Control allocation in the presence of actuator dynamics and limitations

Wissam Sayssouk[1,2], Rodolfo Orjuela[1], Clement Roos[2] and Michel Basset[1]

***Abstract*— A novel linear control allocation method is presented in this paper. It combines an extended version of the classical weighted pseudo-inverse technique to account for actuator dynamics, and reference governors to cope with actuator limitations and maximize their use. Additionally, daisy-chaining is used to allow interconnection between different actuator groups of increasing priority. A comparative analysis is conducted to examine the effects of the algorithm parameters on the transient and steady-state responses of the system.**

I. Introduction

Control allocation (CA) consists of distributing control efforts in an optimal and efficient manner among multiple actuators to achieve desired performance objectives, while taking into account various operational constraints [1]. Several analytical and optimization-based techniques have been proposed [2], [3]. The latter solve the control allocation problem using online solvers, but despite their effectiveness, they usually neglect actuator dynamics, assuming that they can react instantly. As a solution, Model Predictive Control Allocation (MPCA) can be employed to take these dynamics into account, but the high computational time constitutes its main drawback [4], [5]. An alternative is Kalman Filter Control Allocation (KFCA), introduced in [6], and extended in [7] by combining KFCA with daisy-chaining (DC) to handle different actuator dynamics.

Other approaches exist in the literature to manage system constraints. An efficient one is known as Reference Governors (RG) [8]. They operate as a feedforward mechanism, ensuring state and output constraint enforcement by adjusting the reference to maintain a well-designed stable closed-loop system [9]. Different RG approaches exist, such as the Scalar Reference Governor (SRG) [8], which uses a single decision variable in a linear program to govern different channels of coupled systems. In [9], a Decoupled Reference Governor (DRG) is introduced for decoupled multi-input multi-output systems. A few papers also exist, which combine CA and RG. In [11], a novel method is proposed based on RG and a robust CA approach to address physical limitations and uncertainties, while neglecting actuator dynamics. In [12], the authors employ RG as a feedforward mechanism and CA to distribute the remaining actuator efforts following actuator faults, considering actuator reliability.

In this context, this paper introduces a control allocation strategy that takes into account both actuator dynamics and limitations. The Enhanced Weighted Pseudo-Inverse (EWPI) is first introduced. This novel analytical approach to account for actuator dynamics is an extension of the classical Weighted Pseudo-Inverse (WPI) technique, and allows an easy adjustment of the actuator response times. Then, our work departs from the conventional usage of RG as a feedforward mechanism, and integrates it directly into the control loop. The resulting combination of EWPI and RG in a novel control allocation architecture offers the advantage of a low computational time during execution, and therefore allows to optimize real-time constraint management. Finally, daisy-chaining is used to facilitate dynamic interactions between actuators, organizing the exchange of the unrealized commands between different groups of increasing priority.

The paper is organized as follows. Section II outlines the CA problem and introduces the classical WPI approach. Section III presents the proposed EWPI algorithm. Section IV presents RG, starting with the presentation of the existing SRG, followed by its extension in alignment with the proposed approach, including the daisy-chaining technique. Section V concludes the paper by presenting numerical examples that assess the performance of each part of the proposed control allocation architecture.

II. Control Allocation

A. Problem Formulation

The control allocation layer is essential to split the control design into two parts: a high-level control law to generate the virtual control input v_{desire} (typically forces/ moments), and the allocation part itself to distribute v_{desire} among available actuators. This has many benefits like easy reconfiguration in case of actuator change, since control allocation is independent of the high-level controller design.

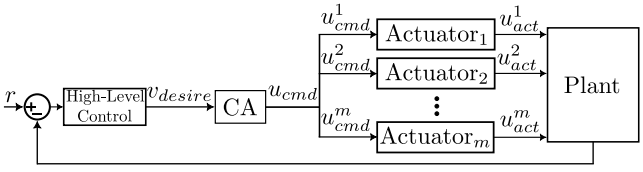

Fig. 1. Control allocation scheme

Let's start by neglecting actuator dynamics, *i.e.* $u^i_{act} = u^i_{cmd} = u^i$, and consider the equation:

$$v_{desire}(k) = Bu(k), \tag{1}$$

where $k \in \mathbb{N}$, $v_{desire} \in \mathbb{R}^f$ is the desired virtual control input, $B \in \mathbb{R}^{f \times m}$ the control effectiveness matrix and

[1] IRIMAS UR 7499, Université de Haute-Alsace, 68093 Mulhouse, France, name.surname@uha.fr

[2] ONERA - The French Aerospace Lab, Information Processing and Systems Department, 31400 Toulouse, France, name.surname@onera.fr

