

Root Locus for SISO Dead-Time Systems: A Continuation Based Approach

Suat Gumussoy^a, Wim Michiels^a

^a*Department of Computer Science, K. U. Leuven,
Celestijnenlaan 200A, 3001, Heverlee, Belgium
(e-mails: {suat.gumussoy, wim.michiels}@cs.kuleuven.be).*

Abstract

We present a numerical method to plot the root locus of Single-Input-Single-Output (SISO) dead-time systems with respect to the controller gain or the system delay. We compute the trajectories of characteristic roots of the closed-loop system on a prescribed complex right half-plane. We calculate the starting, branch and boundary crossing roots of root-locus branches inside the region. We compute the root locus of each characteristic root based on a predictor-corrector type continuation method. To avoid the high sensitivity of roots with respect to the locus parameter in the neighborhood of branch points, the continuation method relies on a natural parameterization of the root-locus trajectory in terms of a distance in the (characteristic root, locus parameter)-space. The method is numerically stable for high order SISO dead-time systems.

Key words: Root locus, dead time, time delay, SISO, characteristic root, stability analysis, relative stability.

1 Introduction

The root locus method is an essential tool in modern control engineering for analysis and synthesis problems [15]. This method is successfully implemented for finite dimensional SISO systems and becomes a fundamental tool in control education [6,10].

The closed-loop of the SISO system with a time delay generically has infinitely many poles in the complex plane (see, e.g., [13]). Therefore obtaining the root locus for dead-time systems is a difficult problem. Unlike the finite dimensional case, the root-locus equation contains a time delay term and standard polynomial root-finding algorithms for the root computation are not applicable.

The root locus of SISO dead-time systems with respect to the *controller gain* is based on two main approaches: methods based on sweeping or gridding parameters on the complex plane and continuation based methods.

The methods in the first group are early methods in the literature. The root locus is obtained on a set of vertical lines [9], horizontal lines [18] or on a rectangular grid [11] in the complex plane by finding points satisfying the root-locus equation. These methods require a large number of grid points for an accurate root-locus plot and may miss the dynamics of some characteristic roots due

to the finite number of grid or sweep parameters.

The methods in the second group are continuation based methods. These methods follow characteristic roots of the root locus by a predictor-corrector algorithm inside the root-locus region and detect the ones entering into the region due to the asymptotic root chains of time delay systems. The predicted characteristic root is obtained by computing the slope of the phase equation of the root-locus equation [2], the solution of nonlinear differential equations [17] or by a triangulation method on the complex plane [14] and the predicted values are corrected by a Newton-Raphson iteration. The characteristic roots entering into the root-locus region are detected by checking the sign of function values on the constant grid points of the region's boundary [2] or computing *another* root locus problem [14,17]. The approach proposed in [2] has two main limitations. First, the root locus is computed on a rectangular region and this is restrictive to observe the overall dynamics of characteristic roots. Second, it requires exhaustive search on a grid for characteristic roots entering into the root-locus region, with a possibility of missing a characteristic root depending on the number of grid points. The articles [14,17] consider general SISO time delay systems, i.e., SISO systems with state delays. The predictor step in both methods, the solution of nonlinear differential equations and the triangulation method, are numerically expensive com-

pared to the simple linear predictor. In order to detect asymptotic roots another root locus problem has to be solved and the number of required roots for a complex region is not given.

The root locus of SISO dead-time systems with respect to the *time delay* allows us to analyze the effects of time delay on system stability and performances and it is less considered in the literature. A computationally expensive brute force approach is to compute the locations of the characteristic roots inside a desired region for a grid over the parameter space using, e.g., a spectral method [4]. A continuation based method in [7] obtains the root locus on the *complex right half-plane* by solving a nonlinear differential equation and correcting with Newton's method. The predictor step is computationally expensive and the method does not allow to analyze the root locus of stable closed-loop systems since the boundary of the root-locus region is the imaginary axis.

All continuation based methods for the root locus of SISO dead-time systems in the existing literature parameterize root-locus trajectories with respect to the controller gain or the time delay. This parameterization is numerically ill-posed due to the high sensitivity in the neighborhood of intersection points characterized by the presence of multiple roots, as reported in [7]. Both the predictor and corrector steps in all methods require evaluations of functions with time delays. These evaluations are numerically expensive and not numerically stable due to the oscillation and exponential increase of time delay terms in the imaginary axis and the positive real axis directions in the complex plane respectively.

In this article, we compute the root locus of SISO dead-time systems on a *given* complex right half-plane up to a predefined controller gain or a time delay. Our main contributions are the following:

- We compute the trajectories of *all characteristic roots* entering into the prescribed region with respect to the controller gain or the time delay up to an upper bound value. By choosing the controller gain or time delay sufficiently large, the asymptotic behavior of the roots can be observed.
- Our continuation method estimates the next root on the complex plane by a simple linear predictor and corrects this prediction with Newton's method. We parameterize the root-locus trajectories in the (root and gain/delay)-space in terms of the arclength which is numerically robust, as we shall see.
- We use an *adaptive step size* in the prediction step depending on the convergence rate of Newton's method and the distance of the root from the root-locus trajectory. This makes our algorithm scalable since it uses different step sizes for different root-locus trajectories and adapts to the curvature of the trajectories.
- We transform the root-locus equation into phase and logarithmic magnitude equations and avoid the oscil-

lation and exponential increase problems due to the time delay term in function evaluations, which are only needed in the correction step.

The paper is organized as follows. Section 2 formulates the root-locus problem with respect to the controller gain and the time delay. The critical points of the root locus are computed in Section 3. The predictor-corrector based continuation method is given in Section 4. The overall algorithm for the root locus is presented in Section 5. Section 6 is devoted to numerical examples. In Section 7, some concluding remarks are presented.

Notation:

- $\mathbb{C}, \mathbb{R}, \mathbb{Z}$: sets of complex, real and integer numbers,
 $\Re(u)$: real part of a complex number u ,
 $\Im(u)$: imaginary part of a complex number u ,
 $|u|, \angle u$: magnitude, phase of a complex number u ,
 $\lceil u \rceil$: the smallest integer larger than or equal to u ,
 $\lfloor u \rfloor$: the largest integer smaller than or equal to u ,
 u^T : the transpose of the vector u ,
 A^+ : the pseudoinverse of the matrix A ,
 $\text{sgn}(u)$: returns $+1, -1, 0$ given a real number u
for $u > 0, u < 0, u = 0$ respectively,
 $\mathbb{N}_{n_1}^{n_2}$: the set of integers from n_1 to n_2 .
 $\text{mod}(u, v)$: returns the remainder on division of the dividend u by the divisor v .

2 Problem Formulation

A SISO dead-time system is a rational, proper SISO plant G with a constant input or output time delay h . The only required input data to compute the root locus of these systems with respect to the controller gain or the time delay are

- the poles of G , $p_i \in \mathbb{C}$ for $i \in \mathbb{N}_1^n$,
- the zeros of G , $z_r \in \mathbb{C}$ for $r \in \mathbb{N}_1^m$,
- the gain of G , $\alpha \in \mathbb{R}$,
- the time delay h , $h \in \mathbb{R}$ and $h > 0$.

Note that the input data can be obtained from state-space matrices or the transfer function of G .

Let $G(s)e^{-hs}$ be the transfer function representation of the SISO dead-time system where

$$G(s) = \alpha \frac{(s - z_1)(s - z_2) \dots (s - z_m)}{(s - p_1)(s - p_2) \dots (s - p_n)}, \quad n \geq m \quad (1)$$

and the real and imaginary parts of system zeros and poles are defined as $z_r = \sigma_{zr} + j\omega_{zr}$ for $r \in \mathbb{N}_1^m$ and $p_i = \sigma_{pi} + j\omega_{pi}$ for $i \in \mathbb{N}_1^n$.

