A globally convergent method to compute the real stability radius for time-delay systems

Francesco Borgioli, Wim Michels, Ding Lu, Bart Vandereycken

Abstract

This paper presents a novel algorithm to compute the real stability radius for a linear delay system of retarded type with multiple delays. The real stability radius is the distance to instability, measured as the minimal real-valued perturbation that renders the system unstable. Our method is based on characterizing this distance to instability as the inverse of the global maximum of a real structured singular value function. We develop a criss-cross type algorithm that *globally* converges to this maximum, and whose convergence rate seems to be superlinear and sometimes quadratic in numerical experiments. The algorithm exploits that the intersections of these singular value functions with constant functions can be written as purely imaginary eigenvalues of certain delay eigenvalue problems (DEP) with positive and negative delays. This is an extension of the well-known linear case (without delays) where this results in algebraic eigenvalue problems. In addition, a novel numerical solver to compute all the imaginary eigenvalues of this DEP is also presented. It combines an approximation using spectral discretizations and an automatic procedure to determine the required number of discretization points. Finally, due to the presence of multiple eigenvalues at the maximum, these approximations are corrected with a block-Newton algorithm for nonlinear eigenvalue problems.

Keywords: Computational methods, Time delay equations, Stability radius, Robust control, Real structured perturbations

1. Introduction

The robustness of stability is a topic of major interest in many engineering applications. When computing the asymptotic stability of a linear dynamical system $\dot{x} = Ax$, it is helpful to also assess whether the system remains stable after perturbations in the matrix A, for instance, when there are uncertainties in A. Adopting a frequency domain approach, this requires evaluating whether such a perturbation is large enough to push the spectral abscissa (namely the real part of the rightmost eigenvalue of the spectrum) into the closed right-half of the complex plane. The stability radius, or equivalently, the distance to instability, is defined as the minimal perturbation (in terms of magnitude) that renders the system unstable. It is called complex or real stability radius depending on whether the perturbations are complex or real. Furthermore, we talk about the *structured* stability radius whenever a specific structure is imposed on the perturbation. For example, it can be that only a submatrix of A is perturbed or one of its coefficients.

In this paper we consider linear time-invariant systems of retarded type with multiple delays, where the nominal matrices are affected by real-valued structured uncertainties as follows:

$$\dot{x}(t) = \sum_{i=1}^{m} (A_i + B\delta A_i C_i)\, x(t-\tau_i). \tag{1}$$

Here, $x(t) \in \mathbb{R}^n$ is the state variable, $A_i \in \mathbb{R}^{n\times n}$ are the nominal matrices, $0 \le \tau_1 < \cdots < \tau_m$ are the time delays, and $B \in \mathbb{R}^{n\times p}$, $\delta A_i \in \mathbb{R}^{p\times q_i}$ and $C_i \in \mathbb{R}^{q_i\times n}$ represent the uncertainties for $i = 1, \ldots, m$. We assume that the unperturbed system is exponentially stable. Observe that the uncertainties δA_i may vary from one matrix A_i to another but the matrix B is the same.[1] The main problem considered in this paper is computing the *real, structured* distance to instability for (1).

Different approaches are presented in the literature to compute the stability radius for a class of linear delay systems affected by such uncertainties, and we refer to the monographs [6, 16, 18]. The Lyapunov-based approach allows for a broad class of uncertainties, such as time-varying perturbations ([13, 5]) and perturbations on delay values ([14]). However, these methods are based on sufficient conditions for stability, which typically leads to conservative lower bounds on the stability radius.

Another set of widely used method is based on the structured singular value. They reformulate the delay eigenvalue problem associated with system (1) as $\det(I - M(\lambda)\Delta) = 0$, where Δ represents the uncertainties affecting the system. Depending on the structure of Δ, the distance to instability can be computed in different ways: if Δ is a real-valued block diagonal matrix, then it is possible to use geometric optimization techniques as in [1]; if Δ is the concatenation of uncertainties δA_i, then the distance to instability is computed as the inverse of the maximum of the $\mu_{\mathbb{R}}$ function (see Section 2 for details). Formulas for the computation of the structured singular value function for complex-valued or real-valued perturbations are provided in [8] and [19]. Regarding delay systems, in [15] the notion of complex stability radius is introduced, while in [17] the sta-

[1] The results in this paper can be easily modified to treat the case of different B_i but common C.

bility radius is computed for a single-delay system. In [9], the real structured stability radius is computed for either retarded or neutral delay systems, where the same hypothesis on matrix B, as in (1), is also adopted.

Closely related to this paper, the method described in [9] is based on computing the maximum of the real structured singular value function $\mu_{\mathbb{R}}$ by sampling the interval $[0, \omega^*]$ on a grid, where ω^* is a user-supplied upper bound on the frequencies ω such that $\lambda = \jmath\omega$ solves the equation $\det(I - M(\lambda)\Delta) = 0$. One of the downsides of this method is that the interval and the grid size are hard to determine in an automated way: when the $\mu_{\mathbb{R}}$ function is very steep around its global maximum, a very fine grid is needed for accurate approximations which will require many evaluations of the $\mu_{\mathbb{R}}$ function using the formula in [19]. To improve this estimation, and with the purpose of speeding up the computation, in this paper we extend to delay-systems the criss-cross algorithm introduced in [12, 20] for the real stability radius of a delay-free system, which is itself an extension of the method [2] for the complex stability radius. As we will show in detail in Section 2, our method exploits the existence of a set of functions $g_2(\jmath\omega, \gamma)$, $\gamma \in (0, 1]$, whose point-wise infimum is the desired $\mu_{\mathbb{R}}$ function. The convergence to the global maximum is guaranteed, using only a few evaluations of the $\mu_{\mathbb{R}}$ function per iteration. This procedure exploits the characterization of the intersections between $g_2(\jmath\omega, \gamma)$ and constant functions as purely imaginary eigenvalues of a delay eigenvalue problem (DEP) with positive and negative delay values, for which a novel solver is also proposed.

The paper is structured as follows: Section 2 is dedicated to the explanation of the theoretical basis and the implementation of the criss-cross algorithm; in Section 3 we present the new DEP solver; in Section 4 we show the effectiveness of the method by means of some numerical simulations, and finally in Section 5 we give our conclusions.

2. Real stability radius: a criss-cross algorithm

Associated to (1), consider the DEP

$$M(\lambda; \Delta)y := \left(\lambda I_n - \sum_{i=1}^{m} A_i e^{-\tau_i\lambda} - \sum_{i=1}^{m} B\delta A_i C_i e^{-\tau_i\lambda}\right)y = 0, \quad (2)$$

where the perturbations δA_i are collected in the matrix

$$\Delta := [\delta A_1, \ldots, \delta A_m] \in \mathbb{R}^{p\times q} \quad \text{with } q = \sum_{i=1}^{m} q_i. \quad (3)$$

We call $(\lambda, y) \in \mathbb{C} \times \mathbb{C}^n$ with nonzero y an eigenpair of the DEP if it satisfies (2). The system is stable if all its eigenvalues have negative real part.

