ARTICLE

Optimal Resource Allocation for Stochastic Systems Performance Optimisation of Control Tasks undergoing Stochastic Execution Times

Daniele Fontanelli[a], Luca Greco[b] and Luigi Palopoli[c]

[a]DII - Dipartimento di Ingegneria Industriale, University of Trento, Trento, Italy; [b]Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, 91190, Gif-sur-Yvette, France; [c]DISI - Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, Trento, Italy

ARTICLE HISTORY
Compiled September 2, 2020

Abstract
The problem addressed in this paper is the optimal allocation of a CPU to a number of software control tasks. Each task is used to implement a feedback controller for a linear and time invariant system and is activated with a fixed period. On every periodic activation, the task executes a job, which collects the output of the system, and produces the control values after executing for a random computation time. If a job's duration exceeds a deadline, then the job is cancelled and the control values are not updated.

The systems to be controlled are affected by process noise. Therefore the performance of each control loop can be evaluated through the steady state covariance of the system's state, which depends on the probability with which the task implementing the controller drops its jobs. We show that by making a proper choice for the scheduling algorithm, this probability can be straightforwardly computed as a function of the scheduling parameters. This observation enables the construction of a very efficient procedure for finding the scheduling parameters that attain the optimal tradeoff between the performance of the different control loops.

KEYWORDS
Real–Time Control, Stochastic Systems, Hybrid Systems, Optimisation

1. Introduction

In embedded control systems a control loop is usually implemented as a software application, called *control task* (or simply *task*). Upon the periodic arrival of new sensor data, a control task executes *a job* in order to compute the control value and deliver it to the actuators. An important requirement motivated by cost and engineering reasons is that several control tasks share the same computing hardware. As an example, the BMW research division (Fürst, 2010) estimates that in a modern car more than 800 control functions share 70 embedded computers (ECU), which corresponds to an average allocation of more than 10 concurrent applications per ECU. New standards (such as AUTOSAR) have been defined to guarantee a safe coexistence of several

CONTACT D. Fontanelli. Email: daniele.fontanelli@unitn.it

concurrent applications at the software level, but the implications of this coexistence on control performance are not yet understood in depth.

The control performance obtained by the computer implementation of a controller is strongly related to the delays introduced in the computation. When a control task shares the CPU with other tasks, its jobs incur two types of delays: the first one caused by their own execution time and the second one by the time spent awaiting their turn to be "scheduled" and take control of the CPU (scheduling interference). Such delays are often time–varying and undermine the typical assumptions of digital control design: regular sampling and negligible (or at least constant) input-to-output latency for control computation. The ever increasing using of sophisticated sensing devices such as cameras, LIDARS, LASER scanners in such systems as autonomous robots and self-driving cars (Economist, 2016), adds complexity to the picture. The time needed to extract the information from such sensors can be conspicuous and it fluctuates significantly depending on the input data set. This justifies its modelling as a stochastic process with potentially long tails in the probability distributions (Fontanelli, Greco, & Palopoli, 2013).

A standard way to keep in check the delays suffered by a control task is by the combination of the time–triggered approach (Kopetz & Bauer, 2003) with the real–time scheduling theory (Liu & Layland, 1973). The former ensures constant delays by forcing the input-output operations of a task to take place at precise points in time (e.g., the sampling period expiration), even if the result is produced earlier. The latter offers conditions to guarantee that all jobs will deliver their results before the planned instants (called *deadlines*), given the scheduling parameters and the worst case execution time of the tasks. An important design problem that emanates from this approach is how to choose activation periods and scheduling parameters in order to maximise some "global" control performance metrics, which means how to optimally share the CPU time between different control tasks.

Related Work. This problem has been tackled in the literature (Palopoli, Pinello, Bicchi, & Sangiovanni-Vincentelli, 2005; Seto, Lehoczky, Sha, & Shin, 1996; Xu, Cervin, & Årzén, 2016; Xu, Cervin, & rzn, 2018) assuming a fixed priority or an Earliest Deadline First (EDF) scheduler (Liu & Layland, 1973). Unfortunately, these results are hardly applicable to the important class of application with widespread probability distributions of the computation time mentioned above, for which a system design based on the worst case execution time could leave the CPU under-utilised for large intervals of time. This problem calls for non conventional approaches in which stochastic fluctuations in the delays are accepted and harnessed (Frías, Palopoli, Abeni, & Fontanelli, 2018). Some authors have investigated on *how to make the control design robust* against an irregular timing behaviour of the implementation, focusing on such effects as packet dropout (Ling & Lemmon, 2002), jitter in computation (Marti, Fuertes, Fohler, & Ramamritham, 2001) and time varying delays (Kao & Rantzer, 2007). More recently, the onset of a new class of control algorithms, event-triggered (Wang & Lemmon, 2011) self-triggered (Aminifar, Tabuada, Eles, & Peng, 2016; Velasco, Martí, & Bini, 2008) dismisses the very idea of periodic sampling and advocates a different idea: the execution of the control action should take place only when necessary. Significant recent work has been made to close the gap between event-triggered control and a periodic implementation (Borgers et al., 2018), pushing the applicability boundary of this idea to the domain of non-linear systems. Other authors have championed innovative models for control applications based on the idea of anytime computing (Greco, Fontanelli, & Bicchi, 2011) or soft-real-time (Fontanelli & Palopoli, 2018). Usually control designers take such approaches as a leap into the unknown, while much can be said sticking to the traditional implementation of a digital control algorithm as a periodic task.

Paper Contribution. Following the steps of other authors (Palopoli et al., 2005; Seto et al., 1996; Xu et al., 2016; Xu et al., 2018), we seek the choice of scheduling parameters for a set of control task that maximises their global control performance. However, in our setting, the computation time of the tasks is not characterised by a single number (the worst case), but by a probability distribution. Based on the observation that occasional delays or data losses can be tolerated by many control systems (Cervin & Eker, 2003; Fontanelli, Greco, & Palopoli, 2013; Mohamed, Awan, Goswami, & Basten, 2019), we consider a model in which a job executing beyond its deadline is cancelled without closing the control loop (the same approach used in networked control to deal with packet dropouts by Nilsson and Bernhardsson (1996)). A key element of our strategy is the use of a scheduling algorithm known as the Constant Bandwidth Server (CBS) (Abeni & Buttazzo, 1998). Server-based scheduling for control tasks is proposed by other authors (Aminifar, Bini, Eles, & Peng, 2016) in a deterministic setting. Our specific motivation is the possibility to control the fraction of CPU time (bandwidth) allocated to each task, and hence the probability that a job of the task will miss its deadline and be cancelled.

In view of these choices, our problem bears a close resemblance to the allocation of bandwidth to channels with stochastic performance, well known in the communication community (Roberts, 2004). The performance metric is in our case the control performance, or Quality of Control (QoC), and the way it is evaluated is an additional contribution of the paper. Assuming that the task controls a plant affected by process noise, we express the QoC as the trace of the steady state covariance of the system's state. In our proposal, the control design can be made using the classical machinery of digital control, and the choice of the sampling period by the well known rules-of-the-thumb developed in decades of scientific literature and industrial practice. Then by assigning a bandwidth, we control the probability of deviating form the design assumption, and hence the QoC degradation. In particular, we show the functional dependence between this task level QoC metric and the bandwidth allocated to the task. By consolidating the QoC of the different control loops into a global QoC metric, we set up an optimisation problem where the values of bandwidth assigned to the tasks are decision variables. The most important contribution of the paper hinges on a few theoretical results on the properties of the optimal solution that exploit the specific structure of the problem and lead to a very efficient numeric algorithm. The paper subsumes and extends the results contained in a conference paper (Fontanelli, Palopoli, & Greco, 2013). In this preliminary work, the analysis was restricted to systems providing a QoC monotonic with respect to the bandwidth and no theoretical results were provided for the analytic computation of the QoC function. Both limitations have been addressed in this paper and the techniques presented here are now applicable to any kind of linear systems.

The paper is organised as follows. In Section 2 we offer a formal description of the problem. In Section 3, we describe how to formulate the optimisation problem, by identifying constraints and cost function, and by showing how the latter can be expressed in analytic closed form. In Section 4, we state our results on the solution of the optimisation problem and show how they can be translated into an efficient algorithm. In Section 5, we show a numeric validation of our technique selecting an insightful paradigmatic example in the wide set of simulation data that we collected. Finally, in Section 6, we offer our conclusions and announce future work directions.

2. Problem Presentation

Our goal is to control a number of independent linear systems $\mathcal{S}_i$, with $i = 1, \ldots, n$, sharing the same CPU. Each system $\mathcal{S}_i$ is assumed to be completely observable and controllable, and is described by the discrete-time dynamic equations:

$$\begin{aligned} x_i(k+1) &= A_i x_i(k) + F_i u_i(k) + w_i(k) \\ y_i(k) &= C_i x_i(k) \end{aligned} \tag{1}$$

where $x_i(k) \in \mathbb{R}^{n_{x_i}}$ represents the system state, $u_i(k) \in \mathbb{R}^{m_i}$ the control input, $w_i(k) \in \mathbb{R}^{n_{x_i}}$ a zero-mean white Gaussian noise term and $y_i \in \mathbb{R}^{p_i}$ the output variables. One step transition for this discrete time system refers to the evolution of the system step across one temporal unit, which is equal to the task period T_i.

The system is controlled by a feedback controller implemented through the control task τ_i. The temporal behaviour of the task follows the time–triggered model of computation Kopetz and Bauer (2003), described by the following rules: 1. at time kT_i, the input $u_i(k)$, which was computed by a previous execution of the task and then stored in a memory location, is retrieved, applied to the actuator and held constant up to the next sample (ZoH policy), 2. the plant output sample $y_i(k)$ is collected from $\mathcal{S}_i$, 3. a job $J_i(k)$ starts taking $y_i(k)$ as an input. Job $J_i(k)$ executes for $c_i(k)$ time units on the CPU to process the data and computes the new control value $u_i(k+1)$, which will be stored in memory and retrieved at time $(k+1)T_i$. The job is required to finish before a *deadline* sets equal to the next sampling time $(k+1)T_i$. If the job correctly finishes before the deadline, then its output is stored into a memory location and sent to the actuators at time $(k+1)T_i$. Otherwise, the job $J_i(k)$ is cancelled and the new one is started without updating the control value.