The *root-locus equation* of a SISO dead-time system with respect to the controller gain or the time delay is

$$f(s, \lambda) = 0 \quad \text{where} \quad f(s, \lambda) = \begin{cases} 1 + \lambda G(s)e^{-hs} & (2) \\ 1 + G(s)e^{-\lambda s} & (3) \end{cases}$$

and λ is the *locus parameter* for $\lambda \in [0, \lambda_{\max}]$ where λ_{\max} is a given positive number. The *root-locus region* is a complex right half-plane

$$\mathbb{C}_{\sigma_0} = \{s \in \mathbb{C} : \Re(s) \geq \sigma_0\} \quad (4)$$

where σ_0 is a negative real number. The corresponding *boundary* of the root-locus region is a vertical line through $s = \sigma_0$ and parallel to the imaginary axis.

We define root-locus problems with respect to the controller gain or the time delay as follows.

Root locus problems: Given $\lambda_{\max} > 0$, compute the locations of characteristic roots of root-locus equations in (2,3) inside the root-locus region \mathbb{C}_{σ_0} for $\lambda \in [0, \lambda_{\max}]$.

The *root-locus trajectory* of the characteristic root is a curve on the complex plane on which each point satisfies the root-locus equation for $\lambda \in [0, \lambda_{\max}]$. The intersection of two or more root-locus trajectories is a *branch point*. The *starting points* of the root locus are the characteristic roots inside \mathbb{C}_{σ_0} for $\lambda = 0$ and the *boundary crossing roots* are the characteristic roots crossing $\Re(s) = \sigma_0$ for any non-negative λ value smaller than λ_{\max} . The *critical points* are the starting points, branch points and boundary crossing roots of the root locus.

For the root locus w.r.t. the controller gain, we can compute branch points inside \mathbb{C}_{σ_0} a priori as we shall see. For the root locus w.r.t. the time delay, we detect the branch points while following the root-locus trajectory. The behavior of a root-locus trajectory around a branch point is given in the following Lemma [17].

Lemma 1 *If \tilde{s} is a root of the root-locus equation in (2) or (3) with a locus parameter $\tilde{\lambda}$ satisfying*

$$\left. \frac{\partial^l f(s, \tilde{\lambda})}{\partial s^l} \right|_{s=\tilde{s}} = 0, \quad l \in \mathbb{N}_1^{N-1} \quad \text{and} \quad \left. \frac{\partial^N f(s, \tilde{\lambda})}{\partial s^N} \right|_{s=\tilde{s}} \neq 0,$$

then N root-locus trajectories intersect at $s = \tilde{s}$. Intersecting trajectories continue straight after the intersection if N is odd and make an angle of $-\frac{\pi}{N}$ angle if N is even.

When a single root s of the root-locus equation crosses the boundary $\Re(s) = \sigma_0$, we determine whether it enters into or leaves the region \mathbb{C}_{σ_0} by computing its *crossing direction* [13] defined as

$$\mathcal{CD}(s, \lambda) := \text{sgn} \left(\Re \left(-\frac{\partial f}{\partial \lambda} \bigg|_{f(s, \lambda)=0} \right) \right). \quad (5)$$

A root on the boundary enters into or leaves the region \mathbb{C}_{σ_0} depending on whether $\mathcal{CD}(s, \lambda) > 0$ or $\mathcal{CD}(s, \lambda) < 0$ respectively.

The computation of root-locus trajectories involves two main tasks: computing the critical points and following the root-locus trajectories. The next section focuses on the computation of the critical points of the root locus w.r.t. the controller gain and the time delay. We follow each root trajectory using a continuation method described in Section 4.

3 Computation of Critical Points

We compute the critical points of two root-locus problems in the next two subsections.

3.1 Controller gain as locus parameter

The root-locus equation w.r.t. the controller gain is given in (2). The starting points of the root locus are the poles of G inside the root-locus region \mathbb{C}_{σ_0} .

The branch points satisfy the root-locus equation and

$$\frac{\partial f(s, \lambda)}{\partial s} = \lambda(G'(s) - G(s)h)e^{-hs} = 0. \quad (6)$$

Thus the branch points are the zeros of the transfer function $G'(s) - G(s)h$ inside the region \mathbb{C}_{σ_0} . This transfer function can be written as

$$\frac{G'(s)}{G(s)} - h = \sum_{r=1}^m \frac{1}{s - z_r} - \sum_{i=1}^m \frac{1}{s - p_i} - h$$

and realized in a state-space representation as a series connection of $m + n$ first-order systems using the poles and zeros of G and the time delay h . Its zeros can be computed by solving an eigenvalue problem constructed from state-space matrices of G . Consequently, the branch points can be determined accurately.

The computation of boundary crossing roots of the root-locus equation in (2) is a difficult problem. These roots and their crossing directions are computed in the following section.

3.1.1 Computation of boundary crossing roots

A root s on the boundary of the root-locus region $\Re(s) = \sigma_0$ for the controller gain λ satisfies the magnitude and phase equations of the root-locus equation in (2). We first find the intervals on the boundary where the magnitude condition holds for some $\lambda \in [0, \lambda_{\max}]$. This is equivalent to finding the intervals on $\Re(s) = \sigma_0$ and $\omega \in [0, \infty)$ such that

$$\Lambda(\omega) := h\sigma_0 - \ln |G(\sigma_0 + j\omega)| \leq \ln \lambda_{\max}. \quad (7)$$

Lemma 2 Assume that G has neither poles nor zeros on the boundary of the root-locus region. The functions $\omega \mapsto \Lambda(\omega)$ and $\omega \mapsto \Lambda'(\omega)$ are continuous and the non-negative zeros of $\Lambda'(\omega)$ are the non-negative real roots of the polynomial

$$\Gamma_z \sum_{i=1}^n \Delta\omega_{pi} \Gamma_p^i - \Gamma_p \sum_{r=1}^m \Delta\omega_{zr} \Gamma_z^r \quad (8)$$

where $\Delta\sigma_{zr} = (\sigma_0 - \sigma_{zr})$, $\Delta\omega_{zr} = (\omega - \omega_{zr})$, $\gamma_{zr}(\omega) = \Delta\sigma_{zr}^2 + \Delta\omega_{zr}^2$, $\Gamma_z^r = \prod_{r_1=1}^m \gamma_{zr}(\omega)$ for $r \in \mathbb{N}_1^m$, $\Delta\sigma_{pi} = (\sigma_0 - \sigma_{pi})$, $\Delta\omega_{pi} = (\omega - \omega_{pi})$, $\gamma_{pi}(\omega) = \Delta\sigma_{pi}^2 + \Delta\omega_{pi}^2$, $\Gamma_p^i = \prod_{i_1=1}^n \gamma_{pi}(\omega)$ for $i \in \mathbb{N}_1^n$, $\Gamma_z = \prod_{r=1}^m \gamma_{zr}(\omega)$, $\Gamma_p = \prod_{i=1}^n \gamma_{pi}(\omega)$.

Proof. Using the transfer function of G in (1), the function $\Lambda(\omega)$ can be written as

$$\Lambda(\omega) = h\sigma_0 - \ln|\alpha| + \frac{1}{2} \left(\sum_{i=1}^n \ln \gamma_{pi}(\omega) - \sum_{r=1}^m \ln \gamma_{zr}(\omega) \right). \quad (9)$$

The first derivative of the function $\Lambda(\omega)$ is

$$\Lambda'(\omega) = \sum_{i=1}^n \frac{\Delta\omega_{pi}}{\gamma_{pi}(\omega)} - \sum_{r=1}^m \frac{\Delta\omega_{zr}}{\gamma_{zr}(\omega)}. \quad (10)$$

The functions $\Lambda(\omega)$ and $\Lambda'(\omega)$ are continuous except the points where $\gamma_{zr}(\omega)$ or $\gamma_{pi}(\omega)$ are equal to zero. These points are the poles or zeros of G on $\Re(s) = \sigma_0$. The continuity results in Lemma 2 follow from the assumption. The polynomial given in (8) is the numerator of the function $\Lambda'(\omega)$ as in equation (10) and the result follows. \square

Corollary 3 Assume that G has neither poles nor zeros on the boundary of the root-locus region. Then the function $\omega \mapsto \Lambda(\omega)$ is monotonic on the intervals whose boundary points are consecutive non-negative zeros of $\Lambda'(\omega)$, 0 and ∞ .

