Nonlinear Eigenvector Methods for Convex Minimization Over the Numerical Range

Ding Lu*

December 20, 2018

Abstract

We consider the optimization problem in which a continuous convex function is to be minimized over the joint numerical range of two Hermitian matrices. When the defining matrices are of large size, solving such problems by convex optimization can be computationally very expensive. The goal of this paper is to present a novel nonlinear eigenvector method to accelerate computation. We will show that the global minimizer corresponds to a solution of a nonlinear eigenvalue problem with eigenvector nonlinearity (NEPv). The special structure of this NEPv allows for an efficient sequential subspace searching algorithm, which is a nonlinear analog to NEPv the commonly applied locally optimal conjugate gradient descent methods for Hermitian linear eigenvalue problems. The global convergence of our new algorithm to an eigenvector can be proven. Implementation details such as block iteration and preconditioning will be discussed. Numerical examples, with applications in computing the coercivity constant of boundary integral operators and solving multicast beamforming problems, show the effectiveness of our approach.

Contents

*Department of Mathematics, University of Geneva, CH-1211 Geneva, Switzerland (Ding.Lu@unige.ch).

1 Introduction

In this paper we consider the following convex minimization problem over the numerical range

$$\min_{y\in W(A,B)} F(y) \quad \text{and} \quad W(A,B) = \left\{ \left[x^H Ax, x^H Bx\right]^T : x \in \mathbb{C}^n,\ \|x\|_2 = 1 \right\} \subset \mathbb{R}^2, \tag{1}$$

where $F : \mathbb{R}^2 \to \mathbb{R}$ is a continuous convex function, A and B are n-by-n Hermitian matrices. The set $W(A, B)$, known as the joint numerical range [1] of the matrix pair (A, B), is a convex region in $\mathbb{R}^2$ due to the famous Toeplitz-Hausdorff theorem.

A number of problems can be written into the form of (2). Take F to be a linear function, then standard Rayleigh quotient minimization problem becomes a trivial special case in this framework, e.g., for $F(y) = y_1$ the minimzer corresponds to the smallest Rayleigh quotient $(x^H Ax)/(x^H x)$ for $x \neq 0$. For more general cases, a famous example is the Crawford number computation problem, where the objective function is given by the 2–norm $F(y) = \|y\|_2$. This problem has been studied since 1970s [2], and is of particular interest in eigenvalue sensitivity analysis [2, 3] and the study of the coercivity constant for boundary operators [4]. Another example, where the objective function is also non-differentiable, is the max-ratio minimization problem with $F(y) = \max\{y_1, y_2\}$. Such a problem arises in a class of homogeneous quadratic minimization [5], which finds applications in multicast beamforming problems in signal processing, see, e.g., [6].

Most of the existing methods for solving (1) are based on convex optimization approaches. As a remarkable property, the boundary of the numerical range $W(A, B)$ can be approximated by sampling its supporting hyperplanes, each computed by solving a Hermitian eigenvalue problem of size n; see, e.g., [7]. The minimizer of (1) can then be estimated on this approximated region by standard convex optimization [8]. For some special functions of F, this approach leads to reformulations of the convex problem (1) as eigenvalue optimization problems; see, e.g., [9] for the Crawford number computation and [5] for the max-ratio minimization. Such reformulations are, however, not available for general convex functions F. And in any case, including eigenvalue optimization, repeated Hermitian eigenvalue evaluations are typically required for the computation. This can be prohibitively expensive for problems with large coefficient matrices, hence necessitating the use of iterative methods as presented in the current paper for acceleration.

The main contribution of this work is to present a novel nonlinear eigenvector method for solving (1). Our approach is based on the reformulation of the convex optimization (1) as the following composite minimization problem with a pair of Rayleigh quotients

$$\min_{x\in\mathbb{C}^n\setminus\{0\}} \mathcal{F}(x) := F\left(\frac{x^H Ax}{x^H x}, \frac{x^H Bx}{x^H x} \right). \tag{2}$$

We will show that the global minimizer of this problem is a solution to a *nonlinear eigenvalue problem* with eigenvector nonlinearity (NEPv), for which a sequential subspace searching algorithm can be applied for efficient computation. Such an algorithm only involves matrix-vector multiplication in each iteration, and allows for block implementation, both are desirable for large scale problems. Moreover, the global convergence to an eigenvector can be proven. Issues such as preconditioning and how to handle non-smoothness in F will also be discussed.

Algorithmically, our sequential subspace methods can be viewed as an extension to NEPv the Locally Optimal Block Precondition Conjugate Gradient (LOBPCG) method, which is commonly applied for Hermitian eigenvalue problems; see, e.g., [10, 11, 12, 13]. In our nonlinear version, a reduced-sized NEPv obtained by subspace projection is solved in each iteration, and the searching subspace is updated by gradient direction. According to numerical experiments, this algorithms is nearly as fast as LOBPCG in terms of the number of iterations, whereas the latter can only solve a linear eigenvalue problem, hence is not applicable here. Such an excellent convergence behavior seems to imply that the considered NEPv is solved at a cost similar to a linear eigenvalue problem of equal size.

The proposed algorithms are intended for general convex objective functions, but they also provide as, a by-product, new ways for the Crawford number computation and solving the max–ratio minimization and related eigenvalue optimization. For those problems, the superior performance of the nonlinear eigenvector approaches, versus the state-of-the-art optimization methods, will be demonstrated by numerical experiments, with applications in coercivity constant computation of boundary operators and multicast transmit beamforming in signal processing.

The rest of this paper is organized as follows. In Section 2, we present the NEPv characterization for the global minimizer of the composite optimization (2). In Section 3, we develop and analyze the basic sequential subspace searching algorithm for NEPv resulting from a smooth objective function F. In Section 4, extensions to non-differentiable F functions are to be discussed. Implementation details are given in Section 5, followed by numerical examples in Section 6, and conclusions in Section 7.

Notation. Throughout the paper, we use notation conventions in matrix analysis. We use $\mathbb{C}^{m\times n}$ for the set of m-by-n complex matrices, with $\mathbb{C}^n = \mathbb{C}^{n\times 1}$ and $\mathbb{C} = \mathbb{C}^1$. For real numbers we have $\mathbb{R}^{m\times n}$, $\mathbb{R}^n$, and $\mathbb{R}$, respectively. We use $\cdot^T$ for transpose and $\cdot^H$ for conjugate transpose. Let X be a matrix, span(X) stands for the subspace spanned by the columns of X, $X(i:j,:)$ for the submatrix consisting row i to j of X, and $X(:,k:\ell)$ for columns k to ℓ. For a Hermitian X, we use $\lambda_{\min}(X)$ for its smallest eigenvalue. Since we are dealing with a real valued objective function $\mathcal{F}(x)$ in complex variables, which is not differentiable in the holomorphic sense, we will work with the Wirtinger derivatives. Namely, we view $\mathcal{F}(x) = \mathcal{F}(p,q)$ as function in the real and imaginary components of $x = p + \jmath q$, and define

$$\nabla^w \mathcal{F}(x) := \nabla_p \mathcal{F}(p,q) + \jmath \nabla_q \mathcal{F}(p,q), \tag{3}$$

where ∇_p and ∇_q are standard partial derivatives; see, e.g., [14]. Clearly, $\nabla^w \mathcal{F}(x)$ has a one-to-one correspondence with the gradient $\nabla_{(p,q)}\mathcal{F}(p,q)$, so its angle with a vector $h = r + \jmath s \in \mathbb{C}^n$ is defined as

$$\angle_w(\nabla^w \mathcal{F}(x), h) := \arccos \frac{\mathrm{Re}(h^H \cdot \nabla^w \mathcal{F}(x))}{\|\nabla^w \mathcal{F}(x)\|_2 \|h\|_2} \equiv \angle \left(\nabla_{(p,q)}\mathcal{F}(p,q), \begin{bmatrix} r \\ s \end{bmatrix} \right), \tag{4}$$

where $\angle$ denotes the angle between two real vectors in standard definition. Hence, $\mathcal{F}(x)$ is descending in the direction of h if $\cos \angle_w(\nabla^w \mathcal{F}(x), h) < 0$.

2 Optimality conditions and nonlinear eigenvalue problems

In this section, we consider the global optimality condition for problem (2). We will use subgradients to present our main results, so that the case of non-differentiable F will also be included. A vector $g \in \mathbb{R}^2$ is called a subgradient of F at y, if it satisfies

$$f(z) \geq f(y) + g^T(z - y), \ \forall z \in \mathbb{R}^2. \tag{5}$$

The set of all subgradients of F at y is called the *subdifferential*, which is denoted by $\partial F(y)$.

By standard non-smooth analysis, a necessary condition for $x \in \mathbb{C}^n$ being a local optimizer is given by

$$0 \ \in \ \overline{\partial^w} \mathcal{F}(x) \equiv \left\{ \frac{2g_1}{x^H x} \cdot \left(A - \frac{x^H A x}{x^H x} I \right) x + \frac{2g_2}{x^H x} \cdot \left(B - \frac{x^H B x}{x^H x} I \right) x \ \middle| \ \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \in \partial F(\rho(x)) \right\}, \tag{6}$$

where

$$\rho(x) = \left[\frac{x^H A x}{x^H x}, \frac{x^H B x}{x^H x} \right]^T \in \mathbb{R}^2, \tag{7}$$

and $\overline{\partial^w} \mathcal{F}(x)$ is the Clarke generalized gradient of $\mathcal{F}(x)$, obtained by chain rule[1] applied to the composition $\mathcal{F} = F \circ \rho$, see, e.g., [15, Prop 2.3.2, Thm 2.3.9] and [16, Thm 2.6, Eq (2.4)]. A vector x satisfying (6) is called *stationary*.

By slight reformulation, the stationary condition (6) can be written into *a nonlinear eigenvalue problem with eigenvector nonlinearity* (NEPv) as

$$H(x)x = \lambda x \qquad \text{and} \qquad x \neq 0, \tag{8}$$

where the coefficient matrix is given by

$$H(x) := g_1(x)A + g_2(x)B, \qquad \text{for some} \quad \begin{bmatrix} g_1(x) \\ g_2(x) \end{bmatrix} \in \partial F(\rho(x)), \tag{9}$$

[1] For $x \in \mathbb{R}^n$, the formula is obtained by straight forward derivation. In the complex case $x \in \mathbb{C}^n$, the formula is obtained in Wirtinger sense, by viewing $\mathcal{F}(x)$ as a function in the real and imaginary part of x and taking derivative w.r.t. each component.

and $\lambda = x^H H(x)x/(x^H x)$. And vice versa, an x satisfying the NEPv has a corresponding $\lambda = x^H H(x)x/(x^H x)$, hence must be stationary.

The NEPv (8) is a type of 'self-consistent' eigenvalue problem, in which the solution (λ_*, x_*) is an eigenpair of the matrix $H(x_*)$ defined by x_* itself. Since the Hermitian matrix $H(x_*)$ has n eigenvalues, which can be ordered as $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$, it holds that $\lambda_* = \lambda_\ell$ for some index $1 \le \ell \le n$. Here the order ℓ turns out to be crucial for the global optimality of x_*. As shown in the following theorem, the stationary condition (8) is also sufficient if $\lambda_* = \lambda_1$ is the smallest eigenvalue of $H(x_*)$. Beside global optimality, such an order of the eigenvalue will also be important to the algorithm design of the NEPv.

Theorem 1. *A vector $x \in \mathbb{C}^n$ is a global minimizer of $\mathcal{F}(x)$ if and only if it satisfies the NEPv* (8) *with the eigenvalue $\lambda = \lambda_{\min}(H(x))$.*

Proof. Let us first show the necessary condition. For clarity, we denote by x_* the global minimizer and $y_* := \rho(x_*) \in W(A,B)$. Let $f_{\min} = \mathcal{F}(x_*) = F(y_*)$, we define the (convex) sub–level set of F

$$L_f := \{z \in \mathbb{R}^2 \colon F(z) < f_{\min}\}.$$

If L_f is empty, then the convex function F achieves minimum at $y_* \in \mathbb{R}^2$. Hence, there exists $g = 0 \in \partial F(y_*)$ by convex analysis (see, e.g., [17]), so that (8) holds with $H(x_*) = 0$. Clearly, $\lambda = 0$ is a smallest eigenvalue.

Now, consider L_f being non-empty. Since $f_{\min}$ is a global minimizer, it holds $F(y) \ge f_{\min}$ for all $y \in W(A,B)$, which implies the two convex sets L_f and $W(A,B)$ are disjoint. By standard convex analysis, the two sets can be divided by a separation hyperplane. Namely, there exist $g \ne 0 \in \mathbb{R}^2$ and $c \in \mathbb{R}$, such that

$$g^T z \le c \text{ for all } z \in L_f \qquad \text{and} \qquad g^T z \ge c \text{ for all } z \in W(A,B). \tag{10}$$

By the definition of L_f, that F is a continuous function implies $y_* \in \overline{L}_f$ lies in the completion. Hence $y_* \in \overline{L}_f \bigcap W(A,B)$, and by (10) we have

$$g^T y_* = c. \tag{11}$$

Geometrically, it implies that $g^T z - c = 0$ defines a supporting hyperplane of L_f at y_*. The vector g, being an outer normal direction of the sublevel set L_f at y_*, thus satisfies $\alpha g \in \partial F(y_*)$ for some scalar $\alpha > 0$; see, e.g., [18, Thm. VI.1.3.5]. Since the defining inequalities of g in (10) still hold with multiplication of α, we can always assume $g \in \partial F(y_*)$.

The condition (11), combined with the right hand side of (10), implies

$$\frac{x_*^H (g_1 A + g_2 B)\, x_*}{x_*^H x_*} \equiv g^T y_* = c = \min_{z \in W(A,B)} g^T z \equiv \min_{x \in \mathbb{C}^n} \frac{x^H (g_1 A + g_2 B)\, x}{x^H x}.$$

Let $H(x_*) = g_1 A + g_2 B$, we have that the Rayleigh quotient $(x^H H(x_*)x)/(x^H x)$ achieves minimum at x_*. Hence x_* is an eigenvector corresponding to the smallest eigenvalue of $H(x_*)$.

Now let us turn to the sufficiency. Let (5) hold with $\lambda = \lambda_{\min}(H(x))$. Since $g \in \partial F(y)$ for $y = \rho(x)$, the subgradient inequality implies $\forall\, \widetilde{y} \in W(A,B)$ that

$$F(\widetilde{y}) - F(y) \ge g^T(\widetilde{y} - y) \ge \min_{z \in W(A,B)} g^T z - g^T y \quad = \min_{\widetilde{x} \ne 0} \frac{\widetilde{x}^H H(x) \widetilde{x}}{\widetilde{x}^H \widetilde{x}} - \frac{x^H H(x) x}{x^H x} = 0,$$

where for the last equation we used eigenvalue minimization principle for the smallest eigenvalue λ. Hence, $F(y) = \mathcal{F}(x)$ is a global minimizer. □

We comment that the eigenvalue ordering property in Theorem 1 is a direct consequence of the convexity of the numerical range and F. The result does not follow immediately from standard second order optimality conditions, which are sufficient conditions for local optimality that take a very different form from the one in Theorem 1; see, e.g., [19, Thm 12.6] for differentiable problems, and [16] for general convex composite minimization problems.