The adoption of these rules allows us to assume a fixed delay between sampling and actuation (assumed equal to the period T_i without loss of generality) whenever the task finishes its job before the deadline, with the control value held constant when it does not. This model is conveniently captured by an additional state variable $\zeta_i \in \mathbb{R}^{m_i}$ to store the control value, hence system (1) becomes

$$\begin{bmatrix} x_i(k+1) \\ \zeta_i(k+1) \end{bmatrix} = \begin{bmatrix} A_i & F_i \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_i(k) \\ \zeta_i(k) \end{bmatrix} + \begin{bmatrix} 0 \\ I_m \end{bmatrix} u_i(k) + \begin{bmatrix} w_i(k) \\ 0 \end{bmatrix}. \tag{2}$$

The system is controlled by the stabilising dynamic controller:

$$\begin{aligned} z_i(k+1) &= H_i z_i(k) + K_i y_i(k) \\ u_i(k) &= N_i z_i(k) + G_i y_i(k) \end{aligned} \tag{3}$$

where $z_i \in \mathbb{R}^{n_{z_i}}$. Define the augmented state $\tilde{x}_i = [x_i^T, \zeta_i^T, z_i^T]^T \in \mathbb{R}^{n_{\tilde{x}_i}}$, with $n_{\tilde{x}_i} = n_{x_i} + n_{z_i} + m_i$.

In the considered scenario, the derivation of the output $y_i(k)$ is a time-varying computation demanding activity (e.g., the sample could be derived from processing an image frame), therefore the computation time $\{c_i(k)\}_{k \in \mathbb{Z}_{\geq 0}}$ of job $J_i(k)$ can be modelled as a stochastic process.

A job $J_i(k)$ could occasionally terminate beyond its deadline and be cancelled. As a result, the system closed loop evolution switches between $\tilde{x}_i(k+1) = A_{c_i}\tilde{x}_i(k) + v(k)$, when $J_i(k)$ finishes before the deadline, and $\tilde{x}_i(k+1) = A_{o_i}\tilde{x}_i(k) + v(k)$, when it is cancelled, where $v(k) \triangleq \left[w(k)^T, 0\right]^T \in \mathbb{R}^{n_{\tilde{x}_i}}$ and

$$A_{c_i} = \begin{bmatrix} A_i & F_i & 0 \\ G_i C_i & 0 & N_i \\ K_i C_i & 0 & H_i \end{bmatrix}, \; A_{o_i} = \begin{bmatrix} A_i & F_i & 0 \\ 0 & I_{m_i} & 0 \\ 0 & 0 & I_{n_{z_i}} \end{bmatrix}. \tag{4}$$

The matrix A_{c_i} is associated with the nominal closed loop evolution and is Schur-stable[1], while A_{o_i} could be not.

Example 2.1. We start here a simple example that will be developed throughout the paper with the aim to elucidate some key aspects of the proposed method.

Let us consider the following (randomly generated) system in the form (1):

$$\begin{aligned} x_i(k+1) &= \begin{bmatrix} 1.0077 & -0.0008 \\ 0.0062 & 1.0154 \end{bmatrix} x_i(k) + \begin{bmatrix} 0.0023 \\ 0.0189 \end{bmatrix} u_i(k) + w_i(k) \\ y_i(k) &= \begin{bmatrix} 0.4957 & 0.2867 \\ 0.7671 & 0.7342 \end{bmatrix} x_i(k) \end{aligned} \tag{5}$$

having two unstable poles in 1.01 and 1.02. The LQG regulator in the form (3) is given by:

$$\begin{aligned} z_i(k+1) &= \begin{bmatrix} 0.04783 & -0.05361 & 0.002259 \\ -0.05404 & 0.06377 & 0.01888 \\ 9.3 & -3.6 & -0.04591 \end{bmatrix} z_i(k) + \begin{bmatrix} 4.612 & -1.729 \\ -4.761 & 3.155 \\ 0 & 0 \end{bmatrix} y_i(k) \\ u_i(k) &= \begin{bmatrix} 9.3 & -3.6 & -0.04591 \end{bmatrix} z_i(k) \end{aligned}$$

being $G_i = 0$. The corresponding closed and open loop matrices in (4) can be readily obtained.

Scheduling Mechanism In order to allocate the CPU to the different tasks, we adopt the Constant Bandwidth Server (CBS) scheduler Abeni and Buttazzo (1998). The CBS allows us to assign a *bandwidth* B_i to each task sharing the CPU. A task receiving bandwidth B_i can be thought of as executing on a processor whose speed is a fraction B_i of the speed of the actual processor. The total allocated bandwidth cannot exceed 100%:

$$\textstyle\sum_{i=1}^{n} B_i \leq 1. \tag{6}$$

A fundamental property of the CBS is the *temporal isolation*, meaning that the ability for a task to respect a temporal constraint only depends on its execution requirement and on its allocated bandwidth and is independent of the behaviour of the other tasks in the system. Let $\mathcal{E}_i(k)$ denote the event "job $J_i(k)$ finishes before the deadline". As a consequence of the temporal isolation property, the event $\mathcal{E}_i(k)$ is independent of any event $\mathcal{E}_j(k)$ with $j \neq i$. Furthermore, in view of the cancellation policy described above, there is no accumulation of delays between jobs. Therefore, for any $\overline{k}$, the event $\mathcal{E}_i(\overline{k})$ is independent of any event $\mathcal{E}_i(k)$ for $k \neq \overline{k}$ and the probability of meeting the

[1]We recall that a discrete–time linear system is said Schur-stable if the absolute value of all eigenvalues of its dynamical matrix is strictly less than 1.

deadline is given by (see Fontanelli, Palopoli, and Greco (2013)):

$$\mu_i \triangleq \Pr\left\{\mathcal{E}_i(k)\right\} = \Pr\left\{\frac{c_i(k)}{T_i} \leq B_i\right\}. \tag{7}$$

Problem Formulation. Let $P_i(k) \triangleq \mathrm{E}\left\{\tilde{x}_i(k)\tilde{x}_i(k)^T\right\}$ be the variance of the augmented state. When $\tilde{P}_i \triangleq \lim_{k\to\infty} P_i(k) < +\infty$ (i.e., the variance converges), we will say that the closed loop system is *mean square stable* Costa, Fragoso, and Marques (2006). We will consider $\mathrm{Tr}\left\{\tilde{P}_i\right\}$ (i.e., the sum of the mean squared values of the state variables) as a QoC metric for control task τ_i. This is a function of the probability μ_i and hence of the bandwidth B_i, which we will denote by $\phi_i(B_i) \triangleq \mathrm{Tr}\left\{\tilde{P}_i\right\}$.

The QoC functions for each system can be consolidated into a global cost function $\Phi(B_1, B_2, \ldots, B_n)$. We do this by computing the infinity norm of the different QoC functions: $\Phi(B_1, \ldots, B_n) \triangleq \max_i |\phi_i(B_i)|$, where the absolute value can be omitted because the ϕ_i functions are non negative by definition. The infinity norm is a sensible choice if the designer seeks a balanced performance between the different tasks (avoiding the case of one control loop performing exceedingly well at the expenses of the other ones). If the designer prefers an unbalanced performance, where some higher importance tasks receive larger bandwidths, the previous global cost function can be adapted accordingly. A suitable rescaling of the individual ϕ_i functions by means of some positive coefficients q_i can easily achieve this goal. In this case, the ensuing global cost function will read $\Phi(B_1, \ldots, B_n) \triangleq \max_i |q_i\phi_i(B_i)|$.

By introducing $\Phi(B_1, \ldots, B_n)$, the optimisation problem can be formally stated as follows[2]:

$$\min_{B\in\mathcal{B}} \Phi(B) = \min_{B\in\mathcal{B}} \max_{i\in\{1,\ldots,n\}} \phi_i(B_i) \tag{8}$$

where $B \triangleq (B_1, \ldots, B_n)$. The set $\mathcal{B}$ is the set of feasible solutions and is given by the polytope:

$$\mathcal{B} \triangleq \left\{(B_1, \ldots, B_n) \in \mathbb{R}^n \middle|\, B_i^{(m)} \leq B_i \leq B_i^{(M)}, \textstyle\sum_{i=1}^n B_i \leq 1\right\}. \tag{9}$$

where the upper bound $B_i^{(M)}$ is to ensure that the task does not receive more bandwidth than it needs to achieve probability 1 of meeting the deadline and the lower bound $B_i^{(m)}$ identifies a minimal critical bandwidth to ensure mean square stability of the feedback loop, and $\sum_{i=1}^n B_i \leq 1$ comes from Condition (6).

3. Cost function and constraints for the optimisation problem

In this section we show how to identify the constraints for Problem (8) and discuss an analytical expression of ϕ_i as function of the probability μ_i of meeting the deadline, and then (via Equation (7)) of the bandwidth B_i. For notational simplicity, we will drop the i subscript whenever the discussion refers to a specific task.

In the following, we assume that the noise $w(\cdot)$ is an i.i.d. process with zero mean and variance $\mathrm{E}\left\{w(k)w(k)^T\right\} = W \in \mathbb{R}^{n_x\times n_x}$. Also the computation time $\{c(k)\}_{k\in\mathbb{Z}_{\geq 0}}$

[2]For sake of exposition, we present here the balanced performance case only.

of the tasks is an i.i.d. random process, which is mutually independent of $w(\cdot)$. Both processes $w(\cdot)$ and $c(\cdot)$ are assumed independent of the state.

The dynamic evolution of the variance $P \in \mathbb{R}^{n_{\tilde{x}} \times n_{\tilde{x}}}$ of the state $\tilde{x}$ is given by

$$\begin{aligned} P(k+1) = &\mathrm{E}\left\{\tilde{x}(k+1)\tilde{x}(k+1)^T\right\} = \\ &= \mu \mathrm{E}\left\{\left(A_c\tilde{x}(k)+v(k)\right)\left(A_c\tilde{x}(k)+v(k)\right)^T\right\} + \\ &\quad + (1-\mu)\mathrm{E}\left\{\left(A_o\tilde{x}(k)+v(k)\right)\left(A_o\tilde{x}(k)+v(k)\right)^T\right\}, \end{aligned}$$

where the probability of meeting the deadline μ stochastically rules the switching between the different (closed and open loop) dynamics. Taking into account the mutual independence of the stochastic processes and the fact that $w(\cdot)$ has zero mean and constant variance W, the equation above can be written as

$$P(k+1) = \mu A_c P(k) A_c^T + (1-\mu)\, A_o P(k) A_o^T + H, \tag{10}$$

where $H \triangleq \left[\begin{smallmatrix} W & 0 \\ 0 & 0 \end{smallmatrix}\right]$.

