# TDS-CONTROL: a MATLAB package for the analysis and controller-design of time-delay systems ⋆

**Pieter Appeltans** * **Haik Silm** * **Wim Michiels** *

* *NUMA, Department of Computer Science, KU Leuven*
*Celestijnenlaan 200A, 3001 Leuven, Belgium*
*(e-mail: {pieter,haik,wim}.{appeltans,silm,michiels}@kuleuven.be)*

**Abstract:** This paper presents TDS-CONTROL, a MATLAB package for the analysis and design of controllers for linear time-invariant systems with discrete delays. The code is based on a state-space representation of such systems in terms of delay-differential algebraic equations with input, output, and state delays. As such, a broad class of (interconnected) systems can be considered, including neutral systems. The controller design algorithms are based on optimizing a certain objective function, such as the spectral abscissa, the H-infinity norm, or the pseudo-spectral abscissa, with respect to the controller parameters. For example, to design a stabilizing controller the spectral abscissa, i.e., the real part of the right-most characteristic root, is minimized. As a strictly negative spectral abscissa is a necessary and sufficient condition for stability, the presented design method is not conservative and a stabilizing controller can be computed whenever it exits. This comes however at the cost of having to solve a non-smooth, non-convex optimization problem. The class of considered controllers consists of static and dynamic output-feedback controllers. By allowing to fix certain entries in the controller matrices, it is also possible to design structured controllers such as decentralized, overlapping and PID-like controllers. Finally, as a wide-range of delay systems can be considered, the software package takes the sensitivity of certain quantities, such as the spectral abscissa and the H-infinity norm, with respect to infinitesimal delay perturbations explicitly into account.

The source code is freely available at `https://gitlab.kuleuven.be/u0011378/tds-control`.

*Keywords:* Systems with time-delays – stability of delay systems – optimization based controller design – fixed-order controller design

## 1. INTRODUCTION

For certain applications, such as embedded aerospace control systems, fixed-order controllers (in which the order of the controller can be smaller than the order of the plant) are desirable (Siqueira et al., 2012). For linear time-invariant (LTI) state-space systems without delays, the design of fixed-order controllers by optimizing a certain control objective (such as the spectral abscissa or the $\mathcal{H}_\infty$-norm) with respect to the parameters of the controller has attracted significant interest, e.g., the MATLAB packages HIFOO (Gumussoy et al., 2009) or HINFSTRUCT (Apkarian and Noll, 2006). The presented software package extends this framework to LTI systems with discrete delays. More specifically, TDS-CONTROL can deal with systems of the form

$$\begin{cases} E\dot{x}(t) = \sum_i A_i x(t-\tau_i^A) + \sum_i B_i u(t-\tau_i^B) - \sum_i H_i \dot{x}(t-\tau_i^H) \\ y(t) = \sum_i C_i x(t-\tau_i^C) + \sum_i D_i u(t-\tau_i^D) \end{cases}$$

with $E$ potentially singular. Delays appear in many real-life applications. For example, in control loops delays typically arise from the time needed for data communication and processing. Delays can however also occur in the dynamical systems themselves, such as in milling processes (Insperger and Stépán, 2000).

Compared to undelayed LTI systems, it is well known that systems with delay are in some sense "infinite dimensional" (Michiels and Niculescu, 2014). For example, similar to undelayed LTI systems, the stability of a LTI delay system can be examined in terms of the roots of an associated characteristic function. However, where for delay-free systems this function is a polynomial with as many roots as the order of the system, the characteristic function associated with a delay system is non-linear and has infinitely many roots. In this sense, it is not possible to fully control the dynamics of a delay system using static or dynamic output feedback controllers as they only contain finitely many control parameters to tune. To solve such synthesis problems one thus has to resort to numerical optimization.

In the main part of this paper we will restrict our attention to closed-loop systems of retarded type. Although the stability of such systems is characterized by infinitely many characteristic roots, the number of roots in any

⋆ This work was supported by the project G092721N of the Research Foundation-Flanders (FWO - Vlaanderen).

right half-plane is finite, which simplifies the presentation. The presented software package can however also deal with systems that exhibit neutral behavior for which the stability analysis is more involved (Michiels and Niculescu, 2014, ch. 1.2). We will briefly comment on how the software deals with neutral(-like) systems at the end of this work.

TDS-CONTROL integrates existing algorithms for stabilization (Vanbiervliet et al., 2008), robust stabilization (Dileep et al., 2018) and $\mathcal{H}_\infty$-norm optimization (Gumussoy and Michiels, 2011) of time-delay systems. It is also allows for structured controllers such as decentralized, overlapping or PID-controllers (Dileep et al., 2020).