We are interested in computing the stability radius of system (1) for perturbations Δ measured in the spectral norm $\|\cdot\|_2$. This can be conveniently written as the minimum perturbation such that some purely imaginary eigenvalue $\lambda = \jmath\omega$ is a root of DEP (2) with $\omega \in \mathbb{R}^+$:

$$\begin{aligned} r_{\mathbb{R}} &:= \inf_{\substack{\lambda=\jmath\omega \\ \omega\geq 0}} \inf_{\Delta\in\mathbb{R}^{p\times q}} \{\|\Delta\|_2 : \det(M(\lambda;\Delta)) = 0\} \\ &= \inf_{\substack{\lambda=\jmath\omega \\ \omega\geq 0}} \inf_{\Delta\in\mathbb{R}^{p\times q}} \{\|\Delta\|_2 : \det(I - G(\lambda)\Delta) = 0\}, \end{aligned} \quad (4)$$

where

$$G(\lambda) = \mathbf{C}(\lambda)D(\lambda)^{-1}B \in \mathbb{C}^{q\times p}, \quad (5)$$

with

$$D(\lambda) = \lambda I - \sum_{i=1}^{m} A_i e^{-\tau_i\lambda} \qquad \text{and} \qquad \mathbf{C}(\lambda) = \begin{bmatrix} C_1 e^{-\tau_1\lambda} \\ \vdots \\ C_m e^{-\tau_m\lambda} \end{bmatrix}.$$

2.1. The $\mu_{\mathbb{R}}$ function

Let us now define the singular value functions

$$g_\ell(\lambda, \gamma) = \sigma_\ell\left(\begin{bmatrix} \Re G(\lambda) & -\gamma\Im G(\lambda) \\ \gamma^{-1}\Im G(\lambda) & \Re G(\lambda) \end{bmatrix}\right), \ \lambda = \jmath\omega, \ \omega \in \mathbb{R}^+, \quad (6)$$

for $\gamma \in (0, 1]$ and $\ell = 1, \ldots, N^*$ with $N^* = \min(q, p)$. We assume throughout that $\operatorname{rank}(\Im G(\jmath\omega)) \geq 2$ for all $\omega \in \mathbb{R}^+$ and treat the case $\operatorname{rank}(\Im G(\jmath\omega)) = 1$ later in Section 2.3. Thanks to the manipulation in (4–5) and the rank condition, we can apply the results from [19] to obtain

$$r_{\mathbb{R}}^{-1} = \sup_{\omega\geq 0} \mu_{\mathbb{R}}(G(\jmath\omega)) \quad \text{with } \mu_{\mathbb{R}}(G(z)) = \min_{\gamma\in(0,1]} g_2(z, \gamma), \quad (7)$$

where $g_2(\jmath\omega, \gamma)$ is a unimodal function in $\gamma \in (0, 1]$. We define

$$\gamma_{\min}(\omega) \in (0, 1] \quad \text{s.t.} \quad \mu_{\mathbb{R}}(G(\jmath\omega)) = g_2(\jmath\omega, \gamma_{\min}(\omega)). \quad (8)$$

For our criss-cross algorithm, we need the following theorem:

Theorem 1. *For a fixed $\gamma \neq 0$ and a level set parameter ξ, we have that*

$$g_\ell(\jmath\omega, \gamma) = \xi \quad \textit{for some } \ell = 1, \ldots, N^*, \quad (9)$$

if and only if $\lambda = \jmath\omega$ is a solution of

$$\mathcal{D}_{\xi\gamma}(\lambda) \cdot y = 0, \qquad \textit{nonzero } y, \quad (10)$$

with

$$\mathcal{D}_{\xi\gamma}(\lambda) = \lambda I + M_0 + \sum_{i\neq 0,\, i=-m}^{m} M_i e^{-\tau_i\lambda} + N_i e^{-2\tau_i\lambda}, \quad (11)$$

where

$$M_0 = \begin{bmatrix} 0 & 0 & -\alpha\Phi & 0 \\ 0 & 0 & 0 & -\alpha\Phi \\ \alpha\Psi & \beta\Psi & 0 & 0 \\ \beta\Psi & \alpha\Psi & 0 & 0 \end{bmatrix},$$

and, for $i = 1, \ldots, m$, we define $\tau_{-i} = -\tau_i$,

$$M_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -A_i^T & 0 & 0 \\ 0 & 0 & -A_i & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad M_{-i} = \begin{bmatrix} A_i^T & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_i \end{bmatrix},$$

and

$$N_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -\beta\Phi_i & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad N_{-i} = \begin{bmatrix} 0 & 0 & 0 & -\beta\Phi_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

with $\alpha = \frac{1+\gamma^2}{2\gamma}$, $\beta = \frac{1-\gamma^2}{2\gamma}$, $\Psi = \xi^{-1}BB^T$, $\Phi_i = \xi^{-1}C_i^TC_i$ *and* $\Phi = \sum_{i=1}^m \Phi_i$.

Proof. For any $\lambda \in \mathbb{C}$ we have by direct verification that

$$\begin{aligned} 2\begin{bmatrix} \Re G(\lambda) & \jmath\gamma\Im G(\lambda) \\ \jmath\gamma^{-1}\Im G(\lambda) & \Re G(\lambda) \end{bmatrix} = \\ = \begin{bmatrix} I_p & \gamma I_p \\ \gamma^{-1}I_p & -I_p \end{bmatrix}\begin{bmatrix} G(\lambda) & \\ & \overline{G(\lambda)} \end{bmatrix}\begin{bmatrix} I_q & \gamma I_q \\ \gamma^{-1}I_q & -I_q \end{bmatrix} = \\ = \underbrace{\begin{bmatrix} \mathbf{C}(\lambda) & \gamma\overline{\mathbf{C}(\lambda)} \\ \gamma^{-1}\mathbf{C}(\lambda) & -\overline{\mathbf{C}(\lambda)} \end{bmatrix}}_{\mathbf{C}_\gamma(\lambda)}\underbrace{\begin{bmatrix} D(\lambda) & \\ & \overline{D(\lambda)} \end{bmatrix}^{-1}}_{D_{[2]}(\lambda)}\underbrace{\begin{bmatrix} B & \gamma B \\ \gamma^{-1}B & -B \end{bmatrix}}_{B_\gamma}, \end{aligned}$$

where we do not indicate the conjugate of B since B is real-valued. The augmented eigenvalue characterization of the singular value (9) then implies

