Generalized split range control using the baton strategy

Adriana Reyes-Lúa

Department of Chemical Engineering, Norwegian University of Science and Technology (NTNU), Sem Sælands vei 4, 7491, Trondheim, Norway

Sigurd Skogestad¹

Department of Chemical Engineering, Norwegian University of Science and Technology (NTNU), Sem Sælands vei 4, 7491, Trondheim, Norway

Abstract

Split range control is used to extend the steady-state operating range for a single output (controlled variable) by using more than one input (manipulated variable). The standard implementation of split range control uses a single controller and a split range block, but this approach has limitations when it comes to tuning. In this paper, we introduce a generalized split range control structure that overcomes these limitations by using multiple independent controllers with the same setpoint. Undesired switching between the controllers is avoided by using a baton strategy where only one controller is active at a time. We compare our proposed structure with standard split range control in a simulation case study.

Keywords: split range control, control structure, PID, tuning, anti-windup, baton, multiple input

1. Introduction

Classical advanced control extends the single-loop PI(D)-controller to cover more difficult control tasks and includes, for example, cascade control, feedforward control, decoupling, selectors, split range control, parallel control, and

¹Corresponding author: sigurd.skogestad@ntnu.no
valve position control (also called input resetting or mid-ranging control) (e.g. [1, 2, 3]). When we need more than one input ($u_i$) to cover the steady-state operating range for a single output ($y$), we can use three alternative classical control structures:

1. (Standard) split range control (Fig. 1),

2. One controller for each input, each with a different setpoint for the output (Fig. 2),

3. Input (valve) position control (Fig. 3).

Figure 1: Classical structure 1: Standard implementation of split range control (SRC) with two inputs ($u_i$) and one output ($y$). A typical SR-block is shown in Fig. 4. $u_0$ contains information about maximum and minimum values for both physical inputs.

Figure 2: Classical structure 2: two controllers for the same output ($y$), each controller with a different setpoint ($y^{sp,1}$ and $y^{sp,2}$).

In this paper we focus on split range control (Fig. 1), although the solution we are proposing is based on using two (or more) independent controllers, as in Fig. 2. Some other names that have been used for split range control are dual control agent [4], range extending [5] and valve sequencing [6]. The most common application of split range control is to extend the steady-state range
when the primary input saturates. For example, we may have two sources of heating and use the least expensive heat source first. In some cases, the available inputs have opposite effects on the controlled variable. A typical example is a process that requires both heating and cooling.

Split range control has been in use for more than 75 years [4, 7], and has been extensively applied in industry [6, 8]. However, except for basic descriptions and examples of applications (see [9, 10, 11, 12, 13, 3, 14, 15, 16]), we have not found a systematic design procedure, and there are almost no academic studies.

Therefore, in a previous paper [17], we proposed a systematic procedure to design a standard (classical) split range controller (Fig. 1). However, as we explain in Section 2, standard split range control has limitations in terms of tuning. For example, we must use the same integral time for all inputs, which is generally not desirable for dynamic performance.

To allow for independent controller tunings, one alternative is to use multiple controllers with different setpoints (Fig. 2). For example, when controlling the temperature in a room \((y = T)\), one may use \(y^{sp,1} = 23^\circ C\) as the setpoint for cooling \((u_1)\) and \(y^{sp,2} = 21^\circ C\) as the setpoint for heating \((u_2)\). The use of different setpoints is to avoid undesired switching between the controllers and possible non-uniqueness when using two controllers with integral action to control the same output [14].

In the present paper, we propose a generalized split range control structure, where the controller for each input can be designed independently. To avoid the use of different setpoints, we use a *baton strategy*, in which undesired switching
is avoided by allowing only one controller to be active at a time.

This paper is organized as follows. In Section 2 we briefly describe standard split range control and its limitations with respect to tuning. In Section 3 we present a new generalized control structure for split range control, which overcomes these limitations. In Section 4 we use a case study to illustrate our proposed generalized structure and compare it to standard split range control. Then, in Section 5 we discuss about improvements for initialization, anti-windup implementation, and the relation of our proposed generalized structure with other control structures. We conclude the paper with some final remarks in Section 6.

2. Standard split range controller

As shown in Fig. 1, in standard split range control there is one common controller (C) which computes the internal signal (v) to the split range block (SR-block), which assigns the value (e.g the valve opening) for each of the inputs (u_i). Importantly, at any particular time, only one input (u_i) is being used to control the output (y), whereas the remaining inputs are fixed, typically saturated at their maximum or minimum values.