An algorithmic estimate of the critical probability. In order to identify the constraints of Problem (8) (and in particular the lower bound $B_i^{(m)}$), we need to identify the critical probability, defined as the infimum of μ for which the system is mean square stable. Using Kronecker product properties we can write the dynamics (10) as

$$\mathrm{vec}(P(k+1)) = \left(\mu A_c^{[2]} + (1-\mu) A_o^{[2]}\right) \mathrm{vec}(P(k)) + \mathrm{vec}(H), \tag{11}$$

where $M^{[2]} \triangleq M \otimes M$ and $\mathrm{vec}(\cdot)$ is the linear operator producing a vector by stacking the columns of a matrix. This is a discrete–time linear time–invariant system in the state $\mathrm{vec}(P(j)) \in \mathbb{R}^{n_{\tilde{x}}^2}$. Hence, it admits a steady state solution $\tilde{P}(\mu)$ w.r.t. the constant input $\mathrm{vec}(H) \in \mathbb{R}^{n_{\tilde{x}}^2}$ if

$$\max_i \left|\lambda_i\left(\mu A_c^{[2]} + (1-\mu) A_o^{[2]}\right)\right| < 1, \tag{12}$$

with $\lambda_i(M)$ denoting i-th eigenvalue of M.

In view of these considerations the critical probability can be defined as $\widetilde{\mu} \triangleq \inf_{\bar{\mu}\in[0,1]} \{\bar{\mu} \mid \forall \mu \geq \bar{\mu} \text{ s.t. Cond. (12) is satisfied}\}$. Note that a solution with $\widetilde{\mu} < 1$ always exists, since A_c is Schur and the eigenvalues are continuous functions of μ. In practice, instead of looking for the infimum in the continuum set $[0,1]$, we can fix an accuracy level t_μ and adopt Algorithm 1 which implements a dichotomic search in the μ space. The function `SchurStableSegment(A,B)` implements the necessary and sufficient algebraic conditions in Theorem 4.*iii* of Elsner and Szulc (2000) and provides a positive answer if all the matrices given by the convex combination of A and B are Schur-stable.

Measuring the QoC of each task. The following Lemma shows an analytic computation for the QoC function $\phi(\cdot)$.

Lemma 3.1. *The QoC function $\phi : [\widetilde{\mu}, 1] \to \mathbb{R}_{>0} \cup \{+\infty\}$, $\mu \mapsto \mathrm{Tr}\left\{\tilde{P}(\mu)\right\}$ can be*

Algorithm 1 Find Critical Probability

```
 1: function FindMu($A_c$,$A_o$)
 2:     $A_C = A_c^{[2]}$
 3:     $A_O = A_o^{[2]}$
 4:     $\widetilde{\mu} = 1$
 5:     $\mu_0 = 0$
 6:     while not($\widetilde{\mu} - \mu_0 < t_\mu \;\wedge\; \widetilde{\mu} < 1 - t_\mu$) do
 7:         $\mu = (\widetilde{\mu} - \mu_0)/2 + \mu_0$
 8:         $A_2 = (1 - \mu) * A_O + \mu * A_C$
 9:         if SchurStableSegment($A_C, A_2$) then
10:             $\widetilde{\mu} = \mu$
11:         else
12:             $\mu_0 = \mu$
13:         end if
14:     end while
15:     return $\widetilde{\mu}$
16: end function
```

expressed as the ratio of two polynomials in μ:

$$\mathrm{Tr}\left\{\tilde{P}(\mu)\right\} = \frac{n_{\tilde{x}}^2 \sum_{j=0}^{n_{\tilde{x}}^2-1} \alpha_j \mu^j}{\sum_{j=0}^{n_{\tilde{x}}^2-1} \gamma_j \mu^j}, \tag{13}$$

where the α_j and γ_j coefficients can be found as algebraic functions of the A_c and A_o matrices.

The proof is reported in the Appendix.

The QoC function $\phi(\cdot)$ is continuous with respect to μ because so are the eigenvalues of $\tilde{P}(\mu)$. Moreover, for any $\mu \in [\widetilde{\mu}, 1]$ Equation (12) is verified by definition and $\mathrm{Tr}\left\{\tilde{P}(\mu)\right\}$ is also finite.

Example 3.2. Coming back to the system of Example 2.1, let us consider a set of possible sampling periods $T_i = \{20, 24, 32, 40, 48, 56\}$ ms. The corresponding set of critical probabilities provided by the Algorithm 1 is then given by $\widetilde{\mu} = \{0.18, 0.2, 0.23, 0.26, 0.29, 0.31\}$.

Let us assume now that the system is affected by three mutually independent noise inputs (gathered in the vector w) of zero mean and standard deviation $\sigma = 0.01$. The associated covariance matrix is $W = \sigma^2 I_3$. The QoC function ϕ in (13) can be explicitly computed by means of the expressions detailed in the Appendix. For the system at hand, we have $n_{\tilde{x}} = 6$, hence 36 coefficients α_j and 36 coefficients γ_j. However, most of the coefficients turn out to be practically zero and the function ϕ becomes a ratio of polynomials of degree 8. The non zero coefficients are reported in Table 1. A graph of the ϕ for $\mu \in [0.18, 1]$ is instead shown in Figure 1. We stress that in deriving all the previous quantities we did not make any assumption on the distribution of the computation time, except for the mentioned independence between the noise sources.

Formalisation of the optimisation problem. Let $\Delta_{c_i} : \mathbb{R}_{\geq 0} \to [0, 1]$ denote the cumulative distribution function (cdf) of the computation time $\{c_i(k)\}_{k\in\mathbb{Z}_{\geq 0}}$ for the task τ_i. In view of (7), the probability μ_i to finish before the deadline is given by $\mu_i = \Pr\{c_i(k) \leq T_i B_i\} = \Delta_{c_i}(T_i B_i)$. By definition Δ_{c_i} is monotone non–decreasing. For the sake of simplicity, we will also let it to be strictly increasing and hence invertible. This leads us to an easy expression for the bandwidth B_i that attains the probability

$\alpha_{27} \to \alpha_{35}$	$\gamma_{27} \to \gamma_{35}$
0.0000005	0.0000029
-0.0000227	-0.0002319
0.0007560	0.0093476
-0.0148452	-0.2007313
0.1475369	2.0286992
-0.5625039	-7.4926220
-0.0150922	0.2150213
0.0092984	-0.0782990
0.0390932	0.3470355

Table 1. Coefficients α_j and γ_j for the QoC function ϕ defined in (13). They have been all multiplied by 10^6.

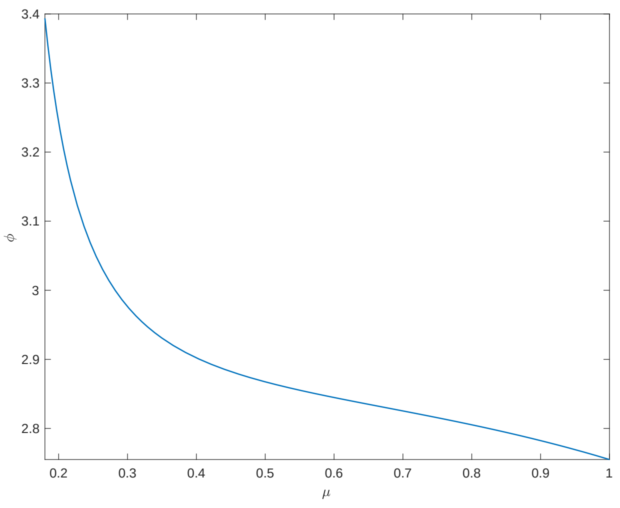

Figure 1. Graph of the function ϕ computed in the Example 3.2.

μ_i in Problem (8): $B_i = \frac{\Delta_{c_i}^{-1}(\mu_i)}{T_i}$, for every $\mu_i \in [0, 1]$. Consequently, the minimum bandwidth to achieve mean square stability is given by $B_i^{(m)} \triangleq \frac{\Delta_{c_i}^{-1}(\widetilde{\mu}_i)}{T_i}$, where $\widetilde{\mu}_i$ is the critical probability. Likewise, the bandwidth necessary for the i-th task to finish within its deadline with probability 1 is given $B_i^{(M)} \triangleq \frac{\Delta_{c_i}^{-1}(1)}{T_i}$.

In our framework the deadlines T_i are fixed, thus the probability μ_i is a strictly increasing function of the bandwidth B_i: $\mu_i = \Delta_{c_i}(B_i)$. The composition $\phi_i \circ \Delta_{c_i}(\cdot)$ is a function of the bandwidth B_i in the set $[B_i^{(m)}, B_i^{(M)}]$. With a slight abuse of notation, in what follows we will write $\phi_i(\cdot)$ directly as a function of B_i to refer such a composition. The expressions we have just shown for $B_i^{(m)}$, $B_i^{(M)}$ and $\phi_i(\cdot)$ allow us to give Problem (8) an explicit form. We will require below that the $\mathcal{B}$ polytope be not empty, and therefore the point $B^{(m)} = (B_1^{(m)}, \dots, B_n^{(m)})$ be feasible.

Example 3.3. Still with reference to the system in Example 2.1 and 3.2, let us assume now that the computation time c_i is described by a uniform distribution $\mathfrak{U}_{[\eta_i, b_c]}$ parametrised by the mean computation time η_i and by the best-case execution time (BCET) b_c ($c_i \geq b_c$). The distribution is defined in the range $[b_c, w_c]$, where

η_i [ms]	6	8	10	12	14	16	20	24	28
$B_i^{(m)}$ %	23.6	27.2	30.8	34.4	38	41.6	48.8	56	63.2
$B_i^{(M)}$ %	40	60	80	100	120	140	180	220	260

Table 2. Minimum and maximum bandwidth computed on the system of Example 2.1.

$w_c = 2\eta_i - b_c$ is the worst case execution time (WCET) and we fix here $b_c = 4$ ms. Fixing the sampling time to $T = 20$ ms we obtain the corresponding critical probability $\widetilde{\mu}_i = 0.18$. Once the mean value η_i is given, the distribution and the cumulative distribution Δ_{c_i} of the computation time c_i are fixed. Hence, the minimum and maximum bandwidth $B_i^{(m)}$, $B_i^{(M)}$ can be easily computed. The Table 2 reports the values of $B_i^{(m)}$ and $B_i^{(M)}$ for different mean values η_i. Notice that for $\eta_i \geq 14$ ms, the task WCET w_c is greater than the deadline T_i and indeed the maximum bandwidth $B_i^{(M)}$ is larger than 100%.