Proof. By Lemma 2, the function $\Lambda(\omega)$ is continuous since σ_0 is chosen such that there are neither poles nor zeros of G on $\Re(s) = \sigma_0$. Therefore it is monotonic inside the intervals determined by its extremum points and the end points of the boundary of the root-locus region, 0 and ∞ . \square

Since $\Lambda(\omega)$ is monotonic on each interval defined in Corollary 3, we find the subinterval in each interval where $\Lambda(\omega)$ satisfies the inequality in (7). This is done as follows. If both values of $\Lambda(\omega)$ at the interval end points are smaller than $\ln \lambda_{\max}$, then all $\Lambda(\omega)$ values in this interval are smaller than $\ln \lambda_{\max}$ because $\Lambda(\omega)$ is monotonic. If one of the values of $\Lambda(\omega)$ at the interval end points is larger and the other one is smaller

than $\ln \lambda_{\max}$, we can find the point where $\Lambda(\omega)$ is equal to $\ln \lambda_{\max}$ by root-finding algorithms for a bracketed root of monotonic continuous function (such as Brent's method [5]) and take the subinterval satisfying the inequality in (7). If both values of $\Lambda(\omega)$ at the interval end points are larger than $\ln \lambda_{\max}$, we discard that interval since all values of $\Lambda(\omega)$ are larger than $\ln \lambda_{\max}$ and the inequality in (7) never holds.

Based on this approach, we can compute the set of intervals I on the boundary of the root-locus region where the magnitude condition (7) is satisfied for some values of $\lambda \in [0, \lambda_{\max}]$.

The boundary crossing roots also satisfy the phase equation of the root-locus equation in (2)

$$(2l+1)\pi = \phi(\omega) \text{ for } l \in \mathbb{Z} \quad (11)$$

over the intervals I on $\Re(s) = \sigma_0$. Here the function $\phi(\omega)$ represents the *continuous* extension of the phase of the transfer function $G(s)e^{-hs}$. It satisfies the following equation,

$$\text{mod} \left(\angle G(s)e^{-hs} \Big|_{s=\sigma_0+j\omega}, 2\pi \right) = \text{mod}(\phi(\omega), 2\pi)$$

where $\omega \in [0, \infty)$. The left-hand side of equation (11) represents constant functions of ω . If we partition the intervals I into the subintervals such that the function $\phi(\omega)$ is monotonic on each subinterval, we can compute the boundary crossing roots by root-finding algorithms for a bracketed root of monotonic continuous function [5]. The following results allow us to compute the intervals on $\Re(s) = \sigma_0$ where the function $\phi(\omega)$ is monotonic.

Lemma 4 Assume that G has neither poles nor zeros on the boundary of the root-locus region. Then the functions $\omega \mapsto \phi(\omega)$ and $\omega \mapsto \phi'(\omega)$ are continuous and the non-negative zeros of $\phi'(\omega)$ are the non-negative real roots of the polynomial

$$\Gamma_p \sum_{r=1}^m \Delta\sigma_{zr} \Gamma_z^r - \Gamma_z \sum_{i=1}^n \Delta\sigma_{pi} \Gamma_p^i - h\Gamma_z \Gamma_p \quad (12)$$

where the polynomials Γ_z^r for $r \in \mathbb{N}_1^m$, Γ_p^i for $i \in \mathbb{N}_1^n$, Γ_z and Γ_p are as defined in Lemma 2.

Proof. Using the transfer function of G in (1), the function $\phi(\omega)$ is written as

$$\phi(\omega) = \phi_1(\omega) + \phi_0 \quad (13)$$

where

$$\phi_1(\omega) := \sum_{r=1}^m \tan^{-1} \frac{\Delta\omega_{zr}}{\Delta\sigma_{zr}} - \sum_{i=1}^n \tan^{-1} \frac{\Delta\omega_{pi}}{\Delta\sigma_{pi}} - h\omega. \quad (14)$$

The offset difference ϕ_0 is defined as $\phi_0 = \angle G(\sigma_0) - \phi_1(0)$ which is equal to 0 or π .

The first derivative of the function $\phi(\omega)$ is

$$\phi'(\omega) = \sum_{r=1}^m \frac{\Delta\sigma_{zr}}{\gamma_{zr}(\omega)} - \sum_{i=1}^n \frac{\Delta\sigma_{pi}}{\gamma_{pi}(\omega)} - h. \quad (15)$$

Following the same arguments in Lemma 2, the functions $\phi(\omega)$ and $\phi'(\omega)$ are continuous by the assumption. The polynomial given in (12) is the numerator of the function $\phi'(\omega)$ as in equation (15) and the result follows. \square

Corollary 5 *Assume that G has neither poles nor zeros on the boundary of the root-locus region. Denote by I_ϕ the set of intervals whose end points are consecutive non-negative zeros of $\phi'(\omega)$, 0 and ∞ . Then the function $\omega \mapsto \phi(\omega)$ is monotonic on each interval in the set I_ϕ .*

Proof. The function $\phi(\omega)$ is continuous. The monotonicity of $\phi(\omega)$ changes only at the points where $\phi'(\omega) = 0$. The assertion follows. \square

The non-negative real roots of the polynomials (8) and (12) are zeros of the rational functions (10) and (15). These rational functions can be realized in a state-space representation as a series connection of second-order systems using the poles and zeros of G and the time delay h and the corresponding zeros can be computed by solving a generalized eigenvalue problem.

The intersection of two sets of intervals I and I_ϕ partitions I into the subintervals as $I = \cup_{i=1}^{n_I} I_i$ where $\phi(\omega)$ is monotonic on each interval I_i . Each intersection of the function $\phi(\omega)$ and the constant functions in the left-hand side of equation (11) over the intervals I corresponds to a boundary crossing root since any such point on I satisfies both the magnitude condition in (7) and the phase equation in (11) of the root-locus equation on $\Re(s) = \sigma_0$. Since the function $\phi(\omega)$ is monotonic on I_i , we can compute each intersection point by root-finding algorithms for a bracketed root of $\phi(\omega)$ over the interval I_i . The value of the ω at the intersection point is the imaginary part of the boundary crossing root on the interval I_i and the corresponding controller gain is the value of $e^{\Lambda(\omega)}$ for this point. If there is no horizontal line intersecting $\phi(\omega)$ on I_i , we discard this interval since there is no root crossing. Based on the above explanation, the following algorithm computes the boundary crossing roots of the root-locus equation in (2).

Algorithm 1

For each interval in $I_i = [\omega_i^L, \omega_i^R]$ of $I = \cup_{i=1}^{n_I} I_i$,

- (1) Compute the maximum and minimum values of the function ϕ over the interval I_i . Since the function is monotonic over the interval, they are the maximum

and minimum values of the function ϕ at the interval ends points, i.e., $\phi_i^{\max} = \max\{\phi(\omega_i^L), \phi(\omega_i^R)\}$ and $\phi_i^{\min} = \min\{\phi(\omega_i^L), \phi(\omega_i^R)\}$.

- (2) Consider the integers l for which the constant function $y = (2l+1)\pi$ has an intersection with the function $y = \phi(\omega)$ over the interval I_i . Note that the integers satisfy $l \in [l_i^{\min}, l_i^{\max}]$ where $l_i^{\min} = \left\lceil \frac{\phi_i^{\min}}{2\pi} - \frac{1}{2} \right\rceil$

and $l_i^{\max} = \left\lfloor \frac{\phi_i^{\max}}{2\pi} - \frac{1}{2} \right\rfloor$.

