ESAIM: COCV 14 (2008) 478–493
DOI: 10.1051/cocv:2007060

ESAIM: Control, Optimisation and Calculus of Variations
www.esaim-cocv.org

A NONSMOOTH OPTIMISATION APPROACH FOR THE STABILISATION OF TIME-DELAY SYSTEMS

Joris Vanbiervliet[1], Koen Verheyden[1], Wim Michiels[1] and Stefan Vandewalle[1]

Abstract. This paper is concerned with the stabilisation of linear time-delay systems by tuning a finite number of parameters. Such problems typically arise in the design of fixed-order controllers. As time-delay systems exhibit an infinite amount of characteristic roots, a full assignment of the spectrum is impossible. However, if the system is stabilisable for the given parameter set, stability can in principle always be achieved through minimising the real part of the rightmost characteristic root, or spectral abscissa, in function of the parameters to be tuned. In general, the spectral abscissa is a nonsmooth and nonconvex function, precluding the use of standard optimisation methods. Instead, we use a recently developed bundle gradient optimisation algorithm which has already been successfully applied to fixed-order controller design problems for systems of ordinary differential equations. In dealing with systems of time-delay type, we extend the use of this algorithm to infinite-dimensional systems. This is realised by combining the optimisation method with advanced numerical algorithms to efficiently and accurately compute the rightmost characteristic roots of such time-delay systems. Furthermore, the optimisation procedure is adapted, enabling it to perform a local stabilisation of a *nonlinear* time-delay system along a branch of steady state solutions. We illustrate the use of the algorithm by presenting results for some numerical examples.

Mathematics Subject Classification. 65Q05, 65K10, 90C26

Received July 3, 2006. Revised January 18, 2007.
Published online November 21, 2007.

1. Introduction

Time delays arise naturally in a wide range of scientific domains. Think of finite transmission times in telecommunication, transportation lags in chemical or biological processes and feedback delays in controlled industrial processes. They can also be introduced in a more artificial fashion, *e.g.* when approximating systems with distributed parameters by low-order delay models. See [14] for more examples. Because the inclusion of time delays may have a large impact on the behaviour and stability of a system, it is important to take them into account in mathematical models. This can be crucial to obtain a qualitatively accurate understanding of the system under consideration. The drawback is that by introducing time delays to a system model, its complexity is heavily increased.

Keywords and phrases. Stabilisation, delay differential equations, nonsmooth optimisation, bundle gradient methods.

[1] Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium; joris.vanbiervliet@cs.kuleuven.be

We consider time-delay systems of the form

$$x'(t) = \sum_{j=0}^{m} A_j(p)\, x(t-\tau_j), \tag{1.1}$$

where $x(t) \in \mathbb{R}^n$ is the vector of state variables, $A_j \in \mathbb{R}^{n\times n}$, $j = 0,\ldots,m$ are the system matrices which smoothly depend on the parameters $p \in \mathbb{R}^{n_p}$, and $\tau_j \geqslant 0$, $j = 0,\ldots,m$ are the fixed delays, with $\tau_0 = 0$.

The stability properties of the steady state solution $x \equiv 0$ of (1.1) are determined by the characteristic roots λ, which are defined as the solutions to the nonlinear *characteristic equation*

$$\det\big(\Lambda(\lambda;\, p)\big) = 0, \tag{1.2}$$

with

$$\Lambda(\lambda;\, p) := \lambda I - A_0(p) - \sum_{j=1}^{m} A_j(p)\mathrm{e}^{-\lambda\tau_j}. \tag{1.3}$$

A system is asymptotically stable if all characteristic roots belong to the open left half plane. In general, the system (1.1) bears an infinite number of characteristic roots and a full placement of all characteristic roots is therefore impossible, since the number of free parameters is finite. However, the number of characteristic roots in any right half-plane, *i.e.* with $\Re(\lambda) \geqslant r$ for $r \in \mathbb{R}$, *is* finite. The stabilisation problem can consequently be interpreted as the problem of finding parameter values p for which the *spectral abscissa function* α, defined by

$$\alpha := p \mapsto \max\{\Re(\lambda) : \lambda \text{ is a characteristic root}\}, \tag{1.4}$$

is strictly negative. This approach, as followed in this paper, consists of directly solving the following optimisation problem:

$$\min_{p} \alpha(p). \tag{1.5}$$

The system is stabilisable if either $\alpha(p)$ is unbounded below, or if (1.5) has a strictly negative solution. In the latter case, the parameters in which this solution is reached, result in an "optimal spectrum", whose characteristic roots are pushed to the left as far as possible in the complex plane and for which the trajectories follow a maximal asymptotic decay rate to the steady state. Note that, from a pure stabilisation point of view, it is not necessary to exactly compute this global minimum, since it suffices to halt the minimisation procedure as soon as parameters are found for which $\alpha(p) < 0$.

Because the spectral abscissa is a nonsmooth function, standard optimisation methods cannot be used to solve (1.5). Instead, we use a so-called bundle gradient method, specifically the *gradient sampling algorithm* [6], an optimisation method that is able to find local minima of general nonsmooth, nonconvex objective functions. In [5], this algorithm was successfully used to find stabilising low-order static output feedback controllers and recently, it was included in HIFOO [7], a MATLAB package for fixed-order controller design and $\mathcal{H}_\infty$-optimisation. In general, these applications can be viewed as finite-dimensional eigenvalue optimisation problems where the number of controller parameters is much smaller than the number of eigenvalues to be controlled. The stabilisation of time-delay systems, being of infinite nature, can consequently be regarded as an extension of the application range of this algorithm to an infinite-dimensional setting.

An alternative approach for the stabilisation of time-delay systems is the class of so-called Lyapunov based methods, where stabilisability conditions and the resulting controller are typically expressed by the solvability of algebraic Riccati equations or by the feasibility of linear matrix inequalities. A drawback of these methods is that, by imposing a special structure on the Lyapunov functional, conservatism is typically induced, in the sense that the resulting stabilisability conditions are sufficient but not necessary. Other possible approaches include algebraic methods (for instance where the delay system is viewed as a system over a ring), methods based on a spectral decomposition (where a finite number of unstable roots is shifted while keeping the others fixed) and methods based on prediction (where a delay-free model is used as a predictor for the system with delay).

These methods typically result in an infinite-dimensional controller, and they are in fact not fit to solve fixed-order stabilisation problems. See [12,18,19] for more on the stabilisation of systems with delays.

The structure of this paper is as follows. Section 2 contains preliminaries on time-delay systems. In Section 3, we examine the optimisation method used to minimise the spectral abscissa $\alpha(p)$. Section 4 outlines the numerical procedures to compute the rightmost, stability-determining characteristic roots, enabling the evaluation of $\alpha(p)$ and its gradient. Finally, in Section 5, three examples illustrate our implementation.

2. Preliminaries on time-delay systems

Some background material on time-delay system of the form (1.1) is presented, based on the monographs [9] and [13]. As the dependence on the model parameters p is not explicitly dealt with in this section, it is suppressed in the notations.

Let $\tau_{\max}$ be the maximal delay of the system (1.1). Contrary to a system of ordinary differential equations, a continuous function on an interval of length $\tau_{\max}$ is needed as a starting condition to fully specify the initial value problem of a time-delay system. The function segment which determines the evolution starting from time t, is called the *state* at t and is denoted by

$$\mathbf{x}_t(\theta) := x(t+\theta), \quad \text{where } \theta \in [-\tau_{\max}, 0].$$

It belongs to the function space $\mathcal{C}([-\tau_{\max}, 0], \mathbb{R}^n)$, *i.e.* the function space of all continuous functions mapping the interval $[-\tau_{\max}, 0]$ into $\mathbb{R}^n$ and equipped with the supremum norm.

