(x)$ of any primal feasible vector x.

4 The PBM Algorithm

In this section we present the PBM algorithm for semidefinite programming. Let $\varphi : I\!R \to I\!R$ be a real-valued penalty function having the following properties [7]:

($\varphi 0$) φ is a strictly decreasing twice differentiable strictly convex function of one variable with $Dom\ \varphi = (b, +\infty)$, $-\infty \leq b < 0$,

($\varphi 1$) $\lim_{t \to b} \varphi'(t) = -\infty$,

($\varphi 2$) $\lim_{t \to \infty} \varphi'(t) = 0$,

($\varphi 3$) $\varphi(0) = 0$,

($\varphi 4$) $\varphi'(0) = -1$.

We introduce a family of penalty functions depending on the positive penalty parameter p

$$\varphi_p(t) = p\varphi(t/p)$$

It is easy to check that if the properties ($\varphi 0 - \varphi 4$) are valid for φ then they are valid for φ_p as well.

Penalty/barrier algorithm. The basic idea of the algorithm is very simple: positive semidefiniteness of the matrix $A(x)$ means non-negativity of its eigenvalues $\lambda_1, .., \lambda_m$. So, we can penalize the LMI constraint of (SDP) using the term

$$Tr\{\varphi_p(A(x))\} = \sum_{i=1}^{m} \varphi_p\left(\lambda_i(x)\right) \tag{6}$$

and build the aggregate

$$F_p(x) = f(x) + Tr\{\varphi_p(A(x))\}\ . \tag{7}$$

As $p \to 0$, the path of minimizers of (7) converges, under mild regularity assumptions, to the optimal set of (SDP), and we just trace this path.

The penalty term $Tr\{\varphi(A)\}$ has the gradient (see Appendix B)

$$\nabla\, Tr\{\varphi(A)\} = \varphi'(A), \tag{8}$$

where φ' is a derivative of the scalar function φ. Hence the corresponding gradient with respect to x is

$$\nabla_x\, Tr\{\varphi(A(x))\} = \mathcal{A}^* \varphi'(A(x))\ .$$

The Hessian of the penalty term and the proof of its convexity are given in Appendix C.

Modified Lagrangian. Using the substitution

$$U = V^2, \tag{9}$$

where $V \in S^m_+$, and the commutativity under trace, one may rewrite the Lagrangian (3) as

$$\widetilde{L}(x, V) = f(x) - Tr\{VA(x)V\}.$$

Now we introduce the modified Lagrangian for the problem (SDP):

$$F_p(x, V) = f(x) + Tr\{\varphi_p(VA(x)V)\}. \tag{10}$$

PBM algorithm. The PBM algorithm for solving (SDP) is iterative. At each iteration the aggregate (10) is minimized with respect to x

$$x^k = \arg\min_x F_{p_k}(x, V_k), \tag{11}$$

and then the multiplier V_k and the penalty parameter p_k are updated:

$$\begin{aligned} V_{k+1} &= [-V_k \varphi'_{p_k}(V_k A(x^k) V_k) V_k]^{\frac{1}{2}} \ , & (12) \\ p_{k+1} &= \pi_k p_k \ , \quad 0 < \pi_k \leq 1. & (13) \end{aligned}$$

Motivation of the multiplier updating formula. After the multiplier update, we would like x^k to become also a minimizer of the standard Lagrangian (3)

$$x^k = \arg\min_x L(x, U_{k+1}) \tag{14}$$

Using $A(x) = \mathcal{A}x + A_0$ and the optimality condition on x^k in (14), we have

$$\nabla_x L(x^k, U_{k+1}) = \nabla f(x^k) - \mathcal{A}^* U_{k+1} = 0. \tag{15}$$

On the other hand, using (8) and the optimality condition on x^k in (11), we have

$$\nabla_x F_{p_k}(x^k, V_k) = \nabla f(x^k) + \mathcal{A}^* V_k \varphi'_{p_k}(V_k A(x^k) V_k) V_k = 0. \tag{16}$$

Comparing (16) and (15), and taking into account (9), we obtain the updating formula (12).

4.1 Choice of the Penalty/Barrier Function

The choice of the function φ influences significantly the efficiency of the method. Here we present few known examples of φ from non-linear programming. In our implementation of the PBM algorithm we prefer to use the *quadratic-logarithmic* penalty function, given in the last example in this subsection. This penalty function has a bounded third derivative and hence is well matched to the Newton method.

1. **Logarithmic Barrier Function** [19]

$$\varphi(t) = -\log t.$$

For this function

$$Tr\{\varphi(A(x))\} = -\sum_{i=1}^{m} \log \lambda_i(x) = -\log \prod_{i=1}^{m} \lambda_i(x) = -\log \det A(x)$$

This function yields a self-concordant barrier for (SDP) and the corresponding algorithm possesses the polynomial complexity bounds [19]. (This function satisfies the properties $(\varphi 0 - \varphi 1)$ only, and cannot be used in the PBM algorithm.) Its gradient may be obtained using result in Appendix B

$$\nabla Tr\{\varphi(A)\} = \varphi'(A) = A^{-1}.$$

2. **Modified Barrier Function** [22]

$$\varphi(t) = -\log(t+1)\ , \qquad -1 < t < \infty.$$

For this function

$$Tr\{\varphi(A(x))\} = -\sum_{i=1}^{m} \log(\lambda_i(x)+1) = -\log \prod_{i=1}^{m} (\lambda_i(x)+1) = -\log \det(A(x)+I),$$

The derivative with respect to A is here

$$\nabla Tr\{\varphi(A)\} = \varphi'(A) = (A+I)^{-1}.$$

3. **Exponential Penalty Function** [16, 11]

$$\varphi(t) = e^{-t} - 1\ .$$

For this function

$$Tr\{\varphi(A(x))\} = Tr\{e^{-A(x)} - I\},$$

the derivative with respect to A is

$$\nabla Tr\{\varphi(A)\} = -e^{-A}.$$

4. **"Mixed Quadratic-Logarithmic" Penalty Function** [6, 7, 29].