4. Solution of the Optimisation Problem

In this section we first state some fundamental results on the structure of the solutions of the optimisation problem introduced above. Then we show how such properties can be used to build an efficient optimisation algorithm.

Degenerate Problem. The optimisation Problem (8) can present some special cases that require a specific consideration.

Definition 4.1. The optimisation Problem (8) is said to be *degenerate* if there exist $i, j \in \{1, \dots, n\}$, $i \neq j$ such that $\phi_i(B_i) \geq \phi_j(B_j)$ for every $B_i \in [B_i^{(m)}, B_i^{(M)}]$ and $B_j \in [B_j^{(m)}, B_j^{(m)}]$. In such a case $\phi_i(\cdot)$ is said *to dominate* $\phi_j(\cdot)$.

If $\phi_j(\cdot)$ is a dominated function the optimal value of the cost function is not influenced by the bandwidth B_i: $\Phi(B) = \max_{h\in\{1,\dots,n\}} \phi_h(B_h) = \max_{h\in\{1,\dots,n\}\setminus\{j\}} \phi_h(B_h)$. Therefore, we can simply set $B_j = B_j^{(m)}$ to secure the largest possible feasibility set for the remaining variables, which will be an $(n-1)$–dimensional facet of the original polytope $\mathcal{B}$. In general, if there are $n' < n$ dominated functions with indices in the set $I' \subset \{1, \dots, n\}$, we can reduce the search for the optimal solution to the $(n - n')$–dimensional facet:

$$\mathcal{B}_{n'} \triangleq \mathcal{B} \setminus \left\{ B \in \mathbb{R}^n_{>0} \mid B_h = B^{(m)}, \forall h \in I' \right\}. \tag{14}$$

Optimal solution with rectified functions. The non monotonicity of the functions $\phi_i(\cdot)$ complicates the analysis of the solution set of the optimisation Problem (8), even in the non degenerate case. Therefore, we first study the solution set of an auxiliary optimisation problem, obtained by forcing all the $\phi_i(\cdot)$ to be non increasing functions, and then we exploit such knowledge to address the original one.

We define the "rectified" functions $\phi_i^{(rect)} : [B_i^{(m)}, B_i^{(M)}] \to \mathbb{R}_{>0} \cup \{+\infty\}$ for every $i \in \{1, \dots, n\}$ as follows (see Figure 2)

$$\phi_i^{(rect)}(B_i) \triangleq \min_{b \in [B_i^{(m)}, B_i]} \phi_i(b), \tag{15}$$

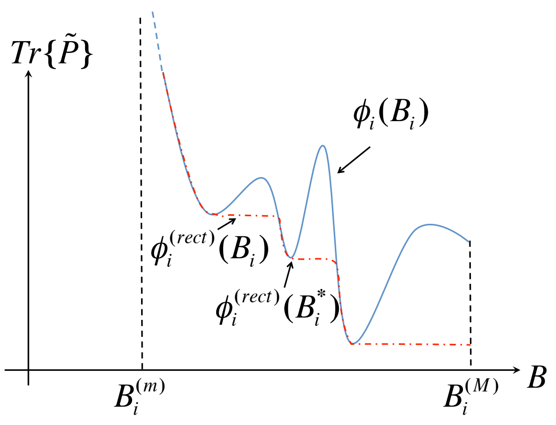

Figure 2. Graphical representation of the rectification of $\phi_i(\cdot)$. $\phi_i^{(rect)}(\cdot)$ holds on the values taken by the corresponding function $\phi_i(\cdot)$ in its local minima.

and we consider the associated optimisation problem

$$\min_{B\in\mathcal{B}} \Phi^{(rect)}(B) = \min_{B\in\mathcal{B}} \max_{i\in\{1,\dots,n\}} \phi_i^{(rect)}(B_i). \tag{16}$$

Remark 1. Differently from the functions $\phi_i(\cdot)$ (which are ratio of polynomials), their rectified counterparts $\phi_i^{(rect)}(\cdot)$, can be constant over intervals. Such functions hold on the values taken by the corresponding functions $\phi_i(\cdot)$ in their local minima. Indeed, if $B_i \in [B_i^{(m)}, B_i^{(M)}]$ is such that $\phi_i^{(rect)}(B_i) \neq \phi_i(B_i)$ then it can be easily verified that $B_i^\star = \min_{\alpha\in\{b\in[B_i^{(m)},B_i^{(M)}] | \phi_i^{(rect)}(b)=\phi_i^{(rect)}(B_i)\}} \alpha$ is a local minimum for $\phi_i(\cdot)$ and $\phi_i^{(rect)}(B_i) = \phi_i(B_i^\star)$ (see also Figure 2).

We now state a result on the structure of the solution for the optimisation Problem (16).

Theorem 4.2. *Assume that the optimisation Problem* (16) *is not degenerate, the functions $\phi_i : [B_i^{(m)}, B_i^{(M)}] \to \mathbb{R}_{>0} \cup \{+\infty\}$ are continuous and differentiable for every $i \in \{1,\dots,n\}$ and the functions $\phi_i^{(rect)} : [B_i^{(m)}, B_i^{(M)}] \to \mathbb{R}_{>0} \cup \{+\infty\}$ are defined as in* (15)*. Define the optimal solution set as $\mathcal{X}^{(rect)^*} \triangleq \arg\min_{B\in\mathcal{B}} \Phi^{(rect)}(B)$, and the following quantities:*

$$\bar{t}^{(M)} \triangleq \max_{i\in\{1,\dots,n\}} \phi_i^{(rect)}(B_i^{(M)}), \quad \underline{t}^{(m)} \triangleq \min_{i\in\{1,\dots,n\}} \phi_i^{(rect)}(B_i^{(m)}),$$
$$\bar{t}^{(m)} \triangleq \max_{i\in\{1,\dots,n\}} \phi_i^{(rect)}(B_i^{(m)}).$$

Let $\mathcal{A}_i(x) \triangleq \{b \in [B_i^{(m)}, B_i^{(M)}] \mid \phi_i^{(rect)}(b) = x\}$ and the vectors $\check{B}^{(m)} = (\check{B}_1^{(m)}, \dots, \check{B}_n^{(m)})$ and $\check{B}^{(M)} = (\check{B}_1^{(M)}, \dots, \check{B}_n^{(M)})$ be defined as:

$$\check{B}_i^{(m)} \triangleq \min_{\alpha\in\mathcal{A}_i(\underline{t}^{(m)})} \alpha, \ \check{B}_i^{(M)} \triangleq \min_{\alpha\in\mathcal{A}_i(\bar{t}^{(M)})} \alpha, \tag{17}$$

for every $i \in \{1,\dots,n\}$. The following mutually exclusive cases are given:

i) $\sum_{i=1}^{n} \check{B}_i^{(M)} \leq 1$, *then* $\check{B}^{(M)} \in \mathcal{X}^{(rect)^*}$ *and the optimal value is* $t^* = \bar{t}^{(M)}$*;*

ii) $\sum_{i=1}^{n} \check{B}_i^{(m)} \leq 1$ *and* $\sum_{i=1}^{n} \check{B}_i^{(M)} > 1$*, then there exists* $\tilde{B} = (\tilde{B}_1, \ldots, \tilde{B}_n)$*,* $\tilde{B} \in \mathcal{X}^{(rect)^*}$ *such that* $\phi_i^{(rect)}(\tilde{B}_i) = \phi_j^{(rect)}(\tilde{B}_j)$ *and* $\sum_{i=1}^{n} \tilde{B}_i = 1$*. The optimal value is* $t^* \in (\bar{t}^{(M)}, \underline{t}^{(m)}]$*;*

iii) $\sum_{i=1}^{n} \check{B}_i^{(m)} > 1$*, then there exists* $B^* = (B_1^*, \ldots, B_n^*)$ *such that for* $\bar{h} = \arg\min_{j \in \{1,\ldots,n\}} \phi_j^{(rect)}(B_j^{(m)})$*, the element* $B_{\bar{h}}^* = B_{\bar{h}}^{(m)}$ *and* $B^* \in \mathcal{X}^{(rect)^*}$*. The optimal values is* $t^* \in (\underline{t}^{(m)}, \bar{t}^{(m)}]$*.*

The intuition behind the rationale of the Theorem is as follows. The optimal solution of Problem (16) belongs to the region in which all the functions $\phi_i^{(rect)}(\cdot)$ assume the same value, if such a region exists. In that case, the optimal solution is either where one of the $\phi_i^{(rect)}(\cdot)$ has reached its minimum value (point *i*)) or where it is not possible to further decrease it (point *ii*)). If the region in which the functions $\phi_i^{(rect)}(\cdot)$ assume the same value does not exist, then there exists at least one function that will not play a role in the minimisation (point *iii*)). A detailed representation of the optimal solution $\tilde{B}$ of type *ii*) and related optimal value t^* is depicted in Figure 3. All the key quantities defined in the Theorem 4.2 are reported, along with the region of admissible bandwidths and the set where the $\phi_i^{(rect)}$ functions assume the same value.

Remark 2. The degenerate case can be addressed by restricting the optimisation domain to the facet $\mathcal{B}_{n'}$ in (14), which is an $(n-n')$–dimensional polytope. In the application of Theorem 4.2, the sum of the remaining bandwidths B_i has to be set to $\sum_{i \in \{1,\ldots,n\} \setminus I'} B_i = 1 - \sum_{i \in I'} B_i^{(m)}$ (rather than 1) to account for the bandwidth distributed to the tasks with degenerate functions $\phi_i(\cdot)$.

Remark 3. The case *iii*) in Theorem 4.2 is similar to the degenerate case. Indeed, once the $\bar{h}$-th component of B^* is fixed to $B_{\bar{h}}^* = B_{\bar{h}}^{(m)}$, the other components are found solving the optimal problem on the sub-polytope $\mathcal{B}' \triangleq \mathcal{B} \cap \{(B_1, \ldots, B_n) \in \mathbb{R}_{\geq 0}^n \mid B_i = B_i^{(m)}\}$ with the constraint $\sum_{i \in \{1,\ldots,n\} \setminus \{\bar{h}\}} B_i = 1 - B_{\bar{h}}^{(m)}$.

Optimal solutions of Problem (8)**.** If the $\phi_i(\cdot)$ functions are strictly decreasing, they coincide with their rectified counterpart in Theorem 4.2. In this case, it is possible to show that in case the situation of point *ii*) takes place, the optimal value is unique. In the discussion below, we show a Corollary tackling the generic case of non-monotone functions $\phi_i(\cdot)$. The characterisation of the optimal set of the Problem (8) is given in terms of the optimal set of the Problem (16). A representation of the optimal solution B^* of the point *i*) of the following Corollary is shown in Figure 3.