Denote by $\mathcal{S}(t)$, for $t \geqslant 0$, the *solution operator* (or *time-integration operator*) which maps an initial state at time zero onto the corresponding state at time t, *i.e.*

$$\mathcal{S}(t)\mathbf{x}_0 = \mathbf{x}_t, \quad \text{with } \mathbf{x}_0 \in \mathcal{C}([-\tau_{\max}, 0], \mathbb{R}^n). \tag{2.1}$$

Since $\mathcal{S}(t)$, for $t \geq 0$, is a strongly continuous semi-group, one can define a corresponding *infinitesimal generator* $\mathcal{A}$. This operator, which can be loosely considered as the right-hand derivative of $\mathcal{S}(t)$ at zero, is defined by

$$\mathcal{A}\mathbf{x}(\theta) = \mathbf{x}'(\theta), \qquad \text{for } \theta \in [-\tau_{\max}, 0]$$

and where $\mathbf{x}$ belongs to the domain

$$\mathcal{D}(\mathcal{A}) := \left\{ \mathbf{x} \in \mathcal{C}([-\tau_{\max}, 0], \mathbb{R}^n) : \mathbf{x}' \text{ continuous on } [-\tau_{\max}, 0] \text{ and } \mathbf{x}'(0) = \sum_{j=0}^{m} A_j \mathbf{x}(-\tau_j) \right\}.$$

This allows to rewrite the time-delay system (1.1) in the form

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \mathcal{A}\mathbf{x}_t, \quad \text{with } \mathbf{x}_0 \in \mathcal{D}(\mathcal{A}). \tag{2.2}$$

In summary, the equation (1.1) can be reformulated over the function space $\mathcal{C}([-\tau_{\max}, 0], \mathbb{R}^n)$, both as the mapping (2.1), involving the operator $\mathcal{S}(t)$, and as the ordinary differential equation (2.2), involving the operator $\mathcal{A}$.

The spectrum of $\mathcal{A}$ solely consists of eigenvalues of finite multiplicity. These eigenvalues are the characteristic roots λ, *i.e.* the solutions of (1.2). For $\lambda \in \sigma(\mathcal{A})$, the corresponding eigenfunction is given by

$$\mathbf{x}(\theta) = v\mathrm{e}^{\lambda\theta}, \quad \text{for } \theta \in [-\tau_{\max}, 0], \tag{2.3}$$

where $v \in \mathbb{C}^n \setminus \{0\}$ satisfies

$$\Lambda(\lambda)v = 0. \tag{2.4}$$

It is important to remark that v is a finite-dimensional vector although $\mathcal{A}$ is an infinite-dimensional operator.

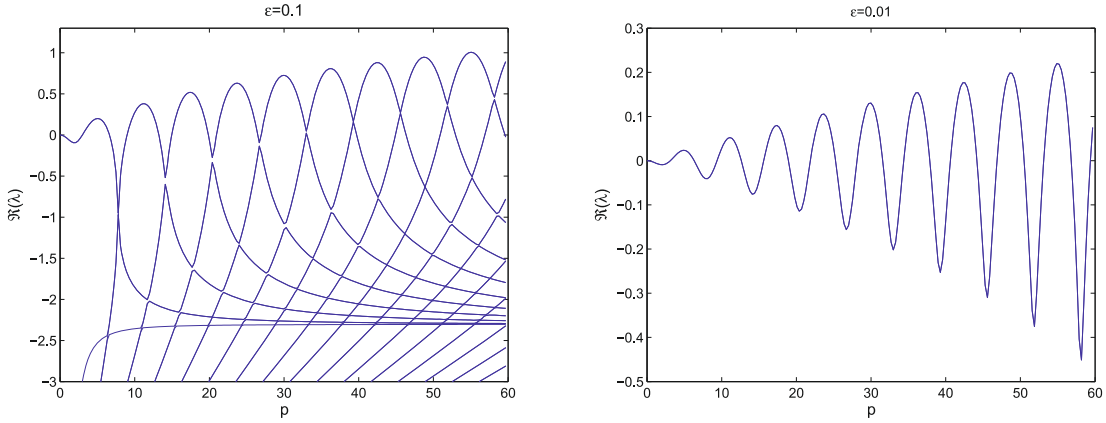

FIGURE 1. The real part of the rightmost characteristic roots of system (3.1) *versus* p for $\epsilon = 0.1$ (left) and $\epsilon = 0.01$ (right).

The spectra of $\mathcal{A}$ and $\mathcal{S}(t)$ are related by

$$\text{closure}\big(\mathrm{e}^{t\sigma(\mathcal{A})}\big) = \sigma(\mathcal{S}(t)). \tag{2.5}$$

Furthermore, if $\lambda \in \sigma(\mathcal{A})$, then $\mathrm{e}^{\lambda t}$ is an eigenvalue of $\mathcal{S}(t)$ with eigenfunction (2.3). As a consequence, the characteristic roots λ can be obtained by solving either the finite-dimensional nonlinear eigenvalue problem (2.4), or by solving the eigenvalue problem for the infinite-dimensional linear operators $\mathcal{A}$ or $\mathcal{S}(t)$, for fixed t.

3. Nonsmooth, nonconvex optimisation

In Section 3.1, we will first examine some properties of the function $\alpha(p)$, as defined in (1.4), that make it such a difficult function to optimise. Then, in Section 3.2, we will look at a particular optimisation method that is able to find local minima for nonsmooth optimisation problems, namely the gradient sampling algorithm.

3.1. The spectral abscissa function

A first important property of the spectral abscissa $\alpha(p)$ regarding its optimisation is that it is nonconvex, and therefore may have many local minima. Consider, as a constructed example, the system

$$x''(t) + p^2 x(t) - \epsilon p^2 x(t-1) = 0, \tag{3.1}$$

where p is the variable parameter and ϵ is a small, fixed value.

Figure 1 shows the real part of the rightmost characteristic roots *versus* p for $\epsilon = 0.1$ (left) and $\epsilon = 0.01$ (right). It is clear that for this example the spectral abscissa has several local minima. Moreover, by taking ϵ sufficiently small, one can even make the number of minima arbitrarily large, since the function $\alpha(p)/\epsilon$ converges uniformly on a compact interval to $-\frac{1}{2}p\sin(p)$ if ϵ tends to zero. It is clear that with such behaviour the global minimum will be very hard to find, and most optimisation algorithms will converge to a local minimum, without being able to give any guarantee about whether or not this is also the global minimum.

But even if we are satisfied with finding a local minimum, standard optimisation algorithms will still fail, because the spectral abscissa $\alpha(p)$ is also a *nonsmooth* function. Indeed, despite the fact that isolated characteristic roots behave smoothly with respect to changes of the parameters, the spectral abscissa does not. The reason for this is the presence of the maximum operator appearing in (1.4). Indeed, if for certain parameter values there is more than one *active* characteristic root, *i.e.* several characteristic roots satisfy $\Re(\lambda) = \alpha(p)$, it is easy to see that at this point the function $\alpha(p)$ is most likely not differentiable.

A special case of characteristic roots having the same real part are those which have an algebraic multiplicity larger than the geometric multiplicity. When this occurs, $\alpha(p)$ is typically non-Lipschitz even, and thus certainly nondifferentiable. This is for example the case at parameter values where a complex pair of characteristic roots is about to split up into two real characteristic roots, or *vice versa*.

As another example, consider the one-dimensional system $x'''(t) = px(t)$, which yields the spectral abscissa

$$\alpha(p) = \begin{cases} \sqrt[3]{p}, & \text{if } p \geqslant 0, \\ \sqrt[3]{-p}/2, & \text{if } p < 0. \end{cases} \tag{3.2}$$

Here, $\alpha(p)$ indeed exhibits a non-Lipschitz point at $p = 0$, corresponding to a triple characteristic root $\lambda^\star = 0$. The parameter value that yields this triple root is also the minimiser of the spectral abscissa function. This is actually not a coincidence, as it is a common observation that the solution of a fixed-order stabilisation problem will have a characteristic root with higher multiplicity (see, for instance, [8]).

For ordinary differential equation systems, the number of characteristic roots with the same real part (coinciding or not) can of course by no means be larger than the dimension of the system. Typical for the delay case is that, since the number of characteristic roots is infinite, the number of active characteristic roots and even their multiplicity *can* be larger than the system's dimension. This is for example illustrated in [16].

