ESAIM: Control, Optimisation and Calculus of Variations Will be set by the publisher
URL: `http://www.emath.fr/cocv/`

AN INEQUALITY-CONSTRAINED SQP METHOD FOR EIGENVALUE OPTIMIZATION

Vyacheslav Kungurtsev[1], Wim Michiels[2] and Moritz Diehl[3]

Abstract. We consider a problem in eigenvalue optimization, in particular finding a local minimizer of the spectral abscissa - the value of a parameter that results in the smallest magnitude of the largest real part of the spectrum of a matrix system. This is an important problem for the stabilization of control systems. Many systems require the spectra to lie in the left half plane in order for stability to hold. The optimization problem, however, is difficult to solve because the underlying objective function is nonconvex, nonsmooth, and non-Lipschitz. In addition, local minima tend to correspond to points of non-differentiability and locally non-Lipschitz behavior. We present a sequential linear and quadratic programming algorithm that solves a series of linear or quadratic subproblems formed by linearizing the surfaces corresponding to the largest eigenvalues. We present numerical results comparing the algorithms to the state of the art.

1991 Mathematics Subject Classification. 90C55, 90C90, 93D21, 90C26, 35P30, 47J10.

May 2014.

Keywords and phrases: Robust control, stabilizing control, eigenvalue optimization, spectral abscissa, sequential quadratic programming, spectral optimization, nonlinear eigenvalue problem

[1] Computer Science Department and Optimization in Engineering Center (OPTEC), KU Leuven, Kasteelpark Arenberg 10, B-3001 Leuven- Heverlee, Belgium. (`vyacheslav.kungurtsev@cs.kuleuven.be`).

[2] Computer Science Department and Optimization in Engineering Center (OPTEC), KU Leuven, Kasteelpark Arenberg 10, B-3001 Leuven- Heverlee, Belgium. (`wim.michiels@cs.kuleuven.be`).

[3] Department of Microsystems Engineering IMTEK, University of Freiburg, Georges-Koehler-Allee 102, 79110 Freiburg, Germany and Dept. ESAT-STADIUS / OPTEC KU Leuven University, Kasteelpark Arenberg 10, 3001 Leuven, Belgium (`moritz.diehl@imtek.uni-freiburg.de`).

1. Eigenvalue Problem

The problem of interest corresponds to solving an optimization problem of the form,

$$\min_{x\in\Re^n} f(x) = \min_x \alpha(F(x)), \tag{1}$$

where the spectral abscissa α is defined to be,

$$\alpha(F(x)) = \max_i \mathfrak{Re}\left(\lambda_i(F(x))\right),$$

with $\{\lambda_i(F(x))\}$ is the (possibly infinite) spectrum of the matrix $F(x)$ and $F(x)$ depends on x in a smooth and linear way. The spectral abscissa corresponds to the largest real part of the eigenvalues of $F(x)$.

For instance, in the field of linear output feedback control, $F(x)$ is defined to be,

$$F(x) = A + BXC,$$

where A is the open-loop matrix for the system, B the input matrix and C the output matrix, and X is formed by arranging the components of x into a matrix of the appropriate dimensions.

The optimization problem is difficult to solve for several reasons:

(1) It is nonconvex, with possibly many local minimizers and, even arbitrarily close to a local minimizer, the spectral abscissa can have negative curvature.
(2) It is nonsmooth. As the parameter changes, each eigenvalue changes as well, usually in a smooth way, however at points where one smooth eigenvalue surface overtakes another one, or there is a non semi-simple eigenvalue and a more complicated splitting of the spectra occurs, points of nonsmoothness arise. Moreover, local minimizers tend to correspond to these points of derivative discontinuity.
(3) It is non-Lipschitz, in the case of a non semi-simple eigenvalue, at which perturbations of the parameter can result in entirely different spectra (for instance, a complex conjugage pair for $x > x_c$ and two real pairs for $x < x_c$, with the inequalities taken component-wise).

These properties make for a challenging task for optimization algorithms. Standard NLP solvers will not be suitable due to the nonsmoothness of the objective $\alpha(F(x))$ and nonsmooth convex solvers are not appropriate due to the persistent negative curvature of the objective function. Typical algorithmic properties of standard optimization procedures, such as rapid local convergence of Newton-like algorithms, and sufficient descent and bounded steps of proximal-type first order algorithms are invalid with a non-Lipschitzian objective near the solution.

