Matlab/Octave toolbox for structurable and robust output-feedback LQR design

Downloaded from: https://research.chalmers.se, 2020-09-14 00:51 UTC

Citation for the original published paper (version of record):
Ilka, A. (2018)
Matlab/Octave toolbox for structurable and robust output-feedback LQR design
Preprints of the 3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control, 51(4):

N.B. When citing this work, cite the original published paper.
Matlab/Octave toolbox for structurable and robust output-feedback LQR design

Adrian Ilka

Abstract: In this paper, a structurable robust output-feedback infinite horizon LQR design toolbox for Matlab and Octave is introduced. The aim of the presented toolbox is to fill the gap between available toolboxes for Matlab/Octave by extending the standard infinite horizon LQR design toolbox (from Matlab/Control System Toolbox, Octave/Control package) to robust and structurable output-feedback LQR design. The toolbox allows to design a robust infinite horizon output-feedback controller in forms like proportional (P), proportional-integral (PI), realizable proportional-integral-derivative (PID), realizable proportional-derivative (PD), realizable derivative (D), dynamic output-feedback (DOF), dynamic output-feedback with integral part (DOFI), dynamic output-feedback with integral and realizable derivative part (DOFID), and dynamic output-feedback with realizable derivative part (DOFD). In addition, the controller structure for all supported controller types is fully structurable. The toolbox relies on Yalmip (A Matlab/Octave Toolbox for Modeling and Optimization) and on linear matrix inequality solvers like SeDuMi, SDPT3, etc. Notions like “simple”, “highly customizable”, and “user-friendly” have been used and considered as main terms during the development process.

Keywords: Linear quadratic regulator, Robust control, Output-feedback, Structured controller.

1. INTRODUCTION

One of the most fundamental problems in control theory is the linear quadratic regulator (LQR) design problem (Kwakernaak and Sivan, 1972). The so-called infinite horizon linear quadratic problem of finding a control function $u^* \in \mathbb{R}^m$ for $x_0 \in \mathbb{R}^n$ that minimizes the cost functional:

$$J^* = \int_0^\infty (x(t)^T Q x(t) + u^T(t) R u(t)) + 2 x^T(t) N u(t) \, dt,$$

with $R > 0$, $Q - NR^{-1}N^T \geq 0$ subject to $\dot{x}(t) = A x(t) + B u(t)$, $x(0) = x_0$ has been studied by many authors (Kwakernaak and Sivan, 1972; Willems, 1971; Molinari, 1977; Trentelman and Willems, 1991). However, many times, it is not possible or economically feasible to measure all the state variables. Therefore, several new algorithms have been developed that resulted in generalization of the above state-feedback problem to output-feedback (Veselý, 2001; Rosinová et al., 2003; Engwerda and Weeren, 2008; Mukhopadhyay, 1978). Subsequently, the robust static output-feedback version of the LQR design has also been studied in many papers (Rosinová and Veselý, 2004; Veselý, 2005, 2006), as well as the LQR-based proportional-integral-derivative (PID) controller design (Rosinová and Veselý, 2007; Veselý and Rosinová, 2011, 2013). The introduction of linear parameter-varying (LPV) systems (Shamma, 2012) has opened new possibilities in LQR design. Several gain-scheduled/LPV-based LQR design techniques appeared in both static output feedback (SOF) and dynamic output-feedback (DOF), not to mention the PID controller design (Veselý and Ilka, 2013; Ilka and Veselý, 2014; Veselý and Ilka, 2015a; Ilka et al., 2016, 2015; Veselý and Ilka, 2015b, 2017; Ilka and Veselý, 2017a; Ilka and McKelvey, 2017; Ilka and Veselý, 2017b).