A problem with characteristic roots of higher multiplicity is that they are very sensitive with respect to small changes to the parameters or to the system's definition. This means that a tiny error in the system model may in practice lead to a huge increase of the spectral abscissa or may even result in the loss of stability. Thus, although (1.5) provides an optimal stability with regard to the spectral abscissa, its solution is often accompanied by a very poor robustness in terms of stability against perturbations. From a practical viewpoint, it is therefore not desirable to achieve the exact minimum of (1.4), and a rough approximation to it is mostly satisfactory. This point could then for example be used as a starting value for a robust stabilisation process, in which another objective, measuring the robustness of the system, is optimised. See [5,16] for a more elaborate explanation of this two-stage approach which is not pursued in this paper.

3.2. **The gradient sampling algorithm**

For practical nonsmooth functions, such as the spectral abscissa, nondifferentiable points will in general only occur on a set with measure zero, meaning that such functions are in fact differentiable *almost everywhere*. Indeed, for randomly selected parameter values p, the gradient of $\alpha(p)$ exists with probability one.

It is this property that is exploited by the gradient sampling algorithm, an optimisation method recently developed by Burke, Lewis and Overton. It employs a sampling technique to construct a gradient bundle, which is used in finding a local minimum of a nonsmooth objective function. Its basic steps are given in Algorithm 1.

Algorithm 1. The gradient sampling algorithm

Input: A function ϕ (continuous and differentiable almost everywhere), $\vec{\nabla}\phi$ when it exists.
Step 0: Initialise a starting value p_0 arbitrarily.
Step 1: Compute the nonsmooth steepest descent direction at p_k, using a gradient sampling technique. IF the norm of this direction is very small, THEN stop (succeed).
Step 2: Perform a line search along the direction computed in the previous step to determine a step size that has a lower function value.
Step 3: IF Step 2 succeeded, update p_k to p_{k+1} and go back to Step 1, ELSE stop (fail).

Essentially, the outline is exactly the same as the classical steepest descent method, except that a special search direction is used, namely the *nonsmooth steepest descent direction*. This direction is defined as

$$\underset{||d||\leqslant 1}{\operatorname{argmax}} \; \min_{z \in \partial_c \phi(p_k)} \langle -z, d \rangle, \tag{3.3}$$

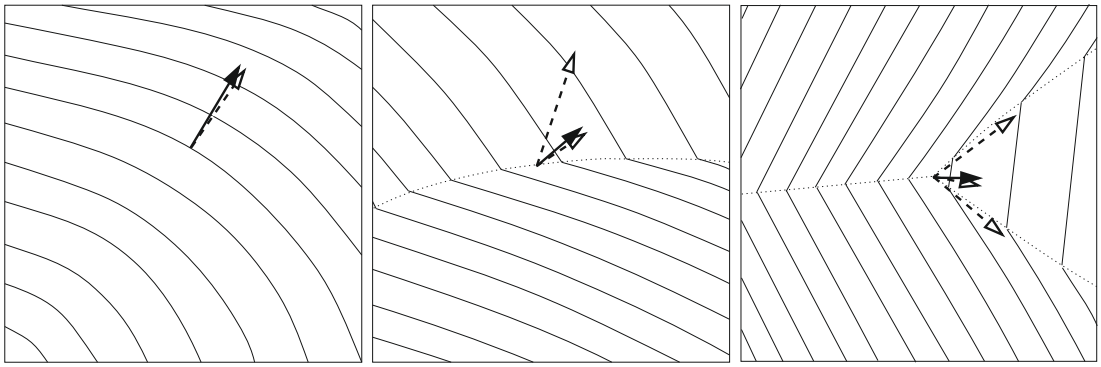

FIGURE 2. Contours of a nonsmooth function with typical configurations of meeting manifolds. The dotted line consists of nondifferentiable points. The dashed vectors are the negative gradients of the respective manifolds and a full vector shows the nonsmooth steepest descent direction.

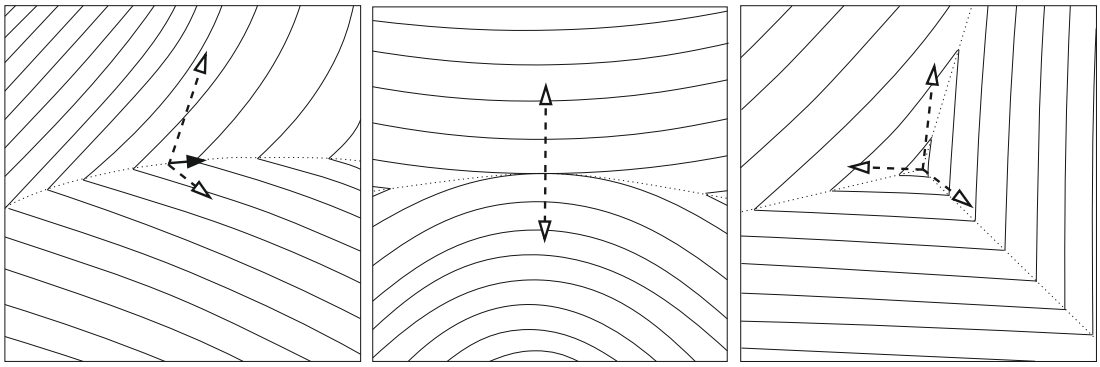

FIGURE 3. (See Fig. 2 for notational conventions.) Left, two conflicting manifolds resulting in a nonsmooth steepest direction lying along the intersection. Right, two possible configurations yielding a Clarke stationary point.

where $\langle \cdot, \cdot \rangle$ is the standard Euclidian inner product, and $\partial_c \phi(p_k)$ denotes the *Clarke subdifferential* (or generalised gradient) at p_k, given by

$$\partial_c \phi(p_k) := \operatorname{conv}\Big\{ \lim_{p \to p_k} \vec{\nabla} \phi(p) : p \in N \Big\}. \tag{3.4}$$

Here, "conv" denotes the convex hull and N is any full-measure subset of a neighbourhood around p_k containing differentiable points. The Clarke subdifferential is thus the set containing all (Clarke) subgradients and the nonsmooth steepest descent direction is opposite to the vector for which the minimum of the inner products with the subgradients is maximal. Note that (3.3) can be equivalently defined as the negative of the vector with smallest norm in the Clarke subdifferential, *i.e.*

$$- \operatorname*{argmin}_{z \in \partial_c \phi(p_k)} ||z||. \tag{3.5}$$

Let us now have a closer look at the behaviour of the nonsmooth steepest descent direction by studying several cases that can occur, in order to explain why it is so useful in the gradient sampling algorithm. In Figures 2 and 3, some typical contour lines of a nonsmooth function are depicted, depending on the configuration of the intersection of smooth manifolds. The dashed vectors represent the negative gradients of these different

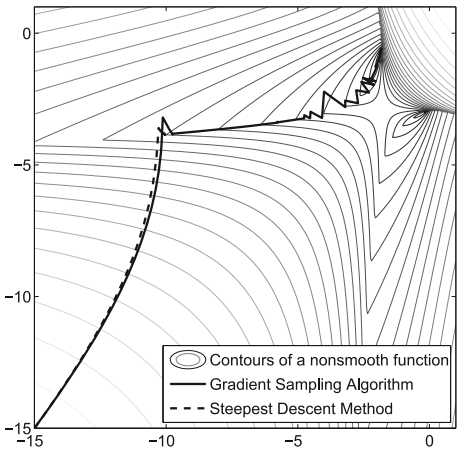

FIGURE 4. Comparison of the behaviour of the classical steepest descent method (dashed line) and the gradient sampling algorithm (full line) on a typical nonsmooth function.

manifolds in a certain nondifferentiable point, and their convex hull can be considered as the "negative" of the Clarke subdifferential. A full vector denotes the nonsmooth steepest descent direction. As can be seen in the left frame of Figure 2, the nonsmooth steepest descent direction lies exactly along the negative gradient when there is no nonsmoothness (the two vectors are drawn slightly shifted for visibility reasons). The middle and right frame show configurations where two, respectively three, manifolds meet, but without their gradients being in conflict. In both cases, the line of nonsmooth points will simply be traversed and the descent can continue on the lowest manifold without difficulty.