- (3) If $(l_i^{\min} > l_i^{\max})$

discard the interval I_i ,
else

for each integer l from l_i^{\min} to l_i^{\max}

- compute the intersection point between the horizontal line $(2l+1)\pi$ and the function ϕ over the interval I_i . This point can be considered as the bracketed root of a monotonic continuous function ϕ over I_i and accurately computed by root-finding algorithms such as Brent's method [5]. Denote this point by ω_{cr} which is equal to the imaginary part of the boundary crossing root.
- compute the corresponding controller gain for the boundary crossing root, i.e., $\lambda_{cr} = e^{\Lambda(\omega_{cr})}$.

By Algorithm 1, we compute all boundary crossing roots of the root-locus equation in (2) and their corresponding controller gain values for $\lambda \in [0, \lambda_{\max}]$. Their crossing directions are determined by the following theorem.

Theorem 6 *The crossing direction of a boundary crossing root $s_{cr} = \sigma_0 + j\omega_{cr}$ only depends on the imaginary part ω_{cr} on the boundary and is equal to*

$$\mathcal{CD}(s_{cr}, \lambda_{cr}) = -\text{sgn}(\phi'(\omega_{cr})). \quad (16)$$

Proof. Using the transfer function G in (1) and (10,15), we obtain

$$G'(s_{cr})G^{-1}(s_{cr}) - h = \phi'(\omega_{cr}) + j\Lambda'(\omega_{cr}). \quad (17)$$

By the crossing direction formula in (5) and the equation in (17), the crossing direction of s_{cr} at $\lambda = \lambda_{cr}$ is equal to

$$\begin{aligned} \mathcal{CD}(s_{cr}, \lambda_{cr}) &= \text{sgn} \left(\Re \left(\left(\lambda_{cr} \left(h - \frac{G'(s_{cr})}{G(s_{cr})} \right) \right)^{-1} \right) \right) \\ &= -\text{sgn}(\phi'(\omega_{cr})). \quad \square \end{aligned}$$

By Theorem 6, the crossing directions of the roots are the same when their imaginary parts are inside the same interval of I_ϕ (see Corollary 5). We determine the crossing directions of boundary crossing roots from their imaginary parts. We group them according to their crossing directions and define the sets $W^{in} = \{s_\nu^I, \lambda_\nu^I\}_{\nu=1}^{n_i}$ and $W^{out} = \{s_\nu^O, \lambda_\nu^O\}_{\nu=1}^{n_o}$ where $s_\nu^I = \sigma_0 + j\omega_\nu^I, \lambda_\nu^I$

for $\nu \in \mathbb{N}_1^{n_i}$ and $s_\nu^O = \sigma_0 + j\omega_\nu^O$, λ_ν^O for $\nu \in \mathbb{N}_1^{n_o}$ are the boundary crossing roots entering into or leaving the root-locus region and their controller gains respectively.

Remark 1: The crossing direction formula in (5) is well-posed (either +1 or -1) if there are neither poles nor zeros of G or branch points on the boundary of the root-locus region.

Remark 2: The closed-loop system is of *neutral* type (see, e.g., [13]) when G is biproper (i.e., $d := G(\infty) \neq 0$). Its asymptotic root chains lie outside the root-locus region \mathbb{C}_{σ_0} for $\lambda_{\max} < \frac{e^{h\sigma_0}}{|d|}$, while the region always has infinitely many roots for larger controller gains. Since it is not numerically possible to follow infinitely many roots, λ_{\max} is assumed to be smaller than $\frac{e^{h\sigma_0}}{|d|}$.

3.2 Time delay as locus parameter

The root-locus equation w.r.t. the time delay is given in (3). The starting points of the root locus are the zeros of the transfer function $1 + G(s)$ inside the root-locus region \mathbb{C}_{σ_0} .

The branch points satisfy the root-locus equation (3) and

$$\frac{\partial f(s, \lambda)}{\partial s} = (G'(s) - G(s)\lambda)e^{-\lambda s} = 0. \quad (18)$$

Since the locus parameter $\lambda = h$ in equation (18) is unknown, the direct computation of branch points is difficult compared to the previous case. Therefore we detect the branch points of the root locus while following the root-locus trajectory as described in Section 4.5.

The relative stability analysis of the closed-loop of SISO dead-time systems w.r.t. the stability boundary $\Re(s) = \sigma_0$ is given in [8]. This analysis is based on the computation of boundary crossing roots of the root-locus equation in (3) which is similar to the computation for the controller gain case in Section 3.1. Further information can be found in [8]. Note that when the plant is biproper, λ_{\max} is assumed to be smaller than $\max\left(0, \frac{\ln|d|}{|\sigma_0|}\right)$ to have finitely many characteristic roots inside the root-locus region \mathbb{C}_{σ_0} (see Remarks 2 and 3 in [8]).

4 Computing a Root-Locus Trajectory

We compute the root-locus trajectory of a characteristic root between two critical points. The trajectory satisfies the root-locus equation w.r.t the controller gain in (2) or the time delay in (3), represented by the locus parameter λ .

The starting points and the characteristic roots entering into the root-locus region are computed in Section 3. We follow each root-locus trajectory by a secant-predictor,

Newton-corrector continuation method [1]. In the prediction step, a line passing through the last two computed roots and locus parameters is used to estimate the next root and the next locus parameter at a certain distance (step length) in the (characteristic root, locus parameter)-space. This estimate is corrected using Newton's method in the correction step. The next iteration continues in a similar way, though the step length is adaptive. The branch points for the controller gain are computed apriori and those for the time delay are detected while following the trajectory. The new branch direction is determined by Lemma 1 and the trajectory is followed until the upper bound of the locus parameter λ_{\max} is reached.

4.1 Parameterization of the root-locus trajectory

Continuation based methods follow the characteristic root s and the locus parameter λ based on the parameterization of the root-locus trajectory. In the literature, the characteristic root is parameterized w.r.t. the locus parameter as $s = s(\lambda)$ for computational purposes [7,17]. In this parameterization, the characteristic roots are highly sensitive w.r.t. changes of the locus parameter in the neighborhood of branch points and relevant numerical problems are reported in [7]. We illustrate this sensitivity on the root locus by means of equation (2), where $h = \frac{4\pi}{3}$ and $G(s) = \frac{s^2}{(s^2+1)(s^2+1)}$.

Figure 1 shows the high sensitivity of the characteristic roots w.r.t. changes of λ at $\lambda = 0$ where a branch point occurs. As can be seen in Figure 1, this branch point appears as a turning point in the root versus locus-parameter space. When a continuation method tries to follow the trajectory 1 or 2, the turning point around $\lambda = 0$ causes several numerical problems if λ is used as a continuation parameter. Take for instance trajectory 1. First the sensitivity or derivative of the root w.r.t. λ tends to infinity when $\lambda \rightarrow 0^-$. Second when the λ passes zero the same branch cannot be followed anymore because the original trajectory turns back. In the best scenario a point on trajectory 2 will be computed but this is unlikely with a local method because trajectory 2 has another direction than trajectory 1 (see Lemma 1), hence the estimate of the root obtained from the previous points will be bad. On the other hand, trajectory 1 and trajectory 2 appear as smooth curves in the complex plane as shown in Figure 1. In the field of numerical bifurcation analysis (see, e.g., [16]), this was the motivation to parameterize the trajectories in terms of a natural notion of an arclength, i.e., a distance measured along the trajectories.

In our method we parameterize the root-locus trajectories in terms of the arclength γ measured along the trajectory in the combined (characteristic root s , locus parameter λ) space, i.e., $s = s(\gamma)$ and $\lambda = \lambda(\gamma)$ for $\gamma > 0$.

