Partial Pole Placement via Delay Action: A Python Software for Delayed Feedback Stabilizing Design

Islam Boussaada, Guilherme Mazanti, Silviu-Iulian Niculescu, Julien Huynh, Franck Sim, Matthieu Thomas

To cite this version:

HAL Id: hal-02897102
https://hal.science/hal-02897102v2

Submitted on 19 Sep 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Partial Pole Placement via Delay Action: A Python Software for Delayed Feedback Stabilizing Design

Islam Boussaada†, Guilherme Mazanti‡, Silviu-Iulian Niculescu*, Julien Huynh†, Franck Sim†, Matthieu Thomas†

†Université Paris-Saclay, CNRS, CentraleSupélec, Inria Laboratoire des signaux et systèmes (L2S)
91190 Gif-sur-Yvette, France
E-mails: {first name.last name} @l2s.centralesupelec.fr
‡Institut Polytechnique des Sciences Avancées (IPSA)
63 boulevard de Brandebourg, 94200 Ivry-sur-Seine, France
E-mails: {first name.last name} @ipsa.fr
∗Cyb’Air Association, 94200 Ivry-sur-Seine, France

Abstract—This paper presents a new Python software for the parametric design of stabilizing feedback laws with time delays, called Partial Pole Placement via Delay Action (P3δ). After an introduction recalling recent theoretical results on the multiplicity-induced-dominancy (MID) and coexisting roots-induced-dominancy (CRRID) properties and their use for the feedback stabilization of control systems operating under time delays, the paper presents the current version of P3δ, which relies on the MID property to compute delayed stabilizing feedback laws for scalar differential equations with a single delay. We detail in particular its graphical user interface (GUI), which allows the user to input the necessary information and obtain the results of the analysis done by the software. These results include the parameters stabilizing the closed-loop system, graphical representations of the spectrum of the closed-loop system, simulations of solutions in the time domain, and a sensitivity analysis with respect to uncertain delays.

Index Terms—Time-delay systems, Controller design, Stability, Stabilization, Python toolbox, GUI

I. INTRODUCTION

Time delays often occur in control systems, mainly due to the time required for acquiring, propagating, or processing information. For this reason, systems with time delays are a frequent topic in the control theory literature, with many works, such as [1]–[3], highlighting the effects of delays on the behavior of control systems, in particular on their stability.

Commonly, time delays lead to desynchronizing or destabilizing effects on the dynamics of the system they appear. However, some works have emphasized that the delay may also have a stabilizing effect in control design. For instance, in [4], a delayed controller is used in order to improve the stability of systems with oscillatory behavior and small damping. The stabilization properties of delayed controllers has also been considered in [5], which uses a proportional-delayed controller, replacing the classical proportional-derivative controller thanks to the “average derivative action” obtained via the time delay, a technique also used in [6]. Further discussion of the stabilizing effects of time delays can be found in [7], which highlights in particular the fact that closed-loop stability may be guaranteed for some control systems precisely by the existence of the delay. A growing literature exhibits the design of delayed controllers in a wide range of applications, such as, for instance, the control of flexible mechanical structures or the regulation of networks (see, e.g., [8], [9]).

In this paper, we consider linear time-invariant differential equations with a single time delay under the form

\[ y^{(n)}(t) + a_{n-1}y^{(n-1)}(t) + \cdots + a_0y(t) + b_m y^{(m)}(t - \tau) + \cdots + b_0 y(t - \tau) = 0, \]

where \( \tau > 0 \) is the positive delay, \( y \) is the real-valued unknown function, \( n \) and \( m \) are nonnegative integers with \( n > m \), and \( a_0, \ldots, a_{n-1}, b_0, \ldots, b_m \) are real coefficients.

The stability analysis of a linear time-invariant time-delay system can be addressed using spectral methods by considering the corresponding characteristic function, whose complex roots determine the asymptotic behavior of solutions of the system, as presented, e.g., in [2], [3]. The characteristic function corresponding to (1) is

\[ \Delta(s) = s^n + \sum_{k=0}^{n-1} a_k s^k + e^{-s\tau} \sum_{k=0}^m b_k s^k, \]
of quasipolynomials for the feedback stabilization of time-delay systems is that quasipolynomials have infinitely many roots, but one only disposes of finitely many parameters in the feedback law to choose the location of these roots and place them in order to guarantee a negative spectral abscissa, and hence exponential stability of the closed-loop system.