On the other hand, the spectral abscissa is a smooth function almost everywhere (a.e.) in any standard measure of $\Re^n$, so the problem is tractable without necessitating the use of sub-gradients, since gradients can be computed at an arbitrary point with probability one. However, local minimizers correspond to points of nonsmoothness, and so any algorithm seeking a minimizer of $\alpha(F(x))$ should still take into account that the points to which the sequence of iterates it generates should be attracted to points at which no unique gradient is defined, even though the algorithm itself can be defined to calculate gradients.

We present the graph of a two-dimensional problem in Figure 1. In this example, first given in [15], $F(x) = A + BK$, with,

$$A = \begin{pmatrix} 0.1 & -0.03 & 0.2 \\ 0.2 & 0.05 & 0.01 \\ -0.06 & 0.2 & 0.07 \end{pmatrix}, \; B = \frac{1}{2}\begin{pmatrix} -1 \\ -2 \\ 1 \end{pmatrix}, \; K^T = \begin{pmatrix} x_1 \\ x_2 \\ 1.4 \end{pmatrix}$$

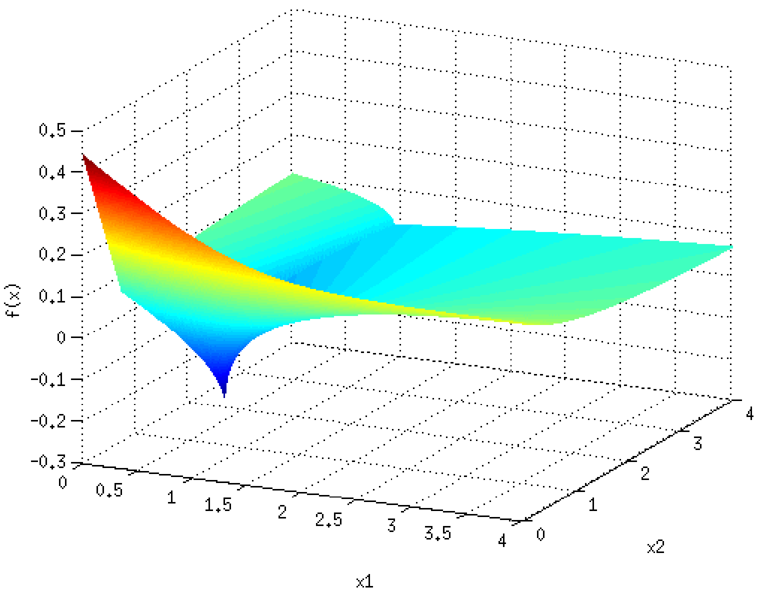

FIGURE 1. $f(x)$ for a two-dimensional eigenvalue optimization problem.

Notice that all of the features of $\alpha(F(x))$ we describe above, nonconvexity, nonsmoothness and non-Lipchitz behavior are evident in the figure.

1.1. **Previous work**

A number of authors have looked at the problem of achieving matrix stability by minimizing the spectral abscissa [1–5, 9, 13]. Typically, a variation of a bundle method is used, in which a "bundle" of gradients is generated by sampling around a point, which serves to approximate the subgradient at a nearby point of nonsmoothness by generating its convex hull from the smooth points around it. This gradient bundle procedure is a generalization of the steepest descent algorithm for nonsmooth problems.

There are a limited number of solvers for eigenvalue optimization problems. In this paper we compare our results to HANSO [14], a code that uses a combination of a BFGS and a gradient bundle search step. BFGS has been shown to exhibit good convergence properties for nonsmooth problems [10]. In particular, we have noticed HANSO tends to interpret the local minimizer cusp-like points as having unboundedly large positive curvature and the resulting second order method is eventually drawn to such points, which the gradient sampling procedure refines.

1.2. **Contribution**

Recall that the spectral abscissa function $\alpha(F(x))$ is smooth almost everywhere. This is because for each eigenvalue, for a.e. x, there locally exists a smooth surface $\lambda_i(x)$ with respect to x that describes how the eigenvalue changes with respect to x. Locally, $\alpha(F(x))$ corresponds to the surface that is associated with the eigenvalue with the maximal real part. Points where $\alpha(F(x))$ is nonsmooth correspond to two situations: 1) a switch in which eigenvalue corresponds to the maximal real part as x changes, or 2) a splitting of the eigenvalue surface set at a non-semi-simple eigenvalue, in which case the maximal surface can be non-Lipschitz.