2.1. The split range block

The split range block has also been called characterization function [16], splitter block [18], and function generator [19].

Fig. 4 depicts a typical split range block for two inputs (u_1 and u_2) for a case when u_1 has a positive effect on the output (y) and u_2 has a negative effect. Fig. 4a depicts the most common case with a split value at v^* = 50%. Also note that the internal variable (v) is in deviation variables, so the actual value of v many not have any physical significance, while the variables u_i are in physical units.
2.2. Slopes ($\alpha_i$) in split range block

In Fig. 4a, the split value is located at the mid-point ($v^* = 50\%$) and the slopes have the same magnitude ($|\alpha_1| = |\alpha_2|$). This choice is used in most examples in the literature (see [20, 21, 22, 3, 13, 23, 16, 24, 25, 26, 27]). However, each input ($u_i$) has a different dynamic and static effect on the output, and the split value ($v^*$ or equivalently, the slopes $\alpha_i$) should generally be located at some other value to compensate for this, as illustrated in Fig. 4b (see [6, 11, 28, 29, 18, 30, 17]).

Let $C_i(s)$ denote the desired controller for each input ($u_i$)$^2$. With standard split range control, there is a common controller $C(s)$ and the resulting controller from $y$ to each $u_i$ is $\alpha_i C(s)$. We would like to make $\alpha_i C(s) = C_i(s)$ for all inputs, but since we only have one design parameter for each input ($\alpha_i$), this is not possible in general. For example, with a PI-controller (with parameters $K_C$ and $\tau_I$), we can get the desired controller gain for each input ($K_{C,i}$) by selecting $\alpha_i$ such that $\alpha_i K_C = K_{C,i}$. The common integral time ($\tau_{I,i}$) is then chosen as a compromise among the desired $\tau_{I,i}$ for every $u_i$.

---

$^2$The desired tunings for these controllers can be found, for example, from the SIMC PID tuning rules [31].


3. Generalized split range control structure

3.1. Proposed structure

The dynamic behavior of each input is generally different and using a common controller \(C(s)\), as in standard split range control (Fig. 1), represents a compromise. Fig. 5 depicts our proposed generalized control structure for split range control applications, where each input has its own controller \(C_i(s)\). Here, \(C_i(s)\) can be any type of controller, but it is commonly a PI controller\(^3\). Each controller produces a signal, which is a deviation variable \((v_i = \Delta u_i)\), and the baton strategy block in Fig. 5 selects and computes the physical inputs \((u_i)\), based on a predefined sequence.

\[\begin{aligned}
C_1 & \quad v_1 \\
G & \quad y_{sp}\end{aligned}\]

\[\begin{aligned}
\text{Baton strategy block} & \quad u_1 \\
\text{(Process)} & \quad y \\
\end{aligned}\]

Figure 5: Proposed generalized split range control using the baton strategy. Note that \(v_i\) are deviation variables, whereas \(u_i\) are in physical units. Note here that \(u_i\) contains the bias information (maximum and minimum values for each input).

3.2. Baton strategy

In order to be able to use multiple controllers for the same output, we must make sure that only one input \((u_i)\) is actively controlling the output \((y)\) at any

\(^3\)Having independent controllers for each input \((u_i)\) allows one to individually tune the controller for each \(u_i\), without any compromise. To design \(C_i(s)\) we suggest using a systematic tuning procedure, such as the SIMC PID tuning rules [31].
given time. The other inputs are required to be at fixed values ($u_i^{\text{min}}$ or $u_i^{\text{max}}$).

We propose to do this by using a baton strategy, similar to what is used by runners in a relay (Fig. 6), where only the runner who holds the baton is active at any given time, and the active runner decides when to pass the baton. This avoids the need for a centralized supervisor. In other words, we let the active input decide when to switch to another input. The active input remains active as long as its not saturated ($u_i^{\text{min}} < u_i < u_i^{\text{max}}$) and will only pass the baton to another input once it becomes saturated (reaches $u_i^{\text{min}}$ or $u_i^{\text{max}}$).

Figure 6: Baton strategy for relay.