Corollary 4.3. *Assume the optimisation Problem* (8) *and* (16) *are not degenerate. Define the corresponding optimal solution sets as* $\mathcal{X}^* \triangleq \arg\min_{B \in \mathcal{B}} \Phi(B)$ *and* $\mathcal{X}^{(rect)^*} \triangleq \arg\min_{B \in \mathcal{B}} \Phi^{(rect)}(B)$*. Given* $\widetilde{B}^* = (\tilde{B}_1^*, \ldots, \tilde{B}_n^*) \in \mathcal{X}^{(rect)^*}$*, we have:*

i) the point $B^* = (B_1^*, \ldots, B_n^*)$ *such that* $B_i^* = \min_{\alpha \in \mathcal{A}_i(\phi_i^{(rect)}(\tilde{B}_i^*))} \alpha$ *for every* $i \in \{1, \ldots, n\}$*, is optimal for the Problem* (8)*, namely* $B^* \in \mathcal{X}^*$*. Moreover, for the optimal value we have* $\Phi(B^*) = \Phi^{(rect)}(\tilde{B}^*)$*;*

ii) if $\phi_i(\tilde{B}_i^*) = \phi_i^{(rect)}(\tilde{B}_i^*)$ *for every* $i \in \{1, \ldots, n\}$*, then* $\tilde{B}^* \in \mathcal{X}^*$*; otherwise the optimal point* $B^* \in \mathcal{X}^*$ *defined at point i) is such that there exists at least one*

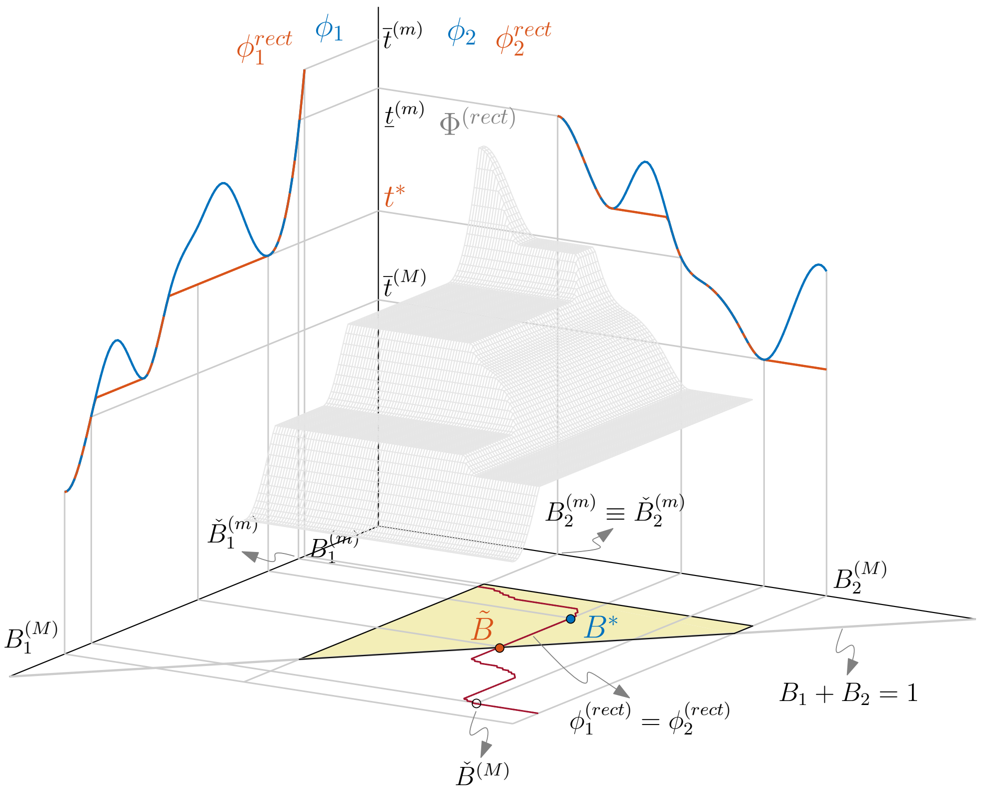

Figure 3. Example of optimisation in the case of two functions ϕ_i. The picture reports all the key quantities defined in Theorem 4.2 point *ii*) and Corollary 4.3 point *i*). In blue are the ϕ_i functions and in red their rectified version. The overall function $\Phi^{(rect)}$ is represented in faint grey. The yellow polytope represents the set of admissible bandwidths (notice the line $B_1 + B_2 = 1$) and the dark magenta curve represents the set of bandwidth for which the $\phi_i^{(rect)}$ functions assume the same value. The optimal value t^* and the optimal solution $\tilde{B}$ of the Problem (16) are reported in red, while the optimal solution B^* of the Problem (8) is reported in blue.

index $\bar{h} \in \{1, \dots, n\}$ *for which* $B^*_{\bar{h}} = B^{\dagger}_{\bar{h}}$ *with* $B^{\dagger}_{\bar{h}}$ *local minimum of* $\phi_{\bar{h}}(\cdot)$.

The Solution Algorithm. Theorem 4.2 and Corollary 4.3 are exploited to devise the Algorithm 2 for the computation of an optimal solution for the Problem (8). First the quantities $\check{B}^{(M)}$ and $\check{B}^{(m)}$ are computed (Lines 6-9), then condition at point *i*) of Theorem 4.2 is readily checked (Lines 10-11) and in case the optimal solution immediately returned. If the condition at point *iii*) holds, the sub-polytope defined in Remark 2 is in case determined (Lines 13-18). Then, for condition at point *ii*), two nested binary search algorithms are employed (Lines 20-32). The inner one, implemented in the function `FIND POINTS` (not reported due to its simplicity), is responsible for computing the generalised inverse of the functions $\phi_i^{(rect)}(\cdot)$: given a value $\check{t} \in [\bar{t}^{(M)}, \underline{t}^{(m)}]$, the algorithm provides the extreme points $\underline{B}_i^{\check{t}} \triangleq \min_{\alpha \in \mathcal{A}_i(\check{t})} \alpha$ and $\overline{B}_i^{\check{t}} \triangleq \max_{\alpha \in \mathcal{A}_i(\check{t})} \alpha$ (if $\phi_i^{(rect)}(\cdot)$ is locally invertible in $\check{t}$, then $\underline{B}_i^{\check{t}} = \overline{B}_i^{\check{t}}$). Notice that the algorithm is proved to converge to the optimal solution in this case in light of point *i*) of Corollary 4.3. Moreover, the problem solution strategy is particularly efficient for the function ϕ_i chosen in this paper, which links the bandwidth B_i to the QoC with an explicit analytic form. Other choices of ϕ_i are legitimated and perfectly compatible with the approach.

Algorithm 2 Solution Algorithm

```
function OptimalSolution($\phi^{(rect)}$, $B^{(m)}$, $B^{(M)}$, $n$)
    $\mathrm{Set}_I = [1, \ldots, n]$;  $\mathrm{Set}_H = \emptyset$;  $\mathrm{Max}_B = 1$
    $\bar{t}^{(M)} = \max_{i \in \mathrm{Set}_I} \phi_i^{(rect)}(B_i^{(M)})$
    $\underline{t}^{(m)} = \min_{i \in \mathrm{Set}_I} \phi_i^{(rect)}(B_i^{(m)})$
    $\bar{t}^{(m)} = \max_{i \in \mathrm{Set}_I} \phi_i^{(rect)}(B_i^{(m)})$
    for $i \in \mathrm{Set}_I$ do
        $[\check{B}_i^{(m)}, a] = \textsc{FindPoints}(\underline{t}^{(m)}, \phi_i^{(rect)}, B_i^{(m)}, B_i^{(M)})$
        $[\check{B}_i^{(M)}, a] = \textsc{FindPoints}(\bar{t}^{(M)}, \phi_i^{(rect)}, B_i^{(m)}, B_i^{(M)})$
    end for
    if $\sum_{i \in \mathrm{Set}_I} \check{B}_i^{(M)} \leq \mathrm{Max}_B$ then
        return $[\check{B}^{(M)}, \bar{t}^{(M)}]$
    else
        $t_a = \bar{t}^{(M)}$;  $t_b = \underline{t}^{(m)}$
        while $\sum_{i \in \mathrm{Set}_I} \check{B}_i^{(m)} > \mathrm{Max}_B$ do
            $\bar{h} = \arg\min_{j \in \mathrm{Set}_I} \phi_j^{(rect)}(B_j^{(m)})$
            $\mathrm{Set}_I = \mathrm{Set}_I \backslash \bar{h}$;  $\mathrm{Set}_H = \mathrm{Set}_H \cup \bar{h}$;  $\mathrm{Max}_B = \mathrm{Max}_B - B_{\bar{h}}^{(m)}$
            $\underline{t}^{(m)} = \min_{i \in \mathrm{Set}_I} \phi_i^{(rect)}(B_i^{(m)})$;  $t_b = \underline{t}^{(m)}$
        end while
        $\check{t} = \frac{t_a + t_b}{2}$
        while $\sum_{i \in \mathrm{Set}_I} \underline{B}_i^{\check{t}} > \mathrm{Max}_B \vee (\sum_{i \in \mathrm{Set}_I} \underline{B}_i^{\check{t}} < \mathrm{Max}_B \wedge \sum_{i \in \mathrm{Set}_I} \overline{B}_i^{\check{t}} \leq \mathrm{Max}_B)$ do
            for $i \in \mathrm{Set}_I$ do
                $[\underline{B}_i^{\check{t}}, \overline{B}_i^{\check{t}}] = \textsc{FindPoints}(\check{t}, \phi_i^{(rect)}, B_i^{(m)}, B_i^{(M)})$
            end for
            if $\sum_{i \in \mathrm{Set}_I} \underline{B}_i^{\check{t}} > \mathrm{Max}_B$ then
                $t_a = \check{t}$
            else
                if $\sum_{i \in \mathrm{Set}_I} \overline{B}_i^{\check{t}} < \mathrm{Max}_B$ then
                    $t_b = \check{t}$
                end if
            end if
            $\check{t} = \frac{t_a + t_b}{2}$
        end while
        return $[\underline{B}_{\mathrm{Set}_I}^{\check{t}} \bigcup B_{\mathrm{Set}_H}^{(m)}, \check{t}]$
    end if
end function
```

As an example, if no explicit form is available, ϕ_i could be numerically computed “by points” and interpolated.