For examples $1-3$, the third derivative of φ is very large for certain values of t. This may cause difficulties in using the Newton method for the minimization of the penalty aggregate (7). This motivates the following choice of φ:

$$\varphi_\tau(t) = \begin{cases} \frac{a}{2}t^2 + bt + c, & t \le \tau \\ d\log(t+e) + f, & t > \tau, \end{cases}$$

where $-1 < \tau \le 1$ is a parameter fixing the join point. The coefficients a, b, c, d, e, f are uniquely determined by the requirement that φ is twice differentiable at $t = \tau$, and $\varphi(0) = 0,\ \varphi'(0) = -1,\ \varphi''(0) = 1$. For example, if $0 \le \tau < 1$ then

$$\varphi_\tau(t) = \begin{cases} \frac{1}{2}t^2 - t, & t \le \tau \\ -(1-\tau)^2 \log\left(\frac{1-2\tau+t}{1-\tau}\right) - \tau + \frac{1}{2}\tau^2 \quad , & t > \tau\ . \end{cases}$$

For this choice of φ the second derivative φ'' *is continuous and bounded* $\forall t \in I\!R$. Thus, we are in good position for the Newton minimization of the penalty aggregate (7). We prefer to use this penalty function in our implementation of the PBM algorithm.

5 Dual Analysis of the PBM Algorithm

In this section we present the dual analysis of the PBM algorithm, based on its connection to an appropriate (non-quadratic) *proximal point algorithm*, applied to the dual problem (5). A similar dual interpretation is well known in nonlinear programming for the quadratic augmented Lagrangian method [23], and for nonquadratic augmented Lagrangians [25, 26, 7, 29]. We show that the sequence of the dual function values, corresponding to the sequence of PBM multipliers, is monotonically nondecreasing and is bounded from above by the optimal primal value.

5.1 Conjugate penalty function

The conjugate function of φ_{p_k} is given by Legendre transformation (see e.g. [23])

$$\psi_{p_k}(t) \equiv \varphi^*_{p_k}(t) = \sup_{\tau}(t\tau - \varphi_{p_k}(\tau)). \tag{17}$$

Its derivative is an inverse function with respect to the original derivative

$$\psi'_{p_k} = (\varphi'_{p_k})^{-1}.$$

Below in this subsection we will use $\varphi,\ \psi$ instead of $\varphi_{pk},\ \psi_{pk}$ for simplicity of notation.

The properties $(\varphi 0) - (\varphi 4)$, given in the Section 4, imply the following properties of ψ:

$(\psi 0)$ ψ is a strictly convex twice differentiable function in $(-\infty, 0)$ (by the property $(\varphi 0)$);

$(\psi 1)$ $\psi(-1) = 0$ (since $\psi(-1) = \varphi^*(-1) =: \sup(-t - \varphi(t)) = -\varphi(0) = 0$);

$(\psi 2)$ $\psi'(-1) = 0$ (since $\varphi'(0) = -1$);

$(\psi 3)$ (i) $\lim\limits_{t\to 0^-} \psi'(t) = +\infty$, (ii) $\psi(t) = +\infty$ for $t \geq 0$ (since $\lim_{t\to\infty} \varphi'(t) = 0$)
(iii) $\lim\limits_{t\to -\infty} \psi(t) = +\infty$ (by $\psi 0,\ \psi 1$ and $\psi 2$);

$(\psi 4)$ $\min \psi(t) = \psi(-1) = 0$ (by $\psi 0,\ \psi 1$ and $\psi 2$).

As a specific example, we take the logarithmic-quadratic function

$$\varphi(t) = \begin{cases} -t + \frac{1}{2}\, t^2, & t \leq \frac{1}{2} \\ -\frac{1}{4}\log(2t) - \frac{3}{8}, & t > \frac{1}{2}. \end{cases} \tag{18}$$

Computing its conjugate function, we get

$$\psi(s) = \varphi^*(s) = \begin{cases} \frac{1}{2}\,(s+1)^2, & s \leq -\frac{1}{2} \\ -\frac{1}{4}\log(-2s) + \frac{1}{8}, & -\frac{1}{2} < s < 0. \end{cases}$$

All the properties $(\psi 0) - (\psi 4)$ can be easily verified for this example.

5.2 Connection between the matrix constraint and the sub-differential of the dual function

Suppose that we compute x^k by (11) and update the multiplier V_{k+1} by (12), taking into account (9). Consider the dual function $G(U)$ given in (4).

Proposition 1 *The matrix* $[-A(x^k)]$ *belongs to the sub-differential of the concave dual function* G *at* U_{k+1}

$$-A(x^k) \in \partial G(U_{k+1}) \ . \tag{19}$$

Proof. By the definition of dual function (4),

$$G(U) \leq f(x^k) - \langle A(x^k), U\rangle = f(x^k) - \langle A(x^k), U_{k+1}\rangle - \langle A(x^k), U - U_{k+1}\rangle. \tag{20}$$

It follows from (14) that

$$G(U_{k+1}) = L(x, U_{k+1}) = f(x^k) - \langle A(x^k), U_{k+1}\rangle.$$

Now using (20) we get the following relation:

$$G(U_{k+1}) \geq G(U) + \langle -A(x^k), U_{k+1} - U\rangle,$$

showing that

$$-A(x^k) \in \partial G(U_{k+1}) \ .$$

(See the sub-differential calculus developed in [23] and [24].) □

5.3 Connection of PBM to the proximal-point algorithm

It is seen from (12) that

$$A(x^k) = U_k^{-\frac{1}{2}} (\varphi'_{p_k})^{-1} (-U_k^{-\frac{1}{2}} U_{k+1} U_k^{-\frac{1}{2}}) U_k^{-\frac{1}{2}}.$$

Combining this with (19) yields

$$0 \in \partial G(U_{k+1}) + U_k^{-\frac{1}{2}} (\varphi'_{p_k})^{-1} (-U_k^{-\frac{1}{2}} U_{k+1} U_k^{-\frac{1}{2}}) U_k^{-\frac{1}{2}},$$

which is precisely the necessary and sufficient condition for U_{k+1} to attain the maximum in the iteration of the following *proximal-point* algorithm for solving the dual problem (5)

$$U_{k+1} = \arg\max_U \{G(U) - D_k(U, U_k)\}, \tag{21}$$

where

$$D_k(U, U_k) = Tr\{\varphi^*_{p_k}(-U_k^{-\frac{1}{2}} U U_k^{-\frac{1}{2}})\} \tag{22}$$

denotes a distance-like function $D : S^m_+ \times S^m_+ \to I\!R$. Here $\varphi^*_{p_k}$ is the conjugate function of φ_{p_k}, given in (17). (We make use of the fact that $(\varphi')^{-1} = (\varphi^*)'$.) It should be noted that the maximum is uniquely attained in (21) by both $(\psi 0)$ and $(\psi 3 - i)$.

Properties of the distance-like function D (Below we will use D instead of D_k for simplicity of notation.)