$u = \left[u^i \dots u^m\right]^T \in \mathbb{R}^m$ is the control input. The vector u is usually subject to position u^p and rate limits u^r as follows:

$$\underline{u}(k) \leq u(k) \leq \overline{u}(k), \tag{2}$$

where:

$$\begin{aligned} \underline{u}(k) &= \min\{u^p_{min}, u(k-T_e) + T_e\, u^r_{min}\}, \\ \overline{u}(k) &= \max\{u^p_{max}, u(k-T_e) + T_e\, u^r_{max}\}, \end{aligned} \tag{3}$$

and T_e is the sampling time. Solving the control allocation problem consists of finding a control input u that satisfies equations (1) and (2). When the system is over-actuated, the number of actuators m is larger than the number of degrees of freedom f to be controlled.

B. Weighted Pseudo-Inverse

WPI is an unconstrained control allocation technique that minimizes the weighted 2-norm of the control input vector u, neglecting the actuator dynamics and limitations [2]. It is the solution that minimizes the following cost function:

$$J(u(k)) = \left\| W^{-1/2} u(k) \right\|^2 \text{ s.t. } \quad v_{desire}(k) = Bu(k), \tag{4}$$

where $W \in \mathbb{R}^{m \times m}$ is a diagonal positive semidefinite matrix. The analytical solution can be derived by using Lagrange multipliers. Consider the following scalar function:

$$L(u(k), \lambda) = \left\| W^{-1/2} u(k) \right\|^2 + \lambda^T (v_{desire}(k) - Bu(k)), \tag{5}$$

where $\lambda \in \mathbb{R}^{f \times 1}$ is a Lagrange multiplier to be determined. The optimality conditions are derived from the partial derivatives of L as follows:

$$\frac{\partial L}{\partial u} = 2u(k)^T W^{-1} - \lambda^T B = 0, \tag{6a}$$

$$\frac{\partial L}{\partial \lambda} = v_{desire}(k) - Bu(k) = 0. \tag{6b}$$

Multiplying (6a) on the left by BW and using (6b), the following result is obtained:

$$2v_{desire}(k) - BWB^T \lambda = 0, \tag{7}$$

and consequently:

$$\lambda = 2(BWB^T)^{-1} v_{desire}(k). \tag{8}$$

Replacing λ from (8) in (6a) finally leads to:

$$u(k) = WB^T (BWB^T)^{-1} v_{desire}(k). \tag{9}$$

C. Actuator Modeling

Let us, now, represent actuator dynamics as a first order system, described by the following transfer function:

$$\frac{u^i_{act}(s)}{u^i_{cmd}(s)} = \frac{K_i}{\tau_i s + 1}, \tag{10}$$

where τ_i and K_i are the time constant and the gain of the i^{th} actuator. u^i_{cmd} and u^i_{act} are the i^{th} commanded and realized control inputs, as shown in Fig. 1, with $i = 1, \dots, m$.

The transfer function (10) is discretized using the first order Euler method to be used later in the control allocation formulation:

$$u^i_{act}(k+1) = u^i_{act}(k) + \frac{T_e}{\tau_i}(K_i u^i_{cmd}(k) - u^i_{act}(k)). \tag{11}$$

Let $K = \text{diag}\,(K_1, \dots, K_m)$ and $\tau = \text{diag}\,(\tau_1, \dots, \tau_m)$.

III. Enhanced Weighted Pseudo-Inverse

To account for actuator dynamics, instead of minimizing the norm of the control input $u = u_{cmd}$, as in Section II-B, the control allocator can be designed to minimize the actuator output u_{act}. This choice allows anticipating the actuator dynamics.

To develop the proposed control allocator, consider the following cost function:

$$\begin{aligned} &J(u_{act}(k+1)) = \left\| W^{-1/2} u_{act}(k+1) \right\|^2 \\ &\text{s.t.} \quad v_{desire}(k) = Bu_{act}(k+1), \end{aligned} \tag{12}$$

where $u_{act}(k+1)$ is defined in (11). Analogously to the preceding case, consider the following scalar function:

$$\begin{aligned} L(u_{cmd}(k), \lambda) = &\left\| W^{-1/2} u_{act}(k+1) \right\|^2 \\ &+ \lambda^T (v_{desire}(k) - Bu_{act}(k+1)), \end{aligned} \tag{13}$$

where:

$$u_{act}(k+1) = u_{act}(k) + \eta (Ku_{cmd}(k) - u_{act}(k)), \tag{14}$$

and $\eta = \tau^{-1} T_e$. This leads to:

$$\begin{aligned} &L(u_{cmd}(k), \lambda) = u_{act}(k+1)^T W^{-1} u_{act}(k+1) \\ &+ \lambda^T \left(v_{desire}(k) - B\left(u_{act}(k) + \eta (Ku_{cmd}(k) - u_{act}(k))\right)\right). \end{aligned} \tag{15}$$