3.3. Sequencing of inputs

Before actually designing the baton strategy, we need to make some initial decisions. First, we need to define the minimum and maximum values for every input ($u_i^{\text{min}}$, $u_i^{\text{max}}$) This is decision D1. Then, we need to choose the sequence of use of the inputs (decision D2). This should be defined considering their effect on the output ($y$) as well as economic aspects. In some cases, operational aspects may be taken into account. The following is used for decision D2:

D2.1 Define the desired or most economical operating point for each input (e.g. fully closed or fully open valve).

D2.2 Consider the effect of every input ($u_i$) on the output ($y$). Then group the inputs into:

(a) Inputs for which the value of the output increases when we move away from the desired operating condition (fully opened or fully closed).
(b) Inputs for which the value of the output decreases when we move away from the desired operating condition (fully opened or fully closed).

D2.3 Within each group, (a) and (b), order the inputs according to which one should be used first (less expensive) to which should be used last (more expensive).

D2.4 In our experience, it is usually helpful to graphically summarize the final sequence in a standard split range block, as the one in Fig. 4 (and Fig. 8 in the case study), but note that the slopes and the split values have no significance when we use the generalized split range control structure that we propose.

3.4. Proposed baton strategy

Once that the sequence of inputs is defined, we can formulate the logic for the baton strategy. Consider that input $k$ is the active input (has the baton). The proposed baton strategy is then:

B.1 Controller $C_k$ computes $u'_k = v_k + u^0_k$, which is the suggested value for the input $k$.

B.2 If $u^\text{min}_k < u'_k < u^\text{max}_k$

(a) keep $u_k$ active, with $u_k \leftarrow u'_k$

(b) keep the remaining inactive inputs at their corresponding constant values ($u^\text{min}_i$ or $u^\text{max}_i$).

B.3 If $u'_k \leq u^\text{min}_k$ or $u'_k \geq u^\text{max}_k$

(a) Set $u_k = u^\text{min}_k$ or $u_k = u^\text{max}_k$ and pass the baton to the new active input $j$. The new active input is selected according to the predefined sequence, depending on which bound is met ($j = k + 1$ or $j = k - 1$).

(b) Set $k = j$ and go to step B.1. The value of the bias $u^0_k$ is the input value just before receiving the baton, that is, either $u^\text{max}_k$ or $u^\text{min}_k$. 

8
One needs to avoid windup for the inputs which are not active. Thus, when
switching, one needs to decide on how to initialize the new active controllers.
There are several alternatives (for example, bumpless transfer). Since we only
want to allow one controller to be active at a time, the simplest and most obvious
strategy is to set all the states of the non active controllers to zero. For a PI
controller (Eq. (1)), this means that the integral action starts at the time of
the switching ($t_k$).

$$u_k'(t) = u_k^0 + K_{C,k} \left( e(t) + \frac{1}{\tau_{I,k}} \int_{t_k}^{t} e(t) \right)$$  \hspace{1cm} (1)

Note that the integration in Eq. (1) starts from $t_k$ and not from 0. Alternative
and anti-windup implementations are described in the discussion section.

4. Case Study: Control of room temperature

In this section, we demonstrate the implementation and performance of our
proposed generalized split range control structure with a temperature control
case study. We compare our proposed generalized control structure with the
standard split range control described in Section 2.

We want to control room temperature ($y = T$) with four inputs ($u_i$), two
sources of cooling and two sources of heating:

- $u_{AC}$: air conditioning
- $u_{CW}$: cooling water
- $u_{HW}$: hot water (district heating)
- $u_{EH}$: electric heating.

The setpoint for the room temperature is $T^{sp} = 18^\circ C$. The main distur-
bance is ambient temperature ($d = T^{amb}$), which is nominally the same as the
setpoint; thus, $T_0^{amb} = 18^\circ C$. This means that no heating or cooling is required
at the nominal operating point ($u_i = 0 \ \forall i$). In this example, all four inputs are
scaled from 0 to 1.
4.1. Model

For simplicity, we model the room as a linear system:

\[ y(s) = G_p(s) u(s) + G_d(s) d(s) \]  \hspace{1cm} (2)

where:

\[
\begin{align*}
    y &= T \\
    u &= [u_{AC} \ u_{CW} \ u_{HW} \ u_{EH}]^T \\
    d &= T^{amb} \\
    G_p(s) &= [G_{AC}(s) \ G_{CW}(s) \ G_{HW}(s) \ G_{EH}(s)]
\end{align*}
\]

Table 1 shows the gains \( (K_{p,i}) \), time constants \( (\tau_i) \) and delays \( (\theta_i) \) for \( G_{p,i}(s) \) and \( G_d(s) \), modeled as first order transfer functions).