The remainder of this paper is organized as follows. In Section 2 we will illustrate the working principles of the software package by designing a stabilizing position feedback controller for a cart–pendulum system. In Section 3 we consider systems with uncertainties and demonstrate how these system can be robustly stabilized using TDS-CONTROL. Section 4 briefly discusses $\mathcal{H}_\infty$-norm optimization and the design of structured controllers. Section 5 discusses the additional functionality of the software package that is needed to deal with (closed-loop) systems that exhibit neutral behavior. Finally, Section 6 draws some concluding remarks.

## 2. STABILISATION

In this paper, we will mainly restrict our attention to LTI state-space systems with discrete delays of the following form:

$$\begin{cases} \dot{x}(t) = \sum_{i=0}^{m} A_i x(t-\tau_i) + \sum_{i=0}^{m} B_i u(t-\tau_i) \\ y(t) = \sum_{i=0}^{m} C_i x(t-\tau_i) \end{cases} \tag{1}$$

with $0 = \tau_0 < \tau_1 < \cdots < \tau_m < \infty$ the delay values and for all $t \in [-\tau_m, +\infty)$ $x(t) \in \mathbb{R}^n$ the state variable, $u(t) \in \mathbb{R}^p$ the control input, and $y(t) \in \mathbb{R}^q$ the observed output. All matrices are real-valued and have appropriate dimensions. To limit the number of subscripts, we use the same delays in the state, in- and output terms, although the user can actually enter different delays in the different terms as shown in the code examples. Note that TDS_CONTROL can also deal with direct feed-through terms and with the more general class of delay descriptor systems, in which the internal dynamics are described by delay-differential algebraic equations (Michiels and Niculescu, 2014, ch. 1.3).

To stabilize system (1), we will use a dynamic fixed-order controller of the form

$$\begin{cases} \dot{x}_c(t) = A_c x_c(t) + B_c y(t) \\ u(t) = C_c x_c(t) + D_c y(t) \end{cases} \tag{2}$$

with $x_c(t) \in \mathbb{R}^{n_c}$ for all $t \geq 0$, $n_c$ the controller order, and $A_c$, $B_c$, $C_c$ and $D_c$ real-valued matrices with appropriate dimensions. If $n_c = 0$, this controller reduces to static output-feedback, i.e., $u(t) = D_c\, y(t)$. In general, it is not trivial to derive an expression for the closed loop of (1) and (2) in the form of delay-differential equations due to the presence of delays in both the in- and output terms. TDS_CONTROL therefore internally uses a representation in terms of delay-differential algebraic equations, which combine differential and algebraic equations with delays, to represent the closed-loop system. More specifically, the closed loop of (1) and (2) can be represented by:

$$S_{cl} \leftrightarrow \underbrace{\begin{bmatrix} I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{E^{cl}} \begin{bmatrix} \dot{x}(t) \\ \dot{\xi}_y(t) \\ \dot{x}_c(t) \\ \dot{\xi}_u(t) \end{bmatrix} = \underbrace{\begin{bmatrix} A_0 & 0 & 0 & B_0 \\ C_0 & -I & 0 & 0 \\ 0 & B_c & A_c & 0 \\ 0 & D_c & C_c & -I \end{bmatrix}}_{A_0^{cl}} \begin{bmatrix} x(t) \\ \xi_y(t) \\ x_c(t) \\ \xi_u(t) \end{bmatrix} + \sum_{k=1}^{m} \underbrace{\begin{bmatrix} A_k & 0 & 0 & B_k \\ C_k & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{A_k^{cl}} \begin{bmatrix} x(t-\tau_k) \\ \xi_y(t-\tau_k) \\ x_c(t-\tau_k) \\ \xi_u(t-\tau_k) \end{bmatrix} \tag{3}$$

in which $\xi_y$ and $\xi_u$ are auxiliary variables used to avoid explicitly eliminating the input and output channels, a generally useful technique in the modelling of interconnected systems (Dileep et al., 2020).