The optimality conditions are derived from the partial derivatives of (13) as follows:

$$\begin{aligned} \frac{\partial L}{\partial u_{cmd}} &= 2\eta^2 KW^{-1} Ku_{cmd}(k) \\ &\quad + 2(1-\eta)\eta KW^{-1} u_{act}(k) - \eta KB^T \lambda = 0, \\ \frac{\partial L}{\partial \lambda} &= v_{desire}(k) - B(u_{act}(k) + \eta (Ku_{cmd}(k) - u_{act}(k))) = 0. \end{aligned}$$

Applying the optimality condition $\dfrac{\partial L}{\partial u_{cmd}} = 0$ leads to:

$$2KW^{-1}(\eta Ku_{cmd}(k) + (1-\eta) u_{act}(k)) - KB^T \lambda = 0, \tag{16}$$

$$\Leftrightarrow 2W^{-1} u_{act}(k+1) - B^T \lambda = 0, \tag{17}$$

$$\Leftrightarrow \lambda = 2(BWB^T)^{-1} v_{desire}(k). \tag{18}$$

Injecting (18) into (16) finally allows to compute the value of $u_{cmd}(k)$ which minimizes $J(u_{act}(k+1))$ in (12):

$$\begin{aligned} &u_{cmd}(k) = \\ &K^{-1}\left(\eta^{-1}\left(WB^T (BWB^T)^{-1} v_{desire}(k) - u_{act}(k)\right) + u_{act}(k)\right). \end{aligned} \tag{19}$$

In practice, the execution of the requested virtual control input v_{desire} at the next sampling time, as requested in (12), poses a significant challenge. Indeed, it constitutes a

hard constraint that cannot be satisfied at all times, due to actuator limitations that are not considered for the moment. To address this problem, a possible solution is to execute v_{desire} over n sampling periods instead of 1. This approach makes it easier to handle strict execution requirements. A way to do that is to rewrite the cost function (12) as follows:

$$\begin{aligned} &J(u_{act}(k+1)) = \left\| W^{-1/2} u_{act}(k+n) \right\|^2 \\ &\text{s.t.} \quad v_{desire}(k) = B u_{act}(k+n), \end{aligned} \tag{20}$$

but the resulting expression of $u_{cmd}(k)$ becomes fairly complicated. In order to keep a straightforward expression and retain the previous formalism, $u_{cmd}(k)$ is instead defined as:

$$\begin{aligned} &u_{cmd}(k) = \\ &K^{-1}\Big(\gamma^{-1}\eta^{-1}\Big(WB^T(BWB^T)^{-1}v_{desire}(k) - u_{act}(k)\Big) + u_{act}(k)\Big), \end{aligned} \tag{21}$$

where γ is a positive definite diagonal matrix representing the n sampling periods.

The choice of γ is important and depends on the application, as it has an impact on the transient response of the actuator. It can decrease or increase the actuator response with respect to its bandwidth, which corresponds to $\gamma < \eta^{-1}$ and $\gamma > \eta^{-1}$ respectively. Note that when $K = I^m$ and $\gamma = \eta^{-1}$, the weighted pseudo-inverse solution discussed in Section II-B is recovered. Moreover, the matrix multiplication in (21) is computationally simple and can be performed online, which allows a real-time implementation of the method.

IV. Reference Governor in the loop

The scalar reference governor (SRG) introduced in [8] is first recalled in this section. This is followed by a detailed discussion on the theoretical and technical extensions needed to integrate it into the EWPI control allocation scheme.

A. Reference Governor

The Reference Governor (RG) is a control mechanism used to ensure that the system's controlled variables do not deviate from a desired reference. The Scalar Reference Governor (SRG) is highlighted in this section, as it requires less computational time compared to other approaches.

Consider the following stable closed-loop discrete-time state-space system:

$$\begin{aligned} x_{cl}(k+1) &= A^{\dagger} x_{cl}(k) + B^{\dagger} v(k) \\ y_{cl}(k) &= C^{\dagger} x_{cl}(k), \end{aligned} \tag{22}$$

where $x_{cl}(k) \in \mathbb{R}^n$ is the closed-loop state vector, $v(k) \in \mathbb{R}^l$ is the governed reference, and $y_{cl}(k) \in \mathbb{Y}$ is the constrained output, where $\mathbb{Y} \subset \mathbb{R}^p$ is a specified set defining the constraints.