In the left frame of Figure 3, another situation occurs. Here, two manifolds come together forming a kind of ridge. A standard optimisation algorithm such as the steepest descent method will jump back and forth across this ridge, taking smaller and smaller steps, and eventually become jammed.

The gradient sampling algorithm, using the nonsmooth steepest descent direction, will behave quite differently. According to definitions (3.3)–(3.5), the nonsmooth steepest descent vector is in this case tangent to the ridge, as it results in an equal decrease of the two intersecting manifolds. Indeed, it is easily seen that precisely along this direction the maximal descent of the overall objective function is achieved.

In the remaining two frames of Figure 3, two possible configurations for a local nonsmooth minimum are depicted. In such a point, no direction can be found that would achieve a descent and both definitions (3.3) and (3.5) yield the zero vector accordingly. It is indeed observed in these frames that the convex hull closure of the Clarke subdifferential of a local nonsmooth minimum contains the zero vector. This is why such points are so-called *Clarke stationary points*.

Figure 4 shows the evolution path of the gradient sampling algorithm on a typical nonsmooth objective function, as compared to the behaviour of the classical steepest descent algorithm. It is seen that the latter, denoted with a dashed line, indeed strands upon reaching a ridge of nonsmooth points, whereas the full line representing the gradient sampling algorithm, is able to proceed towards the nonsmooth local minimum by following the nonsmooth steepest descent direction.

The nonsmooth steepest descent direction has to be determined in the gradient sampling algorithm (Step 1) outlined before. Because it is in practice difficult to compute this vector exactly, an approximation is constructed using the following computational procedure, whose main steps are as follows

Algorithm 2. Approximation of the nonsmooth steepest descent direction

Step 1: Sample a number of gradients in a small neighbourhood around the current point.
Step 2: Collect these gradients into a bundle, serving as an approximation for the Clarke subdifferential.
Step 3: Compute the vector with smallest norm out of this bundle by solving a small quadratic program.

Suppose the current iteration point is p_k. Next to the gradient $\vec{\nabla}\phi(p_k)$, additional gradients are computed in a number of random points sampled in a small neighbourhood around p_k. This neighbourhood is typically chosen to be the ball with centre at p_k and radius ϵ, where ϵ is some small number. If one of the sampled points should happen to be nondifferentiable, it is simply neglected and re-sampled. Along with the gradient in p_k, these sampled gradients are then regarded as a *bundle* of gradients and it was shown in [4] that the closure of the convex hull of this bundle serves as a good approximation to the Clarke subdifferential, as long as a sufficiently large amount of samples is taken. The final direction for the line search is then taken as the negative of the vector out of this convex hull that has the smallest norm. This vector can easily be computed by solving a quadratic program.

The fact that the norm of the nonsmooth steepest descent is in theory exactly zero in a nonsmooth local minimum is used as an indication to decide when to stop the gradient sampling algorithm. In particular, if the norm of the computed search direction is smaller than a threshold value, this point is assumed to be a local minimum. An important factor regarding the accuracy of this approximate minimum is the sampling radius ϵ that was used in the computation of the nonsmooth steepest descent. Because it is very hard to determine an appropriate value for this ϵ beforehand, the gradient sampling algorithm is usually conducted several times, beginning with a relatively large ϵ, and repeatedly using a smaller sampling radius, each time starting off at the endpoint of the previous run. For more practical implementation aspects of the gradient sampling algorithm, and a convergence proof, we refer the reader to [6].

In summary, the gradient sampling procedure can find local minima of a nonsmooth function, under the assumption that it is differentiable almost everywhere. The amount of work this requires is influenced by the number of iterations the algorithm needs to reach a minimum. This factor is not only problem dependent, but can even differ from run to run, because the samples are taken randomly. However, this indeterminism usually has little impact on the total time. The number of optimisation parameters n_p affects the number of samples one should take in each iteration to ensure a good approximation of the nonsmooth steepest descent direction, and thus the total number of function evaluations. Practice has proven that the double is a safe choice. The number of state variables n, *i.e.* the system dimension, is important for the time needed to compute a function evaluation. Because time-delay systems are regarded, instead of ordinary differential equations, the cost of such a function evaluation will turn out to be the dominant calculation task. How this is done exactly in our application is explained in the next section.

4. Evaluation of objective function and gradient

We now go back to the stabilisation of the time-delay system (1.1). As seen in the previous section, we can use the gradient sampling algorithm if we have a procedure at our disposal that is capable of evaluating the objective function of the minimisation problem (1.5), *i.e.* the spectral abscissa α, on the one hand, and its gradient, $\vec{\nabla}\alpha$, on the other hand. First, Section 4.1 discusses how the rightmost characteristic roots of the system (1.1) can be accurately computed, which is needed to determine the spectral abscissa α. Next, Section 4.2 presents how the computation of the gradient $\vec{\nabla}\alpha$ is performed. Section 4.3 extends these procedures to the case of a nonlinear time-delay system. In each of these sections, we also estimate the time complexity of the computational procedure.

4.1. **Computing the right-most characteristic roots of a time-delay system**

There are several numerical techniques to compute the rightmost characteristic roots. For reasons of numerical stability, working directly with the determinant in formula (1.2) is to be avoided. Instead, one usually searches for pairs (λ, v) of a characteristic root λ and corresponding eigenvector $v \in \mathbb{R}^n$ of the nonlinear eigenvalue problem (2.4). Systems of equations such as (2.4) can be solved efficiently and accurately by using Newton's method because (2.4) is finite-dimensional, albeit nonlinear. However, Newton's method only works if good initial guesses for the solutions (λ, v) are available. These can be found by solving an approximate linear

eigenvalue problem. In a next step, these approximations are corrected by using Newton's method. We now describe these two steps in more detail.

In the first step, we consider alternatives to (2.4), specifically the operator eigenvalue problems presented in Section 2. The discretisation of the infinite-dimensional operators $\mathcal{A}$ and $\mathcal{S}(t)$ yields approximating linear eigenvalue problems. Both approaches were studied in the literature: see [2,3,21] for the infinitesimal generator approach and see [1,11,23] for the solution operator approach.

In our implementation, we use the solution operator approach and approximate the eigenvalues $\mu = \mathrm{e}^{\lambda h}$ of the solution operator $\mathcal{S}(h)$ over a time step h. We follow [11,23] and discretise this operator into a matrix using a k-step *linear multistep* method combined with Lagrange interpolation to evaluate the delayed terms. For a more detailed description of our implementation, see [23]. The latter reference also explains the use of special-purpose linear multistep methods that aim at increasing the efficiency while maintaining the accuracy.

In the second step, Newton's method is used to correct the computed characteristic roots up to a desired accuracy. Hence the final accuracy is *not* affected by the discretisation error in the first step. The discretisation method should only return good starting values for Newton's method. Whether a starting value is good enough to allow for Newton's method to converge is in general problem-dependent. The above procedure has proved to be efficient and accurate for real-life problems, as will be shown in Section 5.

By its definition, the spectral abscissa can easily be obtained by letting the algorithm select the characteristic root with largest real part. If there is a complex conjugate pair of rightmost characteristic roots, then the one with positive imaginary part is selected. In case there is a tie between several (pairs of) characteristic roots, we allow the algorithm to arbitrarily pick one, since the optimisation method, using its sampling strategy, does not rely on exactly which characteristic root is chosen.