$$\begin{aligned} 0 &= \det\left(\frac{1}{2}\begin{bmatrix} 0 & B_\gamma^T D_{[2]}^{-H}(\lambda)\mathbf{C}_\gamma^H(\lambda) \\ \mathbf{C}_\gamma(\lambda)D_{[2]}^{-1}(\lambda)B_\gamma & 0 \end{bmatrix} - \xi I\right) \\ &= \det\left(\frac{1}{2}\begin{bmatrix} 0 & \mathbf{C}_\gamma^H(\lambda)\mathbf{C}_\gamma(\lambda)D_{[2]}^{-1}(\lambda) \\ B_\gamma B_\gamma^T D_{[2]}^{-H}(\lambda) & 0 \end{bmatrix} - \xi I\right) \quad (12) \\ &= \det\left(\frac{1}{2}\begin{bmatrix} 0 & \mathbf{C}_\gamma^H(\lambda)\mathbf{C}_\gamma(\lambda) \\ B_\gamma B_\gamma^T & 0 \end{bmatrix} - \xi\begin{bmatrix} D_{[2]}^H(\lambda) & \\ & D_{[2]}(\lambda) \end{bmatrix}\right) \end{aligned}$$

where we exploited in the second equality the identity $0 = \det(XY - \xi I) = \det(YX - \xi I)$ for the following choice of matrices X and Y

$$X = \begin{bmatrix} B_\gamma^T D_{[2]}^{-H}(\lambda) & 0 \\ 0 & \mathbf{C}_\gamma(\lambda)D_{[2]}^{-1}(\lambda) \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & \mathbf{C}_\gamma^H(\lambda) \\ B_\gamma & 0 \end{bmatrix},$$

and in the third equality we right multiplied with matrix Z

$$Z = \begin{bmatrix} D_{[2]}^H(\lambda) & 0 \\ 0 & D_{[2]}(\lambda) \end{bmatrix}.$$

Now we left multiply the last term of equation (12) with ξ^{-1} and plug into equation (12) the following

$$\frac{\mathbf{C}_\gamma^H(\lambda)\mathbf{C}_\gamma(\lambda)}{2} = \sum_{i=1}^m \begin{bmatrix} \alpha/\gamma C_i^T C_i e^{-2\tau_i \Re\lambda} & -\beta C_i^T C_i e^{-2\tau_i\overline{\lambda}} \\ -\beta C_i^T C_i e^{-2\tau_i\lambda} & \alpha\gamma C_i^T C_i e^{-2\tau_i\Re\lambda} \end{bmatrix}$$

and

$$\frac{B_\gamma B_\gamma^T}{2} = \begin{bmatrix} \alpha\gamma BB^T & \beta BB^T \\ \beta BB^T & \alpha/\gamma BB^T \end{bmatrix},$$

then we obtain

$$\det\left(\begin{bmatrix} -D^H(\lambda) & 0 & \alpha/\gamma\Phi(\Re\lambda) & -\beta\Phi(\overline{\lambda}) \\ 0 & -D^H(\overline{\lambda}) & -\beta\Phi(\lambda) & \alpha\gamma\Phi(\Re\lambda) \\ \alpha\gamma\Psi & \beta\Psi & -D(\lambda) & 0 \\ \beta\Psi & \alpha/\gamma\Psi & 0 & -D(\overline{\lambda}) \end{bmatrix}\right) = 0,$$

where $\Phi(\lambda) = \sum_{i=1}^m \Phi_i e^{-2\tau_i\lambda}$. Now, since parameter γ and the sign change operated on the blocks associated with $\Phi(\Re\lambda)$ do not affect the determinant, this is equivalent to

$$\det\left(\begin{bmatrix} -D^H(\lambda) & 0 & -\alpha\Phi(\Re\lambda) & -\beta\Phi(\overline{\lambda}) \\ 0 & D^H(\overline{\lambda}) & -\beta\Phi(\lambda) & -\alpha\Phi(\Re\lambda) \\ \alpha\Psi & \beta\Psi & D(\lambda) & 0 \\ \beta\Psi & \alpha\Psi & 0 & -D(\overline{\lambda}) \end{bmatrix}\right) = 0. \quad (13)$$

Finally, recalling that the singular value equation (9) is defined for $\lambda = \jmath\omega$, $\omega \in \mathbb{R}^+$, we consider a purely complex root of equation (13), hence such that $\overline{\lambda} = -\lambda$; it is easy to observe that such λ is also a solution of equation (11). □

Observe that this DEP contains positive and negative delay values; it is a natural extension to delay systems of the result proved in [20].

2.2. The criss-cross algorithm

We now explain our algorithm to compute $r_{\mathbb{R}}^{-1} = \sup_{\omega\geq 0}\mu_{\mathbb{R}}(G(\jmath\omega))$, which is an extension of that in [2, 12] to the delay case. The main idea of [2] to compute the maximum of a function $\mu(G(\jmath\omega))$ is by iteratively improving a horizontal level ξ that represents the maximum. Starting with a searching level ξ, it constructs the superlevel sets of $\mu(G(\jmath\omega))$ at ξ, that is, the intervals $\{\omega \in \mathbb{R}\colon \mu(G(\jmath\omega)) \geq \xi\}$. Then it updates ξ as the largest value of $\mu(G(\jmath\omega))$ evaluated in the mid points of each of these intervals. This procedure is guaranteed to converge globally; see [2].

To apply these ideas to $\mu_{\mathbb{R}}(G(\jmath\omega))$, we start with (7) to observe that

$$g_2(\jmath\omega,\gamma) \geq \mu_{\mathbb{R}}(G(\jmath\omega)), \qquad \forall\ \gamma \in (0,1],\ \omega \in \mathbb{R}^+,$$

and the equality is attained when $\gamma = \gamma_{\min}(\omega)$ is a global minimizer of $g_2(\jmath\omega,\gamma)$; see (8). This means that for a fixed γ, the function $g_2(\jmath\omega,\gamma)$ is a super function (that is, an overestimator) of $\mu_{\mathbb{R}}(G(\jmath\omega))$ that shares the same function value at that particular ω.

