A Linear Programming Approach to Computing Safe Sets for Software Rejuvenation

T. Arauz[1], J. M. Maestre[1], R. Romagnoli[2], B. Sinopoli[3], E. F. Camacho[1]

***Abstract*— Software rejuvenation was born to fix operating system faults by periodically refreshing the run-time code and data. This mechanism has been extended to protect control systems from cyber-attacks. This work proposes a software rejuvenation design method in discrete-time where invariant sets for the safety and mission controllers are designed to schedule the timing of software refreshes. To compute a *minimal robust positively invariant* (min-RPI) set and the bounded time between software refreshes to ensure system safety, an LP based approach is proposed for stable and unstable systems. Finally, the designed approach is illustrated by the case study of a simulated lab-scale microgrid.**

***Index Terms*— Cybersecurity, Software Rejuvenation, Linear Systems, Invariant Sets, Linear Quadratic Regulator.**

I. Introduction

MALICIOUS cyber-attacks such as *Stuxnet* [1] and *Crash Override* [2] are increasing in number and complexity arising the awareness about their types, modes of action and severe consequences for cyber-physical systems (CPSs) [3]. Different works seek solutions to guarantee CPS safety [4], [5], most of them based on attack detection and modelling [6], [7]. However, attacks can be effectuated in different ways, precluding the modeling and detection of all possibilities.

A powerful strategy to guarantee CPS stability when dealing with unmodeled and undetectable cyber-attacks that threaten the run-time code and data is software rejuvenation [8], [9], which periodically refreshes the run-time control software with a trusted, secure copy to thwart attacks that may have changed the on-line code [10]. Software rejuvenation was initially developed in [11] to address the software aging problem by periodically resetting the software to a secure version to avoid failures caused by non-anticipated states. This idea has been developing ever since [12], e.g., by establishing the software refresh frequency based on the time a system remains safe in case of being cyber-attacked [13] or the time that an attack can be effective to make the system safe against persistent attacks [8]. Since frequent reboots can degrade control performance, the authors in [9] propose an online computation of the time to reboot that also takes into account the consequences of a possible attack. Also, they introduce the *hardware root of trust*, which is a secure onboard module that hosts the necessities for implementing software rejuvenation; the *secure execution interval*, representing the period where external communications are disabled; and the *safety controller*, which is executed after the software refresh. Likewise, another relevant topic in the literature is that of the computation of safe sets, e.g., a safe robust invariant terminal set is defined along with a maximum safe initial set in [14]. Due to the computational complexity of polytope operators [15], [16], these sets typically adopt simple representations, e.g., zonotopes [14] and ellipsoids [10]. Remarkably, this challenge is somewhat common with recent works in the area of learning-based control [17], [18].

This article represents a discrete-time extension of the algorithm presented in [10], which employs linear matrix inequalities to derive a constant time interval for scheduling the software refresh using ellipsoidal invariant sets and off-line reachability computations. Here, software refreshes are defined for systems with real eigenvalues by the off-line computation of two state sets: the *safe set*, reserved for the safety controller, and the *inner safe set* (a subset of the *safe set*), for the mission controller, which is responsible for the CPS operation. However, the way of defining these sets differs because the *inner safe set* is defined as the smallest robust positively invariant (RPI) of a pre-selected *shape* following the linear program (LP) given in [19], which is extended to compute the time schedule for software refreshes and maximize the volume difference with the *safe set* so as to decrease the software refresh frequency. Also, being an LP-based approach, polytopic sets can be kept without significant computational issues for larger systems.

Outline. Section II presents the software rejuvenation strategy and Section III introduces the safe sets and the LP-based approaches for computing the *inner safe set* and the time between software refreshes. Section IV presents the application of the strategy to a case study and discusses results. Finally, concluding remarks are given in Section V.

Notation. The set of natural numbers is $\mathbb{N}$. The set of non-negative reals is $\mathbb{R}_{0+}$. $\lambda\mathcal{X}$ is the scaling of a set $\mathcal{X} \subset \mathbb{R}^n$

This work was supported in part by the European Research Council Advanced Research Grant 769051-OCONTSOLAR, the project GESVIP funded by Junta de Andalucía (ref. US-1265917), and the Spanish Training Program for Academic Staff (FPU19/00127).

[1]T. Arauz, J. M. Maestre and E. F. Camacho are with Department of Ingeniería de Sistemas y Automática, Universidad de Sevilla, Camino de los Descubrimientos, 41092 Sevilla, Spain (e-mail: marauz@us.es, pepemaestre@us.es, efcamacho@us.es).

[2]R. Romagnoli is with the Dept. of Electrical and Computer Engineering, Carnegie Mellon University (CMU), Pittsburgh, PA, USA 15235 (e-mail: rromagno@andrew.cmu.edu).