Exponential stability of (3) is characterized by the roots of the following characteristic function:

$$f(\lambda; S_{cl}) := \det\left( E^{cl}\lambda - A_0^{cl} - \sum_{i=1}^{m} A_i^{cl} e^{-\lambda\tau_i} \right). \tag{4}$$

More specifically, the closed-loop system is stable if and only if all roots of (4) lie to the left of the imaginary axis. By computing these roots, stability of a delay system can thus be assessed without any conservativeness, in contrast to methods relying only on sufficient conditions (like Lyapunov-Krasovskii functionals). To compute the roots of (4) in a given right half-plane (recall from the introduction that the number of roots in such a right half-plane is always finite), TDS_CONTROL implements the algorithm from Wu and Michiels (2012). This algorithm combines a spectral discretization of the infinitesimal generator of the solution operator associated with the closed-loop system (Breda et al., 2006) with Newton corrections based on the nonlinear characteristic function (4). An appealing feature of this algorithm is that the user only needs to specify the desired right half-plane as it automatically selects the required number of spectral discretization points.

Equivalently, stability of (3) can be expressed in terms of the spectral abscissa, which is the real part of the rightmost characteristic root

$$\alpha(S_{cl}) = \max\{\Re(\lambda) : f(\lambda; S_{cl}) = 0\},$$

i.e., closed-loop system (3) is exponentially stable if and only if $\alpha < 0$. Note, as for delay-free systems, the spectral abscissa of a stable delay system gives the exponential decay-rate of the state to the null solution. TDS_CONTROL tries to derive a stabilizing controller with a given order $n_c$ by minimizing this spectral abscissa with respect to the entries of the controller matrices $A_c$, $B_c$, $C_c$ and $D_c$. Similarly as for systems without delays, the resulting optimization problem is typically non-convex and non-smooth (but almost everywhere differentiable). The package therefore uses HANSO (Lewis and Overton, 2013), which implements a hybrid algorithm for non-smooth optimization by combining a quasi-Newton method with gradient sampling, to solve the resulting optimization problems. Since the stability conditions are necessary and sufficient, a stabilizing controller can always be found if it exists. However, due to the non-convexity, the result of the optimization procedure is sensitive to the chosen starting

point. Therefore, different initial values have to be sampled until a satisfactory controller is found.

To illustrate the use of `TDS-CONTROL` on an example, we will consider the set-up depicted in Fig. 1 that consists of a cart that balances an inverted pendulum and that is connected to two walls using two identical springs. Assuming that the angular displacement $\theta$ remains small, this system can be approximated by the following LTI model

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\\ \dot{x}_3(t)\\ \dot{x}_4(t)\end{bmatrix} = \underbrace{\begin{bmatrix}0 & 1 & 0 & 0\\ \frac{-2k}{M} & 0 & \frac{-mg}{M} & 0\\ 0 & 0 & 0 & 1\\ \frac{2k}{Ml} & 0 & \frac{(m+M)g}{Ml} & 0\end{bmatrix}}_{A}\begin{bmatrix}x_1(t)\\ x_2(t)\\ x_3(t)\\ x_4(t)\end{bmatrix} + \underbrace{\begin{bmatrix}0\\ \frac{1}{M}\\ 0\\ \frac{-1}{Ml}\end{bmatrix}}_{B} u(t) \quad (5)$$

with $k$ the spring constant of the two springs, $m$ the mass of the pendulum's bob, $M$ the mass of the cart, $x_1 = p$ the horizontal displacement of the cart with respect to its equilibrium position, $x_2 = \dot{p}$ its horizontal velocity, $x_3 = \theta$ the angular displacement and $x_4 = \dot{\theta}$ the angular velocity. As measured output, the position and the angular displacement are available:

$$y(t) = \underbrace{\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}}_{C} x(t). \quad (6)$$

If we try to stabilize this system using static output-feedback, i.e., $u(t) = [d_1 \;\; d_2]\, y(t)$ with $d_1$ and $d_2$ belonging to $\mathbb{R}$, the stability of the closed loop can be deduced from the roots of the following characteristic polynomial:

$$\lambda^4 + \lambda^2\left(\frac{2k-d_1}{M} - \frac{(M+m)g - d_2}{Ml}\right) - \frac{(2k-d_1)g}{Ml}.$$

As this polynomial is bi-quadratic, it has at least two roots in the closed right-half plane, independent of the chosen control parameters $d_1$ and $d_2$. The considered system can thus not be stabilized using (undelayed) position feedback.