From this short literature survey follows that necessity for preparing a toolbox for LQR-based output-feedback approaches has come to the fore. The plan is to prepare and collect a bunch of functions for structurable LQR-based output-feedback controller design which can be used with Matlab and Octave as well. In this paper, one of the functions prepared for the toolbox (oflqr function) is presented. This function allows to design a robust infinite horizon output-feedback controller in forms like proportional (P), proportional-integral (PI), realizable proportional-integral-derivative (PID), realizable proportional-derivative (PD), realizable derivative (D), dynamic output-feedback (DOF), dynamic output-feedback with integral part (DOFI), dynamic output-feedback with integral and realizable derivative part (DOFID), and dynamic output-feedback with realizable derivative part (DOFD). In addition, the controller structure for all supported controller types is fully structurable. The function relies on Yalmip (Löberg, 2004) on linear matrix inequality solvers like SeDuMi (Sturm, 1999), SDPT3 (Toh et al., 1999) etc.
For the controller design, the system (2) is augmented by the introduction of the outputs for the integral part of the controller. The rest of the paper is organized into four sections. The mathematical notation of the paper is as follows. Given a symmetric matrix $P = P^T \in \mathbb{R}^{n \times n}$, the inequality $P > 0$ ($P \geq 0$) denotes the positive definiteness (semi definiteness) of the matrix. Matrices, if not explicitly stated, are assumed to have compatible dimensions. $I$ denotes the identity matrix of corresponding dimensions.

2. THEORETICAL BACKGROUND

Consider the following uncertain linear time-invariant (LTI) system with polytopic uncertainty as follows:

$$
\dot{x}(t) = A\xi(t)x(t) + B\xi(t)u(t),
$$

$$
y(t) = C\xi(t)x(t) + D\xi(t)u(t),
$$

where $x(t) \in \mathbb{R}^n$, $y(t) \in \mathbb{R}^m$, and $u(t) \in \mathbb{R}^p$ are the state, measurable output, and the control input vectors, respectively. Matrices $A\xi(t) \in \mathbb{R}^{n \times n}$, $B\xi(t) \in \mathbb{R}^{n \times m}$, $C\xi(t) \in \mathbb{R}^{l \times n}$ and $D\xi(t) \in \mathbb{R}^{l \times m}$ belong to the convex set, a polytope with $p$ vertices that can be formally defined as:

$$
\Xi := \left\{ \sum_{j=1}^{p} S_j \xi_j(t), \sum_{j=1}^{p} \xi_j(t) = 1, \xi_j(t) \geq 0 \right\}.
$$

Remark 1. In the system (2), the matrix $D\xi(t)$ can be assumed, without loss of generality, to be zero, see (Zhou et al., 1996).

The function $olfqr$ allows to design different controller types such as $P$, $I$, $PID$, $PD$, $DOF$, $DOFI$, $DOFD$ and $DOFD$. The output-feedback control law for controller type $DOFD$ can be defined as:

$$
\dot{x}_c(t) = -A_c x_c(t) - B_c y(t) - B_2 \int_0^t y_i(t) dt - B_3 y_d(t)
$$

$$
u(t) = -C_c x_c(t) - K_p y(t) - K_i \int_0^t y_i(t) dt - K_d y_d(t)
$$

where $x_c \in \mathbb{R}^{n_c}$ is the state vector of the dynamic controller, $y_i \in \mathbb{R}^{l_i}$ is the measurable output vector for the integral part, $y_d \in \mathbb{R}^{l_d}$ is the vector of filtered output derivatives using a derivative filter with filter coefficient $N_f$:

$$
G_f(s) = \frac{N_f s}{s + N_f}.
$$

Matrices $A_c \in \mathbb{R}^{n_c \times n_c}$, $B_c \in \mathbb{R}^{n_c \times l}$, $B_2 \in \mathbb{R}^{n_c \times l}$, $B_3 \in \mathbb{R}^{n_c \times l}$, $C_c \in \mathbb{R}^{m \times n_c}$, $K_p \in \mathbb{R}^{m \times l}$, $K_i \in \mathbb{R}^{m \times l}$ and $K_d \in \mathbb{R}^{m \times l}$ are the controller’s gain matrices related to the dynamic controller, furthermore $K_p \in \mathbb{R}^{m \times l}$, $K_i \in \mathbb{R}^{m \times l}$ and $K_d \in \mathbb{R}^{m \times l}$ are the proportional, integral and derivative gain matrices, respectively.