Table 1: Parameters for \( G_{p,i}(s) \) from \( u_i \) to \( y = T \) and \( G_d(s) \) from \( d = T^{amb} \) to \( y = T \).

<table>
<thead>
<tr>
<th>( G_i )</th>
<th>( K_{p,i} )</th>
<th>( \tau_i ) (min)</th>
<th>( \theta_i ) (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G_{AC} )</td>
<td>-5</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>( G_{CW} )</td>
<td>-10</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>( G_{HW} )</td>
<td>12</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>( G_{EH} )</td>
<td>8</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>( G_d )</td>
<td>1</td>
<td>15</td>
<td>6</td>
</tr>
</tbody>
</table>

Note that since the gain for the disturbance in ambient temperature is 1 and the inputs are scaled in the range 0 to 1, the gains \( K_{p,i} \) for the four inputs tell us the disturbance range that each input can handle. For example, since \( K_{p,HW} \) is 12 we can handle ambient temperatures down to \( T_{amb} = T_{0}^{amb} - K_{p,HW} = 18^\circ C - 12^\circ C = 6^\circ C \) before we must switch from hot water (HW) to electric heating (EH).

4.2. Standard implementation of split range control

Fig. 7 shows the block diagram for the standard implementation of split range control in this process, using one common PI controller \( (C) \) and the split
range block in Fig. 8. The common PI controller has $K_C = 0.0592$ and $\tau_I = 15$ min. Table A.1 in the Appendix summarizes the parameters for the standard split range block in Fig. 8. The details about the design and tuning of this control structure can be found in [17].

![Figure 7: Block diagram for standard split range control for room temperature control. The SR block is shown in Fig. 8.](image)

![Figure 8: Standard split range block for room temperature control with air conditioning (AC), cooling water (CW), hot water (HW), and electric heating (EH).](image)

### 4.3. Generalized implementation of split range control

Fig. 9 shows the block diagram for the proposed generalized split range control structure. We will use PI controllers for each input and tune each loop "tightly", according to the SIMC tuning rules [31]. This is achieved by selecting
the closed-loop time constant for each input equal to the time delay \( \tau_{c,i} = \theta_i \).

Table 2 gives the PI tuning parameters for each \( C_i(s) \).

<table>
<thead>
<tr>
<th>( u_i )</th>
<th>( \tau_{c,i}(\text{min}) )</th>
<th>( K_{C,i} )</th>
<th>( \tau_{I,i}(\text{min}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_{AC} )</td>
<td>( \theta_{AC} )</td>
<td>-0.4000</td>
<td>8</td>
</tr>
<tr>
<td>( u_{CW} )</td>
<td>( \theta_{CW} )</td>
<td>-0.2500</td>
<td>15</td>
</tr>
<tr>
<td>( u_{HW} )</td>
<td>( \theta_{HW} )</td>
<td>0.1389</td>
<td>10</td>
</tr>
<tr>
<td>( u_{EH} )</td>
<td>( \theta_{EH} )</td>
<td>0.3125</td>
<td>5</td>
</tr>
</tbody>
</table>

We design the generalized split range control structure according to the procedure in Section 3.

4.3.1. Sequencing of outputs

D1 The inputs are normalized, and the operating range for every input is \( u_i = [0, 1] \).

D2.1 The most economical operating point is when \( T^{\text{amb}} = T^{\text{sp}} \), and we can
have all inputs fully closed \((u_i = 0)\).

D2.2 To maintain \(T = T^{sp}\), we need to cool the room if \(T^{amb} > T^{sp}\), and to heat the room if \(T^{amb} < T^{sp}\). We can group the inputs according to their effect on the room temperature into:

- (a) Inputs for which \(y = T\) increases when we open them (move away from the desired operating condition, fully closed). These are the two heating sources: HW and EH.
- (b) Inputs for which \(y = T\) decreases when we open them (move away from the desired operating condition, fully closed). These are the two cooling sources: CW and AC.

D2.3 As CW is less expensive than AC, we prioritize the use of CW over AC for decreasing room temperature. Likewise, we prioritize the use of HW over EH.

D2.4 The final sequence can be summarized in the split range block in Fig. 8. However, note that when using the generalized control structure the values of the slopes \((\alpha_i)\) have no significance except for the sign.

