Structured controller design for closed-loop $D$-stability in convex/non-convex regions: Mixed integer-linear programming approach

Farshad Merrikh-Bayat

Department of Electrical and Computer Engineering, University of Zanjan, Zanjan, Iran

Abstract

Here it is assumed that the characteristic function of a linear control system, which is in the form of a polynomial and its coefficients are desired affine functions of unknown parameters of controller, is given. It is also assumed that the transfer function of controller has a desired order and structure. Hence, some of the coefficients of the characteristic polynomial may not depend on some or any of the controller parameters. The main problem under consideration is to calculate the parameters of the controller such that all roots of the characteristic equation lie (if possible) inside the desired $D$-stability region, which can be any convex/non-convex connected/disconnected subset of complex plane. This problem has important applications in control theory. For example, non-convex $D$-stability regions appear when designing a controller for a fractional-order system is aimed. It is shown that this problem, which is still open even in dealing with general convex regions, is exactly equivalent to a set of linear algebraic equalities/inequalities in mixed real-integer variables, which can be solved efficiently by using the available software. Application of the proposed method for optimal controller design is also studied and three numerical examples are presented.

1 | INTRODUCTION

Stability is the crucial requirement of any control system. In practice, however, stability is not sufficient for good behaviour of the control system and it is desired that the closed-loop poles lie in a subset of complex plane which has some nice properties. For example, one may want to put a lower bound on both the exponential decay rate and the damping ratio of the closed-loop response. This requirement is fairly fulfilled by locating the roots of characteristic equation in a suitable region of complex plane. By definition, a polynomial is $D$-stable if all its roots are in the region $D$ of the complex plane. A more rigorous mathematical definition is as follows. Let $\Gamma$ be a closed Jordan curve and let int $\Gamma$ and ext $\Gamma$ denote its interior and exterior, respectively. Let $D$ denote int $\Gamma$. A characteristic polynomial of degree $n$ is $D$-stable if and only if the image of $\Gamma$ under characteristic equation encircles the origin $n$ times in a counter-clockwise manner [1, Ch. 3].

One important problem in the field of linear control systems is to design a controller such that all poles of the closed-loop system lie inside the desired region in complex plane. This problem, which is also studied under the subjects like relative stability and root clustering, has many applications in control theory. See, for instance, [2] and [3] where the problem of root clustering in parameter space is deeply studied and some applications are also presented. Robust $D$-stabilising controller design for an uncertain discrete-time networked control system with time-varying network-induced delays [4], computation of state-feedback gain matrix for a linear time-invariant (LTI) system such that all the finite poles are placed within a pre-defined stability region [5] and reduced-order controller synthesis with regional pole constraint [6] are some other applications of $D$-stability.

Although most of the $D$-stability regions used in practice are convex, non-convex regions also matter. One of the famous non-convex regions in the field of control theory are those appear in studying the stability of fractional-order systems.
the desired model. The optimisation problem in (3) equivalently can be written as follows:

$$\min \sum_{k=1}^{L} \alpha_k,$$

s.t. $C(\omega) \text{stabilises the feedback system},$

$$-\alpha_k < \Re(C(j\omega_k)P(j\omega_k)(1 - M(j\omega_k)) - M(j\omega_k)) < \alpha_k,$$

$$-\alpha_k < \Im(C(j\omega_k)P(j\omega_k)(1 - M(j\omega_k)) - M(j\omega_k)) < \alpha_k,$$

$$0 < \alpha_k, \quad k = 1, \ldots, L$$

But, in general, for two main reasons the $C(\omega)$ obtained from (2) is not acceptable. The first reason is that it does not have the desired structure while favourite controllers like proportional-integral-derivative (PID) and lead–lag are low order with a specific structure. Recall that in the context of LTI systems, a controller is called full order and unstructured if its degree is equal to the degree of the corresponding plant to be controlled and any entry of its state-space matrices can be adjusted freely. The other reason is that it does not guarantee the stability of closed-loop system. A possible remedy is to calculate $C(\omega)$ from the following optimisation problem:

$$\min \sum_{k=1}^{L} \left| C(j\omega_k)P(j\omega_k)(1 - M(j\omega_k)) - M(j\omega_k) \right|,$$

where $\omega_k$ are the samples taken from the frequency range of interest (totally $L$ samples). Clearly, the controller obtained from (3) stabilises the closed-loop system and minimises the difference between the frequency response of the closed-loop system and the desired model. The optimisation problem in (3) equivalently can be written as follows:

$$\min \sum_{k=1}^{L} \alpha_k,$$

s.t. $C(\omega) \text{stabilises the feedback system},$

$$-\alpha_k < \Re(C(j\omega_k)P(j\omega_k)(1 - M(j\omega_k)) - M(j\omega_k)) < \alpha_k,$$

$$-\alpha_k < \Im(C(j\omega_k)P(j\omega_k)(1 - M(j\omega_k)) - M(j\omega_k)) < \alpha_k,$$

$$0 < \alpha_k, \quad k = 1, \ldots, L$$

where $\Re$ and $\Im$ stand for real and imaginary part, respectively. When $C(\omega)$ is linear in its unknown parameters (e.g. it has the structure of a PID), the inequality constraints in (4c) and (4d) are also linear in these unknown parameters. In this case (4a)–(4c) constitute a linear program which can be solved very effectively by the available software. But, unfortunately, the crucial stability constraint (4b) makes the problem much more complicated and harder to solve. In practice, for reasons like preventing the saturation of actuators, the $D$-stability constraint is considered instead of (4b). In Example 2 of Section 3 it is shown how the above problem can be formulated as a mixed integer-linear program (MILP) and then solved.