A solution could be to resort to full state-feedback. This would however require that the velocities $\dot{p}$ and $\dot{\theta}$ are known. If these quantities can not be measured directly, one could use a finite difference approximation, e.g., $\dot{p}(t) \approx \frac{p(t)-p(t-\tau)}{\tau}$ with $\tau > 0$ a sufficiently small delay. Such an approach can however be sensitive to measurement noise. We will therefore instead consider the following feedback law $u(t) = D_0^c y(t) + D_1^c y(t-\tau)$, which uses the current output as well as an artificially delayed version of the output - similar as in proportional-integral-retarded feedback (Ramírez et al., 2016). To fit the framework given in (1) and (2) we introduce the following extended output

$$y^e(t) = \begin{bmatrix}C\\ 0\end{bmatrix} x(t) + \begin{bmatrix}0\\ C\end{bmatrix} x(t-\tau) \quad (7)$$

with $C$ the output matrix in (6). The aforementioned feedback law now reduces to the static output-feedback law $u(t) = [D_0^c \;\; D_1^c]\, y^e(t)$. To represent system (5) with (7) as output in `TDS-CONTROL`, we use the `tds_create` function. This function takes the system matrices and delays as input arguments and returns an object of the class `tds_ss`. For example, if $k = 1000\,\mathrm{N\cdot m^{-1}}$, $l = 0.4\,\mathrm{m}$, $g = 9.8\,\mathrm{m\cdot s^{-2}}$, $m = 0.1\,\mathrm{kg}$, $M = 1\,\mathrm{kg}$, and $\tau = 0.1\,\mathrm{s}$ the considered system can be represented as follows:

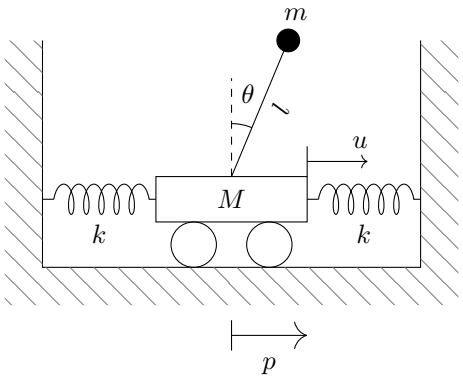



Fig. 1. Inverted pendulum set–up in which the cart is connected to two walls using two identical springs.

```
k = 1000; l = 0.4; g = 9.8; m = 0.1;
M = 1; tau = 0.1;
A = [0 1 0 0;-2*k/M 0 -m*g/M 0;0 0 0 1;
       2*k/(M*l) 0 (m+M)*g/(M*l) 0];
B = [0;1/M;0;-1/(M*l)];
C = [1 0 0 0;0 0 1 0];
sys=tds_create({A},{B},{[C; zeros(size(
   C))] [zeros(size(C)); C]},[0 tau]);
```

To derive a stabilizing static output-feedback controller, we use the `tds_stabopt_static` function:

```
[Dc,cl_sys] = tds_stabopt_static(sys);
```

This function returns the resulting feedback matrix in `Dc`, while the optional output argument `cl_tds` contains a representation for the resulting closed-loop of form (3).

Recall that the obtained controller depends on the initial values in the optimization problem and several starting points may need to be sampled to find a stabilizing controller. The optional argument `options` can be used to specify the number of starting points:

```
options = struct('nstart',5);
```

Now the controller which results in the smallest spectral abscissa over five runs of the optimisation procedure starting from different random initial controllers will be returned. For sake of reproducibility, it is also possible to set the initial feedback matrices, such that the same controller will be found over different runs of `tds_stabopt_static`.

```
tds_stabopt_static(sys,'options',
   options,'initial',zeros(1,4))
```

Starting the optimisation procedure from one starting point (here a zero feedback matrix), we obtain the following feedback matrix (rounded to two significant digits)

$$D_c = 10^3 \times [\underbrace{8.11 \;\; 4.99}_{D_0^c} \;\; \underbrace{-5.71 \;\; -2.48}_{D_1^c}]. \quad (8)$$

For the considered set-up the characteristic function associated with the closed loop reduces to

$$\det\left(\lambda I - A - BD_0^c C - BD_1^c C e^{-\lambda\tau}\right). \quad (9)$$

To compute the roots of this function, we can use the `tds_roots` function:

```
cl_roots = tds_roots(cl_sys,-80);
```