The main idea of our criss-cross algorithm is now as follows: Given an approximate maximizer $\widetilde{\omega} \in \mathbb{R}^+$ and a searching interval Ω that contains the maximizer, we evaluate $\xi = \mu_{\mathbb{R}}(G(\jmath\widetilde{\omega}))$ by applying golden section searching on γ to (7). This also gives us $\widetilde{\gamma} = \gamma_{\min}(\widetilde{\omega})$. We then compute all the intersections between the horizontal line at ξ and the super function $g_2(\jmath\omega,\widetilde{\gamma})$. Theorem 1 shows that this can be achieved by simply computing the purely imaginary eigenvalues of the DEP (10) and choosing only those that correspond to the second singular value that defines $g_2(\jmath\omega,\gamma)$. With these points we can define, after intersection with Ω, the K intervals $[\ell_j, u_j]$ such that

$$\bigcup_{j=1}^K [\ell_j, u_j] \ = \ \{\omega\colon g_2(\jmath\omega,\widetilde{\gamma}) \geq \xi\} \cap \Omega.$$

Then, we compute the $\mu_{\mathbb{R}}$ function in the midpoints $c_j = (\ell_j + u_j)/2$, and update $\widetilde{\omega}$ as

$$\widetilde{\omega}_{\text{new}} = \text{argmax}_{\omega=c_1,\ldots,c_K}\ \mu_{\mathbb{R}}(G(\jmath\omega)).$$

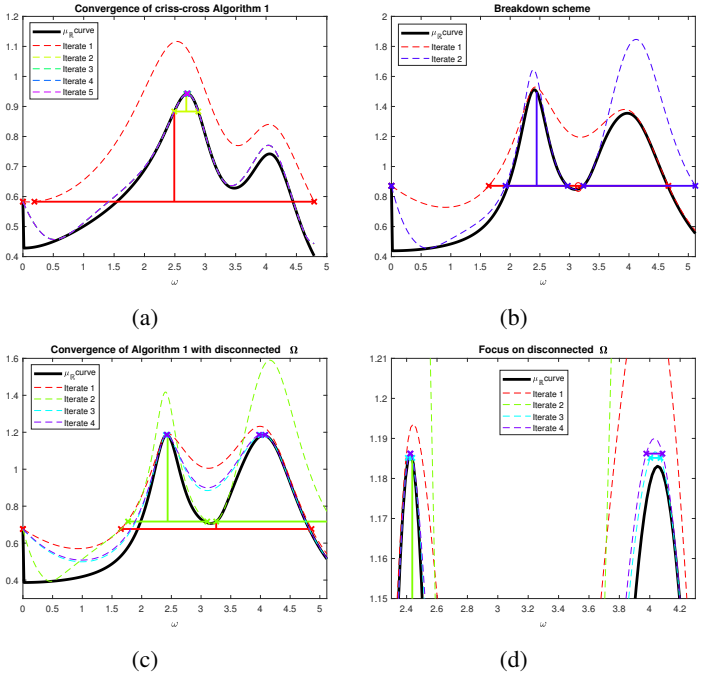


Figure 1: Different behaviors of Algorithm 1 applied to Example 1 in Section 4 are represented: the normal convergence in panel (a), the breakdown case in panel (b), and the normal convergence with two intervals in Ω in panel (c), whose zoom on the local maxima is shown in panel (d).

Usually $\mu_{\mathbb{R}}(G(j\widetilde{\omega}_{\text{new}})) \geq \xi$, which means we can update the searching level and the super function as follows:

$$\xi_{\text{new}} = \mu_{\mathbb{R}}(G(j\widetilde{\omega}_{\text{new}})), \quad \gamma_{\text{new}} = \gamma_{\min}(\widetilde{\omega}_{\text{new}}).$$

Otherwise, we define $c_{\max}$ as the midpoint of the largest interval and update as follows:

$$\xi_{\text{new}} = \xi, \quad \gamma_{\text{new}} = \gamma_{\min}(c_{\max}).$$

This situation is called *break down*. In this case, we keep the old searching level ξ and choose the new super function that belongs to the largest searching interval. This procedure is repeated with the updated values $\widetilde{\omega}_{\text{new}}, \xi_{\text{new}}$, and the searching intervals

$$\Omega_{\text{new}} = \bigcup_{j=1}^{K} \Big([\ell_j, c_j] \cup [c_j, u_j] \Big).$$

The algorithm is always initialized with $\omega = 0$ and $\Omega = \mathbb{R}^+$; since function $g_2(0, \gamma)$ does not actually depend on γ, the value γ is randomly initialized in the set $(0, 1]$. We stop the algorithm when the length of the largest interval in Ω is smaller than a user supplied tolerance.

The resulting scheme is illustrated in Figure 1. In particular in panels (a) and (c) show the usual behavior without breakdown where the algorithm finds a strictly higher searching level ξ at each iteration and eventually converges to the global maximum. The break down case is visible in panel (b), where the midpoint in iteration does not not produce a higher searching level of the $\mu_{\mathbb{R}}$ function. In the figure, $g_2(j\omega, \gamma_{\text{new}})$ is indicated with the blue dashed line.

Algorithm 1 Criss-cross search by touching functions $g_2(j\omega, \gamma)$ ($\text{rank}(\Im G(j\omega)) \geq 2$)

Input: System matrices $A_i \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $C_i \in \mathbb{R}^{q_i\times n}$, and delays $\tau_i \geq 0$, for $i = 1, \ldots, m$. Tolerance tol > 0.
Output: Real stability radius $r_{\mathbb{R}}$.

Quick return: if an eigenvalue of the DEP $D(s)$ has positive real part (i.e., delay system is unstable), return $r_{\mathbb{R}} = 0$.
Initialize: $\omega_0 = 0$, random $\gamma_0 \in (0, 1]$, search level $\xi_0 = \mu_{\mathbb{R}}(G(\omega_0))$, search interval $\Omega_0 = [0, \infty]$.
for $i = 0, 1, \ldots$ until convergence **do**
  Solve the DEP (10) with γ_i and ξ_i, and define the searching intervals $\{[\ell_j, u_j]\}_{j=1}^{K_i} = \{\omega \colon g_2(j\omega, \gamma_i) \geq \xi_i\} \bigcap \Omega_i$.
  if $\max_{j=1}^{K_i} |u_j - \ell_j| <$ tol, **then break**
  for $j = 1, \ldots, K_i$ **do**
    Set midpoint $c_j = (\ell_j + u_j)/2$.
    Evaluate $\mu'_j = \mu_{\mathbb{R}}(G(jc_j))$ which also gives γ'_j.
  end for
  if $\max_{j=1}^{K_i} \{\mu'_j\} > \xi_i$ **then**
    Higher level found, set $\nu = \text{argmax}_{j=1}^{K_i} \{\mu'_j\}$.
  else
    Breakdown detected, set $\nu = \arg\max_{j=1}^{K_i} |u_j - \ell_j|$.
  end if
  $\xi_{i+1} = \mu'_\nu$, $\gamma_{i+1} = \gamma'_\nu$, $\Omega_{i+1} = \cup_{j=1}^{K_i} ([\ell_j, c_j] \cup [c_j, u_j])$.
end for
Return $r_{\mathbb{R}} = \xi_i^{-1}$.

A pseudo-code of the criss-cross algorithm is reported in Algorithm 1. The search strategy based on horizontally intersecting superfunctions is also used in [12]. Whereas we apply this to the $\mu_{\mathbb{R}}$ function of a delay system, in [12] it is used for structured perturbations of LTI systems. The convergence analysis in both cases remains however the same, provided the intersection are computed exactly. In particular, Theorems 1 and 2 in [12] stating global convergence of ξ_i and ω_i also apply here.