To compute $v(k)$, the SRG employs the maximal admissible set (MAS) O_∞. The latter is defined as the set of all states x_{cl} and inputs v such that the predicted response from the initial state x_{cl} and with the input v kept constant satisfies the constraints:

$$O_\infty = \{(v, x_{cl}) : \tilde{y}_{cl}(k) \in \mathbb{Y}, \forall k \in \mathbb{Z}_+\}, \tag{23}$$

where $\tilde{y}_{cl}$ is the output prediction of system (22), defined as:

$$\tilde{y}_{cl}(k) = C^{\dagger} A^{\dagger^k} x_{cl}(k) + C^{\dagger}(I - A^{\dagger})^{-1}(I - A^{\dagger^k}) B^{\dagger} v(k). \tag{24}$$

Based on the currently available state $x_{cl}(k)$, the reference governor scheme computes $v(k)$ so that $(v(k), x_{cl}(k)) \in P$, where $P \subseteq O_\infty$. $P = O_\infty$ is theoretically acceptable, but for computational reasons, the most common preference is $P = \tilde{O}_\infty$, where $\tilde{O}_\infty$ is a tightened version of O_∞. This is achieved by constraining the command v so that the corresponding steady-state output $\bar{y}_{cl} = C^{\dagger}(I - A^{\dagger})^{-1} B^{\dagger} v$ satisfies the constraints $\mathbb{Y}$ with a nonzero margin $\varepsilon > 0$, *i.e.*:

$$\tilde{O}_\infty = O_\infty \cap O^{\varepsilon}, \tag{25}$$

where:

$$O^{\varepsilon} = \{(v, x_{cl}) : \bar{y}_{cl} \in (1 - \varepsilon)\mathbb{Y}\}. \tag{26}$$

Using the above, $\tilde{O}_\infty$ can be computed to be a polytope of the form:

$$\tilde{O}_\infty = \{(v, x_{cl}) : H_x x_{cl} + H_v v \leq h\}, \tag{27}$$

where the matrices H_x, H_v and h are defined in [10].

The SRG computes, at each time instant and based on the current state $x_{cl}(k)$, a governed reference $v(k)$. The latter is the best approximation of the desired reference $r(k)$ along the line segment connecting its previous value $v(k-1)$ and the current reference $r(k)$, ensuring $(v(k), x_{cl}(k)) \in \tilde{O}_\infty$. Formally, the SRG solves the following linear programming (LP) problem at each time instant:

$$\begin{aligned} \kappa(k) = \max_{0 \leq \kappa \leq 1} \ & \kappa \\ \text{s.t.} \quad & v(k) = v(k-1) + \kappa(r(k) - v(k-1)) \\ & (v(k), x_{cl}(k)) \in \tilde{O}_\infty, \end{aligned} \tag{28}$$

where κ represents a scalar adjustable parameter. If no constraint is violated when applying $r(k)$, κ is set to 1, and $v(k) = r(k)$. However, if $r(k)$ leads to constraint violation, the RG adjusts the value of κ accordingly. In the extreme case, κ becomes 0, *i.e.* $v(k) = v(k-1)$, indicating that the RG temporarily isolates the system from further variations of the reference r, so as to avoid constraint violation.

B. Reference Governor in a Control Allocation Framework

The previous section presented the RG as a feedforward mechanism. In the present work, the application of RG is extended beyond its conventional use by integrating it directly into the control loop. This extension involves a synthesis that merges the EWPI control allocation approach with the RG, as depicted in Fig. 2. The methodology includes the offline calculation of the set $\tilde{O}_\infty$ defined in (27). To achieve this, the EWPI control allocation is initially coupled to the actuators, as depicted in Fig. 3. This integration allows the prediction of the dynamic response of the actuators in the presence of EWPI, facilitating the computation of $\tilde{O}_\infty$. Once this set is determined, it is incorporated into the RGs, as shown in Fig. 2, to predict potential constraint violations and filter the commands, u_{cmd}, computed using the EWPI. This process results in generating a feasible $\bar{u}_{cmd}$. The principles

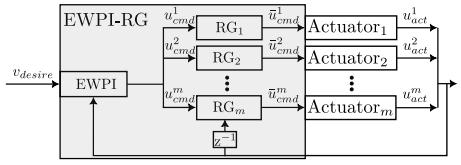

Fig. 2. EWPI-RG control allocation scheme

guiding the proposed control allocation strategy are described in more detail below.

