ELSEVIER Linear Algebra and its Applications 351–352 (2002) 117–145

LINEAR ALGEBRA AND ITS APPLICATIONS

www.elsevier.com/locate/laa

Two numerical methods for optimizing matrix stability

James V. Burke[a,1], Adrian S. Lewis[b,2], Michael L. Overton[c,*,3]

[a] *Department of Mathematics, University of Washington, Seattle, Washington, USA*
[b] *Department of Mathematics, Simon Fraser University, Burnaby, British Columbia, Canada*
[c] *Computer Science Department, Courant Institute of Mathematical Science, New York University, New York, USA*

Received 24 March 2001; accepted 27 November 2001

Submitted by V. Blondel

Abstract

Consider the affine matrix family $A(x) = A_0 + \sum_{k=1}^{m} x_k A_k$, mapping a design vector $x \in \mathbb{R}^m$ into the space of $n \times n$ real matrices. We are interested in the question of how to choose x to optimize the stability of the matrix $A(x)$. A typical motivation is that one wishes to control the stability of the dynamical system $\dot{z} = A(x)z$. A classic example is stabilization by output feedback. We take two approaches. Let $\alpha(X)$ denote the spectral abscissa (the largest real part of the eigenvalues) of a matrix X; as is well known, this quantity bounds the asymptotic decay rate of the trajectories of the associated dynamical system $\dot{z} = Xz$. Our first approach to optimizing stability is to directly minimize the function $\alpha(A(x))$. The spectral abscissa $\alpha(X)$ is a continuous but non-smooth, in fact non-Lipschitz, function of the matrix argument X, and finding a global minimizer of $\alpha(A(x))$ is hard. We introduce a novel random gradient bundle method for approximating *local* minimizers, motivated by recent work on non-smooth analysis of the function $\alpha(X)$. Our second approach is to minimize a related function $\alpha_\delta(A(x))$, where δ is a *robustness* parameter in $(0, 1)$. One motivation for the definition of the "robust spectral abscissa" $\alpha_\delta(X)$ is that it bounds transient peaks as well as asymptotic decay of trajectories of $\dot{z} = Xz$. The function $\alpha_\delta(X)$ is Lipschitz but non-convex for $\delta \in (0, 1)$,

* Corresponding author. Tel.: +212-998-3121; fax: +212-995-4121.

E-mail address: burke@math.washington.edu (J.V. Burke), aslewis@math.sfu.ca (A.S. Lewis), overton@cs.nyu.edu (M.L. Overton).

[1] Supported by U.S. National Science Foundation Grant DMS-9971852.

[2] Supported by Natural Sciences and Engineering Research Council of Canada.

[3] Supported by U.S. National Science Foundation Grant CCR-0098145 and U.S. Department of Energy contract DE-FG02-98ER25352.

0024-3795/02/$ - see front matter © 2002 Elsevier Science Inc. All rights reserved.
PII: S0024-3795(02)00260-4

approaching $\alpha(X)$ as $\delta \to 0$ and the largest eigenvalue of $\frac{1}{2}(X + X^{\mathrm{T}})$ as $\delta \to 1$. We use a Newton barrier method to approximate local minimizers of $\alpha_\delta(A(x))$. We compare the results of the two approaches on a number of interesting test cases. © 2002 Elsevier Science Inc. All rights reserved.

Keywords: Spectral abscissa; Robust spectral abscissa; Optimal stability; Robust stability; Eigenvalue optimization; Non-Lipschitz optimization; Nonlinear semidefinite program; Gradient bundle; Newton barrier

1. The spectral abscissa

The spectral abscissa α of a square matrix is the maximum of the real parts of its eigenvalues. It plays a crucial role in the asymptotic analysis of dynamical systems. Consider the system $\dot{z}(t) = Xz(t)$. As is well known, the trajectory norm $\|z(t)\| = \|e^{tX}z(0)\|$ is bounded asymptotically by $e^{\gamma t}$, that is, $e^{-\gamma t}\|z(t)\| \to 0$ as $t \to \infty$, for any γ greater than $\alpha(X)$, the spectral abscissa of X. In particular, if $\alpha(X) < 0$, the system is asymptotically stable.

Consider the affine matrix family

$$A(x) = A_0 + \sum_{k=1}^{m} x_k A_k, \tag{1}$$

mapping a design vector $x \in \mathbb{R}^m$ into $\mathbf{M}^n(\mathbb{R})$, the space of $n \times n$ real matrices. We are interested in choosing x to optimize stability of the matrix $A(x)$. A typical motivation is to control the stability of the dynamical system $\dot{z}(t) = A(x)z(t)$. If we focus exclusively on asymptotic stability, we have a *spectral abscissa minimization* problem: find a vector $\bar{x}$ that minimizes $\alpha(A(x))$, locally or globally.

For example, the differential equation describing a damped linear oscillator is

$$\ddot{w}(t) + \xi\dot{w}(t) + w(t) = 0.$$

Here $\xi \in \mathbb{R}$ is the parameter that controls damping; when $\xi = 0$, the solutions are pure oscillations. The equivalent first-order system in the vector $z = [w, \dot{w}]^{\mathrm{T}}$ is

$$\dot{z}(t) = A(\xi)z(t), \quad \text{where } A(\xi) = \begin{bmatrix} 0 & 1 \\ -1 & -\xi \end{bmatrix}. \tag{2}$$

The trajectory norm $\|z(t)\|$ is bounded asymptotically by $e^{\gamma t}$ for any γ greater than the spectral abscissa of $A(\xi)$,

$$\alpha(A(\xi)) = \begin{cases} -\frac{\xi}{2} & \text{if } |\xi| \leqslant 2, \\ -\frac{\xi}{2} + \sqrt{\frac{\xi^2}{4} - 1} & \text{if } |\xi| \geqslant 2. \end{cases} \tag{3}$$

This is minimized by $\bar{\xi} = 2$, yielding $\alpha(A(\bar{\xi})) = -1$. Note that $A(\bar{\xi})$ has a non-derogatory double eigenvalue -1. A *non-derogatory* eigenvalue is one whose geometric multiplicity is one, i.e., the eigenvalue is associated with a single Jordan block.

By contrast, a *semisimple* eigenvalue is one whose geometric multiplicity equals its algebraic multiplicity, i.e., the eigenvalue has no Jordan blocks with size two or more.

Another interesting example is

$$A(x) = \begin{bmatrix} -x_1 & 1 & 0 & \cdot & \cdot & 0 \\ x_1 & 0 & 1 & 0 & \cdot & 0 \\ x_2 & 0 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\ x_{n-1} & 0 & \cdot & \cdot & \cdot & 0 \end{bmatrix}. \tag{4}$$

Here $m = n - 1$. Fig. 1 shows the contours of the spectral abscissa $\alpha(A(x))$ in the case $n = 3$ ($m = 2$). The spectral abscissa is minimized by $\bar{x}_1 = \bar{x}_2 = 0$, with $A(\bar{x})$ a Jordan block, with the non-derogatory triple eigenvalue zero. A manifold is clearly visible on which the spectral abscissa is non-smooth; on this manifold, which is tangent to the x_1-axis at $(0, 0)$, $A(x)$ has a non-derogatory double eigenvalue (except at $(0, 0)$ where it is triple). As x_2 is increased above 0, keeping $x_1 = 0$, the triple eigenvalue splits into three distinct eigenvalues, a real eigenvalue in the right half plane and a complex conjugate pair in the left, and the spectral abscissa accordingly increases sharply (the growth is $O(x_2^{1/3})$). When x_2 is decreased below 0, the growth in the spectral abscissa is still $O(|x_2|^{1/3})$, but with asymptotic factor reduced by half; this is because the real eigenvalue is in the left half plane and the complex conjugate pair is in the right half plane. On the other hand, when x_2 is fixed equal to 0 and

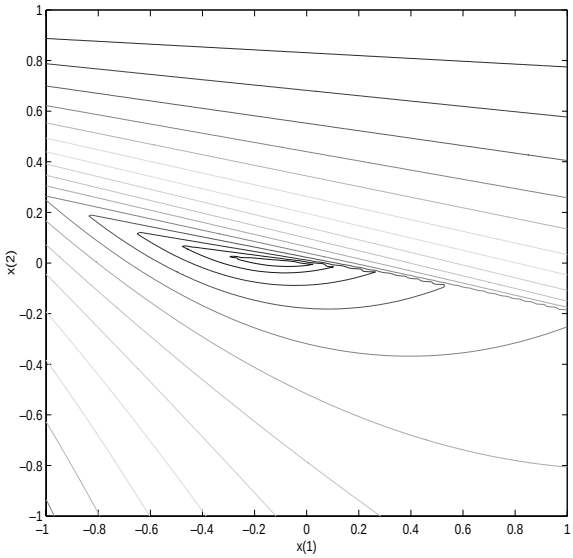

Fig. 1. Contours of the composite function $\alpha(A(x))$, where α denotes spectral abscissa and the affine map A is defined by (4) with $n = 3$.

x_1 is varied, the growth in α is much slower, $O(|x_1|^{1/2})$, though still non-Lipschitz. This simple example illustrates how challenging spectral abscissa minimization is in general, even locally.

More complicated spectral abscissa minimization problems often arise in control applications. A particularly important example (described in [11] as "perhaps the most basic problem in control theory") is *stabilization by output feedback*: given an $n \times n$ matrix $\bar{A}$, an $n \times r$ matrix $\bar{B}$ and an $s \times n$ matrix $\bar{C}$, find (if possible) an $r \times s$ matrix K such that $\bar{A} + \bar{B}K\bar{C}$ is stable, i.e., has all its eigenvalues in the left half of the complex plane. Efficient algorithms are known only for very special cases of this problem, such as $r = n$, $\bar{B} = I$ or $s = n$, $\bar{C} = I$, when the problem reduces to standard pole placement [3]. However, in general, this problem is hard. When interval bounds on the entries of K are specified, the general problem is known to be NP-hard [11,23]. The complexity of the general problem without bounds on K is not known. Since the map $K \mapsto \bar{A} + \bar{B}K\bar{C}$ is affine in K, stabilization by output feedback can be expressed as a spectral abscissa global minimization problem. In a 1995 survey [5], experts in control theory described the characterization of triples $(\bar{A}, \bar{B}, \bar{C})$ such that a stabilizing K exists as a "major open problem". Although we do not expect to devise an efficient algorithm to find global spectral abscissa minimizers, it is clear that reliable and efficient methods to find local minimizers could be of great value in practice. In the past, such methods have not been studied systematically; optimality conditions were not known, and though methods for moving eigenvalues left in the complex plane have been proposed from time to time, they have generally been fairly ad hoc, and usually assume that the eigenvalues in question are simple or perhaps double. The related control literature is huge; besides the surveys already cited, we mention [15,25].

Other spectral functions besides the spectral abscissa arise in practice, particularly the spectral radius, in connection with discrete time systems. For simplicity, in this paper, we confine our attention to the spectral abscissa, but all the ideas extend easily to the spectral radius. An interesting recent application, to which we are currently applying our optimization techniques, may be found in [33].