3 Sequential subspace searching

In this section, we consider numerical methods for solving the NEPv (8) for smooth objective functions. The non-smooth case will be discussed in Section 4. For a smooth function, the subdifferential $\partial F(y)$ reduces to the gradient, so the coefficient matrix $H(x)$ in (8) is uniquely defined by

$$H(x) = F_1(\rho(x)) \cdot A + F_2(\rho(x)) \cdot B, \tag{12}$$

where $F_j(y) = \frac{\partial F}{\partial y_j}(y)$, for $j = 1, 2$, denote the partial derivatives of F.

Note that eigenvalue problems similar to (8) also arise in computational physics and chemistry, as well as a few data analysis applications, see, e.g., [20, 21, 22, 23]. In those applications, a self-consistent-field (SCF) iteration is usually applied for the solution. But directly applying SCF to a large-scale NEPv is computationally expensive. For acceleration, we will develop a sequential subspace searching scheme in the following discussion.

3.1 Nonlinear Rayleigh-Ritz procedure

To reduce the computation cost for a large-size NEPv, we apply the popular subspace technique. The basic idea of subspace searching is as follows. Given a low-dimensional searching subspace $\mathcal{U} \subset \mathbb{C}^n$, we would like to compute a vector $\widehat{x} \in \mathcal{U}$ that best approximates the desired eigenvector. Since we are interested in the minimization problem (2), such a best approximation can be defined as

$$\widehat{x} = \operatorname*{arg\,min}_{x \in \mathcal{U}} \; F\left(\frac{x^H A x}{x^H x}, \frac{x^H B x}{x^H x}\right) = \operatorname*{arg\,min}_{x = Uv,\, v \in \mathbb{C}^k} \; \widehat{\mathcal{F}}(v) := F\left(\frac{v^H \widehat{A} v}{v^H \widehat{M} v}, \frac{v^H \widehat{B} v}{v^H \widehat{M} v}\right), \tag{13}$$

where $\widehat{A} = U^H A U$, $\widehat{B} = U^H B U$, $\widehat{M} = U^H U$, and $U \in \mathbb{C}^{n \times k}$ is a basis matrix of $\mathcal{U}$ (not necessarily orthogonal). Here the right hand side is again a composite minimization[2], but for matrices of a reduced size k. According to Theorem 1, the minimizer $\widehat{v}$ is therefore a solution to the projected NEPv

$$\widehat{H}(v) v = \lambda \widehat{M} v \tag{14}$$

with coefficient matrices

$$\widehat{H}(v) \equiv F_1\big(\widehat{\rho}(v)\big) \cdot \widehat{A} + F_2\big(\widehat{\rho}(v)\big) \cdot \widehat{B} \qquad \text{and} \quad \widehat{\rho}(v) = \left(\frac{v^H \widehat{A} v}{v^H \widehat{M} v}, \frac{v^H \widehat{B} v}{v^H \widehat{M} v}\right)^T, \tag{15}$$

where λ corresponds to the smallest eigenvalue of the matrix pencil $\widehat{H}(v) - \lambda \widehat{M}$. Since the NEPv (14) is of a smaller size k, the eigenvector $\widehat{v}$ can be efficiently computed by SCF iteration (see details below), or by convex optimization approaches as mentioned in the introduction. We can then obtain the desired

$$\widehat{x} = U \widehat{v}. \tag{16}$$

Such a procedure from above is called a *nonlinear Rayleigh-Ritz procedure*, which is a natural generalization of the well-known Rayleigh-Ritz procedure (see, e.g., [24]) to the NEPv (8).

An SCF iteration for the reduced NEPv (14) is summarized as follows: Given an initial vector $v^{(0)}$, we iteratively compute $v^{(1)}, v^{(2)}, \ldots$ satisfying

$$\widehat{H}(v^{(p)}) v^{(p+1)} = \lambda^{(p+1)} \widehat{M} v^{(p+1)}, \quad \text{for} \quad p = 0, 1, \ldots, \tag{17}$$

where $\lambda^{(p+1)}$ corresponds to the smallest eigenvalue of the matrix pencil $\widehat{H}(v^{(p)}) - \lambda \widehat{M}$. Such a fixed-point iteration is commonly applied to NEPvs' in computational physics and chemistry (see e.g., [20]), but its convergence is not always guaranteed. For global convergence, we will employ a safeguarding line search step when the function value $\mathcal{F}(v^{(p+1)})$ is not reduced. Namely, we *damp* the new update by

$$v^{(p+1)} \longleftarrow \alpha_p v^{(p+1)} + (1 - \alpha_p) v^{(p)},$$

[2] We can always write the reduced minimization problem into the form of (2) using $\widehat{x} = U \widehat{L}^{-H} \widehat{u}$ with $\widehat{u} = \arg\min_{u \in \mathbb{C}^k} F\left(\frac{u^H \widetilde{A} u}{u^H u}, \frac{u^H \widetilde{B} u}{u^H u}\right)$, where $\widetilde{A} = \widehat{L}^{-1} \widehat{A} \widehat{L}^{-H}$, $\widetilde{B} = \widehat{L}^{-1} \widehat{B} \widehat{L}^{-H}$, and $\widehat{M} = \widehat{L} \widehat{L}^H$ is the Cholesky factorization.

where $0 \le \alpha_p \le 1$ is a damping factor, which is obtained by a line search for $\alpha_p \in [0,1]$ such that

$$\widehat{\mathcal{F}}\left(v^{(p)} + \alpha d_p\right) < \widehat{\mathcal{F}}\left(v^{(p)}\right) \quad \text{with} \quad d_p = v^{(p+1)} - v^{(p)}.$$

See, for example, Algorithm 1 (line 5 to 8), where this is done by Armijo backtracking. This scheme was also applied in the SCF iteration of [23].

The line search from above works if d_p is already a descent direction, i.e., $\cos \angle_w(\nabla^w \widehat{\mathcal{F}}(v^{(p)}), d_p) < 0$. This always holds when $v^{(p)H} \widehat{M} v^{(p+1)} \neq 0$, by which we can assume the eigenvectors $v^{(p)}$ and $v^{(p+1)}$ are normalized such that $v^{(p)H} \widehat{M} v^{(p)} = 1$ and $v^{(p)H} \widehat{M} v^{(p+1)} > 0$. Then we can derive, using the gradient formula

$$\nabla^w \widehat{\mathcal{F}}(v^{(p)}) = 2\left(\widehat{H}(v^{(p)})v^{(p)} - \widehat{\mu}_p v^{(p)}\right) \quad \text{with } \widehat{\mu}_p = v^{(p)H}\widehat{H}(v^{(p)})v^{(p)}, \tag{18}$$

that

$$\nabla^w \widehat{\mathcal{F}}(v^{(p)})^H \cdot d_p = 2 \cdot v^{(p)H}\left(\widehat{H}(v^{(p)}) - \widehat{\mu}_p \cdot \widehat{M}\right) \cdot (v^{(p+1)} - v^{(p)}) = 2\left(v^{(p)H}\widehat{M}v^{(p+1)}\right) \cdot \left(\lambda^{(p+1)} - \widehat{\mu}_p\right) \; < 0,$$

where we used (17) in the second equation, and $\lambda^{(p+1)} \equiv \min_v (v^H H(v^{(p)})v)/(v^H \widehat{M} v)$ bearing the smallest Rayleigh quotient in the last equation. Hence $\cos \angle_w(\nabla^w \widehat{\mathcal{F}}(v^{(p)}), d_p) < 0$ by definition (4).

In the rare case of $v^{(p)H} \widehat{M} v^{(p+1)} = 0$, the increment d_p is not necessarily a descent direction. In this case, and, more generally, when d_p and $\nabla^w \widehat{\mathcal{F}}(v^{(p)})$ are almost orthogonal, i.e.,

$$-\cos \angle_w(d_p, \nabla^w \widehat{\mathcal{F}}(v^{(p)})) \le \gamma \quad \text{with } \gamma > 0 \text{ small},$$

we reset the search direction d_p as the negative gradient

$$d_p = -\nabla^w \widehat{\mathcal{F}}(v^{(p)}) \,/\, \|\nabla^w \widehat{\mathcal{F}}(v^{(p)})\|_2.$$

This safeguarding strategy ensures that the search direction d_p is descending, and it is gradient related (i.e., its orthogonal projection onto the negative gradient is uniformly bounded by a constant from below). We can immediately conclude the global convergence of the SCF iteration (Algorithm 1 without break), by a simple application of the convergence analysis of line search methods; see, e.g. [25, Prop. 1.2.1]. In particular, any limiting point v_* of $\{v^{(p)}\}_{p=0}^{\infty}$ will be stationary, hence is an eigenvector of the NEPv (14).

Algorithm 1 SCF iteration with line search for the NEPv (14)

Input: Starting vector $v^{(0)}$ with $v^{(0)H}\widehat{M}v^{(0)} = 1$, tolerance `tol_r`, `maxit`, and line search factors $\gamma, c, \tau \in (0,1)$ (e.g., $\gamma = c = \tau = 0.1$).

Output: Approximate eigenvector $\widehat{v}$.

1: **for** $p = 0, 1, \ldots,$ `maxit` **do**
2: Compute residual vector $\widehat{r}_p = \widehat{H}(v^{(p)})v^{(p)} - \widehat{\mu}_p \widehat{M} v^{(p)}$ with $\widehat{\mu}_p = v^{(p)H}\widehat{H}(v^{(p)})v^{(p)}$.
3: Check convergence: **if** $\|\widehat{r}_p\|_2 \le \|v^{(p)}\|_2 \cdot$ `tol_r` **then** break.
4: Solve the Hermitian eigenvalue problem $\widehat{H}(v^{(p)})v = \lambda \widehat{M} v$ for the smallest eigenvalue $\lambda^{(p+1)}$ and the eigenvector $v^{(p+1)}$ (normalized such that $v^{(p+1)H}\widehat{M}v^{(p+1)} = 1$ and $v^{(p)H}\widehat{M}v^{(p+1)} \ge 0$).
5: Prepare for line search: set $\alpha_p = 1$ and $d_p = v^{(p+1)} - v^{(p)}$ (use $d_p = -\widehat{r}_p/\|\widehat{r}_p\|_2$ if $-\cos \angle_w(d_p, \widehat{r}_p) \le \gamma$).
6: **while** $\widehat{\mathcal{F}}\left(v^{(p)}\right) - \widehat{\mathcal{F}}\left(v^{(p+1)}\right) < -\alpha_p \cdot c\,\mathrm{Re}(2d_p^H \widehat{r}_p)$ **do**
7: $\alpha_p := \tau\alpha_p$ and $v^{(p+1)} = v^{(p)} + \alpha_p d_p$. *% backtracking line search*
8: **end while**
9: **end for**
10: Return $\widehat{v} = v^{(p)}$.

3.2 Sequential subspace searching

Based on the nonlinear Rayleigh-Ritz procedure, we can develop the following sequential subspace searching algorithm for the NEPv (8): In iteration k, we search for an approximation x_{k+1} satisfying

$$x_{k+1} \in \mathrm{span}\Big\{\, x_{k-1},\, x_k,\, r_k \,\Big\}, \tag{19}$$

where the searching subspace is spanned by the current iterate x_k, the old iterate x_{k-1}, and the gradient

$$r_k := \frac{\nabla^w \mathcal{F}(x_k)}{\|\nabla^w \mathcal{F}(x_k)\|_2} = \xi_k \cdot \left(H(x_k)x_k - \mu(x_k) \cdot x_k \right) \quad \text{with} \quad \mu(x_k) = \frac{x_k^H H(x_k) x_k}{x_k^H x_k}, \tag{20}$$

and $\xi_k \geq 0$ is a normalization factor such that $\|r_k\|_2 = 1$. By the discussion in Section 3.1, x_{k+1} can be computed by the nonlinear Rayleigh-Ritz procedure. This process can be repeated until convergence; see Algorithm 2 for an outline of the overall searching strategy.

Algorithm 2 Sequential subspace searching for NEPv (8) with smooth F (prototype)

1: Initialize $x_0, x_{-1} \in \mathbb{C}^n$.
2: **for** $k = 0, \ldots$ **do**
3: Compute residual $r_k = H(x_k)x_k - \mu(x_k)x_k$ and normalize $r_k := r_k/\|r_k\|_2$.
4: Let $U_k = [x_{k-1}, x_k, r_k]$, form reduced $\widehat{A} = U_k^H A U_k$, $\widehat{B} = U_k^H B U_k$, and $\widehat{M} = U_k^H U_k$.
5: Solve the reduced NEPv (14) for the eigenvector $\widehat{v}_k$, and update $x_{k+1} = U_k \widehat{v}_k / \|U_k \widehat{v}_k\|_2$.
6: **end for**

Since gradients are used for searching, Algorithm 2 is at least as good as gradient descent in each iteration. We can therefore show the global convergence of the algorithm in the following theorem. The result also implies that the NEPv in line 5 needs not be solved exactly, and that accidental failures of convergence of SCF iteration within `maxit` will not kill the overall algorithm. This provides us with great flexibilities in choosing the parameters `tol_r` and `maxit` for Algorithm 1.

Theorem 2 (Global convergence)**.** *Let F be a smooth convex function over $W(A, B)$. Suppose in Algorithm 2 line 5 the reduced NEPv is solved either exactly, or by running a few steps of SCF iteration in Algorithm 1 starting with e_1. Then the resulting sequence $\{x_k\}_{k=1}^{\infty}$ is monotonic in function values, i.e., $\mathcal{F}(x_{k+1}) \leq \mathcal{F}(x_k)$, for $k = 1, \ldots$, and any of its limiting point $\widetilde{x}$ is an eigenvector of the NEPv* (8)*.*

Proof. It is sufficient to consider the case where the NEPv is solved by SCF iteration. Since safeguarding line search is applied for the reduced NEPv, we will show that x_{k+1} in iteration k is as good as an $\widetilde{x}_{k+1}$ by gradient-related backtracking line search for the original function $\mathcal{F}$. Consequently, the global convergence is obtained by a direct application of [26, Thm. 4.3] for accelerated line search methods.

Consider iteration k of Algorithm 2. Let $\widehat{v}_k^{(1)}$ be the first iterate in the SCF Algorithm 1. Due to the line search in line 5–8, it satisfies the Armijo back-tracking condition

$$\widehat{v}_k^{(1)} = e_1 + \alpha_{k,0} d_{k,0} \quad \text{with} \quad \widehat{\mathcal{F}}_k(\widehat{v}_k^{(1)}) \leq \widehat{\mathcal{F}}_k(e_1) + c \cdot \alpha_{k,0} \mathrm{Re}(d_{k,0}^H \cdot \nabla^w \widehat{\mathcal{F}}_k(e_1)),$$

where $\widehat{\mathcal{F}}_k$ is defined as (13) by $U_k = [x_k, x_{k-1}, r_k]$, and we have used $\alpha_{k,0}$ and $d_{k,0}$ for α_0 and d_0 from the line search to show their dependence on k. Because of safeguarding in line 5, the searching direction satisfies $\cos \angle_w(d_{k,0}, \nabla^w \widehat{\mathcal{F}}_k(e_1)) \leq -\gamma$.