4.3.2. Design of the baton strategy.

We consider the block diagram in Fig. 9 and use Fig. 8 to define the sequence and the choice of bias. The proposed baton strategy logic in steps B.1 to B.3 is written out in detail in Table 3.

When an input receives the baton, the integrator of its corresponding PI controller \((C_i(s))\) is reset to zero, according to Eq. (1). Thus, the initial value for \(u_i\) (at time \(t = t_k\)) will be the proportional term plus the bias

\[
u_k(t_k) = u_k^0 + K_{C_i,k}e(t_k)
\]

Note here that \(u_k^0\) is equal to \(u_k^{max}\) or \(u_k^{min}\), depending on from which side the baton is coming (see Table 3). Note than when \(u_1 = u_{AC}\) and \(u_4 = u_{EH}\)
Table 3: Baton strategy logic for case study.

<table>
<thead>
<tr>
<th>Value of $u_k$</th>
<th>$u_1 = u_{AC}$</th>
<th>$u_2 = u_{CW}$</th>
<th>$u_3 = u_{HW}$</th>
<th>$u_4 = u_{EH}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_k' \leq u_k^{min} = 0$</td>
<td>baton to $u_2$</td>
<td>baton to $u_3$</td>
<td>baton to $u_2$</td>
<td>baton to $u_3$</td>
</tr>
<tr>
<td>$u_0^1 = u_2^{max}$</td>
<td>$u_1^0 = u_3^{min}$</td>
<td>$u_2^0 = u_2^{min}$</td>
<td>$u_3^0 = u_3^{min}$</td>
<td></td>
</tr>
<tr>
<td>$u_k' \geq u_k^{max} = 1$</td>
<td>keep $u_4$ active</td>
<td>baton to $u_1$</td>
<td>baton to $u_4$</td>
<td>keep $u_4$ active</td>
</tr>
<tr>
<td>(max. cooling)</td>
<td>$u_1^0 = u_1^{min}$</td>
<td>$u_4^0 = u_4^{min}$</td>
<td>(max. heating)</td>
<td></td>
</tr>
<tr>
<td>$u_k^{min} &lt; u_k' &lt; u_k^{max}$</td>
<td>keep $u_1$ active</td>
<td>keep $u_2$ active</td>
<td>keep $u_3$ active</td>
<td>keep $u_4$ active</td>
</tr>
<tr>
<td>$u_1 \leftarrow u_1'$</td>
<td>$u_1 \leftarrow u_1^{min}$</td>
<td>$u_1 \leftarrow u_1^{min}$</td>
<td>$u_1 \leftarrow u_2^{min}$</td>
<td></td>
</tr>
<tr>
<td>$u_2 \leftarrow u_2^{max}$</td>
<td>$u_2 \leftarrow u_2'$</td>
<td>$u_2 \leftarrow u_2^{min}$</td>
<td>$u_2 \leftarrow u_4^{min}$</td>
<td></td>
</tr>
<tr>
<td>$u_3 \leftarrow u_3^{min}$</td>
<td>$u_3 \leftarrow u_3^{min}$</td>
<td>$u_3 \leftarrow u_3'$</td>
<td>$u_3 \leftarrow u_3^{max}$</td>
<td></td>
</tr>
<tr>
<td>$u_4 \leftarrow u_4^{min}$</td>
<td>$u_4 \leftarrow u_4^{min}$</td>
<td>$u_4 \leftarrow u_4^{min}$</td>
<td>$u_4 \leftarrow u_4'$</td>
<td></td>
</tr>
</tbody>
</table>

reach their corresponding $u_i^{max}$, we reach the limit of the range within which we can control $y = T$. As there is no other input to pass the baton, they remain the "active" input. In those cases, we loose control of $T$ because all inputs are constrained.

4.3.3. Simulations

The performance of this implementation is tested for rejection of disturbances in $T_{amb}$, which is nominally $18^\circ C$. $T^{sp}$ is kept constant at $18^\circ C$. At $t = 10\,\text{min}$, $T_{amb}$ increases to $20^\circ C$ and at $t = 80\,\text{min}$ to $29^\circ C$. Then, at $t = 140\,\text{min}$, $T_{amb}$ decreases to $24^\circ C$ and at $t = 180\,\text{min}$ to $-1^\circ C$. $T_{amb}$ then increases to $16^\circ C$ at $t = 280\,\text{min}$, and finally to $16^\circ C$ at $t = 350\,\text{min}$.