Recent works such as [8], [13]–[19] have been interested in the design of pole placement techniques for quasipolynomials with the aim of selecting the free parameters of the system in order to choose the location of finitely many roots in the complex plane and guarantee that the dominant root, i.e., the rightmost root on the complex plane, is among the chosen ones. Unlike methods based on finite spectrum assignment such as those from [20], the controllers designed using these techniques do not render the closed-loop system finite dimensional, but control instead its rightmost spectral value.

The works [8], [13]–[19] usually proceed either by assigning a real root of maximal multiplicity and proving that this root is necessarily the rightmost root of the characteristic quasipolynomial (a property known as multiplicity-induced-dominancy, or MID for short) or by assigning a certain amount of real roots (typically equally spaced for simplicity) and proving that the rightmost root among the assigned roots is also the rightmost root of the characteristic quasipolynomial (a property known as coexisting real roots-induced-dominancy, or CRRID for short).

The MID property for (1) is shown, for instance, in [8] in the case \( n = 2 \) and \( m = 0 \), in [16] in the case \( n = 2 \) and \( m = 1 \) (see also [15]), and in [17] in the case of any positive integer \( n \) and \( m = n - 1 \) (see also [18]). The CRRID property is shown, for instance, in [13] in the cases \((n, m) = (2, 0)\) and \((n, m) = (1, 0)\), and in [14] in the case of any positive integer \( n \) and \( m = 0 \). In all these cases, the maximal multiplicity of a real root or, equivalently, the maximal number of coexisting simple real roots is the integer \( n + m + 1 \).

This paper presents the Partial Pole Placement via Delay Action software (P3\( \delta \) for short), a Python software based on the results from [8], [13]–[19] for the parametric design of stabilizing feedback laws with time delays. The first version of P3\( \delta \), presented in the current paper, allows for the design of feedback laws for linear time-invariant differential equations with a single time delay under the form (1) using MID techniques.

Several other softwares have been recently developed for the analysis of time-delay systems from various perspectives, such as stability, robustness, or bifurcation aspects. This is the case, for instance, of the Matlab packages YALTA [21], dedicated to the \( H_{\infty} \) stability analysis of time-delay systems with commensurate delays, TRACE-DDE [22], devoted to the computation of characteristic roots and stability charts of linear autonomous time-delay systems, DDE-BIFTOOL [23], interested in the computation, continuation, and stability analysis of steady-state solutions of time-delay systems and their bifurcations, and QPmR [24], specialized in the computation of roots of quasipolynomials. One of the major novelties of P3\( \delta \) lies in addressing the stabilization of control systems with time-delays by using of the MID property to design stabilizing feedback laws. For that purpose, P3\( \delta \) makes use of both symbolic and numeric computations.

II. DESCRIPTION OF P3\( \delta \)

P3\( \delta \) is freely available for download on https://cutt.ly/p3delta, where installation instructions, video demonstrations, and the user guide are also available. Interested readers may also contact directly any of the authors of the paper.

In the current version of P3\( \delta \), only the MID property is exploited for the stabilization of (1). This can be done in two different ways, named “Classic MID” and “Control-oriented MID”, according to which coefficients of (1) are assumed to be fixed and which are assumed to be free.

A. Classic MID mode

The “Classic MID” mode corresponds to considering that all coefficients of the quasipolynomial \( \Delta \) from (2) are free. The user inputs the values of the delay \( \tau \) and of the desired real root \( s_0 \) and P3\( \delta \) computes all coefficients \( a_0, \ldots, a_{n-1}, b_0, \ldots, b_m \) ensuring that the value \( s_0 \) is a dominant root of \( \Delta \) of maximal multiplicity \( n + m + 1 \). To use the “Classic MID” mode, the user should proceed as follows:

1. Enter the values of the integers \( n \) and \( m \) appearing in the differential equation (1).
2. Select the “Classic MID” option in the drop-down menu “— Choose MID type —.”
3. Enter the values of the desired real root of maximal multiplicity \( s_0 \) and of the delay \( \tau \) in the corresponding fields that appear below the drop-down menu.
4. Enter the bounds \( x_{\min}, x_{\max}, y_{\min}, y_{\max} \) of the rectangle \([x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \subset \mathbb{C} \) in which P3\( \delta \) will look for roots of (2) and press the “Confirm” button.

Once the “Confirm” button is pressed, P3\( \delta \) will compute the values of the coefficients \( a_0, \ldots, a_{n-1}, b_0, \ldots, b_m \) ensuring that \( s_0 \) is a root of maximal multiplicity of the quasipolynomial \( \Delta \) from (2) and show their values. P3\( \delta \) will also numerically compute all roots of \( \Delta \) within the selected rectangle using the computed values of \( a_0, \ldots, a_{n-1}, b_0, \ldots, b_m \) and plot these roots in the plot “Roots” at the lower left corner of the window. This numerical computation is carried out using Python’s cxroots module, which implements numerical methods described in [25].