Since $\widehat{\mathcal{F}}_k(v) = \mathcal{F}(U_k v)$, we have by chain rules $\nabla^w \widehat{\mathcal{F}}(e_1) = U_k^H \cdot \nabla^w \mathcal{F}(x_k)$. Let $g_k = U_k d_{k,0}$, the Armijo condition from above implies

$$\widetilde{x}_{k+1} := U_k \widehat{v}_k^{(1)} = x_k + \alpha_{k,0} g_k \quad \text{with} \quad \mathcal{F}(\widetilde{x}_{k+1}) \leq \mathcal{F}(x_k) + c \cdot \alpha_{k,0} \mathrm{Re}(g_k^H \cdot \nabla^w \mathcal{F}(x_k)).$$

Moreover, the increment g_k is also gradient related due to

$$\cos \angle_w(g_k, \nabla^w \mathcal{F}(x_k)) \equiv \frac{\mathrm{Re}(g_k^H \cdot \nabla^w \mathcal{F}(x_k))}{\|g_k\|_2 \cdot \|\nabla^w \mathcal{F}(x_k))\|_2} = \frac{\mathrm{Re}(d_{k,0}^H \cdot \nabla^w \widehat{\mathcal{F}}_k(e_1))}{\|U_k d_{k,0}\|_2 \cdot \|\nabla^w \mathcal{F}(x_k))\|_2} \leq \frac{\cos \angle_w(d_{k,0}, \nabla^w \widehat{\mathcal{F}}_k(e_1))}{\sqrt{3}} \leq -\frac{\gamma}{\sqrt{3}},$$

where we used $\|U_k d_{k,0}\|_2 \leq \sqrt{3}\|d_{k,0}\|_2$ (since U_k has unitary columns) and $\|\nabla^w \mathcal{F}(x_k)\|_2 = |e_3^H U_k^H \cdot \nabla^w \mathcal{F}(x_k)| \leq \|\nabla^w \widehat{\mathcal{F}}_k(e_1)\|_2$ (since $U_k e_3 = r_k \equiv \zeta_k \nabla^w \mathcal{F}(x_k)$) in the third equation. Therefore, $\widetilde{x}_{k+1}$ satisfies gradient related line search Armijo condition. Finally, due to monotonicity of Algorithm 1, it holds that $\mathcal{F}(x_{k+1}) \leq \mathcal{F}(\widetilde{x}_{k+1})$. □

Note that for linear eigenvalue problems, the sequential subspace searching scheme from above has been applied in the well known LOBPCG algorithm [11]. The Algorithm 2 presented here is a formal extension of such an algorithm to the case of NEPv (8). In particular, when $H(x) \equiv H$ is a constant matrix, the NEPv reduces to an Hermitian linear eigenvalue problem, for which Algorithm 2 coincides with LOBPCG in its simplest form.

In addition to eigenvalue computation, the sequential subspace searching also occurred in [27] for minimizing a quadratic function over a sphere, in [28] for unconstrained optimization problems, and in [26] for Riemannian optimization. Those works all dealt with real variables. Despite looks similar in the formulation, the application here with complex variables, and Wirtinger derivatives, has an effect of extending the searching subspace. More precisely, let $x = p + \jmath q$ and view $\mathcal{F}(x) := \mathcal{F}(p, q)$ as a function in the real variables p and q. Then minimizing $\mathcal{F}(p, q)$ by the real version of sequential subspace scheme results in

$$\begin{bmatrix} p_{k+1} \\ q_{k+1} \end{bmatrix} \in \text{span}\left\{ \begin{bmatrix} p_{k-1} \\ q_{k-1} \end{bmatrix}, \begin{bmatrix} p_k \\ q_k \end{bmatrix}, \begin{bmatrix} \nabla_p \mathcal{F}(x_k) \\ \nabla_q \mathcal{F}(x_k) \end{bmatrix} \right\}.$$

For the complex version in (19), by representing $x_k = p_k + \jmath q_k$, it is straightforward verification that we are searching for local optimizers in a 6-dimensional subspace

$$\begin{bmatrix} p_{k+1} \\ q_{k+1} \end{bmatrix} \in \text{span}\left\{ \begin{bmatrix} p_{k-1} \\ q_{k-1} \end{bmatrix}, \begin{bmatrix} -q_{k-1} \\ p_{k-1} \end{bmatrix}, \begin{bmatrix} p_k \\ q_k \end{bmatrix}, \begin{bmatrix} -q_k \\ p_k \end{bmatrix}, \begin{bmatrix} \nabla_p \mathcal{F}(x_k) \\ \nabla_q \mathcal{F}(x_k) \end{bmatrix}, \begin{bmatrix} -\nabla_q \mathcal{F}(x_k) \\ \nabla_p \mathcal{F}(x_k) \end{bmatrix} \right\},$$

which hence always leads to a better approximation.

3.3 Preconditioned expansion

The sequential subspace algorithm, as a gradient descent method, is linearly convergent in practice. For acceleration, we can extend the subspace (19) with extra searching vectors. We will show in this section how to choose those vectors such that the algorithm achieves superlinear convergence. Our analysis also sheds light on the design of preconditioners to speed up computation.

We begin with Newton's method for the NEPv. Assuming that A, B and x are real numbers, we can write the NEPv $H(x)x = \lambda x$, with the normalization condition $b^T x = 1$, as a root finding problem

$$G(x, \lambda) := \begin{bmatrix} H(x)x - \lambda x \\ 1 - b^T x \end{bmatrix} = 0,$$

for which one Newton's step, starting at (x_k, λ_k) with $b^T x_k = 1$, leads to the equation

$$J(x_k, \lambda_k) \left(\begin{bmatrix} x_{newton} \\ \lambda_{newton} \end{bmatrix} - \begin{bmatrix} x_k \\ \lambda_k \end{bmatrix} \right) = - \begin{bmatrix} r_k \\ 0 \end{bmatrix} \quad \text{with} \quad J(x_k, \lambda_k) = \begin{bmatrix} H(x_k) - \lambda_k I + S(x_k) & -x_k \\ -b^T & 0 \end{bmatrix}, \tag{21}$$

where x_{newton} and λ_{newton} are Newton's update, $r_k = H(x_k)x_k - \lambda_k x_k$, and $S(x) = [Ax, Bx]\nabla^2 F(\rho(x))[Ax, Bx]^H$ is a matrix of rank 2. Provided λ_k is not an eigenvalue of $H(x_k)$ and $\lambda_k \neq 0$, we can there on obtain from the upper block of the equation

$$\begin{aligned} x_{newton} &= x_k - \big(H(x_k) - \lambda_k I\big)^{-1} \big(H(x_k)x_k - \lambda_{newton} x_k + S(x_k) \cdot (x_{newton} - x_k)\big) \\ &= -\big(H(x_k) - \lambda_k I\big)^{-1} \cdot \big((\lambda_k - \lambda_{newton})x_k + S(x_k) \cdot (x_{newton} - x_k)\big) \\ &\in \text{span}\big\{ \big(H(x_k) - \lambda_k I\big)^{-1} \cdot \begin{bmatrix} x_k & Ax_k & Bx_k \end{bmatrix} \big\} \;=\; \text{span}\left\{ \begin{bmatrix} x_k & p_k^{(a)} & p_k^{(b)} \end{bmatrix} \right\}, \end{aligned} \tag{22}$$

where

$$\begin{bmatrix} p_k^{(a)}, p_k^{(b)} \end{bmatrix} := (H(x_k) - \lambda_k I)^{-1} [Ax_k,\ Bx_k], \tag{23}$$

and in the third equation we used $\text{Range}(S(x_k)) \subset \text{span}[Ax_k, Bx_k]$, in the last equation we used that $H(x_k)$ is a linear combination of A and B by definition (12).

Note that the Newton equation (21), derived for real variables, does not hold for general complex numbers. Nevertheless, we show in the proof of the next theorem that the inclusion property (22) is valid in both real and complex cases, and that the quadratic convergence of x_{newton} is guaranteed under the simple eigenvalue assumption.

Theorem 3 (Quadratic convergence)**.** *Let F be a function with continuous second derivative, x_* be a global minimizer of $\mathcal{F}(x)$, and b be an normalization vector with $b^H x_* \neq 0$. Assume that $\lambda_* = \lambda_{\min}(H(x_*))$ is a simple eigenvalue. Then for an x_k with $\tan\angle(x_k, x_*)$ sufficiently small and with $\lambda_k = (b^H H(x_k)x_k)/(b^H x_k)$ being nonzero and not an eigenvalue of $H(x_k)$, there exists an $\widehat{x}$ in the subspace* (22) *such that* $\tan\angle(\widehat{x}, x_*) = \mathcal{O}(|\tan\angle(x_k, x_*)|^2)$.

Proof. For notation simplicity, we use the boldface letters below for the augmented real variables of a complex vector $x \in \mathbb{C}^n$ and a matrix $X \in \mathbb{C}^{n\times n}$

$$\mathbf{x} = \begin{bmatrix}\mathrm{Re}(x)\\ \mathrm{Im}(x)\end{bmatrix}, \quad \overline{\mathbf{x}} = \begin{bmatrix}-\mathrm{Im}(x)\\ \mathrm{Re}(x)\end{bmatrix} \in \mathbb{R}^{2n} \qquad \text{and} \qquad \mathbf{X} = \begin{bmatrix}\mathrm{Re}(X) & -\mathrm{Im}(X)\\ \mathrm{Im}(X) & \mathrm{Re}(X)\end{bmatrix} \in \mathbb{R}^{2n\times 2n}. \tag{24}$$

Let $y = Xx$ and $z = \jmath Xx$, it is straightforward to verify that $\mathbf{y} = \mathbf{X}\mathbf{x}$ and $\mathbf{z} = \mathbf{X}\overline{\mathbf{x}}$ for the augmentation.

Let $\lambda = \alpha + \jmath\beta$, by separating the real and imaginary part of the NEPv (8) and the normalization condition $b^H x = 1$, we obtain an augmented real system of $2n+2$ equations

$$\mathbf{G}(\mathbf{x}, \alpha, \beta) := \begin{bmatrix}\mathbf{H}(\mathbf{x})\mathbf{x} - \mathbf{\Lambda}\mathbf{x}\\ 1 - \mathbf{b}^T\mathbf{x}\\ 0 - \overline{\mathbf{b}}^T\mathbf{x}\end{bmatrix} = 0,$$

where

$$\mathbf{\Lambda} = \begin{bmatrix}\alpha I_n & -\beta I_n\\ \beta I_n & \alpha I_n\end{bmatrix}, \quad \mathbf{H}(\mathbf{x}) = F_1(\rho(\mathbf{x}))\cdot\mathbf{A} + F_2(\rho(\mathbf{x}))\cdot\mathbf{B} \text{ with } \rho(\mathbf{x}) = \left[\frac{\mathbf{x}^T\mathbf{A}\,\mathbf{x}}{\mathbf{x}^T\mathbf{x}}, \frac{\mathbf{x}^T\mathbf{B}\,\mathbf{x}}{\mathbf{x}^T\mathbf{x}}\right]^T.$$

Let $(\mathbf{x}_k,\, \alpha_k, \beta_k)$ with $b^H x_k = 1$ be an approximate eigenpair, then a Newton's iteration leads to

$$\mathbf{J}(\mathbf{x}_k, \alpha_k, \beta_k)\left(\begin{bmatrix}\mathbf{x}_{newton}\\ \alpha_{newton}\\ \beta_{newton}\end{bmatrix} - \begin{bmatrix}\mathbf{x}_k\\ \alpha_k\\ \beta_k\end{bmatrix}\right) = -\begin{bmatrix}\mathbf{r}_k\\ 0\\ 0\end{bmatrix} \text{ with } \mathbf{J}(\mathbf{x}_k, \alpha_k, \beta_k) = \begin{bmatrix}\mathbf{H}(\mathbf{x}_k) - \mathbf{\Lambda}_k + \mathbf{S}(\mathbf{x}_k) & -\mathbf{x}_k & -\overline{\mathbf{x}}_k\\ -\mathbf{b}^T & 0 & 0\\ -\overline{\mathbf{b}}^T & 0 & 0\end{bmatrix}, \tag{25}$$

where $\mathbf{r}_k = \mathbf{H}(\mathbf{x}_k)\mathbf{x}_k - \mathbf{\Lambda}_k\mathbf{x}_k$ and $\mathbf{S}(\mathbf{x}) := [\mathbf{A}\mathbf{x}, \mathbf{B}\mathbf{x}]\cdot\nabla^2 F(\rho(\mathbf{x}))\cdot[\mathbf{A}\mathbf{x}, \mathbf{B}\mathbf{x}]^T \succeq 0$. Here note that (25) is different from the augmented real systems of (21), which has in general a different $\mathbf{S}(\mathbf{x})$ matrix. Next, following the same derivation as in (22), we obtain from the leading $2n$ elements that

$$\mathbf{x}_{newton} \in \mathrm{span}\big\{\mathbf{x}_k, \overline{\mathbf{x}}_k, (\mathbf{H}(\mathbf{x}_k) - \mathbf{\Lambda}_k)^{-1}[\mathbf{A}\mathbf{x}_k, \mathbf{A}\overline{\mathbf{x}}_k, \mathbf{B}\mathbf{x}_k, \mathbf{B}\overline{\mathbf{x}}_k]\big\} = \mathrm{span}\big\{\mathbf{x}_k, \overline{\mathbf{x}}_k, \mathbf{p}_k^{(a)}, \overline{\mathbf{p}}_k^{(a)}, \mathbf{p}_k^{(b)}, \overline{\mathbf{p}}_k^{(b)}\big\}, \tag{26}$$

where $\mathbf{p}_k^{(a)}, \overline{\mathbf{p}}_k^{(a)}, \mathbf{p}_k^{(b)}, \overline{\mathbf{p}}_k^{(b)}$ are augmented real vectors of $[p_k^{(a)}, p_k^{(b)}] = (H(x_k) - \lambda_k I)^{-1}[Ax_k, Bx_k]$, and the last equation can be verified by straightforward calculation (since $\mathbf{H}(\mathbf{x}_k) - \mathbf{\Lambda}_k$ is the real augmentation of $H(x_k) - \lambda_k I$). By writing the real $\mathbf{x}_{newton}$ back to a length-n complex vector x_{newton}, we obtain the inclusion relation (22).

It remains to show that $\mathbf{x}_{newton}$ by (25) leads to quadratic convergence. Since the objective function $\mathcal{F}(x)$ is homogeneous in x, we can always assume x_* is normalized with $b^H x_* = 1$, and $x_k = x_* + \mathcal{O}(|\tan\angle(x_k, x_*)|)$. Hence, the real augmentation $(\mathbf{x}_*, \alpha_*, \beta_*)$ is a solution to $\mathbf{G}(\mathbf{x}, \alpha, \beta) = 0$, and it also satisfies $(\mathbf{x}_*, \alpha_*, \beta_*) = (\mathbf{x}_k, \alpha_k, \beta_k) + \mathcal{O}(|\tan\angle(x_k, x_*)|)$ due to $\lambda_k \equiv (b^H H(x_k)x_k)/(b^H x_k) = \lambda_* + \mathcal{O}(|\tan\angle(x_k, x_*)|)$. For the quadratic convergence of Newton's methods, it is sufficient to show the Jacobian $\mathbf{J}(\mathbf{x}_*, \alpha_*, \beta_*)$ is non-singular; see, e.g., [19, Thm. 11.2].

