A dynamic multi-axis control allocation scheme for real-time applications

Edouard Sadien, Clément Roos, Abderazik Birouche, Mathieu Carton, Christophe Grimault, Louis-Emmanuel Romana, Michel Basset

A dynamic multi-axis control allocation scheme for real-time applications

E. Sadien[a], C. Roos[b], A. Birouche[c], M. Carton[a], C. Grimault[a], L-E. Romana[a] and M. Basset[c]

[a]Airbus Operations S.A.S., Toulouse, France; [b]ONERA The French Aerospace Lab, Toulouse, France; [c]IRIMAS, Université de Haute-Alsace, Mulhouse, France

ARTICLE HISTORY
Compiled January 12, 2023

ABSTRACT
The Dynamic Weighting Control Allocator (DWCA) was introduced by Sadien *et al.* (Control Engineering Practice, 2019) to solve the allocation problem raised by the yaw control of an on-ground aircraft. It handles the classical tradeoff between virtual reference inputs realisation and control power minimization. But it also offers three specific features. First, the actuators reach saturation almost simultaneously, which allows an efficient recovery in case of failure. Then, several industrial requirements are taken into account, such as implementation ease, low computational cost and compatibility with certification constraints. And finally, the actuators can be prioritised, the last ones being used sparingly (to avoid overheating, fatigue, maintenance cost...) only when the virtual reference inputs cannot be realised by the first ones only. But despite its attractiveness, the DWCA is limited to the 1-dimensional case, which means that only one degree of freedom can be controlled. This is not sufficient to deal with today's challenges in the aeronautical and automotive fields for example, where certain control problems are inherently multi-dimensional. With the progressive advent of autonomous cars, new concepts of light hybrid or electric vehicles are for example emerging, where both hydraulic and electric actuators are mounted on each wheel to improve the combined management of speed and steering. In this context, this paper introduces a non-trivial generalisation of the DWCA, which retains the various characteristics and advantages of the initial version, and can be used in the multi-dimensional case.

KEYWORDS
multi-dimensional control allocation; weighted pseudo-inverse with dynamic weight adaptation; actuator prioritisation

1. Introduction

A classical way to control a dynamical system is to design a controller that produces several virtual reference inputs, typically a number of forces and moments equal to the number of degrees of freedom to be controlled, each of them being then realised by a single actuator. But to improve the performance of the system and to make it safer, it can be relevant to use several actuators to control the same degree of freedom, or conversely to involve an actuator in the control of several degrees of freedom. When the number of actuators exceeds the number of degrees of freedom, the system is said

CONTACT C. Roos. Email: clement.roos@onera.fr

to be overactuated, and a control allocation algorithm is used to convert the virtual inputs into individual actuator commands in an optimal way. A good algorithm should mainly allow to produce any realisable virtual inputs while minimising control power, and to minimise the allocation error in case of control power deficiency. It should also cope with the actuator limitations, such as their sometimes slow dynamics or their position and rate saturations. Other criteria resulting from industrial constraints must finally be taken into account. A control allocator should be compatible with implementation constraints such as limited computational power, and in particular avoid time-consuming operations. It should also often be compatible with stringent certification constraints, which can prevent the use of non-deterministic techniques or solver-based approaches. And it should in many cases be able to establish a priority between some primary and secondary actuators, the latter being used sparingly (to avoid overheating, fatigue, maintenance cost...) only when the virtual inputs cannot be realised by the former.

The literature on control allocation is very rich, as shown by the surveys Johansen and Fossen (2013); Oppenheimer, Doman, and Bolender (2010), and more recently Sadien et al. (2020), which presents and compares more than 20 techniques on a realistic aeronautical benchmark. More generally, the applications are numerous, notably in the aerospace (Durham, Bordignon, & Beck, 2017; Sadien, 2020), automotive (Schwartz, Siebenrock, & Hohmann, 2019; Wang, Gaoa, Wang, Wang, & Wang, 2021; Yue, Fang, Zhang, & Shangguan, 2021) and marine (Fossen, Johansen, & Perez, 2008; Ji, Bui, Balachandran, & Kim, 2013; Witkowska & Śmierzchalski, 2018) fields. If most of the existing techniques try to fulfill the main objectives mentioned above, only some of them are able to deal with actuator limitations, and almost none of them take into account a wide range of industry-oriented requirements, as pointed out in Sadien (2020). This observation motivated the development of the Dynamic Weighting Control Allocator (DWCA) in Sadien et al. (2020), inspired from the well-known weighted pseudo-inverse and daisy chaining techniques (Oppenheimer et al., 2010). This algorithm intelligently manages the trade-off between minimising actuators use and attaining the maximum virtual control, while meeting all the aforementioned specifications, in particular the priority management between the various actuators. Promising results were obtained when applied to the on-ground aircraft runway centerline tracking problem. The occasional use of differential braking only when the classical actuators (nose-wheel steering system and rudder) reach their limits indeed significantly reduces the risk of runway excursions during landing with strong crosswinds and degraded runway conditions. Statistical results on a high-fidelity Airbus simulator revealed that the lateral deviation and the actuators use are significantly reduced in difficult situations compared to an Airbus reference control allocator.

Nevertheless, the DWCA is currently limited to the 1-dimensional case, which means that only one degree of freedom can be controlled, specifically the yaw axis for the above example. But it would be particularly interesting to be able to deal with the general multi-dimensional case. For the on-ground aircraft, this would allow a combined management of the longitudinal velocity and the yaw rate, the brakes being used for both actions. The need is also pressing in the automotive field, with the progressive advent of autonomous cars. New concepts of light hybrid or electric vehicles are indeed emerging, where each wheel is equipped with an electric actuator allowing traction and regenerative braking (*i.e.* capturing braking energy and storing it in the battery), as well as a conventional hydraulic braking system. The system is therefore highly overactuated, since no less than eight actuators are present in addition to the main engine and the steering system to control only two degrees of freedom, namely

speed and steering. The objective is to guarantee safety and maneuverability thanks to effective braking without wheel lock, and to increase the vehicle's autonomy thanks to regenerative braking. In this context, and in addition to respecting the industrial constraints mentioned above, the possibility of distributing the actuators into different groups with varying degrees of priority depending on the road context really makes sense. On the one hand, journeys in urban areas, where zero emission is a major objective, are characterised by frequent braking and accelerations. Regenerative braking is very effective in these situations, so electric actuators should be preferred to hydraulic brakes. On the other hand, motorway journeys are characterised by the frequent use of cruise control by the driver (braking and acceleration are quite rare), therefore electric regenerative braking is almost impossible. Moreover, when braking at high speed, it is recommended to use mainly hydraulic brakes to reduce the stopping distance as they are more efficient.

In this context, this paper introduces the Extended Dynamic Weighting Control Allocator (EDWCA), which allows to address the aforementioned challenges. This is indeed a non-trivial generalisation of the DWCA to the multi-dimensional case, which retains its various characteristics and advantages, such as actuator limits management, implementation ease, low computational cost, compatibility with certification constraints and actuator grouping. The 2-dimensional case is presented to make explanations easier and allow a graphical interpretation, but the extension to the n-dimensional case does not raise any technical difficulty. The paper is structured as follows. Section 2 first states the considered control allocation problem with all the associated requirements, and Section 3 recalls a number of useful geometrical considerations. The main contribution of the paper is then presented in Section 4, which introduces the proposed control allocation algorithm (EDWCA) and shows that it meets all specifications. Finally, the behavior of the EDWCA is assessed in Section 5 on a dedicated academic scenario.

2. Problem statement

Mathematically, an allocator solves an underdetermined system of equations, often subject to additional constraints. It is fed by virtual inputs $v(t) \in \mathbb{R}^k$ (typically a number of forces and moments that equals the number of degrees of freedom to be controlled), and it delivers control inputs $u(t) \in \mathbb{R}^m$ to be sent to the actuators, where $m > k$. Actuator models are assumed to be linear in u, which is almost always the case in the literature and not very restrictive for many practical applications. Thus given $v(t)$, the allocation problem reduces to the computation of $u(t)$ such that:

$$Bu(t) = v(t) \tag{1}$$

for all $t \geq 0$, where $B \in \mathbb{R}^{k \times m}$ is the control effectiveness matrix of rank k. Actuators having limited capabilities, some position limits are introduced:

$$u^p_{min}(t) \leq u(t) \leq u^p_{max}(t) \tag{2}$$

where inequalities apply component-wise. The problem of finding $u(t)$ such that equations (1)-(2) are satisfied has an infinite number of solutions when sufficient control power is available, and a secondary objective can be defined such as minimising control

power. On the contrary, no exact solution exists in case of control power deficiency, and a common strategy is to minimise the allocation error $Bu - v$ in some sense.