2.3. The rank one case

Let us now examine the case of $\text{rank}(\Im G(j\omega)) = 1$ for all $\omega \in \mathbb{R}^+$. From (5), we see that this holds, for example, when $p = 1$ since then B and thus G are vectors. From [19] and with $X = \Re G(j\omega)$, we know that in this case

$$\mu_{\mathbb{R}}(G(j\omega)) = \lim_{\gamma\to 0} g_2(j\omega, \gamma) = \max\{\sigma_1(U_2^T X), \sigma_1(X V_2^T)\}, \quad (14)$$

where U_2 comes from the singular value decomposition

$$\Im G(j\omega) = [U_1\ U_2] \begin{bmatrix} \sigma_1(\Im G(j\omega)) & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}. \quad (15)$$

Due to this limit, Theorem (1) does not apply and we therefore cannot formulate the intersections based on a DEP like (10). We therefore set γ to some small value, say, 10^{-6}, and consider function $g_2(j\omega, 10^{-6})$, which by continuity is close to $\mu_{\mathbb{R}}(G(j\omega))$ for each $\omega \in \mathbb{R}^+$. Since γ is now constant, its horizontal intersections can be computed more easily, for example, by the Boyd–Balakrishnan method [2] applied to $g_2(j\omega, 10^{-6})$.

3. A new solver for DEPs with positive and negative delays

In this section, we provide the mathematical details of the DEP solver for (10) used in step 4 of Algorithm 1.

Using Theorem 1, it suffices to compute only the purely imaginary eigenvalues of the DEP (10). Taking $\lambda = \jmath\omega$, we obtain immediately that these eigenvalues are bounded:

$$|\lambda| \leq \|M_0\| + \sum_{i\neq 0,\, i=-m}^{m} \|M_i\| + \|N_i\|. \tag{16}$$

Therefore, as isolated roots of an analytic function, there are only finitely many.

The overall solver has three components: an approximation of the DEP as a generalized eigenvalue problem using a spectral discretization, an automated procedure to determine the resolution of this discretization, and a block-Newton correction to eliminate the approximation error.

Throughout this section we make the following assumption, which can always be achieved by solving for $\tilde{\lambda} = s\lambda$ in $\mathcal{D}_{\xi\gamma}(s^{-1}\tilde{\lambda})$ with a proper scaling s.

Assumption 1. *The maximum delay satisfies* $2\tau_m = 1$.

3.1. Spectral discretization of the eigenvalue problem

The delay eigenvalue problem (10) can be formulated as a linear eigenvalue problem for a differential operator. Indeed, if (λ, x) is an eigenpair of (10), then $F(t) = e^{\lambda t}x$ is easily seen to be a solution of the ODE

$$\dot{F}(t) = \lambda F(t), \;\; t \in [-1,\, 1], \tag{17}$$

with initial value $F(0)$ satisfying

$$(\lambda I + M_0)F(0) + \sum_{i\neq 0,\, i=-m}^{m} M_i F(-\tau_i) + N_i F(-2\tau_i) = 0. \tag{18}$$

Vice-versa, given λ, any solution of (17) is of the form $F(t) = e^{\lambda t}x$ for some $x \in \mathbb{C}^{4n}$. Substituting this expression into (18) yields (10) so that (λ, x) has to be an eigenpair.

To numerically solve (17)–(18), we can use spectral discretization [21, 3]. To this end, let $N \in \mathbb{N}$ and consider a grid on $[-1,\, 1]$ with grid points

$$\theta_{N,i} = -\cos\frac{\pi i}{2N+1}, \; i = 1, \ldots, 2N. \tag{19}$$

We approximate $F(t)$ by a polynomial of degree $2N$,

$$f(t) = \sum_{i=0}^{2N} c_i T_i(t), \quad c_i \in \mathbb{C}^{4n},$$

where T_i is the Chebyshev polynomial of the first kind and order i. The coefficients c_i are determined by collocating (18) and (17) at the points $\theta_{N,i}$:

$$\begin{cases} \lambda f(0) = -M_0 f(0) + \sum_{i\neq 0,\, i=-m}^{m} -M_i f(-\tau_i) - N_i f(-2\tau_i), \\ \dot{f}(\theta_{N,i}) = \lambda f(\theta_{N,i}), \quad i = 1, \ldots, 2N. \end{cases} \tag{20}$$

Computations similar as in Section 2.3 of [10] allow us to turn conditions (20) into a generalized eigenvalue problem of order $4n(2N+1)$,

$$(\Sigma_N - \lambda\Pi_N)\mathbf{c}_N = 0, \tag{21}$$

where $\mathbf{c}_N = [c_0^T\; c_1^T\; \cdots\; c_{2N}^T]^T$,

$$\Pi_N = \begin{bmatrix} T_0(1) & T_1(1) & T_2(1) & \cdots & & & \cdots & T_{2N}(1) \\ 1 & 0 & -\frac{1}{2} & & & & & \\ & \frac{1}{4} & 0 & -\frac{1}{4} & & & & \\ & & \frac{1}{6} & 0 & -\frac{1}{6} & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \frac{1}{2(2N-1)} & 0 & -\frac{1}{2(2N-1)} & \\ & & & & & \frac{1}{4N} & 0 & \end{bmatrix} \otimes I_{4n} \tag{22}$$

and

$$\Sigma_N = \begin{bmatrix} R_0 & R_1 & \cdots & R_{2N} \\ & I_{4n} & & \\ & & \ddots & \\ & & & I_{4n} \end{bmatrix}, \tag{23}$$

with

$$R_k = M_0 T_k(1) + \sum_{i\neq 0, i=-m}^{m} M_i T_k(-\tau_i) + N_i T_k(-2\tau_i)\,.$$

Given an eigenpair $(\lambda, \mathbf{c}_N)$ of (21), we have thus obtained $(\lambda, f(0))$ as approximate eigenpair of the DEP.