By the simplicity of λ_*, the matrix $H(x_*) - \lambda_* I$ is positive semidefinite with the null space spanned by x_*. Hence, the real augmentation $\mathbf{H}(\mathbf{x}_*) - \mathbf{\Lambda}_*$ is also positive semidefinite with $\mathrm{Null}(\mathbf{H}(\mathbf{x}_*) - \mathbf{\Lambda}_*) = \mathrm{span}\{\mathbf{x}_*, \overline{\mathbf{x}}_*\}$ being the null space. Let $\mathbf{z} = [\mathbf{z}_1; z_2] \in \mathbb{R}^{2n+2}$, with $\mathbf{z}_1 \in \mathbb{R}^{2n}$ and $z_2 \in \mathbb{R}^2$, be such that $\mathbf{J}(\mathbf{x}_*, \alpha_*, \beta_*)\mathbf{z} = 0$. We obtain $\mathbf{z}_1 \perp \mathrm{span}\{\mathbf{b}, \overline{\mathbf{b}}\}$, which implies $\mathbf{z}_1 \notin \mathrm{span}\{\mathbf{x}_*, \overline{\mathbf{x}}_*\} \equiv \mathrm{Null}(\mathbf{H}(\mathbf{x}_*) - \mathbf{\Lambda}_*)$ due to $b^H x_* \neq 0$. Combined with

$$0 = \mathbf{z}^T\mathbf{J}(\mathbf{x}_*, \alpha_*, \beta_*)\mathbf{z} = \mathbf{z}_1^T(\mathbf{H}(\mathbf{x}_*) - \mathbf{\Lambda}_* + \mathbf{S}(x_*))\mathbf{z}_1 \geq \mathbf{z}_1^T(\mathbf{H}(\mathbf{x}_*) - \mathbf{\Lambda}_*)\mathbf{z}_1 \geq 0,$$

it implies $\mathbf{z}_1 = 0$. Therefore, $[\mathbf{x}_*, \overline{\mathbf{x}}_*]z_2 = 0$, hence, $z_2 = 0$. Namely, $\mathbf{z} = 0$ is the only solution to $\mathbf{J}(\mathbf{x}_*, \alpha_*, \beta_*)\mathbf{z} = 0$, and the matrix is non-singular. □

Since (22) automatically includes the Newton's update, we can achieve superlinear convergence by expanding the searching subspace (19) as follows

$$\operatorname{span}\left\{ x_{k-1}, x_k, r_k, \;\middle|\; p_k^{(a)}, p_k^{(b)} \right\}. \tag{27}$$

To compute $p_k^{(a)}$ and $p_k^{(b)}$, we need to solve a linear system of equations (23) with two right hand sides. For structured matrices, this can be done efficiently by direct solvers; e.g., using sparse LU.

To reduce cost, one can also apply iterative methods for inexact solutions. In this case, it is more appropriate to represent the expansion vectors as

$$\left[p_k^{(a)}, p_k^{(b)}\right] := (H(x_k) - \lambda_k I)^{-1} \left[r_A(x_k),\ r_B(x_k)\right], \tag{28}$$

where

$$r_A(x) = \left(A - \frac{x^H A x}{x^H x} I\right) x \qquad \text{and} \qquad r_B(x) = \left(B - \frac{x^H B x}{x^H x} I\right) x \tag{29}$$

are residual vectors corresponding to $A - \lambda I$ and $B - \lambda I$, respectively. Assuming $\lambda_k \neq (x_k^H H(x_k) x_k)/(x_k^H x_k)$ is not exactly given by the nonlinear Rayleigh quotient, the defining equation (22) for the two vectors in (28) can be verified by straightforward calculation. Compared to (23), the new formulas from above apply inversion not on x_k but on the residuals, namely, the gradients of the Rayleigh quotients. When a preconditioner $T_k \approx (H(x_k) - \lambda_k I)^{-1}$ is available, it leads to

$$\left[p_k^{(a)}, p_k^{(b)}\right] := T_k \cdot \left[r_A(x_k),\ r_B(x_k)\right], \tag{30}$$

which resembles the preconditioned residual technique and the generalized Davison methods commonly used in linear eigenvalue computation; see, e.g., [12, 29].

3.4 Global certification and restarting

The sequential subspace algorithm, as shown in Theorem 2, is globally convergent to an eigenvector of the NEPv (8), and in practice it always converges to the one corresponding to the smallest eigenvalue, hence, the global minimizer of $\mathcal{F}$. Such a global convergence is partially because of the 'global search' within the subspace, which can more easily escape local optimizers and saddle points than standard line search based optimization techniques.

Even so, the rare but possible convergence to a local optimizer is undesirable. In applications where global optimality is a crucial concern, it is necessary to certify that the computed x_* is indeed a global solution. This can be done by computing the smallest eigenvalue $\lambda_{\min}$, and the corresponding eigenvector $\widehat{x}$, of the matrix $H(x_*)$ with

$$H(x_*)\widehat{x} = \lambda_{\min} \cdot \widehat{x}. \tag{31}$$

By Theorem 1, x_* is a global minimizer if $\lambda_{\min} = \lambda_* \equiv (x_*^H H(x_*) x_*)/(x_*^H x_*)$. The linear eigenvalue problem from above can be solved by iterative methods such as LOBPCG, and in MATLAB by `eigs`.

If global certification fails, namely, $\lambda_{\min} \neq \lambda_*$, we will have to restart the algorithm for a better solution. In this case, the eigenvector $\widehat{x}$ from above provides an ideal restarting vector. As justified in Lemma 1, using

$$x_0 := \widehat{x} \quad \text{and} \quad x_{-1} := x_*, \tag{32}$$

we can always obtain an x_1 with a smaller $\mathcal{F}(x_1) < \mathcal{F}(x_*)$, and hence escape the local optimizer x_*.

Lemma 1. *Let (λ_*, x_*) be an eigenpair of the NEPv (8), and $(\lambda_{\min}, \widehat{x})$ be a solution to the linear eigenvalue problem (31). If $\lambda_* > \lambda_{\min}$, then it holds strictly that*

$$\min_{x \in \operatorname{span}([x_*, \widehat{x}])} \mathcal{F}(x) < \mathcal{F}(x_*). \tag{33}$$

Proof. Let $U = [x_*, \widehat{x}]$, we have U is of full column rank since $\lambda_* \neq \lambda_{\min}(H(x_*))$. Hence, (33) can be written as

$$\min_{y \in W} F(y) < F(\rho(x_*)) \qquad \text{with} \qquad W := \{[v^H \widehat{A} v, v^H \widehat{B} v] \colon v^H \widehat{M} v = 1\},$$

where $\widehat{A} = U^H A U$, $\widehat{B} = U^H B U$, and $\widehat{M} = U^H U$. Due to the convexity of the numerical range W, and that both $\rho(x_*), \rho(\widehat{x}) \in W$, we have the entire line segment $[\rho(x_*), \rho(\widehat{x})] \subset W$. Therefore, it is sufficient to show that F is descending along the line segment $[\rho(x_*), \rho(\widehat{x})]$, which follows from

$$\nabla F(y)|_{y=\rho(x_*)} \cdot (\rho(\widehat{x}) - \rho(x_*)) = \widehat{x}^H H(x_*)\widehat{x} - x_*^H H(x_*)x_* = \lambda_{\min}(H(x_*)) - \lambda_* < 0,$$

where we have assumed that both x_* and $\widehat{x}$ are unitary. □

4 Extension to non-smooth objective functions

In this section, we consider the application of sequential subspace methods to non-differentiable objective functions. One issue that has to be addressed is how to choose the searching subspace. A generalized gradient in $\overline{\partial^w}\mathcal{F}(x)$ can not be used immediately as searching direction, since it is neither uniquely defined nor necessarily descending. But due to the special composite formulation of $\mathcal{F}$, the generalized gradient (6) will always satisfy the following inclusion relation

$$\overline{\partial^w}\mathcal{F}(x) \subset \operatorname{span}\big\{r_A(x),\ r_B(x)\big\}, \tag{34}$$

where $r_A(x)$ and $r_B(x)$ are residuals, defined in (29), for matrix pencils $A - \lambda I$ and $B - \lambda I$, respectively. Therefore, we can take the two-dimensional 'descent subspace', in place of the single gradient direction, for local searching. More precisely, for sequential subspace methods, we propose the following searching strategy in analogy to (19) for the smooth case

$$x_{k+1} \in \operatorname{span}\Big\{\ x_{k-1},\ x_k,\ r_A(x_k),\ r_B(x_k)\ \Big\}, \tag{35}$$

where the subspace is spanned by the current iterate x_k, the old iterate x_{k-1}, and the descent subspace.

The best approximation x_{k+1} can be defined in the same way as (13), and it is a solution to the following reduced NEPv

$$\widehat{H}(v)v = \lambda \widehat{M} v \quad \text{with} \quad \widehat{H}(v) = g_1\widehat{A} + g_2\widehat{B} \qquad \text{for some} \quad \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \in \partial F(\widehat{\rho}(v)), \tag{36}$$

where λ is the smallest eigenvalue of the matrix pencil $\widehat{H}(v) - \lambda\widehat{M}$. This NEPv, unlike (14) in the smooth case, can not be immediately solved by SCF iteration since $\widehat{H}(v)$ is not uniquely defined. Nevertheless, it can be reformulated, and efficiently solved, as a convex minimization in the form of (1) thanks to its small size. For the moment, we assume the solution is always available, but do not make any assumption on how it is computed.

Algorithm 3 Sequential subspace searching for NEPv (8) with nonsmooth F (prototype)

1: Initialize $x_0, x_{-1} \in \mathbb{C}^n$.
2: **for** $k = 0, \ldots$ **do**
3: Compute residuals $r_k^{(a)} = r_A(x_k)$ and $r_k^{(b)} = r_B(x_k)$ using (29).
4: Let $U_k = [x_{k-1}, x_k, r_k^{(a)}, r_k^{(b)}]$, form reduced $\widehat{A} = U_k^H A U_k$, $\widehat{B} = U_k^H B U_k$, and $\widehat{M} = U_k^H U_k$.
5: Solve the reduced NEPv (36) for the eigenvector $\widehat{v}_k$, and update $x_{k+1} = U_k\widehat{v}_k / \|U_k\widehat{v}_k\|_2$.
6: **end for**

We therefore obtain an algorithm framework of sequential subspace searching outlined in Algorithm 3. Despite the non-smoothness in F, we can show a global convergence similar to the smooth version in Theorem 2.

Theorem 4. *The sequence $\{x_k\}_{k=1}^{\infty}$ produced by Algorithm 3 is monotonic in function values $\mathcal{F}(x_{k+1}) \le \mathcal{F}(x_k)$, for $k = 1, \ldots$, and any of its limiting point $\widetilde{x}$ is an eigenvector of the NEPv (8).*

Proof. The monotonicity is a direct consequence of (13), since the current iterate x_k is included in the searching subspace (35). As a result, we have $F(\rho(\widetilde{x})) = \mathcal{F}_\infty$ with $\mathcal{F}_\infty \le \lim_{k\to\infty} \mathcal{F}(x_k)$. For global convergence, it suffices to consider $\rho(\widetilde{x})$ being a non-smooth point of F, since otherwise Theorem 2 applies.

Let us first show that for any non-stationary point $\widetilde{x}$, it holds strictly that $\mathcal{F}(\widetilde{x}) < \min_{x\in\mathcal{U}} \mathcal{F}(x)$ with $\mathcal{U} \equiv \operatorname{span}\{\widetilde{x}, r_A(\widetilde{x}), r_B(\widetilde{x})\}$. Since otherwise, we have $\mathcal{F}(\widetilde{x}) = \min_{x\in\mathcal{U}} \mathcal{F}(x)$. Let U be an orthogonal base of $\mathcal{U}$,

then by the projection formula (13), we have $\widetilde{v} = U^H\widetilde{x}$ is a global minimizer of the reduced $\widehat{\mathcal{F}}(v)$. According to Theorem 1, it satisfies the NEPv $H(\widetilde{v})\widetilde{v} = \widetilde{\lambda}\widetilde{v}$, namely,

$$0 = (g_1 \cdot U^H AU + g_2 \cdot U^H BU - \widetilde{\lambda}I)\widetilde{v} = U^H(g_1 A + g_2 B - \widetilde{\lambda}I)\widetilde{x},$$

for some $g \in \partial F(\rho(\widetilde{x}))$. Since $(g_1A+g_2B-\widetilde{\lambda}I)\widetilde{x} \in \mathcal{U}$, we have $(g_1A+g_2B-\widetilde{\lambda}I)\widetilde{x} = 0$. Hence $\widetilde{x}$ is an eigenvector of the NEPv (8), contradicting $\widetilde{x}$ being non-stationary.

Let us assume $\widetilde{x}$ is non-stationary, then $\exists\ \widehat{x} \in \mathrm{span}\{\widetilde{x}, r_A(\widetilde{x}), r_B(\widetilde{x})\}$, s.t., $\|\widehat{x}\|_2 = 1$ and $\mathcal{F}(\widehat{x}) = \mathcal{F}(\widetilde{x}) - \delta$ for some constant $\delta > 0$. Since $\widetilde{x}$ is a limiting point, we obtain by the continuity of $r_A(x)$, $r_B(x)$ that

$$\forall \varepsilon > 0,\ \exists\, x_k, \quad \text{s.t.,} \quad [\widetilde{x}, r_A(\widetilde{x}), r_B(\widetilde{x})] = [x_k, r_A(x_k), r_B(x_k)] + E \quad \text{and} \quad \|E\|_2 \le \varepsilon,$$

hence $\widehat{x} = \widehat{x}_k + \mathcal{O}(\varepsilon)$ for some $\widehat{x}_k \in \mathrm{span}\{[x_k, r_A(x_k), r_B(x_k)]\}$. This implies

$$\mathcal{F}(x_{k+1}) \le \min_{x \in \mathrm{span}\{[x_k, r_A(x_k), r_B(x_k)]\}} \mathcal{F}(x) \le \mathcal{F}(\widehat{x}_k) = \mathcal{F}(\widehat{x}) + O(\varepsilon) = \mathcal{F}_\infty - \delta + O(\varepsilon).$$

Take $\varepsilon \to 0$, then $\mathcal{F}(x_{k+1}) < \mathcal{F}_\infty$, which contradicts $\lim_{k\to\infty} \mathcal{F}(x_k) \ge \mathcal{F}_\infty$. □

We make two comments about Algorithm 3 in comparison with its smooth counterpart. First, for a differentiable F function, we can also apply the extended subspace (35). In practice, however, this only makes marginal improvement in approximation over the single vector formula (19), whereas the computation cost is increased due to larger projection matrices. Hence, for efficiency consideration, the smooth version Algorithm 2 is always recommended. Secondly, since the matrix $H(x_k)$ is not uniquely defined at a non-smooth point, it is generally difficult to apply the global certification for a computed eigenvector x_k as in Section 3.4.

5 Implementation issues

In this section, we consider implementation details for the sequential subspace searching algorithms. The three-term recurrence (19) and (35) allows for efficient implementation, and many techniques developed for LOBPCG (see, e.g., [11]) can be applied in parallel for our nonlinear version, including the use of difference vectors to improve the condition number of the basis matrix, and block iteration to enhance performance. For simplicity of exposition, those techniques will be explained mainly for the smooth version Algorithm 2, but their application to the non-smooth case is straightforward.

Difference vectors. In the basis matrix U_k of Algorithm 2, the vectors x_{k-1} and x_k tend to be linearly dependent upon convergence, which can bring up numerical issues for solving the reduced problem. One way to address this issue is to use difference vectors to improve the condition number. More precisely, we can introduce a new vector $p_k = x_k - x_{k-1}$ representing the difference between x_{k-1} and x_k. Rather than using explicitly $[x_{k-1}, x_k, r_k]$ as a basis, we keep $[x_k, p_k, r_k]$ in the process and proceed with the recurrence

$$\begin{aligned} x_{k+1} &= \alpha_k x_k + \beta_k r_k + \gamma_k p_k, \\ p_{k+1} &= \beta_k r_k + \gamma_k p_k, \\ r_{k+1} &= H(x_{k+1})x_{k+1} - \mu(x_{k+1})x_{k+1}, \end{aligned}$$

where α_k, β_k and γ_k are coefficients determined by the nonlinear Rayleigh Ritz procedure with $U = [x_k, r_k, p_k]$. By the recurrence relation $p_{k+1} = x_{k+1} - \alpha_k x_k$ and the reasonable assumption $\alpha_k \neq 0$, it holds

$$\mathrm{span}\left\{\begin{bmatrix} x_{k+1} & p_{k+1} & r_{k+1} \end{bmatrix}\right\} = \mathrm{span}\left\{\begin{bmatrix} x_k & x_{k+1} & r_{k+1} \end{bmatrix}\right\}. \tag{37}$$

Hence we can obtain x_{k+2} by $[x_{k+1},\, p_{k+1},\, r_{k+1}]$, and repeat this process until convergence.