Remark 1. Additional constraints can be introduced to generate control inputs that are fully compatible with actuator capacities, such as rate limits $u^r_{min} \leq \dot{u}(t) \leq u^r_{max}$ or limited dynamics.

A wide set of methods exist to solve the aforementioned control allocation problem, see *e.g.* Fossen et al. (2008); Johansen and Fossen (2013); Oppenheimer et al. (2010). A rather exhaustive literature review is proposed in Sadien et al. (2020), which is not reproduced here for the sake of brevity. In that work, about twenty algorithms (including but not limited to those presented in Härkegård (2003); Oppenheimer et al. (2010); Zaccarian (2009)) are thoroughly compared on a realistic aeronautical benchmark, namely runway centerline tracking after landing during the deceleration phase. This benchmark is characterised by several requirements:

Req. 1 minimise control power whenever possible, allow the actuators to produce any realisable virtual inputs, and minimise the allocation error in case of control power deficiency (Enns (1998)), which are classical requirements,

Req. 2 be compatible with implementation constraints such as limited computational power, and in particular avoid time-consuming operations,

Req. 3 be compatible with stringent certification constraints, which notably prevents the use of non-deterministic techniques or solver-based approaches,

Req. 4 reach actuator saturations almost simultaneously to allow efficient recovery in case of failure, since reconfiguration may take too long when a failed actuator is at maximum deflection but not the others,

Req. 5 be able to group the actuators as "primary" or "secondary", the latter being used sparingly (to avoid overheating, fatigue, maintenance cost...) only when the virtual inputs cannot be realised by the former.

None of the control allocation techniques evaluated in Sadien et al. (2020) can handle all these requirements, which motivated the development of a new algorithm, the Dynamic Weighting Control Allocator (DWCA).

Beyond the considered benchmark, most requirements are recurrent in the aeronautical industry, as well as in other fields such as automotive for example, as emphasised in Section 1. Moreover, control problems are often multi-dimensional, with several virtual inputs v to be realised. If the DWCA can deal with all aforementioned requirements, it is unfortunately restricted to the 1-dimensional case, *i.e.* $k = 1$. In this context, **the main contribution of this paper is to propose a non-trivial generalisation of the DWCA to any value of** k. Note that all results are stated for the 2-dimensional case in the sequel, where two virtual forces or moments should be realised simultaneously, *i.e.* $k = 2$. As already highlighted in the introduction, this is not restrictive and just allows to make explanations and graphical interpretations easier.

Notations. The (i, j)-entry, the i^{th} row and the j^{th} column of a matrix M are denoted m_{ij}, $m_{i\times}$ and $m_{\times j}$ respectively. The i^{th} element of a vector v is denoted v_i, while v^j represents either the value of v at point j if j is a number or a letter, or a vector set if j is a symbol. The dependence on t is omitted in the sequel to improve readability.

3. Geometry of the 2-dimensional control allocation problem

This section is mostly derived from Durham (1993); Durham et al. (2017), and is intended to facilitate the understanding of the algorithm proposed in Section 4. The Attainable Moment Set (AMS) contains all values taken by the virtual inputs $v \in \mathbb{R}^2$ when the control inputs $u \in \mathbb{R}^m$ span the feasible control input set $\mathbb{U} = [u^p_{min}\ u^p_{max}]$. This means that problem (1)-(2) has a solution if and only if v belongs to the AMS. The vertices of $\mathbb{U}$ form a set $u^\dagger$ with 2^m elements, and the set of virtual inputs generated by $u^\dagger$ is denoted $v^\dagger$. The elements of $v^\dagger$ are two by two symmetrical with respect to the point $\overline{v} = B\,(u^p_{max} - u^p_{min})$, which is equal to zero in the common case where $u^p_{max} = -u^p_{min}$. Moreover, the AMS is the convex hull of $v^\dagger$ and has $2m$ vertices, which implies that the $2^m - 2m$ remaining elements of $v^\dagger$ lie strictly inside the AMS.

Let's now introduce the set of points that form the boundary of the AMS and the control inputs that generate them, denoted $v^\star$ and $u^\star$ respectively. According to the above, $v^\star$ is a polygon whose $2m$ edges are parallel two by two. Furthermore, along one edge of each pair, each control input stays at either its lower or upper limit (defined by u^p_{min} and u^p_{max}), except one that varies from its lower to its upper limit, let's say the j^{th} one. And along the other edge, each control input stays at its opposite limit, except the j^{th} one that varies in the same way as before. Importantly, both edges are parallel to the vector $b_{\times j}$ formed by the j^{th} column of B, which corresponds to the efficiency of the varying control input. There is a simple way to determine which combination of lower and upper limits should be considered on each of the two edges for all but the j^{th} control inputs. The control effectiveness matrix $B = \begin{bmatrix} b_{11} \dots b_{1m} \\ b_{21} \dots b_{2m} \end{bmatrix}$ is pre-multiplied by the vector t^j defined as follows:

$$t^j = \begin{cases} \begin{bmatrix} -\dfrac{b_{2j}}{b_{1j}} & 1 \end{bmatrix} & \text{if } \ b_{1j} \neq 0 \\ \\ \begin{bmatrix} 1 & 0 \end{bmatrix} & \text{otherwise} \end{cases} \tag{3}$$

so that the j^{th} element of $t^j B \in \mathbb{R}^m$ is equal to zero. On one edge, each control input except the j^{th} one is set to its upper (resp. lower) limit if the corresponding element of $t^j B$ is positive (resp. negative). And on the other edge, the opposite limits are set, as already mentioned above. This is an important property that will play a key role in Section 4.2. If there is more than one zero in $t^j B$, the problem is degenerated and can be solved by ganging the corresponding control inputs, since the latter all have the same efficiency on both axes.

Example: Let's consider $m = 3$ control inputs $u = [u_1\ u_2\ u_3]^T$, which contribute to the generation of a 2-dimensional virtual input $v = [v_1\ v_2]^T$ as follows:

$$v = Bu = \begin{bmatrix} 0.8147 & 0.1270 & 0.6324 \\ 0.9058 & 0.9134 & 0.0975 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{4}$$

with some position constraints u^p_{min} and u^p_{max}:

$$\begin{aligned} -1 \leq u_1 \leq 1 \\ -0.5 \leq u_2 \leq 0.5 \\ -2 \leq u_3 \leq 2 \end{aligned} \tag{5}$$

The corresponding AMS is shown in Figure 1. The elements of $v^\dagger$ are identified by black asterisks, while the boundary $v^\star$ is marked with blue lines. $v^\star$ is composed of $2m = 6$ edges obtained from 6 elements of $u^\dagger$, and the remaining $2^m - 2m = 2$ elements of $v^\dagger$ are strictly inside the AMS as expected. Let's now determine the two opposite edges on which only the first control input varies. According to the above, they are parallel to the efficiency vector $b_{\times 1}$ of the first control input, *i.e.* the first column of B, which is represented in red in Figure 1. These are therefore the edges v^1v^2 and v^3v^4. The matrix B is then pre-multiplied by $t^1 = \left[-\frac{0.9058}{0.8147} \quad 1\right]$, which leads to $t^1B = [0 \quad 0.7722 \quad -0.6056]$. This means that on one (resp. the other) edge, the first control input varies between its lower and upper limits -1 and 1, while the second and third ones are fixed to 0.5 (resp. -0.5) and -2 (resp. 2). Using this result and equation (1), it can finally be concluded that the four vertices $v^1 = [2.0160 \; 0.6441]^T$, $v^2 = [0.3866 \quad -1.1675]^T$, $v^3 = [-0.3866 \; 1.1675]^T$ and $v^4 = [-2.0160 \quad -0.6441]^T$ are generated by $u^1 = [1 \quad -0.5 \quad 2]^T$, $u^2 = [-1 \quad -0.5 \quad 2]^T$, $u^3 = [1 \quad 0.5 \quad -2]^T$ and $u^4 = [-1 \quad 0.5 \quad -2]^T$ respectively. □