Many of the methods developed in recent years for designing low-order structured controllers are based on solving a constrained optimisation problem by employing a gradient-based iterative algorithm. These methods usually begin from an initial stabilising controller and iteratively update it until a (local) optimal solution is found. For example, in [13] a weighted sum of the integral of the absolute errors caused by input and output disturbances is minimised subject to constraints on the peaks of the sensitivity and complementary sensitivity functions using

\[ M(s) = \frac{C(s)P(s)}{1 + C(s)P(s)} = M(s), \]  

or equivalently,

\[ C(s)P(s)(1 - M(s)) - M(s) = 0. \]
the proposed exact gradients. A very similar idea is studied in [14]. In [13] parameters of a filtered PID are calculated by minimising the integral error (IE) performance index (in response to load disturbance) subject to constraints on the probability of violation of the $H_{\infty}$-norm of sensitivity and complementary sensitivity functions. Again the algorithm is gradient based and starts from an initial solution which stabilises the feedback system. In some other references like [16] and [17] the information of sub-Gradients is used to solve a constrained $H_{\infty}$-norm minimisation problem. These methods are also of iterative nature and begin the search from a stabilising controller. The important point is that the solution obtained at the end by the above-mentioned iterative algorithms is, in general, a local optimum whose quality drastically depends on the initial solution used to begin the search. In order to be confident that the structured controller obtained at end has some nice properties, the search should begin from a suitable $D$-stabilising controller.

Here the standard unity-feedback LTI system is considered and it is assumed that the characteristic equation of the closed-loop system with a structured controller in the loop is in the form of $\sum_{k=0}^{n} a_k s^k = 0$ where some of $a_k$'s are fixed real numbers and the others are affine functions of the parameters of controller. Using this assumption, an algorithm for calculating the parameters of controller such that all roots of the characteristic equation lie inside a pre-specified convex/non-convex $D$-stability region is proposed. This problem is still open even when the region is convex. A rather similar problem is studied in [18] where it is aimed to find a minimum order controller such that all poles of the MIMO feedback system are placed inside a pre-specified convex region. That study is based on the properties of polynomial matrices and has the advantage of being directly applicable to MIMO systems. However, the stability region under consideration is necessarily convex and the original non-convex problem is approximated by a convex one.

The method proposed for structured $D$-stabilising controller design finds the parameters of controller by solving an MILP. In recent years MILPs have found many applications in different fields of engineering such as train speed trajectory optimisation [19], optimisation of unit commitment and economic dispatch [20], optimal configuration planning for energy hubs [21], etc. Few applications of MILP in the field of control theory are also reported in the literature. In these applications first often a kind of linear (affine) mathematical model is obtained for the dynamical system under consideration by introducing some auxiliary integer (decision) variables. This approach is especially common in dealing with hybrid systems which involve both continuous and discrete variables. At the next step often a model predictive controller (MPC) is designed for this mathematical model of system by solving an MILP. For example, in [22] a piecewise affine (PWA) system is modelled as a set of linear equalities and inequalities by introducing some binary and continuous-time auxiliary variables. Then it is shown that applying an explicit MPC (eMPC) to the resulting model yields a multi-parametric MILP problem which can be solved using the available software. As another example, in [23] a microgrid management system is modelled by a set of mixed integer-linear equality/inequality constraints. Then these constraints are incorporated in an MPC for further compensation of uncertainties through the feedback mechanism. In this manner, at each time sample the optimal control law is obtained by solving an MILP. Paper [24] is also a classical reference on modelling complex systems by means of logical and integer decision variables.

Although MILP is a very powerful tool for modelling complex systems and problems, it is known to be NP-complete, which means that the resulting program may be extremely difficult to solve in practice. Hence, special care should be taken during formulation of the problem to make sure that the resulting MILP is fairly relax and solvable by using the available commercial software.

The remaining is organised as follows. Main results, including the proposed MILP for designing a $D$-stabilising structured controller are presented in Section 2. Three numerical examples are studied in Section 3 and finally Section 4 concludes the paper.