Block iteration. Block iteration is widely used in eigenvalue computation. The idea is to iterate over ℓ vectors $X_k = [x_k^{(1)}, \ldots, x_k^{(\ell)}]$ simultaneously rather than a single x_k. This is appealing from efficiency perspective for particular computer architectures, in addition to other benefits such as easy for parallelization and faster convergence by extended searching subspace. For a straightforward block implementation of Algorithm 2, we

can take the leading $x_k^{(1)}$ to serve as the approximate eigenvector, and the rest $x_k^{(2)}, \ldots, x_k^{(\ell)}$ as auxiliary vectors. The three-term recurrence (19) can be formally written as

$$X_{k+1} \in \operatorname{span}\{[X_{k-1}, X_k, R_k]\},$$

where "$\in$" holds for each column of X_{k+1}, and $R_k = [r_k^{(1)}, \ldots, r_k^{(\ell)}]$ consists of the residuals

$$r_k^{(j)} = H\left(x_k^{(1)}\right) x_k^{(j)} - \mu_k\left(x_k^{(j)}\right) x_k^{(j)} \quad \text{for} \quad j = 1, \ldots, \ell, \quad \text{with} \quad \mu_k(x) = \frac{x^H H(x_k^{(1)}) x}{x^H x}, \tag{38}$$

which is a natural generalization of the formula (20) to the block version, with the leading $r_k^{(1)} = \nabla^w \mathcal{F}(x_k^{(1)})/2$ corresponding to the gradient at the optimizer $x_k^{(1)}$. In analogy to the single vector version, we can apply nonlinear Rayleigh-Ritz procedure with $U = [X_{k-1}, X_k, R_k]$, and define the new block as

$$X_{k+1} = [X_{k-1}, X_k, R_k] V_{k+1}, \tag{39}$$

where the columns in V_{k+1} consist of the eigenvectors corresponding to the ℓ smallest eigenvalue of the linear eigenvalue problem $\widehat{H}(\widehat{v}_k)v = \lambda \widehat{M} v$, defined by the eigenvector $\widehat{v}_k$ of the reduced NEPv. The first column of V_{k+1} is always set to $\widehat{v}_k$, so that the leading $x_{k+1}^{(1)}$ is the local optimal approximation.

Algorithm 4 Sequential subspace searching for NEPv (8)

Input: Coefficient matrices A, B, block size ℓ, starting vectors $X_0 \in \mathbb{C}^{n\times\ell}$ and $P_0 = 0^{n\times\ell}$, tolerance `tol`.
Output: Approximate eigenvector $x_k^{(1)} = X_k(:, 1)$.

1: **for** $k = 0, \ldots$ **do**
2: Compute Rayleigh quotients $y_k^{(j)} = \rho(x_k^{(j)})$ for $j = 1, \ldots, \ell$.
3: **if** F is differentiable **then** {% smooth case}
4: Compute gradient $g_k = \nabla F(y_k^{(1)})$, and $\mu_k^{(j)} = g_k^T y_k^{(j)}$ for $j = 1, \ldots, \ell$.
5: Compute residual vectors $r_k^{(j)} = \left(H(x_k^{(1)}) - \mu_k^{(j)} I\right) x_k^{(j)}$ for $j = 1, \ldots, \ell$.
6: Convergence check: **if** $\|r_k^{(1)}\| \le$ `tol`, **then** break.
7: (*Optional, for* $\ell \ge 2$) Preconditioned expansion: $R_k := [r_k^{(1)}, \ldots, r_k^{(\ell-2)}, | \, p_k^{(a)}, p_k^{(b)}]$ by (23) or (30).
8: **else** {% non-smooth case, require $\ell = 2s$ is even}
9: Convergence check: **if** $k \ge 1$ and $F(y_k^{(1)}) \ge F(y_{k-1}^{(1)})$ **then** return.
10: Set residuals $R_k = [r_A(x_k^{(1)}), r_B(x_k^{(1)}), \ldots, r_A(x_k^{(s)}), r_B(x_k^{(s)})]$ using (29).
11: **end if**
12: Orthogonalize $R_k = \operatorname{orth}(R_k)$.
13: Let $U_k = [X_k, P_k, R_k]$ and compute projection $\widehat{A} = U_k^H A U_k$, $\widehat{B} = U_k^H B U_k$, and $\widehat{M} = U_k^H U_k$.
14: Solve the reduced NEPv (14) (or (36) for non-smooth F) for the eigenvector $\widehat{v}_k$ and $\widehat{H}(\widehat{v}_k)$.
15: Find eigenvectors $V_{k+1} = [\widehat{v}_k, \widehat{v}_k^{(2)}, \ldots, \widehat{v}_k^{(\ell)}]$ corresponding to the ℓ smallest eigenvalues of $\widehat{H}(\widehat{v}_k) - \lambda \widehat{M}$.
16: Update $X_{k+1} = [X_k, P_k, R_k] \cdot V_{k+1}$.
17: Update $P_{k+1} = [P_k, R_k] \cdot V_{k+1}(\ell + 1 : 3\ell, :)$.
18: **end for**
19: (*Optional*) Global certification: compute the smallest eigenvalue $\lambda_{\min}$ and the eigenvector x of $H(x_k^{(1)})$. If $\lambda_{min} < \mu_k^{(1)}$ then restart the process with X_0 and P_0 satisfying $X_0(:, 1) = x$ and $P_0(:, 1) = x - x_k^{(1)}$.

We summarize in Algorithm 4 the actual sequential subspace algorithm, with a few implementation details listed as follows.

(a) Matrix-vector multiplications (MatVec) $G[X_k, P_k, R_k]$ for $G \in \{A, B\}$ are required in lines 2, 5, 10 and 13. To avoid unnecessary recalculations, we can precompute, and save with 4ℓ auxiliary vectors, the results of $A_k := A[X_k, P_k]$ and $B_k := B[X_k, P_k]$. Then for the projection in line 13, we only need 2ℓ extra MatVecs

$$A_k^r := A \cdot R_k \qquad \text{and} \qquad B_k^r := B \cdot R_k. \tag{40}$$

In the following iterations, A_k and B_k can be updated (using lines 16, 17) without any explicit MatVecs by

$$A_{k+1} = [A_k, A_k^r]X_k, \quad B_{k+1} = [B_k, B_k^r]X_k \quad \text{with} \quad X_k = \begin{bmatrix} V_{k+1}, & \begin{bmatrix} 0 \\ V_{k+1}(\ell+1:3\ell,:) \end{bmatrix} \end{bmatrix}. \tag{41}$$

In this way, only 2ℓ MatVecs in (40) are required in each iteration.

In practice, numerical error will accumulate in the updating formulas (41) as k increases, causing A_k and B_k deviate from the exact $A[X_k, P_k]$ and $B[X_k, P_k]$. To avoid such a problem, both matrices can be recalculated explicitly every N_{rec} iterations, e.g., with $N_{rec} = 50$ as used in our numerical experiments.

(b) In line 6 we use the residual norm of $r_k^{(1)} = \nabla^w \mathcal{F}(x_k^{(1)})/2$ (i.e., the gradient) for the stopping criteria. As a common practice in numerical optimization, we can take the tolerance $\texttt{tol} = \mathcal{O}(\sqrt{\epsilon_{mach}})$ to be of order of square root of the machine precision ϵ_{mach}.

This scheme can not be applied for the non-smooth case, where the gradient is not defined. In line 9, we stop the algorithm when the function values $\mathcal{F}(x_k)$ lose monotonicity, an indication of accuracy close to machine precision.

(c) For efficiency consideration, we implement the preconditioned extension in line 7 by replacing the last two columns of R_k with $p_k^{(a)}$ and $p_k^{(a)}$ (assuming the block size $\ell \geq 3$). In this way, the size of searching subspace (i.e., $[X_k, P_k, R_k]$), averaged by the number of MatVecs (see (a) and (40)) in each iteration, defines the Subspace-to-MatVecs Ratio

$$\text{SMR} := \frac{\text{subspace size}}{\#\text{MatVec}} = \frac{3\ell}{2\ell} = \frac{3}{2}.$$

In comparison, for the naive implementation of appending $p_k^{(a)}$, $p_k^{(a)}$ directly to $R_k := [R_k, p_k^{(a)}, p_k^{(a)}]$, the algorithm would use a searching subspace $[X_k, P_k, R_k]$ of size $3\ell + 2$, and require $2(\ell + 2)$ MatVecs in (40), which leads to smaller $\text{SMR} = (3\ell + 2)/2(\ell + 2) < 3/2$. Namely, by the same number of MatVecs, such a strategy can only search in a smaller subspace.

As the algorithm proceeds, $x_k^{(1)}, p_k^{(a)}, p_k^{(b)}$ tend to be linearly dependent. For numerical stability, we can explicitly project the vectors $[p_k^{(a)}, p_k^{(b)}] := (I - x_k^{(1)} x_k^{(1)H})[p_k^{(a)}, p_k^{(b)}]$.

(d) Due to the same reason as in (c), the columns of R_k, in line 10 for the non-smooth F, are replaced by the s leading r_A and r_B vectors, assuming the block size $\ell = 2s$ is an even number.

(e) For a smooth function F, the reduced NEPv in line 14 can always be solved by the SCF iteration Algorithm 1. In the non-smooth case, convex optimization has to be employed. The exact algorithm will be problem dependent, but usually cheap to apply due to the small problem size; see, e.g., Section 6.2 for discussions about the max-ration minimization.

6 Numerical experiments

In this section, we illustrate the performance of the proposed algorithms with several numerical examples. The experiments are organized as follows. In Section 6.1, we discuss a p-norm distance problem of numerical range, with applications to coercivity constant computation for boundary element operators. In Section 6.2, we consider a non-differentiable max-ratio minimization problem. All numerical experiments are done in MATLAB 2017b, and run on an HP machine with Intel(R) Xeon(R) CPU E5–2620 v3 @ 2.40GHz, and 264 GB memory. No parallelization is applied.

6.1 Smooth objective function

Consider the minimization problem (1) with objective function given by $F(y) = \|y\|_p$ with $p > 1$. Geometrically, the minimizer can be regarded as the p-norm distance from the origin $(0, 0)$ to the given numerical range $W(A, B)$. For the special case of $p = 2$, it corresponds to the Crawford number problem mentioned in the introduction, where the problem also admits the eigenvalue optimization formula

$$\min_{y \in W(A,B)} \|y\|_2 = \left(\max \left\{ \max_{\theta \in (0, 2\pi]} \lambda_{\min}(A \sin\theta + B \cos\theta), \quad 0 \right\} \right), \tag{42}$$

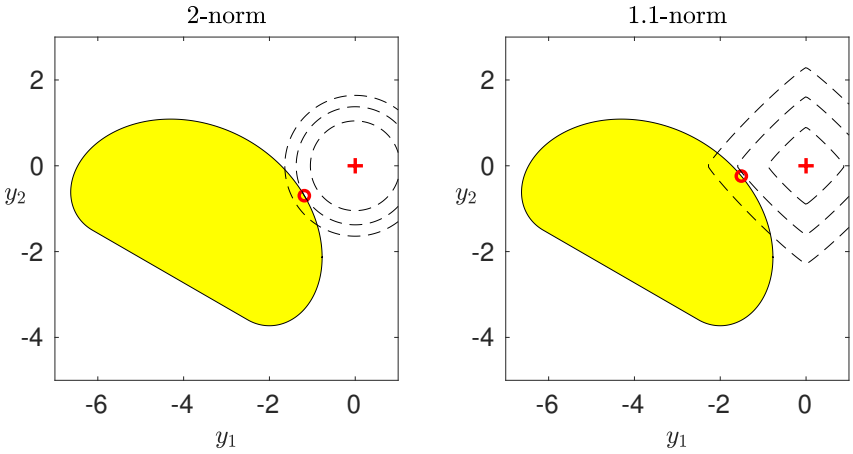

Figure 1: Numerical range and the p-norm distance to $(0, 0)$: 2-norm on the left and 1.1-norm on the right. The nearest points are marked as $\circ$, with dashed lines being the contour plot of F.

see, e.g., [9, Thm. 2.1] and [30, eq. (2.8)]. In our experiment, we will also consider a general $p = 1.1$ problem, where no eigenvalue optimization formula such as (42) is available. It is clear that F is convex and is differentiable for all $y \neq 0$. To apply Algorithm 4, we set the tolerance $\texttt{tol} = 10^{-8} \approx \sqrt{\epsilon_{mach}}$, and we solve the reduced NEPv in line 14 by SCF iteration Algorithm 1 with $\texttt{tol\_r} = \texttt{tol}$ and $\texttt{maxit} = 30$.

Example 1. This is a small size example to show the convergence of Algorithm 4 without preconditioned expansion steps. The testing matrices are given by

$$A = \cos(\pi/3)G - 4I_n, \qquad B = \sin(\pi/3)G - 2I_n,$$

where G is a Grcar matrix of size $n = 120$. The numerical range $W(A, B)$, and the contours of F corresponding to $p = 2$ and 1.1, are depicted in Fig. 1.

In the first experiment, we apply Algorithm 4 with block size $\ell = 1$ and randomly generated starting vectors (normally distributed elements with 0 mean and unit variance). The convergence history is reported in Fig. 2, where the relative error is measured by

$$|\mathcal{F}(x_k) - \mathcal{F}(x_*)|/|\mathcal{F}(x_*)|,$$

and the 'exact' minimizer x_* is computed by SCF iterations applied to the NEPv (8) directly.

For comparison, we also depict the convergence history of LOBPCG[3] for solving the Linear Eigenvalue Problem (LEP) in (31), where the relative error is measured by $|\lambda_k - \lambda_*|/|\lambda_*|$ with λ_* computed by MATLAB function `eig`. Note that this can be regarded as verification of global optimality for a given x_*; see, e.g., Section 3.4. From the reported result, Algorithm 4 solved an NEPv (8) in the number of iterations comparable to LOBPCG for a linear eigenvalue problem, and its convergence also seems less sensitive to the choice of initial vectors.

In our second experiment, we test with block size $\ell = 1, 2, 4, 8$. For each ℓ, the algorithms are repeatedly applied 200 times using random starting vectors. The computation result is reported in Fig. 3. We can observe a great reduction in the number of iterations and its variance as the block size ℓ grows, whereas the total number of MatVecs slightly increases since each iteration needs more matrix vector multiplications. The performance are also similar to LOBPCG for the LEP (31).

Example 2. We consider the preconditioned expansion schemes discussed in Section 3.3. For the testing, Algorithm 4 is applied to the same problem from above, with the *preconditioned expansion* in line 7 set on. In Fig. 4, we illustrate the superlinear convergence when (23) is applied, which is consistent with our theoretical

[3]MATLAB code available from `https://mathworks.com/matlabcentral/fileexchange/48-lobpcg-m`.