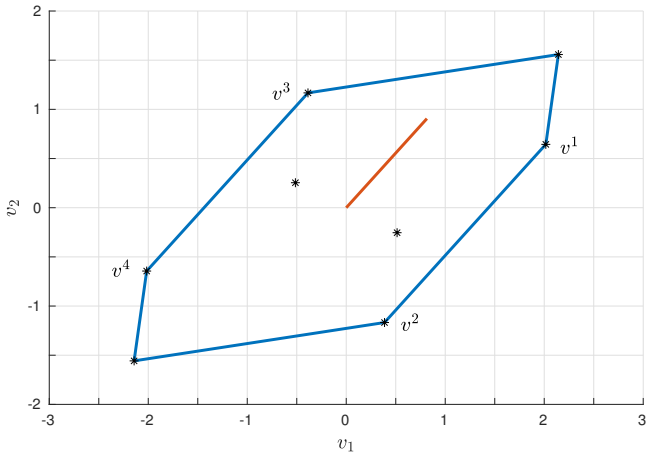

Figure 1. Attainable Moment Set for example (4)-(5)

4. Proposed control allocation technique

The Dynamic Weighting Control Allocator (DWCA) introduced in Sadien et al. (2020) is a pseudo-dynamical control allocation technique, which solves problem (1)-(2) in the 1-dimensional case while meeting all additional requirements listed in Section 2. The Extended Dynamic Weighting Control Allocator (EDWCA) described in this section can be seen as a non-trivial extension of the DWCA to the n-dimensional case. It has the same characteristics as the DWCA (actuator grouping, consideration of actuator dynamics and position limits...) and meets the same requirements (low computational power, compliance with certification constraints...). **But its implementation is**

significantly more complicated than for the DWCA for two main reasons detailed below.

The EDWCA is based on the weighted pseudo-inverse approach (Oppenheimer et al., 2010) and requires to solve:

$$\arg \min_{u \in \mathbb{R}^m} \frac{1}{2} u^T W^{-1} u \quad \text{subject to} \quad Bu = v \tag{6}$$

where $W \in \mathbb{R}^{m \times m} = \text{diag}(w_1, \dots, w_m)$ is a positive weighting matrix, $B = \begin{bmatrix} b_{11} \dots b_{1m} \\ b_{21} \dots b_{2m} \end{bmatrix} \in \mathbb{R}^{2 \times m}$ is the control effectiveness matrix and $v \in \mathbb{R}^2$ is the vector of virtual inputs to be realised. The general solution is:

$$u = WB^T(BWB^T)^{-1} v_{obj} \tag{7}$$

where v_{obj} is the virtual inputs objective, equal to v if v belongs to the AMS, and to a projection of v on the boundary $v^\star$ of the AMS otherwise, as explained in Section 4.1. In the literature, the weighting parameters w_i are often chosen constant and equal to the squared position limits of the actuators, which allows to minimise the required control power, but neither ensures that the actuators saturate simultaneously nor guarantees that any realisable virtual inputs v can be reached (Durham, 1993). In this context, the main originality of the proposed approach lies in the choice of the w_i, which are constantly adapted to meet the desired requirements. This choice was rather straightforward in the 1-dimensional case, but is much more tricky in the 2-dimensional case, since each actuator is characterised by two efficiencies b_{1i} and b_{2i}, which can be completely different. This first issue is addressed in Section 4.2.

The DWCA is also inspired by daisy chaining (Oppenheimer et al., 2010), which separates the control inputs into several groups of decreasing priority (two in the present case), in order to limit the use of the actuators from the last groups. Actuators from the primary group are used first up to their maximum capability, and only then are the actuators from the secondary group used. In the 1-dimensional case, the sharing of the virtual input v_{obj} between both groups is straightforward. All primary actuators are brought to their positions limits, which generates a virtual input v_p. The remaining virtual input $v_s = v_{obj} - v_p$ is then realised with the secondary actuators, which is always possible. The 2-dimensional case is again much more tricky, since attaining the boundary of the AMS no longer implies that all actuators are at their limits, as shown in Section 3. Moreover, the decomposition of v_{obj} into v_p and v_s should be done carefully. There are apparently an infinite number of choices, since v_p can be any point on the boundary $v^\star$ of the AMS of the primary actuators, but some of them (including the most obvious one in some cases, as shown on an example) can make the allocation problem unfeasible although a solution exists. This second issue is addressed in Section 4.3.

But first of all, the feasibility of the virtual inputs v is checked in Section 4.1, where the relation between v and v_{obj} is formally stated.

Remark 2. To make the equations simpler, it is assumed without loss of generality that $u^p_{min} < 0 < u^p_{max}$, which is almost always the case in practice. But the proposed algorithm can be easily extended to any values of u^p_{min} and u^p_{max}, and actuator

jamming may also be considered. To do so, the initial problem (6) is reformulated as:

$$\arg \min_{u \in \mathbb{R}^m} \frac{1}{2} (u - \bar{u})^T W^{-1} (u - \bar{u}) \quad \text{subject to} \quad Bu = v \tag{8}$$

where the i^{th} element of $\bar{u} \in \mathbb{R}^m$ represents the preferred position of the i^{th} actuator in $\left[u^p_{min_i}\ u^p_{max_i}\right]$, or the current position of the i^{th} actuator if the latter is jammed. Note that the preferred position is assumed to be 0 in this paper when $u^p_{min} < 0 < u^p_{max}$, but another choice could be made via $\bar{u}$. The general solution is:

$$u = \bar{u} + W\bar{B}^T(\bar{B}W\bar{B}^T)^{-1}\bar{v} \tag{9}$$

where the corrected control effectiveness matrix $\bar{B}$ and the effective virtual inputs $\bar{v}$ are constructed from B and v as described in Sadien (2020).

4.1. Determination of the virtual inputs objective v_{obj}

The feasibility of the virtual inputs v is first verified, and v_{obj} is computed from the position of v with respect to the AMS. To do that, the edge v^1v^2 of the AMS is searched for, which intersects the half-line starting at the origin O and passing through v. In practice, it is obtained by selecting the two vertices of the AMS whose angles bound that of v in the tightest way. The intersection point $v^B \in v^\star$ corresponds to the largest virtual inputs that can be realised in the direction of v. It is referred to as the boundary virtual inputs and can be expressed as follows:

$$v^B = \rho v = v^1 + \lambda v^{12} \tag{10}$$

where $v^{12} = v^2 - v^1$. The parameters ρ and λ are obtained by solving the linear equation:

$$\begin{bmatrix} v & -v^{12} \end{bmatrix} \begin{bmatrix} \rho \\ \lambda \end{bmatrix} = v^1 \tag{11}$$

where the matrix $\begin{bmatrix} v & -v^{12} \end{bmatrix}$ is non-singular since Ov and v^1v^2 are not parallel, and $\lambda \in [0, 1]$. A value $\rho \geq 1$ indicates that the virtual inputs v lie inside or on the boundary of the AMS, and are therefore feasible. The boundary control inputs $u_B \in u^\star$ which generate v_B, and therefore satisfy $Bu^B = v^B$, are finally computed as follows:

$$u^B = u^1 + \lambda u^{12} \tag{12}$$

where $Bu^1 = v^1$, $Bu^2 = v^2$ and $u^{12} = u^2 - u^1$.

