TEMPERATURE CONTROL OF A SHRINK TUNNEL WITH MULTIPLE HEATING ZONES

SUPPLEMENTARY MATERIAL

Davide Previtali
Department of Management, Information and Production Engineering, University of Bergamo,
Via G. Marconi 5, 24044 Dalmine (BG), Italy
davide.previtali@unibg.it

Leandro Pitturelli
Department of Management, Information and Production Engineering, University of Bergamo,
Via G. Marconi 5, 24044 Dalmine (BG), Italy
leandro.pitturelli@unibg.it

Fabio Previdi
Department of Management, Information and Production Engineering, University of Bergamo,
Via G. Marconi 5, 24044 Dalmine (BG), Italy
fabio.previdi@unibg.it

Antonio Ferramosca
Department of Management, Information and Production Engineering, University of Bergamo,
Via G. Marconi 5, 24044 Dalmine (BG), Italy
antonio.ferramosca@unibg.it

ABSTRACT

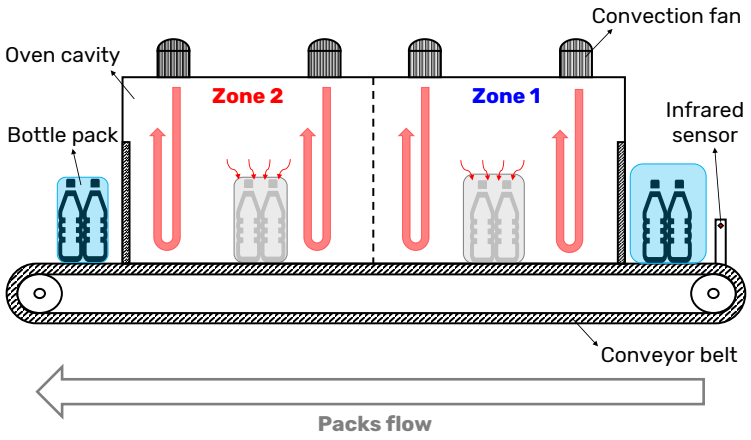
This paper provides additional documentation for the Book Chapter "Temperature control of a shrink tunnel with multiple heating zones" [11]. In particular, we report the mathematical expressions for the matrices that govern the `Heat-Temperatures` and `Packs-Temperatures` models (Section 1.2), the estimation of the parameters for the overall shrink tunnel model (Section 2), and a software guide for the MATLAB/Simulink benchmark available at the GitHub repository `https://github.com/Lenyor/Temperature_control_of_a_shrink_tunnel_with_multiple_heating_zones` (Section 3).

1 Review of the shrink tunnel model

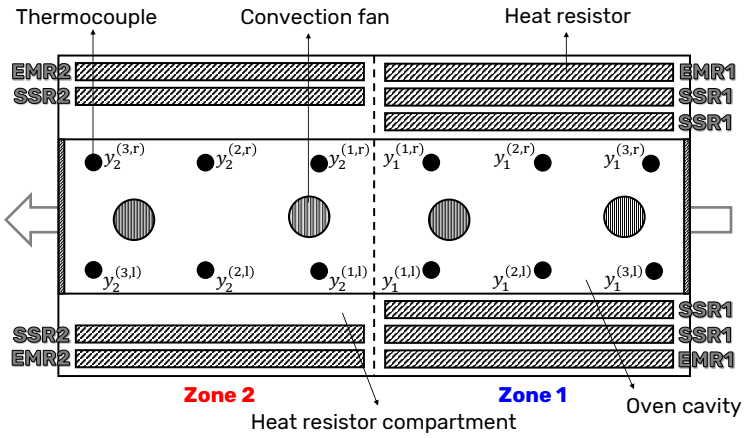
This Section reviews the lumped-parameter state-space model for the shrink tunnel under study, which amounts to:

$$\begin{cases} \boldsymbol{V}_{\text{sq}}(t) = \boldsymbol{f}(\boldsymbol{u}(t), V_{\text{g}}(t)) \\ \dot{\boldsymbol{q}}^{(\text{f})}(t) = A_{ff} \cdot \boldsymbol{q}^{(\text{f})}(t) + B_f \cdot \boldsymbol{V}_{\text{sq}}(t) \\ \dot{\boldsymbol{T}}(t) = A_{TT} \cdot \boldsymbol{T}(t) + A_{Tf} \cdot \boldsymbol{q}^{(\text{f})}(t) + \boldsymbol{b}_T \cdot T_{\text{a}}(t) \\ \delta\dot{\boldsymbol{T}}(t) = A_{dd}(v(t)) \cdot \delta\boldsymbol{T}(t) + \boldsymbol{b}_d(v(t)) \cdot d(t; v(t)) \\ y_{T_i}^{(l,s)}(t) = T_i^{(l,s)}\left(t + L_{T_i}^{(l,s)}\right) \qquad \text{for } i \in \{1,2\}, l \in \{1,2,3\}, s \in \{\text{l},\text{r}\} \\ y_{d_i}^{(l,s)}(t) = \delta T_i^{(l,s)}\left(t + L_{d_i}^{(l,s)}(v(t))\right) \\ y_i^{(l,s)}(t) = y_{T_i}^{(l,s)}(t) + y_{d_i}^{(l,s)}(t) \end{cases} \tag{1}$$

The equations in orange constitute the `Heat-Temperatures` model while those in brown make up the `Packs-Temperatures` model. The outputs of the shrink tunnel model, i.e. the temperatures $y_i^{(l,s)}(t)$, $i \in \{1,2\}, l \in \{1,2,3\}, s \in \{\text{l},\text{r}\}$, measured at each spot depicted in Figure 1, are the sum of the outputs of the two just mentioned models. The reader is referred to [11] for the derivation of (1).



(a) Front view.



(b) Top view.

Figure 1: Schematic of the considered shrink tunnel. $y_i^{(l,s)}$, $i \in \{1,2\}$, $l \in \{1,2,3\}$, $s \in \{\text{l},\text{r}\}$, denote the temperatures measured by the twelve thermocouples. The heat resistors in zone i denoted by EMRi are driven by the same electromechanical relay while those denoted by SSRi are driven by the same solid-state relay.

1.1 Recap of the signals

Before reviewing the `Heat-Temperatures` and `Packs-Temperatures` models, we recap the signals of interest for the shrink tunnel described in [11], whose control scheme is reported in Figure 2. Each signal represents [11]:

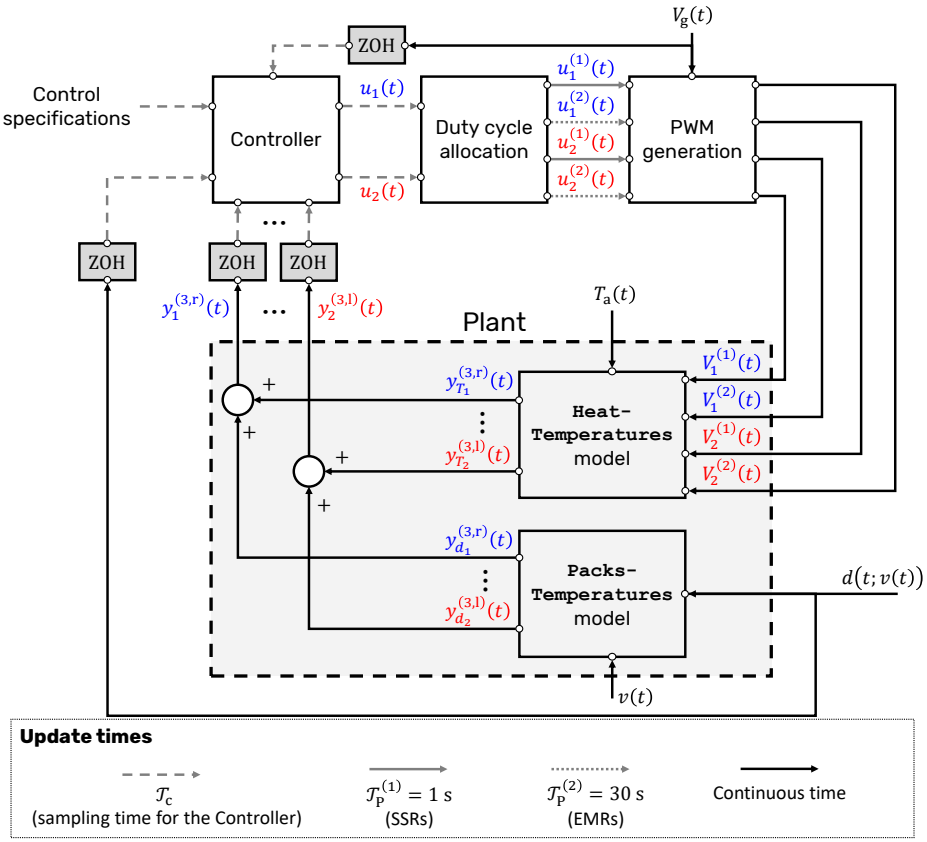


Figure 2: Block diagram of the system under study when operated in closed-loop. The signals in blue are associated with the first zone while those in red are related to the second zone. Instead, the signals in black are not related to any zone in particular. The different arrows highlight how often each signal is updated. Finally, the Zero-Order Holder (ZOH) blocks sample the relevant signals at a sampling time $\mathcal{T}_c$.

- $t \in \mathbb{R}_{\geq 0}$ (in s) is the time.
- $u_1(t), u_2(t) \in [0,1]$ are the control actions produced by the temperature controller for the first and the second heating zone respectively. In particular, $u_1(t)$ and $u_2(t)$ are related to the duty cycles associated with the voltage Pulse-Width Modulated (PWM) signals that need to be supplied to the heat resistors. Notation-wise:
$$\boldsymbol{u}(t) = \begin{bmatrix} u_1(t) & u_2(t) \end{bmatrix}^\top \in [0,1]^2. \tag{2}$$
- $u_1^{(1)}(t)$ and $u_2^{(1)}(t)$ are the duty cycles for the voltage PWM signals across the sets of heat resistors in the first and the second heating zones that are driven by the Solid-State Relays (SSRs). Instead, $u_1^{(2)}(t)$ and $u_2^{(2)}(t)$ are the duty cycles related to the Electro-Mechanical Relays (EMRs). $u_1^{(1)}(t), u_1^{(2)}(t), u_2^{(1)}(t), u_2^{(2)}(t) \in [0,1]$ are generated from $u_1(t), u_2(t)$ based on the Operation Modes (`OMs`) described in [11]. Furthermore, their respective PWM periods are $\mathcal{T}_P^{(1)} = 1$ s (SSRs) and $\mathcal{T}_P^{(2)} = 30$ s (EMRs).
- $V_1^{(1)}(t), V_1^{(2)}(t), V_2^{(1)}(t), V_2^{(2)}(t) \in \mathbb{R}_{\geq 0}$ (in V) are the voltage PWM signals across the sets of heat resistors in each zone with corresponding duty cycles $u_1^{(1)}(t), u_1^{(2)}(t), u_2^{(1)}(t), u_2^{(2)}(t)$. Notation-wise:
$$\boldsymbol{V}(t) = \begin{bmatrix} V_1^{(1)}(t) & V_1^{(2)}(t) & V_2^{(1)}(t) & V_2^{(2)}(t) \end{bmatrix}^\top \in \mathbb{R}^4_{\geq 0}, \tag{3a}$$
$$\boldsymbol{V}_{sq}(t) = \boldsymbol{V}(t) \odot \boldsymbol{V}(t). \tag{3b}$$

- $V_{\mathrm{g}}(t) \in \mathbb{R}_{\geq 0}$ (in V) is the grid voltage.
- $T_{\mathrm{a}}(t) = \bar{T}_{\mathrm{a}}, \forall t \in \mathbb{R}_{\geq 0}, \bar{T}_{\mathrm{a}} \in \mathbb{R}$ (in °C), is the ambient temperature.
- $v(t) \in \{0, 30, 60, 90\}$ (in ppm) is the production rate of the industrial process which produces the beverages fed to the shrink tunnel.
- $d(t; v(t)) \in \{0, 1\}$ is the reading of the infrared sensor at the entrance of the oven which detects the presence (1) or absence (0) of bottle packs (see Figure 1).
- $y_i^{(l,s)}(t) \in \mathbb{R}, i \in \{1, 2\}, l \in \{1, 2, 3\}, s \in \{\mathrm{l}, \mathrm{r}\}$ (in °C), are the overall temperatures within the oven cavity measured by the twelve thermocouples positioned at the different locations depicted in Figure 1. Notation-wise:
$$\boldsymbol{y}(t) = \begin{bmatrix} y_1^{(3,\mathrm{r})}(t) & y_1^{(3,\mathrm{l})}(t) & y_1^{(2,\mathrm{r})}(t) & \dots & y_2^{(2,\mathrm{l})}(t) & y_2^{(3,\mathrm{r})}(t) & y_2^{(3,\mathrm{l})}(t) \end{bmatrix}^\top \in \mathbb{R}^{12}. \tag{4}$$
- $y_{T_i}^{(l,s)}(t) \in \mathbb{R}, i \in \{1, 2\}, l \in \{1, 2, 3\}, s \in \{\mathrm{l}, \mathrm{r}\}$ (in °C), are the temperatures due to the heat produced by the heat resistors and the ambient temperature. They are the outputs of the `Heat-Temperatures` model. Notation-wise:
$$\boldsymbol{y}_T(t) = \begin{bmatrix} y_{T_1}^{(3,\mathrm{r})}(t) & y_{T_1}^{(3,\mathrm{l})}(t) & y_{T_1}^{(2,\mathrm{r})}(t) & \dots & y_{T_2}^{(2,\mathrm{l})}(t) & y_{T_2}^{(3,\mathrm{r})}(t) & y_{T_2}^{(3,\mathrm{l})}(t) \end{bmatrix}^\top \in \mathbb{R}^{12}. \tag{5}$$
- $y_{d_i}^{(l,s)}(t) \in \mathbb{R}, i \in \{1, 2\}, l \in \{1, 2, 3\}, s \in \{\mathrm{l}, \mathrm{r}\}$ (in °C), are the temperature drops due to the flow of bottle packs inserted into the oven cavity. They are the outputs of the `Packs-Temperatures` model. Notation-wise:
$$\boldsymbol{y}_d(t) = \begin{bmatrix} y_{d_1}^{(3,\mathrm{r})}(t) & y_{d_1}^{(3,\mathrm{l})}(t) & y_{d_1}^{(2,\mathrm{r})}(t) & \dots & y_{d_2}^{(2,\mathrm{l})}(t) & y_{d_2}^{(3,\mathrm{r})}(t) & y_{d_2}^{(3,\mathrm{l})}(t) \end{bmatrix}^\top \in \mathbb{R}^{12}. \tag{6}$$

In what follows, we review the `Heat-Temperatures` and `Packs-Temperatures` models more in detail.