5. Numeric Evaluation

In order to show a numeric validation of the bandwidth optimal synthesis described in Section 4, we consider randomly generated, open-loop unstable, reachable and observable linear discrete time systems subject to a linear combination of discrete time noises. The number of independent noise sources equals the number of states and their stochastic processes are normally distributed with zero mean and standard deviations equal to $\sigma = 0.01$. Among the hundreds of test cases synthesised, we report, as an example, three different systems, for which number of states n_{x_i}, inputs m_i, outputs p_i and the open loop unstable poles are reported in Table 3. The controllers are designed using the systematic LQG optimal control synthesis. They are executed in the tasks τ_1, τ_2 and τ_3, respectively, with execution period T_i. Three

Discrete-time linear systems		Controllers			
(n_{x_i}, m_i, p_i)	Unst. poles	Task	T_i	Distr.	$\tilde{\mu}$
(2, 1, 2)	1.01, 1.02	τ_1	20	$\mathfrak{U}_{[\eta^\star,4]}$	0.18
(3, 1, 3)	1.03, 1.01, 1.01	τ_2	56	$\mathfrak{B}_{[6,4]}$	0.33
(4, 1, 3)	1.08	τ_3	56	$\mathfrak{E}_{[6,4]}$	0.33

Table 3. Systems adopted in the simulations. Mean and best case computation times as well as periods are expressed in milliseconds. The system executed by task τ_1 is the one described in Example 2.1.

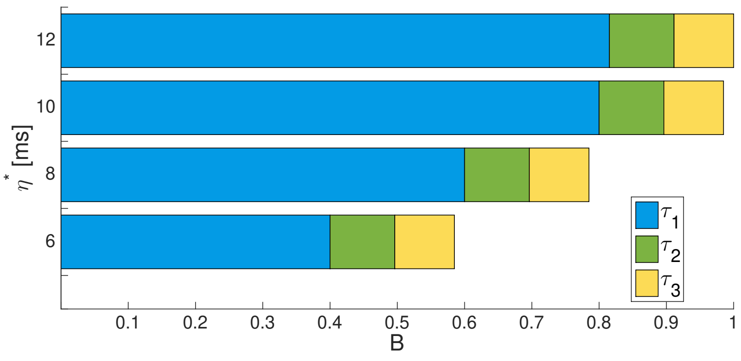

Figure 4. Bandwidth optimal allocation minimising $\mathrm{Tr}\left\{\tilde{P}(\mu)\right\}$ with respect to the systems specified in Table 3 with $\eta^\star \in \{6, 8, 10, 12\}$.

different probability density functions (pdf) have been chosen for the computation times of the control tasks, which are parametrised by the mean value η_i and by the best-case execution time b_c ($c_i \geq b_c$): $\mathfrak{U}_{[\eta_i, b_c]}$ is a uniform distribution defined as in Example 3.3; $\mathfrak{E}_{[\eta_i, b_c]} = \frac{1}{\eta_i} \mathrm{e}^{-\frac{c_i - b_c}{\eta_i}}$ is an exponential distribution, where the WCET is $w_c = +\infty$; $\mathfrak{B}_{[\eta_i, b_c]} = \frac{1}{N} \left(\frac{c_i - b_c}{w_c - b_c}\right)^{\alpha - 1} \left(1 - \frac{c_i - b_c}{w_c - b_c}\right)^{\beta - 1}$, with normalisation factor $N = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1} dx$, is a beta distribution, having WCET $w_c = 60$ ms and parameters $\alpha = 2$ and $\beta = \alpha \frac{w_c - \eta_i}{\eta_i - b_c}$. For all the distributions, the $b_c = 4$ ms. The task set is not hard real-time schedulable since the modelled WCETs are so high that no guarantee to finish the job within the activation period T_i can be given. Table 3 reports the critical probabilities $\tilde{\mu}$ computed with Algorithm 1.

To highlight the relevant features of the solution algorithm presented in Section 4, the possible mean values $\eta^\star$ of the execution times of task τ_1 are chosen in the set $\mathcal{N} = \{6, 8, 10, 12\}$ ms (see also Example 3.3), while η_i is fixed to 6 ms for τ_2 and τ_3. Figure 4 shows the optimal bandwidths, minimising the $\mathrm{Tr}\left\{\tilde{P}(\mu)\right\}$, in the considered cases. For $\eta^\star \leq 10$ ms, the optimal solution is given by $B_1 = B_1^{(M)}$, while $B_i < B_i^{(M)}$, $i = 2, 3$, since point *ii*) of Theorem 4.2 applies. By the optimality of the solution, it is evident that the minimum of $\mathrm{Tr}\left\{\tilde{P}(\mu)\right\}$ is the same for any $\eta^\star \leq 10$ ms. Notice how for increasing $\eta^\star$, the $B_1^{(M)}$ increases as well. When $\eta^\star = 12$ ms, we have a full exploitation of resources with $B_i < B_i^{(M)}$, $i = 1, 2, 3$, $B_1 > B_2 > B_3$ and $B_1 + B_2 + B_3 = 1$ (recall the schedulability condition (6)) and, again, point *ii*) applies. By further increasing $\eta^\star$, we observe a slight increase of B_1 still preserving full exploitation of computing resources, i.e. $B_1 + B_2 + B_3 = 1$. When $\eta^\star = 28$ ms, $B_3 = B_3^{(m)}$, hence point *iii*) of Theorem 4.2 applies. As a consequence, for $\eta^\star > 28$ ms, the optimal allocation for a degenerate case applies.

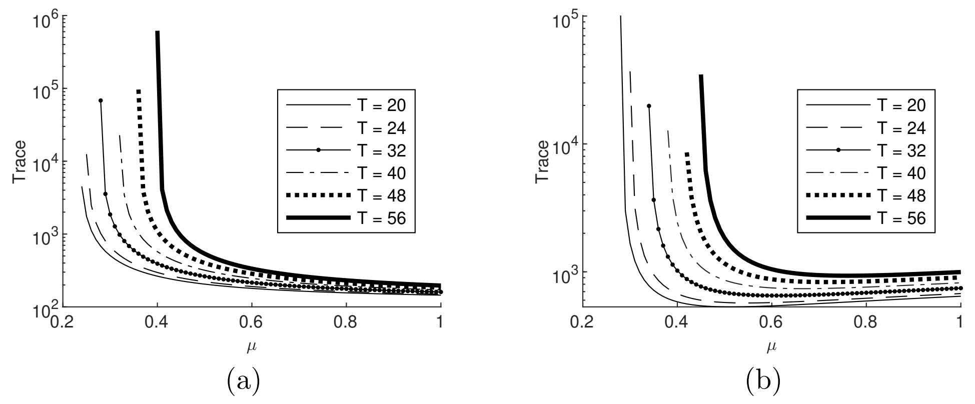

Figure 5. $\mathrm{Tr}\left\{\tilde{P}(\mu)\right\}$ as a function of the probability μ for the systems scheduled by task τ_2 (a) and τ_3 (b) specified in Table 3. The effect of variable execution periods T_i is also considered.

To further substantiate the numerical evaluation, we first present in Figure 5 the value of $\mathrm{Tr}\left\{\tilde{P}(\mu)\right\}$ as a function of the probability μ for the systems scheduled by task τ_2 and τ_3. It is evident that for the four-dimensional system (Figure 5-b), increasing the bandwidth (and hence μ) may lead to worse performance. Moreover, Figure 5 clearly shows the effects of an increased execution period T_i that leads to: i) an higher critical probability (recall Example 3.2); ii) an higher value of the trace (since the model noise $w_i(k)$ in (1) is integrated for a longer period).

We finally present a set of interesting experiments, where the task τ_1 has fixed mean value $\eta^\star = 14$ ms and variable period $T_1 \in \{20, 24, 32, 40\}$ ms. The results of the optimisation is hence influenced by the different periods (see Figure 6). This figure shows how increasing the period has evident benefits on the total allocated bandwidth with a tolerable price to pay on the control performance (the $\mathrm{Tr}\left\{\tilde{P}(\mu)\right\}$ remains almost constant across the selected periods).

6. Conclusions

We considered an application scenario where multiple tasks are used to implement independent feedback loops. The tasks are scheduled through a CBS scheduler and different choices of the bandwidth translate into different value for the trace of the steady state covariance (our QoC metric). We have tackled the problem of optimal bandwidth allocation showing its formulation as an optimisation problem and an efficient solution algorithm. We envisage future work directions in the possible use of different cost functions and of different computation models, where a task execution is not dropped as soon as it violates a deadline and moderate execution delays can be tolerated. As a final consideration, even if this work is directly motivated by a particular industrial problem (CPU sharing), the core of the results in Section 3 and the solution of the optimisation algorithm could potentially apply to a larger class of resource allocation problems for stochastic systems.

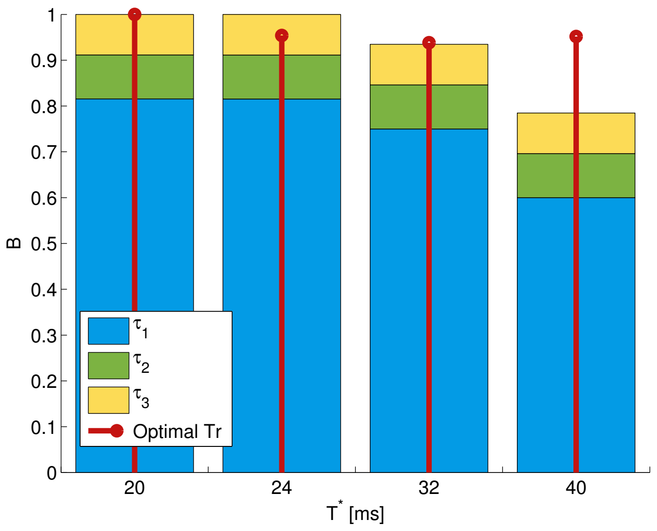

Figure 6. Bandwidth optimal allocation minimising $\mathrm{Tr}\{P\}$ with respect to the task set specified in Table 3 with $\eta^\star = 14$ ms and $T_1 \in \{20, 24, 32, 40\}$ ms. The solid thick line represents the normalised value of the adopted metrics.

References

Abeni, L., & Buttazzo, G. (1998, Dec.). Integrating Multimedia Applications in Hard Real-Time Systems. In *Proc. ieee real-time systems symposium* (pp. 4–13).