## 2 | MAIN RESULTS

### 2.1 | Structured $D$-stabilising controller design

In this paper the characteristic equation of the feedback system is considered as $f(s, \textbf{k}) = 0$ where $s = \sigma + j\omega$ is the Laplace variable and $\textbf{k}$ is the vector containing the unknown parameters of the (structured) controller under consideration. Clearly, for SISO systems $f$ is always an affine function of $\textbf{k}$. For example, assuming the transfer functions of plant and controller as $P(s) = 1/(s + 1)$ and $C(s) = (as + b)/(s + c)$, respectively, the characteristic equation can be written as $f(s, \textbf{k}) = s^2 + s + [s + 1]k = 0$ where $\textbf{k} = [a \ b \ c]^T$.

The aim of the following discussions is to develop a method for calculating a vector $\textbf{k}$, which contains the unknown parameters of the structured controller $C(s)$, such that all roots of $f(s, \textbf{k}) = 0$ lie inside a desired convex/non-convex region in $s$-plane. Obviously, depending on the shape and size of the convex/non-convex region under consideration and the special structure considered for controller, this problem can have infinitely many or even no solutions. Hence, in general, the feasible $\textbf{k}$ is not unique.

The first step of the proposed method for calculating a $D$-stabilising structured controller is meshing the region under consideration. For example, Figure 2 shows a non-convex region and a possible meshing where the nodes of mesh are identified by black spheres. As mentioned earlier, such a non-convex region which comprises points from the right half plane (RHP) appears in studying the stability of FOS. In this paper it is assumed that the nodes are located at the intersection points of some horizontal and vertical lines, and each cell is in the shape of a rectangle. As can be observed in Figure 2, only the cells whose all four nodes belong to the region under consideration are taken into account (these cells are identified by grey painting in Figure 2).

The above discussion turns out that when the border of region consists of slant lines, which is the case in Figure 2, the
cells only can approximately fill inside the region. However, the error caused by this approximation is not a big problem since one can make the difference between the approximate and the original region arbitrary small by making use of smaller cells. As another point, it will be shown later in this section that the width and length of the cells can be different. Moreover, in practice the meshing should be such that no nodes lie on the negative real axis. The reason is that the proposed algorithm puts the roots of the characteristic equation only inside the cells and if the negative real axis does not pass through some of the cells (i.e. the edges and vertices of some of the rectangles lie exactly on the negative real axis), the characteristic equation cannot have any negative real roots. This situation is impossible in dealing with many problems.

In the following, necessary and sufficient conditions to have a root inside a cell will be developed. For the sake of simplicity, first consider the parameter-free equation $f(s) = u(x, y) + iv(x, y) = 0$ where $u = \Re(f(s))$ and $v = \Im(f(s))$ are the real and imaginary parts of $f(s)$, respectively. Obviously, the roots of this equation are the intersection points of the curves $u(x, y) = 0$ and $v(x, y) = 0$ in $x - y$ plane. Suppose that $u(x_0, y) = 0$ and $v(x, y) = 0$ have an intersection point (of order one) at $s_0 = x_0 + iy_0$ which is located inside a rectangular cell as shown in each of the sub-figures of Figures 3 and 4.

Considering the fact that $u$ is a continuous two-variable function of $x$ and $y$, it must be positive for the points located on one side of the curve $u(x, y) = 0$ and negative for the points located on its other side. A same discussion goes on $v(x, y)$. Depending on the relative positions of the curves $u(x, y) = 0$ and $v(x, y) = 0$ with respect to each other, and the signs of $u$ and $v$ on the two sides of these curves, totally exactly eight scenarios are possible as depicted in Figures 3 and 4. On the other hand, when $f$ is an analytic function, which is the case in our problem since $f$ is a polynomial, the Cauchy–Riemann necessary conditions for analyticity of the characteristic polynomial must also be sat-
equivalent algebraic relations, assign the following discussions. For the sake of simplicity in notation, set the set of algebraic relations in suitable decision variables cells of the mesh of interest. In the following, by introducing the roots of \( f(x, y) \) inside the rectangular cell under consideration if and only if the signs of \( u \) and \( v \) satisfy exactly one of the cases shown in Figure 3. This statement is equivalent to saying that exactly one of the following four logical sentences must be true:

**Case 1 (Figure 3a):**
\[ \Re (f(x_1, k)) < 0 \land \Im (f(x_2, k)) > 0 \land \Re (f(x_2, k)) < 0 \land \Im (f(x_2, k)) > 0 \land \Re (f(x_1, k)) > 0 \land \Im (f(x_1, k)) > 0 \land \Re (f(x_1, k)) > 0 \land \Im (f(x_1, k)) < 0, \] (6a)

**Case 2 (Figure 3b):**
\[ \Re (f(x_1, k)) > 0 \land \Re (f(x_2, k)) < 0 \land \Re (f(x_2, k)) > 0 \land \Re (f(x_1, k)) < 0 \land \Im (f(x_1, k)) < 0 \land \Im (f(x_1, k)) > 0 \land \Im (f(x_1, k)) > 0 \land \Im (f(x_2, k)) > 0, \] (6b)

**Case 3 (Figure 3c):**
\[ \Re (f(x_1, k)) < 0 \land \Re (f(x_2, k)) < 0 \land \Re (f(x_2, k)) > 0 \land \Re (f(x_1, k)) < 0 \land \Im (f(x_1, k)) > 0 \land \Im (f(x_1, k)) < 0 \land \Im (f(x_2, k)) > 0, \] (6c)