1.2 Review of the `Heat-Temperatures` model

The `Heat-Temperatures` model describes the relationship between the heat produced by the heat resistors, the ambient temperature, and the temperatures measured by the twelve thermocouples installed inside the shrink tunnel. The derived state-space model is [11]:

$$\begin{cases} \boldsymbol{V}_{\mathrm{sq}}(t) = \boldsymbol{f}(\boldsymbol{u}(t), V_{\mathrm{g}}(t)) \\ \dot{\boldsymbol{q}}^{(\mathrm{f})}(t) = A_{ff} \cdot \boldsymbol{q}^{(\mathrm{f})}(t) + B_f \cdot \boldsymbol{V}_{\mathrm{sq}}(t) \\ \dot{\boldsymbol{T}}(t) = A_{TT} \cdot \boldsymbol{T}(t) + A_{Tf} \cdot \boldsymbol{q}^{(\mathrm{f})}(t) + \boldsymbol{b}_T \cdot T_{\mathrm{a}}(t) \\ y_{T_i}^{(l,s)}(t) = T_i^{(l,s)}\left(t + L_{T_i}^{(l,s)}\right) \quad \text{for } i \in \{1, 2\}, l \in \{1, 2, 3\}, s \in \{\mathrm{l}, \mathrm{r}\}. \end{cases} \tag{7}$$

$\boldsymbol{f}(\boldsymbol{u}(t), V_{\mathrm{g}}(t))$ is a nonlinear function that describes the relationship between the control actions $\boldsymbol{u}(t)$ produced by the controller, the grid voltage $V_{\mathrm{g}}(t)$, and the squared voltages $\boldsymbol{V}_{\mathrm{sq}}(t)$, see [11] for its derivation. In this Section, we focus on the linear part of the system in (7), which is related to the electro-equivalent thermal circuit reported in Figure 3. In particular, let $i \in \{1, 2\}$, $l \in \{1, 2, 3\}$, $l' \in \{1, 2\}$, and $s \in \{\mathrm{l}, \mathrm{r}\}$. The elements that constitute the electro-equivalent thermal circuit are:

- $q_i(t) \in \mathbb{R}_{\geq 0}$ (in $\frac{\mathrm{J}}{\mathrm{s}}$) is the heat flow rate produced by the heat resistors located in zone i. We assume that the heat generated in each zone is equally spread among each of its locations/sides;
- $R_i^{(l,s)} \in \mathbb{R}_{>0}$ (in $\frac{^\circ\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}$) is the thermal resistance between the oven walls of zone i, location l, side s, and the ambient;
- $R_{\mathrm{T}_i}^{(l',s)} \in \mathbb{R}_{>0}$ (in $\frac{^\circ\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}$) is the thermal resistance related to the transfer of heat between two locations of zone i, side s;
- $R_{\mathrm{T}_{12}}^{(s)} \in \mathbb{R}_{>0}$ (in $\frac{^\circ\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}$) is the thermal resistance related to the transfer of heat between the two zones of the oven, for what concerns side s;
- $R_{\mathrm{T_{rl}}} \in \mathbb{R}_{>0}$ (in $\frac{^\circ\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}$) is the thermal resistance related to the transfer of heat between the left and the right side of the oven cavity. For the sake of simplicity, we assume it to be equal in each zone and each location;

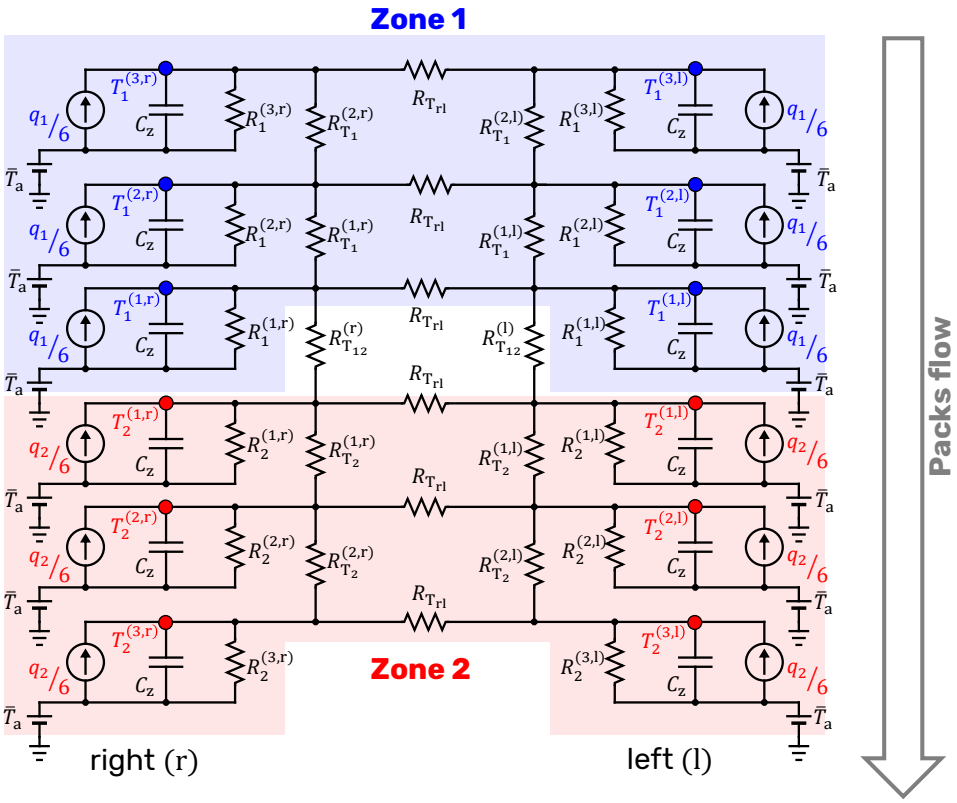


Figure 3: Electro-equivalent thermal circuit describing the relationship between the temperatures and the heat generated in each zone.

- $C_z \in \mathbb{R}_{>0}$ (in $\frac{\text{J}}{^\circ\text{C}}$) is the thermal capacitance of the air in each zone/location/side of the oven cavity, which all have the same volume. Given that the oven cavity is divided into twelve "boxes" of the same volume, and assuming that we are dealing with dry air at constant pressure, we have [3]:

$$\begin{aligned} C_z &= c_{\text{air}} \cdot \rho_{\text{air}} \cdot \frac{\mathcal{V}_{\text{oven}}}{12} \\ &= 1000 \, \frac{\text{J}}{\text{Kg} \cdot ^\circ \text{C}} \cdot 1.225 \, \frac{\text{Kg}}{\text{m}^3} \cdot \frac{2.26 \, \text{m}^3}{12} \\ &= 231 \, \frac{\text{J}}{^\circ\text{C}}, \end{aligned} \tag{8}$$

  where c_{air} is the specific heat of air, ρ_{air} is the air density, and $\mathcal{V}_{\text{oven}}$ is the volume of the oven cavity.

In what follows, we report the mathematical expressions for the matrices and vectors $A_{ff} \in \mathbb{R}^{4\times4}, B_f \in \mathbb{R}^{4\times4}, A_{TT} \in \mathbb{R}^{12\times12}, A_{Tf} \in \mathbb{R}^{12\times4}$, and $\boldsymbol{b}_T \in \mathbb{R}^{12}$, that govern the linear part of the system in (7), which are related to the aforementioned parameters, the time constants $\tau_1^{(\text{f})}, \tau_2^{(\text{f})} \in \mathbb{R}_{>0}$ (one per zone, in s) associated with the propagation of heat from the heat resistors to the air inside the oven cavity, and the ohmic resistance of the heat resistors pairs $R_{\text{heat}} \in \mathbb{R}$ (in Ω), as described in [11].

Let $a, b \in \mathbb{N}$. Notation-wise, we denote the a-dimensional column vectors of ones and zeros as $\mathbf{0}_a$ and $\mathbf{1}_a$ respectively, the $a \times b$ zero matrix as $0_{a\times b}$, and the identity matrix of size a as I_a.

First, we consider the differential equation governing the circuit in Figure 3, which is:

$$\dot{\boldsymbol{T}}(t) = A_{TT} \cdot \boldsymbol{T}(t) + B_q \cdot \boldsymbol{q}(t) + \boldsymbol{b}_T \cdot T_{\text{a}}(t). \tag{9}$$

The expressions for the matrices and vectors in (9) can be readily derived from the electro-equivalent thermal circuit and amount to:

$$B_q = \frac{1}{6 \cdot C_z} \cdot \begin{bmatrix} \mathbf{1}_6 & \mathbf{0}_6 \\ \mathbf{0}_6 & \mathbf{1}_6 \end{bmatrix}, \tag{10}$$

Table 1: Non-zero entries of matrix A_{TT} for the circuit in Figure 3. All the remaining entries are zero.

Entries	i	s	**Value**
$A_{TT}^{(1,1)}$	1	r	$-\frac{1}{C_z}\cdot\left(\frac{1}{R_i^{(3,s)}}+\frac{1}{R_{T_i}^{(2,s)}}+\frac{1}{R_{T_{rl}}}\right)$
$A_{TT}^{(2,2)}$	1	l	
$A_{TT}^{(11,11)}$	2	r	
$A_{TT}^{(12,12)}$	2	l	
$A_{TT}^{(3,3)}$	1	r	$-\frac{1}{C_z}\cdot\left(\frac{1}{R_i^{(2,s)}}+\frac{1}{R_{T_i}^{(1,s)}}+\frac{1}{R_{T_i}^{(2,s)}}+\frac{1}{R_{T_{rl}}}\right)$
$A_{TT}^{(4,4)}$	1	l	
$A_{TT}^{(9,9)}$	2	r	
$A_{TT}^{(10,10)}$	2	l	
$A_{TT}^{(5,5)}$	1	r	$-\frac{1}{C_z}\cdot\left(\frac{1}{R_i^{(1,s)}}+\frac{1}{R_{T_i}^{(1,s)}}+\frac{1}{R_{T_{12}}^{(s)}}+\frac{1}{R_{T_{rl}}}\right)$
$A_{TT}^{(6,6)}$	1	l	
$A_{TT}^{(7,7)}$	2	r	
$A_{TT}^{(8,8)}$	2	l	
$A_{TT}^{(1,3)}, A_{TT}^{(3,1)}$	1	r	$\frac{1}{C_z\cdot R_{T_i}^{(2,s)}}$
$A_{TT}^{(2,4)}, A_{TT}^{(4,2)}$	1	l	
$A_{TT}^{(9,11)}, A_{TT}^{(11,9)}$	2	r	
$A_{TT}^{(10,12)}, A_{TT}^{(12,10)}$	2	l	
$A_{TT}^{(3,5)}, A_{TT}^{(5,3)}$	1	r	$\frac{1}{C_z\cdot R_{T_i}^{(1,s)}}$
$A_{TT}^{(4,6)}, A_{TT}^{(6,4)}$	1	l	
$A_{TT}^{(7,9)}, A_{TT}^{(9,7)}$	2	r	
$A_{TT}^{(8,10)}, A_{TT}^{(10,8)}$	2	l	
$A_{TT}^{(5,7)}, A_{TT}^{(7,5)}$	–	r	$\frac{1}{C_z\cdot R_{T_{12}}^{(s)}}$
$A_{TT}^{(6,8)}, A_{TT}^{(8,6)}$	–	l	
$A_{TT}^{(a,a+1)}, a=1,3,\ldots,11$ (odd)	–	–	$\frac{1}{C_z\cdot R_{T_{rl}}}$
$A_{TT}^{(a,a-1)}, a=2,4,\ldots,12$ (even)	–	–	

$$\boldsymbol{b}_T = \frac{1}{C_z}\cdot\left[\frac{1}{R_1^{(3,r)}} \quad \frac{1}{R_1^{(3,l)}} \quad \frac{1}{R_1^{(2,r)}} \quad \cdots \quad \frac{1}{R_2^{(2,l)}} \quad \frac{1}{R_2^{(3,r)}} \quad \frac{1}{R_2^{(3,l)}}\right]^\top. \tag{11}$$

Instead, A_{TT} is a matrix whose (i,j)-th entry $A_{TT}^{(i,j)}$ is zero except for those reported in Table 1.

Next, let us consider the state equation for the filtered heat flow rates $\boldsymbol{q}^{(\mathrm{f})}(t)$ in (7). Recall that [11]:

$$\dot{q}_i^{(\mathrm{f})}(t) = -\frac{1}{\tau_i^{(\mathrm{f})}}\cdot q_i^{(\mathrm{f})}(t) + \frac{1}{\tau_i^{(\mathrm{f})}}\cdot q_i(t), \quad i\in\{1,2\}, \tag{12}$$

$$q_1(t) = \frac{1}{R_{\mathrm{heat}}}\cdot\left[2\cdot V_1^{(1)}(t)^2 + V_1^{(2)}(t)^2\right], \tag{13}$$

$$q_2(t) = \frac{1}{R_{\text{heat}}} \cdot \left[V_2^{(1)}(t)^2 + V_2^{(2)}(t)^2 \right]. \tag{14}$$

We consider the heat produced by each set of heat resistors separately. Specifically, let $q_i^{(r)}(t) \in \mathbb{R}$ be the heat produced by the heat resistor pairs in zone $i \in \{1,2\}$ driven by relay $r \in \{1,2\}$ ($r = 1$ for the SSRs, $r = 2$ for the EMRs). From (13) and (14) we have:

$$q_i^{(r)}(t) = \frac{V_i^{(r)}(t)^2}{R_{\text{heat}}}, \quad \forall i \in \{1,2\}, r \in \{1,2\}. \tag{15}$$

Then, we define the filtered heat flow rates associated with the heat resistors pairs (see (12)):

$$\dot{q}_i^{(r,\text{f})}(t) = -\frac{1}{\tau_i^{(\text{f})}} \cdot q_i^{(r,\text{f})}(t) + \frac{1}{\tau_i^{(\text{f})}} \cdot q_i^{(r)}(t), \quad i \in \{1,2\}, r \in \{1,2\}. \tag{16}$$

By noting that $\boldsymbol{q}^{(\text{f})}(t) = \begin{bmatrix} q_1^{(1,\text{f})}(t) & q_1^{(2,\text{f})}(t) & q_2^{(1,\text{f})}(t) & q_2^{(2,\text{f})}(t) \end{bmatrix}^\top \in \mathbb{R}^4_{\geq 0}$, and combining (15) with (16), we can define the matrices A_{ff} and B_f in (7) as follows:

$$A_{ff} = \begin{bmatrix} -\frac{1}{\tau_1^{(\text{f})}} & 0 & 0 & 0 \\ 0 & -\frac{1}{\tau_1^{(\text{f})}} & 0 & 0 \\ 0 & 0 & -\frac{1}{\tau_2^{(\text{f})}} & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau_2^{(\text{f})}} \end{bmatrix} \tag{17}$$

$$B_f = \frac{1}{R_{\text{heat}}} \cdot \begin{bmatrix} \frac{1}{\tau_1^{(\text{f})}} & 0 & 0 & 0 \\ 0 & \frac{1}{\tau_1^{(\text{f})}} & 0 & 0 \\ 0 & 0 & \frac{1}{\tau_2^{(\text{f})}} & 0 \\ 0 & 0 & 0 & \frac{1}{\tau_2^{(\text{f})}} \end{bmatrix} \tag{18}$$

Finally, note that, from (13), the filtered heat flow rate for the first zone is given by:

$$q_1^{(\text{f})}(t) = 2 \cdot q_1^{(1,\text{f})}(t) + q_1^{(2,\text{f})}(t), \tag{19}$$

while, due to (14), in the second zone we have:

$$q_2^{(\text{f})}(t) = q_2^{(1,\text{f})}(t) + q_2^{(2,\text{f})}(t). \tag{20}$$

Consequently, the matrix A_{Tf} in (7) amounts to:

$$\begin{aligned} A_{Tf} &= B_q \cdot \begin{bmatrix} 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \\ &= \frac{1}{6 \cdot C_z} \cdot \begin{bmatrix} 2 \cdot \mathbf{1}_6 & \mathbf{1}_6 & \mathbf{0}_6 & \mathbf{0}_6 \\ \mathbf{0}_6 & \mathbf{0}_6 & \mathbf{1}_6 & \mathbf{1}_6 \end{bmatrix}. \end{aligned} \tag{21}$$

1.3 Review of the `Packs-Temperatures` model

The `Packs-Temperatures` model describes the relationship between the flow of bottle packs inserted into the oven and the temperatures measured by the twelve thermocouples installed inside the shrink tunnel. In [11], we model $d(t; v(t))$ using a Bernoulli distribution [12] with a success rate that depends on the production rate:

$$d(t; v(t)) \sim \text{Bernoulli}\left(p_d(v(t))\right). \tag{22}$$

Instead, the relationship between $d(t; v(t))$ and the temperature drop $y_{d_i}^{(l,s)}(t), i \in \{1,2\}, l \in \{1,2,3\}, s \in \{\text{l},\text{r}\}$, is:

$$\delta\dot{T}_i^{(l,s)}(t) = -\frac{1}{\tau_{d_i}^{(l,s)}(v(t))} \cdot \delta T_i^{(l,s)}(t) + \frac{\mu_{d_i}^{(l,s)}(v(t))}{\tau_{d_i}^{(l,s)}(v(t))} \cdot d(t; v(t)), \tag{23a}$$

$$y_{d_i}^{(l,s)}(t) = \delta T_i^{(l,s)}\left(t + L_{d_i}^{(l,s)}(v(t))\right), \tag{23b}$$

where $\mu_{d_i}^{(l,s)}(v(t)), \tau_{d_i}^{(l,s)}(v(t))$, and $L_{d_i}^{(l,s)}(v(t))$ are the production-rate-dependent gains, time constants, and time lags for each $y_{d_i}^{(l,s)}(t)$. We can write (23) more compactly as

$$\begin{cases} \delta\dot{\boldsymbol{T}}(t) = A_{dd}(v(t)) \cdot \delta\boldsymbol{T}(t) + \boldsymbol{b}_d(v(t)) \cdot d(t; v(t)) \\ y_{d_i}^{(l,s)}(t) = \delta T_i^{(l,s)}\left(t + L_{d_i}^{(l,s)}(v(t))\right) \quad \text{for } i \in \{1,2\}, l \in \{1,2,3\}, s \in \{\mathrm{l},\mathrm{r}\}, \end{cases} \tag{24}$$

where $A_{dd}(v(t)) \in \mathbb{R}^{12\times 12}$ and $\boldsymbol{b}_d(v(t)) \in \mathbb{R}^{12}$ are defined as:

$$A_{dd}(v(t)) = \begin{bmatrix} -\frac{1}{\tau_{d_1}^{(3,\mathrm{r})}(v(t))} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -\frac{1}{\tau_{d_1}^{(3,\mathrm{l})}(v(t))} & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\tau_{d_1}^{(2,\mathrm{r})}(v(t))} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\frac{1}{\tau_{d_2}^{(2,\mathrm{l})}(v(t))} & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -\frac{1}{\tau_{d_2}^{(3,\mathrm{r})}(v(t))} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & -\frac{1}{\tau_{d_2}^{(3,\mathrm{l})}(v(t))} \end{bmatrix}, \tag{25}$$

$$\boldsymbol{b}_d(v(t)) = \begin{bmatrix} \frac{\mu_{d_1}^{(3,\mathrm{r})}(v(t))}{\tau_{d_1}^{(3,\mathrm{r})}(v(t))} & \frac{\mu_{d_1}^{(3,\mathrm{l})}(v(t))}{\tau_{d_1}^{(3,\mathrm{l})}(v(t))} & \frac{\mu_{d_1}^{(2,\mathrm{r})}(v(t))}{\tau_{d_1}^{(2,\mathrm{r})}(v(t))} & \cdots & \frac{\mu_{d_2}^{(2,\mathrm{l})}(v(t))}{\tau_{d_2}^{(2,\mathrm{l})}(v(t))} & \frac{\mu_{d_2}^{(3,\mathrm{r})}(v(t))}{\tau_{d_2}^{(3,\mathrm{r})}(v(t))} & \frac{\mu_{d_2}^{(3,\mathrm{l})}(v(t))}{\tau_{d_2}^{(3,\mathrm{l})}(v(t))} \end{bmatrix}^\top. \tag{26}$$