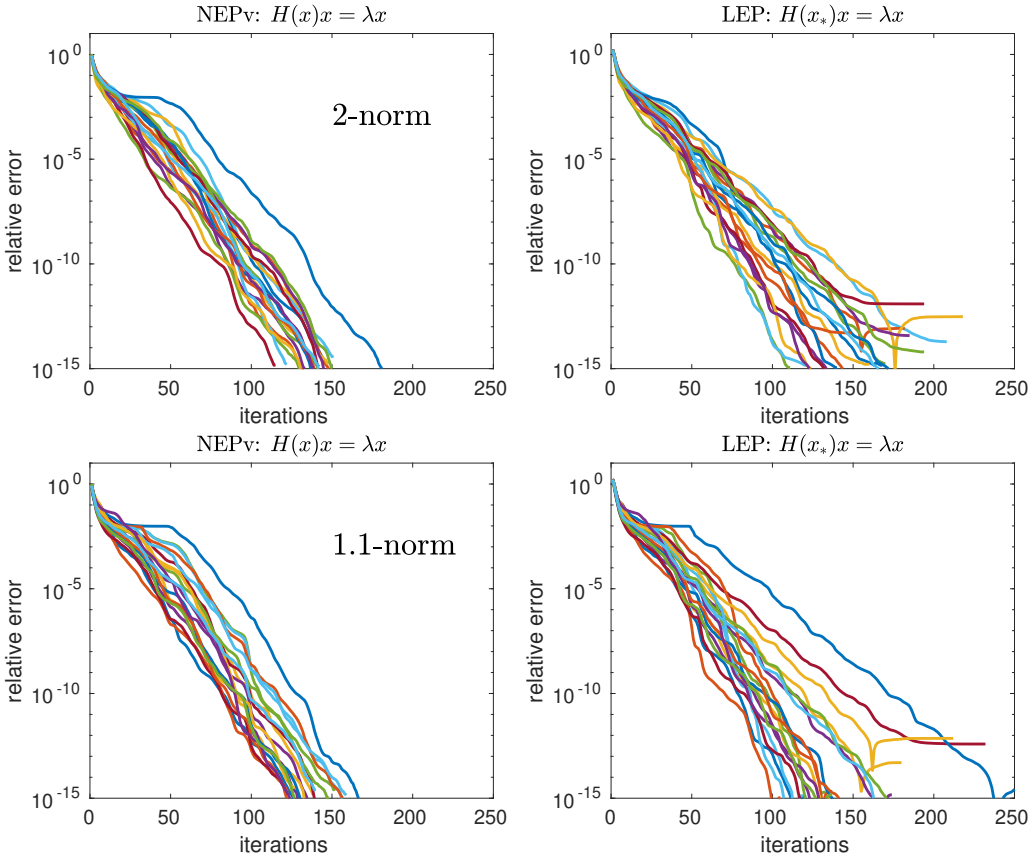

Figure 2: The p-norm distance problem in Example 1. Convergence history for 20 randomly generated starting vectors: (Left) Algorithm 4 for NEPv (8); (Right) LOBPCG for the LEP (31).

analysis in Theorem 3. In Fig. 5, we report the performance of inexactly expanding vectors, which are computed by a few steps of MATLAB `gmres` (only the 2-norm case is reported, the performance for $p = 1.1$ is similar). Both formulas (**??**) are tested. The former seems more affected by the error in the expanding vectors, but it also converges faster when their accuracy increase.

Example 3. In this example, we consider the problem of computing the coercivity constant for boundary integral operators in acoustic scattering [4, 31]. The major computation task is to evaluate

$$\gamma(L) := \min_{u\neq 0} \frac{|u^H L u|}{u^H u} = \min_{u\neq 0} \left(\left| \frac{u^H A u}{u^H u} \right|^2 + \left| \frac{u^H B u}{u^H u} \right|^2 \right)^{1/2},$$

where $A = (L + L^H)/2$ and $B = (L - L^H)/2\jmath$, and L is a discrete boundary integral operator (matrix) of

$$a_\nu(u, v) = \int_\Gamma B_\nu u(y) \cdot \overline{v(y)}\, \mathrm{d}s(y), \quad \text{with} \quad B_\nu = I + K_\nu - \jmath\nu S_\nu,$$

where $\nu > 0$ is the wave number, Γ is the boundary of a sound-soft bounded obstacle in $\mathbb{R}^3$, $u(x), v(x) \in L^2(\Gamma)$, I is the identity operator, and K_ν and S_ν are defined by

$$K_\nu u(x) = 2\int_\Gamma \frac{\partial \Phi(x,y)}{\partial n(x)} u(y)\, \mathrm{d}s(y), \qquad S_\nu u(x) = 2\int_\Gamma \Phi(x,y) u(y)\, \mathrm{d}s(y), \qquad x \in \Gamma.$$

Here, $\Phi(x, y) = e^{\jmath\nu|x-y|}/(2\pi|x - y|)$ for $x, y \in \mathbb{R}^3$, $x \neq y$, and $n(x)$ is the outpoint unit normal at Γ.

In the following test, we consider two different geometry of Γ shown in Fig. 6 (smooth and non-smooth), as well as three wavenumbers $\nu = 1, 2, 5$. Each coefficient matrix $L^{(\nu)}$ is generated by the Galerkin boundary

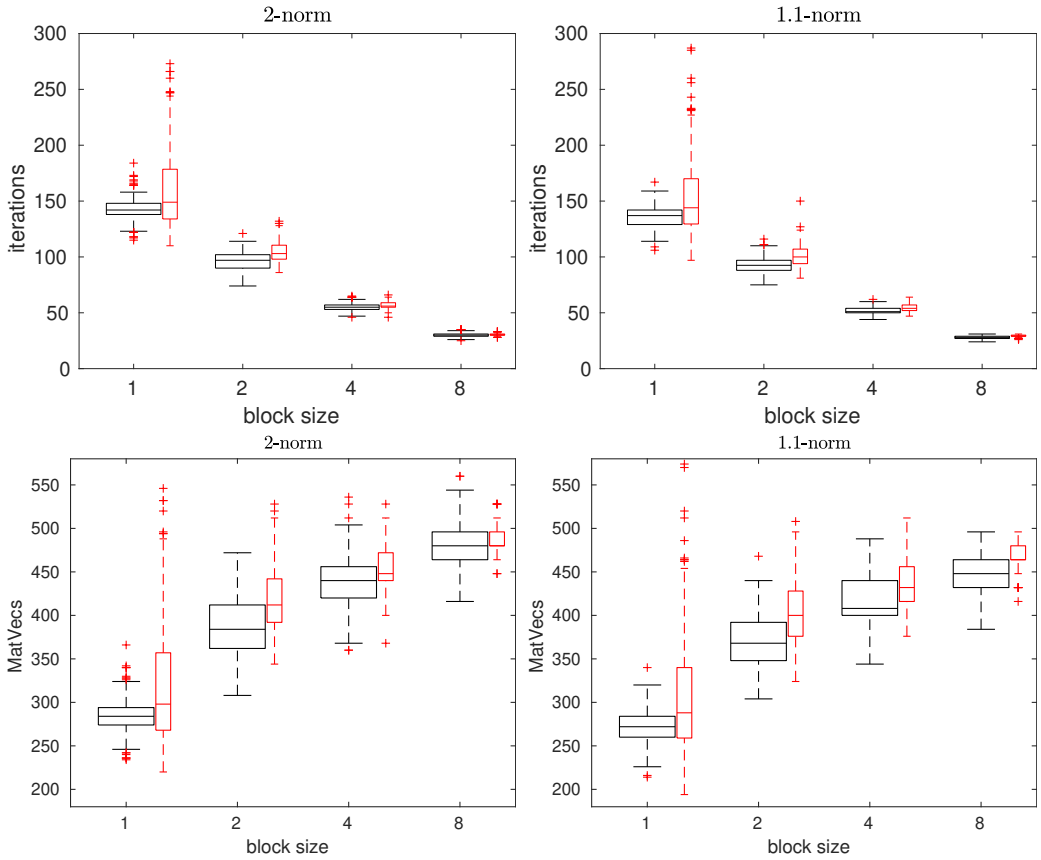

Figure 3: The p-norm distance problem in Example 1. Boxplot of number of iterations (top), and number of matrix vector multiplications (bottom) for Algorithm 4 (marked as black larger boxes), and LOBPCG (marked as red smaller boxes), with block size $p = 1, 2, 4, 8$. Statistics collected from 200 repeated experiments for each p with random starting vectors.

element library `BEM++` [4], with triangular mesh (generated by `gmsh` [5] with Delaunay's algorithm) and piecewise linear basis functions; see [32] for details about the discretization method. For convenience, we use a relatively coarse mesh size $h = 0.1$.

To solve the problem, we applied Algorithm 4 with block size $\ell = 3$. Both the standard and preconditioned version (labeled `pcond`) are tested, where the preconditioning formula (30) is applied with

$$T_k = (A_{\mathcal{H}} \cdot F_1(x_k) + B_{\mathcal{H}} \cdot F_2(x_k) - \lambda_k I)^{-1},$$

where $\lambda_k = (x_k^H H(x_k) x_k)/(x_k^H x_k)$, and $A_{\mathcal{H}}$ and $B_{\mathcal{H}}$ are hierarchical matrix approximation [33] of A and B (generated by `BEM++` using a truncation threshold 0.1). To reduce cost, we apply $T_k x$ by running 10 steps of GMRES instead of solving exactly. This can be done very efficiently since each $A_{\mathcal{H}} x$ (and $B_{\mathcal{H}} x$) takes around 10% of the timing of Ax (and Bx). We repeated the experiment 15 times with random starting vectors. The iteration number and timing statistics (mean and max deviation over the repeated experiments) are reported in Tables 1 and 2. The convergence history is depicted in Fig. 7.

For comparison, we have also applied the full subspace algorithm developed in [31], which is based on the eigenvalue optimization formula in (42). In each iteration, it solves an Hermitian linear eigenvalue problem for the smallest eigenvalue, which was done by LOBPCG [6] using a same block size $\ell = 3$ as Algorithm 4. For comparison, we have also applied LOBPCG using preconditioning (by T_k from above with λ_k replaced by an

[4] The software is available from `http://www.bempp.org/`

[5] The software is available from `http://gmsh.info/`

[6] For efficiency of the full subspace algorithm, we use the eigenvectors from the last iteration to initialize each call to LOBPCG. Since the inner eigenvalue problems need not be solved accurately, we start with a low tolerance 10^{-6} for LOBPCG and gradually increase it to 10^{-12} upon convergence of the full subspace algorithm.

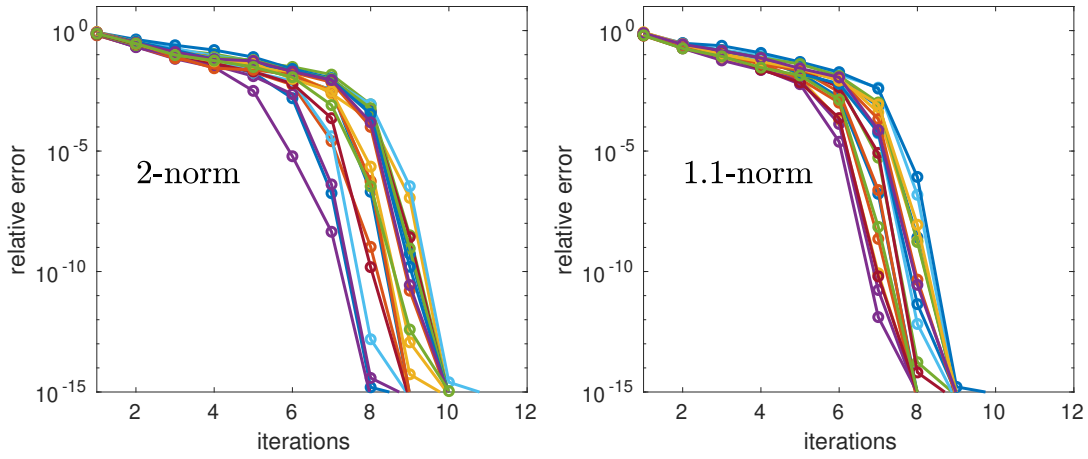

Figure 4: Convergence history of preconditioning scheme (23) for 20 random starting vectors.

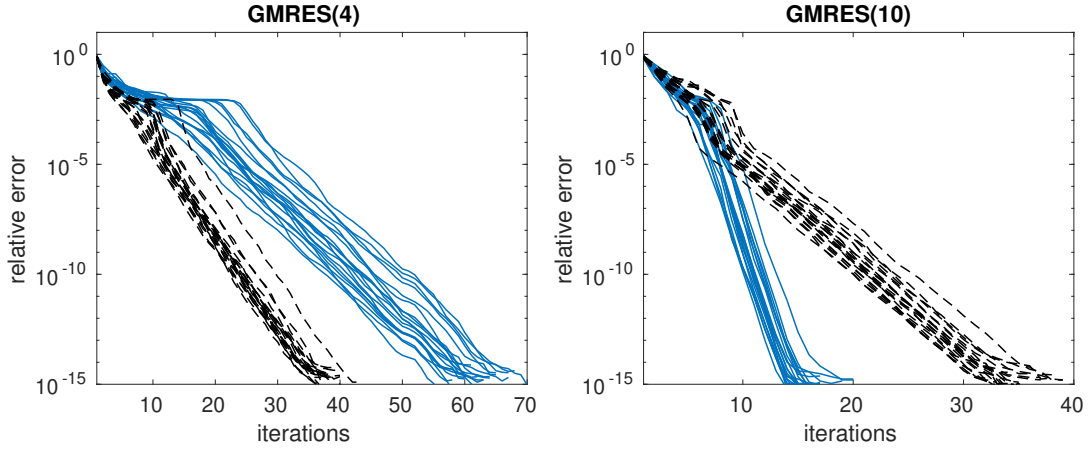

Figure 5: Convergence history of the preconditioned expansion schemes (23) (red solid) and (28) (blue dashed) using 'inexact' linear system solver, where expanding vectors are computed by running 4 and 10 steps of GMRES. Results were collected for the 2-norm problem, from 20 repeated experiments with random starting vectors.

underestimate of the Crawford number available in the full subspace algorithm). From Tables 1 and 2, the algorithms are significantly accelerated, although still slower than Algorithm 4 in the standard version. In the best case, the full subspace algorithm converges in less number of MatVecs than Algorithm 4 (without preconditioning), but each of its MatVec is more costly due to the extra preconditioning step.