The time complexity of this procedure is dominated by the first step, *i.e.* the solution of an approximate linear eigenvalue problem. For this purpose, we employ the robust *QR method*, whose cost grows cubic in the size of the matrix. The matrix size is proportional to n, the number of state variables, and inversely proportional to the steplength h used in the linear multistep method. Hence, the time complexity of the first step of the procedure is proportional to n^3/h^3. In [10], a heuristic choice of the steplength h is presented which aims at a large h — and thus a low computational cost — for which the approximations to the rightmost characteristic roots are still excellent. This steplength heuristic is improved in [23] for a class of special-purpose linear multistep methods. The time complexity of the second step is of order n^3, which is negligible w.r.t. the cost of n^3/h^3 of the first step.

4.2. **Evaluation of the gradient**

A sufficient condition for differentiability of the spectral abscissa is that there is only one active characteristic root (or a complex conjugate pair of roots), or, in other words, that the rightmost characteristic root (or pair) is isolated. For an arbitrary matrix, this is the case almost always.

The gradient $\vec{\nabla}\alpha$ can be computed analytically. Indeed, let u and v be the respective left and right null vector of the matrix $\Lambda(\lambda^\star; p)$, with $\lambda^\star$ an isolated characteristic root. Denote by the superscript "H" the complex conjugate transpose. Differentiating the equation $\Lambda(\lambda^\star; p)v = 0$ results, after some calculus, in the following formula for the gradient of α:

$$\vec{\nabla}\alpha(p) = \Re\left(\frac{\mathrm{d}\lambda^\star}{\mathrm{d}p}\right) = \Re\left(\frac{u^H\left(\frac{\partial A_0}{\partial p} + \sum_{j=1}^m \mathrm{e}^{-\lambda^\star\tau_j}\frac{\partial A_j}{\partial p}\right)v}{u^H\left(I + \sum_{j=1}^m \tau_j \mathrm{e}^{-\lambda^\star\tau_j}A_j\right)v}\right). \tag{4.1}$$

However, tests indicate that in general this formula is not always the most robust choice.

Alternatively, the gradient $\vec{\nabla}\alpha$ can be approximated by a finite difference formula, as follows. The number of parameters in system (1.1), n_p, equals the dimension of the tangent plane at $\alpha(p)$. Thus, this plane can be constructed — or, equivalently, the gradient $\vec{\nabla}\alpha(p)$ — by using values $\alpha(p^i)$ at $n_p + 1$ points p^i close to p. In our implementation, the points p^i are selected as the n_p+1 vertices of a regular polytope in the space of n_p free

parameters with centre of mass p and radius ε. For our experiments we used $\varepsilon = 10^{-7}$. Thus, in principle, one solves $n_p + 1$ independent eigenvalue problems. However, after computing the active characteristic root $\lambda^\star$ at the centre of the polytope p, the active characteristic roots at the vertices p^i can efficiently be computed using Newton's method with starting value $\lambda^\star$. Therefore, the additional cost of order n^3 in computing the gradient is negligible w.r.t. the cost of order n^3/h^3 in solving the approximate linear eigenvalue problem of Section 4.1.

4.3. **Extension to nonlinear time-delay systems**

Up to now, we have considered a linear delay differential system (1.1) and optimised the maximal real part of the characteristic roots (of its zero steady state solution) by varying the system parameters. There are two important differences when considering a nonlinear delay differential system of the form

$$x'(t) = f\left(x(t), x(t-\tau_1), \ldots, x(t-\tau_m); p\right), \tag{4.2}$$

where $f(\cdot)$ is continuously differentiable.

First, the constant $x(t) \equiv 0$ is in general not a solution of (4.2). Moreover, the existence and uniqueness of steady states for fixed system parameters no longer hold. *E.g.*, if a steady state solution $x^\star \equiv x(t)$ exists for certain system parameters, then a nearby point in the parameter space can have a steady state "close" to $x^\star$ or there can be no steady state at all. Hence, the spectral abscissa α not only depends on the parameters, but also on the considered steady state $x^\star$. This implies that for fixed parameter values, the spectral abscissa is not uniquely defined. The investigation of *branches* of solutions of a nonlinear system for varying parameters is called *bifurcation analysis*, see *e.g.* [20].

The second difference is the meaning of the objective function that is optimised. Consider the linearisation of (4.2) about a particular steady state. One can prove that the characteristic roots of this linearisation determine the *local stability* of the original nonlinear system in a neighbourhood of this steady state. Hence by optimising the spectral abscissa of the linearisation, we optimise the reaction of the system to small perturbations.

The consequences on the computational procedure are the following. Most importantly, the value of the steady state must be corrected after each change in the system's parameters. For this purpose, we employ Newton's method using the previous solution as initial guess. If for the current point the value of α is negative, then the optimisation algorithm will not leave the current branch. Indeed, because the values of α decrease monotonically along the optimisation path, the procedure will never cross a bifurcation point, where at least one characteristic root resides on the imaginary axis and the value of α is either zero or positive. If no stabilising parameters are found on the initial branch, additional branches that exist for the same parameter range should therefore be independently explored.

A second consequence is that since α is also dependent on the steady state $x^\star$, we have to adapt formula (4.1) by including a term for the partial derivative w.r.t. $x^\star$. When computing the gradient by using finite differences, the steady state $x^\star$ must be corrected by Newton's method for all vertices p^i of the polytope used in the finite difference formula.

Remark that the influence on the computational cost is again of order n^3 and thus marginal compared to the dominant cost of solving the approximate linear eigenvalue problem of Section 4.1.

5. Numerical examples

Our MATLAB algorithm uses the implementation of the bundle gradient optimisation algorithm which is detailed in [6] and whose main steps were given in Section 3.2. To compute the objective function and its gradient, we use the procedure presented in Section 4.

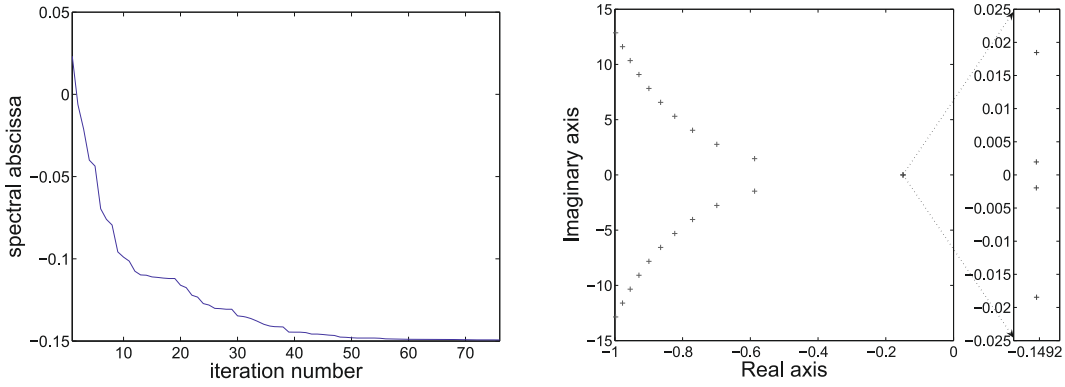

FIGURE 5. Evolution of the spectral abscissa during the optimisation process (left) and resulting spectrum of the stabilised system (5.1) (right).

5.1. **A third-order feedback controller system**

Consider the system

$$x'(t) = Ax(t) + B(p)x(t-\tau) \tag{5.1a}$$

with time delay $\tau = 5$ and

$$A = \begin{bmatrix} -0.08 & -0.03 & 0.2 \\ 0.2 & -0.04 & -0.005 \\ -0.06 & -0.2 & -0.07 \end{bmatrix}, \; B = \begin{bmatrix} -0.1 \\ -0.2 \\ 0.1 \end{bmatrix} \begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix}. \tag{5.1b}$$

For $p = 0$, the spectral abscissa is positive, rendering the open-loop system unstable. This problem stems from a feedback controller problem and its stabilisation by means of the continuous pole placement method served as an illustrative example in [16]. The exact minimiser of $\alpha(p)$ can be computed analytically once it is observed that the minimum is characterised by four coinciding active characteristic roots $\lambda_{1..4} = -0.15$. This illustrates the fact that spectral abscissa functions coming from a time-delay system can have minima with a characteristic root with a multiplicity larger than the system's dimension. The optimal parameter values are given by $p = \begin{bmatrix} 0.471 & 0.504 & 0.602 \end{bmatrix}$.