Example (continued): Consider the vector of virtual inputs $v = [1.5 \ \ 0.25]^T$ represented by the blue asterisk in Figure 2. It can be seen that v is feasible, since it lies inside the AMS. The edge v^1v^2 is obtained by selecting the vertices of the AMS whose angles bound that of v in the tightest way. Here, the latter is $+9.5\,\text{deg}$, while those of $v^1 = [2.0160\ 0.6441]^T$ and $v^2 = [0.3866\ -1.1675]^T$ are $+17.7\,\text{deg}$ and $-71.7\,\text{deg}$ respectively. Solving equation (11) leads to $\rho = 1.1267 > 1$ and $\lambda = 0.2 \in [0\ 1]$. The boundary virtual and control inputs v^B and u^B are finally computed using equa-

tions (10) and (12):

$$v^B = 1.1267 \begin{bmatrix} 1.5 \\ 0.25 \end{bmatrix} = \begin{bmatrix} 1.6900 \\ 0.2817 \end{bmatrix} \tag{13}$$

$$u^B = \begin{bmatrix} 1 \\ -0.5 \\ 2 \end{bmatrix} + 0.2 \begin{bmatrix} -2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.6 \\ -0.5 \\ 2 \end{bmatrix} \tag{14}$$

and v^B is represented by a red circle in Figure 2. □

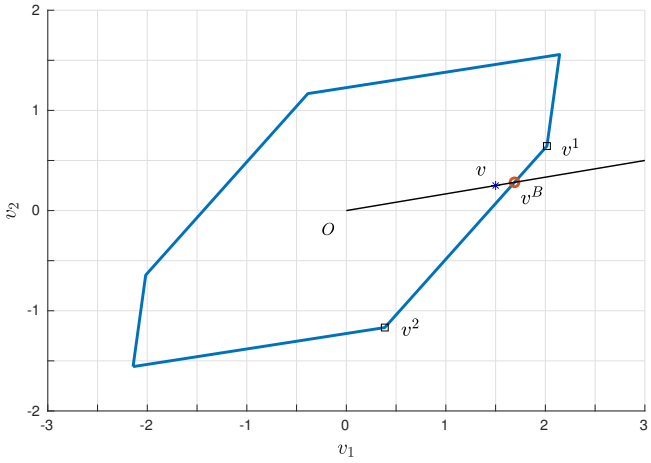

Figure 2. Determination of the boundary virtual inputs v^B

In the proposed approach, the m actuators associated with the m control inputs u are separated into two groups. The use of the secondary group is triggered when the virtual inputs v cannot be realised using the primary group only. Without loss of generality, it is assumed in the sequel that the first n and the last l actuators, where $n + l = m$, form the primary and secondary groups respectively. The AMS of the primary actuators, the secondary actuators and all actuators are denoted AMS_p, AMS_s and AMS_a in the sequel. Similarly, a subscript p, s or a is added to v^B, u^B and ρ to indicate that these variables refer to AMS_p, AMS_s and AMS_a respectively. The following strategy is implemented:

- If $\rho_p \geq 1$, the secondary actuators are not used, since v can be realised using the primary ones only, and $v_{obj} = v$.
- If $\rho_p < 1$ and $\rho_a \geq 1$, v can be realised using all actuators, and $v_{obj} = v$.
- If $\rho_a < 1$, equation (1) has no solution, and a way to make the problem feasible is to set $v_{obj} = \rho_a v = v_a^B$, where v_a^B denotes the boundary virtual inputs that can be realised using all actuators.

The first case is handled in Section 4.2, while the last two are dealt with in Section 4.3.

Remark 3. When v lies outside the AMS of all actuators and is therefore unfeasible ($\rho_a < 1$), the objective is to realise the maximum virtual inputs in the direction of v. But alternative strategies exist, such as prioritising one of the 2 axes. This can be done by projecting (whenever possible) either the first or the second component of v on the boundary of the AMS, *i.e.* by computing $\tilde{v}_1$ or $\tilde{v}_2$ such that $v_{obj} = [\tilde{v}_1 \; v_2]^T \in v^\star$

or $v_{obj} = [v_1 \ \tilde{v}_2]^T \in v^\star$.

4.2. Determination of the weighting matrix W

Let's forget at first the concept of actuator groups and assume that all actuators are treated in the same way. This operating mode may be triggered for instance when entering an emergency situation, where maximum maneuverability has priority over control power minimisation of the secondary actuators. Assume that $\rho \geq 1$, which means that $v_{obj} = v$ can be realised. The control inputs u that solve the allocation problem are computed with equation (7) using the following weighting parameters:

$$w_i = {u^p_{lim_i}}^2 + \frac{1}{\rho}\left(\frac{u_i^B}{L_i} - {u^p_{lim_i}}^2\right) \tag{15}$$

where the effective limits u^p_{lim} of the actuators are defined as follows:

$$u^p_{lim_i} = \begin{cases} u^p_{max_i} & \text{if sign}\left(u_i^B\right) \geq 0 \\ u^p_{min_i} & \text{otherwise} \end{cases} \tag{16}$$

and $L \in \mathbb{R}^m$ should be tuned carefully as explained below. The w_i are defined as the sum of two terms: the control power minimisation term ${u^p_{lim_i}}^2$ and the maximum virtual inputs reaching term $\frac{u_i^B}{L_i} - {u^p_{lim_i}}^2$. The trade-off between these two antagonist terms depends on the feasibility scalar ρ. When ρ is large, the virtual inputs are well inside the AMS. In this case, $w_i \approx {u^p_{lim_i}}^2$ and control power is minimised. But when ρ tends to 1, the virtual inputs tend to the boundary of the AMS as explained in Section 4.1, *i.e.* $v \to v^B$, and $w_i \to \frac{u_i^B}{L_i}$. Lemma 1 then proves that $w_i = \frac{u_i^B}{L_i}$ implies $u = u^B$ provided L is any linear combination of the first and second rows of B. As a consequence, all actuators except one reach saturation at the same time when v reaches v^B, as explained in Section 3.

Lemma 1. Let $(\alpha, \beta) \in \mathbb{R}^2$ such that all elements of $L = \alpha b_{1\times} + \beta b_{2\times} \in \mathbb{R}^m$ are nonzero. If $v = v^B$ and $W = \text{diag}\left(\frac{u_i^B}{L_i}\right)$, then the solution to the optimisation problem (6) is $u = u^B$.

Proof: Assume that $v = v^B$. v is feasible, since v^B lies in the AMS, so $v_{obj} = v^B = Bu^B$ according to Section 4.1 and equation (1). Then according to equation (7):

$$u = WB^T\left(BWB^T\right)^{-1} Bu^B \tag{17}$$

where $W = \text{diag}\left(\frac{u_i^B}{L_i}\right)$. Standard matrix computations lead to:

$$\left(BWB^T\right)^{-1} Bu^B = \frac{1}{\sigma}\begin{bmatrix} -\sum_{i=1}^{m}\sum_{j=1}^{m}\frac{b_{2i}u_i^B u_j^B \phi_{ij}}{L_i} \\ \sum_{i=1}^{m}\sum_{j=1}^{m}\frac{b_{1i}u_i^B u_j^B \phi_{ij}}{L_i} \end{bmatrix} \tag{18}$$

where σ is the determinant of BWB^T given by:

$$\sigma = \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{b_{1i}b_{2j}u_i^B u_j^B \phi_{ij}}{L_i L_j} \tag{19}$$

and $\phi_{ij} = b_{1i}b_{2j} - b_{2i}b_{1j}$. Each term of the sums in equation (18) is multiplied by L_j in both the numerator and denominator. L_j is then replaced with $\alpha b_{1j} + \beta b_{2j}$ in the numerator only:

$$\left(BWB^T\right)^{-1} Bu^B = \frac{1}{\sigma}\begin{bmatrix} -\alpha \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{b_{2i}b_{1j}u_i^B u_j^B \phi_{ij}}{L_i L_j} - \beta \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{b_{2i}b_{2j}u_i^B u_j^B \phi_{ij}}{L_i L_j} \\ \alpha \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{b_{1i}b_{1j}u_i^B u_j^B \phi_{ij}}{L_i L_j} + \beta \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{b_{1i}b_{2j}u_i^B u_j^B \phi_{ij}}{L_i L_j} \end{bmatrix} \tag{20}$$

$$= \frac{1}{\sigma}\begin{bmatrix} -\alpha M_{11} - \beta M_{12} \\ \alpha M_{21} + \beta M_{22} \end{bmatrix} \tag{21}$$

It can directly be seen that $M_{22} = \sigma$. Then by inverting indices i and j, which is possible because the two sums have the same number of terms, it is easy to check that $M_{11} = -\sigma$. Finally:

$$M_{12} = \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{b_{1i}b_{2i}b_{2j}^2 u_i^B u_j^B}{L_i L_j} - \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{b_{1j}b_{2j}b_{2i}^2 u_i^B u_j^B}{L_i L_j} \tag{22}$$

Inverting indices i and j as above in the right-hand component of equation (22) leads to $M_{12} = 0$. Similarly, $M_{21} = 0$. As a result, equation (21) simply becomes:

$$\left(BWB^T\right)^{-1} Bu^B = \begin{bmatrix}\alpha \\ \beta\end{bmatrix} \tag{23}$$