Consider the closed-loop discrete-time system illustrated in Fig. 2, where u^i_{cmd} and $\bar{u}^i_{cmd}$, $i = 1,\ldots,m$, denote actuator inputs and saturated inputs respectively, and u^i_{act} are the constrained actuator outputs. A MAS denotes $\tilde{O}^i_\infty$ is computed for each actuator. As already mentioned above, the sets $\tilde{O}^i_\infty$ are computed offline, enabling the prediction of the future behavior of the closed-loop state x_{cl} online by verifying that the set condition (26) is satisfied. To compute the sets $\tilde{O}^i_\infty$,

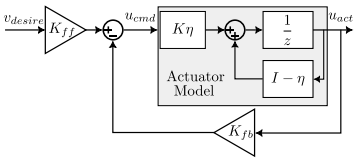

Fig. 3. Control scheme used to calculate $\tilde{O}_\infty$

the actuator model (11) is used in discrete-time mounted with the EWPI CA, as shown in Fig. 3. The matrices, K_{ff} and K_{fb}, are obtained from (21), as follows:

$$\begin{aligned} K_{ff} &= K^{-1}\gamma^{-1}\eta^{-1}WB^T(BWB^T)^{-1}, \\ K_{fb} &= K^{-1}\left(\gamma^{-1}\eta^{-1} - I^m\right). \end{aligned} \tag{29}$$

In practical applications, actuators are subjected to physical position and rate limits, denoted as u_{act} and u^r_{act} in discrete time, respectively. These constraints are considered in the set condition $x_{cl} \in \tilde{O}_\infty$, enabling the prediction of future behavior and constraints violations before they appear. The control scheme illustrated in Fig. 3 involves a state-space representation of u_{act} expressed in terms of the calculated v_{desire}, which can be represented as follows:

$$u_{act}(k+1) = \left[I^m - \eta - K\eta K_{fb}\right]u_{act}(k) + \left[K\eta K_{ff}\right]v_{desire}(k). \tag{30}$$

Furthermore, the rate limit u^r_{act} is explicitly defined as the difference between the current value of u_{act} and its previous value in discrete time, resulting in:

$$\begin{aligned} u^r_{act}(k+1) &= u_{act}(k+1) - u_{act}(k), \\ u^r_{act}(k+1) &= \left[-\eta - K\eta K_{fb}\right]u_{act}(k) + \left[K\eta K_{ff}\right]v_{desire}(k). \end{aligned} \tag{31}$$

To summarize, the discrete state-space representation of x_{cl} is described as follows:

$$\begin{aligned} x_{cl}(k+1) &= \overbrace{\begin{bmatrix} I^m - \eta - K\eta K_{fb} & 0^m \\ -\eta - K\eta K_{fb} & 0^m \end{bmatrix}}^{A^\dagger} x_{cl}(k) + \overbrace{\begin{bmatrix} K\eta K_{ff} \\ K\eta K_{ff} \end{bmatrix}}^{B^\dagger} v_{desire}(k) \\ y_{cl}(k) &= \underbrace{\begin{bmatrix} I^m & 0^m \\ 0^m & I^m \end{bmatrix}}_{C^\dagger} x_{cl}(k), \end{aligned} \tag{32}$$

where I^m and O^m represent the identity and zero matrices, respectively.

To calculate the set $\tilde{O}^i_\infty$ for each actuator, x_{cl} in (32) is decomposed into $x^i_{cl},\ldots,x^m_{cl}$, and prior knowledge of the position and rate limits is required. During the online operation, only actuator output measurements are then required. $\tilde{O}^i_\infty$ is represented by polytopes of the form (27). Let k^* denotes the number of rows of H_x, H_v, and h represented in (33). This k^* represents the upper limit for the prediction horizon to be constructed.

$$\begin{aligned} H^i_x &= \begin{bmatrix} 0 \\ S_i C^\dagger_i \\ S_i C^\dagger_i A^\dagger_i \\ \vdots \\ S_i C^\dagger_i A^{\dagger k^*}_i \end{bmatrix}, H^i_v = \begin{bmatrix} S_i(C^\dagger_i(I_2 - A^\dagger_i)^{-1}B^\dagger_i) \\ 0_2 \\ S_i(C^\dagger_i(I_2 - A^\dagger_i)(I_2 - A^\dagger_i)^{-1}B^\dagger_i) \\ \vdots \\ S_i(C^\dagger_i(I_2 - A^{\dagger k^*}_i)(I_2 - A^\dagger_i)^{-1}B^\dagger_i) \end{bmatrix}, \\ h^i &= \begin{bmatrix}(1-\varepsilon_i)s_i & s_i & s_i & \ldots & s_i\end{bmatrix}^T. \end{aligned} \tag{33}$$