Applying the bundle gradient algorithm yields (nearly) identical numerical values for the optimal parameters, $p = \begin{bmatrix} 0.472 & 0.505 & 0.603 \end{bmatrix}$. The monotonic decrease of the spectral abscissa with the iteration number of the optimisation algorithm can be seen on the left in Figure 5. On the right, the spectrum of characteristic roots at the final iteration point is shown, and we indeed observe the expected quadruple characteristic root at -0.15. The zoomed-in section of this plot reveals that numerically four distinct active characteristic roots are found, due to the ill condition of computing coinciding characteristic roots.

5.2. **Heat transfer model**

In [24], a water-based heating system is introduced. The equations that model this heat transfer process read as follows:

$$T_h \dot{x}_h(t) = -x_h(t-\eta_h) + K_b x_a(t-\tau_b) + K_u x_{h,set}(t-\tau_u), \tag{5.2}$$

$$T_a \dot{x}_a(t) = -x_a(t) + x_c(t-\tau_e) + K_a \left(x_h(t) - \tfrac{1+q}{2} x_a(t) - \tfrac{1-q}{2} x_c(t-\tau_e)\right), \tag{5.3}$$

$$T_d \dot{x}_d(t) = -x_d(t) + K_d x_a(t-\tau_d), \tag{5.4}$$

$$T_c \dot{x}_c(t) = -x_c(t-\eta_c) + K_c x_d(t-\tau_c), \tag{5.5}$$

$$\dot{x}_e(t) = -x_c(t) + x_{c,set}(t), \tag{5.6}$$

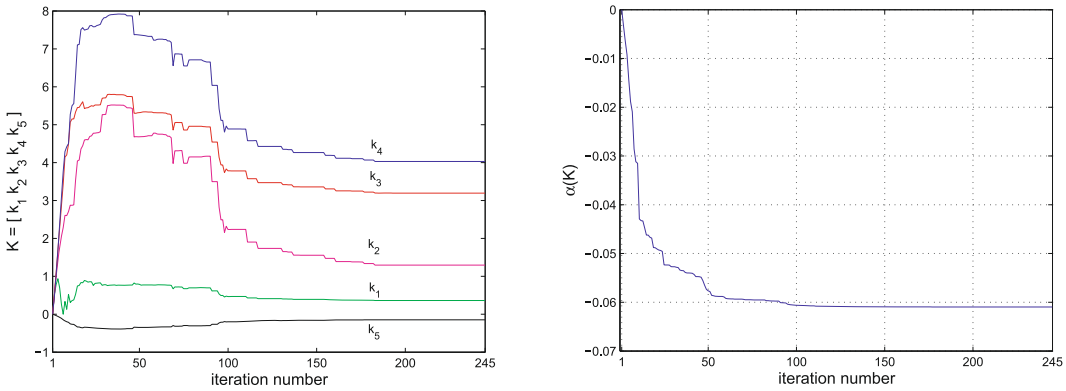

FIGURE 6. Evolution of the controller parameters of the heat transfer model on the right, and the spectral abscissa in function of the algorithms iteration number.

TABLE 1. Parameter values for the water-based heating system (5.2)–(5.6).

$T_h = 14$	$K_b = 0.24$	$\tau_b = 40$	$\eta_h = 6.5$
$T_a = 3$	$K_a = 1$	$\tau_e = 13$	$q = 1$
$T_d = 3$	$K_d = 0.94$	$\tau_d = 18$	
$T_c = 25$	$K_c = 0.81$	$\tau_c = 2.8$	$\eta_c = 9.2$
	$K_u = 0.39$	$\tau_u = 13.2$	

where the x-variables denote the temperature measured at different places in the circuit, and the parameter values are listed in Table 1. The set-point value $x_{h,set}(t)$ is determined by the static state feedback controller

$$x_{h,set}(t) = K \begin{bmatrix} x_h(t) & x_a(t) & x_d(t) & x_c(t) & x_e(t) \end{bmatrix}^T.$$

The (uncontrolled) open loop system has an eigenvalue at zero and is consequently not stable. Using the continuous pole placement method, stabilisation was established by *Vyhlidal* up to a spectral abscissa of $\alpha(K) = -0.0413$.

After applying the stabilisation method proposed in this paper, a spectrum was reached with a rightmost eigenvalue with real part equal to -0.06977, at gain values $K = \begin{bmatrix} 0.364 & 1.30 & 3.20 & 4.03 & -1.46 \end{bmatrix}$. In Figure 6, the variation of the controller parameters during the nonsmooth optimisation process is shown, and the corresponding evolution of the spectral abscissa is plotted next to it. It is seen that the decrease of $\alpha(K)$ towards its minimum again happens quite fast. When only an approximate value is required, for example as a starting point for robust stabilisation, a hundred iterations already give a satisfactory result.

The system's spectrum for these optimal gain values is plotted in Figure 7, along with the spectrum of the initial, uncontrolled system.

5.3. **A semiconductor laser model**

We investigate the stabilisation of a system consisting of a delay differential equation coupled to a partial differential equation. The system models a semiconductor laser subject to conventional optical feedback and lateral carrier diffusion [22]. The system in the complex scalar variable $E(t)$, representing the electric field, and real variable $N(x,t)$, representing the carrier density in the interval $x \in [-0.5, 0.5]$, reads as

$$E'(t) = (1 - i\beta)E(t)\zeta(t) + \eta E(t-\tau)\mathrm{e}^{-i\phi} - ibE(t), \tag{5.7}$$

$$T\frac{\partial N(x,t)}{\partial t} = d\frac{\partial^2 N(x,t)}{\partial x^2} - N(x,t) + P(x) - F(x)(1 + 2N(x,t))|E(t)|^2, \tag{5.8}$$

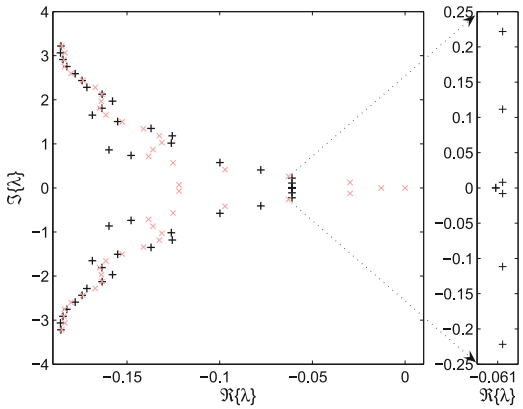

FIGURE 7. Spectrum of the heat transfer model before (×) and after (+) stabilisation.

where "i" is the imaginary unit and $\zeta(t)$ is a weighted average of the carrier density $N(x,t)$, specifically,

$$\zeta(t) = \frac{\int_{-0.5}^{0.5} F(x)N(x,t)\,\mathrm{d}x}{\int_{-\infty}^{\infty} F(x)\,\mathrm{d}x}. \tag{5.9}$$

The functions $P(x)$ and $F(x)$ are specified in [22]. We split (5.7) into real and imaginary parts in order to work only with real numbers. The symmetry of the spatial domain about $x = 0$ is exploited by considering only the interval [0, 0.5] and imposing zero Neumann boundary conditions at $x = 0$, *i.e.*, $\partial N(0,t)/\partial x = 0$. We also use zero Neumann boundary conditions at $x = 0.5$.