Optionally, after the previous computations are completed, the user may also simulate some trajectories of the system in the time domain. This can be done, after completing step 4 above, by the following steps:

5. Choose the type of the initial condition from the drop-down menu “— Initial Solution —.”

The currently supported types are “Constant”, “Polynomial”, “Exponential”, and “Trigonometric”, which corresponds to initial conditions of the forms \( x(t) = c, x(t) = \sum_{k=0}^c c_k t^k \), \( x(t) = A e^{\lambda t} \), and \( x(t) = A \sin(\omega t + \phi) \), respectively, where \( c, r, c_0, \ldots, c_r, A, \gamma, \omega, \phi \) are constants to be chosen by the user and the initial condition is defined in the time interval \([-\tau, 0]\).

6. Enter the simulation time in the corresponding box.
7. Enter the values of the constants appearing in the expression of the initial condition in the corresponding input boxes.
8. After entering all the constants, press “Enter” on the keyboard or click on the “Confirm” button appearing in the same frame as the constants.

After these steps, the numerical solution corresponding to the chosen initial condition will be computed using an explicit Euler scheme in the time interval \([-\tau, T]\), where \(T\) is the value entered in step 6. The corresponding solution will be plotted in the graph on the “Solutions” part of the screen.

Figure 1 shows a screen capture of the “Classic MID” mode of P3\(\delta\). In this figure, we have chosen \(n = 2, m = 1\), the “Classic MID” mode, and the values \(s_0 = -5\) and \(\tau = 1\). After entering \(x_{\text{min}} = -50, x_{\text{max}} = 50, y_{\text{min}} = -100\), and \(y_{\text{max}} = 100\) and having clicked on “Confirm”, P3\(\delta\) shows the values of the coefficients of the quasipolynomial, \(a_1 = 6, a_0 = 11, b_1 \approx -0.1078\), and \(b_0 \approx -0.0135\), ensuring that \(s_0 = -5\) is a root of maximal multiplicity \(n + m + 1 = 4\) in this case. P3\(\delta\) also plots the numerical roots of the quasipolynomial in the selected rectangle in the graph on the lower left corner. After choosing the “Constant” initial condition, selecting the simulation time \(T = 5\) and the value 1 for the constant, the corresponding numerical solution of the system appears in the graph in the lower right corner of the screen.

B. Control-oriented MID mode

The “Control-oriented MID” mode corresponds to considering that the coefficients \(a_0, \ldots, a_{n-1}\) corresponding to the non-delayed terms of (1) are given and that the coefficients \(b_0, \ldots, b_m\) corresponding to the delayed terms are free. The user may choose to input either the value of \(\tau\) or that of \(s_0\) (but not both) and P3\(\delta\) computes all coefficients \(b_0, \ldots, b_m\) ensuring the existence of a dominant root of the quasipolynomial \(\Delta\) from (2) of multiplicity \(m + 2\). P3\(\delta\) also computes the value of the parameter among \(\tau\) or \(s_0\) that has not been fixed by the user.

Remark 1. In the “Control-oriented MID” mode, it may happen to be impossible to choose a real root \(s_0\) of multiplicity \(m + 2\). In this case, P3\(\delta\) warns the user of this fact and provides an equation relating \(s_0\) and \(\tau\). The user should either enter a value of \(s_0\) such that this equation admits a positive root \(\tau\) or a positive value of \(\tau\) such that this equation admits a real root \(s_0\) in order to proceed with the computations.

To use the “Control-oriented MID” mode, the user should proceed as follows:
1. Enter the values of the integers \(n\) and \(m\) appearing in the differential equation (1).
2. Select the “Control-oriented MID” option in the drop-down menu “— Choose MID type —”. After this selection, the window of the program is filled with the places for the other inputs and the outputs of P3\(\delta\).
3. Select from the drop-down menu “— Choose \(s_0\) or tau —” whether to input the value of the multiple root \(s_0\) or the value of the delay \(\tau\).
4. Enter the value of \(s_0\) or \(\tau\), according to the choice of the previous step.
5. Enter the values of the known coefficients \(a_0, \ldots, a_{n-1}\) and press the “Confirm” button located in the same frame.
6. Enter the bounds \(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}\) of the rectangle \([x_{\text{min}}, x_{\text{max}}] \times [y_{\text{min}}, y_{\text{max}}] \subset \mathbb{C}\) in which P3\(\delta\) will look for roots of (2).
7. Press the “Confirm” button.