**Case 4 (Figure 3d):**
\[ \Re (f(x_1, k)) > 0 \land \Re (f(x_2, k)) > 0 \land \Re (f(x_2, k)) < 0 \land \Re (f(x_2, k)) > 0 \land \Im (f(x_1, k)) < 0 \land \Im (f(x_1, k)) > 0 \land \Im (f(x_2, k)) < 0, \] (6d)

where \( \land \) stands for the logical AND, and the points \( x_1, x_2, s_2 \) and \( s_2 \) are considered as shown in Figure 5. The main problem with the logical sentences in (6) is that one cannot directly solve them to find a feasible \( \mathbf{k} \). The other problem is that the number of the roots of \( f(x, k) = 0 \) in the region of interest (e.g. as the one shown in Figure 2) is equal to the degree of \( f(x, k) \), while logical trueness of exactly one of the sentences in (6) guarantees placing only one of the roots of \( f(x, k) = 0 \) in one of the cells of the mesh of interest. In the following, by introducing suitable decision variables we transform the sentences in (6) to a set of algebraic relations in \( \mathbf{k} \) whose solution puts each root of \( f(x, k) = 0 \) in one of the cells of the mesh under consideration. For the sake of simplicity in notation, \( f(x, k) \) is shown as \( f(\mathbf{i}) \) in the following discussions.

In order to get rid of the logical ANDs in (6) and arrive at equivalent algebraic relations, assign five binary decision variables to each cell (denoted as \( \delta_1, \delta_2, \delta_3, \delta_4 \) and \( \lambda_1 \)) and two binary decision variables to each node of the mesh (denoted as \( R_{11}, R_{12}, R_{21}, R_{22}, I_{11}, I_{12}, I_{21} \) and \( I_{22} \)) as shown in Figure 5. These binary decision variables are defined as the following:

\[
\begin{align*}
R_{jk} &= \begin{cases} 
0, & \Re (f(\mathbf{j}_k)) < 0 \\
1, & \Re (f(\mathbf{j}_k)) > 0,
\end{cases} \\
I_{jk} &= \begin{cases} 
0, & \Im (f(\mathbf{j}_k)) < 0 \\
1, & \Im (f(\mathbf{j}_k)) > 0,
\end{cases}
\end{align*}
\]

\[
\lambda_1 = \begin{cases} 
0, & f(\mathbf{i}) = 0 \text{ has no roots in the cell} \\
1, & f(\mathbf{i}) = 0 \text{ has one root in the cell}.
\end{cases}
\]

\[
\delta_1 = \begin{cases} 
0, & \text{sentence (6a) is logically false} \\
1, & \text{sentence (6a) is logically true},
\end{cases}
\]

\[
\delta_2 = \begin{cases} 
0, & \text{sentence (6c) is logically false} \\
1, & \text{sentence (6c) is logically true},
\end{cases}
\]

\[
\delta_3 = \begin{cases} 
0, & \text{sentence (6b) is logically false} \\
1, & \text{sentence (6b) is logically true},
\end{cases}
\]

\[
\delta_4 = \begin{cases} 
0, & \text{sentence (6d) is logically false} \\
1, & \text{sentence (6d) is logically true}.
\end{cases}
\]

Using the above definition for the binary decision variables, for each cell of the mesh write five relations as follows:

\[
\begin{align*}
\delta_1 + \delta_2 + \delta_3 + \delta_4 &= \lambda_1, \\
(8 - \varepsilon)\delta_1 - \varepsilon &\leq R_{21} + R_{22} + R_{12} + R_{11} \\
+ I_{21} + I_{22} + I_{12} + I_{11} &\leq \delta_1 + 7 + \varepsilon,
\end{align*}
\]

\[
(8 - \varepsilon)\delta_2 - \varepsilon \leq R_{21} + R_{22} + R_{12} + R_{11} \\
+ I_{21} + I_{22} + I_{12} + I_{11} &\leq \delta_2 + 7 + \varepsilon,
\]

\[
(8 - \varepsilon)\delta_3 - \varepsilon \leq R_{21} + R_{22} + R_{12} + R_{11} \\
+ I_{21} + I_{22} + I_{12} + I_{11} &\leq \delta_3 + 7 + \varepsilon,
\]

\[
(8 - \varepsilon)\delta_4 - \varepsilon \leq R_{21} + R_{22} + R_{12} + R_{11} \\
+ I_{21} + I_{22} + I_{12} + I_{11} &\leq \delta_4 + 7 + \varepsilon.
\]
of that cell as given in relations (19)–(22), and for any inequalities in binary decision variables of the cell and the nodes binary decision variables of the cell as equation (14), and eight inequalities in binary variables of the set of existing relations

\[
\sum_{i=1}^{N} \lambda_i = n,
\]