We say that an eigenvalue of a matrix X is *active* if its real part equals the spectral abscissa $\alpha(X)$. It is not hard to see that the spectral abscissa map

$$X \in \mathbf{M}^n(\mathbb{R}) \mapsto \alpha(X) \in \mathbb{R} \tag{5}$$

is non-Lipschitz at all $\bar{X}$ for which $\bar{X}$ has a non-semisimple active eigenvalue. Typically, the restricted map $\alpha \circ A$, i.e.,

$$x \in \mathbb{R}^m \mapsto \alpha(A(x)) \in \mathbb{R}, \tag{6}$$

inherits the non-Lipschitz nature of (5). Therefore, even if we were presented with a minimizer $\bar{x}$, verification of local optimality requires non-standard analytical tools. These have been developed at length in [10], using techniques from modern nonsmooth analysis. Although we do not discuss these results in the present work, they motivate a good deal of its development. In particular we remark that the key result in [10] is that the map (5) is subdifferentially regular [27] at $\bar{X}$ if and only if all active

eigenvalues of $\bar{X}$ are non-derogatory. It is not a coincidence that the spectral abscissa minimizers for $A(x)$ defined by (2) and (4) are both associated with non-derogatory multiple eigenvalues. On the contrary, this is typical; minimizing the spectral abscissa tends to make eigenvalues coalesce, and, generically, one expects any such multiple eigenvalues to be non-derogatory. Furthermore, with m variables, one generically expects eigenvalues to have multiplicity at most $m+1$. These phenomena are discussed and analyzed in [8,7].

Section 3 presents an algorithm that can be used to find local minimizers of the spectral abscissa $\alpha(A(x))$. The technique employed is a novel one using randomly generated bundles of gradients. Numerical results are presented in Section 5, showing that the random gradient bundle method is a very effective way to find local minimizers, despite the non-Lipschitz nature of the problem. We defer consideration of convergence analysis for this algorithm to future work, but we have addressed some underlying theoretical questions about random gradient bundles in [9].

2. The robust spectral abscissa

Although pure spectral abscissa minimization is very interesting from both a mathematical and a computational point of view, practical stability questions are often more involved. In particular, asymptotic stability of dynamical systems is far from the whole story; transient growth can be as undesirable as asymptotic instability. Techniques for balancing transient and asymptotic behavior of systems have been discussed for decades and many approaches are possible. In this section, we focus on one that has the appeal of being very clean and well motivated.

The following fundamental result characterizes the spectral abscissa of a general square matrix as an optimization problem over the positive definite matrices, a convex cone in $\mathbf{S}^n$, the space of real symmetric matrices. Let $P \succ 0$ denote the condition that $P \in \mathbf{S}^n$ is positive definite. Then, for any square matrix $X \in \mathbf{M}^n(\mathbb{R})$,

$$\alpha(X) = \inf_{P\in\mathbf{S}^n,\, P\succ 0} \frac{1}{2}\lambda_{\max}\Big(P^{1/2} X P^{-(1/2)} + P^{-(1/2)} X^{\mathrm{T}} P^{1/2}\Big),$$

where $\lambda_{\max}$ denotes the largest eigenvalue of a symmetric matrix. In other words, the spectral abscissa $\alpha(X)$ is the optimal value of

$$\inf_{\gamma\in\mathbb{R},\, P\in\mathbf{S}^n} \quad \gamma \tag{7}$$

$$\text{subject to} \quad P \succ 0, \quad 2\gamma P \succeq PX + X^{\mathrm{T}} P, \tag{8}$$

where the latter inequality means that $2\gamma P - (PX + X^{\mathrm{T}} P)$ is positive semidefinite. Furthermore, the infimum is achieved if and only if the active eigenvalues of X are all semisimple. Notice that we can scale P arbitrarily without changing anything. For example, we could constrain $\operatorname{tr} P = 1$. Although this characterization follows easily from fundamental results of Lyapunov, it is surprisingly little known outside

the control community. See [20] for a proof from first principles. Note that the last constraint is bilinear in the variables γ and P. In the control community, such a constraint is known as a bilinear matrix inequality (BMI).

Asymptotic decay of trajectories of $\dot{z} = Xz$ is guaranteed if $\gamma < 0$ is feasible in (7) for some $P \succ 0$. However, transient growth is associated with the conditioning of P. It is easy to show, by differentiating the Lyapunov function $z(t)^{\mathrm{T}} P z(t)$, that

$$\|z(t)\| \leqslant k\mathrm{e}^{\gamma t}\|z(0)\|, \tag{9}$$

where $k = \kappa(P)$, the condition number of P. In fact, the Kreiss Matrix Theorem and its variations [30] state that enforcing a bound of the form (9) is, in a certain sense, *equivalent* to bounding $\kappa(P)$ in (7).

Introducing the affine map $A(x)$ discussed above, we obtain another formulation for minimization of the spectral abscissa $\alpha(A(x))$,

$$\inf_{\gamma\in\mathbb{R}, x\in\mathbb{R}^m, P\in\mathbf{S}^n} \quad \gamma \tag{10}$$

$$\text{subject to} \quad P \succ 0, \quad 2\gamma P \succeq PA(x) + A(x)^{\mathrm{T}} P. \tag{11}$$

The optimization problem is now over a space whose dimension is generally much larger than m. Furthermore, as already noted, minimizing the spectral abscissa typically leads to non-derogatory multiple eigenvalues and hence to an infimum in (10), (11) that is not achieved, i.e., any minimizing sequence (γ^r, x^r, P^r) has the condition number of P^r diverging to ∞. In other words, optimizing asymptotic decay forces arbitrarily bad transient peaks in the trajectories from some starting points.

This raises an idea: for any square matrix X and any $\delta \in (0, 1)$, define the *robust spectral abscissa* $\alpha_\delta(X)$ as the optimal value of

$$\min_{\gamma\in\mathbb{R}, P\in\mathbf{S}^n} \quad \gamma \tag{12}$$

$$\text{subject to} \quad \delta I \preceq P \preceq I, \quad 2\gamma P \succeq PX + X^{\mathrm{T}} P. \tag{13}$$

We call δ the *robustness* parameter. The feasible region for P is now compact, and $\kappa(P)$ is bounded by $1/\delta$. Since P may be scaled arbitrarily in (11), we have $\alpha_\delta(X) \to \alpha(X)$ (a non-Lipschitz function) as $\delta \to 0$ and $\alpha_\delta(X) \to \frac{1}{2}\lambda_{\max}(X + X^{\mathrm{T}})$ (a convex function) as $\delta \to 1$. The function $\alpha_\delta(X)$ is Lipschitz and non-convex for $\delta \in (0, 1)$. Although computing $\alpha_\delta(X)$ is an optimization problem with a BMI constraint, it is tractable via bisection on γ, by checking a sequence of semidefinite feasibility problems, since the BMI reduces to an linear matrix inequality (LMI) for fixed γ. More specifically, since we know that (13) is feasible for $\gamma \geqslant \frac{1}{2}\lambda_{\max}(X + X^{\mathrm{T}})$ (choose $P = I$) and infeasible for $\gamma < \alpha(X)$ (by virtue of (8)–(7)), the minimizing γ may be found by bisection in the interval $[\alpha(X), \frac{1}{2}\lambda_{\max}(X + X^{\mathrm{T}})]$. This is how Fig. 2 was generated, using $\delta = 0.03$. Fig. 2 shows $\alpha_\delta(A(\xi))$ (dashed curve) and $\alpha(A(\xi))$ (solid curve) as functions of ξ for the damped linear oscillator. Efficient interior point methods for solving semidefinite feasibility problems, or, more generally, semidefi-

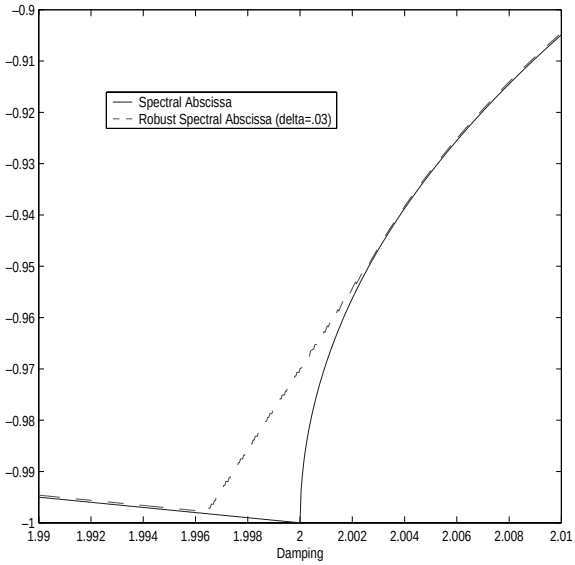

Fig. 2. The spectral abscissa α and robust spectral abscissa α_δ with $\delta = 0.03$, for the damped linear oscillator defined in (2), as a function of the damping parameter ξ.

nite programs, have received much attention in the past decade [24,28], especially in the control community [4,14]. [4]

Given an affine family $A(x)$, then the robust spectral abscissa minimization problem is to minimize $\alpha_\delta(A(x))$, i.e.,

$$\min_{\gamma\in\mathbb{R},x\in\mathbb{R}^m,P\in\mathbf{S}^n} \quad \gamma \tag{14}$$

$$\text{subject to} \quad \delta I \preceq P \preceq I, \quad 2\gamma P \succeq PA(x) + A(x)^{\mathrm{T}}P. \tag{15}$$

For small δ, this is a difficult non-convex problem for which finding a global minimizer is hard. Section 4 presents a Newton barrier method to find local minimizers. Section 5 compares the results for pure and robust spectral abscissa minimization for an interesting set of examples. Section 6 presents some conclusions.

There are many other possible approaches to balancing transient and asymptotic system behavior. For example, since including P as a design variable is expensive, one might define P implicitly via a constraint of the form

$$P(A(x) - \gamma I) + (A(x) - \gamma I)^{\mathrm{T}}P = -I,$$

and devise a minimization objective that suitably weights γ and $\kappa(P)$.

[4] We used both the SeDuMi (http://fewcal.kub.nl/sturm/software/sedumi.html) and SDPpack (http://www.cs.nyu.edu/cs/faculty/overton/sdppack) codes to generate the graph shown in Fig. 2, obtaining essentially the same results with both. This was an interesting test for SDPpack, since the method it uses to detect infeasibility does not have any theoretical foundation, unlike the one used by SeDuMi.

A different approach to the balancing of transient and asymptotic considerations is provided by the notion of pseudospectrum [29]. Given $\epsilon \geqslant 0$, the pseudospectrum of a matrix X is defined as the set of eigenvalues of all matrices Y satisfying $\|Y - X\|_2 \leqslant \epsilon$. Let $\tilde{\alpha}_\epsilon(X)$ denote the pseudospectral abscissa of X, i.e., the maximum of the real parts of the pseudo-eigenvalues of X (the elements of the pseudospectrum). One of the consequences of the Kreiss Matrix Theorem is that, for any $\delta \in (0, 1)$ and fixed n, there exist ϵ_1, ϵ_2, C_1 and C_2 such that

$$C_1\tilde{\alpha}_{\epsilon_1}(X) < \alpha_\delta(X) < C_2\tilde{\alpha}_{\epsilon_2}(X)$$

for every matrix X. Thus, in some sense, the notions of robust spectral abscissa and pseudospectral abscissa are equivalent, although the dependence of ϵ_1, ϵ_2, C_1 and C_2 on δ is complicated. This justifies the use of the word "robust" in the name for α_δ, since it is immediate from the definition that, if $\tilde{\alpha}_\epsilon(X) = \gamma$, then $\alpha(Y) \leqslant \gamma$ whenever $\|Y - X\|_2 \leqslant \epsilon$. Direct pseudospectral abscissa optimization is a topic we are currently studying, but is not discussed in this paper.