Once the “Confirm” button is pressed, P3\(\delta\) will compute the values of the coefficients \(b_0, \ldots, b_m\) ensuring that \(s_0\) is a root of multiplicity \(m + 2\) of the quasipolynomial \(\Delta\) from (2) and show their values. Similarly to the “Classic MID” option, P3\(\delta\) will also numerically compute all roots of \(\Delta\) within the selected rectangle by using Python’s \texttt{croots} module and output the result in the plot “Roots”.

As in the “Classic MID” case, the user may plot solutions in the time domain. After completing step 7 above, this can be done by following the same steps 8–9 from Section II-A.

Figure 2 shows a screen capture of the “Control-oriented MID” mode of P3\(\delta\). In this figure, we have chosen \(n = 2, m = 1\), the “Control-oriented MID” mode, the input of \(\tau\), the value \(\tau = 1\), and the coefficients \(a_0 = -3\) and \(a_1 = 2\). After entering \(x_{\text{min}} = -50, x_{\text{max}} = 50, y_{\text{min}} = -100\), and \(y_{\text{max}} = 100\) and having clicked on “Confirm”, P3\(\delta\) shows the values of the coefficients of the quasipolynomial, \(b_1 \approx 1.6717\) and \(b_0 \approx 3.1104\), recalls the value \(\tau = 1.0\), and also shows the value of the root of multiplicity \(m + 2 = 3\), \(s_0 \approx -0.550510\). P3\(\delta\) also plots the numerical roots of the quasipolynomial in the selected rectangle in the graph on the lower left corner. After choosing the “Constant” initial condition, selecting the simulation time \(T = 20\) and the value 1 for the constant, the corresponding numerical solution of the system appears in the graph in the lower right corner of the screen.

In addition to these outputs, which are similar to the “Classic MID” case, the “Control-oriented MID” option can also perform a numerical sensitivity analysis of the computed roots with respect to variations in the delay \(\tau\). To do so, the user should follow the above steps up to step 7, selecting to enter the value of \(\tau\) in step 3. Then, the steps to get the sensitivity plot are the following:
1. Select the “Sensitivity” tab in the “Roots” plot.
2. Select “tau sensitivity” in the drop-down menu “— Sensitivity —” above the “Roots” plot.
3. Enter the value of the step \(\epsilon\) and the number of iterations \(K\) in the corresponding boxes.
4. Enter the bounds \(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}\) of the rectangle \([x_{\text{min}}, x_{\text{max}}] \times [y_{\text{min}}, y_{\text{max}}] \subset \mathbb{C}\) in which P3\(\delta\) will look for roots of (2).

Since the sensitivity computation may take quite some time, it is highly recommended to choose a smaller rectangle containing few roots of \(\Delta\), including the dominant multiple root.

5. Press the “Confirm” button in the frame of the bounds of the rectangle.

Once these steps are completed, the sensitivity plot appears in the “Roots” plot. This plot contains the roots of \(\Delta\) in the selected rectangle for the values of delays \(\tau + k\epsilon\) for \(k \in \{-K, -K + 1, \ldots, K - 1, K\}\). Roots computed with negative values of \(k\), corresponding to values of the delay smaller than \(\tau\), are represented in shades of blue, with darker blue representing \(k = -K\) and lighter tones representing increasing values of \(k\). Roots computed with positive values of \(k\), corresponding to values of the delay larger than \(\tau\), are represented in shades of orange to red, with darker red representing \(k = K\) and lighter tones moving to orange representing decreasing values of \(k\). The roots computed with
Fig. 1. “Classic MID” mode of P3δ.

$k = 0$, corresponding to the nominal value of $\tau$ selected by
the user, are represented by black diamonds.

Figure 3 represents the part of P3δ screen corresponding to
the sensitivity computation. After having completed the steps
that led to the screen shown in Figure 2, we have chosen the
step $\varepsilon = 0.005$ and the number of steps $K= 4$ and selected
the default rectangle $[-5,5] \times [-10,10] \subset \mathbb{C}$. Clicking on
“Confirm”, P3δ outputs the graph shown in Figure 3. We
observe that, when the delay $\tau$ is perturbed, the root $s_0 \approx
-0.550510$ of multiplicity 3 splits into three simple roots.

III. ILLUSTRATIVE EXAMPLES

As illustrations of the use of P3δ, this section revisits two
examples from [16].