3.2. Determining the number of discretization points

By Prop. 1 in [22], the eigenvalue λ of the discretized eigenvalue problem (21) is also an eigenvalue of

$$\mathcal{D}_{\xi\gamma}^N(\lambda) = \lambda I + M_0 + \sum_{i\neq 0,\, i=-m}^{m} M_i p_N(-\tau_i;\lambda) + N_i p_N(-2\tau_i;\lambda), \tag{24}$$

where $p_N(t;\lambda)$ is a polynomial of degree $2N$ in t that satisfies

$$\begin{cases} \dot{p}_N(\theta_{N,i};\; \lambda) = \lambda p_N(\theta_{N,i};\; \lambda), \quad i = 1, \ldots, 2N, \\ p_N(0;\; \lambda) = 1. \end{cases} \tag{25}$$

Observe that (24) is obtained by replacing $e^{\lambda t}$ in the original eigenvalue problem (10) by $p_N(t;\lambda)$. Hence, $\mathcal{D}_{\xi\gamma}^N$ can be seen as the following perturbation of (10):

$$\mathcal{D}_{\xi\gamma}^N(\lambda) = \mathcal{D}_{\xi\gamma}(\lambda) + \mathcal{E}^N(\lambda)$$

with $e_N(t;\lambda) = p_N(t;\lambda) - e^{\lambda t}$ and

$$\mathcal{E}^N(\lambda) = \sum_{i\neq 0,\, i=-m}^{m} M_i e_N(-\tau_i;\lambda) + N_i e_N(-2\tau_i;\lambda).$$

Since $\det\mathcal{D}_{\xi\gamma}^N(\lambda) = \det\mathcal{D}_{\xi\gamma}(\lambda) + O(\|\mathcal{E}^N(\lambda)\|)$, the computed eigenvalue λ of (21) (as root of an analytic function) will be a good approximation to an eigenvalue of the DEP (10) if $|e_N(t;\lambda)|$ is small for $t \in [-1, 1]$.

Given an accuracy $\varepsilon > 0$, we can therefore choose N as follows. Since we are interesting only in the purely complex eigenvalues of the DEP, we define

$$N(\omega) = \min\{n \in \mathbb{N} \colon |e_n(t;\jmath\mu)| < \varepsilon,\; \forall \mu \in [0,\omega], t \in [-1,1]\}\,.$$

This number $N(\omega)$ is the smallest degree needed to accurately approximate any purely imaginary eigenvalue $\lambda \in [0, \jmath\omega]$ since $\mathcal{E}^{N(\omega)}(\lambda) = O(\varepsilon)$.

Since $N(\omega)$ is problem independent, we can compute it beforehand, for instance, by evaluating $e_n(t; j\mu) = p_n(t; j\mu) - e^{j\mu t}$ for $n = 0, 1, \ldots$ on a sufficiently fine grid for $t \in [-1, 1]$ and $\mu \in [0, \omega]$. In turn, for given μ, the polynomial $p_n(t; j\mu)$ can be computed in a Chebyshev expansion, as outlined in the proof of Prop. A.1 of [7]. An example for $\epsilon = 10^{-4}$ is visible in Figure 2, which shows that $N(\omega)$ is approximately linear in ω.

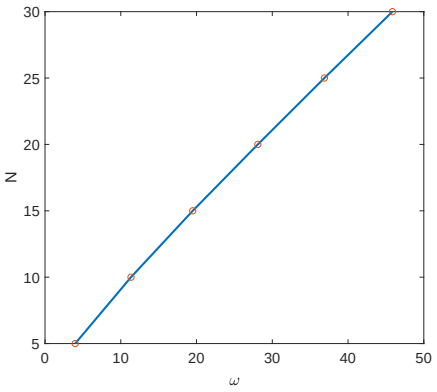


Figure 2: The function $N(\omega)$ for accuracy $\epsilon = 10^{-4}$.

It remains to determine an interval $[0, \jmath\omega_c]$ that contains all the imaginary eigenvalues. A conservative choice to estimate ω_c is (16). However, it is usually leads to unnecessarily large values of ω_c, and thus of N. Instead, we have adapted the approach from [22] that originally computes all the eigenvalues in right half plane of a DEP.

Let λ be an eigenvalue of the DEP (10), then from (11), we see that $-\lambda$ is also an eigenvalue of the matrix

$$M_0 + \sum_{i\neq 0, i=-m}^{m} M_i e^{-\tau_i \lambda} + N_i e^{-2\tau_i \lambda}.$$

For purely imaginary eigenvalue $\lambda = \jmath\omega$, this matrix therefore belongs to the set

$$\mathcal{M} = \bigcup_{\theta\in[0,2\pi]^m} \left(M_0 + \sum_{i\neq 0, i=-m}^{m} M_i e^{-j\theta_i} + N_i e^{-j2\theta_i} \right),$$

where $\theta_{-i} = -\theta_i$ for $i = 1, \ldots, m$. Hence, the original eigenvalue $-\lambda$ is also an eigenvalue of some matrix in $\mathcal{M}$. A bound for ω_c can then be determined by sampling the matrices in $\mathcal{M}$ on a grid for the parameters $\theta_i \in [0, 2\pi]$. We refer to [22] for details how to exploit commensurate delays.

In Algorithm 1 we can start with γ_0 and ξ_0 to obtain the initial estimation ω_c. Since the searching intervals will be monotonically reducing (including also the corresponding eigenvalues), this ω_c can be reused for the following iterations as well. This is especially important in the rank one case: the strategy from above with our use of $\gamma = 10^{-6}$ (see Section 2.3) would otherwise lead to unnecessarily large values of ω_c since $M_i = O(1/\gamma)$. We therefore use $\gamma_0 = 1$ in this case.

3.3. Correcting the eigenvalues with block Newton

The approximation error on the eigenpairs after spectral discretization is typically too large to obtain a good approximation of the stability radius during the criss-cross search. Instead of increasing the number of discretization points, which would be computationally too expensive, we correct the approximated eigenpairs by the block Newton method from [11]. This method applies the standard Newton iteration to find a pair $(X, S) \in \mathbb{C}^{n\times k} \times \mathbb{C}^{k\times k}$ that satisfies

$$XS + M_0 XS + \sum_{i=-1}^{m} \left(M_i X e^{-\tau_i S} + N_i X e^{-2\tau_i S} \right) = 0 \tag{26}$$

and the normalization condition

$$\left[X^T \; (XS)^T \; \ldots \; (XS^{l-1})^T \right] W - I_k = 0, \tag{27}$$

for some fixed integer l and matrix $W \in \mathbb{C}^{ln\times k}$. See [11] for implementation details.