For the controller design, the system (2) is augmented (with the assumption that $D = 0$, see Remark 1) with the state vector of the dynamic controller $x_c(t)$, with integral of the outputs for the integral part $z(t) = \int_0^t y_i(t) dt$, and with the filtered outputs for the controller’s derivative part $y_f(t)$:

$$
\dot{x}(t) = A_{aug}(\xi(t))\tilde{x}(t) + B_{aug}(\xi(t))\tilde{u}(t)
$$

$$
y(t) = C_{aug}(\xi(t))\tilde{x}(t)
$$

where $\tilde{x}(t)^T = [x(t)^T, x_c(t)^T, z(t)^T, y_f(t)^T]$ is the augmented state vector, $\tilde{y}(t)^T = [y(t)^T, x_c(t)^T, z(t)^T, y_f(t)^T]$ is the augmented output vector, $\tilde{u}(t) = [u(t)^T, x_c(t)^T]$ is the augmented control input vector, and

$$
\Delta_{aug}(\xi(t)) = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 \\ B_{aug}(\xi(t)) \\ c_5 & c_6 & c_7 & c_8 \\ C_{aug}(\xi(t)) \end{bmatrix}
$$

Finally, for the controller design the control law (4) is transformed to a form:

$$
u(t) = F\tilde{y}(t) = FC_{aug}(\xi(t))\tilde{x}(t),
$$

where

$$
F = \begin{bmatrix} r_1 & c_1 & c_2 & c_3 \\ r_2 & B_{aug}(\xi(t)) & c_5 & c_6 \\ r_3 & c_7 & c_8 & c_9 \\ r_4 & C_{aug}(\xi(t)) \end{bmatrix}
$$

Remark 2. For controller types $P$, $I$, $PID$, $PD$, $DOF$, $DOFI$ or $DOFD$ one can simply neglect the unwanted parts of the integrals, $JCu(\xi(t)) \in \mathbb{R}^{l_i \times n}$ is the output matrix for the integrals, and $JCu(\xi(t)) \in \mathbb{R}^{l_d \times n}$ is the output matrix for the derivatives.

Theorem 1. For the uncertain LTI system (2) an optimal (suboptimal) stabilizing controller exists in the form (4) minimizing the cost function (1), if for the given positive definite matrix $X$, and weighting matrices $Q$, $R$ and $N$, the following problem has a solution:

$$
\min_{F,P} \text{trace} (P),
$$

subject to LMIs:

$$
M_j \leq 0, \quad j = 1, \ldots, p
$$

$$
P > 0
$$

where

$$
M_j = \begin{bmatrix} A_{aug}^T P + P A_{aug} + Q + H_j & G_j^T \\ \vec{G}_j, & R^{-1} \end{bmatrix},
$$

$$
G_j = FC_{aug} - R^{-1} \left( P_{aug} + N T \right),
$$
The first derivative of the Lyapunov function (14) is then:

\[ H_j = -(XB_{aug} + N)R^{-1}(B_{aug}^T P + N^T) \]

\[-(PB_{aug} + N)R^{-1}(B_{aug}^T X + N^T) \]

\[(+XB_{aug} + N)R^{-1}(B_{aug}^T X + N^T). \]

**Proof 1.** Let us choose the Lyapunov function as:

\[ V(t) = \tilde{x}(t)^T P \tilde{x}(t), \]

The first derivative of the Lyapunov function (14) is then:

\[ \dot{V}(\xi(t)) = \tilde{x}(t)^T P \tilde{x}(t) + \tilde{x}(t)^T P \tilde{x}(t) \]

\[ = \tilde{x}(t)^T (A_c(\xi(t))^T P + PA_c(\xi(t))) \tilde{x}(t), \]

where

\[ A_c(\xi(t)) = A_{aug}(\xi(t)) + B_{aug}(\xi(t))FC_{aug}(\xi(t)). \]

By substituting the control law (7) to the cost function (1) we can obtain:

\[ J_{\infty} = \int_0^\infty \tilde{x}(t)^T J(\xi(t)) \tilde{x}(t) dt \]

where

\[ J(\xi(t)) = Q + C_{aug}(\xi(t))^T T R F C_{aug}(\xi(t)) \]

\[ + N F C_{aug}(\xi(t)) + C_{aug}(\xi(t))^T T N^T. \]

By summarizing the equations (15) and (18) the Bellman-Lyapunov inequality can be obtained in the form:

\[ M(\xi(t)) = \dot{V}(\xi(t)) + J(\xi(t)) \leq 0. \]