where \( N \) is the total number of cells in the mesh under consideration, \( n \) is the degree of the characteristic function, and \( \lambda_i \) is the binary variable assigned to the \( i \)th cell (recall that each cell has one \( \lambda \) and four \( \delta \) variables which are related as given in Equation (14)). Considering the fact that \( \lambda_i \)'s in Equation (27) are binary variables, it is concluded from this equation that exactly \( n \) out of the totally \( N \), \( \lambda_i \) variables will be equal to unity and the others will be equal to zero. In the following it is shown that when \( \lambda_i = 1 \) then exactly one root of the characteristic equation sits inside the \( i \)th cell. Note that before solving the corresponding MILP it is not known for sure which \( \lambda_i \)'s will be equated to unity and which cells are selected by the algorithm to put the roots of characteristic equation inside them. Of course, it may also happen that the problem has no solutions.

For example, in Figure 5 assuming \( \lambda_1 = 1 \) and considering the fact that \( \delta_j \)'s (\( j = 1, \ldots, 4 \)) are binary variables, it is concluded from Equation (14) that only one of the \( \delta_j \)'s in this equation can be equal to one and the others must be equal to zero. The \( \delta_j \) which is equal to one is determined only after solving the program. For now, without any loss of generality, assume that \( \delta_1 = 1 \) and \( \delta_i = \delta_3 = \delta_4 = 0 \). Substitution of these values in inequalities (15)–(18) yields:

\[
8 - 2\varepsilon \leq \overline{R}_{21} + \overline{R}_{22} + \overline{R}_{12} + \overline{R}_{11} + T_{21} + T_{22} + I_{12} + I_{11} \leq 8 + \varepsilon,
\]

\[
-\varepsilon \leq R_{21} + R_{22} + R_{12} + \overline{R}_{11} + T_{21} + T_{22} + I_{12} + I_{11} \leq 7 + \varepsilon,
\]

\[
-\varepsilon \leq R_{21} + R_{22} + R_{12} + R_{11} + T_{21} + T_{22} + I_{12} + I_{11} \leq 7 + \varepsilon,
\]

\[
-\varepsilon \leq R_{21} + R_{22} + R_{12} + R_{11} + T_{21} + T_{22} + I_{12} + I_{11} \leq 7 + \varepsilon.
\]

The binary variables cannot be determined uniquely from inequalities (29)–(31), but inequality (28) is satisfied only when \( R_{21} = R_{22} = R_{12} = R_{11} = T_{21} = T_{22} = I_{12} = I_{11} = 1 \) or equivalently, \( R_{21} = R_{11} = I_{21} = I_{22} = 0 \) and \( R_{22} = R_{12} = I_{12} = I_{11} = 1 \). (Note that inequality (28) implies the equation \( \overline{R}_{21} + \overline{R}_{22} + \overline{R}_{12} + \overline{R}_{11} + T_{21} + T_{22} + I_{12} + I_{11} = 8 \) which has a unique solution. This unique solution satisfies the relax inequalities (29)–(31) as well.) Substitution of these values in
A step-by-step algorithm for implementation of the proposed method is presented in this section. One can use the following algorithm to solve the problem under consideration without getting involved in theoretical details.

Data: \( M, m, \varepsilon \) and characteristic equation of the closed-loop system whose coefficients are affine functions of the controller parameters \( (M \) is a sufficiently large positive real constant, for example \( M = 10^4 \), \( m \) is a sufficiently large negative real constant, for example \( m = -10^4 \), and \( \varepsilon \) is a sufficiently small positive real constant, for example \( \varepsilon = 0.01 \)).

Step 1: Mesh the \( D \)-stability region under consideration in complex plane with horizontal and vertical lines. The cells of the mesh does not need to be identical.

Step 2: Assign two binary decision variables to every node and five binary decision variables to every cell of the mesh as shown in Figure 5.

Step 3: For every cell of the mesh write one equation in binary decision variables of that cell as Equation (14), and eight inequalities in binary decision variables of that cell and its nodes as (19)–(22).

Step 4: For every node of the mesh write four mixed real-binary inequalities as (23)–(26).

Step 5: For the given \( D \)-stability region write only one equation in binary variables as (27) where \( N \) is the total number of cells in the mesh and \( n \) is the degree of the characteristic equation of the closed-loop system.

Step 6: Solve the resulting MILP. If an admissible solution is found then finish the algorithm. Else, go to Step 1 and try a mesh with smaller cells.

3 | NUMERICAL EXAMPLES

Three numerical examples are presented in this section to verify the effectiveness of the proposed method. All of the following simulations are performed assuming \( M = -m = 10^4 \) and \( \varepsilon = 0.01 \). No considerable change was observed in the results as the values assigned to \( M, m \) and \( \varepsilon \) were changed reasonably.