The partial differential system (5.7)–(5.8) cannot be treated directly by our method. It is first transformed to a system of differential equations of the form (4.2). For this purpose, we use a standard finite difference discretisation in space, in particular a second order central difference formula with constant stepsize $\Delta x = 0.5/128$. Moreover, (5.9) is approximated using the trapezoidal quadrature rule. This transformation gives a time-delay system of size 131 in the unknowns $\Re(E(t))$, $\Im(E(t))$ and $N(x_j,t)$ for $x_j := j\Delta x \in [0, 0.5]$, with $j = 0, \ldots, 128$. Note that this transformation introduces a discretisation error. However, since the characteristic roots determining the linear stability after space discretisation are computed up to a desired accuracy (*cf.* Sect. 4), the computed right-most characteristic roots correspond to the original system (5.7)–(5.8) if the error of the *spatial* discretisation is sufficiently small. That is, if the system's properties are captured sufficiently well by the spatial discretisation. The quality of the characteristic roots is confirmed by our experiments using different numbers of mesh intervals in space.

Due to the form of system (5.7)–(5.8), every solution (E, N) belongs to a family of solutions of the form (cE, N) where c is an arbitrary complex number on the unit circle, *i.e.* $|c| = 1$. It is said that the system is *rotationally symmetric* in E. The tangent at this continuum of solutions (cE, N) is the eigenvector corresponding to a characteristic root at zero. This characteristic root at zero would not occur if one particular solution is selected of the continuum of solutions by imposing a so-called pinning condition or phase condition. It can therefore be safely ignored. For this reason, our algorithm removes the computed characteristic root that is zero, up to rounding error, before the active (pair of) characteristic root(s) is selected.

We fix the parameters to the values $\beta = 3$, $\phi = 0$, $T = 1000$ and delay $\tau = 1000$; and we optimise w.r.t. parameters η and d, representing the feedback strength and diffusion coefficient respectively. To increase the numerical stability of the computations, the time variable and the time delay are rescaled by a factor of $1/1000$ and the parameters η and d are rescaled by a factor of 1000. Recall that a nonlinear system can have multiple steady state solutions for a given set of parameters, *cf.* Section 4.3. In this case, it can be shown that for

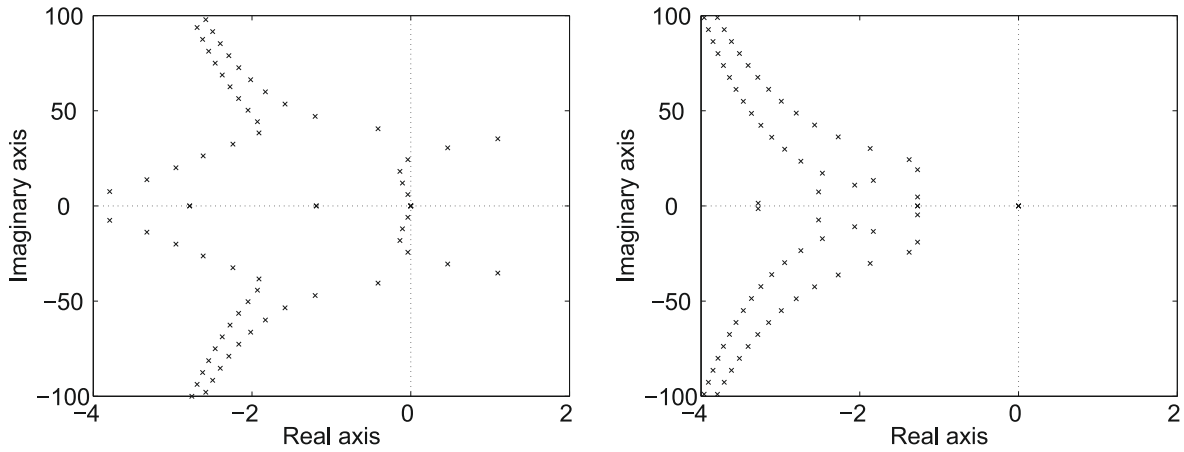

FIGURE 8. The characteristic roots λ of the starting point of the optimisation (left) and of a point where the spectral abscissa is locally minimal (right). The characteristic root at zero is induced by the rotational symmetry of the time-delay system.

η between 0 and 7×10^{-3} and arbitrary d there are four branches that contain *stable* steady state solutions. This is illustrated on [22], Figure 3.

The starting point of the optimisation algorithm is an unstable steady state with $\eta = 6.4051 \times 10^{-3}$ and $d = 1.68 \times 10^{-2}$. This initial steady state lies on one of the branches mentioned above. The algorithm returned a local minimum $\alpha^\star = -1.2713$ on this branch at $\eta^\star \approx 1.9926 \times 10^{-3}$ and $d^\star \approx 2.3203 \times 10^{-2}$. Figure 8 shows the rightmost characteristic roots for the starting point of the optimisation (left) and for the computed local minimum (right).

Figure 9 gives a view on the computed local minimum and some contours of the spectral abscissa α (spaced 0.02 apart) in the parameter space η-d. It also shows the path followed by the optimisation algorithm (dashed line and tick markers) and the points of nondifferentiability (dotted lines). In this figure one can clearly see how the optimisation algorithm tends to follow these lines of nonsmoothness to converge to the local minimum.

6. Conclusions

The effectiveness of eigenvalue optimisation algorithms, in particular the gradient sampling optimisation algorithm of [6], in solving fixed-order stabilisation problems for infinite-dimensional time-delay systems was shown. The stabilisation approach we used was based on a direct optimisation of the spectral abscissa function. This was possible by combining the optimisation algorithm with efficient state-of-the-art numerical algorithms for the computation of the stability-determining characteristic roots of a time-delay system. Here, note that, due to the Newton corrections involved in the computation of the characteristic roots in the procedure of Section 4.2, the effect of a time discretisation is completely cancelled out. Hence, the optimisation algorithm works with the exact value of the spectral abscissa of the original delay equation, and not of a discretised version.

Our implementation was tested on the stabilisation of some practical examples and the results were presented. To the best of the authors knowledge, this is the first time that an eigenvalue optimisation approach was applied to infinite-dimensional systems.

An alternative method for the stabilisation of time-delay systems is created in this way, with the following advantages w.r.t. other existing approaches. First of all, the optimisation approach results in a nonconservative stabilisation procedure, in the sense that stabilising parameters can in principle always be found as long as they exist. Most existing methods provide solutions for the stabilisation problem whose construction is based on the feasibility of certain conditions (see, for instance, [12]). Conservatism is then introduced because these