Let us assume that the production rate stays constant at a value $v(t) = \bar{v}, \forall t \in \mathbb{R}_{\geq 0}, \bar{v} \in \{30, 60, 90\}$. Moreover, notice that in (24) the temperature drop $\delta T_i^{(l,s)}(t), i \in \{1,2\}, l \in \{1,2,3\}, s \in \{\mathrm{l},\mathrm{r}\}$, does not affect the temperature drop $\delta T_{i'}^{(l',s')}(t)$ with $i' \in \{1,2\}, l' \in \{1,2,3\}, s' \in \{\mathrm{l},\mathrm{r}\}$, and at least one among the indexes (i', l', s') differing from (i, l, s) (in fact, $A_{dd}(v(t))$ is a diagonal matrix). Consequently, we can consider the effect of $d(t; v(t)) = d(t; \bar{v})$ on each $y_{d_i}^{(l,s)}(t)$ separately. Then, we can apply the Laplace transform [8] to (23) to obtain the transfer function $H_i^{(l,s)}(\varsigma; \bar{v})$ linking $D(\varsigma; \bar{v}) = \mathcal{L}(d(t; \bar{v}))$ with $Y_i^{(l,s)}(\varsigma) = \mathcal{L}\left(y_{d_i}^{(l,s)}(t)\right)$:[1]

$$\begin{aligned} H_i^{(l,s)}(\varsigma; \bar{v}) &= \frac{Y_i^{(l,s)}(\varsigma)}{D(\varsigma; \bar{v})}, \quad \forall i \in \{1,2\}, l \in \{1,2,3\}, s \in \{\mathrm{l},\mathrm{r}\} \\ &= \frac{\mu_{d_i}^{(l,s)}(\bar{v})}{1 + \varsigma \cdot \tau_{d_i}^{(l,s)}(\bar{v})} \cdot \exp\left\{-\varsigma \cdot L_{d_i}^{(l,s)}(\bar{v})\right\}. \end{aligned} \tag{27}$$

Notice that (27) is a First Order Lag Plus time Delay (FOLPD) transfer function [7].

[1] We use $\varsigma \in \mathbb{C}$ as the complex variable for the Laplace transform to avoid confusion with $s \in \{\mathrm{l},\mathrm{r}\}$.

2 Parameter estimation

In this Section, we estimate the parameters of the models reviewed in Section 1 based on data obtained from an experimental campaign carried out on the shrink tunnel under study.

2.1 Estimation of the parameters of the `Heat-Temperatures` model

First, we consider the `Heat-Temperatures` model reviewed in Section 1.2. To estimate its parameters, we rely on data coming from both open-loop and closed-loop experiments performed on the shrink tunnel under study. In both cases, relevant data has been acquired with a sampling time[2] of $\mathcal{T}_\mathrm{s} = 0.01\,\mathrm{s}$. For what concerns the open-loop experiments, we rely on two step response tests: in the former test, only the heat resistors of the first zone are actuated, while the opposite is true for the second step response. Instead, in the closed-loop experiment, the temperature controller is composed of two Proportional-Integral (PI) regulators, a decoupler, and an anti-windup strategy [1]. Only two temperatures are actually controlled in closed-loop, namely $y_1^{(1,\mathrm{r})}(t)$ and $y_2^{(2,\mathrm{l})}(t)$, which are in the central region of the oven (see Figure 1). In particular, during the closed-loop trial, the shrink tunnel switches between different operating points. In all the experiments considered in this Section, the production rate is $v(t) = 0, \forall t \in \mathbb{R}_{\geq 0}$, i.e. there is no flow of bottle packs being inserted into the oven. Consequently, we have (see (1)):

$$\boldsymbol{y}_d(t) = \mathbf{0}_{12}, \forall t \in \mathbb{R}_{\geq 0} \implies \boldsymbol{y}(t) = \boldsymbol{y}_T(t). \tag{28}$$

We are only interested in identifying the parameters that govern the linear part of the system in (7). That is because the nonlinear part of the system, i.e.

$$\boldsymbol{V}_\mathrm{sq}(t) = \boldsymbol{f}(\boldsymbol{u}(t), V_\mathrm{g}(t)),$$

does not depend on any unknown parameters. Therefore, we only have to estimate the thermal resistances $R_i^{(l,s)}, R_{\mathrm{T}_i}^{(l',s)}, R_{\mathrm{T}_{12}}^{(s)}, R_{\mathrm{T_a}}, i \in \{1,2\}, l \in \{1,2,3\}, l' \in \{1,2\}, s \in \{\mathrm{l},\mathrm{r}\}$, the resistance of the heat resistors R_heat, the time constants $\tau_i^{(\mathrm{f})}$, and the time delays $L_i^{(l,s)}$ (see Section 1.2).

First, we estimate the time delays from the open-loop step responses in a heuristic fashion by means of a threshold [6]. Notation-wise, we denote the estimated time delays as $\hat{L}_{T_i}^{(l,s)}, i \in \{1,2\}, l \in \{1,2,3\}, s \in \{\mathrm{l},\mathrm{r}\}$. Then, we shift the output signals $y_i^{(l,s)}(t)$ of the available closed-loop experiment backwards by their corresponding estimated time delays $\hat{L}_{T_i}^{(l,s)}$, obtaining (see (1) and (28))

$$\begin{aligned}\tilde{y}_i^{(l,s)}(t) &= y_i^{(l,s)}\left(t - \hat{L}_{T_i}^{(l,s)}\right) \\ &\approx T_i^{(l,s)}(t),\end{aligned} \tag{29}$$

assuming $\hat{L}_{T_i}^{(l,s)} \approx L_{T_i}^{(l,s)}$. Next, we group all the $\tilde{y}_i^{(l,s)}(t)$'s inside the vector

$$\tilde{\boldsymbol{y}}(t) = \begin{bmatrix}\tilde{y}_1^{(3,\mathrm{r})}(t) & \tilde{y}_1^{(3,\mathrm{l})}(t) & \tilde{y}_1^{(2,\mathrm{r})}(t) & \dots & \tilde{y}_2^{(2,\mathrm{l})}(t) & \tilde{y}_2^{(3,\mathrm{r})}(t) & \tilde{y}_2^{(3,\mathrm{l})}(t)\end{bmatrix}^\top \in \mathbb{R}^{12}.$$

By proceeding as in (29), and recalling that there are no unknown parameters that govern the nonlinear part of the system, we only have to identify the following delay-free linear state-space model (cf. (7)):

$$\begin{cases}\dot{\boldsymbol{x}}_T(t) = A(\boldsymbol{\theta}) \cdot \boldsymbol{x}_T(t) + B(\boldsymbol{\theta}) \cdot \begin{bmatrix}\boldsymbol{V}_\mathrm{sq}(t) \\ T_\mathrm{a}(t)\end{bmatrix} \\ \tilde{\boldsymbol{y}}(t) = C \cdot \boldsymbol{x}_T(t)\end{cases}, \tag{30}$$

where:

$$\boldsymbol{x}_T(t) = \begin{bmatrix}\boldsymbol{q}^{(\mathrm{f})}(t) \\ \boldsymbol{T}(t)\end{bmatrix} \in \mathbb{R}^{16}, \tag{31a}$$

$$A(\boldsymbol{\theta}) = \begin{bmatrix}A_{ff} & 0_{4\times 12} \\ A_{Tf} & A_{TT}\end{bmatrix} \in \mathbb{R}^{16\times 16}, \tag{31b}$$

$$B(\boldsymbol{\theta}) = \begin{bmatrix}B_f & \mathbf{0}_4 \\ 0_{12\times 4} & \boldsymbol{b}_T\end{bmatrix} \in \mathbb{R}^{16\times 5}, \tag{31c}$$

[2] In the identification experiments, we have sampled relevant signals at a sampling time $\mathcal{T}_\mathrm{s}$ that is much lower than the sampling time $\mathcal{T}_\mathrm{c}$ for the controller, which is either $\mathcal{T}_\mathrm{c} = 1\,\mathrm{s}$ or $\mathcal{T}_\mathrm{c} = m \cdot 30\,\mathrm{s}, m \in \mathbb{N}$ (see [11]), to ensure proper estimation of the parameters for the continuous-time model in (1).

$$C = [0_{12\times 4} \quad I_{12}] \in \mathbb{R}^{12\times 16}. \tag{31d}$$

In (30) and (31), we have made explicit the dependency of the system's matrices on the remaining unknown parameters $\boldsymbol{\theta} \in \mathbb{R}^{n_{\boldsymbol{\theta}}}$, which are $R_i^{(l,s)}, R_{\mathrm{T}_i}^{(l',s)}, R_{\mathrm{T}_{12}}^{(s)}, R_{\mathrm{T_{rl}}}, i \in \{1,2\}, l \in \{1,2,3\}, l' \in \{1,2\}, s \in \{\mathrm{l},\mathrm{r}\}$, R_{heat}, and $\tau_i^{(\mathrm{f})}$. In particular, $A(\boldsymbol{\theta})$ and $B(\boldsymbol{\theta})$ depend on $n_{\boldsymbol{\theta}} = 26$ parameters in total. To estimate them, we follow the output-error estimation methodology [13], i.e. we minimize the cost function:

$$J(\boldsymbol{\theta}) = \frac{1}{N} \cdot \sum_{k=0}^{N-1} \left\| \tilde{\boldsymbol{y}}(k \cdot \mathcal{T}_{\mathrm{s}}) - \hat{\tilde{\boldsymbol{y}}}(k \cdot \mathcal{T}_{\mathrm{s}}; \boldsymbol{\theta}) \right\|_2^2. \tag{32}$$

In (32), $N \in \mathbb{N}$ is the number of samples acquired during the closed-loop experiment, $k \in \mathbb{N} \cup \{0\}$ is the sample index, and $\hat{\tilde{\boldsymbol{y}}}(k \cdot \mathcal{T}_{\mathrm{s}}; \boldsymbol{\theta})$ is the simulated output of the system in (30) for a given parametrization $\boldsymbol{\theta}$ and using the same input sequence $\left[\boldsymbol{V}_{\mathrm{sq}}(t)^\top \quad T_{\mathrm{a}}(t)\right]^\top$ that produced $\tilde{\boldsymbol{y}}(t)$. In particular, we use MATLAB's `greyest` method [4] to minimize the cost function in (32). Notation-wise, we denote the estimate of the parameters as $\hat{\boldsymbol{\theta}}$, i.e.

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}_{>0}} J(\boldsymbol{\theta}).$$

Table 2 reports the identified parameters for the `Heat-Temperatures` model. Then, Table 3 reports the goodness of fit of each simulated output[3] $\hat{y}_i^{(l,s)}\left(k \cdot \mathcal{T}_s; \hat{\boldsymbol{\theta}}\right), i \in \{1,2\}, l \in \{1,2,3\}, s \in \{\mathrm{l},\mathrm{r}\}$, computed as (1 − NRMSE):

$$\mathrm{Fit}_i^{(l,s)} = \left(1 - \frac{\sqrt{\frac{1}{N} \cdot \sum_{k=0}^{N-1} \left| y_i^{(l,s)}(k \cdot \mathcal{T}_{\mathrm{s}}) - \hat{y}_i^{(l,s)}\left(k \cdot \mathcal{T}_{\mathrm{s}}; \hat{\boldsymbol{\theta}}\right)\right|^2}}{\max_{k\in\{0,\ldots,N-1\}} y_i^{(l,s)}(k \cdot \mathcal{T}_{\mathrm{s}}) - \min_{k\in\{0,\ldots,N-1\}} y_i^{(l,s)}(k \cdot \mathcal{T}_{\mathrm{s}})}\right) \cdot 100 \quad [\%]. \tag{33}$$

Finally, Figure 4 compares the simulated outputs with the real data coming from the closed-loop experiment used for the identification of the parameters.

2.2 Estimation of the parameters of the `Packs-Temperatures` model

Now, we consider the `Packs-Temperatures` model reviewed in Section 1.3. In this case, we need to estimate the success rate of the Bernoulli distribution in (22) and the parameters of the model in (24), which all depend on the production rate. For identification purposes, we have carried out three experiments on the shrink tunnel under study in the following fashion. The oven is controlled in open-loop, with constant duty cycles $u_1(t), u_2(t)$. Initially, we let the temperatures inside the oven cavity reach an equilibrium point, denoted as $\bar{\boldsymbol{y}}_T \in \mathbb{R}^{12}$, without inserting any bottle packs into the shrink tunnel. Then, we started the production line at a constant production rate $\bar{v}$ of 30 ppm and waited for the temperatures to (roughly) settle. We repeated the same procedure for $\bar{v} = 60$ ppm and $\bar{v} = 90$ ppm. By proceeding this way, we can easily derive the effect of the bottle packs on the temperatures by subtracting the previously reached equilibrium temperatures due to the constant control actions (and ambient temperature), i.e. (see (1))

$$\boldsymbol{y}_d(t) = \boldsymbol{y}(t) - \bar{\boldsymbol{y}}_T,$$

for $t > t_{\mathrm{eq}}, t_{\mathrm{eq}}$ being the time required to reach the equilibrium point $\bar{\boldsymbol{y}}_T$. Given that the production rate does not change during the just mentioned experiments (i.e. $v(t) = \bar{v}, \forall t \in \mathbb{R}_{\geq 0}$), we can estimate the parameters of interest for each $\bar{v} \in \{30, 60, 90\}$ as follows. The success rates $p_d(\bar{v})$ in (22) are found by taking the sample mean of the signals $d(t; \bar{v})$, leading to:

$$\begin{aligned} \hat{p}_d(30) &= 0.26, \\ \hat{p}_d(60) &= 0.48, \\ \hat{p}_d(90) &= 0.74. \end{aligned}$$

Instead, the parameters $\mu_i^{(l,s)}(\bar{v}), \tau_i^{(l,s)}(\bar{v}), L_i^{(l,s)}(\bar{v}), i \in \{1,2\}, l \in \{1,2,3\}, s \in \{\mathrm{l},\mathrm{r}\}$, of the FOLPD models $H_i^{(l,s)}(\varsigma; \bar{v})$ in (27) are estimated from the signals $d(t; \bar{v})$ and $y_{d_i}^{(l,s)}(t)$ by means of MATLAB's `procest` method [5]. Table 4 reports their estimates and the corresponding goodness of fit, which is computed analogously to (33). Instead, Figure 5 compares the real output $y_{d_1}^{(3,\mathrm{r})}(t)$ of the `Packs-Temperatures` model with the simulated ones for all possible production rates.

[3] Differently from $\hat{\hat{y}}_i^{(l,s)}\left(k \cdot \mathcal{T}_{\mathrm{s}}; \hat{\boldsymbol{\theta}}\right)$, $\hat{y}_i^{(l,s)}\left(k \cdot \mathcal{T}_{\mathrm{s}}; \hat{\boldsymbol{\theta}}\right)$ also takes into account of the nonlinearities in (7) and the previously identified time delays, i.e. it simulates the system using (7) instead of (30) with the estimated parameters $\hat{\boldsymbol{\theta}}$.

Table 2: Estimated parameters for the `Heat-Temperatures` model reviewed in Section 1.2. C_z is known, see (8).