Let $U_k \succ 0$. Then by the properties of $(\psi 1 - \psi 4)$, the distance-like function (22) is strictly convex and twice differentiable with respect to the first argument, and the following properties hold:

$(D1)$ ["distance" property] (i) $D(U, U_k) \geq 0$ and (ii) $D(U, U_k) = 0$ iff $U = U_k$,

$(D2)$ [barrier property] (i) $D(U, U_k) = +\infty$ if U is not positive definite and (ii) $\lim_{\|U_k\| \to \infty} D(U, U_k) = +\infty$,

$(D3)$ for any fixed $\overline{U} \succ 0$ the function $D(\overline{U}, \cdot)$ has bounded level sets,

$(D4)$ for any fixed $\overline{U} \succ 0$ and any sequence of positive definite matrices $\{U_k\}$, $D(\overline{U}, U_k) \to 0$ iff $U_k \to \overline{U}$.

Indeed, (22) shows that the function $D(\overline{U}, \cdot)$ is continuous at $\overline{U}$, with respect to its second argument. The result follows from both this continuity property and $(D1 - ii)$.

5.4 Monotonicity and boundedness of the dual sequence generated by PBM

Proposition 2 *Let $\{(U_k)\}$ be a sequence of multipliers, generated by the PBM algorithm (11 – 13). Then the corresponding sequence of the dual function values $\{G(U_k)\}$ is monotonically nondecreasing and has a limit bounded from above by the optimal primal value.*

Proof. The proof is based on the connection with the proximal-point algorithm. We have by (21)

$$G(U_{k+1}) - D(U_{k+1}, U_k) \geq G(U_k) - D(U_k, U_k) = G(U_k), \quad \text{by } \textit{(D1-ii)} .$$

Hence the sequence $\{G(U_k)\}$ is monotonically nondecreasing:

$$G(U_{k+1}) - G(U_k) \geq D(U_{k+1}, U_k) \geq 0, \quad \text{by } \textit{(D1-i)} .$$

Taking into account the duality theorem (section 3), we see that $\lim_{k\to\infty} G(U_k)$ exists and is at most the optimal primal value. □

6 Computationally Tractable Dual Bounds

In this section[4] we present computationally tractable dual bounds for the problem (SDP) with the linear objective function

$$f(x) = c^T x \tag{23}$$

[4]Results of this section are mainly based on [18, 20]

and additional constraint

$$\|x\| \leq R. \tag{24}$$

Presented bounds are based on the conjugate penalty function and may be efficiently used in situations, where an inexact minimizer of the penalty aggregate is available. In the last part of this section we give some rules that are helpful in computing matrix conjugate function of the penalty term in PBM.

The penalty aggregate (7) or (10) can be represented as:

$$F(x) = c^T x + \Phi(A(x)), \tag{25}$$

where $\Phi : I\!R^{m \times m} \to I\!R$ is C^2 smooth convex function. By the properties ($\varphi 0,\ \varphi 3$)

$$\Phi(A(x)) \leq 0 \quad \text{for} \quad A(x) \succeq 0. \tag{26}$$

From now on we assume that we know the conjugate penalty function given by the Legendre transformation:

$$\Phi^*(B) = \sup_{A \in Dom\Phi} \{\langle A, B \rangle - \Phi(A)\}, \tag{27}$$

i.e. we are able to compute $\Phi^*(B)$ if $B \in Dom\Phi^*$ (if the dual matrix B does not belong DomΦ^*, we set $\Phi^*(B) = +\infty$). The explicit expressions for the conjugate functions of the penalties Φ corresponding to (7) and (10) can be found in the appendix D.

Denote the optimal value of the objective function by

$$f_* \equiv \min_{\|x\| \leq R} \{c^T x : A(x) \succeq 0\}.$$

We start with the following simple observation.

Proposition 3 [Dual bound by the minimum of the aggregate]

The minimum of the aggregate

$$F_* \equiv \min_{\|x\| \leq R} F(x) \tag{28}$$

is a lower bound for f_*.

Proof is an immediate consequence of (26). □

Now we can formulate the basic result of the section.

Theorem 1 [Dual bound by the conjugate penalty function]

Let $B \in Dom\Phi^*$ satisfy the linear equation

$$c + \mathcal{A}^* B = \Delta \ . \tag{29}$$

Then the quantity

$$f_B = \langle A_0, B \rangle - \Phi^*(B) - \|\Delta\| R \tag{30}$$

is a lower bound for the problem (SDP) with (23,24).

Proof. Let $B \in Dom \Phi^*$ satisfy (29). Since Φ^* is the Legendre transformation of Φ, we have

$$\begin{aligned}\Phi(A(x)) \geq \langle A(x), B\rangle - \Phi^*(B) &= \langle A_0 + \mathcal{A}x, B\rangle - \Phi^*(B) \\ &= \langle A_0, B\rangle + (\mathcal{A}^* B)^T x - \Phi^*(B) \\ &= \langle A_0, B\rangle + (\Delta - c)^T x - \Phi^*(B) \quad \text{(by (29))}\end{aligned}$$

or

$$c^T x + \Phi(A(x)) \geq \langle A_0, B\rangle - \Phi^*(B) + \Delta^T x. \tag{31}$$

For $\|x\| \leq R$, one has $\Delta^T x \geq -R\|\Delta\|$. Substituting this into (31) and using (25) and (30), we have

$$F(x) \geq f_B \qquad \forall x \in I\!R^n.$$

Hence $F_* \geq f_B$ and by Proposition 3, $f_* \geq f_B$. □

Connection between the bounds (28) and (29–30). If a given point $\widehat{x}$ is an exact minimizer of the penapty aggregate, then (28) gives us the dual bound F_*. The same result can be obtained by the bound (30), if we choose

$$B = \Phi'(A(\widehat{x}))\ , \tag{32}$$

that satisfies equation (29) under $\Delta = 0$. In the case when $\widehat{x}$ is not the exact minimizer, the value of Δ in (29) is equal to the gradient of the aggregate in $\widehat{x}$. Thus the last term in (30) may make our dual estimate worse.