We conclude this section by thanking Steve Boyd, Alex Megretski and Nick Trefethen for very helpful conversations regarding the issues discussed here.

3. A random gradient bundle method

In this section we present a random gradient bundle method to approximate minimizers of general non-smooth functions. The ideas are simple and we believe they will be useful in a very general setting. The method is inspired by the great success of gradient bundle methods for convex optimization [17]. Bundle and related methods have been extended to be applicable to Lipschitz non-convex functions [18], but we are not aware of related work for non-Lipschitz functions such as the spectral abscissa. Our basic assumptions are simply that (a) the function f being minimized is continuous and is continuously differentiable almost everywhere and (b) where it is defined, the gradient of f is easily computed.

The assumptions just mentioned hold for the spectral abscissa map (5) since α is differentiable at X if and only if (i) X either has exactly one real active eigenvalue, say λ, or exactly one complex conjugate pair of active eigenvalues, say $\lambda \pm \mathrm{i}\tilde{\lambda}$, and (ii) this active eigenvalue or conjugate pair of eigenvalues is simple, i.e., has multiplicity one. Suppose this is the case. Then the gradient of (5) is easily obtained by computing the eigenvectors of X. When the active eigenvalue is real, with real right and left eigenvectors v and u satisfying

$$Xv = \lambda v, \quad X^{\mathrm{T}}u = \lambda u \tag{16}$$

with $u^{\mathrm{T}}v = 1$, it is well known [16, Theorem 6.3.12] that the gradient is given by

$$\nabla\alpha(X) = uv^{\mathrm{T}}.$$

When a conjugate pair of eigenvalues $\lambda \pm \mathrm{i}\tilde{\lambda}$ is active, we break the eigenvectors associated with $\lambda \pm \mathrm{i}\tilde{\lambda}$ into real and imaginary parts, so that

$$X(v+\mathrm{i}\tilde{v}) = (\lambda+\mathrm{i}\tilde{\lambda})(v+\mathrm{i}\tilde{v}), \quad X^{\mathrm{T}}(u+\mathrm{i}\tilde{u}) = (\lambda-\mathrm{i}\tilde{\lambda})(u+\mathrm{i}\tilde{u}) \tag{17}$$

with $(u-\mathrm{i}\tilde{u})^{\mathrm{T}}(v+\mathrm{i}\tilde{v}) = 1$. Then it is easily seen that

$$\nabla\alpha(X) = uv^{\mathrm{T}} + \tilde{u}\tilde{v}^{\mathrm{T}}.$$

It follows from the chain rule that the gradient of (6) is given by

$$(\nabla(\alpha\circ A)(x))_k = u^{\mathrm{T}}A_k v$$

when $A(x)$ has exactly one real active eigenvalue, with corresponding eigenvectors satisfying (16), and

$$(\nabla(\alpha\circ A)(x))_k = u^{\mathrm{T}}A_k v + \tilde{u}^{\mathrm{T}}A_k\tilde{v}$$

when $A(x)$ has exactly one conjugate pair of active eigenvalues, with corresponding left and right eigenvectors satisfying (17).

We illustrate the main idea of the random gradient bundle method by considering the example whose contours are displayed in Fig. 1. First, consider an ordinary steepest descent method with a line search, assuming that, at every iterate, $\alpha(A(x))$ is differentiable. Fig. 3 plots the steepest descent iterates using small circles, starting from the initial point $(-1,-1)$. The behavior is typical of the steepest descent method applied to a non-smooth function; progress is good at first, but then the iterates jam near the manifold on which the function is not differentiable. The iterates of our gradient bundle algorithm are shown using asterisks in the same Fig. 3. Instead of calculating only one gradient per iteration, the algorithm evaluates a *bundle of gradients* at points *nearby* the current iterate, chosen by sampling from a uniform

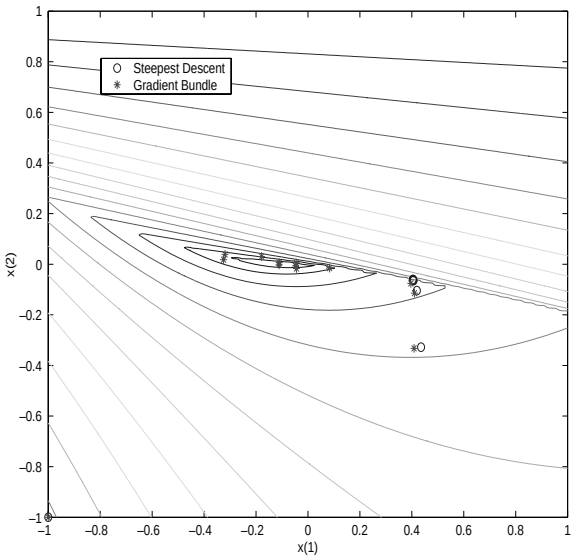

Fig. 3. Gradient bundle and steepest descent iterates for the spectral abscissa minimization problem shown in Fig. 1.

distribution scaled by a parameter ϵ. The algorithm's progress is similar to the steepest descent method at first, because all the gradients in the bundle are qualitatively similar at first. However, when the iterates approach the region where the steepest descent method jams, the bundle includes gradients from *both sides* of the manifold on which the spectral abscissa is not smooth. The algorithm is based on a generalized steepest descent method for convex optimization, using a search direction d with the property that $-d$ is the *convex combination of the gradients in the bundle whose* 2*-norm is minimized* (a convex quadratic subproblem that is easily solved). Therefore, in the region where steepest descent jams, the bundle algorithm is able to use the information from both sides of the manifold to "turn the corner" and make progress towards the minimizer, as shown by the asterisks in Fig. 3. Eventually, it finds an iterate at which 0 is in the convex hull of its gradient bundle, and it terminates. At this point, we would restart the minimization with a smaller value of ϵ.

We now formally define the random gradient bundle algorithm. It is assumed that f, the function to be minimized, is continuous, and continuously differentiable almost everywhere. When we write $\nabla f(x)$ below, we are implicitly assuming that f is differentiable at x, a property that holds with probability one by assumption if x is a random point. The symbol G is used for the set of gradients making up a bundle, and the notation conv G means the convex hull of G. We restrict our iterates x to the feasible region defined by

$$\|x\|_\infty \leqslant \chi,$$

for some moderately large constant χ.

Algorithm 1 (*Gradient Bundle*).

0. (*Initialization*) Choose an initial feasible point $x \in \mathbb{R}^m$, an initial positive value for the sampling radius ϵ, a positive reduction factor $\theta < 1$, a positive integer N defining the number of gradients per bundle, and two positive integers M_1 and M_2 to terminate the iterations. Initialize $k = 1$.
1. (*Inner iteration*) Carry out minimization k, controlled by the sampling radius ϵ, as follows. Initialize $j = 0$.
   (a) Define a bundle G as the set of N gradients $\{\nabla f(y)\}$, where y takes on N values: the current iterate x, and $N - 1$ other vectors differing from x by vectors whose entries are obtained by sampling from a uniform distribution on $[-\epsilon/2, \epsilon/2]$.
   (b) Define the search direction

   $$d = -\arg\min\left\{\|v\|_2 \colon v \in \operatorname{conv} G\right\}. \tag{18}$$

   If $d = 0$, go to Step 2.
   (c) Use a line search (see below) to find a positive steplength satisfying

   $$f(x + td) < f(x), \quad t \in (0, \bar{t}], \tag{19}$$

   where $\bar{t} = \arg\max\{t : \|x + td\|_\infty \leqslant \chi\}$.
   (d) Replace x by $x + td$. If $t = \bar{t}$, terminate. If $j < M_2$, increment j and return to Step (a).

2. (*Decrease sampling parameter*) If $k < M_1$, replace ϵ by the smaller value $\theta\epsilon$, increment k and return to Step 1. Otherwise terminate.

The condition imposed by the line search is simply a reduction in the value of the objection function. It is a standard practice in line searches for smooth and convex optimization to impose a somewhat more demanding "sufficient decrease" condition. However, in the non-Lipschitz case, function values can be extremely sensitive to changes in the argument in the neighborhood of a minimizer, and the imposition of any sufficient decrease condition often leads to failure in the line search.

The definition of d in (18) guarantees the descent condition

$$\nabla f(x)^{\mathrm{T}} d < 0$$

as long as d is not zero, since $\nabla f(x)$ is one of the gradients in the bundle G. To see this, observe that if d is not 0, there is a separating hyperplane between 0 and conv G, so $-d$, the closest element of conv G to 0 in the 2-norm, must satisfy the condition $h^{\mathrm{T}}(-d) > 0$ for every $h \in \operatorname{conv} G$, including $h = \nabla f(x)$. The descent condition ensures that (19) must hold for sufficiently small t.

We use a very simple line search:

Algorithm 2 (*Line Search*).

0. (*Initialization*) Set $s = 0$, $t = 1$. If (19) is violated, go to Step 2.
1. (*Doubling phase*) Repeatedly do the following: replace s by the value of t and replace t by $\min\left(2t, \bar{t}\right)$, until $t = \bar{t}$ or $f(x + td) > f(x + sd)$. In the latter case replace t by its previous value s. In both cases, terminate.
2. (*Bisection phase*) Repeatedly replace t by $t/2$, until t satisfies (19).

In this simple line search, either a doubling process (Step 1) or a bisection process (Step 2) is executed. The doubling phase is initiated if the initial point satisfies (19) and it continues until an increase in f, compared to its value at the previous point, is obtained, at which point the line search terminates with the previous point, or the bound on the step is reached. The bisection phase is initiated if the initial point violates (19) and it continues until a point is found satisfying (19). In exact arithmetic, it must terminate since (19) must hold for t sufficiently small. In practice, we place a limit of 50 on the number of bisection steps, and if this is exceeded, the inner loop in Algorithm 1 is terminated, with control passing to its Step 2.

Algorithm 1 terminates in Step 1(d) if an iterate x is generated with $\|x\|_\infty = \chi$, the bound on the feasible region being considered. We also experimented with a version of the algorithm that adds bounds to the quadratic program solved in Step 1(b), in order to minimize f over the box defined by $\|x\|_\infty \leqslant \chi$, but we do not discuss this here as we prefer to keep the algorithm description as simple as possible.