Furthermore, if \( P \) is positive definite then the Bellman-Lyapunov inequality (19) can be rewritten to this form:

\[ \dot{V}(\xi(t)) + J(\xi(t)) \leq 0 \]

Integrating both sides form 0 to \( \infty \) one can obtain:

\[ J_{\infty} \leq V(0) - V(\infty) = \tilde{x}(0)^T P \tilde{x}(0). \]

It follows that by minimizing \( trace(P) \) and by satisfying \( M(\xi(t)) \leq 0 \) as well as \( P > 0 \), the closed-loop system will be quadratically stable with guaranteed cost defined by (21). In order to obtain LMI conditions, the matrix \( M(\xi(t)) \) can be rewritten to:

\[ M(\xi(t)) = A_c(\xi(t))^T P + PA_c(\xi(t) + Q + C_{aug}(\xi(t))^T T R F C_{aug}(\xi(t)) \]

\[ + N F C_{aug}(\xi(t)) + C_{aug}(\xi(t))^T T N^T. \]

Let us define:

\[ G(\xi(t)) = FC_{aug}(\xi(t)) - R^{-1}(B_{aug}(\xi(t))^T P + N^T) \]

Substituting (23) to (22) and applying the Schur complement we can obtain:

\[ M(\xi(t)) = \begin{bmatrix} M_{11}(\xi(t)), G(\xi(t))^T \end{bmatrix}, \]

where

\[ M_{11}(\xi(t)) = A(\xi(t))^T P + PA(\xi(t) + Q + H(\xi(t)), \]

\[ H(\xi(t)) = -(PB(\xi(t)) + N)R^{-1}(B(\xi(t))^T P + N^T). \]

We can linearize the nonlinear part in (26) as:

\[ \text{lin}(H(\xi(t))) = \]

\[-(XB_{aug}(\xi(t)) + N)R^{-1}(B_{aug}(\xi(t))^T P + N^T) \]

\[ -(PB_{aug}(\xi(t)) + N)R^{-1}(B_{aug}(\xi(t))^T X + N^T) \]

\[ +(XB_{aug}(\xi(t)) + N)R^{-1}(B_{aug}(\xi(t))^T X + N^T), \]

hence, we get an iterative procedure, where in each iteration holds \( X_i = P_{i-1} \) (i - actual iteration step). The iteration ends if \( |trace(P_i) - trace(P_{i-1})| \leq \epsilon \), where \( \epsilon \) can be set by the designer. Since \( M(\xi(t)) \) is convex in the uncertain parameter \( \xi \), therefore \( M(\xi(t)) \) will be negative semi-definite if and only if it is negative semi-definite at the corners of \( \xi \). Hence, semi-definiteness splits to \( \rho \) inequalities \( \rightarrow (9) \).

**Remark 4.** For the first iteration \( X_1 \) is a freely chosen positive definite matrix. It can be set by the designer or can be calculated/approximated by a standard LQR design using the nominal system.

**Remark 5.** The weighting matrices \( Q, R \) and \( N \) are also augmented since the state and control input vectors are augmented as well.

### 3. FUNCTION DESCRIPTION

The following command (in Matlab/Octave):

\[ [F,P,E] = oflqr(sys, Q, R, N, ct, opt) \]

calculates the (sub)optimal robust structurable output-feedback gain matrix \( F \) such that, for a continuous-time polytopic state-space model \( sys \), the output-feedback law defined with \( ct \) (control type: P, PI, PID, PD, D or DOF, DOFI, DOFID, DOFD) guarantees the robust closed-loop stability (quadratic stability) and minimizes the cost function (1), subject to the system dynamics:

\[ \dot{x}(t) = A x(t) + B_j u(t), \quad j = 1, \ldots, p \]

\[ y(t) = C_j x(t); \quad y(t) = C_1 x(t); \quad y(t) = C_d x(t), \]

where \( x(t), u(t) \) and \( y(t) \) are state, control input and measurable output vectors, respectively. Furthermore, \( y(t) \) and \( y(t) \) are measurable output vectors for the integral and derivative parts of the controller.