We observe that both the standard and the generalized implementation maintain $T = T^{sp}$ at steady-state, but the generalized structure reaches steady-state much faster, except in the initial period when CW (cooling water) is the active input. This is expected because the integral time for the common controller for standard split range control is $\tau_I = 15\,\text{min}$, which is the same as when we use cooling water with split range control (see Table 2). For the other inputs, the integral time for the generalized split range controller is smaller (8,
10, and 5 min), resulting in a faster return to the setpoint.

5. Discussion

5.1. Alternative implementations of generalized split range control

In standard split range control, we can use the slopes in the split range block to adjust the controller gain for each input, but we have to use the same value for the other controller settings, like the integral or derivative times. By “generalized split range control” we mean an implementation where the controllers for each input can be changed independently. Various statements on using independent controllers have appeared in the literature [32, 11, 12] but we did not find any details on how it should be implemented or whether it had been used in practice.

During the work with this project, we tried several alternative implementations. Our first attempt was to use a common integrator and put the dynamics after the split range block in Fig. 1. For example, to change the PI-tunings from the set 1 (inside the block $C$) to the set 2, we may add a block $K_{C,2}/K_{C,1}(1+1/\tau_2 s)/(1+1/\tau_1 s)$ on the signal $u_2$ exiting the split range block. However, the signal $u_2$ is already a physical signal, which already includes its maximum or minimum value, and adding dynamics to the signal creates non-uniqueness in the switching.

Our next attempt was to have one controller $C(s)$, as in Fig. 1, and use different sets of parameters in $C(s)$ based on the output from the split range block, which tells which input is active. Åström and Wittenmark [32] and Hägglund [11] refer to this idea as a special type of gain scheduling. However, the term gain scheduling is generally used for the case where the inputs and outputs are fixed and we change the controller parameters depending on the operating parameters, for example, the setpoint ($y^{sp}$) or the disturbance ($d$). On the other hand, split range control is used to extended the steady state range of $y$ by using a sequence of different inputs. In any case, we encountered problems with cycling in our implementation of this approach. This is because
when we change the controllers parameters for $C(s)$, the signal $v$ from $C(s)$ changes, which may cause the selector block to change the active input.

We therefore decided to remove the common split range block from the implementation and use independent controllers. However, only one controller must be active at the time, and to select which one, we propose the baton strategy. The baton strategy has the advantage that the selection of the active input is not centralized. Each active controller only needs to know which two controllers it can give the baton to if it reaches its maximum or minimum value, respectively.

5.2. Comparison with multiple controllers with different setpoints

As mentioned in the introduction, an alternative to split range control is to use multiple controllers with different setpoints. In this case, all controllers are active at any given time (although some inputs may be saturated), so to avoid undesired switching and fighting, one has to separate the setpoints.

Our new generalized split range controller may be viewed as an extension of this, which avoids the use of different setpoints (Fig. 2). We avoid undesired switching by using the baton strategy. This requires a logic block. The use of different setpoints has the advantage of avoiding the logic block, as the sequence of the inputs is indirectly given by the value of the setpoints. For example, for our room temperature case study, we could have used four controllers with setpoints $20^\circ C$ for AC, $19^\circ C$ for CW, $18^\circ C$ for HW and $17^\circ C$ for EH.

5.3. Anti-windup for generalized split range control

In the proposed generalized structure for split range control there are multiple controllers for the same output. In section 3.4 and in the case study, windup is overcome by letting only one controller to be active at a time and resetting the integrator term to zero when a controller becomes active (see Eq. (1)). The proportional and derivative terms of the controller may potentially cause large output changes when the switch occurs. This may be partly seen by the line for
$u_{AC}$ in Fig. 10 at $t = 100$ min. Thus, we do not have bumpless transfer, which may be an advantage because it may give a faster response (see below).

Windup can be avoided by implementing other anti-windup schemes, such as input tracking with back-calculation [14]. Fig. 11 shows how input tracking with back-calculation can be implemented for one input ($u_i$) with the generalized split range control structure. In the block diagram in Fig. 11, time constant $\tau_T$ is sometimes known as tracking coefficient [33] and allows to reset the integrator dynamically [14]. If we implement this anti-windup scheme in combination with the generalized split range controller proposed in this paper, all inputs ($u_i$) are calculated at every time, and we do not reset the integrator of the input that becomes active (receives the baton). Otherwise, the switching logic to transfer the baton remains the same.