We implemented Algorithm 1 in MATLAB. If we were able to solve the quadratic program in Step 1(b) of Algorithm 1 exactly, no tolerance would be needed

when comparing d to 0. In practice, a tolerance is needed; we use the test $\|d\|_2 \leqslant \tau$, where τ is a positive parameter. We use MOSEK [22] to solve the quadratic program. MOSEK implements one of a class of interior-point methods to solve convex programs that are known to be very efficient both in theory and in practice. It is available as a MATLAB toolbox. It is more efficient than various other MATLAB-callable codes that we tried.

Two key questions regarding Algorithm 1 remain open. First, under what conditions can we guarantee that it will terminate with the test on d satisfied ($d = 0$ in exact arithmetic; $\|d\| \leqslant \tau$ in floating point arithmetic)? Secondly, if this test is satisfied, what can be concluded about approximate optimality? Both these questions are under study at present, and a partial answer to the second one is given in [9], using non-smooth analysis. Briefly, the motivation for termination is that if the test on d is satisfied, 0 lies in the convex hull of the bundle of gradients of f evaluated near x (exactly or approximately, depending on whether exact or floating point arithmetic is assumed). Intuitively, this suggests that x is near a critical point of f; for example, in the case that f is a univariate function, the test is satisfied when the bundle includes two derivatives with opposite sign, evaluated at two points near each other but on either side of a critical point (0, for $f(x) = x^2$ or $f(x) = |x|$, for example). The descent property of the line search seeks to ensure that minimizers will be found in preference to maximizers.

In Section 5, we report numerical results for minimizing the spectral abscissa $\alpha(A(x))$ by Algorithm 1.

4. A Newton barrier method

In this section, we briefly describe a Newton barrier method to solve (14), (15). As in the previous section, we impose an a priori upper bound on the norm of the solution x. Since we are using a barrier method, the 2-norm is a convenient choice, so we constrain $\|x\|_2 \leqslant \psi$ for some moderately large constant ψ. Consider the barrier function $B_\mu : \mathbb{R} \times \mathbb{R}^m \times \mathbf{S}^n \to \mathbb{R}$ defined by

$$B_\mu(\gamma, x, P) = \gamma - \mu \log\det(I - Pt) - \mu \log\det(P - \delta I) \\ -\mu \log\det(2\gamma P - (PA(x) + A(x)^{\mathrm{T}} P)) - \mu \log(\psi^2 - x^{\mathrm{T}} x).$$

Recall that δ is the fixed robustness parameter. The new parameter μ is a positive *barrier parameter*, and we interpret $\log\det(M)$ to have the value $-\infty$ when its symmetric matrix argument M is not positive definite. The three log determinant terms impose the constraints in (15), and the final log term imposes the bound on $\|x\|_2$. The use of log barrier functions to impose non-linear inequality constraints goes back to the 1950s and was used extensively in the 1960s [12]. The approach fell out of favor in the 1970s but was revived in the 1980s and 1990s [32], following the explosion of interest in interior point methods for linear and convex programming. The role

of the log determinant function in the analysis of interior point methods for convex semidefinite programs is particularly well known, but its use in the implementation of barrier methods for problems with non-linear semidefinite constraints is also quite well established; see e.g. [19,26].

As the title of [12] suggests, the standard approach for solving non-linear programs by barrier methods is to solve a sequence of unconstrained problems, i.e., minimize a sequence of barrier functions, ideally with monotonically decreasing values for the barrier parameter μ. As is also standard, we use Newton's method to solve these unconstrained problems. The barrier function $B_\mu(\gamma, x, P)$ is smooth but generally non-convex. It must be locally convex at any non-degenerate minimizer, but the smaller the parameter δ is, the smaller is the region around the solution where the barrier function is convex. Consequently, it is essential to implement Newton's method in a way that recognizes and deals with non-convexity. Indeed, the barrier function $B_\mu(\gamma, x, P)$ makes an excellent class of test functions for Newton barrier codes.

The gradient and Hessian of $B_\mu(\gamma, x, P)$ are derived in Appendix A. Let us write $v = (\gamma, x^{\mathrm{T}}, \mathrm{svec}(P)^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^{n(n+1)/2} = \mathbb{R}^p$, where svec is the standard isometry from $\mathbf{S}^n$ to $\mathbb{R}^{n(n+1)/2}$ and $p = m + 1 + n(n+1)/2$. In order to implement Newton's method for minimizing $B_\mu(\gamma, x, P)$, we actually work in the space $\mathbb{R}^p$. We denote the associated barrier function in this space by $\hat{B}_\mu(v)$. The formula for $\nabla^2 \hat{B}_\mu(v)$ involves symmetrized Kronecker products [1,31]. We omit the details, but we note that it is important to use symmetrized Kronecker products, not standard Kronecker products, since $P \in \mathbf{S}^n$, not $\mathbf{M}^n(\mathbb{R})$. If Newton's method is implemented in $\mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^{n^2}$ instead of $\mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^{n(n+1)/2}$, so that standard Kronecker products are used instead of symmetrized Kronecker products, the resulting Hessian of the barrier function in this space is necessarily singular. Our MATLAB m-file for computing $\nabla \hat{B}_\mu(v)$ and $\nabla^2 \hat{B}_\mu(v)$ is available at [6].

There are two standard approaches to implementing Newton's method for minimizing smooth non-convex functions: trust region methods and modifed Cholesky factorization methods. We chose to take neither approach, but to implement a simpler algorithm. We start by attempting to compute the Cholesky factorization of the Hessian matrix $\nabla^2 \hat{B}_\mu(v)$. If the factorization succeeds, a standard Newton direction is passed to a line search. If the factorization breaks down, i.e., a negative or zero pivot is encountered, we compute the eigenvalues of $\nabla^2 \hat{B}_\mu(v)$ and recompute the Cholesky factorization of a shifted Hessian matrix which is positive definite but nearly singular. This yields an excellent modified Newton search direction, dominated by the eigenvector corresponding to $\lambda_{\min}(\nabla^2 \hat{B}_\mu(v))$ (the direction of greatest negative curvature) unless it is very nearly orthogonal to the gradient. This direction is normalized before being passed to the line search. Computing multiple factorizations is, in principle, much more expensive than computing one modified Cholesky factorization such as described in [13], but in MATLAB it is much more efficient than implementing the modified Cholesky algorithm as an m-file, and in any case, in our

application the cost is trivial, since forming the Hessian matrix is far more expensive than computing its Cholesky factorization or its eigenvalues.

We now summarize the Newton barrier method. We state it so that it is applicable to any family of barrier functions $\hat{B}_\mu(v)$. In our application, given an initial iterate x, we initialize γ to $\frac{1}{2}\lambda_{\max}(A(x) + A(x)^{\mathrm{T}}) + 1$ and P to $\frac{1}{2}I$ (since we are interested only in $\delta < \frac{1}{2}$).

Algorithm 3 (*Newton Barrier*).

0. (*Initialization*) Choose an initial point v with $\hat{B}_\mu(v) < \infty$, a positive reduction factor $\theta < 1$, an increase factor $\tilde{\theta}$ satisfying $1 < \tilde{\theta} < 1/\theta$, a very small positive shift parameter ν, a termination tolerance τ, and two positive integers M_1 and M_2 to terminate the iterations. Initialize the barrier parameter μ and set $k = 1$.
1. (*Inner iteration*) Carry out minimization k, controlled by the barrier parameter μ, as follows. Initialize $j = 0$, and set $\hat{v}$ to v (i.e., save a copy of the starting point).
   (a) Compute the gradient $\nabla\hat{B}_\mu(v)$. If $\|\nabla\hat{B}_\mu(v)\| \leqslant \tau$, go to Step 2. Compute the Hessian matrix $\nabla^2\hat{B}_\mu(v)$ and attempt to compute its Cholesky factorization $\nabla^2\hat{B}_\mu(v) = LL^{\mathrm{T}}$, with L lower triangular. If this breaks down, compute the eigenvalues of $\nabla^2\hat{B}_\mu(v)$ and recompute the Cholesky factorization of the shifted matrix,
   $$\nabla^2\hat{B}_\mu(v) + \left(\nu\|\nabla^2\hat{B}_\mu(v)\| - \lambda_{\min}(\nabla^2\hat{B}_\mu(v))\right) I = LL^{\mathrm{T}}$$
   with L lower triangular.
   (b) Define the search direction
   $$d = -L^{-\mathrm{T}}L^{-1}\nabla\hat{B}_\mu(v). \tag{20}$$
   (c) Use a line search to find a positive steplength satisfying
   $$\hat{B}_\mu(v + td) < \hat{B}_\mu(v). \tag{21}$$
   (d) Replace v by $v + td$. If $j < M_2$, increment j and return to Step (a).
2. (Update barrier parameter) If the test $\|\nabla\hat{B}_\mu(v)\| \leqslant \tau$ was satisfied in Step 1(a), go to Step 2(b).
   (a) If $k < M_1$, replace μ by the larger value $\tilde{\theta}\mu$, discard the current iterate v and replace it by $\hat{v}$, increment k and return to Step 1. Otherwise terminate.
   (b) If $k < M_1$, replace μ by the smaller value $\theta\mu$, increment k and return to Step 1. Otherwise terminate.

For simplicity, we use the line search of the previous section (Algorithm 2), without the upper bound on t, although it would be more standard to impose a sufficient decrease condition since the barrier function is smooth. The absence of the sufficient decrease condition, so essential for convergence analysis, does not seem to have much practical effect. In fact, limited experiments with a more standard line search imposing the well-known Wolfe conditions resulted in somewhat worse overall performance.

Note the parallel inner–outer iteration structure of Algorithms 1 and 3. The outer iteration of the former reduces the sampling parameter ϵ, while the outer iteration of the latter reduces the barrier parameter μ. A key difference concerns the behavior when the maximum number of iterates allowed for the inner iteration is reached. Algorithm 1 proceeds to reduce ϵ anyway, since the spectral abscissa is monotonically decreasing throughout the algorithm. There is no corresponding monotonicity on which Algorithm 3 can rely, so if an inner iteration fails to reduce $\|\nabla \hat{B}_\mu(v)\|$ to the desired level, the algorithm backtracks, trying a less aggressive choice of μ. An alternative strategy would be to check whether γ, the bound on the robust spectral abscissa and the first component of v, is lower than the first component of the inner iteration starting point $\hat{v}$.

5. Numerical results

We have experimented with various affine matrix families $A(x)$, applying Algorithm 1 (Gradient Bundle) to find local minimizers of the spectral abscissa $\alpha(A(x))$ and Algorithm 3 (Newton Barrier) to find local minimizers of the robust spectral abscissa $\alpha_\delta(A(x))$. We focus our attention on small values of n, the matrix dimension, and m, the number of variables, because, as we shall see, even small problems are quite challenging. The reason for this is that, for most of our examples, local minimizers of the spectral abscissa are associated with at least one active non-derogatory multiple eigenvalue (see Section 1). Suppose that the optimal matrix has a non-derogatory multiple eigenvalue with multiplicity $k > 2$. Then, as is well known [21], small perturbations to the matrix with magnitude t result in eigenvalue perturbations whose dominant term is $(ct)^{1/k}$ for some complex number c. For $k > 2$, at least one of the perturbed eigenvalues moves right in the complex plane in all but trivial cases, resulting in a spectral abscissa perturbation of the form $rt^{1/k}$ for some $r > 0$. (When $k = 2$, it may be only the imaginary part of the eigenvalues that changes rapidly under perturbation; see (3).) For example, if $k = 5$, $c = 1$ and t is the machine precision for the IEEE double format arithmetic used by MATLAB (about 10^{-16}), the spectral abscissa is increased by approximately 10^{-3}. Thus, the optimal value of the spectral abscissa is extremely sensitive to even very small perturbations.