The procedures implemented and analyzed thus far only use the maximal eigenvalue surface, and only after extensively sampling regions with other maximal eigenvalues does it refine its steps relative to the behavior of the function between these regions. Instead, we consider the possibility of explicitly including every one, or at least a subset, of the eigenvalue surfaces at each point in the calculation of a step with a linearized or quadratic approximation. In the case where nonsmoothness corresponds to changes in which one eigenvalue surface overtakes another one, this should encourage better steps by anticipating in the model where the maximal eigenvalue changes, and as a result taking a step that considers both surfaces and, e.g., moving down a ridge in the topography of $\alpha(F(x))$. In the case of points corresponding to the spectral abscissa associated with a non semi-simple eigenvalue, we introduce memory to include the surface on either side of the point of nonsmoothness, which we expect to decrease the number of iterations needed for the optimization procedure to take steps that decrease the objective function near these non-Lipschitz points.

2. SL/QP for Eigenvalue Optimization

In this section we present a sequential quadratic programming (SQP) method, as well as a more simplified Sequential Linear Programming (SLP) method for solving eigenvalue problems. The algorithm is a trust-region based method based on the realization that the problem $\min_{x\in\Re^n} \alpha(F(x))$ can be rewritten as,

$$\begin{array}{rl} \min_{\gamma\in\Re, x\in\Re^n} & \gamma, \\ \text{subject to} & \gamma \geq \mathfrak{Re}(\lambda_i(F(x))) \text{ for all } i. \end{array} \tag{2}$$

One key observation to inspire the algorithm is that the number of points at which the objective function $\alpha(F(x))$ in (1) is nonsmooth is of measure zero in the Lebesgue space $\Re^n$. This implies that for a.e. x, the function $\alpha(F(x))$ is a locally smooth surface. This surface corresponds to the value of $\lambda_0(F(x))$ as a function of x. However, since the points of interest tend to be nonsmooth regions of the original objective function that typically corresponds to the intersection of different eigenvalue surfaces, we wish to include at least several eigenvalue surfaces corresponding to the next largest real eigenvalues.

Locally, we can express this as a plane through the point $(x, \alpha(F(x)))$ with the gradient $\nabla_x \alpha(F(x)) = \nabla_x \lambda_i(F(x))$. In the case of a simple eigenvalue, which is the case with probability one, we can calculate both this vector as well as $\nabla^2_{xx}\lambda_i(F(x))$ by the formulas [11],

$$\nabla_x \lambda_i(F(x)) = \frac{u_i^* \frac{dF}{dx} v_i}{u_i^* v_i}, \tag{3}$$

$$\nabla^2_{xx}\lambda_i(F(x)) = \frac{2u_i^* \left(\frac{dF}{dx}\right) K_i(\lambda_i I - F(x))^\dagger K_i \left(\frac{dF}{dx}\right) v_i}{u_i^* v_i}, \tag{4}$$

where u_i and v_i are the left and right eigenvectors of $F(x)$ corresponding to eigenvalue i, u^* corresponds to the conjugate of u, $\dagger$ is the pseudo-inverse, and $K = I - v_i u_i^*/(u_i^* v_i)$.

The Lagrangian function for the problem (2) is defined as,

$$L(\gamma, x) = \gamma - \sum_i y_i(\gamma - \mathfrak{Re}\left(\lambda_i(F(x))\right)), \tag{5}$$

where y is the vector of Lagrange multipliers.

This naturally suggests the SQP method wherein a sequence of subproblems of the form,

$$\begin{array}{rl} \min_{\Delta x, \Delta\gamma} & \Delta\gamma + \frac{1}{2}\Delta x^T H_k \Delta x, \\ \text{subject to} & \Delta\gamma + \alpha(F(x_k)) \geq \mathfrak{Re}(\lambda_i(F(x_k))) + \mathfrak{Re}(\nabla_x \lambda_i(F(x_k)))^T \Delta x, \forall i, \end{array} \tag{6}$$