6.1 Computing a dual bound by non-exact minimizer of the aggregate

To improve somehow the situation in the case, when $\widehat{x}$ is not the exact minimizer of the aggregate, we project B, obtained by (32), onto the plane

$$c + \mathcal{A}^* B = 0.$$

A reasonable way is to set

$$B^+ = B - \Phi''(A)\mathcal{A}[F''(x)]^{-1}\Delta,$$

where $F''(x)$ is a Hessian of the aggregate (25), so that

$$c + \mathcal{A}^* B^+ = c + \mathcal{A}^* B - \mathcal{A}^*\Phi''(A)\mathcal{A}[F''(x)]^{-1}(c + \mathcal{A}^* B) = c + \mathcal{A}^* B - c - \mathcal{A}^* B = 0. \tag{33}$$

In this case, by Theorem 1, the dual bound is

$$f_{B^+} = \langle A_0, B^+\rangle - \Phi^*(B^+).$$

In practice, because of limited accuracy of computations, equality (33) will be satisfied within an error Δ^+, where

$$\Delta^+ = c + \mathcal{A}^* B^+,$$

so that by Theorem 1 the dual bound will be

$$f_B^+ = \langle A_0, B^+\rangle - \Phi^*(B^+) - \|\Delta^+\| R.$$

7 Implementation of the PBM algorithm

Our implementation has been written in C++ programming language using LAPACK and BLAS linear algebra libraries. We also use fast routines [12] for computing the operators $\mathcal{A}$ and $\mathcal{A}^*$ in various application domains.

The program is built in the way that any penalty function $\varphi(\cdot)$ with the properties $(\varphi 0) - (\varphi 4)$ may be easily incorporated. In our experience the *logarithmic-quadratic* function (18) gives the best results, and we used it in computational experiments presented here.

As a tool for unconstrained minimization, we use the Newton method with *frozen Hessian*, i.e. we solve the Newton system by Cholesky decomposition, and then produce a few steps by substitution the new gradients into the decomposed system. If the number of such steps exceeds N^{tol} (for example, $N^{tol} = 10$), we evaluate a new Hessian matrix and produce the decomposition again.

7.1 Algorithm

We will use the following notation:

$$\begin{array}{lll} \Phi(A(x)) \equiv Tr\{\varphi_p(VA(x)V)\}, & \Phi' \equiv \Phi'(A(x)), & \Phi'' \equiv \Phi''(A(x)), \\ F \equiv F(x,V,p) = c^T x + \Phi(A(x)), & F' \equiv F'_x(x,V,p), & F'' \equiv F''_{xx}(x,V,p). \end{array}$$

The formal representation of the algorithm is the following:

```
given  $an\ arbitrary\ initial\ point\ x\ (not\ necessarily\ feasible)$
1. setup  $i := N^{tol}$, $p := p_0$, and $V := V_0$
repeat  [Outer iteration]
    repeat  [Loop of unconstrained minimization]
        2. if $(i \geq N^{tol})$  Compute $F''$ and $[F'']^{-1}$
        for  i=1 to $N^{tol}$  [Newton iteration with frozen Hessian]
            3. Compute $F$ and $F'$
            4. Compute the Newton direction $d = -[F'']^{-1}F'$
            5. Linesearch: $\bar{t} \approx \arg\min_t F(x + td, V, p)$,  $x := x + \bar{t}d$
            6. Compute the stopping criterion: $\delta = (F')^T[F'']^{-1}F'$
            if $(\delta < \gamma p)$ break  $(0 < \gamma < 1)$
        end for
    until  $\delta < \gamma p$
    7. Compute the dual bound:
            $B = \Phi'$ and $\Delta = c + \mathcal{A}^* B$
            $B^+ = B - \Phi'' \mathcal{A}[F'']^{-1}\Delta$ and $\Delta^+ = c + \mathcal{A}^* B^+$
            $f_{B^+} = \langle A_0, B^+ \rangle - \Phi^*(B^+) - \|\Delta^+\| R$
    8. Update the multiplier: $V := [-V\varphi'_p(VA(x)V)V]^{\frac{1}{2}}$
    9. Update the penalty parameter: $p := \pi p$  $(0 < \pi \leq 1)$
until  $(c^T x - f_{B^+})/\max\{1, |c^T x|\} < \epsilon$
```

7.2 Linesearch

In item 5 of the algorithm, we use safeguarded cubic linesearch with an early stopping by the Goldstein rule

$$\left|\frac{F(x+td,V,p)-F(x,V,p)}{td^TF'}-\frac{1}{2}\right|<\epsilon_{gold}\ ,$$

where the choise of the constant was $\epsilon_{gold}=0.3$. Typically the linesearch converges in 1–3 iterations. We explain such a good behaveor by the boundedness of the third derivative of the quadratic-logarithmic penalty function.

7.3 Diagonal preconditioning of the Newton system

In item 4 of the algorithm we need to solve the system

$$Hd=g, \tag{34}$$

where $H\equiv F''$ and $g\equiv -F'$ are an $n\times n$ symmetric positive definite matrix and n-dimensional vector. In practical situations this system may be *ill-conditioned.* That's why we use the following scaling procedure. First we compute

$$\overline{H}=D^{-1}HD^{-1},\ g=D^{-1}g, \tag{35}$$

where $D^{-1}=diag\{H_{11}^{-\frac{1}{2}},...,H_{nn}^{-\frac{1}{2}}\}$. Taking into account (35), one may rewrite (34) as

$$D\overline{H}Dd=D\overline{g}$$

or

$$\overline{H}\ \overline{d}=\overline{g}, \tag{36}$$

where $\overline{d}=Dd$. To solve (36), we compute the Cholesky factorization

$$\overline{H}=LL^T$$

of the matrix $\overline{H}$ and obtain $\overline{d}$ by back-substitution. Now it is not difficult to compute

$$d=D^{-1}\overline{d}.$$

7.4 Numerical Results

The PBM algorithm has been tested on a wide variety of applications. Here we report numerical results for the problems from

- Robust Control (Appendix E);
- Robust Truss Topology Design (Appendix F, [5]);
- Free Material Design [3].

All presented problems are of type (SDP) with the linear objective and the additional constraint

$$\|x\|\le 10^9.$$

The CPU times are given for the IBM RS/6000 workstation.

Table 1 demonstrates a small number of Newton steps for ill-conditioned problems of robust control.

Table 1
Experimental Results for Robust Control
(by the PBM algorithm)

Problem	n	m	# Newton steps	Relative error	CPU time
8	73	40	16	8×10^{-4}	14”
12	157	62	12	4×10^{-4}	46”
16	273	82	12	6×10^{-4}	2’48”
20	421	100	12	3×10^{-4}	7’11”
24	601	118	9	2×10^{-4}	14’22”
28	813	136	15	2×10^{-4}	50’00”
32	1057	156	8	2×10^{-4}	1h00’00”

Table 2 demonstrates a high accuracy of the PBM algorithm. It should be noted that for problems of robust truss topology design, the algorithm was stopped by the guaranteed relative error

$$(c^T x - f_{B^+})/max\{1, |c^T x|\} < \epsilon$$

with $\epsilon = 1 \times 10^{-7}$.