By replacing L_i again with $\alpha b_{1i} + \beta b_{2i}$ in W, a few more standard computations lead to:

$$u = WB^T \begin{bmatrix}\alpha \\ \beta\end{bmatrix} = u^B \tag{24}$$

which proves Lemma 1. This results holds for all $(\alpha, \beta) \in \mathbb{R}^2$ such that $L_i \neq 0$ for all $i \in [1\ m]$. □

Recall now that the control inputs u which generate the desired virtual inputs v are obtained as the solution of the L_2-norm minimisation problem (6), where W is a positive matrix. This means that all w_i must be positive. According to equation (15), w_i can take any value between ${u_{lim_i}^p}^2$ and $\frac{u_i^B}{L_i}$, since to $0 \leq \frac{1}{\rho} \leq 1$. The first term ${u_{lim_i}^p}^2$ being always positive, it is therefore sufficient to ensure that $\frac{u_i^B}{L_i} \geq 0$. In this context, a way to tune the parameters α and β such that $L_i = \alpha b_{1i} + \beta b_{2i}$ has the same sign as u_i^B for all $i \in [1\ m]$ is proposed in Lemma 2. It is assumed that all entries of B are nonzero for simplicity reasons, but this result can be adapted to the general case.

Lemma 2. Recall that v^B lies on one of the edges of the AMS, where each control input stays at either its lower or upper limit, except one, let's say the j^{th}. Let t^j be defined by equation (3) and:

$$\tilde{s} = \text{sign}\left(t^j b_{\times i} u_i^B\right) \quad \text{for all} \quad i \in [1\ m], i \neq j \tag{25}$$

$$s_i = \text{sign}\left(b_{2i} u_i^B\right) \quad \text{for all} \quad i \in [1\ m] \tag{26}$$

$$\mathcal{I} = \{i \in [1\ \ m] : s_i = -s_j\} \tag{27}$$

If $L = \alpha b_{1\times} + \beta b_{2\times}$, where:

$$\alpha = \tilde{s} t_1^j \tag{28}$$

$$\beta = \tilde{s} t_2^j + \frac{1}{2} s_j \min_{i \in \mathcal{I}} \left| \frac{t^j b_{\times i}}{b_{2i}} \right| \tag{29}$$

then $\frac{u_i^B}{L_i} \geq 0$ for all $i \in [1\ m]$.

Proof: As explained in Section 3, all the elements of $t^j B$ have either the same sign or the opposite sign as the corresponding elements of u^B, except the j^{th} one which is zero. In other words, $\tilde{s} = \text{sign}\left(t^j b_{\times i} u_i^B\right)$ has the same value for all $i \neq j$. So setting $\tilde{L} = \tilde{s} t^j B = \tilde{s} t_1^j b_{1\times} + \tilde{s} t_2^j b_{2\times} = \alpha b_{1\times} + \beta_1 b_{2\times}$ implies that $\tilde{L}_j = 0$ and $\frac{u_i^B}{\tilde{L}_i} \geq 0$ for all $i \neq j$. Let now $L = \tilde{L} + \beta_2 b_{2\times}$, where β_2 should be tuned to ensure that $\frac{u_j^B}{L_j} \geq 0$ while continuing to guarantee that $\frac{u_i^B}{L_i} \geq 0$ for all $i \neq j$. The condition $\frac{u_j^B}{L_j} \geq 0$ requires that $\text{sign}\left(\beta_2\right) = \text{sign}\left(b_{2j} u_j^B\right) = s_j$. Let then $\beta_2 = s_j \gamma$, where $\gamma > 0$. There exists a value of γ which makes the sign of L_i differ from that of $\tilde{L}_i$, *i.e.* $\frac{u_i^B}{L_i} < 0$, if and only if $s_i = -s_j$. And in that case, this happens if and only if $\gamma > -\frac{s_j \tilde{L}_i}{b_{2i}} > 0$. So choosing $\gamma = \frac{1}{2} \min_{i \in \mathcal{I}} \left| \frac{\tilde{L}_i}{b_{2i}} \right|$, where $\mathcal{I} = \{i \in [1\ \ m] : s_i = -s_j\}$, ensures that $\frac{u_i^B}{L_i} \geq 0$ for all $i \in [1\ \ m]$. This results in $L = \tilde{s} t_1^j b_{1\times} + \left(\tilde{s} t_2^j + \frac{1}{2} s_j \min_{i \in \mathcal{I}} \left| \frac{t^j b_{\times i}}{b_{2i}} \right|\right) b_{2\times} = \alpha b_{1\times} + \beta b_{2\times}$. □

Example (continued): Consider again the vector of virtual inputs $v = [1.5\ \ 0.25]^T$ represented by a blue asterisk in Figure 2. The corresponding boundary virtual inputs v^B lies on the edge $v^1 v^2$, whose vertices are generated by $u^1 = [1\ -0.5\ \ 2]^T$ and $u^2 = [-1\ \ -0.5\ \ 2]^T$, as computed in Section 3. So the only control input which varies on $v^1 v^2$ is the first one, *i.e.* $j = 1$, and $t^1 B = \left[-\frac{b_{21}}{b_{11}}\ \ 1\right] B = [0\ \ 0.7722\ \ -0.6056]$. Combining this with equation (14) leads to $\tilde{s} = \text{sign}\left(t^1 b_{\times 2} u_2^B\right) = \text{sign}\left(t^1 b_{\times 3} u_3^B\right) = -1$, and it can be checked that $\tilde{L} = [0\ \ -0.7722\ \ 0.6056]$ satisfies $\frac{u_i^B}{\tilde{L}_i} \geq 0$ for all $i \neq 1$. Let now $L = \tilde{L} + \beta_2 b_{2\times}$. As $u_1^B > 0$ in (14) and $b_{21} > 0$ in (4), β_2 needs to be positive to ensure that $\frac{u_1^B}{\tilde{L}_1} \geq 0$, *i.e.* $s_1 = 1$. The term $\beta_2 b_{2i}$ being positive, it cannot make $\frac{u_3^B}{\tilde{L}_3}$ negative. But if $\gamma > -\frac{\tilde{L}_2}{b_{22}} = 0.8454$, then $\frac{u_3^B}{\tilde{L}_3} < 0$, which is undesirable. The value $\gamma = \frac{1}{2} \frac{0.7722}{b_{22}} = 0.4227$ is finally chosen, which leads to:

$$L = [0\ \ -0.7722\ \ 0.6056] + 0.4227 \times [0.9058\ \ 0.9134\ \ 0.0975] = [0.3829\ \ -0.3861\ \ 0.6468]$$

It can be checked that L_i has the same sign as u_i^B, *i.e.* $\frac{u_i^B}{L_i} \geq 0$, for all $i \in [1\ \ 3]$. □

Let's now come back to the notion of actuator groups. If $\rho_p \geq 1$, then $v_{obj} = v$ is realised using the primary actuators only. In this case, the same strategy as above is applied, and the weighting parameters w_i are defined from equation (15) as follows:

$$\begin{aligned} w_{i\in[1\ \ n]} &= {u^p_{lim_i}}^2 + \frac{1}{\rho_p}\left(\frac{u^B_{pi}}{L_i} - {u^p_{lim_i}}^2\right) \\ w_{i\in[n+1\ \ m]} &= 0 \end{aligned} \tag{30}$$

where u^B_{pi} are determined as explained in Section 4.1 for $i \in [1\ \ n]$, and $u^B_{pi} = 0$ for $i \in [n+1\ \ m]$. The control inputs u that solve the allocation problem are then easily computed with equation (7), and satisfy $u_i = 0$ for all $i \in [n+1\ \ m]$.

4.3. Decomposition of v_{obj} into v_p and v_s

If $\rho_p < 1$, the secondary actuators are used in addition to the primary ones to generate the virtual inputs objective $v_{obj} = \min(1, \rho_a)\, v$. The latter must therefore be decomposed as the sum of v_p and v_s, which represent the virtual inputs objectives to be realised by the primary and secondary actuators respectively. Much care must be taken when doing this. Certain decompositions indeed yield an unfeasible objective v_s although v_{obj} lies in AMS_a, as illustrated below.