for Δx and $\Delta\gamma$, where H_k is a Lagrangian Hessian term at k. Note that this is a slightly more "accurate" version of a standard SQP algorithm, in that at each iteration we discard $\Delta\gamma$ and compute $\gamma_k = \alpha(F(x_k))$ instead. To determine the Lagrangian Hessian, we do not use the multipliers from the quadratic programs as estimates for the Lagrangian multipliers. Recall that one eigenvalue, possibly with a usually small multiplicity, has the largest real part. In the sense of problem (2), these are the *active* constraints. So, at the start of each iteration, we set the Lagrange multiplier to have $1/m$ in each component corresponding to a maximal eigenvalue of multiplicity m, and 0 otherwise. Using this multiplier, we calculate the Hessian,

$$H_k = \sum_{i \in \mathfrak{M}_k} \frac{1}{m} \mathfrak{Re} \nabla^2_{xx} \lambda_i(F(x_k))),$$

where $\mathfrak{M}_k$ is the set of maximal eigenvalues and $m = |\mathfrak{M}_k|$. Note that the Lagrangian function (5) is linear with respect to γ and so the Hessian only has blocks corresponding to x. At each step of the SQP we need only update $x_{k+1} = x_k + \Delta x$. The solution $\Delta\gamma$ is discarded.

There are two primary additional features to the basic procedure we have presented in order to make the method more practically successful. First, recall from section 1, that the underlying problem is non-convex. This implies that at any local quadratic approximation of an eigenvalue surface, the Hessian could be indefinite or even negative definite. This implies that the approximating quadratic program (6) could be unbounded below. In practice, SQP algorithms typically modify ("convexify") the Hessian to be positive definite. However, this is typically a strategy to ensure descent with the intention that eventually the algorithm will reach a region of positive curvature, and the Hessian approximation converges to the exact Hessian in the reduced space of the active constraints. In this case, even at the solution the Hessian could have negative curvature, and so this framework is not appropriate. Instead, we constrain the problem with a trust-region. Since we have linear constraints, we use an infinity norm trust-region, which acts as a "box" limiting the magnitude of the maximal component of Δx.

$$\begin{array}{ll} \min_{\Delta x, \Delta\gamma} & \Delta\gamma + \frac{1}{2}\Delta x^T H_k \Delta x, \\ \text{subject to} & \Delta\gamma + \alpha(F(x_k)) \geq \mathfrak{Re}(\lambda_i(F(x_k))) + \mathfrak{Re}(\nabla_x \lambda_i(F(x_k)))^T \Delta x, \, \forall i, \\ & ||\Delta x||_\infty \leq \Delta_k. \end{array} \tag{7}$$

We update the trust-region simply by increasing it if we achieve descent, and decreasing it otherwise. For consistency with convergence theory [6], we would enforce sufficient decrease conditions with respect to a merit function. However a) since we seek a minimizer of $\alpha(F(x))$ alone, and b) liberal criteria of acceptance of the step is practically equivalent to this condition, we proceed as in the line-search criteria for the gradient bundle method [5] to just enforce descent.

In order to not discard iterations, we follow the mixed trust-region/line-search procedure presented by Gertz [7], in which a backtracking line search reduces the size of the step t until decrease is achieved $(\alpha(F(x_k + t\Delta x)) < \alpha(F(x_k)))$, and the next trust-region radius corresponds to $t||\Delta x||$.

$$\Delta_{k+1} = \begin{cases} \gamma\Delta_k & \text{if } \alpha(F((x_k + \Delta x)) < \alpha(F(x_k)) \\ t||\Delta x|| & \text{otherwise,} \end{cases} \tag{8}$$

where γ is a constant satisfying $\gamma > 1$.

In addition, recall that one eigenvalue surfaces overtaking another one is just one possible case causing nonsmoothness of $\alpha(F(x))$. The other case corresponds to a non semi-simple eigenvalue, for which there could be qualitatively different eigenvalue surface combinations on either side of the ridge of nonsmoothness. In order to account for this, we added another feature to the algorithm, after observing a certain phenomenon that was typical with the original SQP algorithm with the trust region but without this additional feature. In many cases, the algorithm jammed near ridges of this kind and would frequently converge onto the ridge rather than move down along it. This is because locally, the slope of $\alpha(F(x))$ is steeper in the direction towards the ridge than perpendicular to it, and furthermore has negative curvature along that direction, so a local approximation that regards only the eigenvalue surfaces at a point on one side of the ridge will result in the step of steepest decrease being in this direction. Since the surface on the other side of the ridge is not accounted for on the original side, this is not incorporated directly into the subproblem.