Table 2
Experimental Results for Robust Truss Topology Design

Problem	n	m	# Newton steps	Relative error	CPU time
1	6	13	8	1×10^{-8}	2”
2	8	19	7	3×10^{-8}	3”
3	86	301	29	3×10^{-8}	2’22”
4	172	451	36	3×10^{-8}	8’10”
5	171	451	36	6×10^{-8}	7’30”

Table 3 demonstrates both a high accuracy of the PBM algorithm and a fast convergence to the solution. Note that the problems of free material design has been solved by sparse linear algebra techniques. Here the guaranteed stopping criterion was $\epsilon = 1 \times 10^{-5}$.

Table 3
Experimental Results for Free Material Design

Problem	n	m	# Newton steps	Relative error	CPU time
1	512	1921	45	1×10^{-6}	1’40”
2	800	3041	47	4×10^{-6}	4’20”
3	1568	6049	52	2×10^{-6}	9’00”
4	3869	6469	57	1×10^{-5}	26’00”
5	6145	11905	60	1×10^{-5}	50’00”

We also solved the same free material design problems with the projective algorithm [12]. Implementation of this method for the MATLAB *LMI Control Toolbox* [13] supports block-diagonal matrix structure. Comparison of the two methods is presented at the Table 4. As we can see, the number of Newton steps is significantly larger for the projective algorithm (it failed to solve the two largest problems).

Table 4
Comparison Between the PBM and Projective Algorithm
for Free Material Design

Problem	# Newton steps, PBM	# Newton steps, Projective alg.
1	45	70
2	47	84
3	52	123
4	57	*
5	60	*

* – failed to get 4 decimal digits of accuracy

Table 5 indicates the distribution of the CPU time (in %) between the various linear algebra tasks. From this table one may see the high cost of the assembling the Hessian.

Table 5
Distribution of CPU time (in %)

Problem	Assembling of Hessian	Cholesky factorization	Other
8	74	3	23
12	77	4	19
16	83	5	12
20	87	6	7
24	88	7	5
28	89	7	4
32	90	8	2

7.5 Some conclusions of the computational experiments

In many cases after achieving an accuracy of 3-4 digits, the PBM algorithm almost does not require new Hessian computations. Each new unconstrained minimization takes typically only one or few gradient steps in the metric of the old Hessian. This remarkable property may be explained as follows. In the neighborhood of solution the multiplier changes only slightly by the update formula. That's why the aggregate also changes only slightly and can be easily minimized again.

The total number of Newton steps is almost independent of the problem size and typically ranges between 10 and 15 for well-conditioned problems of robust control, and between 40 and 60 for the problems of robust truss topology design and free material design.

The PBM algorithm is rather fast and gives a high guaranteed accuracy (of 5–7 digits) for the problems of robust truss topology design and free material design.

Acknowledgement We are deeply indebted to Arkadi Nemirovski, who supported and essentially influenced this work. Also we wish to express many thanks to Aharon Ben-Tal, Dmitri Falikman, Roman Polyak, and Brian Smith for helpful discussions, from which we benefited in representing the results.

Appendix A. Optimality Conditions and the Lagrangian Saddle Set

Theorem 2 *A vector x^* is a solution of the problem (SDP) iff there exists a nonzero matrix $U^* \in S_+^m$ such that*

$$\begin{aligned} \nabla f(x^*) - \mathcal{A}^* U^* &= 0 \quad \textit{(KKT condition)}, && (37)\\ A(x^*) &\succeq 0 \quad \textit{(feasibility)}, && (38)\\ \langle A(x^*), U^* \rangle &= 0 \quad \textit{(complementarity slackness)}. && (39) \end{aligned}$$

Proof. Follows directly from [19]. □

Theorem 3 *A vector x^* is a solution of the problem (SDP) iff there exists a nonzero matrix $U^* \in S_+^m$ such that the pair (x^*, U^*) is a saddle point of the Lagrangian (3):*

$$L(x^*, U) \le L(x^*, U^*) \le L(x, U^*) \quad \forall x \in I\!R^n,\ \forall U \in S_+^m. \tag{40}$$

Proof. Suppose that x^* is the solution of the problem (SDP). Then by the Theorem 2, the conditions (37) – (39) hold. By (37)

$$\nabla_x L(x^*, U^*) = 0,$$

so that by convexity of the Lagrangian with respect to x

$$x^* \in \arg\min_x L(x, U^*) \tag{41}$$

Furthermore, by (39) and (38)

$$f(x^*) \ - \langle U^*, A(x^*) \rangle \ = \ f(x^*) \ \ge \ f(x^*) \ - \ \langle U, A(x^*) \rangle \quad \forall U \succeq 0. \tag{42}$$

Hence by (41) and (42), the pair (x^*, U^*) is a saddle point.

Conversely, suppose that (x^*, U^*) is a saddle point of the Lagrangian. Then (41) holds. Hence (37) holds as well. Furthermore, from (40) we have

$$f(x^*) \ - \langle U^*, A(x^*) \rangle \ \ge \ f(x^*) \ - \ \langle U, A(x^*) \rangle \quad \forall U \succeq 0. \tag{43}$$

This implies that we must have $A(x^*) \succeq 0$, otherwise (43) can be violated by appropriate choice of matrix U such that $\langle U, A(x^*) \rangle \ < \ 0$ and multiplying it by an infinitely large scalar. Now taking $U = 0$ in (43), we obtain $\langle U^*, A(x^*) \rangle \ \le \ 0$. Noting that $A(x^*) \succeq 0$ and $U^* \succeq 0$ imply that $\langle U^*, A(x^*) \rangle \ \ge \ 0$, we must have $\langle U^*, A(x^*) \rangle \ = \ 0$. Hence, conditions (37) – (39) hold, and by Theorem 2, the vector x^* is an optimal solution for the problem (SDP). □