Before presenting the results, we specify the parameters defining the algorithms. We set χ and ψ, the bounds on $\|x\|_\infty$ and $\|x\|_2$ in Gradient Bundle and Newton Barrier, respectively, to 1000, the tolerance τ used in Step 1(b) of Algorithm 1 (see discussion following statement of algorithm) and Step 1(a) of Algorithm 3 to 10^{-6}, and the outer and inner iteration limits M_1 and M_2 in both algorithms to 6 and 100, respectively. For Gradient Bundle, we set the initial value for the sampling radius ϵ to 0.1, its reduction factor θ to 0.1, and N, the number of gradients per bundle, to $2m$ (twice the number of variables). For Newton Barrier, we set the initial value for the barrier parameter μ to 0.1, its reduction factor θ to 0.1, its increase factor $\tilde{\theta}$ to 3, and the Hessian shift parameter ν to 10^{-14}.

We initialized all runs of Gradient Bundle with random initial vectors x whose components were drawn from a normal distribution with mean 0 and variance 1. In the case of Newton Barrier, since we wished to minimize $\alpha_\delta(A(x))$ for a variety of δ, we always began with $\delta = 10^{-1}$, initializing x randomly, and then we initialized runs for smaller values of δ, ranging from 10^{-2} to 10^{-6}, with the x iterate found in the previous minimization. We also tried doing the same with the γ and P iterates, instead of initializing them as described just before the statement of Algorithm 3, but this is not a good idea; as is well known, interior point methods should not be initialized with points too close to the boundary of the feasible region.

We present the examples one at a time, illustrating various points as we proceed. All figures show eigenvalues as points in the complex plane. When we refer to multiple eigenvalues below, it may be assumed that they are non-derogatory except as noted.

Example OneBlock. Our first example is (4), which we repeat here for convenience. Given n, define $m = n - 1$, and

$$A(x) = \begin{bmatrix} -x_1 & 1 & 0 & \cdot & \cdot & 0 \\ x_1 & 0 & 1 & 0 & \cdot & 0 \\ x_2 & 0 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\ x_{n-1} & 0 & \cdot & \cdot & \cdot & 0 \end{bmatrix}.$$

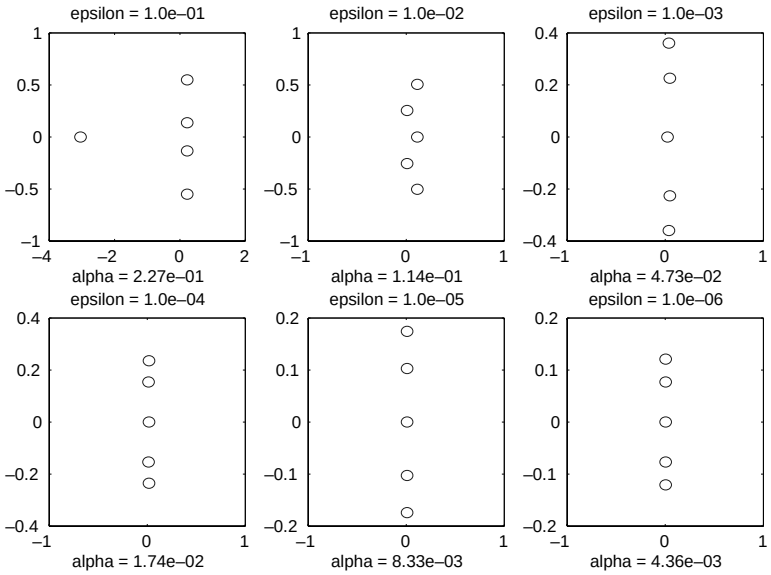

Fig. 4. Example OneBlock. Eigenvalues after *each* outer iteration of Gradient Bundle method to minimize $\alpha(A(x))$.

This example is constructed so that the global minimizer is $\bar{x} = 0$, with $A(\bar{x})$ having an active multiple eigenvalue 0 with multiplicity n [8]. We used $n = 5$ for our experiments. Fig. 4 shows the results of a typical run of Gradient Bundle on this problem, displaying the eigenvalues of $A(x)$ for x generated at the end of *each* of the inner iterations of Gradient Bundle, i.e., for the sampling radius $\epsilon = 10^{-1}$ through $\epsilon = 10^{-6}$, respectively. Each subplot also displays the final value of the spectral abscissa α that was found. Note the steady reduction in the value of α obtained as ϵ is reduced. Note also the reduction in the size of the *imaginary* parts of the eigenvalues, showing that the eigenvalues are moving, as the minimization proceeds, towards the limiting multiple zero eigenvalue. The results for Gradient Bundle did not vary much with different runs, although every run starts from a different initial point and generates different bundles of random gradients. In rare cases a run terminated with iterates on the boundary of the feasible region, i.e., with $\|x\|_\infty = 1000$. This was more common for larger n, and indicates the existence of recession directions for α, so that when constrained by $\|x\|_\infty \leqslant 1000$, there may be one or more local minimizers of α on the boundary, although we know there are no unconstrained local minimizers besides 0 and that 0 is a global minimizer. In order to find constrained minimizers, the algorithm would have to be modified, as mentioned earlier.

Fig. 5 shows results for six runs of Newton Barrier, for values of the robustness parameter δ ranging from 10^{-1} to 10^{-6}, respectively. Each one of these runs requires

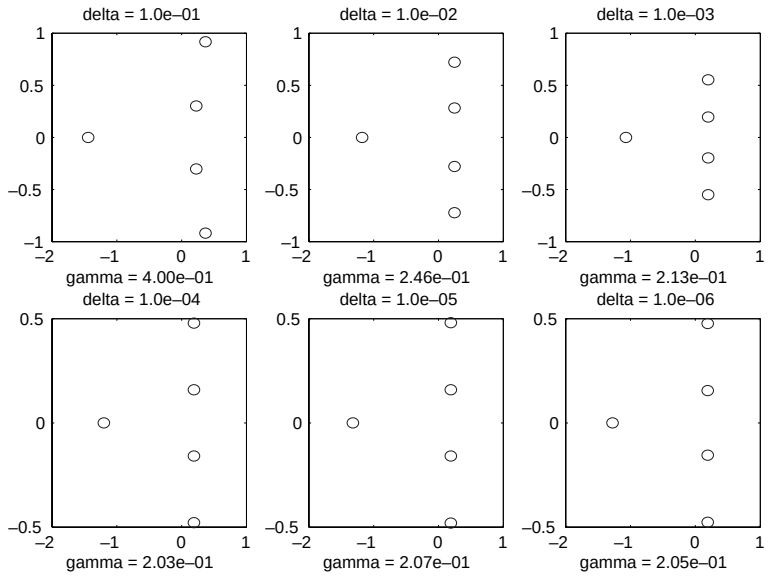

Fig. 5. Example OneBlock. Eigenvalues after *all* outer iterations of Newton Barrier method to minimize $\alpha_\delta(A(x))$, for various choices of δ.

an outer iteration to vary the barrier parameter μ, and each of these requires an inner iteration to minimize the barrier function. The amount of work required to generate Fig. 5 is clearly at least an order of magnitude more than the work required to generate Fig. 4; actually, it is much more than this, because the time required for one Newton step is substantially more than the time required for one bundle step, as more variables are involved. Despite this huge increase in computational work, the information displayed by Fig. 5 is rather unimpressive compared with that in Fig. 4. In fact, all subplots of Fig. 5 show eigenvalues that are strikingly similar to the first plot in Fig. 4, which is trivial to compute by comparison. The reason that the results for the smaller values of δ do not show better convergence to the multiple zero eigenvalue is apparently that the barrier functions are so illconditioned that the Newton method is unable to minimize them to the specified accuracy. The values of γ shown are the best of the bounds on the robust spectral abscissa computed by Newton Barrier, not necessarily the final bound, as the minimization of the barrier function for the final barrier parameter μ may fail. Though the results shown in Fig. 5 are typical, from some starting points the algorithm found minimizers near the boundary of the feasible region (see the discussion above for Gradient Bundle).

Example TwoBlock. Given n_1 and n_2, define $m = n_1 + n_2 - 1$, $n = n_1 + n_2$, and

$$A(x) = \begin{bmatrix}
-x_1 & 1 & 0 & \cdot & \cdot & 0 & x_{n_1} & x_{n_1+1} & \cdot & \cdot & \cdot & x_{n_1+n_2-1} \\
x_1 & 0 & 1 & 0 & \cdot & 0 & 0 & 0 & \cdot & \cdot & \cdot & 0 \\
x_2 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & 0 & \cdot & \cdot & \cdot & \cdot & 0 \\
x_{n_1-1} & 0 & \cdot & \cdot & \cdot & 0 & 0 & \cdot & \cdot & \cdot & \cdot & 0 \\
x_{n_1} & 0 & \cdot & \cdot & \cdot & 0 & -1 & 0 & \cdot & \cdot & \cdot & 0 \\
x_{n_1+1} & 0 & \cdot & \cdot & \cdot & 0 & 0 & -1 & \cdot & \cdot & \cdot & 0 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\
x_{n_1+n_2-1} & 0 & \cdot & \cdot & \cdot & 0 & 0 & 0 & \cdot & \cdot & 0 & -1
\end{bmatrix}.$$

This example is constructed so that the global minimizer is $x = 0$, with $A(0)$ having an active non-derogatory multiple eigenvalue 0, with multiplicity n_1, and an inactive semisimple eigenvalue -1, with multiplicity n_2. Example OneBlock is the special case $n_2 = 0$. We used $n_1 = 5$, $n_2 = 5$ for our experiments. The results are very similar to the results for Example OneBlock, except for the presence of the additional eigenvalue -1, so we do not display them.

Example SimpleEigs. Given an even integer n, let $m = (n/2) - 1$, with

$$A(x) = \begin{bmatrix} x_1 & 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ -1 & x_1 & 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & x_2 & 2 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & -2 & x_2 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x_m & m & 0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -m & x_m & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 0 & y & m+1 \\ 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & -m-1 & y \end{bmatrix},$$

where $y = -\sum_{k=1}^{m} x_k$. This example is constructed so that the global minimizer is $\bar{x} = 0$, with $A(\bar{x})$ having $m + 1$ simple complex conjugate pairs of eigenvalues, $\pm k\mathrm{i}$, $k = 1, \ldots, m$, all of them active, and no real eigenvalues. We used $n = 10$. Because all the active eigenvalues are simple, this problem is very easily solved to high accuracy. Fig. 6 shows the eigenvalues computed in just one inner iteration of Gradient Bundle with sampling radius $\epsilon = 0.1$ and one run of Newton Barrier (six inner iterations) with $\delta = 0.1$. Even at these coarse levels of accuracy, the results accurately identify the optimal eigenvalues.