$\hat{R}_1^{(3,\mathrm{r})}$	$\hat{R}_1^{(3,\mathrm{l})}$	$\hat{R}_1^{(2,r)}$	$\hat{R}_1^{(2,\mathrm{l})}$	$\hat{R}_1^{(1,r)}$	$\hat{R}_1^{(1,\mathrm{l})}$	$\hat{R}_{\mathrm{T}_1}^{(2,\mathrm{r})}$	$\hat{R}_{\mathrm{T}_1}^{(2,\mathrm{l})}$	$\hat{R}_{\mathrm{T}_1}^{(1,\mathrm{r})}$	$\hat{R}_{\mathrm{T}_1}^{(1,\mathrm{l})}$
$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$
3.28	2.01	3.99	19.45	5.92	9.20	10.00	0.12	0.65	0.86

$\hat{R}_2^{(3,\mathrm{r})}$	$\hat{R}_2^{(3,\mathrm{l})}$	$\hat{R}_2^{(2,\mathrm{r})}$	$\hat{R}_2^{(2,\mathrm{l})}$	$\hat{R}_2^{(1,\mathrm{r})}$	$\hat{R}_2^{(1,\mathrm{l})}$	$\hat{R}_{\mathrm{T}_2}^{(2,\mathrm{r})}$	$\hat{R}_{\mathrm{T}_2}^{(2,\mathrm{l})}$	$\hat{R}_{\mathrm{T}_2}^{(1,\mathrm{r})}$	$\hat{R}_{\mathrm{T}_2}^{(1,\mathrm{l})}$
$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$
2.17	2.45	84.89	100.00	2.65	2.52	0.88	1.08	0.37	1.73

$\hat{R}_{\mathrm{T}_{12}}^{(\mathrm{r})}$	$\hat{R}_{\mathrm{T}_{12}}^{(\mathrm{l})}$	$\hat{R}_{\mathrm{T}_{\mathrm{rl}}}$	C_{z}	$\hat{R}_{\mathrm{heat}}$	$\hat{\tau}_1^{(\mathrm{f})}$	$\hat{\tau}_2^{(\mathrm{f})}$
$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{°\mathrm{C}}{\mathrm{J}} \cdot \mathrm{s}\right]$	$\left[\frac{\mathrm{J}}{°\mathrm{C}}\right]$	$[\Omega]$	[s]	[s]
1.85	1.04	1.00	231	1239	184.50	35.67

$\hat{L}_{T_1}^{(3,\mathrm{r})}$	$\hat{L}_{T_1}^{(3,\mathrm{l})}$	$\hat{L}_{T_1}^{(2,\mathrm{r})}$	$\hat{L}_{T_1}^{(2,\mathrm{l})}$	$\hat{L}_{T_1}^{(1,\mathrm{r})}$	$\hat{L}_{T_1}^{(1,\mathrm{l})}$	$\hat{L}_{T_2}^{(1,\mathrm{r})}$	$\hat{L}_{T_2}^{(1,\mathrm{l})}$	$\hat{L}_{T_2}^{(2,\mathrm{r})}$	$\hat{L}_{T_2}^{(2,\mathrm{l})}$	$\hat{L}_{T_2}^{(3,\mathrm{r})}$	$\hat{L}_{T_2}^{(3,\mathrm{l})}$
[s]	[s]	[s]	[s]	[s]	[s]	[s]	[s]	[s]	[s]	[s]	[s]
66.17	75.18	62.90	67.37	58.27	60.78	50.38	53.25	36.77	42.95	66.73	63.34

Table 3: Goodness of fit of each simulated output of the `Heat-Temperatures` model.

$\mathrm{Fit}_i^{(l,s)}$ [%]

$y_1^{(3,\mathrm{r})}$	$y_1^{(3,\mathrm{l})}$	$y_1^{(2,\mathrm{r})}$	$y_1^{(2,\mathrm{l})}$	$y_1^{(1,\mathrm{r})}$	$y_1^{(1,\mathrm{l})}$	$y_2^{(1,\mathrm{r})}$	$y_2^{(1,\mathrm{l})}$	$y_2^{(2,\mathrm{r})}$	$y_2^{(2,\mathrm{l})}$	$y_2^{(3,\mathrm{r})}$	$y_2^{(3,\mathrm{r})}$
97.8%	97.5%	97.5%	97.2%	97.5%	95.9%	97.5%	97.1%	97.0%	96.4%	97.6%	95.8%

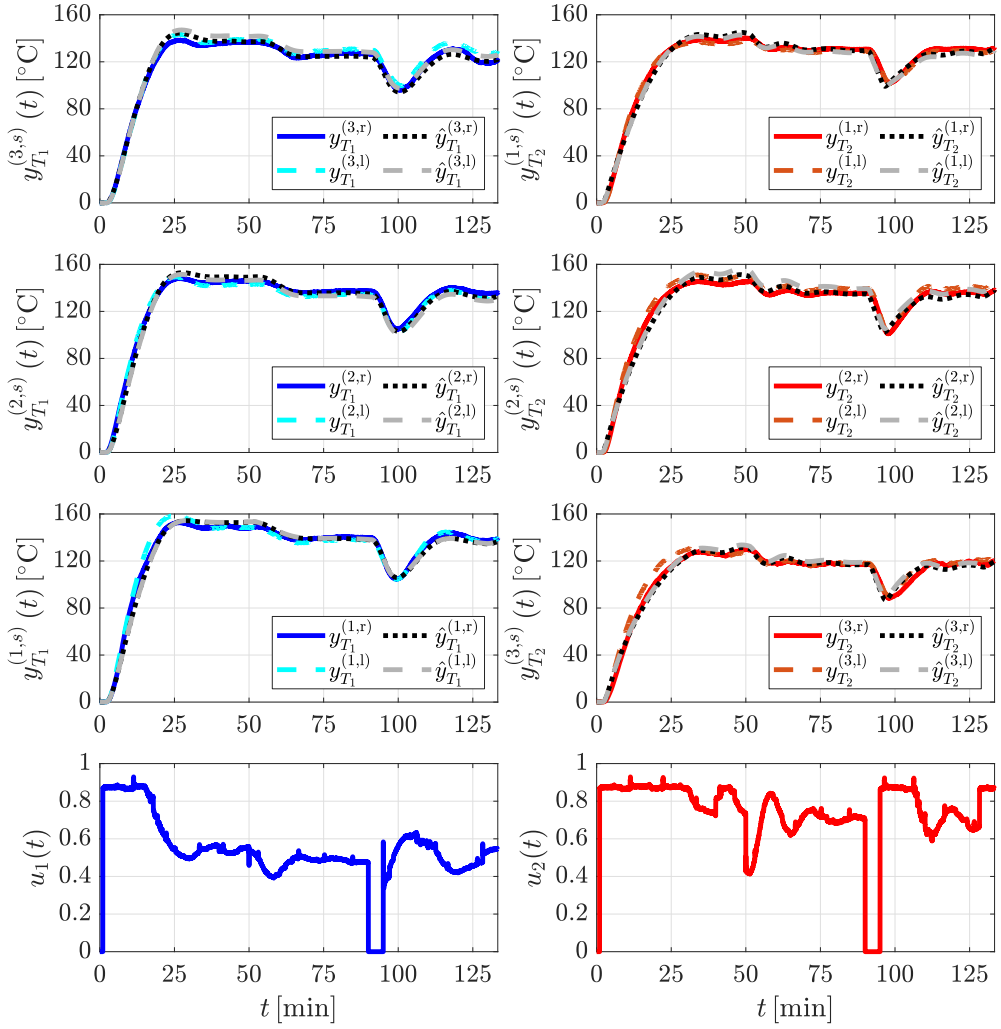


Figure 4: Comparison of the real and the simulated outputs for the `Heat-Temperatures` model reviewed in Section 1.2. The temperatures on the right side of the oven are represented in blue and red for the first and the second zone respectively; their estimates are depicted with dotted black lines. Instead, the temperatures on the left side of the oven are represented in cyan (zone 1) and orange (zone 2) dashed lines, while their estimates are reported as dashed gray lines. The ambient temperature was removed beforehand.

Table 4: Estimated parameters and fits for the `Packs-Temperatures` model reviewed in Section 1.3.

	$\bar{v} = 30\,\text{ppm}$				$\bar{v} = 60\,\text{ppm}$				$\bar{v} = 90\,\text{ppm}$			
	$\hat{\mu}_{d_i}^{(l,s)}(\bar{v})$ [°C]	$\hat{\tau}_{d_i}^{(l,s)}(\bar{v})$ [s]	$\hat{L}_{d_i}^{(l,s)}(\bar{v})$ [s]	$\text{Fit}_i^{(l,s)}$ [%]	$\hat{\mu}_{d_i}^{(l,s)}(\bar{v})$ [°C]	$\hat{\tau}_{d_i}^{(l,s)}(\bar{v})$ [s]	$\hat{L}_{d_i}^{(l,s)}(\bar{v})$ [s]	$\text{Fit}_i^{(l,s)}$ [%]	$\hat{\mu}_{d_i}^{(l,s)}(\bar{v})$ [°C]	$\hat{\tau}_{d_i}^{(l,s)}(\bar{v})$ [s]	$\hat{L}_{d_i}^{(l,s)}(\bar{v})$ [s]	$\text{Fit}_i^{(l,s)}$ [%]
$y_1^{(3,\mathrm{r})}$	−61.60	216.96	6.19	97.7%	−78.33	312.33	0.24	95.1%	−46.74	178.92	28.16	97.7%
$y_1^{(3,\mathrm{l})}$	−59.31	422.56	14.82	95.8%	−61.62	395.55	5.96	96.0%	−41.28	229.47	1.48	95.8%
$y_1^{(2,\mathrm{r})}$	−36.07	373.52	0.46	98.7%	−39.68	487.53	18.11	97.0%	−30.78	490.73	0.00	98.7%
$y_1^{(2,\mathrm{l})}$	−46.04	568.67	14.94	97.6%	−41.93	478.38	17.06	96.9%	−30.54	351.33	4.15	97.7%
$y_1^{(1,\mathrm{r})}$	−13.05	569.49	34.82	97.0%	−23.13	491.36	23.08	95.1%	−26.44	623.44	56.00	97.0%
$y_1^{(1,\mathrm{l})}$	−45.15	742.20	11.28	98.2%	−37.64	544.92	8.40	97.1%	−28.64	439.11	0.00	98.2%
$y_2^{(1,\mathrm{r})}$	−17.94	587.892	2.25	97.7%	−21.68	520.13	12.03	96.1%	−22.17	601.79	30.30	97.7%
$y_2^{(1,\mathrm{l})}$	−26.06	1057.52	28.46	98.3%	−23.08	589.10	2.80	96.4%	−19.42	497.82	0.00	98.3%
$y_2^{(2,\mathrm{r})}$	−21.76	537.59	15.62	97.6%	−24.30	404.65	1.01	95.6%	−21.19	511.00	0.00	97.6%
$y_2^{(2,\mathrm{l})}$	−28.67	666.59	0.00	98.2%	−22.89	484.28	32.33	96.7%	−17.35	465.74	0.00	98.2%
$y_2^{(3,\mathrm{r})}$	−42.79	319.95	15.19	95.1%	−39.76	209.30	19.22	93.9%	−27.72	533.19	0.00	95.1%
$y_2^{(3,\mathrm{l})}$	−24.34	1053.91	26.96	95.0%	−14.40	695.26	44.27	91.7%	−12.07	670.96	40.00	95.0%

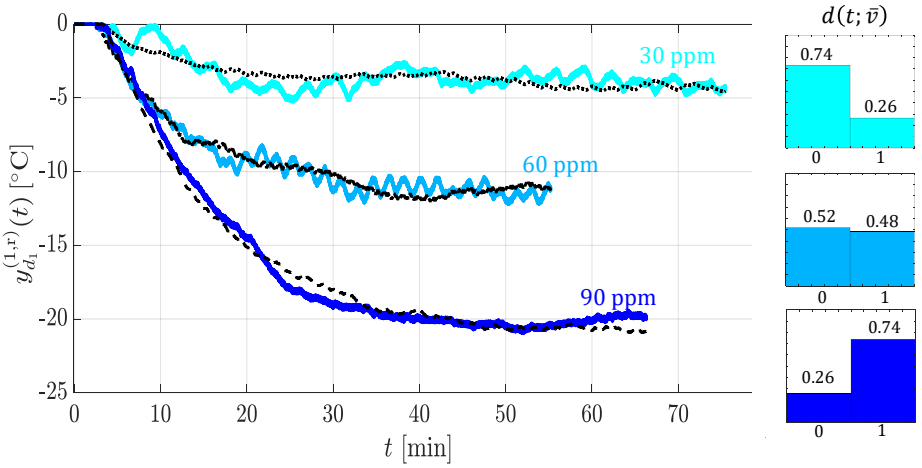


Figure 5: Comparison of the real (continuous colored lines) and the simulated (dashed black lines) output $y_{d_1}^{(3,\mathrm{r})}(t)$ of the `Packs-Temperatures` model reviewed in Section 1.3 for different production rates. The histograms represent the distributions of the signals $d(t;\bar{v})$ obtained from the performed experiments.

3 Software documentation

This Section describes the MATLAB scripts and the Simulink models that allow the user to interact with the benchmark system introduced in [11]. Specifically, the user can either simulate the system in closed-loop (see Section 3.2), i.e. as in the scheme in Figure 2, or in open-loop (see Section 3.1), i.e. without the controller, as in Figure 6. Finally, Section 3.3 reports an example of controller for the benchmark. In any case, the simulation time for Simulink is set to $\mathcal{T}_{\text{sim}} = 0.01$ s ($\mathcal{T}_{\text{sim}} < \mathcal{T}_{\text{c}}$), consistently with the sampling time $\mathcal{T}_{\text{s}}$ used during the identification experiments (see Section 2).

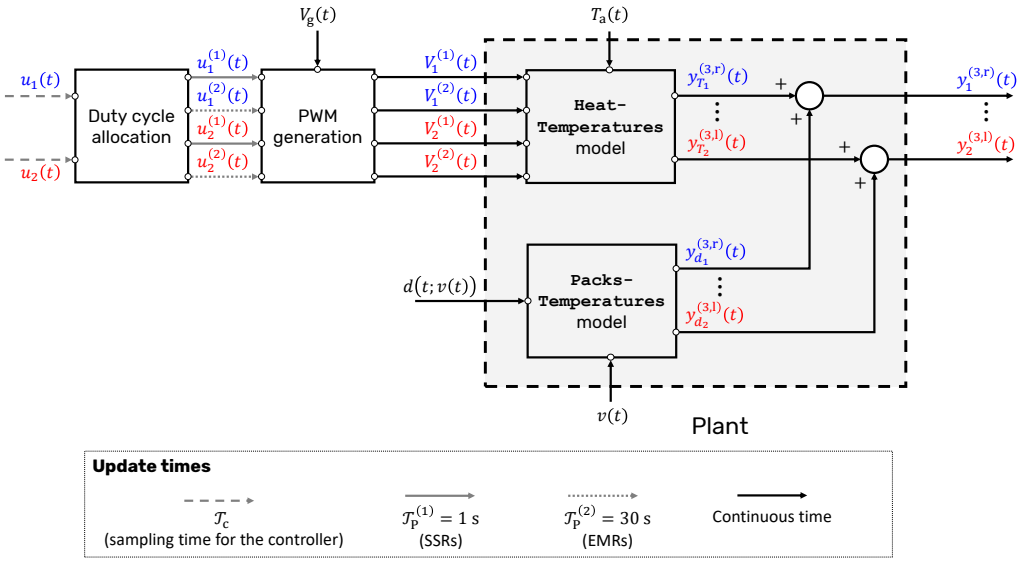


Figure 6: Block diagram of the system under study when operated in open-loop. The signals in blue are associated with the first zone while those in red are related to the second zone. Instead, the signals in black are not related to any zone in particular. The different arrows highlight how often each signal is updated.

3.1 Simulation of the shrink tunnel in open loop

`complete_model_ol.slx` is the Simulink model that implements the scheme in Figure 6 for open-loop simulation. There are several subsystems that compose it, which match exactly those represented in the figure:

1. The duty cycle allocation subsystem can either be `duty_cycle_allocation_T_P_1.slx` (`OM1`) or `duty_cycle_allocation_mT_P_2.slx` (`OM2`) depending on the operation mode (see [11]). It corresponds to the duty cycle allocation block in Figure 6;
2. The `PWM_generation.slx` subsystem corresponds to the PWM generation block in Figure 6;
3. The `plant.slx` subsystem matches the plant block in Figure 6.

Further details are given in Section 3.1.1. The MATLAB script `simulate_OL.m` can be used to run the `complete_model_ol.slx` Simulink model in a straightforward fashion, as described in Section 3.1.2.