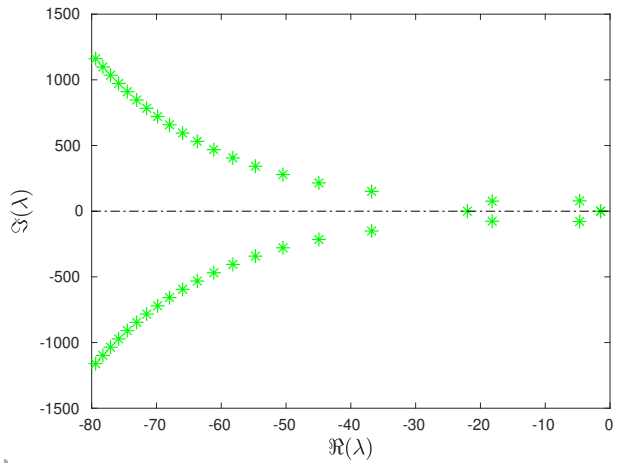



Fig. 2. The roots of (9) with $D_0^c$ and $D_1^c$ as in (8) in the right half-plane $\{\lambda \in \mathbb{C} : \Re(\lambda) \geq -80\}$.

with the first argument, `cl_sys`, the aforementioned object that represents the closed-loop system. The second arguments indicates the right half-plane in which we want to compute the roots. In this case we consider the right half-plane $\{\lambda \in \mathbb{C} : \Re(\lambda) \geq -80\}$. Fig. 2 plots these roots (this figure is obtained using the `tds_eigenplot` function). We observe that all roots lie in the open left half-plane meaning that the resulting closed-loop system is stable and the spectral abscissa is equal to -1.4059 (rounded to five significant digits).

Similarly, using `tds_stabopt_dynamic`, one can design a dynamic controller for the system. Below we design a dynamic feedback controller of order one:

```
dynC = tds_stabopt_dynamic(sys,1);
```

in which the second argument specifies the order of the controller and `dynC` is a state-space system of the form (2). Starting from $A_c = -1$, $B_c = [1\ 1\ 1\ 1]$, $C_c = 1$ and $D_c$ as in (8), the spectral abscissa now reduces to -5.1002, which is to be expected as we now have more free control parameters to tune.

## 3. ROBUST STABILIZATION

In the previous section we assumed that all parameters in the model are exactly known. In practice however, these parameters can only be measured up to some limited accuracy. To deal with such uncertainties one typically integrates these in the model (Hinrichsen and Pritchard, 2006). `TDS-CONTROL` can handle structured and real-valued uncertainties, which allows to target certain entries in the system matrices since only certain parameters are affected typically. For example, in (5) only the entries on positions $(2,1)$, $(2,3)$, $(4,1)$, and $(4,3)$ of the state matrix $A$ can be affected by parameter uncertainties, while the other entries can not change without violating the physical model.

More specifically, the package focuses on uncertain systems of the form

$$\begin{cases} \dot{x}(t) = \sum_{i=0}^{m} \tilde{A}_i(\delta) x(t - \tilde{\tau}_i(\delta\tau)) + \sum_{i=0}^{m} \tilde{B}_i(\delta) u(t - \tilde{\tau}_i(\delta\tau)) \\ y(t) = \sum_{i=0}^{m} \tilde{C}_i(\delta) x(t - \tilde{\tau}_i(\delta\tau)) \end{cases} \tag{10}$$

with $\delta = \{\delta_1, \ldots, \delta_L\} \in \mathbb{R}^{q_1 \times r_1} \times \cdots \times \mathbb{R}^{q_L \times r_L}$ a set of matrix uncertainties and $\delta\tau = \{\delta\tau_1, \ldots, \delta\tau_K\} \in \mathbb{R}^K$ a set containing the delay uncertainties. Note that the same uncertainty can affect multiple matrices/delays simultaneously. The matrix uncertainties affect the system matrices in a linear way: for any matrix $R$, $\tilde{R}(\delta)$ can be written as follows:

$$\tilde{R}(\delta) = R + \sum_{j=1}^{S^R} G_j^R \delta_{l_j^R} H_j^R$$

with $R$ the nominal value of the matrix, $S^R$ the number of uncertainty terms, $l_j^R$ an integer between 1 and $L$ indicating the matrix uncertainty, and $G_j^R$ and $H_j^R$ real-valued shape matrices that allow to target specific blocks in the matrix. This formulation allows the same uncertainty to affect the same system matrix using different shape matrices, which is, for example, necessary to model repeated block diagonal uncertainties. The delay uncertainties target the delays in the same way:

$$\tilde{\tau}_i = \tau_i + \sum_{j=1}^{K} w_j^i \delta\tau_j$$

with $w_j^i$ a real-valued constant. To quantify the size of these uncertainties we introduce the following measure

$$\|(\delta, \delta\tau)\| = \left\| \begin{matrix} v_1 \|\delta_1\|_F \\ \vdots \\ v_L \|\delta_L\|_F \\ z_1 \ |\delta\tau_1| \\ \vdots \\ z_K |\delta\tau_K| \end{matrix} \right\|_\infty \tag{11}$$

in which $v_1, \ldots, v_L$ and $z_1, \ldots, z_K$ are strictly positive constants, $\|\cdot\|_F$ is the Frobenius norm and $\|\cdot\|_\infty$ is the infinity norm, i.e., the maximal value of the vector.