Example (continued): Let's now consider $m = 6$ control inputs $u = [u_1\ \ldots\ u_6]^T$, which contribute to the generation of a 2-dimensional virtual control input v as follows:

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0.8147 & 0.1270 & 0.6324 & 0.5200 & 0.1200 & 1.3540 \\ 0.9058 & 0.9134 & 0.0975 & 0.2310 & 0.5700 & 0.8900 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix} \tag{31}$$

with the following position limits u^p_{min} and u^p_{max}:

$$\begin{aligned} -1 \leq u_1 \leq 1 \\ -0.5 \leq u_2 \leq 0.5 \\ -2 \leq u_3 \leq 2 \\ -1 \leq u_4 \leq 1 \\ -1.25 \leq u_5 \leq 1.25 \\ -0.5 \leq u_6 \leq 0.5 \end{aligned} \tag{32}$$

The primary and secondary groups are composed of the first three and the last three actuators respectively. This is an extension of the illustrative example discussed previously in Sections 3 and 4, where only the primary actuators were considered. AMS_p and AMS_a are represented by blue and red/purple lines respectively in Figure 3, while AMS_s is represented by blue lines in Figure 4. Let's now consider the virtual inputs $v = [3.2683\ 2.5957]^T$ represented by the blue asterisk in Figure 3. Visually, it can be seen that v lies inside AMS_a and is therefore feasible, *i.e.* $v_{obj} = v$. A straightforward minimum norm decomposition $v_{obj} = \tilde{v}_p + \tilde{v}_s$ is shown by the yellow and red vectors re-

spectively. $\tilde{v}_p$ can be realised using the primary actuators, since it lies on the boundary of AMS_p. However, $\tilde{v}_s$ cannot be realised by the secondary actuators alone, as it lies outside AMS_s in Figure 4. This obvious decomposition of v_{obj} is thus not suitable. □

Fortunately, a systematic method can be implemented to decompose any v_{obj} into feasible v_p and v_s. To show that, let's first come back to the geometrical representation of the AMS. The edges of AMS_a are exactly composed of the edges of both AMS_p and AMS_s. This can for example be observed in Figures 3 and 4, where they are displayed in red and purple respectively. The region between AMS_p and AMS_a in Figure 3 is then divided into P and H zones. Each P zone is a parallelogram formed by one edge of AMS_p (blue) and the corresponding parallel edge of AMS_a (red), the other two edges (black) being referred to as the trailing edges. The H zones, represented in light blue in Figures 3 and 4, correspond to the areas left uncovered by the P zones. Each of them contains exactly one vertex of AMS_p. When all actuators are necessary ($\rho_p < 1$), the virtual inputs objective v_{obj} lies in either a P or H zone.

- If v_{obj} lies in a P zone Pi, v_p is chosen as the intersection point between the boundary of AMS_p on the one hand, and the line passing through v_{obj} and parallel to the trailing edges of Pi on the other hand.
- If v_{obj} lies in a H zone Hi, v_p is chosen as the only vertex of Hi that belongs to AMS_p.
- If v_{obj} lies between two adjacent zones, any of them can be considered and lead to the same decomposition.

Example (continued): Let's consider the same virtual inputs $v = [3.2683\ 2.5957]^T$ as above. Using the proposed decomposition method, $v_{obj} = v \in P1$ is written as the sum of v_p and v_s, which are represented by the pink and green vectors respectively in Figures 3 and 4. It is easily checked that v_p and v_s are both feasible, since they belong to AMS_p and AMS_s respectively. □

Now that v_p and v_s are defined, they are realised using the primary and secondary actuators respectively.

- The virtual inputs $v_p = v_p^B \in v_p^\star$ always lie on the boundary of AMS_p and are exactly realised with $u_p = u_p^B$, where u_{pi}^B are determined as explained in Section 4.1 for $i \in [1\ \ n]$, and $u_{pi}^B = 0$ for $i \in [n+1\ \ m]$. There is no need here to define weighting parameters w_i as in Section 4.2.
- The virtual inputs v_s are realised following the same strategy as in Section 4.2. The weighting parameters w_i are defined from equation (15) as follows:

$$\begin{aligned} w_{i\in[1\ n]} &= 0 \\ w_{i\in[n+1\ m]} &= {u_{lim_i}^p}^2 + \frac{1}{\rho_s}\left(\frac{u_{si}^B}{L_i} - {u_{lim_i}^p}^2\right) \end{aligned} \tag{33}$$

  where $u_{si}^B = 0$ for $i \in [1\ \ n]$ and u_{si}^B are determined as explained in Section 4.1 for $i \in [n+1\ \ m]$. The control inputs u_s that solve this allocation problem are then easily computed with equation (7), where $v_{obj} = v_s$, and satisfy $u_{si} = 0$ for all $i \in [1\ \ n]$.

The control inputs u that solve the whole allocation problem are finally obtained as

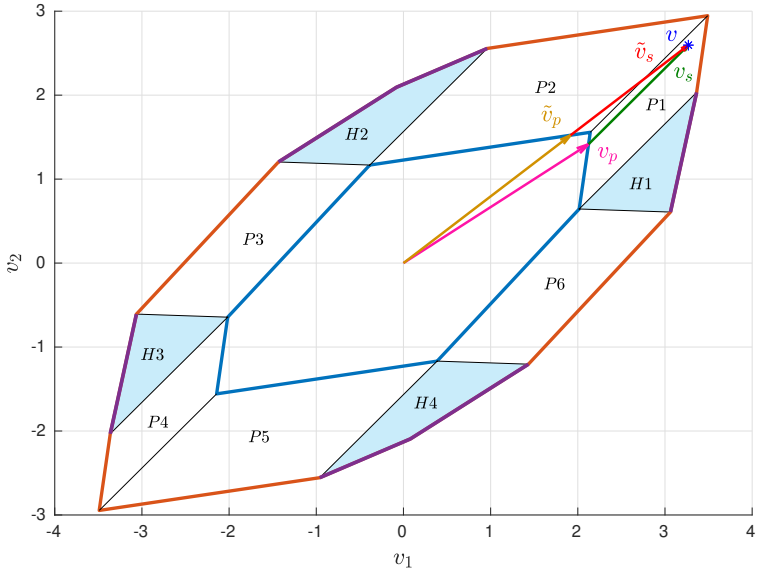

Figure 3. AMS$_p$ (blue lines) and AMS$_a$ (red/purple lines) with P and H zones

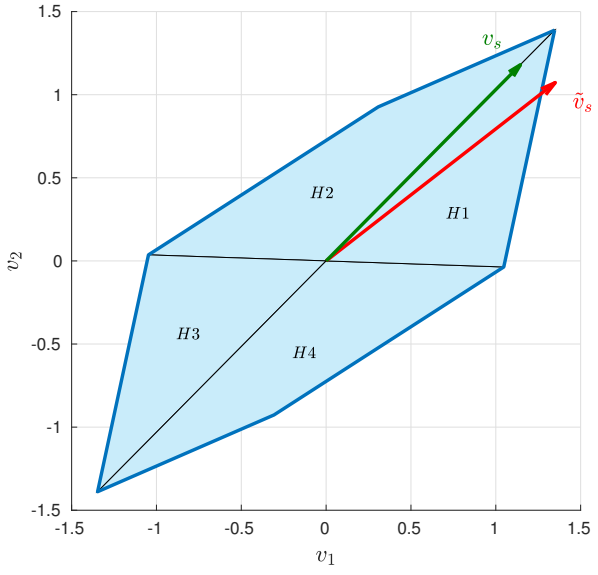

Figure 4. AMS$_s$ (blue lines) with H zones

follows:

$$u = u_p + u_s \tag{34}$$

Remark 4. The allocator being embedded in a digital system with sample time T, the inclusion of rate limits $u^r_{min} \leq \dot{u}(t) \leq u^r_{max}$ can be done by conversion into effective position limits as follows:

$$\begin{aligned} u^p_{min} &\leftarrow \max\left\{u^p_{min}\,,\, u_{-T} + T u^r_{min}\right\} \\ u^p_{max} &\leftarrow \min\left\{u^p_{max}\,,\, u_{-T} + T u^r_{max}\right\} \end{aligned} \tag{35}$$

where u_{-T} denotes the control inputs at the previous time step. Another solution to generate control inputs that are fully compatible with actuator capacities is to filter the weighting parameters w_i, which allows to consider both the dynamics and the rate limits of the actuators. First-order filters are introduced, whose time constants τ_i equal that of the actuators (or are chosen higher if rate limits are more stringent):

$$w_i \leftarrow \frac{1}{1+\tau_i s} w_i \tag{36}$$

4.4. Algorithm summary and compliance with the requirements

The Extended Dynamic Weighting Control Allocator (EDWCA) is briefly summarised below and the way it satisfies the five requirements stated in Section 2 is highlighted:

- If the virtual inputs v lie inside AMS_p (*i.e.* $\rho_p \geq 1$), they can be realised using solely the primary actuators and the virtual inputs objective is defined as $v_{obj} = v$. The weighting parameters (30) associated to these actuators are defined to manage automatically the trade-off between minimising control power and realising the largest possible virtual inputs, thus satisfying **Req. 1**. Moreover, as the virtual inputs v approach the boundary of AMS_p, all primary actuators except one reach saturation at the same time, as specified by **Req. 4**. On the other hand, the secondary actuators are not used in accordance with **Req. 5**, and the corresponding weighting parameters are set to zero in equation (30).
- If the virtual inputs v lie outside AMS_p (*i.e.* $\rho_p < 1$), all actuators are required and the virtual inputs objective is defined as $v_{obj} = \min(1, \rho_a)\, v$. A systematic approach is proposed to decompose v_{obj} into feasible v_p and v_s. v_p always lies on the boundary of AMS_p, implying that all primary actuators except one are at their position limits. On the other hand, the secondary actuators are treated as the primary ones in the previous case to meet **Req. 1**. As a consequence, when v approaches the boundary of AMS_a, all actuators except one primary and one secondary reach saturation at the same time, thus verifying here also **Req. 4**.
- In all cases, the weighting parameters are filtered as in equation (36) to cater for actuator dynamics and rate limits, before the control inputs (7) are finally computed.

The EDWCA being deterministic and based on simple solver-free algebraic calculations, it is compatible with the implementation and certification constraints of **Req. 2** and **Req. 3**.

Remark 5. The objective of the proposed control allocation algorithm is to satisfy

as much as possible the five requirements stated in Section 2. It cannot be claimed that the computed control inputs u are optimal, as they do not minimise any cost function covering all these requirements. This would indeed require to solve a complicated optimization problem, which is incompatible with meeting **Req. 2** and **Req. 3**. Nevertheless, the proposed decomposition of v into v_p and v_s guarantees that all virtual inputs inside AMS_p (resp. AMS_a) can be realised using the primary (resp. all) actuators. And for this choice, the amplitude of the control inputs is minimised. This is the EDWCA answer to **Req. 1**, which gathers the requirements that should classically be satisfied by a control allocation algorithm. But this decomposition is not unique, and another might lead to lower amplitude control inputs. This is the price to pay for a very fast algorithm compatible with demanding real-time applications characterised by **Reqs. 2-5**. This trade-off between dealing with numerous requirements and keeping a very low computational cost is illustrated in Section 5.

5. Numerical results and analysis

Let's consider again the control allocation problem (31). A scenario is considered, where the virtual inputs v pass through various zones inside and outside the AMS. More precisely, the following spiral is used:

$$v(t) = \begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix} = 0.0023t \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} \cos(-\omega t + \phi_0) \\ 0.6\sin(-\omega t + \phi_0) \end{bmatrix} \tag{37}$$

where $\psi = 0.8824$, $\omega = 0.006$ and $\phi_0 = 3$ are constant parameters. The EDWCA described in Section 4 is applied at $t = 1, 2, \ldots, 2300$ to compute the control inputs $u = [u_1 \ \ldots \ u_6]^T$ such that Bu best approximates the reference virtual inputs (37), while respecting the position limits (32). As already mentioned, the primary and secondary groups are composed of the first three and the last three actuators respectively. And for simplicity, actuator dynamics are ignored, which amounts to setting $\tau_i = 0$ in equation (36). Figures 5 and 6 show the reference (solid black lines) and realised (solid blue/red/yellow lines) virtual inputs in the (v_1, v_2) plane and as a function of time respectively, while Figure 7 displays the corresponding control inputs. Three cases can be observed:

- For $t \leq 772s$ and $1077s \leq t \leq 1138s$, the reference virtual inputs v lie inside AMS_p. They are therefore realised exactly using solely the primary actuators, *i.e.* $v = B\,[u_1 \ u_2 \ u_3 \ 0 \ 0 \ 0]^T$, and the blue solid lines overlap the black solid lines in Figures 5 and 6. No actuator reaches its limit in Figure 7, except at $t = 772s$ and $t = 1138s$ where two out of three do simultaneously, since the reference virtual inputs reach the boundary of AMS_p at those precise moments.
- For $772s < t < 1077s$, $1138s < t \leq 1301s$ and $1546s \leq t \leq 1711s$, the reference virtual inputs v lie outside AMS_p but inside AMS_a. They are therefore realised exactly using all actuators, *i.e.* $v = B\,[u_1 \ u_2 \ u_3 \ u_4 \ u_5 \ u_6]^T$, and the red solid lines overlap the black solid lines in Figures 5 and 6. At least two primary actuators reach their limit in Figure 7. But no secondary actuator does, except at $t = 1301s$ and $t = 1711s$ where two out of three do simultaneously, since the reference virtual inputs reach the boundary of AMS_a at those precise moments.
- For $1301s < t < 1546s$ and $t > 1711s$, the reference virtual inputs v lie outside AMS_a. They cannot be realised exactly, even if all actuators are used. The yellow

solid lines no longer overlap the black solid lines in Figures 5 and 6, but they follow the boundary of AMS_a in Figure 5, thus realising at any time the maximum virtual inputs in the direction of the reference ones.

Let's now analyse the performance of the EDWCA in terms of computational time. The above scenario is executed using Matlab R2018b on a Windows 10 laptop from 2019 with an Intel Core i5-8400H CPU running at 2.5 GHz and 16 GB of RAM. As emphasised above, the EDWCA is run 2300 times and three main tasks are performed each time:

(1) determine the virtual inputs objective v_{obj} and the required actuators, depending on whether v belongs or not to AMS_p or AMS_a (see Section 4.1),
(2) decompose v_{obj} into v_p and v_s in case all actuators are necessary (see Section 4.3),
(3) compute the vector L as explained in Lemma 1, then the weighting matrix W using equations (30) and (33), and finally the control inputs u with equation (7).

The computational times associated with these three tasks are 0.0204s, 0.0035s and 0.0182s respectively, which represents 0.0421s in total. The average computational time per execution of the algorithm is thus equal to $0.0421/2300 = 1.832 \times 10^{-5}$s. Noting that an implementation in C code would be even faster, it can be concluded that the EDWCA is compatible with real-time applications.

Remark 6. The example considered in this paper is academic, but not so trivial. First, it implies no less than 6 actuators (3 primary and 3 secondary). Then, it has been developed so that the virtual inputs v to be realised pass through all possible areas: inside AMS_p where only the primary actuators are needed, inside AMS_a but outside AMS_p where all actuators are needed, and outside AMS_a where v cannot be realised even if all actuators are used. Finally, all possible transitions between these 3 areas are studied, in both directions.

6. Conclusion

This paper presents a non-trivial generalisation of the control allocation technique initially proposed in Sadien et al. (2020), which shares the same characteristics and benefits as the original version, but can be applied to multi-dimensional control problems. Based on the classical pseudo-inverse approach, it handles the usual tradeoff between virtual reference inputs realisation and control power minimisation. But it also offers three specific features. First, the actuators reach saturation almost simultaneously thanks to a dynamic weight adaptation mechanism, which allows an efficient recovery in case of failure. Then, several industrial requirements are taken into account, such as implementation ease, low computational cost and compatibility with certification constraints. And finally, the actuators can be classified as primary or secondary to prioritise the use of some over the others. But beyond this distribution of the actuators in several groups, it would also be relevant to allow the groups to evolve dynamically over time. Such an improvement is typically suggested by the automotive example mentioned in the introduction, where the primary actuators are not the same if the vehicle evolves in a urban area or on a highway. In this perspective, the smooth variation of the control inputs should be ensured during the transition, which will be the subject of future research.