Aminifar, A., Bini, E., Eles, P., & Peng, Z. (2016, March). Analysis and design of real-time servers for control applications. *IEEE Transactions on Computers*, *65*(3), 834-846.

Aminifar, A., Tabuada, P., Eles, P., & Peng, Z. (2016, March). Self-triggered controllers and hard real-time guarantees. In *2016 design, automation test in europe conference exhibition (date)* (p. 636-641).

Bernstein, D. S. (2009). *Matrix mathematics: theory, facts, and formulas.* Princeton University Press.

Borgers, D., Postoyan, R., Anta, A., Tabuada, P., Nei, D., & Heemels, W. (2018). Periodic event-triggered control of nonlinear systems using overapproximation techniques. *Automatica*, *94*, 81-87.

Cervin, A., & Eker, J. (2003). The control server: A computational model for real-time control tasks. In *Ecrts* (p. 113-120). IEEE Computer Society.

Costa, O., Fragoso, M., & Marques, R. (2006). *Discrete-time markov jump linear systems.* Springer.

Elsner, L., & Szulc, T. (2000). Convex sets of schur stable and stable matrices. *Linear and Multilinear Algebra*, *48*(1), 1–19.

Fontanelli, D., Greco, L., & Palopoli, L. (2013, July). Soft RealTime Scheduling for Embedded Control Systems. *Automatica*, *49*, 2330–2338.

Fontanelli, D., & Palopoli, L. (2018, June). On soft real–time implementation of LQG Controllers. In *Ieee intl. symposium on industrial embedded systems (sies)* (p. 1-8). Graz, Austria: IEEE.

Fontanelli, D., Palopoli, L., & Greco, L. (2013, April). Optimal CPU Allocation to a Set of Control Tasks with Soft Real–Time Execution Constraints. In *Hybrid systems: Computation and control (hscc)* (pp. 233–242). Philadelphia, PA, USA: ACM.

Frías, B. V., Palopoli, L., Abeni, L., & Fontanelli, D. (2018, November). The PROSIT tool: Towards the optimal design of probabilistic soft real–time systems. *Journal of Software: Practice and Experience*, *48*(11), 1940-1967.

Fürst, S. (2010, March). Challenges in the design of automotive software. In *2010 design, automation test in europe conference exhibition (date 2010)* (p. 256-258).

Greco, L., Fontanelli, D., & Bicchi, A. (2011). Design and stability analysis for Anytime Control via stochastic scheduling. *IEEE Trans. on Automatic Control*, *56*(3), 571–585.

Kao, C., & Rantzer, A. (2007, June). Stability analysis of systems with uncertain time-varying delays. *Automatica*, *43*(6), 959-970.

Kopetz, H., & Bauer, G. (2003). The time-triggered architecture. *Proceedings of the IEEE*, *91*(1), 112–126.

Ling, Q., & Lemmon, M. (2002, Dec.). Robust performance of soft real-time networked control systems with data dropouts. In *Proc. ieee conf. on decision and control* (Vol. 2, pp. 1225–1230).

Liu, C. L., & Layland, J. (1973). Scheduling alghorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, *20*(1).

The long, winding road for driverless cars. (2016, May, 25th). The Economist.

Marti, P., Fuertes, J., Fohler, G., & Ramamritham, K. (2001, Dec.). Jitter compensation for real-time control systems. In *Proc. ieee real-time systems symposium* (pp. 39–48).

Mohamed, S., Awan, A. U., Goswami, D., & Basten, T. (2019). Designing image-based control systems considering workload variations. In IEEE (Ed.), *Proceeding of 58th conference on decision and control.* Nice, France.

Nilsson, J., & Bernhardsson, B. (1996, Dec.). Analysis of real-time control systems with time delays. In *Proc. ieee conf. on decision and control* (Vol. 3, pp. 3173–3178).

Palopoli, L., Pinello, C., Bicchi, A., & Sangiovanni-Vincentelli, A. (2005, Nov.). Maximizing the stability radius of a set of systems under real-time scheduling constraints. *IEEE Trans. on Automatic Control,*, *50*(11), 1790-1795.

Roberts, J. (2004). A survey on statistical bandwidth sharing. *Computer Networks*, *45*(3), 319 - 332. (In Memory of Olga Casals)

Seto, D., Lehoczky, J., Sha, L., & Shin, K. (1996). On task schedulability in real-time control systems. In *Ieee real-time systems symposium* (pp. 13–21).

Velasco, M., Martí, P., & Bini, E. (2008). Control-driven tasks: Modeling and analysis. In *Ieee real-time systems symposium* (p. 280-290). IEEE Computer Society.

Wang, X., & Lemmon, M. D. (2011). Event-triggering in distributed networked control systems. *IEEE Trans. Automat. Contr.*, *56*(3), 586-601.

Xu, Y., Cervin, A., & Årzén, K. E. (2016, Aug). Harmonic scheduling and control co-design. In *2016 ieee 22nd international conference on embedded and real-time computing systems and applications (rtcsa)* (p. 182-187).

Xu, Y., Cervin, A., & rzn, K. (2018, June). Jitter-robust lqg control and real-time scheduling co-design. In *2018 annual american control conference (acc)* (p. 3189-3196).

7. Appendix

Proof of Lemma 3.1

If μ is greater than the critical probability $\tilde{\mu}$, then the steady state covariance $\bar{P}$ converges and can be found identifying the equilibrium of the recursive equation (11):

$$\mathrm{vec}(\bar{P}) = S(\mu)^{-1}\mathrm{vec}(H), \tag{18}$$

with $S(\mu) \triangleq \Gamma_0 + \Gamma_1\mu$, $\Gamma_0 \triangleq I - A_o^{[2]}$ and $\Gamma_1 \triangleq A_o^{[2]} - A_c^{[2]}$, where $S(\mu)$ is invertible whenever Condition (12) holds, leading to a unique solution for the equilibrium.

Recalling that $\mathrm{Tr}\{AB\} = \mathrm{vec}(A^T)^T\mathrm{vec}(B)$ and (18), we can write $\mathrm{Tr}\{\bar{P}(\mu)\} = \mathrm{vec}(I)^T\mathrm{vec}(\bar{P}(\mu)) = \mathrm{vec}(I)^T S(\mu)^{-1}\mathrm{vec}(H)$. In order to compute $S(\mu)^{-1}$ we make use of the recursive algorithm in Bernstein (2009), Fact 2.16.28, Applying this fact if we construct a sequence of matrices $\Lambda_{ij} \in \mathbb{R}^{n_{\tilde{x}}^2 \times n_{\tilde{x}}^2}$ using the following recursion (for $i = 1, \dots, n_{\tilde{x}}^2 - 1$)

$$\begin{aligned}
&\Lambda_{00} = I, \;\; \Lambda_{i0} = \Gamma_0 \Lambda_{i-1,0} - \frac{1}{i}\mathrm{Tr}\,\{\Gamma_0 \Lambda_{i-1,0}\} I, \\
&\Lambda_{ij} = \Gamma_0 \Lambda_{i-1,j} + \Gamma_1 \Lambda_{i-1,j-1} + \\
&\quad - \frac{1}{i}\mathrm{Tr}\,\{\Gamma_0 \Lambda_{i-1,j} + \Gamma_1 \Lambda_{i-1,j-1}\} I, \, \forall j = 1, \dots, i-1, \\
&\Lambda_{ii} = \Gamma_1 \Lambda_{i-1,i-1} - \frac{1}{i}\mathrm{Tr}\,\{\Gamma_1 \Lambda_{i-1,i-1}\} I.
\end{aligned}$$

then we can compute $S(\mu)^{-1}$ as

$$S(\mu)^{-1} = \frac{n_{\tilde{x}}^2}{\sum_{j=0}^{n_{\tilde{x}}^2-1} \gamma_j \mu^j} \sum_{j=0}^{n_{\tilde{x}}^2-1} \Lambda_{n-1,j} \mu^j, \tag{19}$$

with

$$\begin{aligned}
&\gamma_0 \triangleq \mathrm{Tr}\left\{\Gamma_0 \Lambda_{n_{\tilde{x}}^2-1,0}\right\}, \;\; \gamma_{n_{\tilde{x}}^2-1} \triangleq \mathrm{Tr}\left\{\Gamma_1 \Lambda_{n_{\tilde{x}}^2-1, n_{\tilde{x}}^2-1}\right\}, \\
&\gamma_j \triangleq \mathrm{Tr}\left\{\Gamma_0 \Lambda_{n_{\tilde{x}}^2-1,j} + \Gamma_1 \Lambda_{n_{\tilde{x}}^2-1,j-1}\right\}, \, \forall j = 1, \dots, n_{\tilde{x}}^2 - 2.
\end{aligned}$$

This leads us to (13), where

$$\alpha_j \triangleq \mathrm{vec}(I)^T \Lambda_{n_{\tilde{x}}^2-1,j} \mathrm{vec}(H), \, \forall j = 0, \dots, n_{\tilde{x}}^2 - 1.$$

Proof of Theorem 4.2

We start with some preliminary considerations. By the non degenerate hypothesis we have that $\bar{t}^{(M)} \leq \underline{t}^{(m)}$, hence the set $[\bar{t}^{(M)}, \underline{t}^{(m)}]$ is not empty. If we define $t_i^{(M)} \triangleq \phi_i^{(rect)}(B_i^{(M)})$ and $t_i^{(m)} \triangleq \phi_i^{(rect)}(B_i^{(m)})$ for every $i \in \{1, \dots, n\}$, it is easy to see that $[\bar{t}^{(M)}, \underline{t}^{(m)}] = \cap_{i=1}^n [t_i^{(M)}, t_i^{(m)}]$, thus $[\bar{t}^{(M)}, \underline{t}^{(m)}] \subseteq [t_i^{(M)}, t_i^{(m)}]$. As a consequence, being all the $\phi_i^{(rect)}(\cdot)$ surjective, the points $\check{B}^{(m)}$ and $\check{B}^{(M)}$ exist and are unique (owing to the minimum operator).

Proof of i). If $\sum_{i=1}^n \check{B}_i^{(M)} \leq 1$, $\check{B}^{(M)}$ is feasible. Moreover, $\Phi(\check{B}^{(M)}) = \bar{t}^{(M)}$, which is the minimal value achievable by $\Phi(\cdot)$ on $\mathcal{B}$. Hence, $\check{B}^{(M)} \in \mathcal{X}^{(rect)^*}$.