By considering a bound $\epsilon \geq 0$ on the size of the uncertainty, we can introduce the pseudo-spectral abscissa of a system as the maximal spectral abscissa over all uncertainty values that are smaller than $\epsilon$:

$$\alpha_{\text{ps}}(\epsilon, S_{cl}) = \max \left\{ \alpha\big(\tilde{S}_{cl}(\delta, \delta\tau)\big) : \|(\delta, \delta\tau)\| \leq \epsilon \right\},$$

with $\tilde{S}_{cl}(\delta, \delta\tau)$ the closed-loop system of the form (3) with the system matrices perturbed by $\delta$ and the delays perturbed by $\delta\tau$. For a given $\epsilon$, a negative pseudo-spectral abscissa thus gives the worst-case exponential decay rate of the closed-loop system over all perturbations smaller than $\epsilon$. On the other hand, the distance to instability is defined as the smallest $\epsilon$ for which the pseudo-spectral abscissa becomes positive:

$$d_{\text{ins}}(S_{cl}) = \min\{\epsilon \geq 0 : \alpha_{\text{ps}}(\epsilon) \geq 0\},$$

i.e., for all perturbations with size strictly smaller than $d_{\text{ins}}(S_{cl})$ the closed-loop system remains stable and there exists a perturbation of size $d_{\text{ins}}(S_{cl})$ that renders the nominal closed loop unstable.

Let us illustrate the concepts above with the example from the previous section. Assume that there is a matrix

uncertainty $\delta_1 \in \mathbb{R}^{2\times 2}$ that affects the matrix $A$ in the following way

$$A + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \delta_{1,11} & \delta_{1,12} \\ \delta_{1,21} & \delta_{1,22} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

and a delay uncertainty $\delta\tau_1$ that affects the artificially introduced delay $\tau$, with $v_1 = 1$ and $z_1 = 500$. To create a TDS-CONTROL representation for the matrix uncertainty, we use the function `matrix_uncertainty`.

```
munc = [matrix_uncertainty('rf','frob',
          1,2,2)];
```

The first argument of this function indicates the type of the perturbation; The second argument indicates the matrix norm that is used in (11); The third argument is equal to $1/v_l$ with $v_l$ as defined in (11); The final two argument specify the dimensions of the uncertainty. Similarly, the delay uncertainty can be represented with the `delay_uncertainty` function that takes as argument $1/z_k$ with $z_k$ as defined in (11).

```
dunc = [delay_uncertainty(0.002)];
```

Finally, `utds_struct` and `utds_ss` allow to store how these uncertainty affect the open-loop system.

```
ustr = utds_strct('A',{struct('l',1,'G'
    ,G,'H',H)},'hC1',{struct([]),struct(
    'l',1,'w',1)});
usys = utds_ss(sys,munc,dunc,ustr);
```

To design a static output-feedback controller that minimizes the pseudo-spectral abscissa with respect to the control parameters, we can use the `utds_psaopt_static` function. For $\epsilon$ equal to 5, the code becomes as follows:

```
epsi = 5;
Dc = utds_psaopt_static(usys,epsi);
```

Starting from $D_c$ as in (8) we obtain the feedback matrix

$$D_c = 10^3 \times [9.04\ 5.09\ -5.09\ -2.26].$$

To compute the pseudo-spectral abscissa and the distance to instability of the resulting closed-loop system we use the functions `utds_psa` and `utds_dins`, respectively. We find that for $\epsilon = 5$ the pseudo-spectral abscissa is equal to -0.2870 while the distance to instability is equal to 5.168.

## 4. OTHER FUNCTIONALITY

Besides designing (robustly) stabilizing controllers, the presented software package can also be used to design controllers that minimize performance measures such as the $\mathcal{H}_\infty$-norm, which can be computed using the function `tds_hinorm`. Similarly as for (robust) stabilization, a controller that minimizes this quantity can be designed using the `tds_hiopt` function.