A. A first order equation

We consider here the delay-differential equation

$$\dot{y}(t) + a_0 y(t) + b_0 y(t - \tau) = 0,$$  \hspace{1cm} (3)

whose characteristic quasipolynomial is $\Delta(s) = s + a_0 +
b_0 e^{-s \tau}$. We then have $n = 1$ and $m = 0$. According to [16],
the maximal multiplicity of a real root of $\Delta$ is 2, and it is
attained if and only if $a_0 = -a_0 = \frac{1}{2}$ and $b_0 = e^{a_0 \tau}$.

Inputting $n = 1$, $m = 0$, selecting “Classic MID”,
and choosing $s_0 = -2$ and $\tau = 1$ in P3δ, we obtain $a_0 = 1$
and $b_0 \approx 0.1353$, which is in accordance with the above
expressions for $a_0$ and $b_0$. We also obtain the roots of $\Delta$ in
a given rectangle, represented in Figure 4(a) for the rectangle
$[-50,50] \times [-100,100]$, and time simulations of solutions,
for instance the one from Figure 4(b), obtained with initial
condition $x(t) = \sin(10t)$.

B. Stabilization of the double integrator

Let us consider a double integrator $\ddot{y}(t) = u(t)$ with the
delayed feedback control $u(t) = -b_1 \dot{y}(t - \tau) - b_0 y(t - \tau)$,
which yields the delay-differential equation

$$\dot{y}(t) + b_1 \dot{y}(t - \tau) + b_0 y(t - \tau) = 0,$$  \hspace{1cm} (4)

whose characteristic quasipolynomial is $\Delta(s) = s^2 + (b_1 s +
b_0) e^{-s \tau}$. This corresponds to $n = 2$, $m = 1$, and $a_0 = a_1 =
0$. According to [16], the maximal achievable multiplicity for
a root $s_0$ of $\Delta$ is 3, which is attained if and only if

$$b_1 = 2 \left(\sqrt{2} - 1\right) e^{-2+\sqrt{2}}, \quad b_0 = 2 \left(5\sqrt{2} - 7\right) e^{-2+\sqrt{2}},$$

$$s_0 = -2 - \frac{\sqrt{2}}{\tau}. \hspace{1cm} (5)$$

Inputting $n = 2$, $m = 1$, selecting “Control-oriented MID”,
and choosing $\tau = 1$, $a_0 = 0$, and $a_1 = 0$ in P3δ, we obtain $b_1 \approx 0.4612$, $b_0 \approx 0.0791$, and $s_0 \approx -0.585786$, which is
in accordance with (5). We also obtain the roots of $\Delta$ in
a given rectangle, represented in Figure 4(c) for the rectangle
$[-50,50] \times [-100,100]$, and time simulations of solutions,
for instance the one from Figure 4(d), obtained with initial
condition $x(t) = \sin(10t)$.

IV. CONCLUSION AND PLANNED DEVELOPMENTS

By using the recent theoretical results on the MID prop-
erty for linear delay-differential equations under the form
(1), P3δ computes the system parameters ensuring that a
given real root $s_0$ attains its maximal multiplicity and is
thus dominant. P3δ currently works in two modes, “Classic MID” and “Control-oriented MID”, which differ on which coefficients of (1) are assumed to be free. In all modes, P3δ computes the values of the free coefficients ensuring maximal multiplicity, performs a numerical computation of the roots of the characteristic quasipolynomial, and is able to perform time-domain simulations. The “Control-oriented MID” option also offers the possibility of a numerical study of the sensitivity of the roots with respect to variations in the delay.

There are currently several plans for future development of P3δ in its future versions, including sensitivity analysis with respect to the known parameters $a_0, \ldots, a_{n-1}$ in the “Control-oriented MID” option, the inclusion of new options allowing for a more flexible choice of which coefficients are assumed to be fixed and which are assumed to be free, and the use of CRRID-based results to place coexisting real roots.

**ACKNOWLEDGMENTS**

The authors wish to acknowledge the work of the full P3δ development team, which, in addition to the authors, also include Mickael Alcaniz, Yoann Audet, Thomas Charbonnet, Honoré Curlier, Adrien Leclerc, Max Perraudin, Pierre-Henry Poret, and Achrafy Said Mohamed. The development of P3δ was also made possible thanks to the work of the Cyb’Air Association.

**REFERENCES**

Fig. 4. Example of the usage of P3δ (a, b) in the “Classic MID” mode for the delay-differential equation (3) and (c, d) in the “Control-oriented MID” mode for the delay-differential equation (4). (a, c) Roots of the characteristic quasipolynomial in the rectangle $[-50, 50] \times [-100, 100]$. (b, d) Numerical simulation of a solution with initial condition $x(t) = \sin(10t)$. 