Appendix B. Gradient of the matrix function $\Phi(A) = Tr\{\varphi(A)\}$

Let $A \in S^m$ and assume that the scalar function φ has the following expansion:

$$\varphi(t) = \sum_{i=0}^{k} c_i t^i,$$

so that

$$\Phi(A) = Tr\{\sum_{i=0}^{k} c_i A^i\}.$$

Now let $H \in S^m$ be some matrix, which is small in norm. Then

$$\begin{aligned} \Phi(A+H) - \Phi(A) &= Tr\{\sum_{i=0}^{k} c_i[(A+H)^i - A^i]\} \\ &= \sum_{i=1}^{k} c_i \sum_{j=0}^{i-1} Tr\{A^j H A^{i-1-j}\} + o(\|H\|) \\ &= \sum_{i=1}^{k} ic_i \; Tr\{A^{i-1}H\} + o(\|H\|) \\ &= Tr\{\varphi'(A)H\} + o(\|H\|) \\ &= \langle \varphi'(A), H\rangle \; + o(\|H\|) \;, \end{aligned} \tag{44}$$

where φ' is a derivative of the scalar function φ. On the other hand, by definition of a gradient

$$\Phi(A+H) - \Phi(A) = \langle \nabla\Phi(A), H\rangle + o(\|H\|). \tag{45}$$

So, by (44) and (45)

$$\nabla\Phi(A) = \varphi'(A) \; .$$

It is easy to get corresponding gradient with respect to x

$$\nabla_x \Phi(A(x)) = \mathcal{A}^* \varphi'(A(x)) \; .$$

Appendix C. The second derivative of the matrix function $\Phi(A) = Tr\{\varphi(A)\}$

Denote the gradient of $\Phi(A)$

$$\Psi(A) \equiv \nabla\Phi(A) = \varphi'(A), \quad A \in S^m$$

and let the derivative of the scalar function φ have an expansion

$$\varphi'(t) = \sum_i \alpha_i t^i, \tag{46}$$

so that

$$\Psi(A) = \sum_i \alpha_i A^i. \tag{47}$$

Let also the eigenvalue decomposition of A be

$$A = S\Lambda S^T,$$

where S is an orthogonal matrix with eigenvectors in its columns, and Λ is a diagonal matrix of eigenvalues $\lambda_1, .., \lambda_m$.

Theorem 4 *(based on [18])*
The differential of $\Psi(A)$ in the direction $H \in S^m$ is

$$D\Psi(A)[H] = S(Q \circ \widetilde{H})S^T, \tag{48}$$

where $\widetilde{H} = S^T H S$, " $\circ$" denotes an element-by-element multiplication, and the matrix $Q \in S^m$ is given by

$$Q_{rs} = \begin{cases} \frac{\varphi'(\lambda_r)-\varphi'(\lambda_s)}{\lambda_r-\lambda_s}, & \lambda_r \neq \lambda_s, \\ \varphi''(\lambda_r), & \textit{otherwise.} \end{cases} \qquad r, s = 1, .., m. \tag{49}$$

Proof We have by (47)

$$\Psi(A+H) - \Psi(A) = \sum_i \alpha_i((A+H)^i - A^i). \tag{50}$$

Consider the term

$$\begin{aligned} (A+H)^i - A^i &\approx \sum_{k=1}^{i} A^{i-k} H A^{k-1} \\ &= S(\sum_{k=1}^{i} \Lambda^{i-k} \widetilde{H} \Lambda^{k-1})S^T = S G_i S^T, \end{aligned} \tag{51}$$

where G_i is a matrix from S^m with the elements:

$$(G_i)_{rs} = \widetilde{H}_{rs} \sum_{k=1}^{i} \lambda_r^{i-k} \lambda_s^{k-1}. \tag{52}$$

We can observe, that

$$\sum_{k=1}^{i} \lambda_r^{i-k} \lambda_s^{k-1} = \frac{\lambda_r^i}{\lambda_s} \sum_{k=1}^{i} \Big(\frac{\lambda_r}{\lambda_s}\Big)^k = \begin{cases} \frac{\lambda_r^i - \lambda_s^i}{\lambda_r - \lambda_s} & \lambda_r \neq \lambda_s \\ i\lambda_r^{i-1}, & \text{otherwise} \end{cases} \tag{53}$$

Substituting (51, 52, 53) into (50), and using (46) we get (48, 49). □

Corollary 1 *The i-th column of the Hessian matrix $\nabla_{xx}\Phi(A(x))$ is given by*

$$\Big[\nabla_{xx}\Phi(A(x))\Big]_i = \mathcal{A}^*\Big[S\Big(Q \circ (S^T A_i S)\Big)S^T\Big], \tag{54}$$

where A_i is the i-th data matrix, which defines the linear operator $\mathcal{A}$ in (1), and $\mathcal{A}^*$ is the corresponding conjugate operator, given by (2).

Corollary 2 *Let the matrix P be element-wise square root of the matrix Q and denote*

$$R_i \equiv P \circ (S^T A_i S), \quad i = 1, ..., n.$$

Then the elements of the Hessian matrix are given by

$$\Big[\nabla_{xx}\Phi(A(x))\Big]_{ij} = \langle R_i, R_j\rangle, \quad i, j = 1, .., n. \tag{55}$$

Corollary 3 *Let R be a $n \times m^2$ matrix, which* i-th *row contains all the elements of the matrix R_i, taken row-by-row. Then it's easy to see from (55), that the Hessian*

$$\nabla_{xx}\Phi(A(x)) = RR^T$$

is a positive-semidefinite matrix, hence $\Phi(A(x))$ is a convex function.

We should note, that in many practical situations fast computation of the conjugate operator $\mathcal{A}^*$ may be available. That's why the use of the formula (54) can be preferable to the formula (55).

Appendix D. Computing conjugate functions of matrices

In the following theorems we obtain explicit expressions for the conjugate functions of the penalties Φ corresponding to (7) and (10).