A pair satisfying the equations (26)–(27) is called a minimal invariant pair of the DEP (10) and can be used to compute the eigenpairs of the DEP. For example, if (λ, y) is an eigenpair of S, then (λ, Xy) is also an eigenpair of (10). Vice-versa, if (10) has two non-defective eigenpairs (λ_1, x_1) and (λ_2, x_2) such that the x_1 and x_2 are linearly independent, then

$$(X, S) = \left(\begin{bmatrix} x_1 & x_2 \end{bmatrix}, \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \right) \tag{28}$$

is a minimal invariant pair. On the other hand, for a defective eigenvalue $\lambda_1 = \lambda_2$ with geometric multiplicity one,

$$(X, S) = \left(\begin{bmatrix} x & v \end{bmatrix}, \begin{bmatrix} \lambda_1 & 1 \\ 0 & \lambda_1 \end{bmatrix} \right) \tag{29}$$

is easily seen to be a minimal invariant pair. Here, x is the eigenvector and v is the first generalized eigenvector of problem (10), satisfying the Jordan chain condition

$$\mathcal{D}_{\xi,\gamma}(\lambda) \cdot v + \frac{\partial \mathcal{D}_{\xi,\gamma}(\lambda)}{\partial \lambda} \cdot x = 0. \tag{30}$$

The approximate eigenpairs computed by spectral discretization give approximate minimal invariant pairs that can be used as initial values for the Newton method. Let $(\widehat{\lambda_1}, \widehat{x_1})$ and $(\widehat{\lambda_2}, \widehat{x_2})$ be two such pairs that correspond to an intersecting interval, then $\widehat{\lambda_1} \neq \widehat{\lambda_2}$ will usually be non-defective. Hence, we can use as initial guess

$$(X_0, S_0) = \left(\begin{bmatrix} \widehat{x_1} & \widehat{x_2} \end{bmatrix}, \begin{bmatrix} \widehat{\lambda_1} & 0 \\ 0 & \widehat{\lambda_2} \end{bmatrix} \right).$$

However, as the horizontal level approaches its maximal value, the two intersections will converge leading to multiple purely imaginary eigenvalue. As similarly observed in [7, Fig. 1], due to the Hamiltonian structure, the eigenvalue typically has geometric multiplicity one. In this case, the initial guess becomes

$$(X_0, S_0) = \left(\begin{bmatrix} \widehat{x} & \widehat{v} \end{bmatrix}, \begin{bmatrix} \widehat{\lambda_1} & 1 \\ 0 & \widehat{\lambda_1} \end{bmatrix} \right),$$

Table 1: Results for the convergence of Algorithm 1.

k	ξ_i	$\lvert\xi_i - \xi_{i-1}\rvert$	ω_i	$\lvert\Omega_i\rvert$
1	0.5828159		2.490258410	4.6031×10^{0}
2	0.8832408	3.004×10^{-1}	2.688229335	3.960×10^{-1}
3	0.9420780	5.883×10^{-2}	2.704386454	3.232×10^{-2}
4	0.9425446	4.665×10^{-4}	2.704501012	2.292×10^{-4}
5	0.9425446	2.348×10^{-8}	2.704501012	3.836×10^{-8}

with $\widehat{x}$ the approximate eigenvector of $\widehat{\lambda}_1 \approx \widehat{\lambda}_2$, and $\widehat{v}$ is defined analogously to (30) by solving the near-singular system

$$\mathcal{D}_{\xi,\gamma}(\widehat{\lambda}) \cdot v = -\frac{\partial}{\partial \lambda} \mathcal{D}_{\xi,\gamma}(\widehat{\lambda}) \cdot \widehat{x}$$

using the pseudo-inverse (that sets the smallest singular value to zero). We emphasize that the block-Newton method computes the invariant pair regardless of the eigenpair being defective or not.

4. Numerical experiments

The first example gives rise to the illustrations in Figure 1 which we used to graphically describe the scheme of Algorithm 1; moreover, we observe that this algorithm exhibits a *superlinear* convergence. In the second example we compute the distance to instability when matrix Δ of uncertainties has rank one.

Example 1. *We consider the following system*

$$\dot{x}(t) = \sum_{i=1}^{3} (A_i + B\delta A_i C_i)\, x(t - \tau_i), \tag{31}$$

where $(\tau_1,\ \tau_2,\ \tau_3) = (0,\ 0.1702,\ 0.5681)$, $B = I_3$

$$A_1 = \begin{bmatrix} -0.090 & -0.816 & -0.228 \\ 0.769 & -1.325 & -1.380 \\ 0.412 & 1.523 & -0.760 \end{bmatrix}, A_2 = \begin{bmatrix} -0.869 & 0.136 & -1.077 \\ -0.149 & -0.939 & 0.445 \\ 0.476 & 1.862 & -0.191 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} -0.462 & 0.389 & -0.752 \\ 0.517 & -0.042 & 1.058 \\ -0.270 & -1.106 & -2.480 \end{bmatrix},$$

and

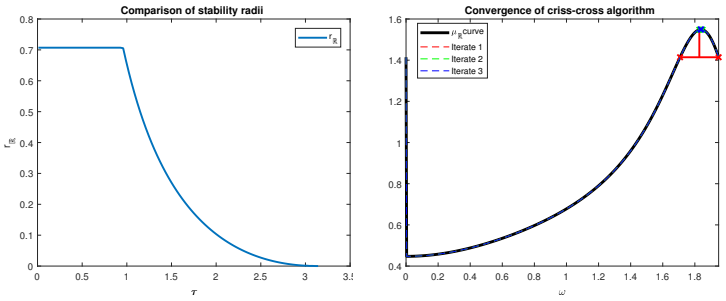
$$C_1 = \begin{bmatrix} 0.057 \\ 0.204 \\ -0.063 \end{bmatrix}^T, \quad C_2 = \begin{bmatrix} 0.157 \\ -0.921 \\ 0.221 \end{bmatrix}^T, \quad C_3 = \begin{bmatrix} 0.816 \\ -0.639 \\ -0.418 \end{bmatrix}^T.$$

In Figure 1a we observe the convergence of Algorithm 1 to the global maximum of $\mu_{\mathbb{R}}$, where we initialized $\gamma_0 = 0.95$. In Table 1 we report the iterations of the algorithm to analyze its convergence. In the first column we show the iteration number i, in the second the level set ξ_i, in the third the distance between two consecutive level sets, in the forth the corresponding frequency ω_i and in the last the width of the interval Ω_i. From these results, the algorithm seems quadratically convergent.

In Figure 1b the breakdown scheme is triggered by initializing $\gamma_0 = 0.7$ and by changing B and C_3 with $\widetilde{C}_3 = 1.41 C_3$ and $\widetilde{B}$ as follows

$$\widetilde{B} = \begin{bmatrix} 0.932 & 0.104 & -0.780 \\ 0.635 & -1.077 & 0.057 \\ 0.170 & 0.433 & -1.725 \end{bmatrix}.$$

Finally, Figures 1c and 1d can be reproduced by adopting $\widetilde{B}$ and C_3.



(a) (b)

Figure 3: Left pane shows the stability radius $r_{\mathbb{R}}$ w.r.t. the delay τ; right pane shows the convergence of the algorithm for a fixed $\tau = 1$.