To remedy this, we added "memory" to the SQP method with a set $\mathcal{M}$. If, at any point during a failed step, which occurs if the step skips across the ridge and rises along the other side too far (typically, in the subsequent iteration, the size of the trust-region decreases), the procedure stores the failed point $x^{(|\mathcal{M}|+1)} = x^{(i)} = x_k + \Delta x_k$ and the value $\mathfrak{Re}(\lambda_{(i)}(F(x^{(i)})))$ and slope $\mathfrak{Re}(\nabla_x \lambda_{(i)}(F(x^{(i)})))$ of the surface at that point in the memory set $\mathcal{M}$. Then, if in a future iteration, the current point x_l satisfies $||x_l - x^{(i)}||_\infty \leq \Delta_k$, then we include this linearized surface in the QP subproblem.

$$\begin{array}{ll} \min_{\Delta x,\Delta\gamma} & \Delta\gamma + \frac{1}{2}\Delta x^T H_k \Delta x, \\ \text{subject to} & \Delta\gamma + \alpha(F(x_k)) \geq \mathfrak{Re}(\lambda_i(F(x_k))) + \mathfrak{Re}(\nabla_x \lambda_i(F(x_k)))^T \Delta x,\ \forall i \\ & \Delta\gamma + \alpha(F(x_k)) \geq \mathfrak{Re}(\lambda_{(i)}(F(x^{(i)}))) \\ & \qquad\qquad + \mathfrak{Re}(\nabla_x \lambda_{(i)}(F(x^{(i)})))^T (x_k + \Delta x - x^{(i)}),\ i \in M_k \\ & ||\Delta x||_\infty \leq \Delta_k, \end{array} \tag{9}$$

where $M_k \subset \mathcal{M}$ represents the points $x^{(i)}$ satisfying $||x^{(i)} - x_k|| \leq \Delta_k$.

Finally, we consider two simplifications of the subproblem. First, we consider dropping the Hessian term, to formulate a sequential linear programming procedure. We note that the lack of a Lipschitz property makes local superlinear convergence impossible, and it is quite possible that second derivatives will not reduce the computation time. We present the SLP version below, for simplicity, and discuss a comparison of the two in the numerical results section.

Second, in the case of large scale problems, where we use iterative procedures to compute a subset of eigenvalues, we include only these, or a subset of these, in the subproblem. The set N_k of eigenvalues we consider may change each iteration. We may, for example, want to calculate all eigenvalues in the right half plane relative to a certain minimal value of $\bar{\lambda}_c$. We have to decide how many eigenvalues to compute (N_k) each iteration. This value should be greater than or equal to the number of active (maximal) eigenvalues, and otherwise is a heuristic choice depending on how many surfaces we want to keep track of, locally. Typically, we use $2n$ maximal eigenvalues for these large-scale problems, which is the same heuristic as the number of sample points to use in a gradient bundle method [5].

$$\begin{array}{ll} \min_{\Delta x,\Delta\gamma} & \Delta\gamma, \\ \text{subject to} & \Delta\gamma + \alpha(F(x_k)) \geq \mathfrak{Re}(\lambda_i(F(x_k))) \\ & \qquad\qquad + \mathfrak{Re}(\nabla_x \lambda_i(F(x_k)))^T \Delta x,\ i \in \{0, ..., N_k\} \\ & \Delta\gamma + \alpha(F(x_k)) \geq \mathfrak{Re}(\lambda_{(i)}(F(x^{(i)}))) \\ & \qquad\qquad + \mathfrak{Re}(\nabla_x \lambda_{(i)}(F(x^{(i)})))^T (x_k + \Delta x - x^{(i)}),\ i \in M_k \\ & ||\Delta x||_\infty \leq \Delta_k. \end{array} \tag{10}$$

We present a summary of the large-scale version here. Since the problem is nonconvex, we use multiple starting points in order to attempt to find a local minimizer with a low global value as well.

The stopping criterion corresponds to the step becoming small, without any new information (memory) being added at the current iteration.