As an example, we implemented the back-calculation tracking scheme in Fig. 11, with tracking constant $\tau_T = 1$ for all inputs for the case study analyzed in Section 4. We use the same tight tunings as in Table 2 and the switching logic in Table 3. Fig. 12 compares back-calculation (dashed lines) with the strategy of resetting the integrator (solid lines), using the same disturbances described in Section 4.3.3. As expected, we observe a less aggressive initial response when we use back-calculation. For example, if we reset the integrator when one input becomes active, $y = T$ overshoots when $d = T_{amb}$ decreases from $24^\circ C$ to $-1^\circ C$ at $t = 180$ min. This does not happen with back-calculation because electric heating ($u_{EH}$) starts closing before than when we reset the integrator. The integral of the absolute error (IAE) for the simulations in Fig. 12 is higher with back-calculation, but still lower than when using standard split range control (see Table A.2 in the Appendix).

5.4. Active input-constraint changes using split range control

The input saturation pairing rule [34] suggests to pair high priority controlled variables (outputs) with manipulated variables (inputs) that are not likely to saturate (reach an input-constraint). When this is not possible, we need to implement a split range control scheme together with a min/max selector, such
that when the primary input saturates we start using another input to keep control of the high priority output (and give up controlling a low priority output). For a more detailed discussion of this application of split range control, the reader is referred to [35]. In these case, the bias of the input affected by the selector can be updated in the split range control logic block to improve the dynamic response.

6. Conclusions

Split range control is extensively used in industry, but it has not been studied much in academia. In this work, we introduce a new generalized control structure using a baton strategy that allows for using individual controllers for each available input, without a centralized supervisor. We demonstrated the feasibility of implementing this structure in a case study with four available inputs and one controlled variable. This new generalized structure has a better dynamic performance than the standard split range controller. The proposed baton strategy is illustrated in Fig. 5 and Table 3.

Conflicts of interest

The authors declare no conflicts of interest.

Acknowledgments

The authors acknowledge Cristina Zotică for her insights for this paper.

Appendix: Additional information for Case Study

Parameters for standard split range controller for room temperature control

Table A.1 summarizes the information that describes the standard split range block in Fig. 8, where $u_0^i$ corresponds to the bias, the slopes are $\alpha_i$ and $\Delta v_i$ is the range of the internal variable for each input.
Table A.1: Values for $\alpha_i$, $\Delta v_i$ and $u_i^0$.

<table>
<thead>
<tr>
<th></th>
<th>AC</th>
<th>CW</th>
<th>HW</th>
<th>EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_i$</td>
<td>-6.7600</td>
<td>-4.2250</td>
<td>2.3472</td>
<td>5.2813</td>
</tr>
<tr>
<td>$\Delta v_i^0$</td>
<td>0.1479</td>
<td>0.2367</td>
<td>0.4260</td>
<td>0.1893</td>
</tr>
<tr>
<td>$u_i^0$</td>
<td>1.0000</td>
<td>1.6250</td>
<td>-0.9028</td>
<td>-4.2813</td>
</tr>
</tbody>
</table>

Comparison of integrated absolute error (IAE) for room temperature control

Due to the improved dynamic response, the integrated absolute error (IAE) is reduced from with the proposed generalized structure. Table A.2 shows the values for IAE for the simulations in Figs. 10 and 12.

Table A.2: Comparison of integral absolute error (IAE) with standard and generalized split range control for room temperature.

<table>
<thead>
<tr>
<th>Case</th>
<th>IAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard split range control</td>
<td>448.6</td>
</tr>
<tr>
<td>Generalized split range control, with integrator resetting (Eq. (1))</td>
<td>202.4</td>
</tr>
<tr>
<td>Generalized split range control, with back-calculation tracking</td>
<td>235.7</td>
</tr>
</tbody>
</table>

References


Figure 10: Comparison of generalized and standard split range controller (SRC) for room temperature.

Ambient temperature ($d = T_{amb}$)

Room temperature ($y = T$)

Manipulated variables for generalized SRC

Figure 10: Comparison of generalized and standard split range controller (SRC) for room temperature.
Figure 11: Antiwindup with bumpless transfer: input tracking with back-calculation for input $u_i$. 
Figure 12: Comparison of anti-windup strategies in generalized split range control for room temperature. The dashed lines correspond to back-calculation and the solid lines correspond to the strategy of resetting the integrator (Eq. (1)), which is also depicted in Fig. 10.