$A^\dagger_i$, $B^\dagger_i$, $C^\dagger_i$ are the i^{th} block of the block-diagonal matrices $A^\dagger$, $B^\dagger$ and $C^\dagger$. S_i and s_i represent the polytopic set $S^i y^i_{cl} \le s^i$ associated to the position and rate constraints. The choice of these parameters depends on the constraints imposed by the actuators. Once the MAS are calculated, the SRG denoted as RG$_i$ in Fig. 2 only requires solving online the LP problem (28). This can be done implicitly through LP solvers or explicitly. For low computational time, the explicit algorithm used to solve the LP is as follows [9]:

Algorithm 1: Explicit SRG Algorithm

```
let a = H^i_v (u^i_cmd(k) − ū^i_cmd(k−1))
let b = h^i − H^i_x x^i_cl(k) − H^i_v ū^i_cmd(k−1)
set κ = 1
for j = 1 to k* do
    if a(j) > 0 then
        κ = min(κ, b(j)/a(j))
    end
end
κ^i(k) = max(κ, 0)
```

C. Daisy-Chaining Reference Governor

The daisy-chaining technique allows actuators to be organized according to distinct priority levels and assigns control inputs to them sequentially. If a group becomes saturated, the unrealized command is then sent to the next group. This process continues until all actuators are used actively.

As previously presented, the SRG operates in a decoupled manner, with no interconnection between actuator groups. Daisy-chaining is therefore coupled with reference governor in this section to address this limitation. The concept is as follows. The m control inputs are divided into M groups of decreasing priority:

$$u_{cmd} = \begin{bmatrix} U^1_{cmd} & \ldots & U^M_{cmd} \end{bmatrix}^T, \tag{34}$$

$$u_{act} = \begin{bmatrix} U^1_{act} & \ldots & U^M_{act} \end{bmatrix}^T, \tag{35}$$

where U^i_{cmd} and U^i_{act} consist of the commanded and realized control inputs of the i^{th} group. In line with this, the control effectiveness matrix is divided as follows:

$$B = \begin{bmatrix} B_1 & \ldots & B_M \end{bmatrix}, \tag{36}$$

and the control allocation problem is now expressed as:

$$v_{desire} = B_1 U^1_{act} + \ldots + B_M U^M_{act}.$$

If only the first actuator group is used, *i.e.* $U^2_{act} = 0, \ldots, U^M_{act} = 0$, the solution computed using the EWPI is given by (19), where u_{cmd} and u_{act} are replaced with U^1_{cmd} and U^1_{act}. If U^1_{cmd} satisfies the constraints, *i.e.* $U^1_{cmd} = \bar{U}^1_{cmd}$, the allocation is successful and the algorithm is interrupted. Otherwise, the unrealized virtual control input $v_{desire} - B_1 U^1_{act}$ is sent to the second actuator group. In practice, the scalar adjustable parameter κ is used to interconnect daisy-chaining with reference governor. If $\kappa = 1$, no saturation is predicted and the first actuator group is sufficient. But if $\kappa \in [0\ 1[$, the following control allocation problem is now to be solved:

$$v_{desire} - B_1 U^1_{act} = B_2 U^2_{act}, \tag{37}$$

where U^1_{act} is fixed to the previously computed value, and U^2_{act} is obtained using (19). The same process is repeated with the other groups until the unrealized virtual control input becomes zero or all actuator groups are used.

V. Numerical Examples

In this section, a scalar virtual control input v_{desire} is distributed among two first-order actuators described by (11), with gain $K_i = 1$, time constant $\tau_i = 0.4$ s, and sampling time $T_e = 10$ ms. Four different approaches are compared: WPI, EWPI and EWPI-RG w/o daisy-chaining.

Case 1 (EWPI vs WPI)
In this first comparison between WPI and EWPI, the same weighting matrix $W = I^{2\times 2}$ is used for both approaches. The effectiveness matrix B is set to $[1\ 1]$, which ensures a uniform distribution among the two actuators, and v_{desire} is set to 10. Three different values of γ are used for the EWPI:

1) $\gamma = \text{diag}([20,\ 20]) < \eta^{-1}$,
2) $\gamma = \text{diag}([40,\ 40]) = \eta^{-1}$,
3) $\gamma = \text{diag}([60,\ 60]) > \eta^{-1}$.

Referring to Fig. 4, the actuator outputs converge toward 5 in all cases, as expected, given that v_{desire} is equally distributed between the two actuators. Additionally, by adjusting γ, actuator responses slower, similar or faster than with WPI can be observed. This variability in γ enables the modification of the transient response of the actuators.