2.1. **Statement of the algorithm**

Algorithm 1 SLP Algorithm for Eigenvalue Optimization.

```
1: Define constants $\gamma > 1$, $\Delta_m > 0$, and $\delta_m > 0$.
2: for $S$ times do
3:     Randomly select starting point $x_0$.
4:     Calculate initial $\{\lambda_i(F(x_0))\}$ and $\{\nabla_x \lambda_i(F(x_0))\}$ for $i \in \{0, .., N_0\}$.
5:     while $((\Delta_k < \Delta_m$ and $||\Delta x|| < \delta_m)$ or $\mathcal{M}_k \neq \mathcal{M}_{k-1})$ do
6:         Solve (10) for $\Delta x_k$.
7:         Calculate $\{\lambda_i(F(x_k + \Delta x_k))\}$ and $\{\nabla_x \lambda_i(F(x_k + \Delta x_k))\}$ for $i \in \{0, .., N_k\}$
8:         if $\alpha(F(x_k + \Delta x_k)) < \alpha(F(x_k))$ then
9:             Set $x_{k+1} \leftarrow x_k$
10:            Set $\Delta_{k+1} \leftarrow \gamma \Delta_k$.
11:        else
12:            Store $x^{(|\mathcal{M}|+1)} = x_k + \Delta x_k$ and its slope and value of $\alpha(F(\cdot))$ in $\mathcal{M}$.
13:            Find $t$ such that $\alpha(F(x_k + t\Delta x_k)) < \alpha(F(x_k))$.
14:            Set $x_{k+1} \leftarrow x_k + t\Delta x_k$.
15:            Set $\Delta_{k+1} \leftarrow t||\Delta x_k||$.
16:        end if
17:        Set $k \leftarrow k + 1$.
18:    end while
19:    Add the last point $(x_f, \alpha(F(x_f)))$ to $\mathcal{F}$.
20: end for
       return $\{x_f, \alpha(F(x_f))\}$ corresponding to the lowest value of $\alpha(F(x_f))$ in $\mathcal{F}$.
```

3. Numerical Results for Linear Eigenvalue Problems

We compare SLP/SQP and HANSO on a set of linear control problems arising from COMPlib [8]. COMPlib returns a set of linear control matrices, of which we take three to for the system $F(x) = A + BXC$. Out of 124 examples, we picked 99 for which the number of rows (and columns) of A was less than or equal to 50.

For these comparisons, we will compare both the time of execution as well as the value of the final solution. We summarize the results below in Table 1. In general, for most problems (about two-thirds), SLP finds a lower minimum than HANSO. It appears to be faster than HANSO with gradient sampling, and slower than BFGS alone. Interestingly, gradient sampling does not, on average, tend to improve the performance of HANSO vis-a-vis SLP. The difference in values were not trivial, also. In particular, for SLP, there were 74 problems where the final spectral abscissa (median of 50 runs) was less than zero, as compared to 63 for HANSO with gradient bundle and 59 for HANSO without.

We also performed a comparison of SLP versus SQP on the test problems. The results, in Table 2 indicate that SLP tends to take more iterations but is more reliable. We conjecture that this is because second derivatives provide for a more accurate solution to the subproblems, but can be unstable due to the pseudoinverse in the formula for $\nabla_{xx}\lambda_i(F(x))$ (4).

Table 1. Number of times SLP outperformed HANSO (out of 99 problems, median of 50 runs).

	in value	in time
HANSO	86	91
HANSO without gradient sampling	86	4

Table 2. Performance of SLP as compared to SQP (out of 99 problems, median of 50 runs). (Notice that the sum of the best in value column does not add to 99, this is because for four problems the result was exactly equal)

Algorithm	best in value	best in time
SLP	68	22
SQP	27	77

4. Nonlinear Eigenvalue Problems

In this section we consider time-delay systems of the form,

$$v'(t) = \sum_{j=0}^{m} A_j(x)v(t-\tau_j).$$

In this case, we have a nonlinear eigenvalue problem. To solve for the eigenvalues, we find the solutions $\lambda(x)$ of,

$$\det(\Lambda(\lambda;x)) = 0,$$

with,

$$\Lambda(\lambda;x) = \lambda I - A_0(x) - \sum_{j=1}^{m} A_j(x)e^{-\lambda\tau_j}.$$

The number of eigenvalues in this case is generally infinite, but within any right half-plane the number of eigenvalues is finite [12]. In the numerical experiments, we find all of those which are to the right of $r = -1/\tau_m$, where τ_m is the maximum of the delays. If none are found, we repeatedly double r until at least one eigenvalue appears.