3.1.1 The `complete_model_ol.slx` Simulink model

The `complete_model_ol.slx` Simulink model is depicted in Figure 7. Notice its similarity to the block diagram in Figure 6. Via the script `simulate_OL.m` (Section 3.1.2), the user sets the sampling time for the controller $\mathcal{T}_{\text{c}}$ (or, rather, the update time for the duty cycles $u_1(t), u_2(t)$), which can either be:

- $\mathcal{T}_{\text{c}} = 1$ s (operation mode 1 [11]) or
- $\mathcal{T}_{\text{c}} = m \cdot 30$ s, $m \in \mathbb{N}$ (operation mode 2 [11]),

and supplies the following signals to the Simulink model:

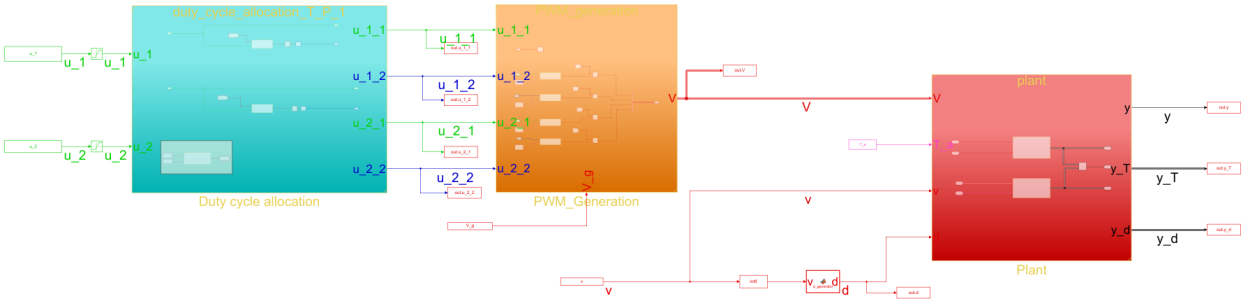


Figure 7: The `complete_model_ol.slx` Simulink model. The signals in green are updated every second (i.e. $\mathcal{T}_{\mathrm{P}}^{(1)}$), those in blue are updated every 30 s (i.e. $\mathcal{T}_{\mathrm{P}}^{(2)}$), the ones in red every $\mathcal{T}_{\mathrm{sim}}$ seconds, and finally the signals in black are "continuous-time signals".

1. The duty cycles $u_1(t), u_2(t) \in [0,1]$;
2. The grid voltage $V_{\mathrm{g}}(t) \in \mathbb{R}_{\geq 0}$;
3. The production rate $v(t) \in \{0, 30, 60, 90\}$.

Instead, the ambient temperature is already fixed at $T_{\mathrm{a}}(t) = 15°\mathrm{C}, \forall t \in \mathbb{R}_{\geq 0}$, while the signal $d(t; v(t))$ is generated as in (22). The outputs of the simulation are the temperatures $\boldsymbol{y}(t)$, $\boldsymbol{y}_T(t)$, and $\boldsymbol{y}_d(t)$. Next, we review each subsystem that composes the `complete_model_ol.slx` Simulink model more in detail.

Firstly, there is the duty cycle allocation subsystem, which is set automatically based on $\mathcal{T}_{\mathrm{c}}$ and derives the duty cycles for each heat resistors pairs, namely

$$u_1^{(1)}(t), u_1^{(2)}(t), u_2^{(1)}(t), u_2^{(2)}(t),$$

based on

$$u_1(t), u_2(t).$$

When $\mathcal{T}_{\mathrm{c}} = 1\,\mathrm{s}$, we employ the `duty_cycle_allocation_T_P_1.slx` Simulink model (see Figure 8a), which basically implements the following mathematical expressions $\forall i \in \{1,2\}$ [11]:

$$u_i^{(1)}\left(k \cdot \mathcal{T}_{\mathrm{P}}^{(1)}\right) = u_i\left(k' \cdot \mathcal{T}_{\mathrm{c}}\right), \quad \forall k \in \mathbb{N} \cup \{0\}, k' = k, \tag{34a}$$

$$u_i^{(2)}\left(k \cdot \mathcal{T}_{\mathrm{P}}^{(2)}\right) = \begin{cases} \frac{1}{\gamma} \cdot \sum_{k'=0}^{\gamma-1} u_i\left((k-1) \cdot \mathcal{T}_{\mathrm{P}}^{(2)} + k' \cdot \mathcal{T}_{\mathrm{c}}\right) & \forall k \in \mathbb{N} \\ 0 & \text{for } k = 0 \end{cases}, \tag{34b}$$

where $\gamma = \frac{\mathcal{T}_{\mathrm{P}}^{(2)}}{\mathcal{T}_{\mathrm{P}}^{(1)}} = 30$. Instead, when $\mathcal{T}_{\mathrm{c}} = m \cdot 30\,\mathrm{s}, m \in \mathbb{N}$, the `duty_cycle_allocation_mT_P_2.slx` Simulink model is used (see Figure 8b), which computes $\forall i \in \{1,2\}$:

$$u_i^{(1)}\left(k \cdot \mathcal{T}_{\mathrm{P}}^{(1)}\right) = u_i\left(k' \cdot \mathcal{T}_{\mathrm{c}}\right), \quad \forall k : \gamma \cdot m \cdot k' \leq k < \gamma \cdot m \cdot (k'+1), \tag{35a}$$

$$u_i^{(2)}\left(k \cdot \mathcal{T}_{\mathrm{P}}^{(2)}\right) = u_i\left(k' \cdot \mathcal{T}_{\mathrm{c}}\right), \quad \forall k : m \cdot k' \leq k < m \cdot (k'+1). \tag{35b}$$

Next, there is the `PWM_generation.slx` subsystem (Figure 9) which, given the duty cycles

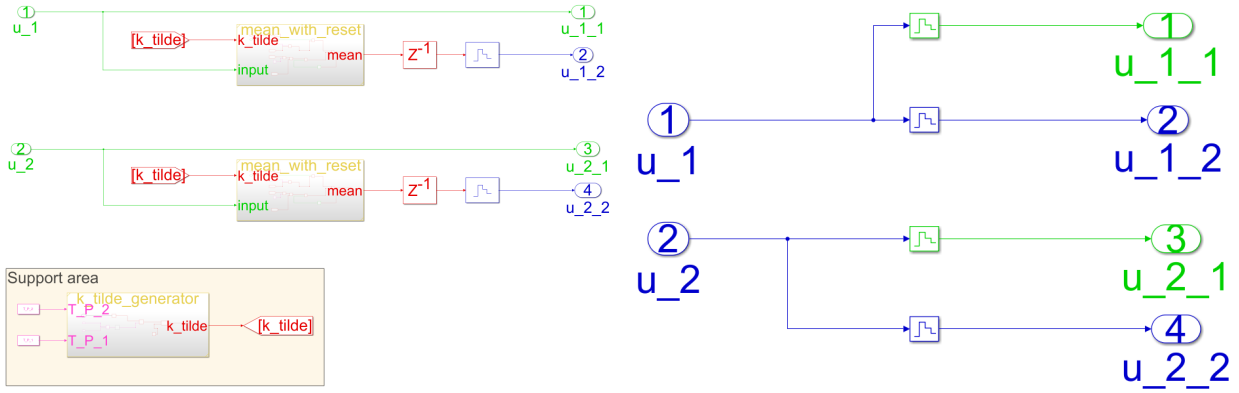
$$u_1^{(1)}(t), u_1^{(2)}(t), u_2^{(1)}(t), u_2^{(2)}(t),$$

and the grid voltage $V_{\mathrm{g}}(t)$, computes the voltage drop across each heat resistor pair, i.e.

$$V_1^{(1)}(t), V_1^{(2)}(t), V_2^{(1)}(t), V_2^{(2)}(t).$$

In details, for each $u_i^{(r)}(t), i \in \{1,2\}, r \in \{1,2\}$, the `PWM.slx` Simulink subsystem in Figure 9 generates the corresponding PWM signal as in Figure 10, i.e. according to:

$$\sigma_i^{(r)}(t) = \begin{cases} 1 & \text{for } k \cdot \mathcal{T}_{\mathrm{P}}^{(r)} \leq t < \left[k + u_i^{(r)}(t)\right] \cdot \mathcal{T}_{\mathrm{P}}^{(r)} \\ 0 & \text{for } \left[k + u_i^{(r)}(t)\right] \cdot \mathcal{T}_{\mathrm{P}}^{(r)} \leq t < [k+1] \cdot \mathcal{T}_{\mathrm{P}}^{(r)} \end{cases}, \forall k \in \mathbb{N} \cup \{0\}. \tag{36}$$



(a) `duty_cycle_allocation_T_P_1.slx`. (b) `duty_cycle_allocation_mT_P_2.slx` ($m = 1$).

Figure 8: Possible Simulink models for the duty cycle allocation block in Figure 6. The signals in green are updated every second (i.e. $\mathcal{T}_{\mathrm{P}}^{(1)}$) while those in blue are updated every 30 s (i.e. $\mathcal{T}_{\mathrm{P}}^{(2)}$).

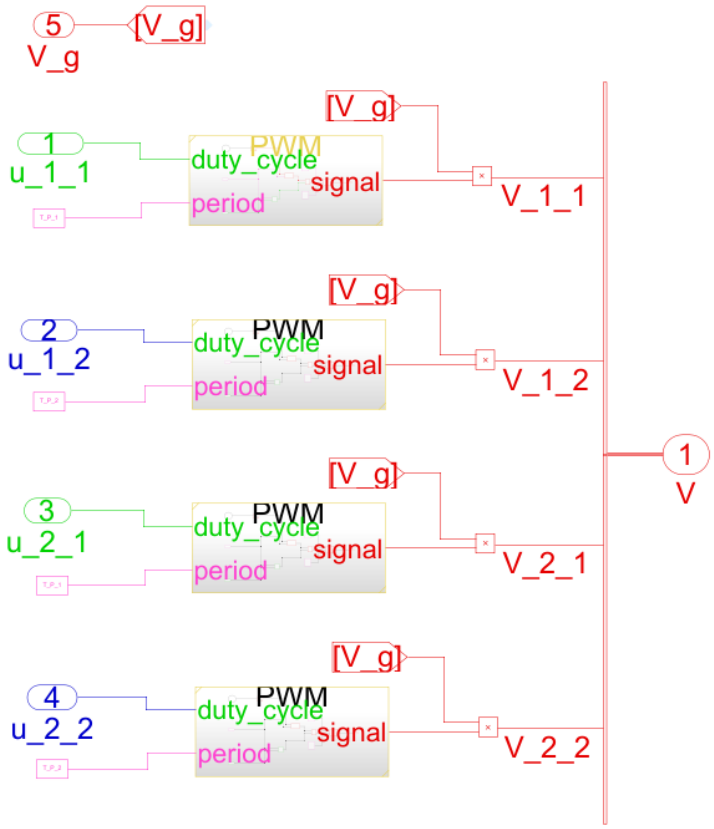


Figure 9: The `PWM_generation.slx` Simulink model. The signals in green are updated every second (i.e. $\mathcal{T}_{\mathrm{P}}^{(1)}$), those in blue are updated every 30 s (i.e. $\mathcal{T}_{\mathrm{P}}^{(2)}$), and the ones in red every $\mathcal{T}_{\mathrm{sim}}$ seconds.

Then, $\sigma_i^{(r)}(t)$ is multiplied by $V_{\mathrm{g}}(t)$ in the subsystem `PWM_generation.slx` to give rise to $V_i^{(r)}(t), i \in \{1, 2\}, r \in \{1, 2\}$. Notice that in Figure 9 the voltage drops are updated every $\mathcal{T}_{\mathrm{sim}}$ seconds.

Finally, there is the `plant.slx` subsystem, which implements the shrink tunnel model in (1) as in Figure 11. Consistently with the model derived in [11] and as highlighted by the scheme in Figure 6, we consider two submodels, i.e. the `Heat-Temperatures` and `Packs-Temperatures` models. For each submodel, we have coded its correspond-

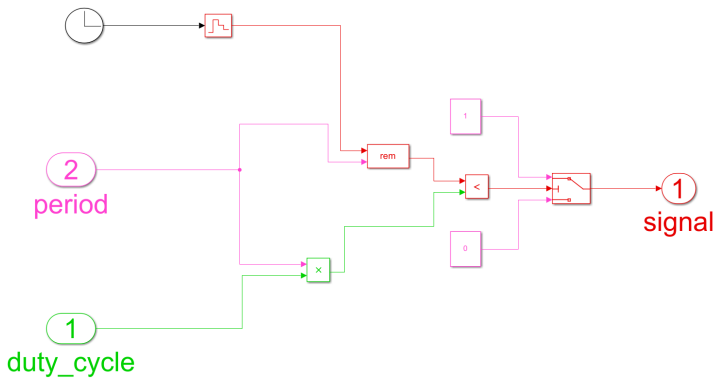


Figure 10: The `PWM.slx` Simulink model that is present in the `PWM_generation.slx` Simulink model (Figure 9).

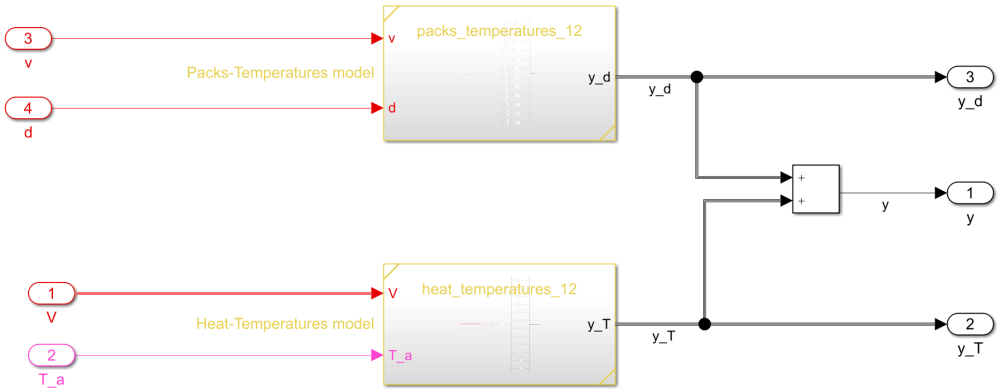


Figure 11: The `plant.slx` Simulink model. The signals in red are updated every $\mathcal{T}_{\text{sim}}$ seconds while those in black are "continuous-time signals".

ing subsystem. Specifically, the subsystem for the `Heat-Temperatures` model (`heat_temperatures_12.slx`) is reported in Figure 12. To implement (7), we first compute the square of the voltages $V_i^{(r)}(t), i \in \{1,2\}, r \in \{1,2\}$, that were derived by the `PWM_generation.slx` subsystem. Then, we group the squared voltages $\boldsymbol{V}_{\text{sq}}(t)$ with the ambient temperature $T_{\text{a}}(t)$ inside a vector and feed it to an LTI System Simulink block which basically implements the following linear state-space model (see (7)):

$$\begin{cases} \dot{\boldsymbol{q}}^{(\text{f})}(t) = A_{ff} \cdot \boldsymbol{q}^{(\text{f})}(t) + B_f \cdot \boldsymbol{V}_{\text{sq}}(t) \\ \dot{\boldsymbol{T}}(t) = A_{TT} \cdot \boldsymbol{T}(t) + A_{Tf} \cdot \boldsymbol{q}^{(\text{f})}(t) + \boldsymbol{b}_T \cdot T_{\text{a}}(t) \end{cases} \tag{37}$$

using the parameters in Table 2. The system in (37) is simulated starting from $\boldsymbol{q}^{(\text{f})}(0) = \boldsymbol{0}_4$ (shrink tunnel turned off) and $\boldsymbol{T}(0) = 15 \cdot \boldsymbol{1}_{12}$ (ambient temperature). Finally, each temperature is delayed as in:

$$y_{T_i}^{(l,s)}(t) = T_i^{(l,s)}\left(t + L_{T_i}^{(l,s)}\right) \quad \text{for } i \in \{1,2\}, l \in \{1,2,3\}, s \in \{\text{l},\text{r}\},$$

obtaining $\boldsymbol{y}_T(t)$. For what concerns the `Packs-Temperatures` model, its Simulink implementation (`packs_temperatures_12.slx`) relies on one Varying Transfer Function block for each temperature drop $y_{d_i}^{(l,s)}(t), i \in \{1,2\}, l \in \{1,2,3\}, s \in \{\text{l},\text{r}\}$, in (24). In details, the Varying Transfer Function Simulink block allows coding transfer functions with varying parameters, such as the one in (27), which we report here for the sake of clarity:

$$H_i^{(l,s)}(\varsigma;\bar{v}) = \frac{\mu_{d_i}^{(l,s)}(\bar{v})}{1 + \varsigma \cdot \tau_{d_i}^{(l,s)}(\bar{v})} \cdot \exp\left\{-\varsigma \cdot L_{d_i}^{(l,s)}(\bar{v})\right\}.$$