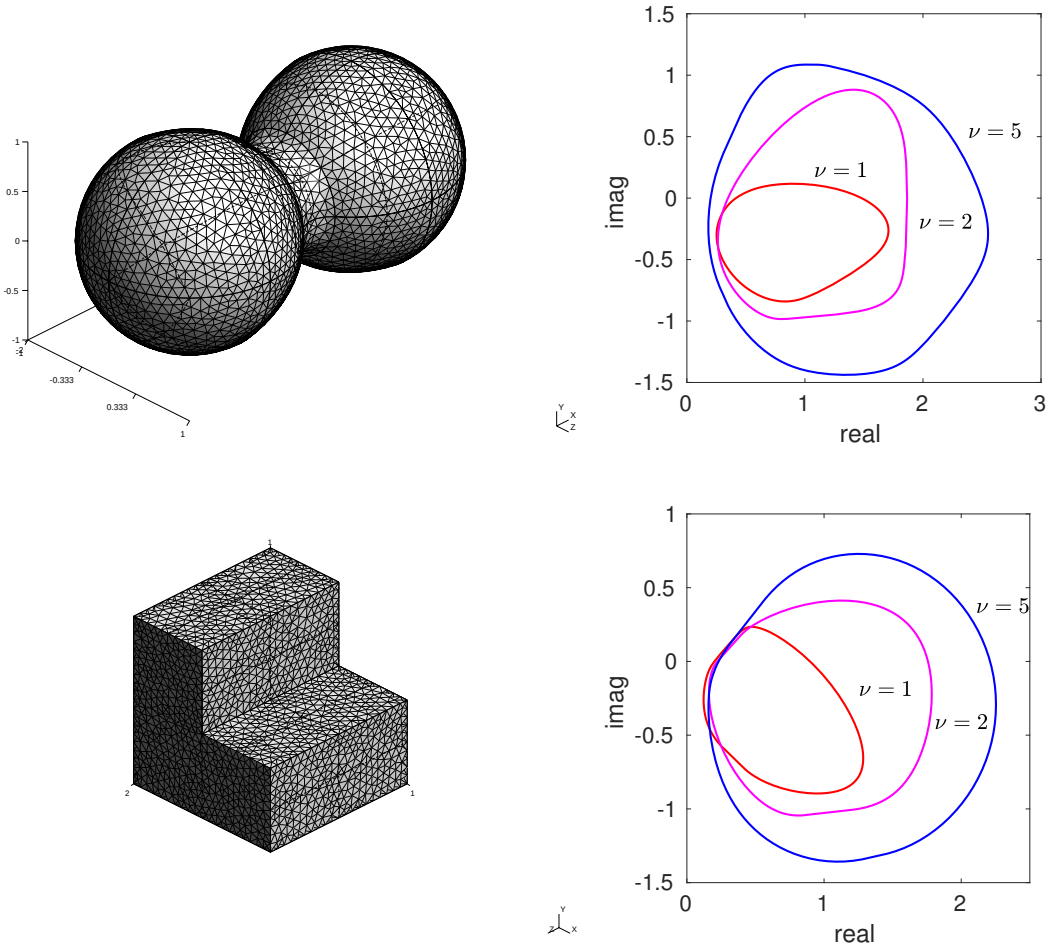

Figure 6: The geometry of Γ and the corresponding numerical range of the discrete operator $L^{(\nu)}$.

Table 1: Computation results for Example 3: peanut-shaped domain, size $n = 18,060$.

ν		$\gamma(L^{(\nu)})$	its	MatVec	timing (s)
1	Alg. 1 in [31]	3.231643542030261E-01	5 (LOBPCG)	1188	1810
	Alg. 1 in [31](pcond)	3.231643542030258E-01	5 (LOBPCG)	366	839
	Algorithm 4	3.231643542030256E-01 ($\pm 1 \cdot 10^{-15}$)	47 ($\pm$4)	286($\pm$28)	426 ($\pm$40)
	Algorithm 4 (pcond)	3.231643542030258E-01 ($\pm 3 \cdot 10^{-16}$)	15 ($\pm$4)	88($\pm$22)	175 ($\pm$40)
2	Alg. 1 in [31]	3.227569492274415E-01	5 (LOBPCG)	822	1277
	Alg. 1 in [31](pcond)	3.227569492274416E-01	5 (LOBPCG)	252	592
	Algorithm 4	3.227569492274416E-01 ($\pm 4 \cdot 10^{-16}$)	62 ($\pm$6)	378 ($\pm$36)	567 ($\pm$54)
	Algorithm 4 (pcond)	3.227569492274415E-01 ($\pm 3 \cdot 10^{-16}$)	15 ($\pm$2)	94 ($\pm$16)	182 ($\pm$23)
5	Alg. 1 in [31]	1.985483883517590E-01	6 (LOBPCG)	1032	597
	Alg. 1 in [31](pcond)	1.985483883517590E-01	6 (LOBPCG)	504	1168
	Algorithm 4	1.985483883517627E-01 ($\pm 3 \cdot 10^{-15}$)	56 ($\pm$6)	342($\pm$36)	503 ($\pm$61)
	Algorithm 4 (pcond)	1.985483883517589E-01 ($\pm 3 \cdot 10^{-16}$)	15 ($\pm$2)	93($\pm$12)	183 ($\pm$24)

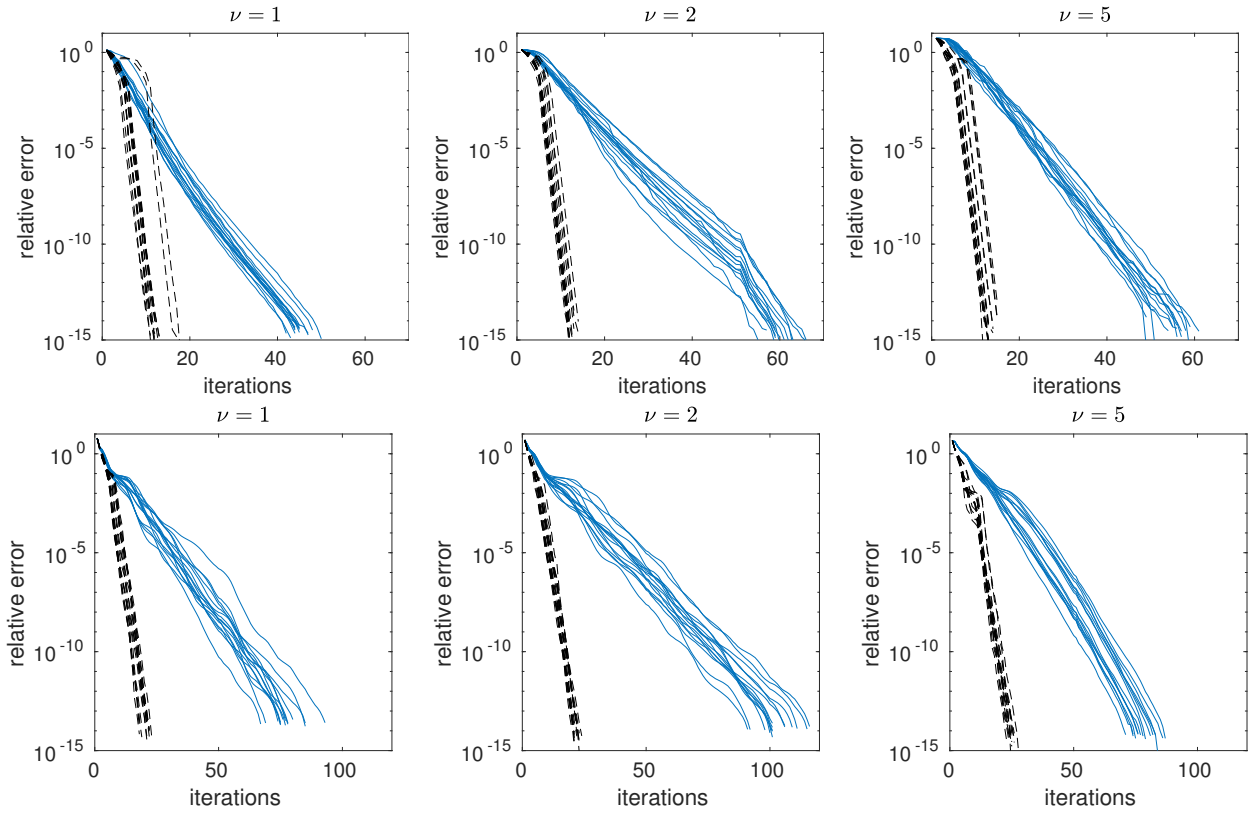

Figure 7: Example 3 with wave number $\nu = 1, 2, 5$. Convergence history of Algorithm 4 with block size $\ell = 3$ for peanut-shaped (top) and L-shaped (bottom) domains. Solid lines are for the standard version, and dashed for the preconditioned version. For relative error, we take the minimal eigenvalue of the 15 random runs as the 'exact' solution. Results were collected from 15 repeated experiments with random starting vectors.

Table 2: Computation results for Example 3: L-shaped domain, size $n = 16,116$.

ν		$\gamma(L^{(\nu)})$	its	MatVec	timing (s)
1	Alg. 1 in [31]	1.719991140659281E-01	6 (LOBPCG)	2562	3108
	Alg. 1 in [31](pcond)	1.719991140659281E-01	6 (LOBPCG)	474	907
	Algorithm 4	1.719991140659368E-01 ($\pm 7 \cdot 10^{-15}$)	81 ($\pm$13)	490 ($\pm$76)	588 ($\pm$89)
	Algorithm 4 (pcond)	1.719991140659283E-01 ($\pm 3 \cdot 10^{-16}$)	22 ($\pm$3)	130 ($\pm$16)	212 ($\pm$26)
2	Alg. 1 in [31]	1.950697449793223E-01	6 (LOBPCG)	3126	3767
	Alg. 1 in [31](pcond)	1.950697449793218E-01	6 (LOBPCG)	528	1022
	Algorithm 4	1.950697449793266E-01 ($\pm 3 \cdot 10^{-15}$)	104 ($\pm$13)	634 ($\pm$82)	762 ($\pm$95)
	Algorithm 4 (pcond)	1.950697449793223E-01 ($\pm 4 \cdot 10^{-16}$)	23 ($\pm$2)	138 ($\pm$12)	231 ($\pm$19)
5	Alg. 1 in [31]	2.100416346643941E-01	7 (LOBPCG)	1896	2334
	Alg. 1 in [31](pcond)	2.100416346643939E-01	7 (LOBPCG)	456	897
	Algorithm 4	2.100416346643952E-01 ($\pm 6 \cdot 10^{-16}$)	80 ($\pm$8)	486 ($\pm$48)	585 ($\pm$57)
	Algorithm 4 (pcond)	2.100416346643941E-01 ($\pm 3 \cdot 10^{-16}$)	27 ($\pm$3)	160 ($\pm$16)	267 ($\pm$25)

6.2 Non-smooth objective function

We consider the max-ratio minimization problem with $F(y) := \max\{y_1, y_2\}$, which is non-differentiable at y with $y_1 = y_2$. To apply Algorithm 4, the reduced problem in line 14 will be solved by the equivalent eigenvalue

optimization problem in the following lemma, for which a similar result can be found in [5], but we prove the optimizing domain is $[0,1]$ instead of $\mathbb{R}$.

Lemma 2. *Let A and B be Hermitian matrices.*

(a) It holds

$$\min_{x\in\mathbb{C}^n} \max\left\{\frac{x^HAx}{x^Hx}, \frac{x^HBx}{x^Hx}\right\} = \max_{t\in[0,1]} \lambda_{\min}\Big(tA+(1-t)B\Big). \tag{43}$$

(b) Let t_* be an optimizer of (43), and x_* be the eigenvector corresponding to the smallest eigenvalue λ_* of

$$H(x_*) := A(1-t_*) + Bt_*. \tag{44}$$

It holds that $\begin{bmatrix} t_* \\ 1-t_* \end{bmatrix} \in \partial F(\rho(x_*))$ for $F(y) := \max\{y_1, y_2\}$.

Proof. The proof is deferred to Appendix A. □

The eigenvalue optimization (43) is convex in $t \in \mathbb{R}$ [34], therefore, it can be conveniently solved, at least for problems with a small size n, by golden section search (e.g., the MATLAB function `fminbd`) or other eigenvalue optimization techniques (see, e.g., [35]). In our implementation, we adopt the level-set based criss-cross search [36] for the maximizer, which works quite well in the numerical experiments.

For testing problem, we consider a multicast transmit beamforming problem [6] in a simple setting that the base station sends a common signal to two receivers a and b using N antennas. Following the problem setting up as in [37, Sec C] we arrive at a quadratic optimization problem

$$\min_{w\in\mathbb{C}^N} w^Hw, \qquad \text{s.t. } w^HR_aw \geq \sigma_a^2\tau_a,\ w^HR_bw \geq \sigma_b^2\tau_b,$$

where τ_a, τ_b, σ_a, τ_b are prescribed parameters, and in the case the antenna array is linear and the receivers are located at θ_a and θ_b relative array broadside, the covariance matrices $R_i \in \mathbb{C}^{N\times N}$ are defined with (ℓ, p) elements

$$[R_i]_{\ell p} = \exp\left(\pi\jmath(\ell-p)\sin\theta_i\right)\cdot\exp\left(-\frac{(\pi(\ell-p)s_i\cos\theta_i)^2}{2}\right), \quad \text{for} \quad i=\{a,b\}, \tag{45}$$

where s_i is the spread angle of local scatterers for user i, see, e.g., [38] for details. For convenience, we use parameters $\theta_a = -5^\circ$ and $\theta_b = 10^\circ$, $s_i = 2^\circ$, and $\tau_i = 1/\sigma_i^2$ in our experiment.

By straightforward derivation, the quadratic optimization from above can be reformulated as [7]

$$\min_{w\in\mathbb{C}^n}\left(\max\left\{\frac{w^HAw}{w^Hw}, \frac{w^HBw}{w^Hw}\right\}\right) \quad \text{with} \quad A = -\frac{R_a}{\sigma_a^2\tau_a},\ B = -\frac{R_b}{\sigma_b^2\tau_b}. \tag{46}$$

Therefore, Algorithm 4 can be applied for the solution. For convenience, we use the block size $\ell = 2$ in the computation.

Example 4. In the first experiment, we consider a small size problem with $n = 120$. The Rayleigh quotients at the optimizer $\widehat{x}$, computed by solving the eigenvalue optimization (43) directly, are given by

$$\frac{\widehat{x}^HA\widehat{x}}{\widehat{x}^H\widehat{x}} = -11.27112794653678, \quad \frac{\widehat{x}^HB\widehat{x}}{\widehat{x}^H\widehat{x}} = -11.27112794653939.$$

Hence, $\widehat{x}$ is a numerically non-smooth point of $\mathcal{F}(x)$. The convergence history of Algorithm 4 is reported in Fig. 8, where for the relative error the 'exact' eigenvalue is computed using the eigenvalue optimization formula (43). For comparison we also applied LOBPCG (with a same block size of 2) for the smallest eigenvalue of the Hermitian matrix $H(x_*)$ at the optimizer in (44). Again we observe a similar convergence rates for both algorithms.

[7] Let $\gamma = -(w^Hw)^{-1}$, the minimization problem is equivalent to $\min\gamma$ s.t. $(w^HAw)/(w^Hw) \leq \gamma$ and $(w^HBw)/(w^Hw) \leq \gamma$.

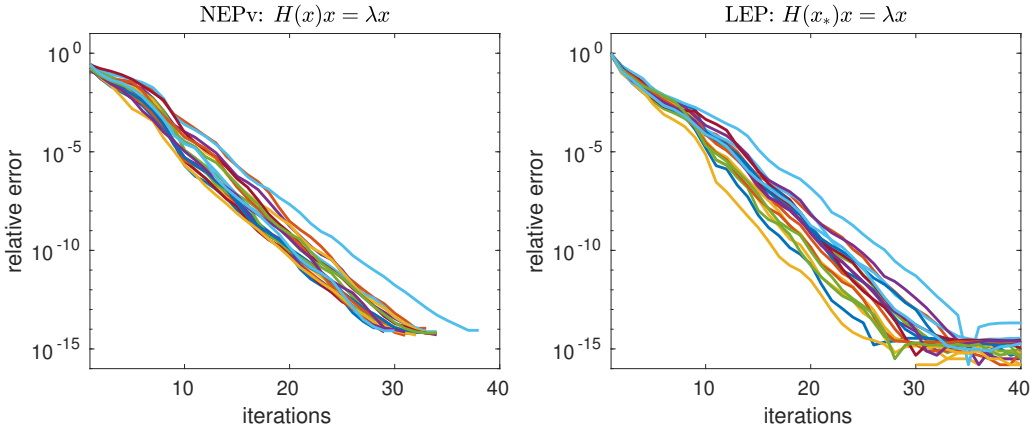

Figure 8: Convergence history for 20 randomly generated starting vectors: (Left) Algorithm 4 for the NEPv (8); (Right) LOBPCG for computing the smallest eigenvalue of $H(x_*)$ in (44).