It can be shown that, in the case where each $A_i(x)$ depends smoothly on x and where the eigenvalue has multiplicity 1, the derivative of the surface corresponding to each eigenvalue λ_i is equal to [12],

$$\nabla_x\lambda_i = \frac{u_i^*\left(\frac{\partial A_0}{\partial x} + \sum_{j=1}^{m}\frac{\partial A_j}{\partial x}e^{-\lambda_i\tau_j}\right)v_i}{u_i^*\left(I + \sum_{j=1}^{m}\tau_j e^{-\lambda\tau_j}A_j\right)v_i}.$$

Note that the term in the numerator, $\frac{\partial A_0}{\partial x} + \sum_{j=1}^{m}\frac{\partial A_j}{\partial x}e^{-\lambda_i\tau_j}$ corresponds to $\nabla_x F(x)$, and so we can use the same formula for the second derivative, (4) as in the linear case.

We compare the performance of 500 trials of SLP and HANSO with and without gradient sampling for two time-delay systems described in [16].

The first example is a third-order feedback controller system of the form,

$$v'(t) = Av(t) + B(x)v(t-5),$$

Table 3. Number of times SLP outperformed HANSO and SQP (out of 500 sample runs).

	in value	in time
HANSO	250	500
HANSO without gradient sampling	257	447
SQP	227	344

Table 4. Mean (standard deviation) for values and times for HANSO, HANSO without gradient sampling, SLP, and SQP (out of 500 sample runs).

	value	time
HANSO	-0.074 (0.036)	177 (30.3)
HANSO without gradient sampling	-0.069 (0.036)	6.1 (2.7)
SLP	-0.081 (0.053)	4.6 (1.1)
SQP	-0.088 (0.055)	5.3 (1.6)

with A and $B(x)$ defined to be,

$$A = \begin{pmatrix} -0.08 & -0.03 & 0.2 \\ 0.2 & -0.04 & -0.005 \\ -0.06 & -0.2 & -0.07 \end{pmatrix}$$

and

$$B(x) = \begin{pmatrix} -0.1 \\ -0.2 \\ 0.1 \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix}.$$

We present the results below in Table 3. In this case, statistically, SLP finds no lower or higher a minimizer than HANSO with or without gradient sampling. However, it almost always finds it in less time. In this case, SLP appears to also perform similarly in terms of finding the lowest minimizer as SQP, but now takes less time. We also include the mean and standard deviation of the values and times in Table 4. If we take a Student t-test for the difference in means for the times, then for 0.0001 significance, the difference in both mean times and values is significant between SLP and HANSO (both with and without gradient sampling).

The next example is given below,

$$\begin{aligned}
&T_h\dot{x}_h(t) = -x_h(t-\eta_h) + K_b x_a(t-\tau_b) + K_u x_{h,set}(t-\tau_u),\\
&T_a\dot{x}_a(t) = -x_a(t) + x_c(t-\tau_e) + K_a(x_h(t) - \tfrac{1+q}{2}x_a(t) - \tfrac{1-q}{2}x_c(t-\tau_e)),\\
&T_d\dot{x}_d(t) = -x_d(t) + K_d x_a(t-\tau_d),\\
&T_c\dot{x}_c(t) = -x_c(t-\eta_c) + K_c x_d(t-\tau_c),\\
&\dot{x}_e(t) = -x_c(t) + x_{c,set}(t),
\end{aligned}$$

with,

$$x_{h,set}(t) = \begin{pmatrix} K_1 & K_2 & K_3 & K_4 & K_5 \end{pmatrix} \begin{pmatrix} x_h(t) & x_a(t) & x_d(t) & x_c(t) & x_e(t) \end{pmatrix}^T.$$

The results, which are qualitatively similar as in the first example, are given in Tables 5 and 6.

Table 5. Number of times SLP outperformed HANSO (out of 500 sample runs).

	in value	in time
HANSO	268	500
HANSO without gradient sampling	283	466

Table 6. Mean (standard deviation) for values and times for HANSO, HANSO without gradient sampling, SLP, and SQP (out of 500 sample runs).

	value	time
HANSO	-0.077 (0.0052)	2,200 (6400)
HANSO without gradient sampling	-0.067 (0.0054)	69.0 (28)
SLP	-0.083 (0.0062)	83.5 (140)
SQP	-0.088 (0.0103)	75.5 (76)

5. Conclusion

In this paper we studied the eigenvalue optimization problem of minimizing the spectral abscissa, the maximum real eigenvalue part. This problem is important in the stabilization of control and delay systems. We presented an algorithm that incorporated linear and quadratic models of eigenvalue surfaces corresponding to different eigenvalues in a sequential linear and sequential quadratic programming framework. We expected this to produce a faster and possibly more reliable algorithm for finding minima of the spectral abscissa, since the model is capable of approximating the spectral abscissa surface past points of nonsmoothness.