The Simulink implementation of the relationship between $\bar{v}, d(t;\bar{v})$, and $y_{d_1}^{(3,\text{r})}(t)$ is depicted in Figure 13. The values for the parameters $\mu_{d_i}^{(l,s)}(\bar{v})$ and $\tau_{d_i}^{(l,s)}(\bar{v})$ are set as in Table 4 using a look-up table. In practice, for ease of Simulink

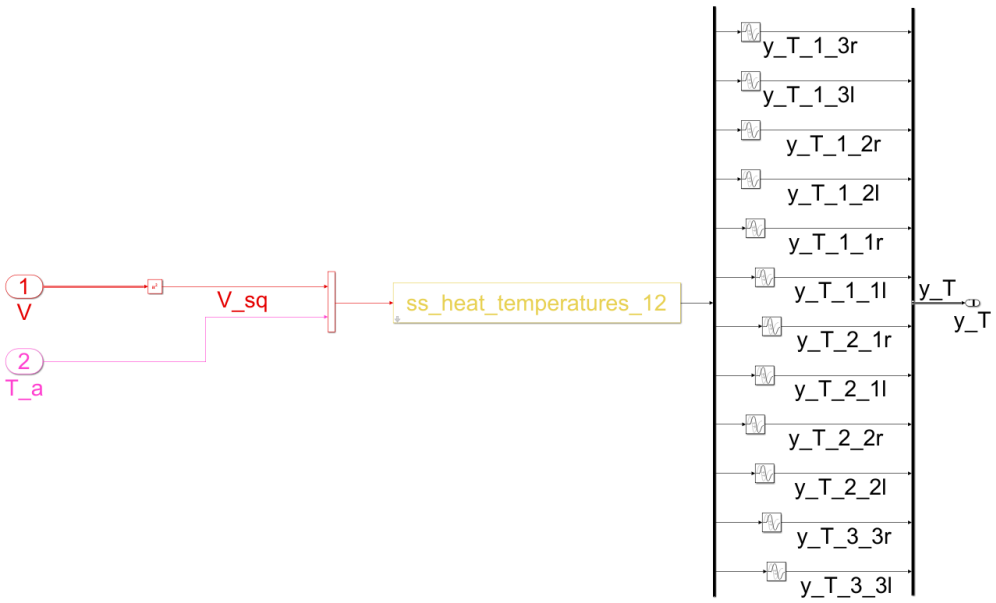


Figure 12: The `heat_temperatures_12.slx` Simulink model. The signals in red are updated every $\mathcal{T}_{\text{sim}}$ seconds while those in black are "continuous-time signals".

implementation (and mitigate computational burdens connected to varying time delays), the delays are not production-rate dependent in the `packs_temperatures_12.slx` subsystem. Instead, we have chosen to use

$$L_{d_i}^{(l,s)} = L_{d_i}^{(l,s)}(60)$$

regardless of $\bar{v}$. Consequently, the corresponding FOLPD transfer functions amount to:

$$H_i^{(l,s)}(\varsigma;\bar{v}) = \frac{\mu_{d_i}^{(l,s)}(\bar{v})}{1+\varsigma\cdot\tau_{d_i}^{(l,s)}(\bar{v})}\cdot\exp\left\{-\varsigma\cdot L_{d_i}^{(l,s)}\right\}, \quad \forall i\in\{1,2\}, l\in\{1,2,3\}, s\in\{\mathrm{l},\mathrm{r}\}.$$

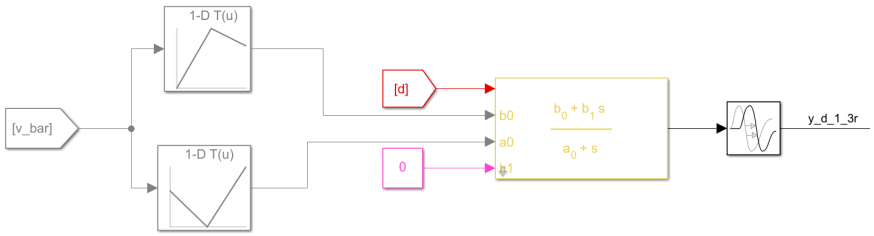


Figure 13: Snippet of the `packs_temperatures_12.slx` Simulink model for what concerns the temperature drop $y_{d_1}^{(3,\mathrm{r})}(t)$. The signals in red are updated every $\mathcal{T}_{\text{sim}}$ seconds while those in black are "continuous-time signals".

3.1.2 The `simulate_OL.m` MATLAB script

To interact and run the `complete_model_ol.slx` Simulink model, the user executes the `simulate_OL.m` MATLAB script. Here, we report the lines of code that the user can change:

Listing 1: `simulate_OL.m`, first part.

```
%% Setup
clc
clear
close all
```

```
5
6       T_c = 1; % In practice, this is used for defining the operating mode (duty cycle
            allocation)
7       simulink_type = 'ol';
8
9       set_up_parameters_for_simulink
10
11      %% Signal generation
12      % Trial length [min]
13      trial_length = 60 * 6; % 6 hours
14      times = (0 : T_sim : trial_length*60)'; % [s]
15      N = length(times);
16
17      % For simplicity, constant grid voltage and equal to the nominal voltage of
18      % the heat resistors
19      V_g = 380 * ones(N, 1);
20      V_g = timeseries(V_g, times);
21
22      % Duty cycles
23      u_1 = 0.7 * ones(N, 1);
24      u_1 = timeseries(u_1, times);
25      u_2 = 0.7 * ones(N, 1);
26      u_2 = timeseries(u_2, times);
27
28      % Production rate
29      % First 2 hours -> 0 ppm (let the oven temperatures settle)
30      % 1 hour -> 30 ppm
31      % 1 hour -> 60 ppm
32      % 1 hour -> 90 ppm
33      % 1 hour -> 0 ppm (let the oven settle back to the original temperatures)
34      v = zeros(N, 1);
35      v((2*60*60)/T_sim + 1 : 3*60*60/T_sim) = 30;
36      v((3*60*60)/T_sim + 1 : 4*60*60/T_sim) = 60;
37      v((4*60*60)/T_sim + 1 : 5*60*60/T_sim) = 90;
38      v = timeseries(v, times);
```

First of all, the user should select $\mathcal{T}_\mathrm{c}$ (line 6, Listing 1) as they see fit (either $\mathcal{T}_\mathrm{c} = 1\,\mathrm{s}$ or $\mathcal{T}_\mathrm{c} = m \cdot 30\,\mathrm{s}, m \in \mathbb{N}$). Then, they can change the length of the simulation, the grid voltage, the duty cycles, and the production rate. All the signals should be supplied as `timeseries` objects to the Simulink model `complete_model_ol.slx`. In the available example (`simulate_OL.m`), we have created an experiment of 6 hours with:

- $V_\mathrm{g}(t) = 380\,\mathrm{V}, \forall t \in \mathbb{R}_{\geq 0}$. In general, the grid voltage may fluctuate between 370 V and 420 V;
- $u_1(t) = u_2(t) = 0.7, \forall t \in \mathbb{R}_{\geq 0}$;
- $v(t) = \begin{cases} 0\,\mathrm{ppm} & \text{for } 0\,\mathrm{h} \leq t < 2\,\mathrm{h} \\ 30\,\mathrm{ppm} & \text{for } 2\,\mathrm{h} \leq t < 3\,\mathrm{h} \\ 60\,\mathrm{ppm} & \text{for } 3\,\mathrm{h} \leq t < 4\,\mathrm{h} \\ 90\,\mathrm{ppm} & \text{for } 4\,\mathrm{h} \leq t < 5\,\mathrm{h} \\ 0\,\mathrm{ppm} & \text{for } 5\,\mathrm{h} \leq t < 6\,\mathrm{h} \end{cases}$.

To run the `complete_model_ol.slx` Simulink model, the user executes the following lines of code:

Listing 2: `simulate_OL.m`, second part.

```
1       %% Simulation
2       set_param(simulink_name, 'Solver', 'ode45', 'StartTime', num2str(times(1)),
            'StopTime', num2str(times(end)))
3
4       warning('off', 'Simulink:blocks:SwitchIgnoringThreshold')
```

```
    out = sim(simulink_name, 'ReturnWorkspaceOutputs', 'on');
    warning('on', 'Simulink:blocks:SwitchIgnoringThreshold')
```

After carrying out the simulation, the user may plot the results using the subsequent lines of code in the `simulate_OL.m` MATLAB script. In particular, using the previously mentioned settings, we obtain the results reported in Figure 14.

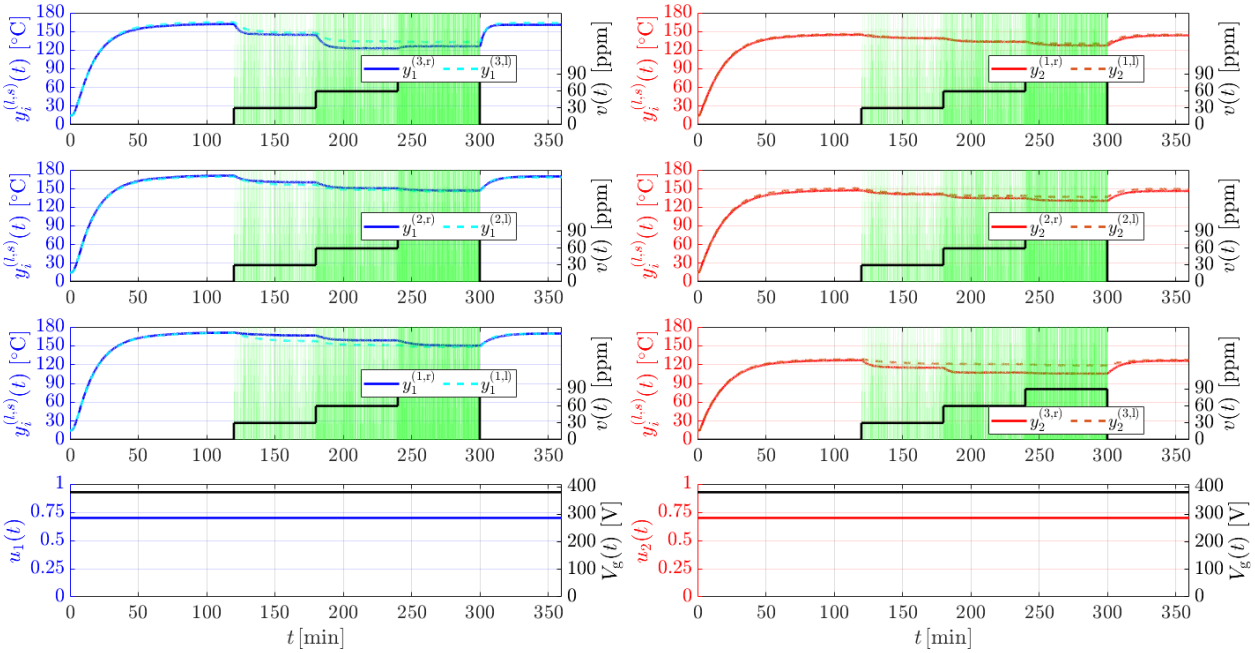


Figure 14: Results obtained by the open-loop simulation (`simulate_OL.m`). We report the temperatures $y_i^{(l,s)}(t), i \in \{1,2\}, l \in \{1,2,3\}, s \in \{\mathrm{l},\mathrm{r}\}$, duty cycles $u_1(t), u_2(t)$, grid voltage $V_\mathrm{g}(t)$, and production rate $v(t)$. The green vertical lines denote when $d\,(t; v(t)) = 1$.

We remark that the `simulate_OL.m` MATLAB script relies on an auxiliary MATLAB script called `set_up_parameters_for_simulink.m` that opens the Simulink model and sets up all its parameters. We report its code here:

Listing 3: `set_up_parameters_for_simulink.m`.

```
%% Path
mydir  = pwd;
addpath(genpath([pwd, filesep, 'file_mat']))
addpath(genpath([pwd, filesep, 'support']))

%% General parameters
T_sim = 1e-2; % Simulation time (and sampling time) [s]
T_a = 15; % Ambient temperature
T_P_1 = 1; % Period of the SSRs [s]
T_P_2 = 30; % Period of the EMRs [s]

simulink_name_ol = 'complete_model_ol';
simulink_name_cl = 'complete_model_cl';

%% Other parameters
bufferSize = 10000; % for transport delays

%% General imports and definitions
load('success_rates_packs_temperatures.mat')
```

```
load('delay_heat_temperatures.mat')

rng(1)

%% User defined parameters and imports
if not(exist('T_c', 'var'))
    error('You must define the sampling time T_c')
end

if not(exist('simulink_type', 'var'))
    error('You must define simulink_type, either ''ol'' or ''cl''')
end

if T_c == T_P_1
    duty_cycle_allocation_name = 'duty_cycle_allocation_T_P_1';
elseif floor(T_c/T_P_2) == T_c/T_P_2
    duty_cycle_allocation_name = 'duty_cycle_allocation_mT_P_2';
else
    error('Wrong choice of T_c. T_c must be either be 1 or a multiple of 30')
end

p = 12; % Number of outputs
heat_temperatures_model_name = 'heat_temperatures_12';
packs_temperatures_model_name = 'packs_temperatures_12';
load('ss_heat_temperatures_12.mat')
load('params_packs_temperatures_12.mat')

switch simulink_type
    case 'ol'
        simulink_name = simulink_name_ol;
    case 'cl'
        simulink_name = simulink_name_cl;
    otherwise
        error('Wrong choice of simulink_type. simulink_type must be either ''ol'' or
            ''cl''')
end

%% Open Simulink and set subsystems
open([simulink_name, '.slx'])
set_param([simulink_name, '/Plant/Heat-Temperatures model'], 'ReferencedSubsystem',
    heat_temperatures_model_name);
set_param([simulink_name, '/Plant/Packs-Temperatures model'], 'ReferencedSubsystem',
    packs_temperatures_model_name);
set_param([simulink_name, '/PWM_Generation'], 'ReferencedSubsystem',
    'PWM_generation');
set_param([simulink_name, '/Duty cycle allocation'], 'ReferencedSubsystem',
    duty_cycle_allocation_name);
```

Notice that, in Listing 3, we define the ambient temperature $T_a(t)$, which is constant for all $t \in \mathbb{R}_{\geq 0}$, the PWM periods for the SSRs ($\mathcal{T}_P^{(1)}$) and EMRs ($\mathcal{T}_P^{(2)}$), and the simulation time for Simulink $\mathcal{T}_{sim}$. The estimated parameters (Table 2 and Table 4) are saved in different `.mat` files that are loaded by the `set_up_parameters_for_simulink.m` MATLAB script in Listing 3. Finally, there are several checks on $\mathcal{T}_c$ that prevent the user from selecting wrong sampling times for the controller.

3.2 Simulation of the shrink tunnel in closed loop

This Section describes the MATLAB scripts and the Simulink models that allow the user to actually run the benchmark in closed-loop as intended in [11], deriving the performance indicators of interest. `complete_model_cl.slx` is the

Simulink model that implements the scheme in Figure 2, which is composed of several subsystems, most of which were already described in Section 3.1:

- The `controller.slx` subsystem corresponds to the controller block in Figure 2;
- The duty cycle allocation subsystem can either be `duty_cycle_allocation_T_P_1.slx` (`OM1`) or `duty_cycle_allocation_mT_P_2.slx` (`OM2`) depending on the operation mode (see [11]). It corresponds to the duty cycle allocation block in Figure 2;
- The `PWM_generation.slx` subsystem corresponds to the PWM generation block in Figure 2;
- The `plant.slx` subsystem matches the plant block in Figure 2.

Further details are given in Section 3.2.1. The MATLAB script `run_benchmark.m` can be used to run the `complete_model_cl.slx` Simulink model to effectively test the user-implemented controller, as described in Section 3.2.2. An example is given in Section 3.3.