[3]B. Sinopoli is with the Department of Electrical and Systems Engineering, Washington University, St. Louis, MO, USA 63130 (e-mail: bsinopoli@wustl.edu).

by $\lambda \in \mathbb{R}$, defined as $\{\lambda x \mid x \in \mathcal{X}\}$. $A\mathcal{X}$ represents the image of a set $\mathcal{X} \subset \mathbb{R}^n$ under the linear map $A : \mathbb{R}^n \to \mathbb{R}^p$ given by $\{Ax \mid x \in \mathcal{X}\}$. The spectral radius of a matrix A is ρ_A, which corresponds to its dominant eigenvalue. The support function of a set $\mathcal{X}$ according to vector v is $h_{\mathcal{X}}(v) \triangleq \sup\{v^\top x \mid x \in \mathcal{X}\}$. For $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$, the Minkowski sum is $\mathcal{X} \oplus \mathcal{Y} \triangleq \{x + y \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$. $\mathcal{R}(x_0, k; CP)$ represents the set of reachable states at time step $k > 0$ from an initial state $x(0) = x_0$ under policy CP, with the natural extension to an initial state set $\mathbb{X}_0$.

II. Software Rejuvenation Control Strategy

The CPS is modeled as a discrete-time LTI system:

$$x(k+1) = Ax(k) + Bu(k) + Dw(k), \tag{1}$$

where $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$ and $w \in \mathbb{R}^{n_w}$ are the states, inputs and external disturbances of the system, respectively. Hence, $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$ and $D \in \mathbb{R}^{n_x \times n_w}$.

Assumption 1: The eigenvalues of matrix A are real.

Assumption 2: The system state x is measurable and the pair (A, B) is controllable.

Assumption 3: The system is subject to polytopic constraints containing the origin in their interiors, i.e.,

$$x \in \mathbb{X} \triangleq \{x \in \mathbb{R}^{n_x} \mid C^x x \leq a\}, \tag{2}$$

$$u \in \mathbb{U} \triangleq \{u \in \mathbb{R}^{n_u} \mid C^u u \leq b\}, \tag{3}$$

where $C^x \in \mathbb{R}^{r_x \times n_x}$, $C^u \in \mathbb{R}^{r_u \times n_u}$, $a \in \mathbb{R}^{r_x}_{0+}$ and $b \in \mathbb{R}^{r_u}_{0+}$.

Assumption 4: External disturbances are assumed to lie in a polytopic (compact and convex) set $\mathbb{W}$ that also contains the origin in its interior:

$$\mathbb{W} \triangleq \{w \in \mathbb{R}^{n_w} \mid C^w w \leq g\}, \tag{4}$$

where $C^w \in \mathbb{R}^{r_w \times n_w}$ and $g \in \mathbb{R}^{r_w}_{0+}$.

Software rejuvenation divides the operation in three modes (Fig. 1): *mission control* (MC), *software refresh* (SR), and *safety control* (SC), which take place during T_{SR}, T_{MC} and T_{SC} time steps, respectively. Likewise, Figure 2 illustrates the scenarios for which safe operation during the mission of the CPS needs to be guaranteed.

A. Mission control mode

The system exchanges information through the network during the MC mode generating a source of vulnerability that can be exploited to hijack or disrupt the control signal.

Assumption 5: The consequence of an attack is that $u(k) = u_a(k)$, where $u_a(k)$ is uncertain but bounded by the limits of the mission control operation, i.e., $u_a(k) \in \mathbb{U}_{\mathrm{UC}}$, which is defined at the end of this section.

During MC, the system performs its mission requiring external communications, being exposed to cyber-attacks, which may steer the system away from its goal, which is to minimize the following stage cost over an infinite horizon:

$$\ell(x(k), u(k)) = x(k)^\top Q_{\mathrm{MC}} x(k) + u(k)^\top R_{\mathrm{MC}} u(k), \tag{5}$$

where $Q_{\mathrm{MC}} \in \mathbb{R}^{n_x \times n_x}$ and $R_{\mathrm{MC}} \in \mathbb{R}^{n_u \times n_u}$ are positive-definite weighting matrices.

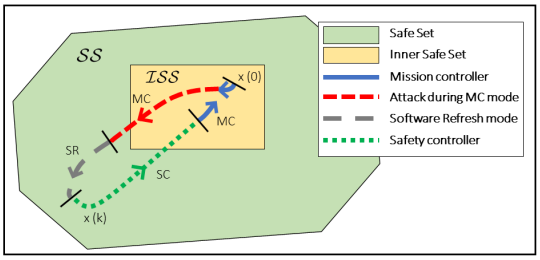

Fig. 1: *Safe set* (green) and *inner safe set* (yellow). The system starts in MC mode with open communications (continuous blue line) and then is captured by an attacker (dashed red line). Before the state leaves the *safe set*, SR is performed and communications are switched off keeping the last control input of MC (spaced dashed grey line). After SR finishes, the safety controllers starts and drives the state into the *inner safe set* (dotted green line), from where the cycle starts over.