Theorem 5 [The conjugate function of $p\Phi(p^{-1}A)$]

Denote $\Phi_p(A) \equiv p\Phi(p^{-1}A)$, where p is a positive scalar, and let $\Phi_p^*(B)$ be the Legendre transformation of $\Phi_p(A)$. Then

$$\Phi_p^*(B) = p\Phi^*(B).$$

Proof. By the definition of the conjugate function

$$\Phi^*(B) \quad = \sup_{p^{-1}A \in Dom\ \Phi} \{\langle B, p^{-1}A\rangle - \Phi(p^{-1}A)\}.$$

Hence we see that

$$\Phi_p^*(B) = \sup_{A \in Dom\ \Phi_p} \{\langle B, A\rangle - p\Phi(p^{-1}A)\} \ = p \sup_{p^{-1}A \in Dom\ \Phi} \{\langle B, p^{-1}A\rangle - \Phi(p^{-1}A)\} \ = p\Phi^*(B).$$

□

Theorem 6 [The conjugate function of $\Phi(VAV)$]

Denote $\Phi_V(A) \equiv \Phi(VAV)$ and let ${\Phi^*}_V(B)$ be the Legendre transformation of $\Phi_V(A)$. Then

$${\Phi^*}_V(B) = \Phi^*(V^{-1}BV^{-1}).$$

Proof. By the definition of the conjugate function

$${\Phi^*}_V(B) \quad = \sup_{A \in Dom \Phi_V} \{\langle A, B\rangle - \Phi_V(A)\}.$$

Taking into account that

$$\langle A, B\rangle \equiv Tr\{AB\} = Tr\{VAVV^{-1}BV^{-1}\} = \langle VAV, V^{-1}BV^{-1}\rangle,$$

we obtain

$${\Phi^*}_V(B) = \sup_{VAV \in \mathrm{Dom}\,\Phi} \{\langle VAV, V^{-1}BV^{-1}\rangle - \Phi(VAV)\} \ = \Phi^*(V^{-1}BV^{-1}).$$

□

Theorem 7 [The conjugate function of $Tr\{\varphi(A)\}$]

Let φ^* be the Legendre transformation of a scalar function φ and let

$$\Phi(A) = Tr\{\varphi(A)\}.$$

Then

$$\Phi^*(B) = Tr\{\varphi^*(B)\}. \tag{56}$$

Proof. Let $\widehat{A}$ be an optimal point in (27), i.e.

$$\Phi^*(B) = \langle \widehat{A}, B \rangle - \Phi(\widehat{A}) = Tr\{\widehat{A}B - \varphi(\widehat{A})\} \ . \tag{57}$$

By optimality condition for (27) and Appendix B

$$B = \Phi'(\widehat{A}) = \varphi'(\widehat{A}).$$

Hence B and $\widehat{A}$ can be diagonalized in the same orthobasis. Computing $\varphi^*(B)$ in this basis, we immediately get

$$\varphi^*(B) = \widehat{A}B - \varphi(\widehat{A}) \ .$$

Comparing this with (57), we get finally (56). □

Note that a similar result for Hermitian matrices may be found in [17].

Appendix E. SDP in Control and Systems Theory

An extremely powerful source of semidefinite problems comes from modern control and systems theory. Boyd, El Ghaoui, Feron, and Balakrishnan catalog many examples in [9]. We will describe one simple example here [27].

Consider the differential inclusion

$$\frac{dx}{dt} \in Co\{A_1,\ \dots\ , A_L\}x(t), \tag{58}$$

where $x(t) \in \mathbb{R}^n$ and the matrices $A_1,\ \dots\ , A_L$ are given, and $Co\{A_1,\ \dots\ , A_L\}$ denotes the convex hull of $A_1,\ \dots\ , A_L$. We seek an ellipsoidal invariant set, i.e., an ellipsoid Ω such that for any x that satisfies (58) $x(T) \in \Omega$ implies $x(t) \in \Omega$ for all $t \geq T$. The existence of such an ellipsoid implies, for example, that all solutions of the differential inclusion (58) are bounded.

The ellipsoid $\Omega = \{x : x^T Ex \leq 1\}$, where $E = E^T \succ 0$, is invariant iff the function $g(t) = x(t)^T Ex(t)$ is non-increasing for any solution x of (58). In this case we say that g is a quadratic Lyapunov function that proves stability of the differential inclusion (58).

Thus, Ω is invariant iff

$$\frac{d}{dt}g(x(t)) = x(t)^T(A(t)^T E + EA(t))x(t) \leq 0,$$

for any $x(t) \in \mathbb{R}^n$ and $A(t) \in Co\{A_1,\ \dots\ , A_L\}$. This is equivalent to $-A^T E - EA \succeq 0$ for all $A \in Co\{A_1,\ \dots\ , A_L\}$, which in turn is equivalent to the condition

$$-{A_k}^T E - EA_k \succeq 0,\ \ k = 1, ..., L.$$

These are LMI constraints in the matrix E, considered as the variable.

To find an invariant ellipsoid for the differential inclusion (58) (or verify that none exists), we need to solve the feasibility problem

$$E \succ 0,\quad -{A_k}^T E - EA_k \succeq 0,\ \ k = 1, ..., L \tag{59}$$

for the matrix variable E. Several standard methods can be used to convert this feasibility problem into a semidefinite program that has an obvious initial feasible point. For instance, we can solve the following semidefinite problem:

$$Minimize\ \ t$$

subject to:

$$-{A_k}^T E - EA_k \succeq 0,\ \ k = 1,\ \dots\ , L,$$

$$tI + E \succeq 0,\quad I + E \succeq 0.$$

The last constraint is added, without loss of generality, to normalize the otherwise homogeneous problem. This semidefinite problem can be initialized with $E = 0$, $t = 1$ and then solved. The optimum value of t is negative iff (59) is feasible.

Appendix F. SDP in Structural Optimization (Truss Topology Design)

Ben-Tal and Bendsøe in [2] consider the following problem from truss topology design. A structure of n linear elastic bars connect a set of N nodes in the truss (the maximum number of potential bars in the truss is $\frac{1}{2}N(N-1)$). The task is to size the bars, i.e., determine appropriate cross-sectional areas for the bars. In the simplest version of the problem we consider one fixed set of externally applied nodal forces $s_j,\ j = 1, ..., m$, where m is the number of analysis variables $m = 2N$ ($2D$-trusses) or $m = 3N$ ($3D$-trusses) (more complicated versions consider multiple loading scenarios [5]). The m-dimensional vector of the displacements of the nodes will be denoted d. The objective is the elastic stored energy $s^T d$, which is a measure of the inverse of the stiffness of the structure. We also need to take into account constraint on the total volume of the truss v.