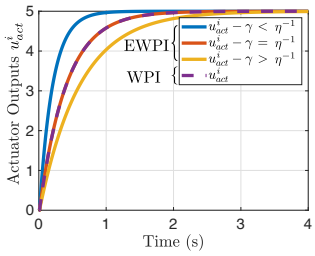

Fig. 4. WPI vs EWPI

Moreover, the EWPI approach offers the capability to selectively speed up or slow down the response of one actuator relative to another based on the choice of γ. To illustrate this, let $\gamma = \text{diag}([\gamma_1,\ \gamma_2])$, where $\gamma_1 < \frac{\tau_1}{T_e}$ and $\gamma_2 = \frac{\tau_2}{T_e}$. As depicted in Fig. 5, both actuator outputs converge to 5, indicating that the requested virtual control input v_{desire} is completely executed, but the first exhibits a faster response than the second.

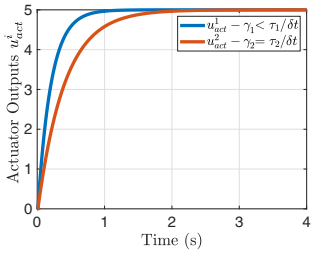

Fig. 5. EWPI for different values of γ

Case 2 (EWPI-RG)
The second example aims to demonstrate the benefits of integrating the reference governor in the EWPI framework, so as to take into account the position and rate limits of the actuators. In the two scenarios considered, the weighting function and the effectiveness matrix are $W = \text{diag}([1,\ 1])$ and $B = [1,\ 1]$, which means that a similar distribution is expected for both actuators. The tuning parameter γ is set to $\text{diag}([20,\ 20])$, and the position and rate limits are as follows:

$$u_{max} = \begin{bmatrix} 30 & 20 \end{bmatrix}, \quad u^r_{max} = \begin{bmatrix} 2 & 2 \end{bmatrix}. \tag{38}$$

In the first scenario, denoted as case 2.1, the virtual control input v_{desire} is set to 20, which is achievable since $v_{\text{desire}} < Bu_{max}$. This allows to assess the proposed approach when position limits do not occur. As illustrated in Fig. 6, Bu_{act} converges to v_{desire} and v_{desire} is distributed equally among both actuators as shown in Fig. 7.

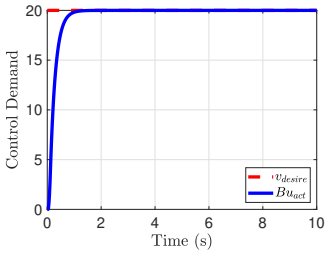

Fig. 6. Control allocation effectiveness in case 2.1

Furthermore, Fig. 7 shows that the position and rate limits imposed on the actuator inputs are well respected.

For the second scenario, denoted as case 2.2, v_{desire} is set to 70. In this case, $v_{desire} > Bu_{max}$, and Fig. 8 shows that this desired value cannot be reached. Fig. 9 confirms that both actuators reach their position limits. Furthermore, it can be checked that neither position not rate limits are violated.

Case 3 (EWPI-RG w/o daisy-chaining)
The performance of EWPI-RG is now compared w/o daisy-chaining. The same example is used as in case 2, with the only difference that v_{desire} is set to 45. As $v_{desire} < Bu_{max}$, it can be realized by both actuators. But as depicted in

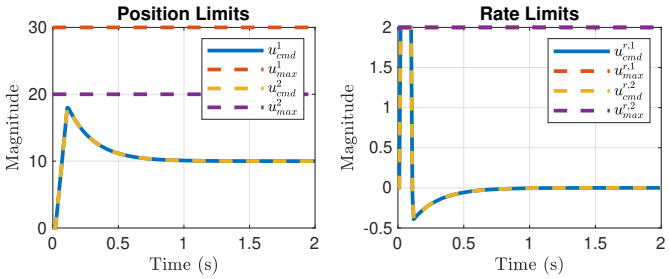

Fig. 7. Position and rate limits in case 2.1

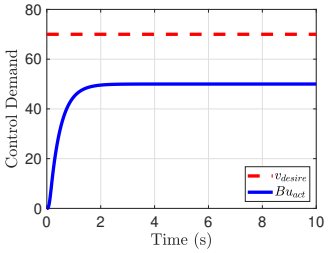

Fig. 8. Control allocation effectiveness in case 2.2

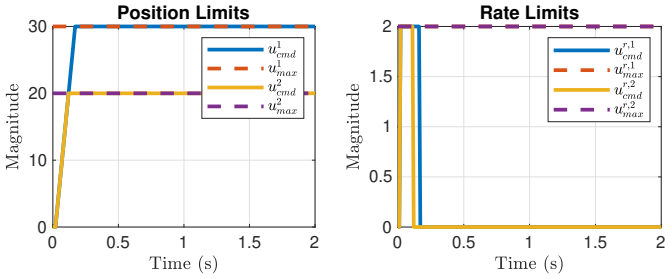

Fig. 9. Position and rate limits in case 2.2

Fig. 10, a steady-state error is observed when daisy-chaining is not used due to the absence of interconnection between the different actuators. On the contrary, the control input Bu_{act}^{RG-DC} realized with daisy-chaining exactly reaches v_{desire}. Furthermore, as illustrated in Fig. 11, both actuators operate within their position and rate limits when daisy-chaining is used. All this demonstrates the benefits of the proposed approach.