The MILPs are solved using the \texttt{intlinprog} function of Matlab R2014b, where the lower and upper bounds of all variables in all of the following examples are considered equal to \(-10^4\) and \(10^4\), respectively.
Example 1. Consider a fractional-order (FO) process with transfer function

\[ P(s) = \frac{b_1 s^\alpha + b_2}{s + a_1 s^\alpha + a_2}, \tag{36} \]

where \( a_1 = -1.5 \), \( a_2 = -0.5 \), \( b_1 = -0.15 \), and \( b_2 = 1 \). With these values for parameters the process has one RHP pole and one non-minimum phase (NMP) zero. Hence, unavoidable initial undershoot and limitations on performance are expected. The controller under consideration is an FOPD with transfer function:

\[ C(s) = k_p + k_d \frac{s^\alpha}{1 + T s^\alpha}, \tag{37} \]

which is located in series with process in a standard error-feedback loop. In this example the time constant of the derivative filter, \( \tau \), is considered equal to 0.05. Recall that, considering the characteristic equation as \( f(s) = 1 + C(s)P(s) = 0 \), this closed-loop system is bounded-input bounded-output (BIBO) stable if and only if all roots of \( f(w) : = 1 + C(s)P(s)|_{s=w} = 0 \) lie inside the non-convex sector defined by \( [7, 8] \):

\[ C_\alpha : = \left\{ \lambda \in \mathbb{C} : |\arg(\lambda)| > \frac{\alpha \pi}{2} \right\}. \tag{38} \]

In this example two types of problems can be solved using the proposed method: the feasibility problem and the gain optimisation problem. The aim of the feasibility problem is to calculate \( k_p \) and \( k_d \) in (37) such that all roots of \( f(w) = 0 \) lie inside the non-convex \( D \)-stability region shown in Figure 7, where the border of region is identified by solid lines. Note that this\( D \)-stability region is a subset of \( C_{0.5} \). The gain optimisation problem is defined as maximising \( |k_p| \) subject to the constraint that all roots of \( f(w) = 0 \) lie inside the \( D \)-stability region under consideration. The solution obtained for the gain optimisation problem minimises the tracking and disturbance rejection errors at steady state. For this purpose we can solve both

\[ \min k_p \tag{39} \]

subject to

\[ w_i \in D : f(w_i) = 1 + C(s)P(s)|_{s=w_i} = 0, i = 1, \ldots, n, \tag{40} \]

and

\[ \min -k_p \tag{42} \]

subject to

\[ w_i \in D : f(w_i) = 1 + C(s)P(s)|_{s=w_i} = 0, i = 1, \ldots, n, \tag{43} \]

where \( D \) is the set of all points which belong to the \( D \)-stability region. After solving the above two problems, we consider the \( k_p \) with the larger amplitude and the corresponding \( k_d \) as the final solutions.

First assume \( \alpha = 0.5 \). By employing the step-by-step algorithm of Section 2.2 with the meshing shown in Figure 7, the gains of the optimal controller are obtained as \( k_p = 20.6848 \) and \( k_d = 6.7588 \). In this case the corresponding MILP consists of 2202 variables which takes about 83 s to solve using MATLAB R2014b running on an i5 2.20-GHz Intel core processor with 8GB of RAM. Locations of the closed-loop poles with this optimal FOPD controller are identified by \( \times \) in Figure 7. Interesting point is that the algorithm has placed closed-loop poles in the corners of the non-convex \( D \)-stability region which shows the importance of having access to an algorithm which can deal with non-convex regions. Solving the feasibility problem yields \( k_p = 0.7448 \) and \( k_d = 5.3092 \). The corresponding closed-loop poles are identified by \( + \) in Figure 7 in this case. The MILP of the feasibility problem is solved much faster than the optimal controller design problem, approximately in 4.26s.

For the sake of simplicity, in order to simulate the response of the closed-loop system to a step command we assume that both the controller and plant are initially at rest, which means that all initial conditions are equal to zero. Under this assumption, the well-known definitions for fractional derivative like Riemann–Liouville, Grünwald–Letnikov and Caputo lead to the same transfer functions \([25, \text{Ch. 2}]\). Then, the time-domain step response of the closed-loop system is obtained by calculating the numerical inverse Laplace transform of the FO closed-loop transfer function multiplied by \( 1/\tau \) using the method developed in \([26]\). The Matlab function for numerical evaluation of inverse
Laplace transform using this method is named `invlap.m` and can be downloaded from [27].

The response of the closed-loop system to the unit step command is shown in Figure 8 for four different cases. As is expected and also observed in this figure, the controller obtained by solving the gain maximisation problem for $\alpha = 1/2$ minimises the steady-state error compared to the controller obtained by solving the feasibility problem. This improvement in steady-state error is obtained at the cost of larger overshoots in the step response (compare the dash-dotted plot with the dotted one in Figure 8). The reason is the optimal controller locates the complex-conjugate closed-loop poles closer to the border of instability region compared to the feasible controller (compare the location of the poles identified as $\times$ with those identified as $+$ in Figure 7). No considerable overshoot is observed in the simulations of Figure 8 corresponding to $\alpha = 1/3$. The reason is that in this case the sector of stability in $\omega$-plane is $|\arg(\omega)| > \alpha \pi / 2 = \pi / 6$, and consequently, the closed-loop poles are fairly faraway from the border of instability. Recall that some initial undershoot in the response is unavoidable because of the NMP zero of process.