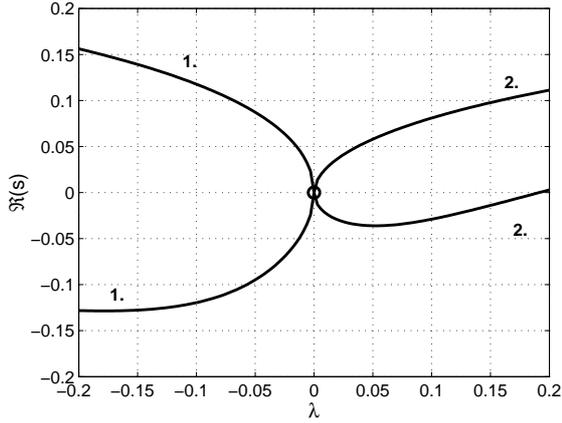


Fig. 1. The rightmost characteristic roots of the root-locus equation in (2) for the plant with $h = \frac{4\pi}{3}$ and $G(s) = \frac{s^2}{(s^2+1)(s^2+1)}$ as a function of the controller gain λ .

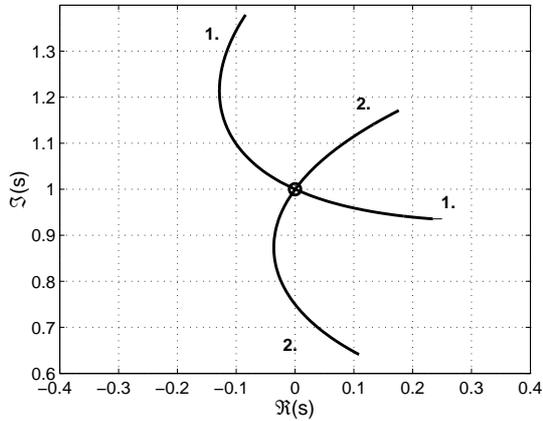


Fig. 2. The root-locus trajectories 1 and 2 in Figure 1 on the complex plane.

For the example given in (7) the two root-locus trajectories 1 and 2 are shown in Figure 3. The trajectories are smooth and using equal arclength steps for γ (where each step is shown as a point on the trajectory) the trajectories are traced without any numerical problem. Figure 3 also illustrates that a branch point is characterized by an extremum of the function $\gamma \mapsto \lambda(\gamma)$ which can be used to detect the presence of a branch point along the trajectory.

4.2 Prediction step

The predicted root and the locus parameter computation in the prediction step requires the previous root, the locus parameter, a direction and a step length. For each root-locus trajectory, the starting point s_0 is available. The direction of the prediction step d_i is computed as follows:

- The initial directions $d_0^* \in \mathbb{C}$ for the poles of G inside \mathbb{C}_{σ_0} and the boundary crossing roots are computed by the root-locus equation, i.e., by solving

$$\left[\frac{\partial f}{\partial s} \quad \frac{\partial f}{\partial \lambda} \right] \Big|_{f(s,\lambda)=0} d_0^* = 0 \quad (19)$$

where $d_0^* = [d_0^s \ d_0^\lambda]^T$ is normalized. Set the root-locus direction as $d_i = [\Re(d_0^s) \ \Im(d_0^s) \ d_0^\lambda]^T$. Note that (19) is the mathematical characterization of the *tangent* vector to the trajectory in the (s, λ) space [16].

- The directions in other iterations are computed using the real and imaginary parts of last two corrected roots and locus parameters, $y_i^c = [\sigma_i^c \ \omega_i^c \ \lambda_i^c]^T$ and $y_{i-1}^c = [\sigma_{i-1}^c \ \omega_{i-1}^c \ \lambda_{i-1}^c]^T$ by

$$d_i = \frac{y_i^c - y_{i-1}^c}{\|y_i^c - y_{i-1}^c\|} \text{ for } i \geq 1. \quad (20)$$

The real and imaginary parts of the predicted root and the locus parameter $y_{i+1}^p = [\sigma_{i+1}^p \ \omega_{i+1}^p \ \lambda_{i+1}^p]^T$ are computed using a line equation with a step length h_i

$$y_{i+1}^p = y_i^c + d_i h_i \text{ for } i \geq 0. \quad (21)$$

The geometric illustration of the prediction step is given in Figure 4 where the predicted point y_{i+1}^p is shown as a gray dot. The initial step length h_0 is fixed. The step lengths in other iterations are calculated adaptively based on previous values as we outline later on.

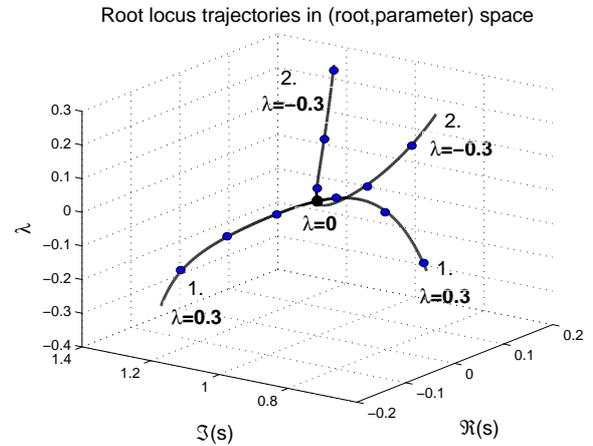


Fig. 3. The root-locus trajectories 1 and 2 in Figure 1 in the (characteristic root, locus parameter)-space.

4.3 Correction step

We use Newton's method to solve a set of nonlinear equations to find the real and imaginary parts of the corrected root and the locus parameter $y_{i+1}^c = [\sigma_{i+1}^c \ \omega_{i+1}^c \ \lambda_{i+1}^c]^T$.

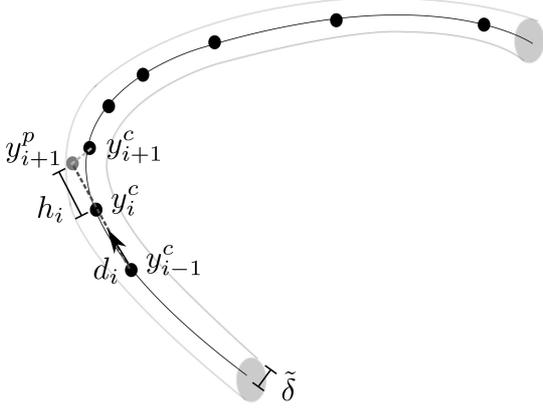


Fig. 4. Computing the root-locus trajectory: Prediction, correction steps and adaptively chosen step lengths.

These equations are given by

$$M(\sigma_{i+1}^c, \omega_{i+1}^c, \lambda_{i+1}^c) = 0, \quad (22)$$

$$P(\sigma_{i+1}^c, \omega_{i+1}^c, \lambda_{i+1}^c) = 0, \quad (23)$$

$$(y_{i+1}^c - y_{i+1}^p)d_i = 0. \quad (24)$$

The functions M and P are equivalent representations of the magnitude and phase equations of the root-locus equation. Equation (24) guarantees that the linearized distance of the corrected root and the locus parameter y_{i+1}^c from the predicted root and the locus parameter y_{i+1}^p is equal to the step size h_i . Because of the linearization the proposed method can be seen as a *pseudo arclength continuation* method [16].

The function M and P are defined using the root-locus equation w.r.t. the controller gain in (2) or the time delay in (3) as

$$M(\sigma, \omega, \lambda) = \begin{cases} \mathcal{M}(\sigma, \omega, \lambda, h) \\ \mathcal{M}(\sigma, \omega, 1, \lambda) \end{cases}, \quad P(\sigma, \omega, \lambda) = \begin{cases} \mathcal{P}(\sigma, \omega, \lambda, h) \\ \mathcal{P}(\sigma, \omega, 1, \lambda) \end{cases}$$

where

$$\mathcal{M}(\sigma, \omega, k, h) = \ln |\alpha| + \frac{1}{2} \sum_{r=1}^m (\ln(\sigma - \sigma_{zr})^2 + (\omega - \omega_{zr})^2) - \frac{1}{2} \sum_{i=1}^n (\ln(\sigma - \sigma_{pi})^2 + (\omega - \omega_{pi})^2) - h\sigma + \ln k, \quad (25)$$

$$\mathcal{P}(\sigma, \omega, k, h) = \angle \alpha + \sum_{r=1}^m \tan^{-1} \frac{\omega - \omega_{zr}}{\sigma - \sigma_{zr}} - \sum_{i=1}^n \tan^{-1} \frac{\omega - \omega_{pi}}{\sigma - \sigma_{pi}} - h\omega - \pi. \quad (26)$$

Note that the controller gain k and the time delay h have constant values in the root-locus equation w.r.t. the time delay in (3) and the controller gain in (2) respectively.