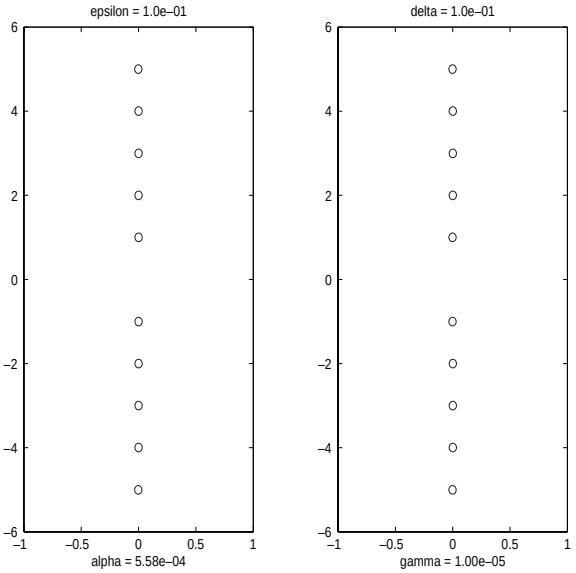

Fig. 6. Example SimpleEigs. Gradient Bundle and Newton Barrier move the eigenvalues close to the imaginary axis even with $\epsilon = 0.1$ and $\delta = 0.1$, respectively.

Example PolShc. This consists of three specific small examples from [25].

(a) $n = 3$, $m = 2$, with

$$A(x) = \begin{bmatrix} -x_1 & 1 & 0 \\ 13 + 5x_1 - x_2 & 0 & 1 \\ -x_2 & 0 & 0 \end{bmatrix}.$$

(b) $n = 5$, $m = 3$, with

$$A(x) = \begin{bmatrix} -3 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 3 - x_2 & 0 & 0 & 1 & 0 \\ 2 - x_1 & 0 & 0 & 0 & 1 \\ -x_3 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

(c) $n = 3$, $m = 3$, with

$$\begin{aligned} A(x) = {} & \begin{bmatrix} 2 & 2 & 0 \\ -1 & 0 & 0 \\ -1 & 0 & 2 \end{bmatrix} + x_1 \begin{bmatrix} -0.4582 & 0.4027 & 0.9691 \\ 0.737 & -0.4511 & -0.3452 \\ -0.7406 & 0.816 & 0.7331 \end{bmatrix} \\ & + x_2 \begin{bmatrix} -0.589 & 0.5471 & -0.6725 \\ 0.1761 & 0.5744 & 0.4972 \\ -0.7262 & -0.4928 & -0.8219 \end{bmatrix} \\ & + x_3 \begin{bmatrix} -0.9571 & -0.3868 & -0.1505 \\ 0.4578 & -0.656 & 0.3161 \\ -0.1786 & 0.4769 & 0.5364 \end{bmatrix}. \end{aligned}$$

Apparently Examples PolShc(a) and PolShc(b) have only one local minimizer of $\alpha(A(x))$. In the case of Example PolShc(a), many runs of Gradient Bundle from different random starting points led to the same minimizer, approximately $\bar{x} = [17.73, 206.4]^{\mathrm{T}}$, for which $\alpha(A(\bar{x})) = -5.909$, with an active triple eigenvalue (and no inactive eigenvalues). We refer to this minimizer as stable, since its spectral abscissa is negative. Many runs also led to the boundary, indicating the probable presence of local constrained minimizers with higher objective values.

In the case of Example PolShc(b), all runs of Gradient Bundle led to the same stable minimizer, approximately $\bar{x} = [2.008, 3.135, 0.0002]^{\mathrm{T}}$, for which $\alpha(A(\bar{x})) = -0.0900$, with an active quadruple eigenvalue (and one inactive simple eigenvalue). Figs. 7 and 8 show the results for Example PolShc(b) for Gradient Bundle (at the end of each inner iteration) and Newton Barrier (for various choices of the robustness parameter δ), respectively, as was done in Figs. 4 and 5 for Example OneBlock. As in that example, ill conditioning of the barrier functions prevented Newton Barrier from obtaining accurate robust spectral abscissa bounds γ for the two smallest values of δ.

In Example PolShc(c), the spectral abscissa is apparently unbounded below; almost all runs of Gradient Bundle generated stable iterates x with $\|x\|_\infty = 1000$ and

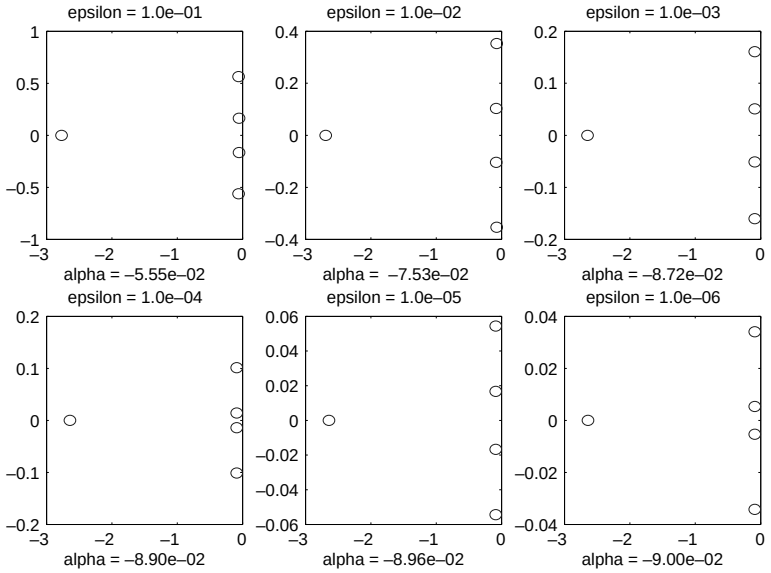

Fig. 7. Example PolShc(b). Eigenvalues after *each* outer iteration of Gradient Bundle method to minimize $\alpha(A(x))$.

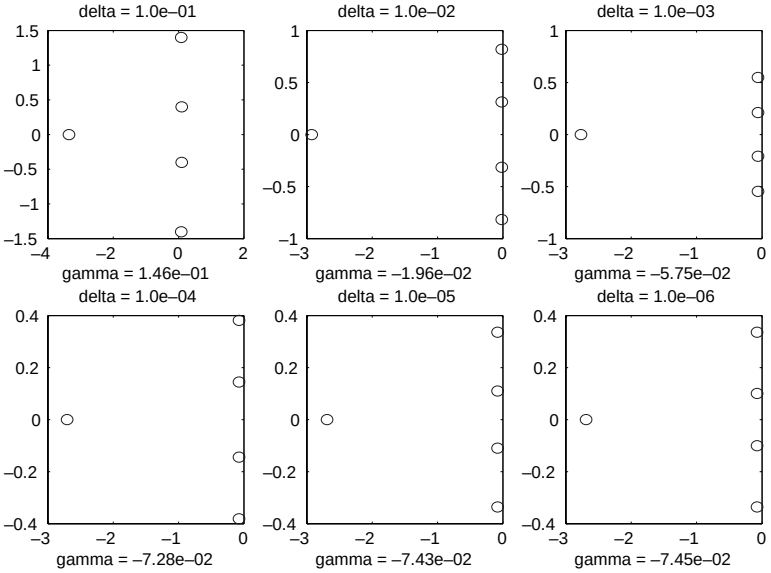

Fig. 8. Example PolShc(b). Eigenvalues after *all* outer iterations of Newton Barrier method to minimize $\alpha_\delta(A(x))$, for various choices of δ.

every run of Newton Barrier generated stable iterates x with $\|x\|_2$ nearly equal to 1000. Some runs of Gradient Bundle converged to an unstable local minimizer, $\bar{x} = [-1.45, 0.42, 1.75]^{\mathrm{T}}$, for which $\alpha(A(\bar{x})) = 0.672$, with an active triple eigenvalue.

In the results reported in [25], no attempt was made to optimize stability. The technique discussed there generates marginally stable matrices, e.g., for Example PolShc(b), with $\alpha(A(x)) = -0.01$.

Example ABKC0. Given an $n \times n$ matrix $\bar{A}$, an $n \times r$ matrix $\bar{B}$ and an $s \times n$ matrix $\bar{C}$, let $m = rs - 1$ and define the affine family $A(x)$ by $\bar{A} + \bar{B}\,\mathrm{mat}(x)\,\bar{C}$, where mat is the natural isometry from $\mathbb{R}^m$ to the space of r by s matrices with lower right entry fixed to zero. We fix one entry to zero because, without this restriction, we find that randomly generated matrices $\bar{A}$, $\bar{B}$ and $\bar{C}$ lead invariably to problems where the first step of Gradient Bundle generates an iterate x on the boundary of the feasible region, i.e., with $\|x\|_\infty = 1000$. Though the restriction is artificial, it serves the purpose of generating some problems with interior local minimizers and therefore allowing us to keep the discussion of Gradient Bundle as simple as possible. We set $n = 6$, $r = 3$ and $s = 2$, and generated the matrices $\bar{A}$, $\bar{B}$ and $\bar{C}$ randomly; the matrix data are available at [6].

We ran Gradient Bundle from 100 different starting points and repeatedly found approximations to two distinct interior local minimizers of $\alpha(A(x))$. Fig. 9 shows the eigenvalues (after all six inner iterations of Gradient Bundle) for the best approximations found for each of these two minimizers, as well as the number of times each was found. Local Minimizer 1 has three active simple complex conjugate pairs of eigenvalues. In contrast, Local Minimizer 2 has an active double real eigenvalue and two active simple conjugate pairs. Almost all of the 39 runs that did not converge to either minimizer generated iterates on or near the boundary. The two approximate minimizers $\mathrm{mat}(\bar{x})$ are available at [6].

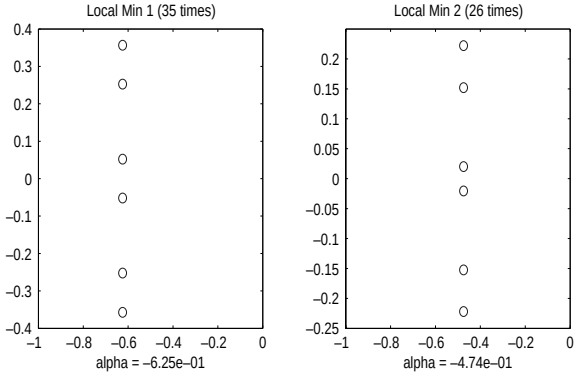

Fig. 9. Example ABKC0. Eigenvalues after *all* outer iterations of Gradient Bundle method to minimize $\alpha(A(x))$. Runs from 100 different starting points found two interior local minimizers.

For this problem, every run of Newton Barrier generated iterates x near its boundary, i.e., with $\|x\|_2$ nearly equal to 1000; we do not show these results.

Example Random. Given n and m, we randomly generated the matrices A_k, $k = 0, \ldots, m$. The randomly generated data may be found in [6].