**INPUTS**

**REQUIRED:**

- **SYS** - state-space LTI systems (in convex polytopic form)
- **ct** - ct=’p’: Proportional (P) controller
  - \( u(t) = -K_p y(t), \)
  - \( F = [K_p]. \)
- **ct** =’pi’: Proportional-Integral (PI) controller
  - \( u(t) = -K_p y(t) - K_i \int_0^t y(t) dt, \)
  - \( F = [K_p,K_i]. \)
- **ct** =’pid’: Proportional-Integral-Derivative (PID) controller
  - \( u(t) = -K_p y(t) - K_i \int_0^t y(t) dt - K_d y_d(t), \)
  - \( F = [K_p,K_i,K_d]. \)
- **ct** =’pd’: Proportional-Derivative (PD) controller
\[ u(t) = -K_p y(t) - K_d y_d(t), \]
\[ F = [K_p, K_d], \]

where \( y_d \) is the vector of filtered derivatives, using derivative filter (5) (default \( N_f = 100 \)).

- **ct=’d’**: Derivative (D) controller
  \[ u(t) = -K_d y_d(t), \]
  \[ F = [K_d], \]

where \( y_d \) is the vector of filtered derivatives, using derivative filter (5) (default \( N_f = 100 \)).

- **ct=’of’**: Dynamic output-feedback with order \( n_c \) (default \( n_c = 2 \))
  \[ \dot{x}(t) = -A_c x_c(t) - B_c y(t) - B_{c2} \int_0^t y(t) \, dt, \]
  \[ u(t) = -C_c x_c(t) - K_P y(t) - K_i \int_0^t y(t) \, dt, \]
  \[ F = \begin{bmatrix} K_p, C_c, K_i \\ B_{c2}, A_c, B_{c3} \end{bmatrix}, \]

- **ct=’ofd’**: Dynamic output-feedback with filtered derivative part; order \( n_c \) (default \( n_c = 2 \))
  \[ \dot{x}(t) = -A_c x_c(t) - B_c y(t) - B_{c2} y_d(t), \]
  \[ u(t) = -C_c x_c(t) - K_P y(t) - K_d y_d(t), \]
  \[ F = \begin{bmatrix} K_p, C_c, K_d \\ B_{c2}, A_c, B_{c3} \end{bmatrix}, \]

where \( y_d \) is the vector of filtered derivatives, using derivative filter (5) (default \( N_f = 100 \)).

- **ct=’ofd i’**: Dynamic output-feedback with integral and filtered derivative part; order \( n_c \) (default \( n_c = 2 \))
  \[ \dot{x}(t) = -A_c x_c(t) - B_{c1} y(t) - B_{c2} \int_0^t y(t) \, dt, \]
  \[ u(t) = -C_c x_c(t) - K_P y(t) - K_i \int_0^t y(t) \, dt - K_d y_d(t), \]
  \[ F = \begin{bmatrix} K_p, C_c, K_i, K_d \\ B_{c2}, A_c, B_{c3} \end{bmatrix}, \]

where \( y_d \) is the vector of filtered derivatives, using derivative filter (5) (default \( N_f = 100 \)).

**OPTIMAL**: 
- **Opt.iter**: maximal number of iterations (default: 100).
- **Opt.eps**: epsilon for the stopping criteria (default: \( eps = 10^{-8} \)).
- **Opt.epsP**: epsilon for the positive definiteness test \( P \geq epsP \, I \) (default: \( epsP = 2.2204 \times 10^{-16} \)).

**OUTPOUTS**
- **F**: static output-feedback gain matrix
- **P**: Lyapunov matrix
- **E**: Closed-loop system eigenvalues

**OTHER INFO**
- **Weighting matrix size** (Q,R,N):
  - \( ct=’p’ \): \( Q(n,n), R(m,m), N(n,m) \)
  - \( ct=’pi’ \): \( Q(n+l+li,n+li), R(m,m), N(n+l+li,m) \)
  - \( ct=’pid’ \): \( Q(n+l+ld,n+2*li+ld), R(m,m), N(n+l+ld,m) \)
  - \( ct=’pd’ \): \( Q(n+l,d,n+ld), R(m,m), N(n+l,d,m) \)
  - \( ct=’d’ \): \( Q(n+l,d,n+ld), R(m,m), N(n+l,d,m) \)
  - \( ct=’of’ \): \( Q(n+nc,n+nc), R(m+nc,m+nc), N(n+nc,m+nc) \)
  - \( ct=’ofi’ \): \( Q(n+nc+li,n+nc+li), R(m+nc,m+nc), N(n+nc+li,m+nc) \)
  - \( ct=’ofid’ \): \( Q(n+nc+li+ld,n+nc+li+ld), R(m+nc,m+nc), N(n+nc+li+ld,m+nc) \)