The relation between s and d is linear: $(\mathcal{A}x)d = s$, where the matrix $\mathcal{A}x$ is called the stiffness matrix. This matrix is given in terms of matrices A_i, which are all symmetric positive semidefinite $m \times m$ matrices. Each A_i contains information on the geometry of the connection of node i to the other nodes. The optimization problem then becomes

$$\begin{array}{rl} \text{Minimize} & s^T d \\ \\ \text{Subject to:} & (\mathcal{A}x)d = s, \\ & \displaystyle\sum_{i=1}^{n} x_i = v, \\ & x_i \geq 0, \quad i = 1, ..., n, \end{array}$$

where x is a vector of the bar volumes. For simplicity, we assume that $\mathcal{A}x \succ 0$ for all positive values of x_i. We can then eliminate d and write

$$\begin{array}{rl} \text{Minimize} & s^T (\mathcal{A}x)^{-1} s \\ \\ \text{Subject to:} & \displaystyle\sum_{i=1}^{n} x_i = v, \\ & x_i \geq 0, \quad i = 1, ..., n, \end{array}$$

or

$$\begin{array}{rl} \text{Minimize} & t \\ \\ \text{Subject to:} & \begin{pmatrix} t & s^T \\ s & \mathcal{A}x \end{pmatrix} \succeq 0, \\ & \displaystyle\sum_{i=1}^{n} x_i = v, \\ & x_i \geq 0, \quad i = 1, ..., n. \end{array}$$

Robust formulation of the truss topology design problem can be found in [5]. It is considerably more complicated than the standard one presented here.

References

[1] F. Alizadeh, *Interior Point Methods in semidefinite programming with Applications to Combinatorial Optimization*, SIAM J. Optim. 5 (1995) 13-51.

[2] A. Ben-Tal and M.P. Bendsøe, *A New Method for Optimal Truss Topology Design*, SIAM J. Optim. 3 (1993) 322-358.

[3] A. Ben-Tal, M. Kocvara, A. Nemirovski and J. Zowe *Free Material Design via semidefinite programming, Multi-Load Case with Contact Conditions*, Optimization Laboratory, Faculty of Industrial Engineering and Management, Technion - Israel Institute of Technology (Haifa, Israel, 1997).

[4] A. Ben-Tal and A. Nemirovski, *Convex Optimization in Engineering* Faculty of Industrial Engineering and Management, Technion (Haifa, Israel, 1998).

[5] A. Ben-Tal and A. Nemirovski, *Robust Truss Topology Design via Semidefinite programming*, SIAM J. Optim. 7 (1997), 991-1016.

[6] A. Ben-Tal, I. Yuzefovich, and M. Zibulevsky, *Penalty/Barrier Multiplier Methods for Min-max and Constrained Smooth Convex Programs*, Research Report 9, Optimization Laboratory, Faculty of Industrial Engineering and Management, Technion (Haifa, Israel, 1992).

[7] A. Ben-Tal and M. Zibulevsky, *Penalty/Barrier Multiplier Methods for Convex Programming Problems*, SIAM J. Optim. 7 (1997) 347-366.

[8] S. Boyd and L. El Ghaoui, *Method of Centers for Minimizing Generalized Eigenvalues*, Linear Algebra Appl. 188 (1993) 63-111.

[9] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, SIAM Studies in Applied Mathematics (SIAM, Philadelphia, PA, 1994).

[10] Breitfeld, M.G. and Shanno, D.F. (1994). "A Globally Convergent Penalty-Barrier Algorithm for Nonlinear Programming and its Computational Performance", Rutcor Research Report, Rutgers University, New Jersey.

[11] Doljansky, M. and Teboulle, M. *An Interior Proximal Algorithm and the Exponential Multiplier Method for Semidefinite Programming*, SIAM J. Optim. Vol 9, Num 1 (1998) 1-13.

[12] P. Gahinet and A. Nemirovski, *The Projective Method for Solving Linear Matrix Inequalities*, Math. Programming, Series B 77 (1997) 163-190.

[13] P. Gahinet, A. Nemirovski, A.J. Laub, and M. Chilali, *LMI Control Toolbox*, The MathWorks Inc. (1995).

[14] A.N. Iusem, B. Svaiter, and M. Teboulle, *Entropy-Like Proximal Methods in Convex Programming*, Math. Oper. Res. 19 (1994) 790-814.

[15] F. Jarre, *An Interior-Point Method for Minimizing the Maximum Eigenvalue of a Linear Combination of Matrices*, SIAM J. Control Optim. 31 (1993) 1360-1377.

[16] B.W Kort and D.P. Bertsekas, D. P. Multiplier Methods for Convex Programming, *Proc 1073 IEEE Conf. Decision Control*, (San-Diego, Calif., 428–432, 1973)

[17] A.S. Lewis, *Convex Analysis on the Hermitian Matrices*, SIAM J. Optim. 6 (1996) 164-177.

[18] A. Nemirovski, personal communication.

[19] Yu. Nesterov and A. Nemirovski, *Interior Point Polynomial Algorithms in Convex Programming: Theory and Applications*, SIAM Studies in Applied Mathematics (SIAM, Philadelphia, PA, 1994).

[20] Yu. Nesterov and A. Nemirovski, *Multi-Parameter Surfaces of Analytic Centers and long-step path-following interior point methods*, Research Report 2, Optimization Laboratory, Faculty of Industrial Engineering and Management, Technion - Israel Institute of Technology (Haifa, Israel, 1994).

[21] M.L. Overton and R.S. Womersley, *Optimality Conditions and Duality Theory for Minimizing sums of the largest eigenvalues of symmetric matrices*, Math. Programming, Series B 62 (1993) 321-357.

[22] R. Polyak, *Modified Barrier Functions: Theory and Methods*, Math. Programming 54 (1992) 177-222.

[23] R.T. Rockafellar, *Convex Analysis*, Princeton University Press (Princeton, NJ, 1970).

[24] N.Z. Shor, *Minimization Methods for Nondifferentiable Functions* (Springer, Berlin, 1985).

[25] M. Teboulle, Entropic Proximal Mappings with Applications to Nonlinear Programming, *Mathematics of Operations Research* **17** (1992).

[26] P. Tseng and D. Bertsekas, *On the Convergence of the Exponential Multiplier Method for Convex Programming*, Math. Programming 60 (1993) 1-19.

[27] L. Vandenberghe and S. Boyd, *Positive-Definite Programming*, in: J.R. Birge and K.G. Murty, eds., Mathematical Programming: State of the Art 1994 (University of Michigan, 1994).

[28] M. Zibulevsky, *New Penalty/Barrier and Lagrange Multiplier Approach for Semidefinite Programming*, Research Report 5, Optimization Laboratory, Faculty of Industrial Engineering and Management, Technion - Israel Institute of Technology (Haifa, Israel, 1995). http://iew3.technion.ac.il:8080/˜mcib/

[29] M. Zibulevsky, *Penalty/Barrier Multiplier Methods for Large-Scale Nonlinear and Semidefinite Programming*, Ph.D. Thesis (Technion, Haifa, Israel, 1996). http://iew3.technion.ac.il:8080/˜mcib/