Proof of ii). If $\sum_{i=1}^n \check{B}_i^{(m)} \leq 1$ then the optimal value t^* is such that $t^* \leq \underline{t}^{(m)}$. By the fact that $\sum_{i=1}^n \check{B}_i^{(M)} > 1$, the non-increasing property of all $\phi_i^{(rect)}(\cdot)$ and the definition of $\check{B}_i^{(M)}$ in (17), we can deduce that $t^* > \bar{t}^{(M)}$. By the same arguments we also conclude that $\bar{t}^{(M)} \neq \underline{t}^{(m)}$, thus the set $(\bar{t}^{(M)}, \underline{t}^{(m)}]$ is not empty. If $\sum_{i=1}^n \check{B}_i^{(m)} = 1$, then clearly $\tilde{B} = \check{B}^{(m)}$ is the optimal point and $t^* = \underline{t}^{(m)}$ is the optimal value. Hence, we will henceforth assume $\sum_{i=1}^n \check{B}_i^{(m)} < 1$.

The conditions $\sum_{i=1}^n \check{B}_i^{(m)} < 1$ and $\sum_{i=1}^n \check{B}_i^{(M)} > 1$ are equivalent to state that $\check{B}^{(M)}$ and $\check{B}^{(m)}$ are points laying in the two opposite sides of $\mathbb{R}^n$ with respect to the hyperplane $\mathcal{P} \triangleq \{(B_1, \dots, B_n) \in \mathbb{R}^n \mid \sum_{i=1}^n B_i = 1\}$. By definition, $\check{B}^{(m)}$ and $\check{B}^{(M)}$ belong to the set

$$\mathcal{G} \triangleq \{B \in \mathbb{R}_{\geq 0}^n \mid \phi_i^{(rect)}(B_i) = \phi_j^{(rect)}(B_j), \, \forall i, j, \, i \neq j\}. \tag{20}$$

In order to show that a point $\tilde{B}$, as defined in the claim, exists, we must show first that $\mathcal{G}$ is compact and path-connected. In this case, indeed, we can easily construct a continuous path connecting $\check{B}^{(M)}$ and $\check{B}^{(m)}$, whose image belongs to $\mathcal{G}$. Such a path would have at least one point $\tilde{B}$ of intersection with the hyperplane $\mathcal{P}$.

The continuous functions $\phi_i^{(rect)} : [B_i^{(m)}, B_i^{(M)}] \to [t_i^{(M)}, t_i^{(m)}]$ are surjective but only almost everywhere injective. Indeed, let us consider a point $\check{t} \in [t_i^{(M)}, t_i^{(m)}]$ for which the injectivity is lost. By the non-increasing property of $\phi_i^{(rect)}(\cdot)$ we know that there exists a closed, connected, not empty interval $\mathcal{I}_i^{\check{t}} \triangleq [\underline{B}_i^{\check{t}}, \overline{B}_i^{\check{t}}]$, with $\underline{B}_i^{\check{t}} \triangleq \min_{\alpha \in \mathcal{A}_i(\check{t})} \alpha$ and $\overline{B}_i^{\check{t}} \triangleq \max_{\alpha \in \mathcal{A}_i(\check{t})} \alpha$, such that for any $B \in \mathcal{I}_i^{\check{t}}$ we have $\phi_i^{(rect)}(B) = \check{t}$. In the light of Remark 1 we can say that there are $q_i \in \mathbb{Z}_{\geq 0}$ such points (and intervals) and that q_i is less or equal than the number of local minima of $\phi_i(\cdot)$ in the set $[B_i^{(m)}, B_i^{(M)}]$. Let us name the finite set of points where the injectivity of $\phi_i^{(rect)}(\cdot)$ is lost as $\mathcal{T}^i \triangleq \{t_1^i, \dots, t_{q_i}^i\}$. We can define the generalised inverse of the function $\phi_i^{(rect)}(\cdot)$, $i \in \{1, \dots, n\}$ as follows:

$$\left[\phi_i^{(rect)}\right]^{-1} : [t_i^{(M)}, t_i^{(m)}] \to [B_i^{(m)}, B_i^{(M)}]$$

$$\left[\phi_i^{(rect)}\right]^{-1}(t) = \begin{cases} \phi_i^{-1}(t) & \text{for } t \in [t_i^{(M)}, t_i^{(m)}] \setminus \mathcal{T}^i \\ \mathcal{I}_i^t & \text{for } t \in \mathcal{T}^i. \end{cases} \tag{21}$$

It is worth noting that $\left[\phi_i^{(rect)}\right]^{-1}(\cdot)$ is not a function, but a multi-valued mapping. It can be verified that it is continuous. Let us consider now the multi-valued mapping [3]

$$\Psi : [\overline{t}^{(M)}, \underline{t}^{(m)}] \to \bigtimes_{i=1}^{n} [\check{B}_i, \max_{\alpha \in \mathcal{A}_i(\overline{t}^{(M)})} \alpha]$$
$$\Psi(t) \triangleq \bigtimes_{i=1}^{n} \left[\phi_i^{(rect)}\right]^{-1}(t).$$

It is a continuous mapping by virtue of the continuity of all $\left[\phi_i^{(rect)}\right]^{-1}(\cdot)$, but it is not surjective. Indeed, it is easy to verify that the image of $[\overline{t}^{(M)}, \underline{t}^{(m)}]$ along $\Psi(\cdot)$ is exactly the set $\mathcal{G}$ defined in (20). $\mathcal{G}$ is a compact set made up of continuous paths and hypercubes (see the definition of $\left[\phi_i^{(rect)}\right]^{-1}(\cdot)$ in (21)). In particular, for any point $\check{t} \in [\overline{t}^{(M)}, \underline{t}^{(m)}]$ such that $\check{t} \in \mathcal{T}^{i_j}$ for every $i_j \in \{1, \dots, n\}$, $j \in \{1, \dots, l_{\check{t}}\}$, $l_{\check{t}} \leq n$, we can define the $l_{\check{t}}$-dimensional hypercube $\mathcal{H}_{\check{t}} \subset \mathcal{G}$ as $\mathcal{H}_{\check{t}} \triangleq \bigtimes_{i=i_1}^{i_{l_{\check{t}}}} \mathcal{I}_i^{\check{t}}$. By the continuity of $\Psi(\cdot)$ we have $\lim_{t \downarrow \check{t}} \Psi(t) \in \mathcal{H}_{\check{t}}$ and $\lim_{t \uparrow \check{t}} \Psi(t) \in \mathcal{H}_{\check{t}}$, hence $\mathcal{G}$ is path-connected and in particular we can construct a path belonging to $\mathcal{G}$ and connecting $\check{B}^{(m)}$ and $\check{B}^{(M)}$ as desired. Let us name $\tilde{B}$ a point of intersection of such path with the hyperplane $\mathcal{P}$. We must show now that it is actually an optimal point. We proceed by contradiction by assuming that there exists a feasible optimal point $B' = (B'_1, \dots, B'_n)$ such that $\phi_i^{(rect)}(B'_i) < \tilde{t} \triangleq \Phi^{(rect)}(\tilde{B})$. The latter inequality, the fact that in $\tilde{B}$ we have $\phi_i^{(rect)}(\tilde{B}_i) = \phi_j^{(rect)}(\tilde{B}_j)$ for every $i, j \in \{1, \dots, n\}$ and the non-increasing property of

[3]With the symbol $\times_{i=1}^{n}$ we mean the cartesian product of n indexed objects.

$\phi_i^{(rect)}(\cdot)$ require that $B'_i > \tilde{B}_i$ for every $i \in \{1, \ldots, n\}$. But, being $\sum_{i=1}^n \tilde{B}_i = 1$, the point B' is not feasible. Hence, $\tilde{B}$ is optimal.

Proof of iii). If $\sum_{i=1}^n \check{B}_i^{(m)} > 1$ clearly $t^* \in (\underline{t}^{(m)}, \bar{t}^{(m)}]$, and the feasibility set has been required to be non-empty. We show now that there exists an optimal point B^* as defined in the thesis of the theorem. Assume that there exists another optimal point $B' = (B'_1, \ldots, B'_n)$ such that $B'_{\bar{h}} \neq B_{\bar{h}}^{(m)}$, then necessarily $B'_{\bar{h}} > B_{\bar{h}}^{(m)}$. We know that $t' \triangleq \Phi^{(rect)}(B') \in (\underline{t}^{(m)}, \bar{t}^{(M)}]$, but $\phi_{\bar{h}}^{(rect)}(B'_{\bar{h}}) < \phi_{\bar{h}}^{(rect)}(B_{\bar{h}}^{(m)}) = \underline{t}^{(m)}$, hence $B'_{\bar{h}}$ does not influence the optimal value t', that is $t' = \max_{j \in \{1,\ldots,n\}/\{\bar{h}\}} \phi_j^{(rect)}(B'_j)$. The point B' is feasible by definition, hence $\sum_{i=1}^n B'_i \leq 1$. Let us define the point B^* such that $B^*_{\bar{h}} = B_{\bar{h}}^{(m)}$ and $B^*_j = B'_j$ for every $j \in \{1, \ldots, n\}/\{\bar{h}\}$. The point B^* is feasible since $\sum_{i=1}^n B^*_i = \sum_{i=1}^n B'_i - B'_{\bar{h}} + B^*_{\bar{h}}$ and $B^*_{\bar{h}} = B_{\bar{h}}^{(m)} < B'_{\bar{h}}$. This proves the optimality of B^*.

Proof of Corollary 4.3

Proof of i). The point B^* is clearly feasible since $B^*_i \leq \tilde{B}^*_i$ and $B^*_i \in [B_i^{(m)}, B_i^{(M)}]$ for every $i \in \{1, \ldots, n\}$. Moreover, by the definition of B^* we have $\phi_i^{(rect)}(B^*_i) = \phi_i^{(rect)}(\tilde{B}^*_i)$ for every $i \in \{1, \ldots, n\}$ and, by means of the Remark 1, also $\phi_i(B^*_i) = \phi_i^{(rect)}(B^*_i)$. Therefore, the optimal value is $\Phi(B^*) = \Phi^{(rect)}(\tilde{B}^*)$. By the definition of rectified function (15) we know that $\phi_i(b) \geq \phi_i^{(rect)}(b)$ for every $b \in [B_i^{(m)}, B_i^{(M)}]$ and every $i \in \{1, \ldots, n\}$, thus $\phi_i(b) \geq \phi_i^{(rect)}(B^*_i)$ for every $b \in [B_i^{(m)}, B_i^{(M)}]$, which implies the optimality of B^*.

The proof of ii) is a straightforward consequence of the definition of rectified function and of the Remark 1.