The function $P(\sigma, \omega, \lambda)$ has a range $(-\pi, \pi]$ and the arc-tangent functions in $P(\sigma, \omega, \lambda)$ have ranges $(-\pi, \pi]$ (similarly to the two argument function `atan2` in MATLAB).

We compute the corrected point y_{i+1}^c by Newton's method on (22-24), i.e., by iteratively solving

$$J_{i+1}^k (\tilde{y}_{i+1}^{k+1} - \tilde{y}_{i+1}^k) = f_{i+1}^k, \text{ for } k = 0, 1, \dots \quad (27)$$

where J_{i+1}^k and f_{i+1}^k are the Jacobian and the gradient vector evaluated at \tilde{y}_{i+1}^k . The iteration is initialized by the predicted point such that $\tilde{y}_{i+1}^0 = y_{i+1}^p$. We stop the iteration when the prescribed accuracy is reached and set the corrected point y_{i+1}^c to the last point of the iteration. The geometric illustration is given in Figure 4.

Regarding the computational complexity, the above computation requires evaluating the functions on the left-hand side of the equations in (22-24) and the corresponding Jacobian and gradient of these equations at each iteration. This computation is numerically cheap and uses only the poles and zeros of G and the time delay h .

4.4 Adaptive step length

The step length computation for the next prediction step depends on two factors [1]

- the *contraction rate* of the first two consecutive Newton steps in the corrector step, i.e., $\kappa_{i+1} := \frac{\|(J_{i+1}^0)^+ f_{i+1}^1\|}{\|(J_{i+1}^0)^+ f_{i+1}^0\|}$;
- the *distance* to the root-locus trajectory

$$\delta_{i+1} = \left\| 1 - e^{M(\sigma_{i+1}^c, \omega_{i+1}^c, \lambda_{i+1}^c) + jP(\sigma_{i+1}^c, \omega_{i+1}^c, \lambda_{i+1}^c)} \right\|.$$

The individual deceleration factors are calculated as $\kappa_{df} = \sqrt{\frac{\kappa_{i+1}}{\tilde{\kappa}}}$ and $\delta_{df} = \sqrt{\frac{\delta_{i+1}}{\tilde{\delta}}}$ where $\tilde{\kappa}$ and $\tilde{\delta}$ are the nominal contraction rate and the distance. The overall deceleration factor h_{df} of the step length is the maximum of individual deceleration factors, κ_{df} and δ_{df} limited to $[\frac{1}{2}, 2]$, i.e., $h_{df} := \max\{\min\{\max\{\kappa_{df}, \delta_{df}\}, 2\}, \frac{1}{2}\}$.

Note that if $h_{df} = 2$, the predictor step is repeated with a reduced step length. This check is done inside the corrector step to avoid unnecessary Newton iterations (see [1] for further details). The step length for the next prediction step is $h_{i+1} = h_i/h_{df}$.

The adaptive step length selection allows us to follow the root-locus trajectory in a computationally efficient way with a prescribed accuracy. When the trajectory has sharp (resp. wide) curves, the step lengths are smaller (resp. larger). This behavior is illustrated in Figure 4.

4.5 Detection of branch points

We directly computed the branch points of the root locus w.r.t. the controller gain in Section 3.1. The branch

point detection of the root locus w.r.t. the time delay requires monitoring the time delay parameter along the root-locus trajectory. By Lemma 1, the root-locus trajectory passes through the branch point when the multiplicity of the branch point is odd. In this case, there is no need to detect the branch point. Note that the locus parameter is strictly increasing for this case.

When the multiplicity of the branch point is even, the continuation method follows another root-locus trajectory after passing the branch point. The locus parameter on this trajectory is strictly decreasing since we follow the branch in the opposite direction. When we detect that the current time delay value is smaller than previous time delay values, we know that we have passed a branch point. Since we bracketed the branch point by the current and previous points, we then solve a set of nonlinear equations characterizing the branch point. Based on the multiplicity of the branch point, we decide the branch direction by Lemma 1. The branch point and its direction are added into the list for the next trajectories.

5 Algorithm

The overall algorithm is as follows.

- (1) Compute the critical points of the root locus.
- (2) For each root-locus trajectory:
 - (a) Starting with the initial point, find the next point by computing the predicted point y_{i+1}^p , the corrected point y_{i+1}^c and update the step length for the next step h_{i+1} .
 - (b) Continue to compute the next point until one of the following conditions holds:
 - (i) The trajectory reaches a branch point. Go to Step 2 – a) and compute the next point using the new direction by Lemma 1.
 - (ii) The locus parameter exceeds λ_{\max} or the trajectory leaves \mathbb{C}_{σ_0} . Stop the computation for this trajectory. Update the initial point with the starting point of a new root-locus trajectory and Go to Step 2 – a).
- (3) Stop if there is no remaining root-locus trajectory.

Note that due to the symmetry of the spectrum, on the real axis it is sufficient to compute the branch points and continue from the next branch point, i.e., a continuation method is not needed. In our implementation, we set the nominal contraction rate, the nominal distance and the tolerance for the corrector step to $\tilde{\kappa} = 0.5$, $\tilde{\delta} = 10^{-3}$ and 10^{-5} .

6 Examples

We consider three examples with unique characteristics on their root locus. Example 7 has circular trajectories

close to each other. Trajectories of Example 8 require different order of accuracy. Example 9 has a branch point, a trajectory going out of the region and trajectories with large complex numbers. We show root-locus plots only on the complex upper half-plane since the root-locus plots are symmetric w.r.t. the real axis due to real-valued coefficients of the plants in Examples.

Example 7 In [12], the stability of equation (3) is considered for the oscillator system, $G(s) = \frac{\epsilon s^2}{(s^2 + \omega_1^2)(s^2 + \omega_2^2)}$. The behavior of the rightmost characteristic roots is analyzed as a function of the delay parameter for small values of a gain parameter. In a particular example, the parameters of the system are set to $\epsilon = 1$, $\omega_1 = 2$, $\omega_2 = 4$ and the oscillatory nature of the rightmost characteristic root and the stability of the closed-loop system are shown.

We plot the root locus in equation (3) of this plant inside $\Re(s) \geq -1$ for $\lambda \in [0, 5]$ in Figure 5. It is well-known that the characteristic roots of time delay systems can only cross the imaginary axis at a finite number of points when the delay is varied. This property can be clearly seen in Figures 5-8. Note in particular in Figure 8 how characteristic roots are passing through same points on the imaginary axis. Root-locus plots in Figures 6-8 illustrate the oscillatory behavior of the characteristic root trajectories centered around $\pm 2j$, $\pm 4j$ and the asymptotic characteristic roots coming from the left. This phenomenon is qualitatively explained in [12]. The closed-loop system is stable for the time delay $\lambda \in [0.83, 1.50] \cup [4.11, 4.50]$.

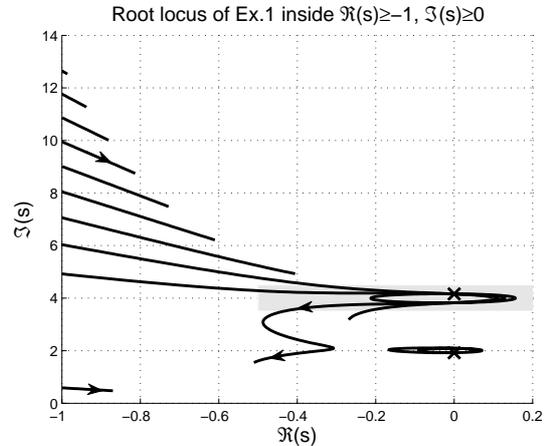


Fig. 5. The root locus of Example 7 for time delay $\lambda \in [0, 5]$ inside $\Re(s) \geq -1$, $\Im(s) \geq 0$.