It is also possible to impose structure on the designed controllers in TDS-CONTROL, i.e., only certain elements of the matrices in (2) are allowed to changed. By imposing such a structure, one can design decentralized, overlapping and PID controllers (Dileep et al., 2020). Let us illustrate this by designing a stabilizing PI controller for the example problem from Section 2. Notice that such a PI controller can be written in the form of (2)

$$\begin{cases} \dot{x}_c(t) &= K_I y(t) \\ u(t) &= x_c(t) + K_P y(t) \end{cases}$$

in which only the elements of $B_c$ and $D_c$ are allowed to change, while $A_c$ and $C_c$ should be fixed to the zero and identity matrix, respectively. To impose this structure in the software package we use the `basis` and `mask` options

```
basis = tds_create({0},{zeros(1,4)
    },{1},{zeros(1,4)});
mask = tds_create({false},{true(1,4)},{
    false},{true(1,4)});
[PI,cl_PI] = tds_stabopt_dynamic(tds,1,
    'basis',basis,'mask',mask);
Ki = PI.B; Kp = PI.D;
```

The option `basis` allows to specify the fixed values of $A_c$ and $C_c$, while the option `mask` is used to indicated that only the elements of $B_c$ and $D_c$ may vary. Starting the optimization routine from $K_I$ and $K_P$ both equal to zero, we obtain the feedback gains

$$K_P = 10^3 \times [4.49\ 3.88\ 0.09\ -0.39]$$

and

$$K_I = 10^3 \times [1.16\ 5.90\ 1.02\ 5.52].$$

The spectral abscissa of the corresponding closed loop system is equal to $-2.1085$.

## 5. COMMENTS ON NEUTRAL BEHAVIOR

Up until now, we have limited ourselves to time-delay systems which are retarded. However TDS-CONTROL can also deal with neutral(-like) systems. For such systems, the property that any right half-plane only contains finitely many characteristic roots no longer holds. More specifically, for such systems the spectrum, i.e., the set of the characteristic roots, may include sequences for which the imaginary part approaches infinity while the real part goes to a bounded value (Michiels and Niculescu, 2014, chs. 1.2 and 1.3). Furthermore, the location of these vertical chains might be sensitive to infinitesimal perturbations on the delays. As a consequence, also the stability of the corresponding system might be sensitive to arbitrary small delay changes. TDS-CONTROL therefore aims to design closed-loop systems that are strongly stable (Hale and Lunel, 2002), which guarantees that stability is preserved for sufficiently small delay perturbations, by minimizing the so-called strong spectral abscissa.

To illustrate the sensitivity of the characteristic roots with respect to an infinitesimal perturbation on the delay vector, we will consider the following neutral delay-differential equation

$$\dot{x}(t) - \frac{3}{4}\dot{x}(t-1) + \frac{1}{2}\dot{x}(t-2) = \frac{1}{4}x(t) - \frac{1}{3}x(t-1). \quad (12)$$

To represent this differential equation in TDS-CONTROL, we use the following code:

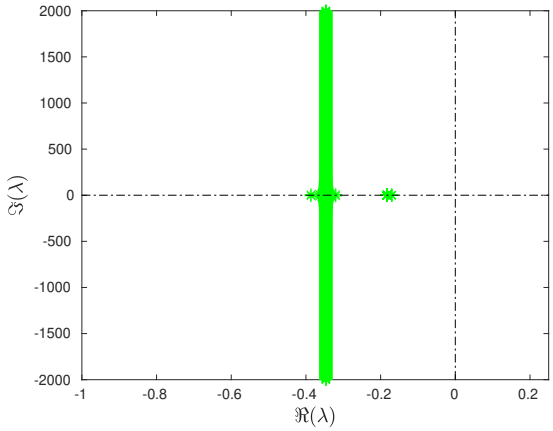



Fig. 3. The characteristic roots of (12) in the region $[-1, 0.25] \times \jmath[-2000, 2000]$.

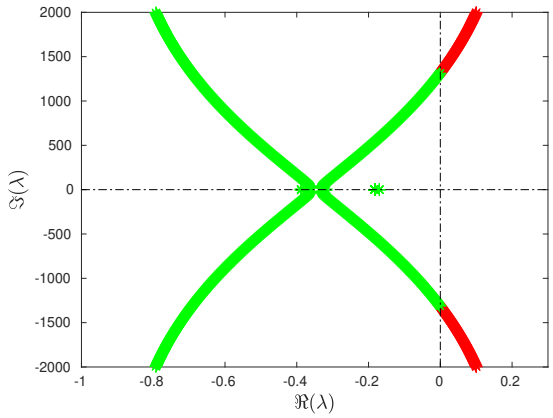



Fig. 4. The characteristic roots of (12) in which the second delay is replaced by 2.001 in the region $[-1, 0.25] \times \jmath[-2000, 2000]$.

```
sys = tds_create_neutral({-0.75,0.5},
    [1  2],{0.25,  -1/3},[0  1]);
```