Example 2. Consider again the FO process of Example 1 as defined in (36) and assume $a_1 = -1.5$, $a_2 = 0.5$, $b_1 = 0.5$, $b_2 = 10$, and $\alpha = 0.5$. With these values for parameters the process has two unstable poles at 1 and 0.25. Our aim here is to design an FOPI controller with transfer function

$$C(s) = k_p + \frac{k_i}{s^\alpha},$$

such that firstly all of the closed-loop poles lie in the $D$-stability region of Figure 9, and secondly, frequency response of the closed-loop system is as close as possible to the frequency response of the desired model defined as

$$M(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}.$$
the desired closed-loop system: one with \( \zeta = 0.5 \) and \( \omega_n = 10 \text{ rad/s} \), which yields \( k_p = 0.5501 \) and \( k_i = 0.5055 \), and the other with \( \zeta = 0.5 \) and \( \omega_n = 100 \text{ rad/s} \), which yields \( k_p = 2.3234 \) and \( k_i = 8.3878 \). As it is observed in this figure, step response of the closed-loop system changes effectively according to the changes made in the transfer function of model. Poles of the closed-loop system with the resulting two different controllers are shown in Figure 9. In this figure \( + \) and \( \times \) corresponds to the models with \( \omega_n = 100 \text{ rad/s} \) and \( \omega_n = 10 \text{ rad/s} \), respectively.

Figure 11 shows the effect of the size of cells on the value of cost function as defined in (4a) and the optimal values obtained for \( k_p \) and \( k_i \). In order to calculate the data points in this figure the non-convex \( D \)-stability region is considered similar to Figure 9 and it is assumed that the nodes have a same vertical and horizontal distance from each other. Then the side of square cells is varied between 0.5 and 2 with steps of 0.1, and in each case the values of cost function, \( k_p \) and \( k_i \) are calculated. As is observed, the plot of cost function versus the length of sides is fairly flat which means that the algorithm is not so sensitive to the size of cells.

Example 3. Consider a two-input two-output process with the following state-space matrices:

\[
A_p = \begin{bmatrix} 1 & 2.5 & 1.5 & 1 \\ 0 & -2 & 1 & 1 \\ 0.01 & 0 & -3 & -0.5 \\ 0.1 & 0.2 & 0 & 0.4 \end{bmatrix}, \quad B_p = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad (49)
\]

\[
C_p = \begin{bmatrix} 1 \\ 0.5 \\ 1.5 & -2 \\ 1 & -2 \end{bmatrix}, \quad D_p = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad (50)
\]

which has four poles at 1.2065, 0.2493, \(-3.0238\), and \(-2.0320\). It can easily be verified that this system is controllable, which means that all of the closed-loop poles can be placed at the predetermined desired locations by applying a full static state feedback. But, here it is assumed that the state-feedback matrix is structured as follows:

\[
K = \begin{bmatrix} k_{11} & 0 & k_{13} & 0 \\ 0 & k_{22} & 0 & 0 \end{bmatrix}, \quad (51)
\]

and the aim is to calculate \( k_{11}, k_{13}, k_{22} \in \mathbb{R} \) (if possible) such that all poles of the closed-loop system lie inside the rectangle defined as

\[
\{ s \in \mathbb{C} : -8 < |\Re(s)| < -3, |\Im(s)| < 5.5 \}, \quad (52)
\]

in \( s \)-plane. Note that by applying the state-feedback matrix (51) we use a limited information of states to generate controls.
(state number four is not used at all and the second control is calculated only based on the information of the second state). Figure 12 shows the $D$-stability region as defined in (52), the nodes of the mesh under consideration and the closed-loop poles after solving the problem. The state-feedback gains are obtained as $k_{11} = 21.5078$, $k_{13} = 1.5573 \times 10^4$ and $k_{22} = 1.4562 \times 10^4$.

Clearly, by modifying the state-space matrices, or structure of the state-feedback matrix, or the definition of $D$-stability region we may arrive at a problem with no solutions.

**4 | CONCLUSION**

A method for designing a structurally constrained controller to achieve closed-loop $D$-stability in a desired convex/non-convex region is developed. It was especially shown that this problem is exactly equivalent to finding a solution to a set of linear algebraic equalities/inequalities in mixed integer/linear variables. Although this problem is NP-complete, the proposed formulation leads to fairly relax equalities/inequalities, which can be solved effectively by a trivial software like Matlab. In fact, problems with thousands of mixed integer/linear variables are solved successfully and the results are reported.

Three numerical example were also designed and solved by using the proposed method. In two of these examples the $D$-stability region is non-convex and the designed optimal controller locates the poles of closed-loop system in the corners of non-convex region. This observation justifies the importance of non-convex corners and having access to an algorithm that can deal with them. In the third example it was shown that the proposed method can be used to calculate the static state-feedback gains of a two-input two-output process such that all poles of the closed-loop system lie inside the pre-specified $D$-stability region. This problem is also challenging since the state-feedback matrix is assumed to be structured.