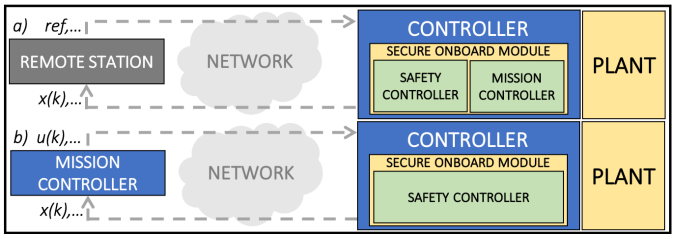

Fig. 2: Considered set-up: a) On-board mission controller connected to a remote station; b) Remote mission controller.

This mode ends when the SR mode is triggered by a refresh clock located within the safe module, i.e., at the *hardware root of trust*, which cannot be compromised.

Assumption 6: The control law of the MC is

$$u(k) = -K_{\mathrm{MC}} x(k), \tag{6}$$

where K_{MC} is the LQR controller, with $K_{\mathrm{MC}}\mathbb{X} \subseteq \mathbb{U}$.

Then, the closed-loop system dynamics under safety control are stable and given by

$$x(k+1) = A_{\mathrm{MC}} x(k) + Dw(k), \tag{7}$$

with $A_{\mathrm{MC}} \triangleq (A - BK_{\mathrm{MC}})$. Finally, the set of admissible states for the MC mode is composed of all states that comply with both state and input constraints: $\mathbb{X}_{\mathrm{MC}} \triangleq \mathbb{X} \cap \mathbb{X}^{\mathcal{U}}_{\mathrm{MC}}$, where $\mathbb{X}^{\mathcal{U}}_{\mathrm{MC}}$ represents the mapping of input constraints during the MC mode into the state space, i.e., $\mathbb{X}^{\mathcal{U}}_{\mathrm{MC}} \triangleq \{x \in \mathbb{R}^{n_x} \mid C^u K_{\mathrm{MC}} x \leq b_{\mathrm{UC}}\}$, where $b_{\mathrm{UC}} = \alpha_{\mathrm{UC}} b$ with the scaling factor $\alpha_{\mathrm{UC}} \in (0, 1)$.

The period of *uncertain control* (UC), T_{UC}, represents the time steps that an attack can be tolerated, which can be increased reducing the maximum input allowed during this period by a scaling factor $\alpha_{\mathrm{UC}} \in (0, 1)$, mitigating the consequences of attacks, i.e., $\mathbb{U}_{\mathrm{UC}} = \alpha_{\mathrm{UC}}\mathbb{U} = \{x \in \mathbb{R}^{n_x} \mid C^u x \leq b_{\mathrm{UC}}\}$, where $b_{\mathrm{UC}} = \alpha_{\mathrm{UC}} b$. Note that parameter α_{UC} and its related functionality are in the secure onboard module.

The UC period comprises software refresh and mission control modes ($T_{\mathrm{UC}} = T_{\mathrm{MC}} + T_{\mathrm{SR}}$), where an attacker may

have gained control. Typically, T_{SR} is a fixed value initially set by the operation characteristics, but T_{UC} is calculated to ensure safety. Notice that T_{UC} must be greater than T_{SR}, since T_{MC} has to be greater than zero. Otherwise, the problem turns infeasible.

Remark 1: The MC controller can be given by any method that guarantees: *i)* $x(k) \in \mathcal{ISS}$, which is defined in Section II.D, and *ii)* $u(k) \in \mathbb{U}_{\mathrm{UC}} = \alpha_{\mathrm{UC}}\mathbb{U}$ for $0 \leq k \leq T_{\mathrm{MC}}$.

B. Software refresh mode

SR takes place when the operating software is restored, eliminating possible modifications from cyber-attacks so that the system recovers its initial safe configuration. During this time, external communications are switched off and the actuators maintain the last control input provided during MC, which may be corrupted in case of attack. Therefore, it must be carefully designed to ensure system safety.

C. Safety control mode

This mode is activated before transferring control back to the mission controller in case the state is out of the safe set during the software refresh. The safety controller steers the state of the system back to the safe set while communications are still switched off. While this mode is active, a different goal can be considered for safety reasons, i.e., during SC the controller minimizes the stage cost

$$\ell(x(k), u(k)) = x(k)^\top Q_{\mathrm{SC}} x(k) + u(k)^\top R_{\mathrm{SC}} u(k), \tag{8}$$

over an infinite horizon, where $Q_{\mathrm{SC}} \in \mathbb{R}^{n_x \times n_x}$ and $R_{\mathrm{SC}} \in \mathbb{R}^{n_u \times n_u}$ are positive-definite weighting matrices.