Example 5. We now test with larger problem size $n = 1000, 2000, 4000$. In the experiment, we treat matrices A and B as linear operators, and only allow for matrix-vector (or matrix-matrix) multiplication in computation. The testing results for 20 repeated experiments with random starting vectors are reported in Table 3, with the convergence history depicted in Fig. 9. Recall that the max-ratio minimization problem can also be reformulated and solved as an eigenvalue optimization problem. For comparison, we applied the `leigopt`[8] to (43) (with convergence tolerance set to `tol` $= 10^{-13}$). This algorithm is based on the subspace framework presented in [39], and in each iteration it needs to solve a large-scale linear eigenvalue problem, which is done by the MATLAB function `eigs`.

In all testing cases, Algorithm 4 used less number of MatVecs, which consists the dominant cost of both algorithms as the problem size n grows. For `leigopt`, the percentages of time spent for MatVecs are about 72%, 84% and 90%, for $n = 1000$, 2000, 4000, respectively. From Table 3, we can also observe that the saving of Algorithm 4 in the computing time is more than that in the number of MatVecs. Take $n = 2000$ for example, the ratio between the MatVecs of Algorithm 4 and `leigopt` is $1770/3082 \approx 0.57$, whereas that for the computation time is $7.7/23.3 \approx 0.33$. This difference is largely because of the block operations: The 1770 MatVecs (on average) in Algorithm 4 are executed as approximate $1770/2 = 885$ number of matrix-matrix multiplications $A \cdot R_k$ and $B \cdot R_k$, rather than 1770 sequential matrix-vector multiplications. On a hierarchical memory machine, the former multiplication is more efficient since it uses level-3 BLAS operation. Such block operation is, however, not exploited by the MATLAB function `eigs` in `leigopt`.

Table 3: Computation results for Example 4.

size n		optimal value	its	MatVec	timing (s)
1000	leigopt	-1.15337555620605E+01	6 (eigs)	1772	3.0
	Algorithm 4	-1.15337555620603E+01 ($\pm 1 \cdot 10^{-13}$)	229 ($\pm$87)	903 ($\pm$357)	1.4 ($\pm$0.5)
2000	leigopt	-1.15372647515872E+01	6 (eigs)	3082	23.3
	Algorithm 4	-1.15372647515869E+01 ($\pm 4 \cdot 10^{-13}$)	435 ($\pm$81)	1770 ($\pm$330)	7.7 ($\pm$1.7)
4000	leigopt	-1.15381560642041E+01	7 (eigs)	5760	159.8
	Algorithm 4	-1.15381560642033E+01 ($\pm 1 \cdot 10^{-12}$)	809 ($\pm$258)	3295 ($\pm$1053)	49.7 ($\pm$16.3)

[8]Available at `http://home.ku.edu.tr/~emengi/software.html`. We slightly modified the calling of `eigs` function to accept linear operators as input.

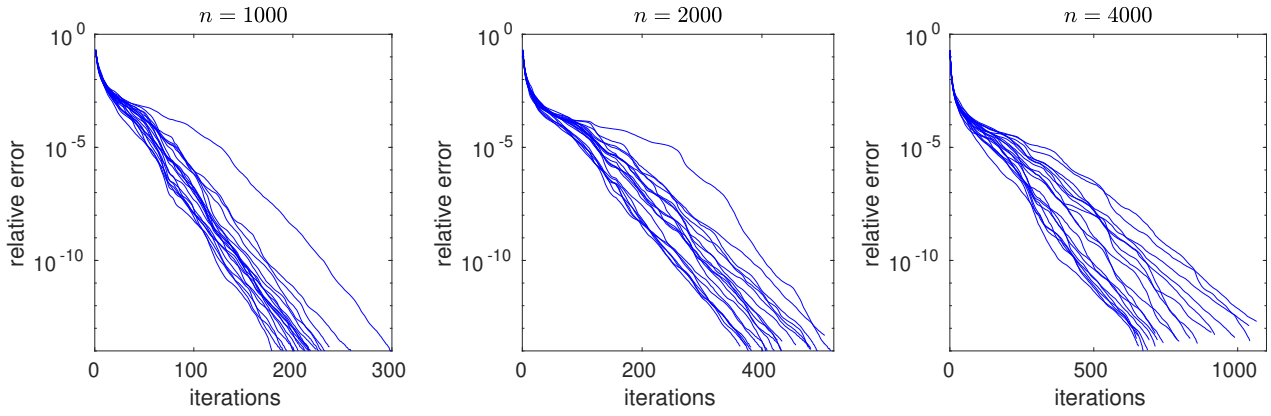

Figure 9: Convergence history of Example 4 with 20 randomly generated starting vectors. For relative error, we take the minimal eigenvalue of the 20 runs as the 'exact' solution.

7 Conclusions

In this paper, we considered the convex minimization problem over the joint numerical range of a pair of Hermitian matrices. For such problems, a nonlinear eigenvalue problem characterization of the global optimizer has been established. Iterative methods based on locally optimal subspace search was proposed, with both smooth and non-smooth objective function considered. The convergence of the algorithms, as well as their implementation details, were also discussed. The effectiveness and efficiency of the proposed nonlinear eigenvector approach were demonstrated by numerical examples for computing the coercivity constant of boundary integral operators and solving multicast beamforming problems.

The theory and algorithms considered in this paper can be naturally extended to convex minimization over the joint numerical range of a d-tuple of Hermitian matrix $A_i \in \mathbb{C}^{n\times n}$ for $i = 1, \ldots, d$:

$$W(A_1, A_2, \ldots, A_d) = \left\{ \left(x^H A_1 x, x^H A_2 x, \ldots, x^H A_d x\right) : x \in \mathbb{C}^n,\ \|x\|_2 = 1 \right\},$$

provided that the considered joint numerical range is a convex set in $\mathbb{R}^d$. Such an assumption holds in particular in the case of $d = 3$ and $n \geq 3$ as shown by [40], while for more general cases of $d \geq 3$, the convexity can only hold in certain conditions (see, e.g., [41, 1]). Under the convexity assumption, most of the results in this paper can be applied, but a detailed technical treatment is beyond the scope of this paper and is left for future research.

A Proof of Lemma 2

In this section, we provide proof of Lemma 2 for the max–ratio problem, for which a similar result has been presented in [5], where the optimizing parameter t is on $\mathbb{R}$. By exploiting the convexity of the numerical range $W(A, B)$, we can simplify the proof and obtain a bounded region for the parameter t, which makes the optimization problem more tractable in practice.

(a) We can reformulate the left hand side as

$$\min_{x\in\mathbb{C}^n} \max\left\{\frac{x^H A x}{x^H x}, \frac{x^H B x}{x^H x}\right\} = \min_{y\in W(A,B)} \max\left\{y_1,\ y_2\right\} = \min_{y\in W(A,B)} \max_{t\in[0,1]}\ t y_1 + (1-t) y_2.$$

Note that $W(A, B)$ is convex, and $f(t, y) = ty_1 + (1 - t)y_2$ is linear in t and y, respectively. By von Neumann's minimax theorem the min and max in the equation above can switch position, namely

$$\min_{y\in W(A,B)} \max_{t\in[0,1]}\ f(t,y) = \max_{t\in[0,1]} \min_{y\in W(A,B)} f(t,y). \tag{47}$$

The inner minimization satisfies

$$\min_{y\in W(A,B)} f(t,y) = \min_{x\in\mathbb{R}^n} \frac{x^T\left(tA+(1-t)B\right)x}{x^Tx} = \lambda_{\min}\Big(tA+(1-t)B\Big),$$

where in the last equation we applied eigenvalue minimization principle for Hermitian definite matrix pair.

(b) Due to the minimax relation (47) from above, we have $(t_*, \rho(x_*))$ to be an equilibrium point of the minimax problem. Therefore, it holds

$$f(t_*,\rho(x_*)) = \min_{y\in W(A,B)} \max_{t\in[0,1]} f(t,y) = \max_{t\in[0,1]} f(t,\rho(x_*)) = F(\rho(x_*)).$$

It is straightforward to verify that for all y it holds

$$F(y) = \max_{t\in[0,1]} f(t,y) \geq f(t_*,y) = f(t_*,\rho(x_*)) + f(t_*, y-\rho(x_*)) = F(\rho(x_*)) + [t_*,(1-t_*)]\cdot(y-\rho(x_*)).$$

Hence, by the definition of subgradient it holds $[t_*,(1-t_*)]^T \in \partial F(\rho(x_*))$.

References

[1] E. Gutkin, E. A Jonckheere, and M. Karow. Convexity of the joint numerical range: topological and differential geometric viewpoints. *Linear Algebra Appl.*, 376:143–171, 2004.

[2] C. R. Crawford. A stable generalized eigenvalue problem. *SIAM J. Numer. Anal.*, 13(6):854–860, 1976.

[3] G. W. Stewart. Pertubation bounds for the definite generalized eigenvalue problem. *Linear Algebra Appl.*, 23:69–85, 1979.

[4] T. Betcke and E. A. Spence. Numerical estimation of coercivity constants for boundary integral operators in acoustic scattering. *SIAM J. Numer. Anal.*, 49(4):1572–1601, 2011.

[5] D. Dileep Gaurav and K. V. S. Hari. A fast eigen solution for homogeneous quadratic minimization with at most three constraints. *IEEE Signal Process. Lett.*, 20(10):968–971, 2013.

[6] N. D. Sidiropoulos, T. N. Davidson, and Z.-Q. Luo. Transmit beamforming for physical-layer multicasting. *IEEE Trans. Signal Process.*, 54(6):2239–2251, 2006.

[7] C. R. Johnson. Numerical determination of the field of values of a general complex matrix. *SIAM J. Numer. Anal.*, 15(3):595–602, 1978.

[8] Frank Uhlig. On computing the generalized Crawford number of a matrix. *Linear Algebra Appl.*, 438(4):1923–1935, 2013.

[9] S. H. Cheng and N. J. Higham. The nearest definite pair for the Hermitian generalized eigenvalue problem. *Linear Algebra Appl.*, 302:63–76, 1999.

[10] A. V. Knyazev. A preconditioned conjugate gradient method for eigenvalue problems and its implementation in a subspace. In *Numerical Treatment of Eigenvalue Problems Vol. 5/Numerische Behandlung von Eigenwertaufgaben Band 5*, pages 143–154. Springer, 1991.

[11] A. V. Knyazev. Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.*, 23(2):517–541, 2001.

[12] A. Stathopoulos and J. R. McCombs. PRIMME: preconditioned iterative multimethod eigensolver–methods and software description. *ACM Trans. Math. Softw.*, 37(2):21, 2010.

[13] R.-C. Li. Rayleigh quotient based optimization methods for eigenvalue problems. In *Matrix Functions and Matrix Equations*, pages 76–108. World Scientific, 2015.

[14] D.H. Brandwood. A complex gradient operator and its application in adaptive array theory. In *IEE Proceedings H–Microwaves, Optics and Antennas*, volume 130, pages 11–16. IET, 1983.

[15] F. H. Clarke. *Optimization and nonsmooth analysis*, volume 5. SIAM, 1990.

[16] J. V. Burke. Second order necessary and sufficient conditions for convex composite NDO. *Math. Program.*, 38, 1987.

[17] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

[18] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I: Fundamentals*, volume 305. Springer-Verlag Berlin Heidelberg, 1993.

[19] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, 2nd edition, 2006.

[20] C. Le Bris. Computational chemistry from the perspective of numerical analysis. *Acta Numer.*, 14:363–444, 2005.

[21] R. Meyer. Nonlinear eigenvector algorithms for local optimization in multivariate data analysis. *Linear Algebra Appl.*, 264:225–246, 1997.

[22] L.-H. Zhang. On optimizing the sum of the Rayleigh quotient and the generalized Rayleigh quotient on the unit sphere. *Comput. Opt. Appl.*, 54(1):111–139, 2013.

[23] Z. Bai, D. Lu, and B. Vandereycken. Robust rayleigh quotient minimization and nonlinear eigenvalue problems. *SIAM J. Sci. Comput.*, 40(5):A3495–A3522, 2018.

[24] B. N. Parlett. *The Symmetric Eigenvalue Problem*, volume 20. SIAM, 1998.

[25] D. P. Bertsekas. *Nonlinear programming*. Athena Scientific Belmont, 1999.

[26] P.-A. Absil and K. A. Gallivan. Accelerated line-search and trust-region methods. *SIAM J. Numer. Anal.*, 47(2):997–1018, 2009.

[27] W. Hager. Minimizing a quadratic over a sphere. *SIAM J. Optim.*, 12(1):188–208, 2001.

[28] G. Narkiss and M. Zibulevsky. Sequential subspace optimization method for large-scale unconstrained problems. Technical report, EE Dept., Technion, Haifa, Israel, September 2005. CCIT No 559.

[29] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the solution of algebraic eigenvalue problems: a practical guide*. SIAM, 2000.

[30] N. J. Higham, F. Tisseur, and P. M. Van Dooren. Detecting a definite hermitian pair and a hyperbolic or elliptic quadratic eigenvalue problem, and associated nearness problems. *Linear Algebra Appl.*, 351:455–474, 2002.

[31] D. Kressner, D. Lu, and B. Vandereycken. Subspace acceleration for the Crawford number and related eigenvalue optimization problems. *SIAM J. Matrix Anal. Appl.*, 39(2):961–982, 2018.

[32] W. Śmigaj, T. Betcke, S. Arridge, J. Phillips, and M. Schweiger. Solving boundary integral problems with BEM++. *ACM Trans. Math. Softw.*, 41(2):6:1–6:40, 2015.

[33] W. Hackbusch. *Hierarchical Matrices: Algorithms and Analysis*, volume 49. Springer, 2015.

[34] M. L. Overton. On minimizing the maximum eigenvalue of a symmetric matrix. *SIAM J. Matrix Anal. Appl.*, 9(2):256–268, 1988.

[35] E. Mengi, E. A. Yildirim, and M. Kilic. Numerical optimization of eigenvalues of Hermitian matrix functions. *SIAM J. Matrix Anal. Appl.*, 35(2):699–724, 2014.

[36] S. Boyd and V. Balakrishnan. A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its l_∞-norm. *Syst. Control Lett.*, 15(1):1–7, 1990.

[37] Y. Huang and D. P. Palomar. Randomized algorithms for optimal solutions of double-sided QCQP with applications in signal processing. *IEEE Trans. Signal Process.*, 62(5):1093–1108, 2014.

[38] M. Bengtsson and B. Ottersten. Optimal downlink beamformingusing semidefinite optimization. In *37th Annual Allerton Conference on Communication, Control, and Computing*, pages 987–996, 1999.

[39] F. Kangal, K. Meerbergen, E. Mengi, and W. Michiels. A subspace method for large-scale eigenvalue optimization. *SIAM J. Matrix Anal. Appl.*, 39(1):48–82, 2018.

[40] M. K. H. Fan and A L Tits. On the generalized numerical range. *Linear Multilinear Algebra*, 21(3):313–320, 1987.

[41] C.-K. Li and Y.-T. Poon. Convexity of the joint numerical range. *SIAM J. Matrix Anal. Appl.*, 21(2):668–678, 2000.