One limitation of the proposed method is that all poles of the resulting closed-loop system will be necessarily distinct. In fact, taking into account the possibility of occurring repeated poles drastically increases the complexity of the problem due to the new decision variables required for its formulation. However, it is not a big loss of generality because the distinct poles can appear arbitrarily close to each other and mimic the behaviour of repeated poles.

**REFERENCES**

1. Djafaris, T.E.: Robust Control Design: A Polynomial Approach. Springer, Berlin (1995)
2. Gutman, S.: Root Clustering in Parameter Space. Springer Berlin, Heidelberg (1990)
3. Gutman, S., Jury, E.: A general theory for matrix root-clustering in subregions of the complex plane. IEEE Trans. Autom. Control 26(4), 853–865 (1981)
4. Chen, Q., Liu, A.: D-stability and disturbance attenuation properties for networked control systems: switched system approach. J. Syst. Eng. Electron. 27(5), 1108–1114 (2016)
5. Datta, S.: Feedback controller norm optimization for linear time invariant descriptor systems with pole region constraint. IEEE Trans. Autom. Control 62(6), 2794–2806 (2017)
6. Datta, S., et al.: Reduced-order controller synthesis with regional pole constraint. Int. J. Control 89(2), 221–234 (2016)
7. Matignon, D.: Stability results for fractional differential equations with applications to control processing. Comput. Eng. Syst. Appl. 2, 963–968 (1996)
8. Matignon, D.: Stability properties for generalized fractional differential systems. ESAIM Proc. Differ. Syst.: Models Methods Appl. 5, 145–158 (1998)
9. Ibir, S.: Sufficient conditions for domain stabilisability of uncertain fractional-order systems under static-output feedbacks. IET Control Theory Appl. 11(12), 2004–2011 (2017)
10. Mohsenipour, R., Jegarkandi, M.F.: Robust D-stability analysis of fractional order interval systems of commensurate and incommensurate orders. IET Control Theory Appl. 13(8), 1039–1050 (2019)
11. Shao, K., et al.: Necessary and sufficient D-stability condition of fractional-order linear systems. 36th Chinese Control Conference (CCC) pp. 44–48, Dalian, China, 26–28 July (2017)
12. Franklin, G.F., et al.: Digital Control of Dynamic Systems, 3rd edn. Addison-Wesley, Boston (2005)
13. Grimholt, C., Skogestad, S.: Improved optimization-based design of PID controllers using exact gradients. In: Gernay, K.V., Huusom, J.K., Gani, R. (eds.) Computer Aided Chemical Engineering, vol. 37, pp. 1751–1756. Elsevier, Amsterdam (2015)
14. Grimholt, C., Skogestad, S.: Optimization of fixed-order controllers using exact gradients. J. Franklin Inst. 354(18), 8217–8231 (2017)
15. Apkarian, P., Noll, D.: Boundary control of partial differential equations using frequency domain optimization techniques. Syst. Control Lett. 135, 104577 (2020)
16. Apkarian, P., Noll, D.: Structured $H_\infty$ control of infinite-dimensional systems. Int. J. Robust Nonlinear Control 26(9), 3212–3238 (2016)
17. Datta, S., et al.: Reduced-order controller synthesis with regional pole constraint. Int. J. Control 89(2), 221–234 (2016)
18. Lu, S., et al.: Partial train speed trajectory optimization using mixed-integer linear programming. IEEE Trans. Intell. Transp. Syst. 17(10), 2911–2920 (2016)
19. Nemati, M., et al.: Optimization of unit commitment and economic dispatch in microgrids based on genetic algorithm and mixed integer linear programming. Appl. Energy 210, 944–963 (2018)
20. Wang, Y., et al.: Mixed-integer linear programming-based optimal configuration planning for energy hubs: Starting from scratch. Appl. Energy 210, 1141–1150 (2018)
21. Rivotti, P., Pistikopoulos, E.N.: A dynamic programming based approach for explicit model predictive control of hybrid systems. Comput. Chem. Eng. 72, 126–144 (2015)
22. Parisio, A., et al.: Stochastic model predictive control for economic/environmental operation management of microgrids: An experimental case study. J. Process Control 43, 24–37 (2016)
23. Bemporad, A., Morari, M.: Control of systems integrating logic, dynamics, and constraints. Automatica 35, 407–427 (1999)
24. Podlubny, I.: Fractional Differential Equations. Academic Press, Cambridge (1999)
25. Valsa, J., Brancik, L.: Approximate formulae for numerical inversion of Laplace transforms. Int. J. Numer. Model. El. 11, 153–166 (1998)
26. How to cite this article: MERRIKH-BAYAT F. Structured controller design for closed-loop $D$-stability in convex/non-convex regions: Mixed integer-linear programming approach. IET Control Theory Appl. 2021;15:77–87. https://doi.org/10.1049/cth2.12027