Example 8 The following SISO dead-time system is borrowed from [3], p. 80-81 where $h = 12.48$ and

$$G(s) = 10^{-3} (s^6 - 6 \cdot 10^{-4} s^5 + 1.4081634 s^4 - 5.6326533 \cdot 10^{-4} s^3 + 0.43481891 s^2 - 8.6963771 \cdot 10^{-5} s + 2.6655565 \cdot 10^{-2})^{-1}.$$

The root-locus plot for equation (2) of this plant is given in Figure 9. As shown with cross markers inside gray regions in the figure, the plant has 6 unstable poles. They

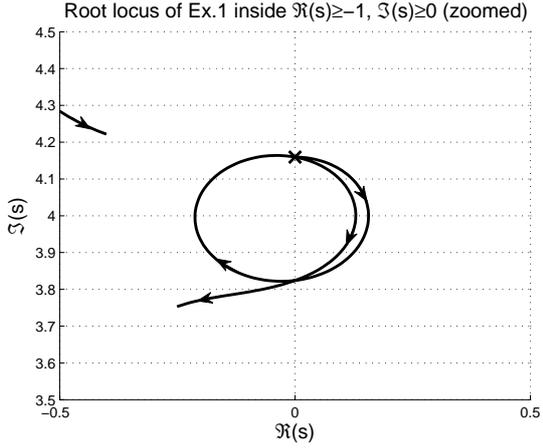


Fig. 6. The root locus inside the gray region in Figure 5 for time delay $\lambda \in [0, 2.7]$.

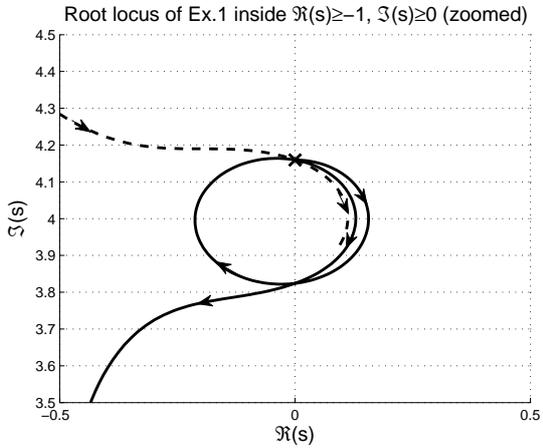


Fig. 7. The root locus inside the gray region in Figure 5 for time delay $\lambda \in [0, 3.8]$.

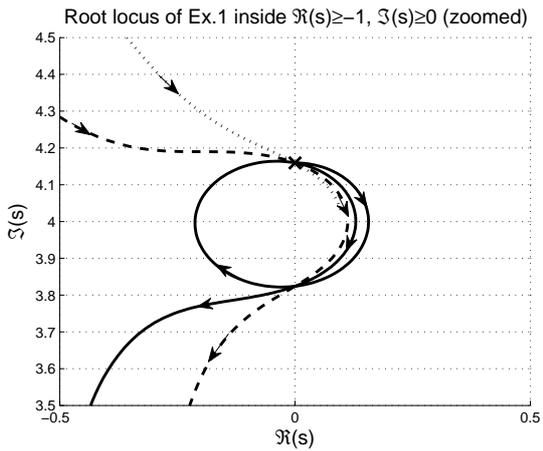


Fig. 8. The root locus inside the gray region in Figure 5 for time delay $\lambda \in [0, 5]$.

have all moved to the complex left half-plane at $\lambda = 1.860$ and the closed-loop system becomes stable for the con-

troller gain $\lambda \in [1.860, 4.469]$. Note that the root-locus trajectories have lengths in the order of magnitudes, $1, 10^{-2}, 10^{-3}, 10^{-4}$ respectively as illustrated in Figures 9-12. We follow each trajectory without any problem due to the adaptive step length in the algorithm.

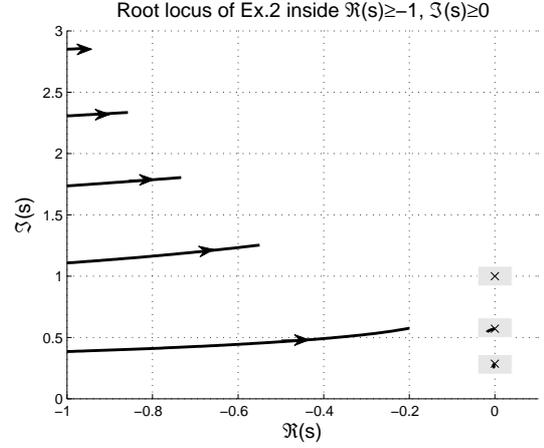


Fig. 9. The root locus of Example 8 for controller gain $\lambda \in [0, 6]$ inside $\Re(s) \geq -1, \Im(s) \geq 0$.

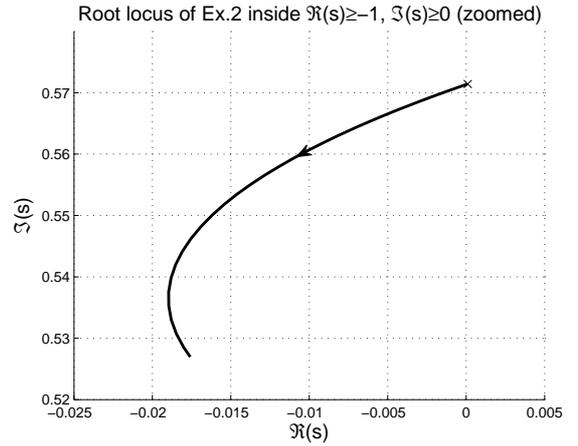


Fig. 10. The root locus inside the middle gray region in Figure 9 for controller gain $\lambda \in [0, 6]$.

Example 9 Finally, we consider the SISO dead-time system where $h = 1$ and $G(s) = \frac{s^2 - 10s + 50}{s^3 + 4s^2 + 4.25s + 1.25}$. We plot its root-locus trajectories of equation (2) for the controller gain interval $\lambda \in [0, 5]$ inside $\Re(s) \geq -3.5$. Figure 13 illustrates the general behavior of the root-locus trajectories. As the controller gain increases, more characteristic roots cross the boundary $\Re(s) = -3.5$ and asymptotic root trajectories get closer to the imaginary axis. Figure 14-15 show the local behavior around the starting points, $s = -0.5, -1, -2.5$. In Figure 14, we see that the characteristic roots cross the imaginary axis at $\lambda = 0.07$ and the closed-loop system becomes unstable. In Figure 15, the root locus has a branch point $s = -0.6976$ at $\lambda = 0.0009$ and the trajectory starting with $s = -2.5$ leaves the region at $\lambda = 0.0023$.

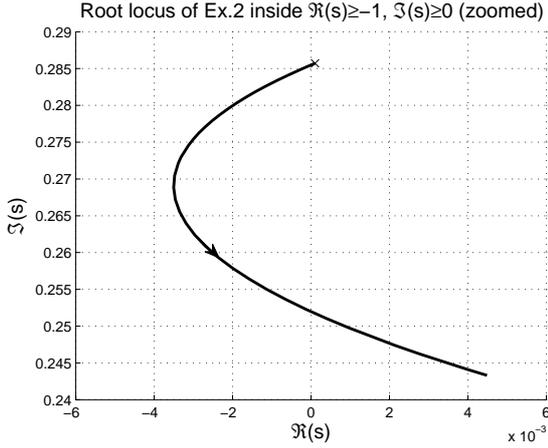


Fig. 11. The root locus inside the bottom gray region in Figure 9 for controller gain $\lambda \in [0, 6]$.