Next we use `tds_roots` to compute the roots of the associated characteristic function and plot these roots in Fig. 3. We observe that the null solution of (12) is exponentially stable as all roots lie in the open left half-plane bounded away from the imaginary axis. However, if we replace the second delay by 2.001, the spectrum looks like Fig. 4 and we no longer have stability. Indeed, it can be shown that stability of (12) is fragile with respect to the delays, in the sense that for any given bound, no matter how small, there exists a change on the delays smaller than this bound that destabilizes the system.

## 6. CONCLUSION

In this paper we have illustrated the core features of the software packet `TDS-CONTROL`, which allows to analyze and design feedback controllers for a broad range of LTI delay systems under various performance measures. The software code is licensed under the GNU GPL version 3 and is accompanied by a complete documentation.

## REFERENCES


Apkarian, P. and Noll, D. (2006). Nonsmooth H$\infty$ synthesis. *IEEE Transactions on Automatic Control*, 51(1), 71–86.

Breda, D., Maset, S., and Vermiglio, R. (2006). Pseudospectral approximation of eigenvalues of derivative operators with non-local boundary conditions. *Applied numerical mathematics*, 56(3-4), 318–331.

Dileep, D., Borgioli, F., Hetel, L., Richard, J.P., and Michiels, W. (2018). A scalable design method for stabilising decentralised controllers for networks of delay-coupled systems. *IFAC-PapersOnLine*, 51(33), 68–73. 5th IFAC Conference on Analysis and Control of Chaotic Systems CHAOS 2018.

Dileep, D., Van Parys, R., Pipeleers, G., Hetel, L., Richard, J.P., and Michiels, W. (2020). Design of robust decentralised controllers for MIMO plants with delays through network structure exploitation. *International Journal of Control*, 93(10), 2275–2289.

Gumussoy, S., Henrion, D., Millstone, M., and Overton, M.L. (2009). Multiobjective robust control with HIFOO 2.0. *IFAC Proceedings Volumes*, 42(6), 144–149.

Gumussoy, S. and Michiels, W. (2011). Fixed-order H-infinity control for interconnected systems using delay differential algebraic equations. *SIAM Journal on Control and Optimization*, 49(5), 2212–2238.

Hale, J.K. and Lunel, S.M.V. (2002). Strong stabilization of neutral functional differential equations. *IMA Journal of Mathematical Control and Information*, 19(1,2), 5–23.

Hinrichsen, D. and Pritchard, A.J. (2006). *Mathematical Systems Theory I: Modelling, State Space Analysis, Stability and Robustness*, volume 48. Springer Science & Business Media.

Insperger, T. and Stépán, G. (2000). Stability of the milling process. *Periodica Polytechnica Mechanical Engineering*, 44(1), 47–57.

Lewis, A.S. and Overton, M.L. (2013). Nonsmooth optimization via quasi-newton methods. *Mathematical Programming*, 141(1), 135–163. Code available from `https://cs.nyu.edu/~overton/software/hanso/`.

Michiels, W. and Niculescu, S.I. (2014). *Stability, control, and computation for time-delay systems. An eigenvalue based approach*. SIAM, Philadelphia (PA), 2nd edition.

Ramírez, A., Mondié, S., Garrido, R., and Sipahi, R. (2016). Design of proportional-integral-retarded (PIR) controllers for second-order LTI systems. *IEEE Transactions on Automatic Control*, 61(6), 1688–1693.

Siqueira, D., Moreira, F.J.O., and Paglione, P. (2012). Design of robust static output feedback flight control law using structured singular value optimization. *Proceedings of the 15th IFAC Workshop on Control Applications of Optimization*.

Vanbiervliet, J., Verheyden, K., Michiels, W., and Vandewalle, S. (2008). A nonsmooth optimisation approach for the stabilisation of time-delay systems. *ESAIM: Control, Optimisation and Calculus of Variations*, 14(3), 478–493.

Wu, Z. and Michiels, W. (2012). Reliably computing all characteristic roots of delay differential equations in a given right half plane using a spectral method. *Journal of Computational and Applied Mathematics*, 236(9), 2499–2514.