Our numerical results for both linear and nonlinear problems borne out our expectation, and we find that, in particular, the sequential linear variant of the algorithm tends to, in general, outperform the most comparable competitor, HANSO. For linear problems, it is faster and more reliable than HANSO with gradient sampling, and more reliable but slower than HANSO without gradient sampling. For nonlinear problems, SLP is faster than and at least equally as reliable as both HANSO and HANSO without gradient sampling.

We acknowledge that this paper does not include a convergence proof, and we believe with the algorithm as it stands, it may not be certifiably convergent, since the stopping criteria do not necessarily correspond to a stationary Clarke subdifferential condition, unlike as in gradient sampling. Future research may include adapting the algorithm to ensure convergence, on the theoretical end, and testing on interesting large-scale delay and control applications, on the applied end.

6. Acknowledgements

This research was supported by Research Council KUL: PFV/10/002 Optimization in Engineering Center OPTEC, GOA/10/09 MaNet and GOA/10/11 Global real- time optimal control of autonomous robots and mechatronic systems. Flemish Government: FWO: PhD/postdoc grants; IWT: PhD Grants, projects: Eurostars SMART; Belgian Federal Science Policy Office: IUAP P7 (DYSCO, Dynamical systems, control and optimization, 2012-2017); EU: FP7-SADCO (MC ITN-264735), FP7-TEMPO(MC ITN-607957), ERC HIGHWIND (259 166).

References

[1] P. Apkarian, D. Noll, and O. Prot. A trust region spectral bundle method for nonconvex eigenvalue optimization. *SIAM Journal on Optimization*, 19(1):281–306, 2008.

[2] J. Burke, A. Lewis, and M. Overton. Optimizing matrix stability. *Proceedings of the American Mathematical Society*, 129(6):1635–1642, 2001.

[3] J. V. Burke, D. Henrion, A. S. Lewis, and M. L. Overton. Stabilization via nonsmooth, nonconvex optimization. *Automatic Control, IEEE Transactions on*, 51(11):1760–1769, 2006.

[4] J. V. Burke, A. S. Lewis, and M. L. Overton. Two numerical methods for optimizing matrix stability. *Linear Algebra and its Applications*, 351:117–145, 2002.

[5] J. V. Burke, A. S. Lewis, and M. L. Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005.

[6] A. R. Conn, N. I. Gould, and P. L. Toint. *Trust region methods*, volume 1. Siam, 2000.

[7] E. M. Gertz et al. *Combination trust-region line-search methods for unconstrained optimization*. PhD thesis, 1999.

[8] F. Leibfritz. Compleib: Constrained matrix optimization problem library, 2006. *URL http://www. complib. de*, 2010.

[9] A. S. Lewis and M. L. Overton. Eigenvalue optimization. *Acta numerica*, 5:149–190, 1996.

[10] A. S. Lewis and M. L. Overton. Nonsmooth optimization via quasi-newton methods. *Mathematical Programming*, 141(1-2):135–163, 2013.

[11] J. R. Magnus. On differentiating eigenvalues and eigenvectors. *Econometric Theory*, 1(2):pp. 179–191, 1985.

[12] W. Michiels and S.-I. Niculescu. *Stability and stabilization of time-delay systems: an eigenvalue-based approach*, volume 12. Siam, 2007.

[13] D. Noll and P. Apkarian. Spectral bundle methods for non-convex maximum eigenvalue functions: first-order methods. *Mathematical programming*, 104(2-3):701–727, 2005.

[14] M. Overton. Hanso: a hybrid algorithm for nonsmooth optimization. *Available from cs. nyu. edu/overton/software/hanso*, 2009.

[15] J. Vanbiervliet, B. Vandereycken, W. Michiels, S. Vandewalle, and M. Diehl. The smoothed spectral abscissa for robust stability optimization. *SIAM Journal on Optimization*, 20(1):156–171, 2009.

[16] J. Vanbiervliet, K. Verheyden, W. Michiels, and S. Vandewalle. A nonsmooth optimisation approach for the stabilisation of time-delay systems. *ESAIM: Control, Optimisation and Calculus of Variations*, 14(03):478–493, 2008.