Example 2. *Let us consider the uncertain system*

$$\dot{x}(t) = (A_0 + B\delta A_0 C_0)x(t) + (A_1 + B\delta A_1 C_1)x(t - \tau),$$

where $B = [0\ \ 1]^T$, $C_0 = C_1 = [1\ \ 1]$,

$$A_0 = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix}, \quad A_1 = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}.$$

Observe that δA_0 and δA_1 are both scalars; as a consequence its concatenation Δ and matrix $G(\jmath\omega)$ as defined in (5) have rank one independently of ω. The nominal system is proved to be unstable for all the delay values $\tau \geq \pi$ (see [4] for a proof). In Figure 3a we plot the radius $r_{\mathbb{R}}$ against the delay parameter τ varying in $(0, \pi)$. Note that the stability radius is constant for delay values smaller than $\tau^* \approx 1$; the reason is that for $\tau < \tau^*$ the smallest perturbation generating instability shifts an eigenvalue on the origin. This eigenvalue is then solution of the characteristic equation independently *of the delay value τ. In Figure 3b we show the convergence of Boyd-Balakrishnan method [2] applied to $g_2(\jmath\omega, 0.05)$ for $\tau = 1$: obviously dashed lines corresponding to different iterations are perfectly overlapping. From this figure we also observe that $g_2(\jmath\omega, 0.05)$ is a very good approximation of $\mu_{\mathbb{R}}$.*

Finally, in Table 2 we report the computation of the distance to instability for the benchmark problems[2] that were also used in [7] on computing the complex pseudospectral abscissa; in these cases, we considered *unstructured* matrix perturbations, so we set $B = C_i = I_n$ for each $i = 1, \ldots, m$. The first column shows the plant number in the benchmark; the second column show the size n and the number of delays m; the third column shows the computed distance to instability; the forth column shows the frequency maximizing $\mu_{\mathbb{R}}$, and the last column shows the computation time in seconds on a PC with an Intel Core i5 2.50 GHz processor with 8 GB RAM. We set the tolerance for the interval length equal to 10^{-5}. Note that plants 4 and 5 are not included as they define an unstable problem.

5. Concluding remarks

We presented a novel criss-cross type algorithm for computing the distance to instability for linear time-invariant delay-system affected by real valued perturbations on the matrices,

[2] http://www.cs.kuleuven.be/~wimm/software/psa/

Table 2: Speed of convergence of Algorithm 1 on different systems.

Plants	(n,m)	$r_{\mathbb{R}}$	ω	time
1	$(3,2)$	9.1595e-03	0	0.062
2	$(1,2)$	1.0607e-00	0	0.100
3	$(3,4)$	1.9917e-01	3.2287	0.910
6	$(10,8)$	3.1091e-01	1.5447	6.670
7	$(20,10)$	2.6094e-01	0	1.643
8	$(40,3)$	8.4134e-02	0.1717	19.887
9	$(5,2)$	3.2642e-01	1.2259	0.488
10	$(4,4)$	7.1411e-01	0	0.082

which is globally converging. It complements the fundamentally different algorithm of [1] nature, for which only local convergence can be guaranteed but allows to consider a broader class of perturbations (e.g different B-matrices for shaping the perturbation structure).

References

[1] F. Borgioli and W. Michiels. Computing the distance to instability for delay systems with uncertainties in the system matrices and in the delay terms. In *Proceedings of the ECC*, Limassol, Cyprus, 2018.

[2] S. Boyd and V. Balakrishnan. A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its $\mathcal{L}_\infty$-norm. *Systems & Control Letters*, 15:1–7, 1990.

[3] D. Breda, S. Maset, and R. Vermiglio. Pseudospectral differencing methods for characteristic roots of delay differential equations. *SIAM Journal on Scientific Computing*, 27(2):482–495, 2005.

[4] J. Chen, G. Gu, and C. A. Nett. A new method for computing delay margins for stability of linear delay systems. *Systems & Control Letters*, 26:107–117, 1995.

[5] C.E. de Souza and X. Li. Delay-dependent robust $\mathcal{H}_\infty$ control of uncertain linear state-delayed systems. *Automatica*, 35:1313–1321, 1999.

[6] K. Gu, V.L. Kharitonov, and J. Chen. *Stability of time-delay systems*. Birkhauser, 2003.

[7] S. Gumussoy and W. Michiels. A predictor-corrector type algorithm for the pseudospectral abscissa computation of time-delay systems. *Automatica*, 46(4):657–664, 2010.

[8] D. Hinrichsen and A.J. Pritchard. Stability radius for structured perturbations and the algebraic riccati equation. *Systems & Control Letters*, 8:105–113, 1986.

[9] G. Hu and E. J. Davison. Real stability radii of linear time-invariant time-delay systems. *Systems & Control Letters*, 50:209 – 219, 2003.

[10] E. Jarlebring, K. Meerbergen, and W. Michiels. A Krylov method for the delay eigenvalue problem. *SIAM Journal on Scientific Computing*, 32(6):3278–3300, 2010.

[11] D. Kressner. A block Newton method for nonlinear eigenvalue problems. *Numerische Mathematik*, 114(2):355–372, 2009.

[12] C.T. Lawrence, A. Tits, and P. Van Dooren. A fast algorithm for the computation of an upper bound on the μ-norm. *Automatica*, 36(3):449 – 456, 2000.

[13] X. Li and C.E. de Souza. Criteria for robust stability and stabilization of uncertain linear systems with state delay. *Automatica*, 33(9):1657–1662, 1997.

[14] W. Michiels, E. Fridman, and S.-I. Niculescu. Robustness assessment via stability radii in delay parameters. *International Journal of Robust and Nonlinear Control*, 19(13):1405–1426, 2009.

[15] W. Michiels, K. Green, T. Wagenknecht, and S.I. Niculescu. Pseudospectra and stability radii for analytic matrix functions with applications to time-delay systems. *Linear Algebra and its Applications*, 418(1):315–335, 2006.

[16] W. Michiels and S.I. Niculescu. *Stability and stabilization of time-delay systems. An eigenvalue based approach*. SIAM, 2007.

[17] W. Michiels and D. Roose. An eigenvalue based approach for the robust stabilization of linear time-delay systems. *International Journal of Control*, 76(7):678–686, 2003.

[18] S.I. Niculescu. *Delay effects on stability. A robust control approach*, volume 269 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, 2001.

[19] L. Qiu, B. Bernhardsson, A. Rantzer, E. Davison, P. Young, and J. Doyle. A formula for computation of the real stability radius. *Automatica*, 31:879–890, 1995.

[20] J. Sreedhar, P. Van Dooren, and A.L. Tits. A fast algorithm to compute the real structured stability radius. volume 121 of *International Series of Numerical Mathematics*, pages 219–230. 1996.

[21] Trefethen. *Spectral methods in MATLAB*, volume 10 of *Software, Environments, and Tools*. SIAM, 2000.

[22] Z. Wu and W. Michiels. Reliably computing all characteristic roots of delay differential equations in a given right half plane. *Journal of Computational and Applied Mathematics*, 236:2499–2514, 2012.