where
- \( n \) - number of states,
- \( m \) - number of inputs,
- \( l \) - number of outputs,
- \( li \) - number of outputs for integral part (def. \( li=1 \)),
- \( ld \) - number of outputs for deriv. part (def. \( ld=1 \)),
- \( nc \) - order of the dynamic controller (def. \( nc=2 \)).

**REQUIREMENTS**
- **Matlab**: Control System Toolbox installed.
- **YALMIP** installed (R2015xxx or newer).
- **LMI solver** installed (sdpt3, sedumi, mosek, ...).
- **Octave**: Control package installed and loaded.
- **YALMIP** installed (R2015xxx or newer).
- **LMI solver** installed (sdpt3, sedumi, ...).
4. EXAMPLES

In order to show the viability of the previous proposed method, the following examples have been chosen.

Example 1. The first example is the Rosenbrock system (Rosenbrock, 1970), which will be used to demonstrate and compare the proposed method with the standard LQR design. The transfer function of the system is as follows:

\[ G(s) = \left[ \begin{array}{c} \frac{1}{s+1} \\ \frac{2}{s} \end{array} \right] \]

which can be transformed to the form (2) with matrices:

\[ A = \left[ \begin{array}{ccc} -1, & 0, & 0 \\ 0, & -3, & 0 \\ 0, & 0, & -1 \end{array} \right], \quad B = \left[ \begin{array}{c} 1, 0 \\ 0, 2 \\ 0, 1 \end{array} \right], \quad C = \left[ \begin{array}{c} 1, 1, 0, 0 \\ 0, 0, 1, 1 \end{array} \right], \quad D = \left[ \begin{array}{c} 0, 0 \end{array} \right]. \]

Different controller types were designed using the oflqr function. Beside types P, PI, PID, PD, DOF, DOFI, DOFD and DOFID, state-feedbacks like static state-feedback (SSF), dynamic state-feedback (DOF) and their variations were also designed (by changing the \( C \) matrix). Numerical solution has been carried out by SDPT3 (Toh et al., 1999) solver under OCTAVE 4.0 using YALMIP.

The obtained guaranteed cost (\( J_\infty \)) for \( x_0 = [1, 1, 1, 1] \), \( Q = I_n \), \( R = I_m \), and \( N = 0.1 \times \text{ones}(n^*, m^*) \) can be found in Table 1. (\( n^* \), \( m^* \) denotes the augmented number of states and inputs for the given control type).