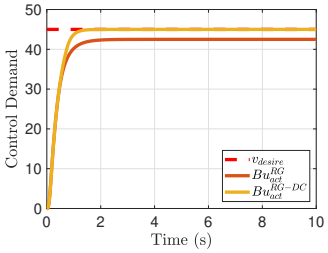

Fig. 10. EWPI-RG w/o daisy-chaining

It is also noteworthy that reference governor is not solely employed to address actuator constraints. As seen in cases 2 and 3, it also enables the use of the maximum performance potential of the actuators. This is due to the prediction

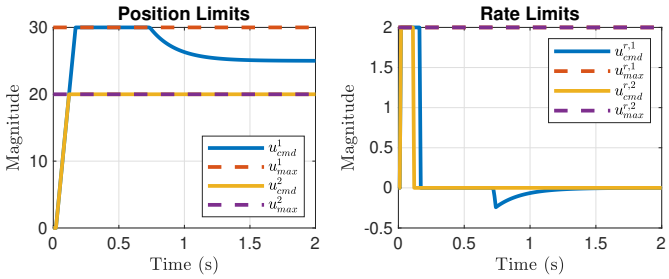

Fig. 11. Position and rate limits in case 3 with daisy-chaining

capabilities of RG in managing saturation and the choice of γ in the EWPI.

VI. CONCLUSION

This study explores control allocation strategies capable of taking actuator dynamics into account. The proposed approach combines an enhanced version of the classical weighted pseudo-inverse technique with reference governors and daisy-chaining. Its effectiveness is demonstrated in several scenarios, showing that actuator constraints can be satisfied while maximizing system performance. The incorporation of RG introduces a predictive element for effective position/rate saturation management, while daisy-chaining enhances control input distribution, therefore improving the system response.

REFERENCES

[1] Johansen, T. and Fossen, T. (2013). Control allocation - A survey. Automatica, 49(5), 1087–1103.

[2] Bodson, M. (2002). Evaluation of optimization methods for control allocation. Journal of Guidance, Control, and Dynamics, 25(4), 703–711.

[3] Oppenheimer, M., Doman, D., and Bolender, M. (2010). Control allocation. The Control Handbook, Control System Applications.

[4] Chatrath, K., Zheng, Y., and Shyrokau, B. (2020). Vehicle dynamics control using model predictive control allocation combined with an adaptive parameter estimator. SAE International Journal of Connected and Automated Vehicles, 3(2), 103-117.

[5] Kissai, M., Monsuez, B., Mouton, X., Martinez, D., Tapus, A. (2019). Model predictive control allocation of systems with different dynamics. Proceedings of the IEEE Intelligent Transportation Systems Conference, Auckland, New Zealand, 4170–4177.

[6] Morani, G., Nebula, F., Corraro, F., and Ariola, M. (2019). Dynamic control allocation through Kalman filtering. American Journal of Engineering and Applied Sciences, 12, 46–56.

[7] Sayssouk, W., Orjuela, R., Cassaro, M., Roos, C. and Basset, M. (2023). Daisy chaining Kalman filter control allocation. Proceedings of the 9th International Conference on Control, Decision and Information Technologies (CoDIT), Rome, Italy.

[8] Garone, E., Di Cairano, S., Kolmanovsky, I. (2017). Reference and command governors for systems with constraints: A survey on theory and applications. Automatica, 75(1), 306-328.

[9] Liu, Y., Osorio, J. and Ossareh, H. (2018). Decoupled reference governors for multi-input multi-output systems. Proceedings of the IEEE Conference on Decision and Control (CDC), Miami, FL, USA, 1839-1846.

[10] Liu, Y. (2022). Reference governors for MIMO systems and preview control: Theory, algorithms, and practical applications. PhD Thesis, University of Vermont.

[11] De Almeida, F. (2016). Robust off-line control allocation. Aerospace Science and Technology. 52, 1–9.

[12] Weber, P. et al. (2012). Reconfigurable control design with integration of a reference governor and reliability indicators. International Journal of Applied Mathematics and Computer Science. 22. 139-148.