3.2.1 The `complete_model_cl.slx` Simulink model

The `complete_model_cl.slx` Simulink model is depicted in Figure 15. Notice its similarity to the block diagram in Figure 2. Via the script `simulate_OL.m` (Section 3.1.2), the user sets the sampling time for the controller $\mathcal{T}_c$, which

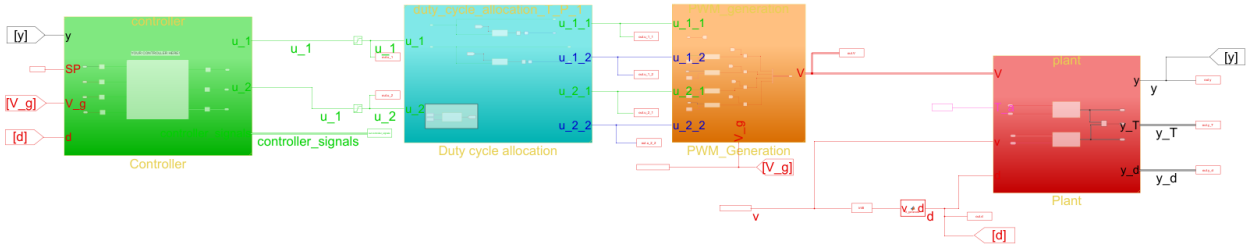


Figure 15: The `complete_model_cl.slx` Simulink model. The signals in green are updated every second (i.e. $\mathcal{T}_P^{(1)}$), those in blue are updated every 30 s (i.e. $\mathcal{T}_P^{(2)}$), the ones in red every $\mathcal{T}_{sim}$ seconds, and finally the signals in black are "continuous-time signals". In this case, the sampling time for the controller was set to $\mathcal{T}_c = 1$ s.

can either be:

- $\mathcal{T}_c = 1$ s (operation mode 1 [11]) or
- $\mathcal{T}_c = m \cdot 30$ s, $m \in \mathbb{N}$ (operation mode 2 [11]),

depending on the computational burden of their regulator. The signals of interest for the benchmark are set up as described in [11] (see also Figure 16), namely:

1. The setpoint is constant $\mathrm{SP}(t) = \bar{\mathrm{SP}}, \bar{\mathrm{SP}} = 160°\mathrm{C}, \forall t \in \mathbb{R}_{\geq 0}$;
2. The grid voltage $V_g(t) \in \mathbb{R}_{\geq 0}$ fluctuates between 380 V and 410 V;
3. The production rate $v\,(t) \in \{0, 30, 60, 90\}$ exhibits several rate changes and breaks.

Instead, the ambient temperature is already fixed at $T_a(t) = 15°\mathrm{C}, \forall t \in \mathbb{R}_{\geq 0}$, while the signal $d\,(t; v(t))$ is generated as in (22). The outputs of the simulation are the temperatures $\boldsymbol{y}\,(t)\,, \boldsymbol{y}_T(t)$, and $\boldsymbol{y}_d(t)$. Next, we review the `controller.slx` subsystem for the `complete_model_cl.slx` Simulink model. All the other subsystems were already reviewed in Section 3.1.1.

The Simulink implementation of the `controller.slx` subsystem is presented in Figure 17. At the core of it there is a subsystem reference block called "`Your_controller`" where the user plugs in their controller. In practice, this is done automatically by the `run_benchmark.m` MATLAB script as explained in Section 3.2.2. An example of Simulink model that can be referenced by the "`Your_controller`" block is reported in Section 3.3. In any case, the signals available to the controller are:

- The outputs of the shrink tunnel model $\boldsymbol{y}\,(t) \in \mathbb{R}^{12}$;
- The setpoint $\mathrm{SP}(t) \in \mathbb{R}$;

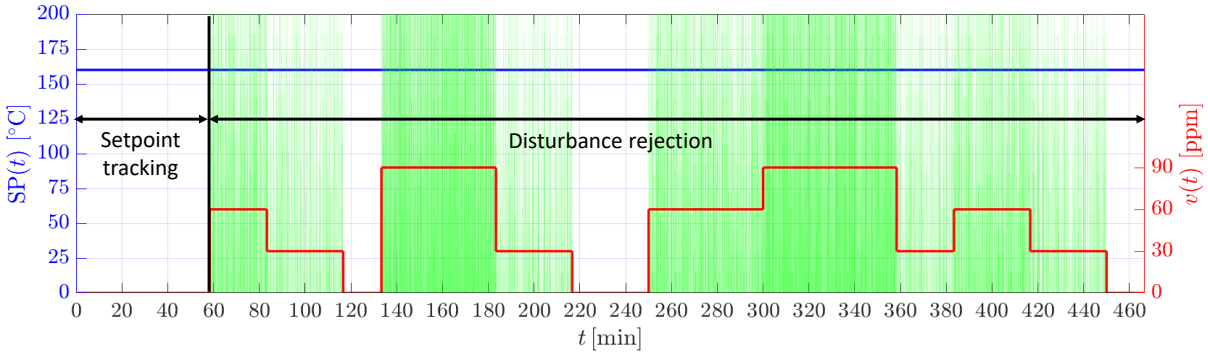


Figure 16: Signals of interest for the benchmark: setpoint SP (t) and production rate $v\,(t)$. The green vertical lines denote when $d\,(t; v(t)) = 1$.

- The grid voltage $V_g(t) \in \mathbb{R}_{\geq 0}$;
- The signal $d\,(t; v(t))$ detecting presence/absence of bottle packs at the entrance of the oven cavity.

These are all sampled at the sampling time $\mathcal{T}_c$. The controller must produce the control actions $u_1\,(t)$ and $u_2\,(t)$, but also the user may choose to save some controller-related signals for debugging purposes.

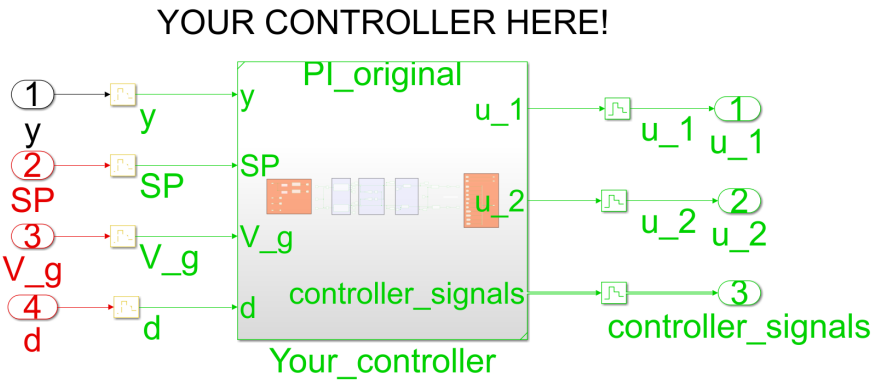


Figure 17: The `controller.slx` Simulink model. The signals in green are updated every $\mathcal{T}_c$ seconds (in this case $\mathcal{T}_c = 1$ s), the ones in red every $\mathcal{T}_{sim}$ seconds, and finally the signals in black are "continuous-time signals".

3.2.2 The `run_benchmark.m` MATLAB script

First of all, the user should implement their controller in a `.slx` file, taking into account the inputs and outputs of the "`Your_controller`" block inside the `controller.slx` Simulink model (see Section 3.2.1). The `.slx` file should be saved inside the folder `\support\controller\implemented` as in Figure 18. In the provided code, the user can find an example of controller that was implemented by the authors, namely the `PI_original.slx` controller. Further details are given in Section 3.3. Together with the `.slx` file, the user may also code a MATLAB script that sets the parameters of their regulator, such as the `PI_original_parameters.m` MATLAB script for the provided controller (`PI_original.slx`). After this preliminary implementation step, the user moves on to the `run_benchmark.m` MATLAB script. The first lines of code amount to:

Listing 4: `run_benchmark.m`, first part.

```
%% Setup
clc
clear
close all

% Choose the sampling of your controller
```

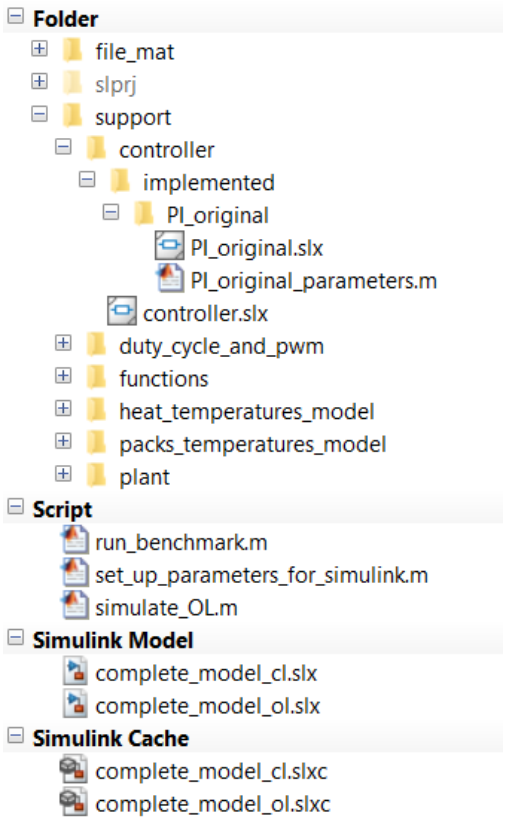


Figure 18: Folder for the user-defined controller.

```
7     % Either 1 or a multiple of 30
8     T_c = 1; % [s]
9     simulink_type = 'cl';
10
11    set_up_parameters_for_simulink
12
13    % Reference your controller in Simulink
14    your_controller_name = 'PI_original';
15    set_param([simulink_name, '/Controller/Your_controller'], 'ReferencedSubsystem',
          your_controller_name);
16    % Run the script which sets the parameters of your controller
17    PI_original_parameters
```

The user should select $\mathcal{T}_c$ (line 8, Listing 4) so that it is in line with the computational burden of their controller (either $\mathcal{T}_c = 1\,\mathrm{s}$ or $\mathcal{T}_c = m \cdot 30\,\mathrm{s}, m \in \mathbb{N}$). Next, they must specify the name of the controller (line 14, Listing 4) and run the MATLAB script that sets its parameters (if present, line 17, Listing 4).

The next lines of code are used to create the signals for the benchmark as described in [11] (see also Figure 16).

Listing 5: `run_benchmark.m`, second part.

```
1     %% Signal generation
2     trial_length = 28000; % [s]
3     times = (0 : T_sim : trial_length)'; % [s]
4     N = length(times);
5
6     % Grid voltage
7     V_g = zeros(N, 1);
```

```
    for i = 1 : N
        V_g(i,1) = randi([380 410], 1, 1);

    end
    V_grid_filter = designfilt('lowpassiir','FilterOrder',2, ...
        'HalfPowerFrequency',0.05,'DesignMethod','butter');

    V_g = filtfilt(V_grid_filter, V_g);
    V_g = timeseries(V_g, times);

    % Production rate
    v = zeros(N, 1);
    v(350000:500000) = 60;
    v(500000:700000) = 30;
    v(700000:750000) = 0;
    v(800000:1100000) = 90;
    v(1100000:1300000) = 30;
    v(1300000:1350000) = 0;
    v(1500000:1800000) = 60;
    v(1800000:2100000) = 90;
    v(2100000:2150000) = 90;
    v(2150000:2300000) = 30;
    v(2300000:2500000) = 60;
    v(2500000:2700000) = 30;
    v = timeseries(v, times);

    % Setpoint
    SP = 160*ones(N, 1);
    SP = timeseries(SP, times);
```

Finally, the user can run the simulation of the `complete_model_cl.slx` Simulink model with the following lines of code.

Listing 6: `run_benchmark.m`, third part.

```
    %% Simulation
    set_param(simulink_name, 'Solver', 'ode45', 'StartTime', num2str(times(1)),
        'StopTime', num2str(times(end)))

    warning('off', 'Simulink:blocks:SwitchIgnoringThreshold')
    out = sim(simulink_name, 'ReturnWorkspaceOutputs', 'on');
    warning('on', 'Simulink:blocks:SwitchIgnoringThreshold')
```

The remaining lines of code are used to visualize the achieved results and compute the performance indicators reported in [11].

3.3 Benchmark example

In this Section, we describe the controller that is already implemented in the provided code, which is made up of the `PI_original.slx` Simulink model and the `PI_original_parameters.m` MATLAB script. This is the baseline controller for the benchmark and it actually matches the same regulator that was used during the closed-loop experiment used for parameters estimation (Section 2).

The `PI_original_parameters.m` controller is made up of:

1. Two PI controllers [1]: one regulates the temperature $y_1^{(1,\mathrm{r})}$ while the other controls $y_2^{(2,\mathrm{l})}$. Basically, rather than considering all the thermocouples in each zone, we only use the information coming from one sensor per zone to simplify the control architecture. Clearly, this approach can undermine the performances.
2. A static decoupler [2].

3. A conditioned transfer anti-windup strategy [9].
4. A rescaling strategy that prevents the average power supplied to the heat resistors from exceeding $P_{\max}$ [11].

The block diagram of the just mentioned regulator is reported in Figure 19.

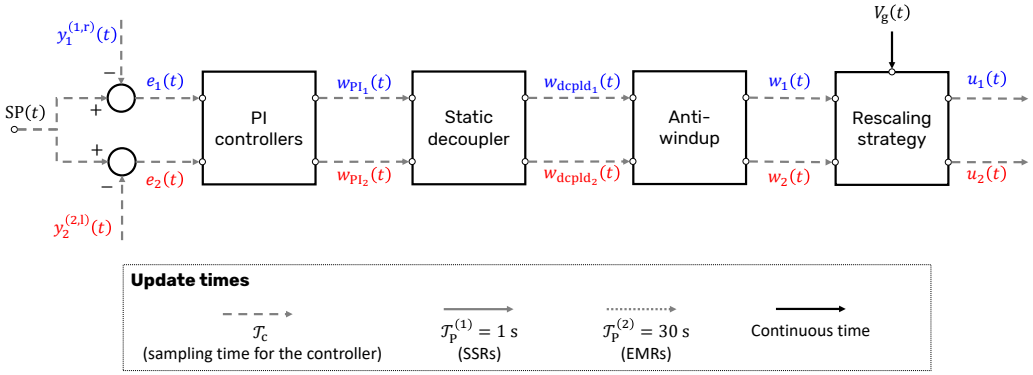


Figure 19: Block diagram of the control architecture for the `PI_original.slx` Simulink model. The signals in blue are associated with the first zone while those in red are related to the second zone. Instead, the signals in black are not related to any zone in particular. The different arrows highlight how often each signal is updated.

In practice, the PI controllers assume that the shrink tunnel always operates in the nominal operating conditions, i.e. at $V_g(t) = V_{\text{nom}} = 380\,\text{V}, \forall t \in \mathbb{R}_{\geq 0}$ (nominal voltage). For this reason, the control architecture firstly produces the control actions $w_i(t) \in [0,1], i \in \{1,2\}$, under this assumption. Then, a rescaling strategy is employed to convert $w_i(t)$ into $u_i(t)$ to take into account that the grid voltage may differ from V_{nom} and may vary over time. In any case, the PI controllers, the static decoupler, and the anti-windup strategy were tuned based on a simplified model of the shrink tunnel. More specifically, the relationship between the duty cycles $w_1(t), w_2(t)$, and the temperatures $y_1^{(1,r)}(t), y_2^{(2,l)}(t)$ was approximated by four FOLPD transfer functions, i.e. a FOLPD model for each input/output pair. The FOLPD models were estimated from two experiments on the shrink tunnel carried out in the nominal operating conditions. See [10] for more details. Due to the simplicity of the control architecture in Figure 19, we have chosen to run the controller at a sampling time $\mathcal{T}_c = 1$ s (the lowest possible). Finally, notice that this regulator does not use the signal $d(t; v(t))$, leading to subpar disturbance rejection performances.