(a) The first of our two random examples has $m = 5$ and $n = 8$; the randomly generated matrices are dense. We ran Gradient Bundle from 100 different starting points, and found 12 distinct approximate minimizers. Fig. 10 shows the eigenvalues for the best approximation found for each of these, with subplots ordered by spectral abscissa value and indicating how many times each one was found.

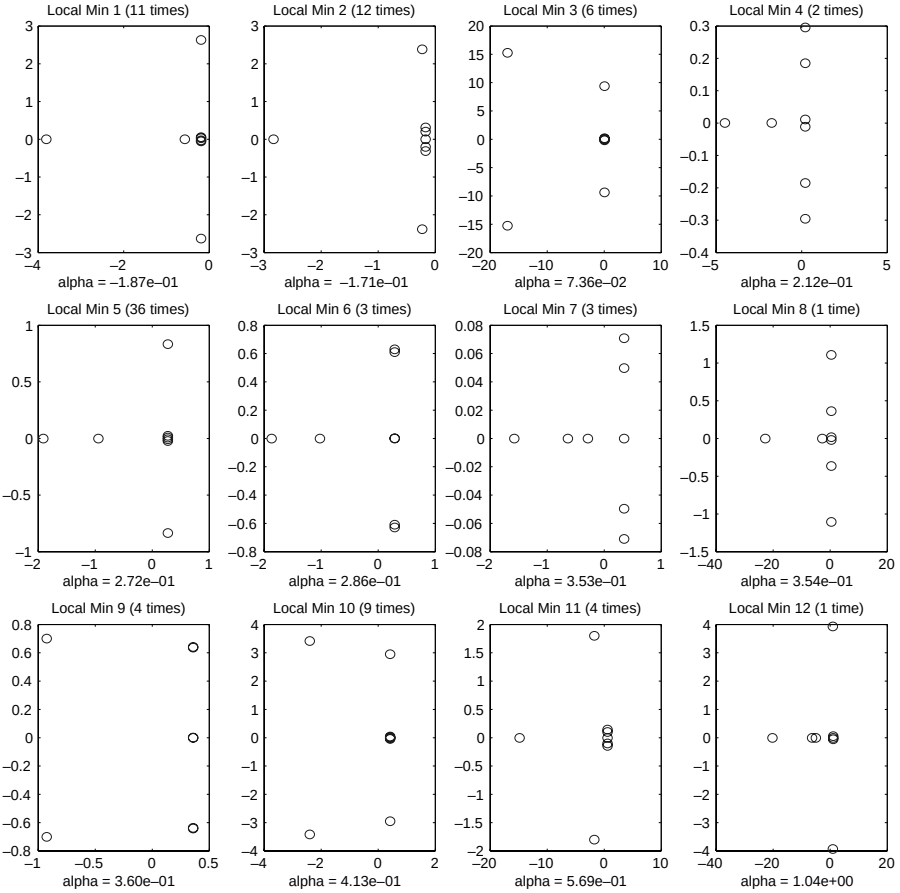

Fig. 10. Example Random(a). Eigenvalues after *all* outer iterations of Gradient Bundle method to minimize $\alpha(A(x))$. Runs from 100 different starting points found 12 local minimizers.

One minimizer, Local Minimizer 5, was found much more frequently than others. Comparison of the data and examination of the intuitive conditions for optimality discussed at the end of Section 3 gave us some confidence that each subplot in Fig. 10 indeed corresponds to a different local minimizer. It is interesting to compare the various patterns of active eigenvalue multiplicities. Seven runs of Gradient Bundle generated an iterate on the boundary and one run failed to converge to an approximate minimizer; these are not shown. The 12 approximate minimizers are available in [6].

In contrast, all runs of Newton Barrier from many different starting points led to the same minimizer (actually, the same set of minimizers for the various δ); see Fig. 11. Comparison of the iterates indicates that, as the robustness parameter δ decreases, the Newton Barrier minimizer converges either to Gradient Bundle's Local Minimizer 5, with an active quadruple real eigenvalue and an active simple conjugate pair, or (somewhat more likely) to its Local Minimizer 6, with an active double real eigenvalue and an active double conjugate pair. The inaccuracy inherent in the computations prevents us from being certain which is the case. Indeed, it is possible, though doubtful, that Local Minimizers 5 and 6 are actually both approximations to the same exact local minimizer of $\alpha(A(x))$. In any case, both approximate minimizers are unstable, in the sense that the spectral abscissa is positive; Newton Barrier was unable to find a stable matrix. However, Gradient Bundle found two stable Local minimizers (1 and 2).

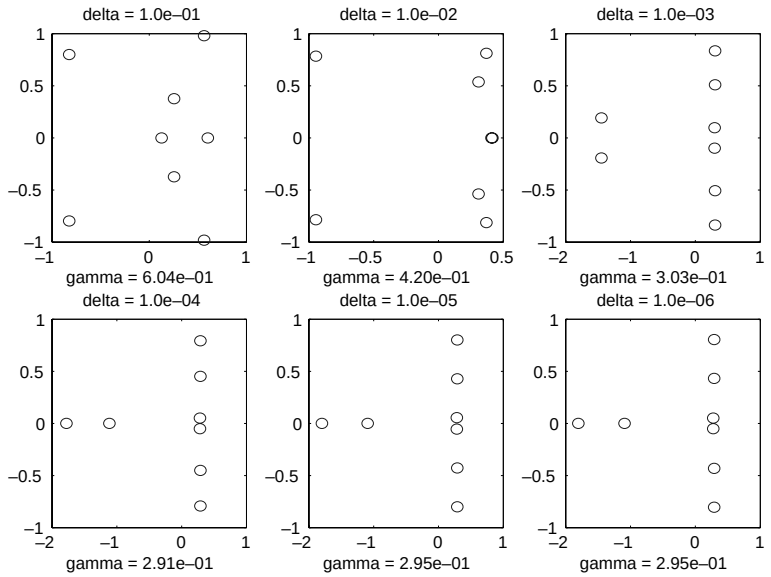

Fig. 11. Example Random(a). Eigenvalues after *all* outer iterations of Newton Barrier method to minimize $\alpha_\delta(A(x))$, for various choices of δ. Runs from many different starting points found the same local minimizer.

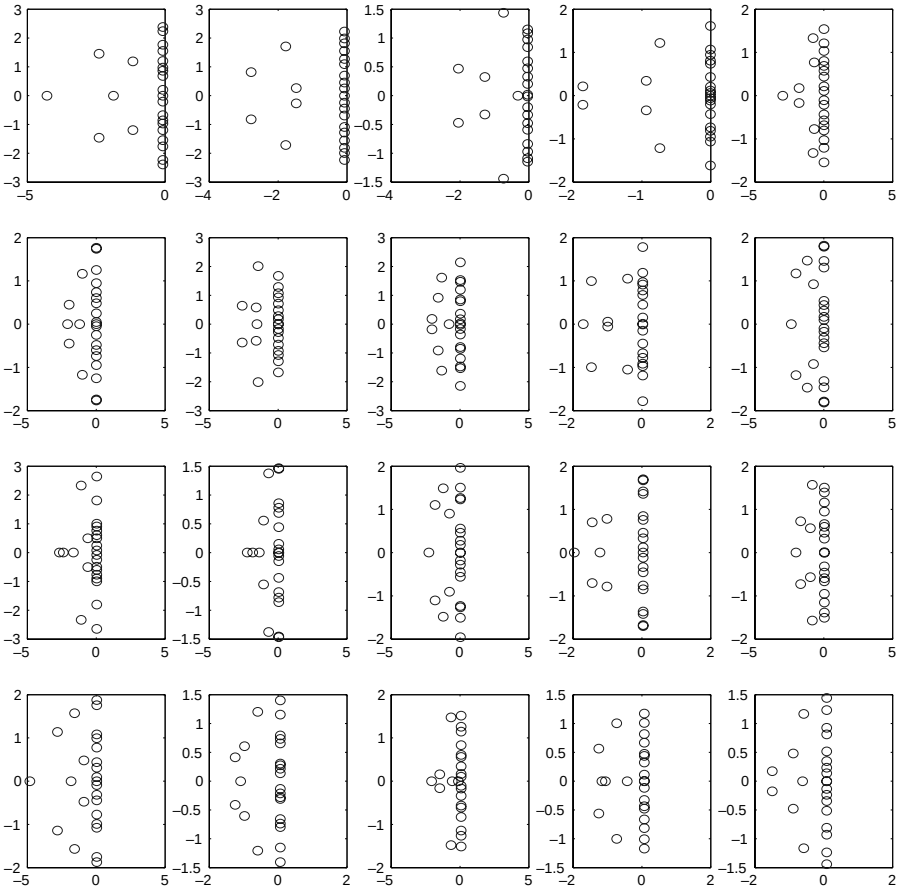

Fig. 12. Example Random(b). Eigenvalues after *all* outer iterations of Gradient Bundle method to minimize $\alpha(A(x))$. Runs from 20 different starting points apparently found 20 local minimizers.

(b) The second of our two random examples has $n = m = 25$; the randomly generated matrices are sparse. Fig. 12 shows the eigenvalues generated by 20 different runs of Gradient Bundle. The subplots are ordered by spectral abscissa; the first four minimizers are stable. We cannot be sure whether the 20 approximate minimizers found are each approximations to 20 distinct associated local minimizers, or whether some of them are poor approximations to the same minimizer, but given the multitude of solutions for the much smaller example Random(a), the former seems likely. This example is much too large for Newton Barrier to run in a reasonable amount of time, but Gradient Bundle had no difficulties with it. However, local minimizers are hard to find with any accuracy because of the high eigenvalue multiplicities.

6. Conclusions

We have presented two methods for optimizing matrix stability. Gradient Bundle is a novel algorithm, inspired by bundle methods for convex optimization, but applicable to non-Lipschitz problems. We believe it offers great versatility and reliability at a very reasonable cost. Applied to spectral abscissa minimization, it demonstrates beyond doubt that spectral abscissa minimizers are typically associated with multiple eigenvalues, and that local minimizers can nonetheless be approximated quite accurately, thus locally optimizing asymptotic stability of the associated dynamical systems.

Newton Barrier is a well-known-approach to smooth, non-convex optimization with inequality constraints, including non-linear semidefinite constraints. We used Newton Barrier to carry out robust spectral abscissa minimization, demonstrating that the minimizers indeed, as expected, converge to spectral abscissa minimizers as the robustness parameter δ is decreased. The motivation is that large transient peaks in the associated dynamical system solution trajectories can be avoided by choosing δ not too small. Although an interesting concept, the approach is too expensive to be practical for any but the smallest problems. Furthermore, if δ is chosen too large, it may be impossible to achieve asymptotic stability, while if δ is chosen too small, ill conditioning may prevent convergence.

Although optimizing stability is an attractive approach to stabilization, inherent difficulties remain. One is the longstanding question of how best to balance transient system behavior with asymptotic stability. Even if an ideal compromise minimization objective were known, another stumbling block is that there may be so many local minimizers that finding any that are stable remains very difficult.