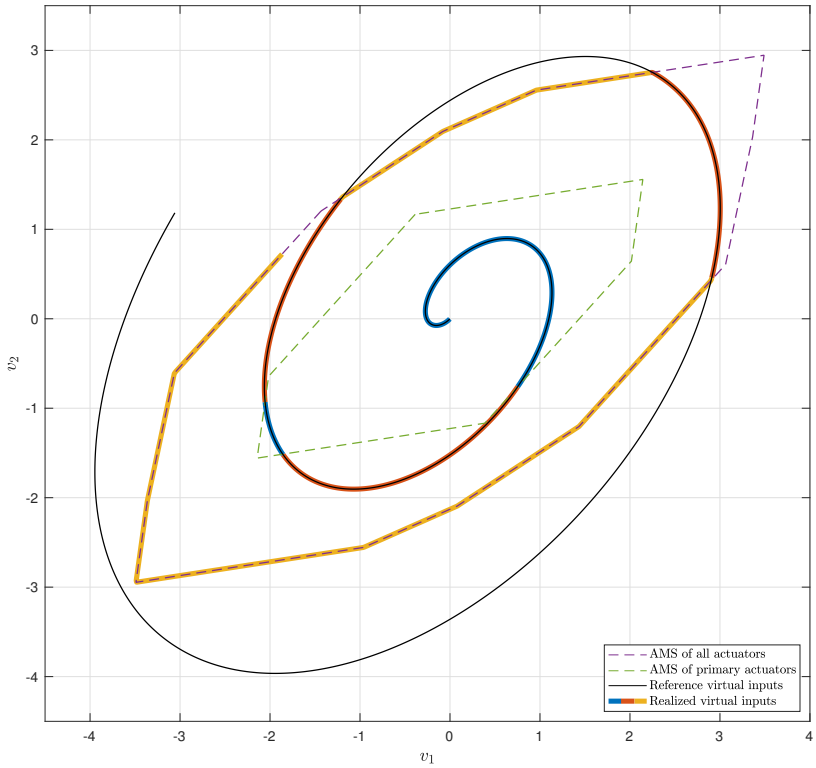

Figure 5. Reference virtual inputs v (solid black lines) and realised virtual inputs Bu (solid blue/red/yellow lines)

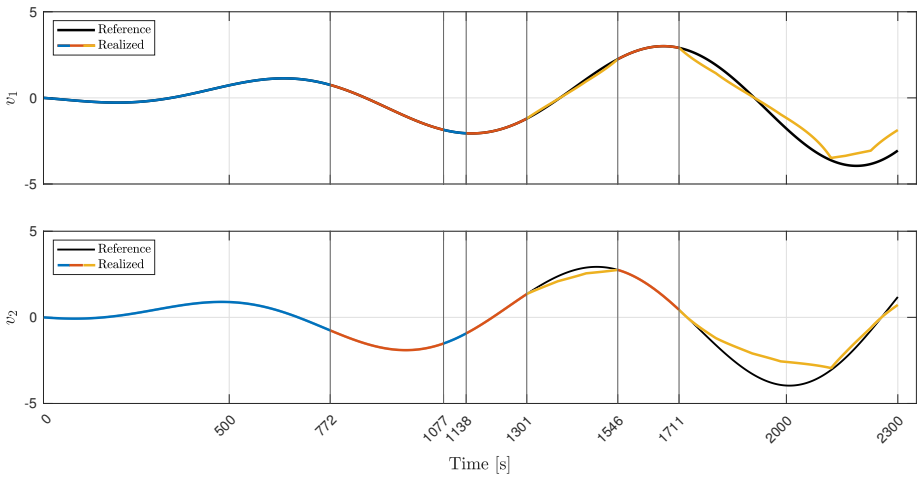

Figure 6. Reference virtual inputs v (solid black lines) and realised virtual inputs Bu (solid blue/red/yellow lines)

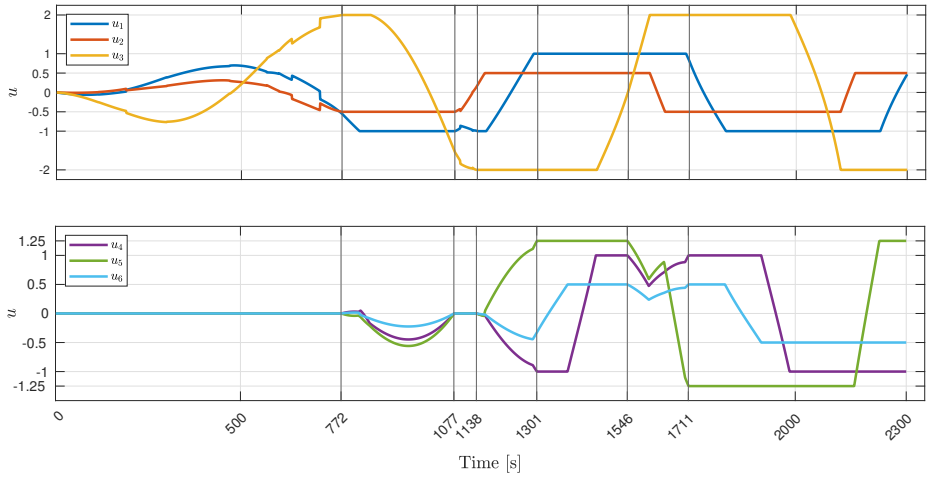

Figure 7. Primary (upper plot) and secondary (lower plot) control inputs u

Funding

This research is supported by Airbus Operations S.A.S., the National Association of Research and Technology (ANRT) and the French Ministry of Higher Education, Research and Innovation under the CIFRE contract 2016/1058.

Disclosure statement

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Durham, W. (1993). Constrained control allocation. *Journal of Guidance, Control and Dynamics*, *16*(4), 717 - 725.

Durham, W., Bordignon, K., & Beck, R. (2017). *Aircraft control allocation.* Wiley.

Enns, D. (1998). Control allocation approaches. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference* (p. 98 - 108). Boston, USA.

Fossen, T., Johansen, T., & Perez, T. (2008). Chapter 7: A survey of control allocation methods for underwater vehicles. In A. Inzartsev (Ed.), *Underwater vehicles* (p. 109 - 128). InTech.

Härkegård, O. (2003). *Backstepping and control allocation with applications to flight control* (Unpublished doctoral dissertation). Linköping University, Linköping, Sweden.

Ji, S.-W., Bui, V. P., Balachandran, B., & Kim, Y.-B. (2013). Robust control allocation design for marine vessel. *Ocean Engineering*, *63*, 105-111.

Johansen, T., & Fossen, T. (2013). Control allocation – A survey. *Automatica*, *49*(5), 1087 - 1103.

Oppenheimer, M., Doman, D., & Bolender, M. (2010). Chapter 8: Control allocation. In W. Levine (Ed.), *The control handbook, control system applications.* CRC Press.

Sadien, E. (2020). *Design of aircraft integrated ground control laws* (PhD thesis). Université de Haute-Alsace, Mulhouse, France.

Sadien, E., Roos, C., Birouche, A., Carton, M., Grimault, C., Romana, L.-E., & Basset, M. (2020). A simple and efficient control allocation scheme for on-ground aircraft runway centerline tracking. *Control Engineering Practice*, *95*. (doi:10.1016/j.conengprac.2019.104228)

Schwartz, M., Siebenrock, F., & Hohmann, S. (2019). Model predictive control allocation of an over-actuated electric vehicle with single wheel actuators. In *Proceedings of the 10th IFAC Symposium on Intelligent Autonomous Vehicles* (p. 162-169).

Wang, J., Gaoa, S., Wang, K., Wang, Y., & Wang, Q. (2021). Wheel torque distribution optimization of four-wheel independent-drive electric vehicle for energy efficient driving. *Control Engineering Practice*, *110*, 104779.

Witkowska, A., & Śmierzchalski, R. (2018). Adaptive dynamic control allocation for dynamic positioning of marine vessel based on backstepping method and sequential quadratic programming. *Ocean Engineering*, *163*, 570-582.

Yue, M., Fang, C., Zhang, H., & Shangguan, J. (2021). Adaptive authority allocation-based driver-automation shared control for autonomous vehicles. *Accident Analysis & Prevention*, *160*, 106301.

Zaccarian, L. (2009). Dynamic allocation for input redundant control systems. *Automatica*, *45*(6), 1431 - 1438.