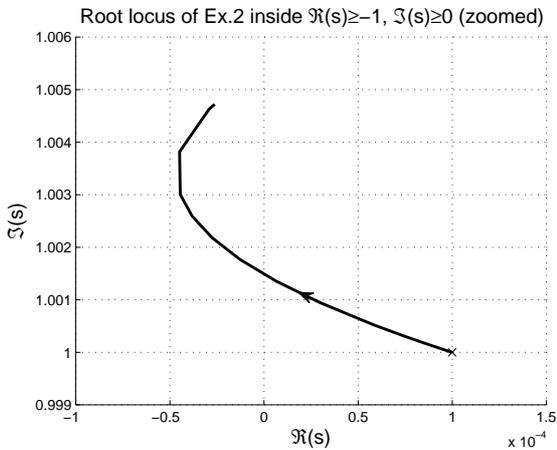


Fig. 12. The root locus inside the top gray region in Figure 9 for controller gain $\lambda \in [0, 6]$.

The computation times of root-locus plots for Examples 7-9 are 7.6, 1.4 and 2.5 seconds on a PC with an Intel Core Duo 2.53 GHz processor with 2 GB RAM respectively.

7 Concluding Remarks

The root locus of SISO dead-time systems with respect to the controller gain or the time delay is computed inside a given complex right half-plane. The method calculates the starting points of root-locus trajectories including the ones crossing the boundary of the root-locus region. Each trajectory is followed by a predictor-corrector continuation method. The characteristic roots on each root-locus trajectory are predicted by a secant method where the step length is adaptive and the predicted values are corrected by Newton's method. This continuation approach, which stems from numerical bifurcation analysis, is general and not restricted to parameterized equations of the form in equations (2,3). The implemen-

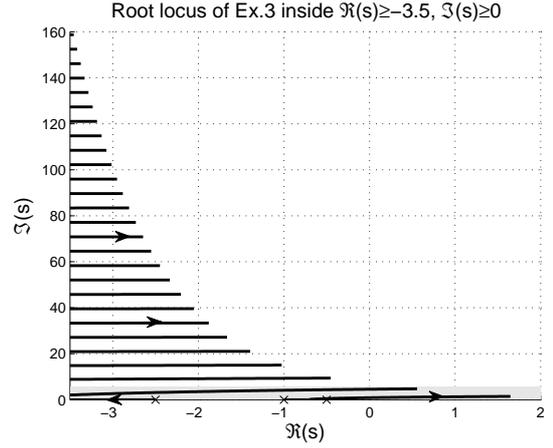


Fig. 13. The root locus of Example 9 for controller gain $\lambda \in [0, 5]$ inside $\Re(s) \geq -3.5$, $\Im(s) \geq 0$.

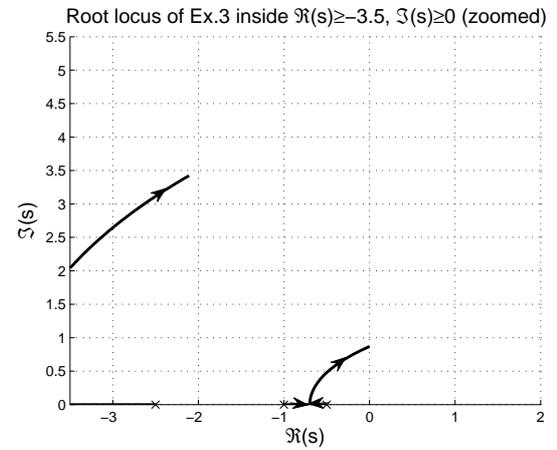


Fig. 14. The root locus inside the gray region in Figure 13 for controller gain $\lambda \in [0, 0.07]$.

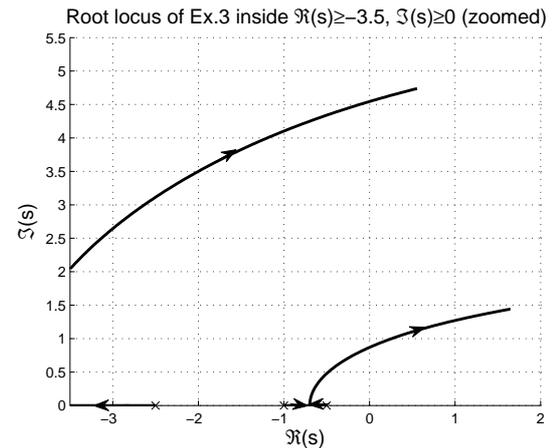


Fig. 15. The root locus inside the gray region in Figure 13 for controller gain $\lambda \in [0, 5]$.

tation of the method is numerically stable for high order systems.

Acknowledgements

This work has been supported by the Programme of Interuniversity Attraction Poles of the Belgian Federal Science Policy Office (IAP P6- DYSCO), by OPTEC, the Optimization in Engineering Center of the K.U.Leuven, by the project STRT1-09/33 of the K.U.Leuven Research Council and the project G.0712.11N of the Research Foundation - Flanders (FWO).

References

- [1] E. L. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*. SIAM, 2003.
- [2] R. H. Ash and G. R. Ash. Numerical computation of root loci using Newton-Raphson technique. *IEEE Transactions on Automatic Control*, 13(5):576–582, 1968.
- [3] F. M. Atay, editor. *Complex Time-Delay Systems: Theory and Applications*. Springer, 2010.
- [4] 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.
- [5] R. Brent. *Algorithms for Minimization Without Derivatives*. Prentice-Hall, 1973.
- [6] G. W. Evans. Bring root locus to the classroom - The story of Walter R. Evans and his textbook Control-System Dynamics. *IEEE Control Systems Magazine*, 24(6):74–81, 2004.
- [7] A. Fioravanti, C. Bonnet, H. Özbay, and S.-I. Niculescu. A numerical method to find stability windows and unstable poles for linear neutral time-delay systems. In *9th IFAC Workshop on Time Delay Systems*, Prague, Czech Republic, 2010.
- [8] S. Gumussoy. On the relative stability of closed-loop SISO dead-time systems: Non-imaginary axis case. *Accepted for publication in Transactions of the Institute of Measurement and Control*, 2011. doi: 10.1177/0142331210385854.
- [9] I. B. Huang and L. L. C. Li. Root locus determination of linear systems with transport lag. *IEEE Transactions on Automatic Control*, 12(5):632–634, 1967.
- [10] W. Krajewski and U. Viaro. Root-locus invariance - exploiting alternative arrival and departure points. *IEEE Control Systems Magazine*, 27(1):36–43, 2007.
- [11] A. M. Krall. Root locus method: A survey. *SIAM Review*, 12(1):64–72, 1970.
- [12] W. Michiels. Stability analysis of oscillatory systems subject to large delays: a synchronization point of view. *Journal of Vibration and Control*, 16(7-8):1087–1111, 2010.
- [13] W. Michiels and S. I. Niculescu. *Stability and Stabilization of Time-Delay Systems: An Eigenvalue Based Approach*. SIAM, 2007.
- [14] K. Nishioka, N. Adachi, and K. Takeuchi. Simple pivoting algorithm for root-locus method of linear systems with delay. *International Journal of Control*, 53(4):951–966, 1991.
- [15] K. Ogata. *Modern Control Engineering*. Prentice Hall, New Jersey, 3rd edition, 1997.
- [16] R. Seydel. *Practical Bifurcation and Stability Analysis: From Equilibrium to Chaos*. Springer Verlag, 1994.
- [17] I. H. Suh and Z. Bien. A root-locus technique for linear systems with delay. *IEEE Transactions on Automatic Control*, 27(1):205–208, 1982.
- [18] K. S. Yeung and W. T. Wong. Root-locus plot of systems with time-delay. *Electronics Letters*, 18(11):480–481, 1982.