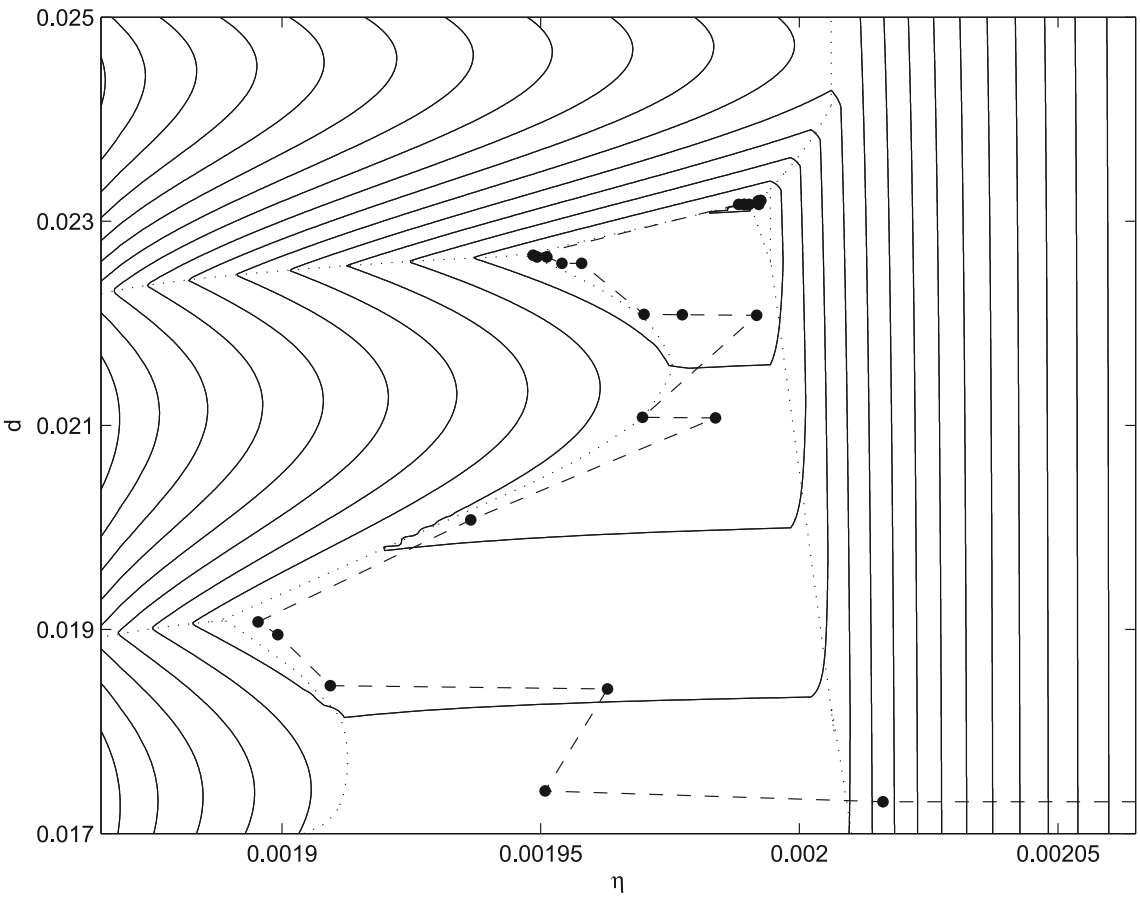

FIGURE 9. The contour lines of the spectral abscissa α w.r.t. parameters η and d about the optimum $\approx (1.9926 \times 10^{-3}, 2.3203 \times 10^{-2})$, the path followed by the optimisation algorithm (dashed line and tick markers) and the points of nondifferentiability (dotted lines). (See the text for further explanation.)

conditions are sufficient, but often not necessary, that is, the system may be stabilisible although the conditions cannot be satisfied.

Another advantage is that additional non-physical characteristic roots, which may be introduced by certain model transformations, can be dealt with in an elegant way. Indeed, the non-physical roots are typically known beforehand, and can be explicitly ignored by the optimisation algorithm. Such transformations are for instance necessary when a system with distributed delays needs to be converted into a system with constant delays [15]. The same thing happens when the system exhibits a rotational symmetry. This is illustrated in the semiconductor laser example, where a non-physical zero characteristic root, due to such a rotational symmetry, is known *a priori* and can be easily ignored explicitly by the optimisation procedure. Note that existing criteria for asymptotic stability cannot be used for such a problem, as the non-physical unstable characteristic root would, mathematically speaking, render the system unstable for all values of the controller parameters.

Furthermore, no interaction from the user is needed, *i.e.* the procedure is fully automatic. This is unlike *e.g.* the eigenvalue based continuous pole placement method [17], where the evolution of the characteristic roots needs to be monitored.

Finally, the strategy is also very general in the sense that there is no limitation on the number of time delays nor on the type of delays. The extension to periodically varying delays and *distributed* (or so-called continuous)

delays is fairly straightforward, since the method only requires that a procedure to compute the rightmost characteristic roots be at hand.

Acknowledgements. We greatly acknowledge J.V. Burke, A.S. Lewis and M.L. Overton for making the MATLAB implementation of the gradient sampling algorithm described in [6] publicly available. This research presents results of the Project IUAP P5/22 funded by the Interuniversity Attraction Poles Programme - Belgian Science Policy and by the Centre of Excellence on Optimisation in Engineering of the K.U. Leuven. The scientific responsibility rests with the authors. K.V. is a Research Assistant and W.M. is a Postdoctoral Fellow of the Fund for Scientific Research – Flanders (Belgium).

References

[1] D. Breda, Solution operator approximation for delay differential equation characteristic roots computation via Runge-Kutta methods. *Appl. Numer. Math.* **56** (2005) 318–331.
[2] D. Breda, S. Maset and R. Vermiglio, Computing the characteristic roots for delay differential equations. *IMA J. Numer. Anal.* **24** (2004) 1–19.
[3] D. Breda, S. Maset and R. Vermiglio, Pseudospectral differencing methods for characteristic roots of delay differential equations. *SIAM J. Sci. Comput.* **27** (2005) 482–495.
[4] J. Burke, A. Lewis and M. Overton, Approximating subdifferentials by random sampling of gradients. *Math. Oper. Res.* **22** (2002) 567–584.
[5] J. Burke, A. Lewis and M. Overton, A nonsmooth, nonconvex optimization approach to robust stabilization by static output feedback and low-order controllers, in *Proceedings of ROCOND 2003*, Milan, Italy (2003).
[6] J. Burke, A. Lewis and M. Overton, A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM J. Opt.* **24** (2005) 567–584.
[7] J. Burke, D. Henrion, A. Lewis and M. Overton, HIFOO - A MATLAB Package for Fixed-Order Controller Design and H-infinity optimization, in *Proceedings of ROCOND 2006*, Toulouse, France (2006).
[8] J. Burke, D. Henrion, A. Lewis and M. Overton, Stabilization *via* nonsmooth, nonconvex optimization. *IEEE Trans. Automat. Control* **51** (2006) 1760–1769.
[9] O. Diekmann, S. van Gils, S.V. Lunel and H.-O. Walther, Delay Equations. *Appl. Math. Sci.* **110**, Springer-Verlag (1995).
[10] K. Engelborghs and D. Roose, On stability of LMS methods and characteristic roots of delay differential equations. *SIAM J. Numer. Anal.* **40** (2002) 629–650.
[11] K. Engelborghs, T. Luzyanina and D. Roose, Numerical bifurcation analysis of delay differential equations using DDE-BIFTOOL. *ACM Trans. Math. Softw.* **28** (2002) 1–21.
[12] K. Gu, V. Kharitonov and J. Chen, *Stability of time-delay systems*. Birkhauser (2003).
[13] J. Hale and S.V. Lunel, *Introduction to Functional Differential Equations*, *Applied Mathematical Sciences* **99**. Springer-Verlag, (1993).
[14] V. Kolmanovskii and A. Myshkis, *Introduction to the theory and application of functional differential equations, Math. Appl.* **463**. Kluwer Academic Publishers (1999).
[15] T. Luzyanina and D. Roose, Equations with distributed delays: bifurcation analysis using computational tools for discrete delay equations. *Funct. Differ. Equ.* **11** (2004) 87–92.
[16] W. Michiels and D. Roose, An eigenvalue based approach for the robust stabilization of linear time-delay systems. *Int. J. Control* **76** (2003) 678–686.
[17] W. Michiels, K. Engelborghs, P. Vansevenant and D. Roose, Continuous pole placement for delay equations. *Automatica* **38** (2002) 747–761.
[18] S.-I. Niculescu, *Delay effects on stability: A robust control approach*, *LNCIS* **269**. Springer-Heidelberg (2001).
[19] J.-P. Richard, Time-delay systems: an overview of some recent and open problems. *Automatica* **39** (2003) 1667–1694.
[20] R. Seydel, *Practical Bifurcation and Stability Analysis: From Equilibrium to Chaos, Interdisciplinary Applied Mathematics* **5**. Springer-Verlag, 2nd edn. (1994).
[21] K. Verheyden and D. Roose, Efficient numerical stability analysis of delay equations: a spectral method, in *Proceedings of the IFAC Workshop on Time-Delay Systems 2004* (2004) 209–214.
[22] K. Verheyden, K. Green and D. Roose, Numerical stability analysis of a large-scale delay system modelling a lateral semiconductor laser subject to optical feedback. *Phys. Rev. E* **69** (2004) 036702.
[23] K. Verheyden, T. Luzyanina and D. Roose, Efficient computation of characteristic roots of delay differential equations using LMS methods. *J. Comput. Appl. Math.* (in press). Available online 5 March 2007.
[24] T. Vyhlídal, *Analysis and synthesis of time delay system spectrum*. Ph.D. thesis, Department of Mechanical Engineering, Czech Technical University, Czech Republic (2003).