<table>
<thead>
<tr>
<th>Controller type</th>
<th>( J_\infty )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard infinite-horizon LQR</td>
<td>0.9743</td>
</tr>
<tr>
<td>Proposed method:</td>
<td></td>
</tr>
<tr>
<td>SSF</td>
<td>0.9743</td>
</tr>
<tr>
<td>SSFI</td>
<td>2.8167</td>
</tr>
<tr>
<td>SSFID</td>
<td>3.5114</td>
</tr>
<tr>
<td>DSF (( n_c = 1 ))</td>
<td>1.9618</td>
</tr>
<tr>
<td>DSF (( n_c = 2 ))</td>
<td>0.9649</td>
</tr>
<tr>
<td>DSFI (( n_c = 2 ))</td>
<td>0.9554</td>
</tr>
<tr>
<td>DSFID (( n_c = 2 ))</td>
<td>2.8077</td>
</tr>
<tr>
<td>DSFD (( n_c = 2 ))</td>
<td>3.4493</td>
</tr>
<tr>
<td>Centralized P</td>
<td>1.8904</td>
</tr>
<tr>
<td>Decentralized P</td>
<td>0.9797</td>
</tr>
<tr>
<td>Centralized PI</td>
<td>1.1566</td>
</tr>
<tr>
<td>Decentralized PI</td>
<td>2.9633</td>
</tr>
<tr>
<td>Centralized PD</td>
<td>3.2784</td>
</tr>
<tr>
<td>Decentralized PD</td>
<td>3.5635</td>
</tr>
<tr>
<td>Decentralized PID</td>
<td>4.1662</td>
</tr>
<tr>
<td>Centralized PD</td>
<td>1.9695</td>
</tr>
<tr>
<td>Decentralized PD</td>
<td>2.7269</td>
</tr>
<tr>
<td>Centralized DOF (( n_c = 1 ))</td>
<td>0.9572</td>
</tr>
<tr>
<td>Decentralized DOF (( n_c = 2 ))</td>
<td>0.9607</td>
</tr>
<tr>
<td>Decentralized DOF (( n_c = 1 ))</td>
<td>1.1412</td>
</tr>
<tr>
<td>Centralized DOFI (( n_c = 2 ))</td>
<td>2.8542</td>
</tr>
<tr>
<td>Decentralized DOFD (( n_c = 2 ))</td>
<td>3.5008</td>
</tr>
<tr>
<td>Decentralized DOFID (( n_c = 2 ))</td>
<td>4.1458</td>
</tr>
<tr>
<td>Centralized DOFD (( n_c = 2 ))</td>
<td>1.8981</td>
</tr>
</tbody>
</table>

Example 2. The second example is the aircraft pitch control problem from the Control Tutorials for Matlab and Simulink (Messner et al., 2017). The state-space model for one of Boeing’s commercial aircraft is given as:

\[ \begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -0.313, & 56.7, & 0 \\ -0.0139, & -0.426, & 0 \\ 0, & 56.7, & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 0.232 \\ 0.0203 \end{bmatrix} \delta, \quad y = [0, 0, 1][\alpha, q, \theta]^T \]

where \( \alpha \) is the angle of attack, \( q \) is the pitch rate, \( \theta \) is the pitch angle and \( \delta \) is the elevator deflection angle.

The design requirements are the following: overshoot less than 10\%, rise time less than 2 seconds, settling time less than 10 seconds, steady-state error less than 2\%.

Using the LQR-based controller design approaches the controller parameter’s tuning is replaced by the tuning of the weighting parameters. This can relevantly reduce the time and complexity of the tuning process, mainly for large-scale multi-input multi-output applications. For more information and tuning approaches the readers are referred to books (Athans and Falb, 1966; Dorato et al., 2000) and references therein.

After a short iterative tuning all the requirements were fulfilled by \( Q = \text{diag}(0; 0; 0; 0; 0.6 \times 28) \), \( R = 1 \) and \( N = [0; 0; 0; 0; -4.5] \) (step response: Fig. 1) with rise time: 0.954 seconds, settling time: 9.5603 seconds, overshoot: 7.496\%, steady-state error: 0\%. The obtained realizable PID gains are: \( K_p = 5.1724, K_i = 1.7031, K_d = 3.0076 \), with filter coefficient \( N_f = 100 \).

Example 3. The third example is a simple uncertain MIMO system, which will be used to demonstrate the freedom in structurability what the oflqr can give. For example, different controller types can be designed for each subsystem at once. The system with parametric uncertainty is given as:

\[ G(s) = \left[ \begin{array}{c} \frac{1}{s} \\ \frac{2}{s+1} \end{array} \right] \]

where \( \alpha \) is the angle of attack, \( q \) is the pitch rate, \( \theta \) is the pitch angle and \( \delta \) is the elevator deflection angle. Assume that we want to design a fully decentralized controller, more precisely a PI controller for the first subsystem and a PID for the second subsystem. In order to do so, let’s define the output matrix for the derivative part just for the second subsystem: \( \text{Opt.Cd}=C(2,:) \). Finally, let us construct the structure matrix \( \text{Opt.CS} \):
Numerical solution has been carried out by SDPT3 (Toh et al., 1999) solver under OCTAVE 4.0 using YALMIP. Numerical solution has been carried out by SDPT3 (Toh et al., 1999) solver under OCTAVE 4.0 using YALMIP. Numerical solution has been carried out by SDPT3 (Toh et al., 1999) solver under OCTAVE 4.0 using YALMIP.