Concerning the constraints described in [11], the conditioned transfer anti-windup strategy [9] ensures that the duty cycles are such that

$$0 \leq w_i(t) \leq 1, \quad \forall i \in \{1,2\}, t \in \mathbb{R}_{\geq 0}.$$

However, we also need to enforce:

$$\frac{1}{\mathcal{T}_P^{(r)}} \cdot \int_{k \cdot \mathcal{T}_P^{(r)}}^{(k+1) \cdot \mathcal{T}_P^{(r)}} \frac{V_i^{(r)}(t)^2}{R_{\text{heat}}}\, dt \leq P_{\max}, \quad \forall i \in \{1,2\}, r \in \{1,2\}, k \in \mathbb{N} \cup \{0\}. \tag{38}$$

First, notice that $\mathcal{T}_c = \mathcal{T}_P^{(1)} = 1$ s (i.e. operation mode 1 in (34)), leading to:

$$u_i^{(1)}(t) = u_i(t), \quad \forall i \in \{1,2\}, t \in \mathbb{R}_{\geq 0}.$$

To derive $u_i(t)$ from $w_i(t)$, we match the average power in the nominal conditions with the average power in the perturbed conditions (i.e. for varying $V_g(t)$). In particular, consider the PWM period

$$k \cdot \mathcal{T}_P^{(1)} \leq t \leq (k+1) \cdot \mathcal{T}_P^{(1)}$$

for the solid-state relays ($r = 1$). The average power in the perturbed conditions amounts to [11]:

$$\frac{1}{\mathcal{T}_P^{(1)}} \cdot \int_{k \cdot \mathcal{T}_P^{(1)}}^{(k+1) \cdot \mathcal{T}_P^{(1)}} \frac{V_i^{(1)}(t)^2}{R_{\text{heat}}}\, dt = \frac{1}{\mathcal{T}_P^{(1)}} \cdot \int_{k \cdot \mathcal{T}_P^{(1)}}^{\left[k + u_i\left(k \cdot \mathcal{T}_P^{(1)}\right)\right] \cdot \mathcal{T}_P^{(1)}} \frac{V_g(t)^2}{R_{\text{heat}}}\, dt, \quad \forall i \in \{1,2\}. \tag{39}$$

By substituting $V_g(t) = V_{\text{nom}}$ and $u_i(t)$ with $w_i(t)$ we get the average power in the nominal conditions:

$$\frac{1}{\mathcal{T}_P^{(1)}} \cdot \int_{k \cdot \mathcal{T}_P^{(1)}}^{\left[k + w_i\left(k \cdot \mathcal{T}_P^{(1)}\right)\right] \cdot \mathcal{T}_P^{(1)}} \frac{V_{\text{nom}}^2}{R_{\text{heat}}}\, dt = w_i\left(k \cdot \mathcal{T}_P^{(1)}\right) \cdot \frac{V_{\text{nom}}^2}{R_{\text{heat}}}, \quad \forall i \in \{1,2\}. \tag{40}$$

We match the average powers in (39) and (40):

$$w_i\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)\cdot\frac{V_{\mathrm{nom}}^2}{R_{\mathrm{heat}}}=\frac{1}{\mathcal{T}_{\mathrm{P}}^{(1)}}\cdot\int_{k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}}^{\left[k+u_i\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)\right]\cdot\mathcal{T}_{\mathrm{P}}^{(1)}}\frac{V_{\mathrm{g}}\left(t\right)^2}{R_{\mathrm{heat}}}\,dt. \tag{41}$$

Next, assuming that the grid voltage stays roughly constant in each PWM period, we have [11]:

$$\int_{k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}}^{\left[k+u_i\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)\right]\cdot\mathcal{T}_{\mathrm{P}}^{(1)}}V_{\mathrm{g}}\left(t\right)^2\,dt\approx V_{\mathrm{g}}\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)^2\cdot u_i\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)\cdot\mathcal{T}_{\mathrm{P}}^{(1)}. \tag{42}$$

By substituting the above expression into (41), we get:

$$\begin{aligned}
w_i\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)\cdot\frac{V_{\mathrm{nom}}^2}{R_{\mathrm{heat}}}&\approx\frac{1}{\mathcal{T}_{\mathrm{P}}^{(1)}}\cdot\frac{V_{\mathrm{g}}\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)^2}{R_{\mathrm{heat}}}\cdot u_i\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\\
w_i\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)\cdot V_{\mathrm{nom}}^2&\approx u_i\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)\cdot V_{\mathrm{g}}\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)^2\\
&\Downarrow\\
u_i\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)&\approx\frac{V_{\mathrm{nom}^2}}{V_{\mathrm{g}}\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)^2}\cdot w_i\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right),\quad\forall i\in\{1,2\}.
\end{aligned}$$

To account for the fact that $V_{\mathrm{g}}(t)$ may be lower than V_{nom}, which would lead to $u_i\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)>1$, we define the rescaling strategy in Figure 19 as:

$$u_i\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)=\min\left\{\frac{V_{\mathrm{nom}}^2}{V_{\mathrm{g}}\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right)^2}\cdot w_i\left(k\cdot\mathcal{T}_{\mathrm{P}}^{(1)}\right),1\right\},\quad\forall k\in\mathbb{N}\cup\{0\},i\in\{1,2\}. \tag{43}$$

Consequently, according to operation mode 1, the duty cycles of the SSRs and EMRs are computed as in (34). Then, under the assumption in (42), the constraint (38) is surely satisfied for the voltages generated by the solid-state relays ($V_1^{(1)}(t)$ and $V_2^{(1)}(t)$). Instead, for what concerns the voltages driven by the electro-mechanical relays $V_1^{(2)}(t)$ and $V_2^{(2)}(t)$, which are produced based on an average duty cycle as in (34), the constraint (38) is roughly satisfied if the grid voltage does not vary by much within $\mathcal{T}_{\mathrm{P}}^{(2)}=30\,\mathrm{s}$.

3.3.1 Performances

We test the control architecture in Figure 19 as described in Section 3.2. After running the `run_benchmark.m` MATLAB script, we visualize the outputs returned by the Simulink model (see Figure 20). Notice that only $y_1^{(1,\mathrm{r})}$ and $y_2^{(2,\mathrm{l})}$ closely follow $\mathrm{SP}(t)$ due to how we have defined the controller. Concerning the performance indicators reported in [11], we report their respective values in Table 5. We also display a graphical representation of the obtained results in Figure 21 (setpoint tracking) and Figure 22 (disturbance rejection).

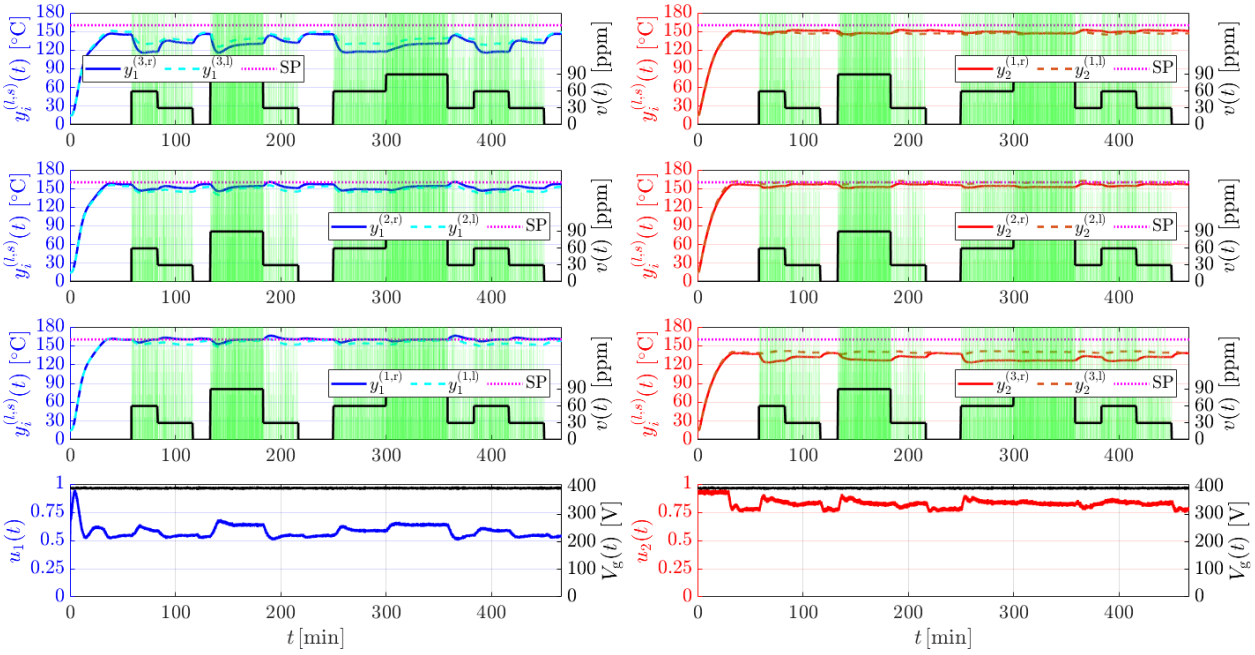


Figure 20: Results obtained by the closed-loop simulation (`run_benchmark.m`) with the `PI_original.slx` controller. We report the temperatures $y_i^{(l,s)}(t), i \in \{1,2\}, l \in \{1,2,3\}, s \in \{\mathrm{l},\mathrm{r}\}$, duty cycles $u_1(t), u_2(t)$, grid voltage $V_\mathrm{g}(t)$, and production rate $v(t)$. The green vertical lines denote when $d\,(t; v(t)) = 1$.

Table 5: Performances of the controller in Figure 19 w.r.t. the indicators reported in [11].

	$y_1^{(3,\mathrm{r})}$	$y_1^{(3,\mathrm{l})}$	$y_1^{(2,\mathrm{r})}$	$y_1^{(2,\mathrm{l})}$	$y_1^{(1,\mathrm{r})}$	$y_1^{(1,\mathrm{l})}$	$y_2^{(1,\mathrm{r})}$	$y_2^{(1,\mathrm{l})}$	$y_2^{(2,\mathrm{r})}$	$y_2^{(2,\mathrm{l})}$	$y_2^{(3,\mathrm{r})}$	$y_2^{(3,\mathrm{l})}$	**Average**
$t_{\mathrm{s}_i}^{(l,s)}$ [min]	33.12	33.20	32.92	33.03	32.51	32.42	30.32	30.77	29.72	29.73	29.59	29.50	31.40
$e_{\mathrm{SS}_i}^{(l,s)}$ [°C]	14.76	10.47	3.37	5.96	0.19	0.04	9.97	11.77	4.55	0.17	23.23	21.17	8.80
$\mathrm{AUC}_i^{(l,s)}$ [°C · min]	6196.37	4874.57	1925.15	2482.50	575.00	2048.96	686.97	551.85	762.65	267.67	2855.05	509.67	1978.03
E_V [W · h]	2772.43												2772.43

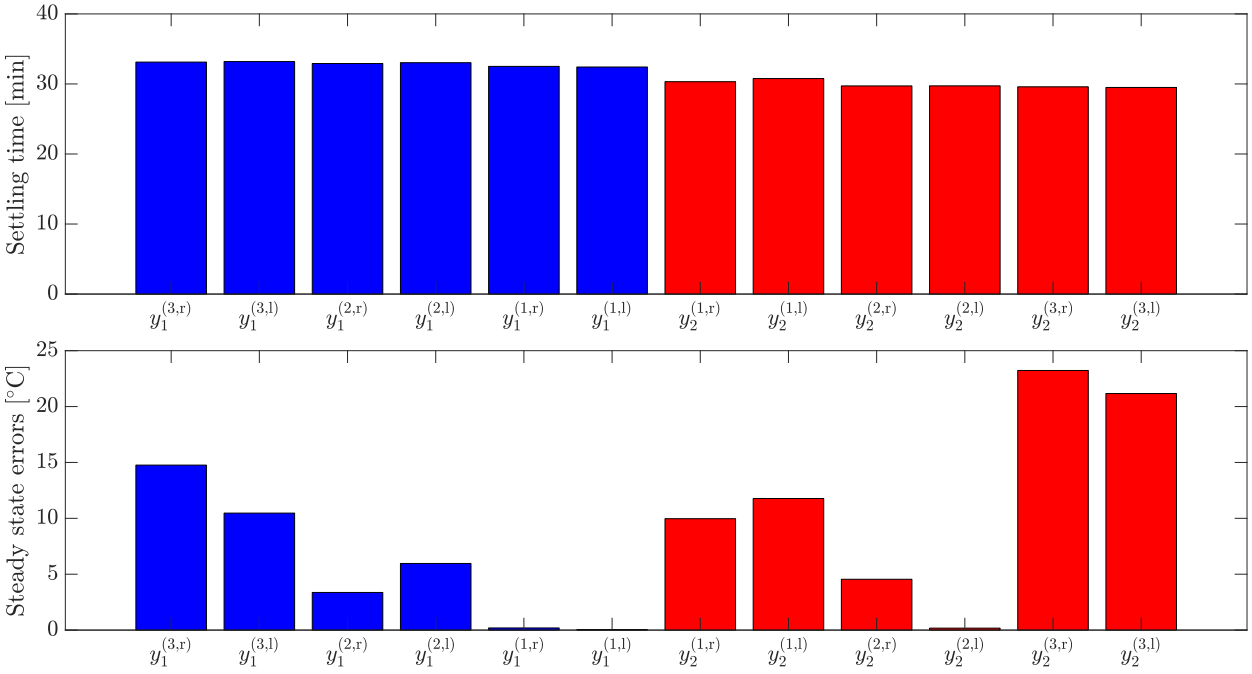


Figure 21: Results obtained by the closed-loop simulation (`run_benchmark.m`) with the `PI_original.slx` controller for what concerns the setpoint tracking indicators (settling times and absolute steady-state errors).

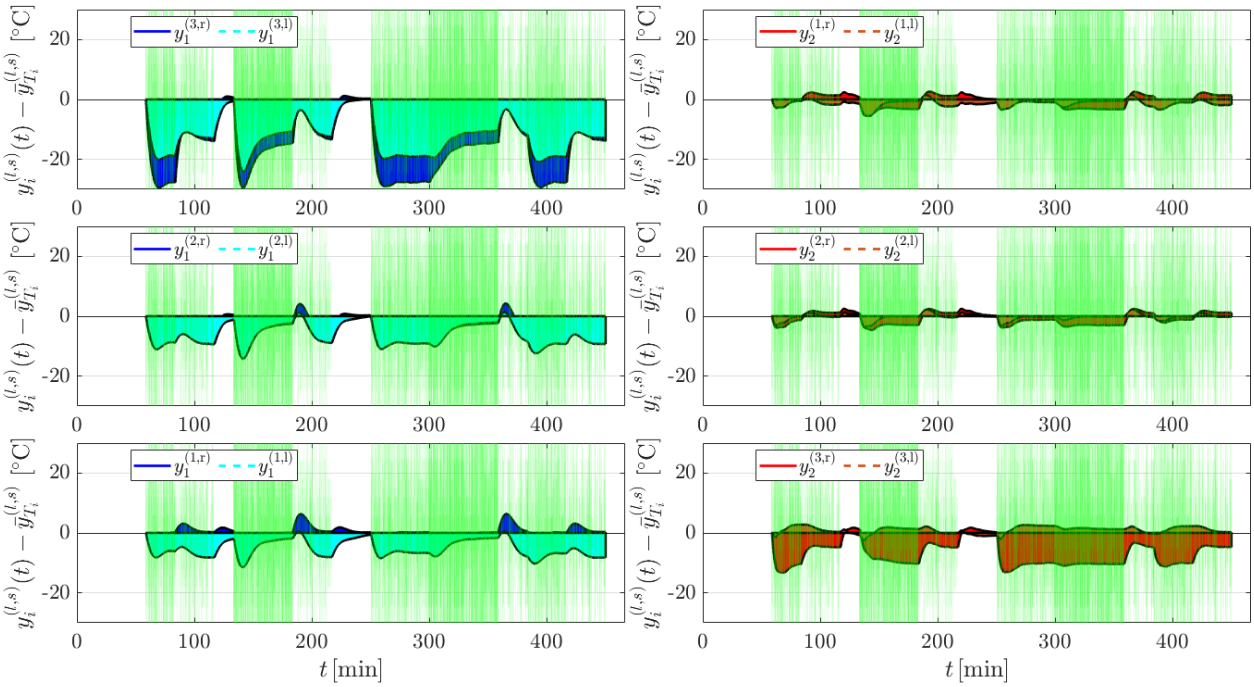


Figure 22: Results obtained by the closed-loop simulation (`run_benchmark.m`) with the `PI_original.slx` controller for what concerns the disturbance rejection indicator (area under the curve). The colored areas represent, graphically, the AUC in [11].

References

[1] Karl J. Astrom and Tore Hagglund. *PID controllers: theory, design, and tuning*. The Instrumentation, Systems, and Automation Society, 1995.

[2] Juan Garrido, Francisco Vázquez, and Fernando Morilla. An extended approach of inverted decoupling. *Journal of process control*, 21(1):55–68, 2011.

[3] David Halliday, Robert Resnick, and Jearl Walker. *Fundamentals of physics*. John Wiley & Sons, 2013.

[4] MATLAB. Matlab's greyest method. `https://www.mathworks.com/help/ident/ref/greyest.html`. Accessed: 2023-07-06.

[5] MATLAB. Matlab's procest method. `https://www.mathworks.com/help/ident/ref/procest.html`. Accessed: 2023-07-06.

[6] Julio Elias Normey-Rico and Eduardo F. Camacho. *Control of dead-time processes*, volume 462. Springer, 2007.

[7] Aidan O'dwyer. *Handbook of PI and PID controller tuning rules*. Imperial College Press, 2009.

[8] Katsuhiko Ogata. *Modern control engineering*, volume 5. Prentice hall Upper Saddle River, NJ, 2010.

[9] Youbin Peng, Damir Vrancic, and Raymond Hanus. Anti-windup, bumpless, and conditioned transfer techniques for pid controllers. *IEEE Control systems magazine*, 16(4):48–57, 1996.

[10] Davide Previtali, Leandro Pitturelli, Fabio Previdi, and Antonio Ferramosca. Model-based design of the temperature controller of a shrink tunnel. *Accepted for pubblication in the proceedings of the IFAC ADCHEM 2024 conference*.

[11] Davide Previtali, Leandro Pitturelli, Fabio Previdi, and Antonio Ferramosca. Temperature control of a shrink tunnel with multiple heating zones. In *Control Systems Benchmarks: Key Resources for Researchers and Practitioners*. Chapter accepted, book not published yet.

[12] Sheldon M. Ross. *Introduction to probability and statistics for engineers and scientists*. Academic press, 2020.

[13] Michel Verhaegen and Vincent Verdult. *Filtering and system identification: a least squares approach*. Cambridge University Press, 2007.