Appendix A. The barrier gradient and Hessian

In addition to our function $A(\cdot)$, we define functions of the variable

$$\begin{pmatrix} \gamma \\ x \\ P \end{pmatrix} \in \mathbb{R} \times \mathbb{R}^m \times \mathbf{S}^n$$

(which we write for convenience (γ, x, P)) by

$$\Lambda = (I - P)^{-1},$$

$$\Delta = (P - \delta I)^{-1},$$

$$Q = (2\gamma P - PA - A^{\mathrm{T}}P)^{-1},$$

$$\beta = (\psi^2 - x^{\mathrm{T}}x)^{-1}.$$

So the barrier function is

$$B_\mu = \gamma - \mu(\log\det \Lambda^{-1} + \log\det \Delta^{-1} + \log\det Q^{-1} + \log \beta^{-1}).$$

We also define functions, for $k = 1, 2, \ldots, m$,

$$P_k = PA_k + A_k^{\mathrm{T}} P,$$

$$Q_k = A_k Q + Q A_k^{\mathrm{T}}.$$

A straightforward application of the chain rule and the fact that the function $X \in \mathbf{S}^n \mapsto \log\det X$ has gradient X^{-1} show that the gradient of the barrier function is given by

$$\nabla B_\mu = \begin{pmatrix} D_\gamma B_\mu \\ \nabla_x B_\mu \\ \nabla_P B_\mu \end{pmatrix} = \begin{pmatrix} 1 - 2\mu\mathrm{tr}(PQ) \\ \mu(\mathrm{tr}(P_k Q))_1^m + 2\mu\beta x \\ \mu(\Lambda - \Delta - 2\gamma Q + AQ + QA^{\mathrm{T}}) \end{pmatrix},$$

where $(\cdot_k)_1^m$ denotes a vector of length m indexed by k.

Consider a base point $(\bar\gamma, \bar x, \bar P)$: we indicate the value of any of the above functions at this point by a bar. In particular, the gradient of the barrier at this point is just

$$\begin{pmatrix} 1 - 2\mu\mathrm{tr}(\bar P \bar Q) \\ \mu(\mathrm{tr}(\bar P_k \bar Q))_1^m + 2\mu\bar\beta\bar x \\ \mu(\bar\Lambda - \bar\Delta - 2\bar\gamma\bar Q + \bar A\bar Q + \bar Q\bar A^{\mathrm{T}}) \end{pmatrix}.$$

Now consider a nearby point $(\bar\gamma, \bar x, \bar P) + (\mathrm{d}\gamma, \mathrm{d}x, \mathrm{d}P)$. Expanding up to first-order shows that at this point the various functions above have values

$$\begin{aligned} \Lambda &= \bar\Lambda + \bar\Lambda\,\mathrm{d}P\,\bar\Lambda + \mathrm{o}(\cdot), \\ \Delta &= \bar\Delta - \bar\Delta\,\mathrm{d}P\,\bar\Delta + \mathrm{o}(\cdot), \\ Q &= \bar Q - \mathscr{D} + \mathrm{o}(\cdot), \\ \beta &= \bar\beta + 2\bar\beta^2 \bar x^{\mathrm{T}}\,\mathrm{d}x + \mathrm{o}(\cdot), \end{aligned}$$

where $\mathrm{o}(\cdot)$ denotes terms of small order with respect to the size of the perturbation $(\mathrm{d}\gamma, \mathrm{d}x, \mathrm{d}P)$, and

$$\mathscr{D} = \bar Q\left(2\bar\gamma\,\mathrm{d}P + 2\,\mathrm{d}\gamma\,\bar P - \mathrm{d}P\,\bar A - \bar A^{\mathrm{T}}\mathrm{d}P - \sum_k \bar P_k\,\mathrm{d}x_k\right)\bar Q.$$

The Hessian of the barrier function B_μ at the point $(\bar\gamma, \bar x, \bar P)$ is the self-adjoint linear operator

$$\bar H : \mathbb{R}\times\mathbb{R}^m\times\mathbf{S}^n \to \mathbb{R}\times\mathbb{R}^m\times\mathbf{S}^n$$

with the property that

$$\bar H\begin{pmatrix}\mathrm{d}\gamma\\ \mathrm{d}x\\ \mathrm{d}P\end{pmatrix} = \nabla B_\mu\left(\begin{pmatrix}\bar\gamma\\ \bar x\\ \bar P\end{pmatrix} + \begin{pmatrix}\mathrm{d}\gamma\\ \mathrm{d}x\\ \mathrm{d}P\end{pmatrix}\right) - \nabla B_\mu\begin{pmatrix}\bar\gamma\\ \bar x\\ \bar P\end{pmatrix} + \mathrm{o}(\cdot).$$

It is then a calculation to check that the Hessian is defined by

$$\bar{H}\begin{pmatrix} \mathrm{d}\gamma \\ \mathrm{d}x \\ \mathrm{d}P \end{pmatrix} = \mu \begin{pmatrix} 2\mathrm{tr}(\bar{P}\mathscr{D} - \bar{Q}\,\mathrm{d}P) \\ (\mathrm{tr}(-\bar{P}_k\mathscr{D} + \bar{Q}_k\,\mathrm{d}P))_1^m + 2\bar{\beta}\,\mathrm{d}x + 4\bar{\beta}^3\bar{x}\bar{x}^{\mathrm{T}}\,\mathrm{d}x \\ \bar{\Lambda}\,\mathrm{d}P\bar{\Lambda} + \bar{\Delta}\,\mathrm{d}P\bar{\Delta} - 2\bar{Q}\,\mathrm{d}\gamma + 2\bar{\gamma}\mathscr{D} + \sum_k \bar{Q}_k\,\mathrm{d}x_k - \bar{A}\mathscr{D} - \mathscr{D}\bar{A}^{\mathrm{T}} \end{pmatrix}.$$

A Matlab implementation is available in [6].

References

[1] F. Alizadeh, J.-P.A. Haeberly, M.L. Overton, Primal–dual interior-point methods for semidefinite programming: convergence rates, stability, and numerical results, SIAM J. Optim. 8 (1998) 746–768.

[2] V.I. Arnold, On matrices depending on parameters, Russian Math. Surv. 26 (1971) 29–43.

[3] S. Barnett, Matrices in Control Theory, revised ed., Krieger, Malabar, FL, 1984.

[4] S. Boyd, L. El Ghaoui, E. Feron, V. Balakrishnan, Linear Matrix Inequalities in System and Control Theory, SIAM, Philadelphia, 1994.

[5] V. Blondel, M. Gevers, A. Lindquist, Survey on the state of systems and control, Eur. J. Control 1 (1995) 5–23.

[6] J.V. Burke, A.S. Lewis, M.L. Overton, Data for the example problems and Matlab *m*-file for the barrier gradient and Hessian, available from http://www.cs.nyu.edu/faculty/overton/papers/numstab/.

[7] J.V. Burke, A.S. Lewis, M.L. Overton, Optimal stability and eigenvalue multiplicity, Found. Comput. Math. 1 (2001) 205–225.

[8] J.V. Burke, A.S. Lewis, M.L. Overton, Optimizing matrix stability, Proc. Am. Math. Soc. 129 (2001) 1635–1642.

[9] J.V. Burke, A.S. Lewis, M.L. Overton, Approximating subdifferentials by random sampling of gradients, Math. Oper. Res. (2002) (to appear).

[10] J.V. Burke, M.L. Overton, Variational analysis of non-Lipschitz spectral functions, Math. Programming 90 (2001) 317–352.

[11] V. Blondel, J.N. Tsitsiklis, NP-hardness of some linear control design problems, SIAM J. Control Optim. 35 (1997) 2118–2127.

[12] A.V. Fiacco, G.P. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques, Wiley, New York, 1968.

[13] P.E. Gill, W. Murray, M.H. Wright, Practical Optimization (Republished by SIAM Philadelphia, 1990), Academic Press, New York and London, 1981.

[14] L. El Ghaoui, S.-I. Niculescu (Eds.), Recent Advances in Linear Matrix Inequality Methods for Control, SIAM, Philadelphia, 2000.

[15] A. Hassibi, J.P. How, S.P. Boyd. Low-authority controller design via convex optimization, in: Proceedings of IEEE Conference on Decision and Control, Tampa, FL, December 1998.

[16] R.A. Horn, C.R. Johnson, Matrix Analysis, Cambridge University Press, Cambridge, UK, 1985.

[17] J.B. Hiriart-Urruty, C. Lemaréchal, Convex Analysis and Minimization Algorithms, Springer, New York, 1993.

[18] K.C. Kiwiel, Methods of Descent for Nondifferentiable Optimization (Two volumes), Lecture Notes in Mathematics, vol. 1133, Springer, Berlin and New York, 1985.

[19] F. Leibfritz, E.M.E. Mostafa, An interior point constrained trust region method for a special class of nonlinear semi-definite programming problems, Technical report, Universität Trier, Germany, July 2000.
[20] A.S. Lewis, M.L. Overton, Eigenvalue optimization, Acta Numer. (1996) 149–190.
[21] J. Moro, J.V. Burke, M.L. Overton, On the Lidskii-Vishik-Lyusternik perturbation theory for eigenvalues of matrices with arbitrary Jordan structure, SIAM J. Matrix Anal. Appl. 18 (1997) 793–817.
[22] MOSEK Optimization Software, 2001, available from http://www.mosek.com/.
[23] A. Nemirovskii, Several NP-hard problems arising in robust stability analysis, Math. Control Signals Syst. 6 (1993) 99–105.
[24] Y. Nesterov, A. Nemirovskii, Interior Point Polynomial Algorithms in Convex Programming, SIAM, Philadelphia, 1994.
[25] B.T. Polyak, P.S. Shcherbakov, Numerical search of stable or unstable element in matrix or polynomial families: a unified approach to robustness analysis and stabilization, in: A. Garulli, A. Tesi, A. Vicino (Eds.), Robustness in Identification and Control, Lecture Notes Control and Information Sciences, vol. 245, Springer, London, 1999, pp. 344–358.
[26] U.T. Ringertz, Eigenvalues in optimum structural design, in: A.R. Conn, L.T. Biegler, T.F. Coleman, F. Santosa (Eds.), Proceedings of an IMA workshop on Large-scale Optimization, 1996.
[27] R.T. Rockafellar, R.J.B. Wets, Variational Analysis, Springer, New York, 1998.
[28] M.J. Todd, Semidefinite programming, Acta Numer. 10 (2001) 515–560.
[29] L.N. Trefethen, Pseudospectra of linear operators, SIAM Rev. 39 (1997) 383–406.
[30] K.-C. Toh, L.N. Trefethen, The Kreiss matrix theorem on a general complex domain, SIAM J. Matrix Anal. Appl. 21 (1999) 145–165.
[31] M.J. Todd, K.C. Toh, R.H. Tütüncü, On the Nesterov–Todd direction in semidefinite programming, SIAM J. Optim. 8 (1998) 769–796.
[32] M.H. Wright, Interior methods for constrained optimization, Acta Numer. 1 (1992) 341–407.
[33] Thomas P.-Y. Yu, Approxmation order/smoothness tradeoff in Hermite subdivision schemes, in: A.F. Laine, M.A. Unser, A. Aldroubi (Eds.), Wavelet Applications in Signal and Image Processing IV Proc. SPIE 4478, 2001, pp. 117–128.