If Assumption 2 holds, the control law becomes

$$u(k) = -K_{\mathrm{SC}} x(k), \tag{9}$$

where K_{SC} is the corresponding discrete-time LQR controller. It is also assumed that $K_{\mathrm{SC}}\mathbb{X} \subseteq \mathbb{U}$. Then, the closed-loop system dynamics under safety control are stable and given by

$$x(k+1) = A_{\mathrm{SC}} x(k) + Dw(k), \tag{10}$$

with $A_{\mathrm{SC}} \triangleq (A - BK_{\mathrm{SC}})$. Therefore, the set of admissible states for SC mode is composed of all states that comply with both state and input constraints: $\mathbb{X}_{\mathrm{SC}} \triangleq \mathbb{X} \cap \mathbb{X}^{\mathcal{U}}_{\mathrm{SC}}$, where $\mathbb{X}^{\mathcal{U}}_{\mathrm{SC}}$ represents the input constraint referred to states for this mode, i.e., $\mathbb{X}^{\mathcal{U}}_{\mathrm{SC}} \triangleq \{x \in \mathbb{R}^{n_x} \mid C^u K_{\mathrm{SC}} x \leq b\}$.

D. System safety sets

Safety conditions are expressed by defining two sets regarding different controllers: the *safe set* and the *inner safe set*. They guarantee that the given constraint sets in the system state space are not violated in any way. Fig. 1 illustrates these sets and the evolution of the system states in all the different operation modes in case of attack.

1) The *safe set* ($\mathcal{SS}$) is the maximum set containing all admissible system states complying with state and input constraints during the SC mode, i.e.,

$$\mathcal{SS} = \mathbb{X}_{\mathrm{SC}} = \{x \in \mathbb{R}^{n_x} \mid P_{\mathrm{SS}} x \leq q_{\mathrm{SS}}\}, \tag{11}$$

where $P_{\mathrm{SS}} \in \mathbb{R}^{r_{\mathrm{SS}} \times n_x}$ and $q_{\mathrm{SS}} \in \mathbb{R}^{r_{\mathrm{SS}}}_{0+}$.

2) The *inner safe set* ($\mathcal{ISS}$) is the subset of the *safe set*, $\mathcal{ISS} \subseteq \mathcal{SS}$ containing all admissible states during the MC mode, i.e.,

$$\mathcal{ISS} = \{x \in \mathbb{R}^{n_x} \mid P_{\mathrm{ISS}} x \leq q_{\mathrm{ISS}}\}, \tag{12}$$

where $P_{\mathrm{ISS}} \in \mathbb{R}^{r_{\mathrm{ISS}} \times n_x}$ and $q_{\mathrm{ISS}} \in \mathbb{R}^{r_{\mathrm{ISS}}}_{0+}$. This set must ensure the following safety conditions:

(13a) The attacker is not able to take the system out of the *safe set*, i.e., $\mathcal{R}(\mathcal{ISS}, k; \mathrm{UC}) \subseteq \mathcal{SS}$, $\forall 0 \leq k \leq T_{\mathrm{UC}}$.

(13b) The safety controller is able to return the system from the *safe set* to this set in a bounded time, i.e., $\mathcal{R}(\mathcal{SS}, \overline{T}_{\mathrm{SC}}; \mathrm{SC}) \subseteq \mathcal{ISS}$.

(13c) The system stays in $\mathcal{ISS}$ during MC, i.e., $\mathcal{R}(\mathcal{ISS}, k; \mathrm{MC}) \subseteq \mathcal{ISS}$, $\forall 0 \leq k \leq T_{\mathrm{MC}}$.

(13d) $\mathcal{ISS} \subseteq \mathbb{X}_{\mathrm{MC}}$.

The *inner safe set* has to comply with all these requirements considering that the attack might try to drive the system to unsafe states ($x \notin \mathcal{SS}$) as fast as possible. Therefore, the $\mathcal{ISS}$ computation requires considering the UC period and the system dynamics under attack, which become

$$x(k+1) = Ax(k) + Bu_a(k) + Dw(k), \tag{14}$$

where u_a represents the attacker input signal: $u_a \in \mathbb{U}_{\mathrm{UC}} = \alpha_{\mathrm{UC}}\mathbb{U}$ with $\alpha_{\mathrm{UC}} \in (0, 1)$. Note that $x(k) \in \mathcal{SS}$, $\forall k$, has to be ensured. Likewise, $T_{\mathrm{SR}} < T_{\mathrm{UC}}$.

In this study, we define $\mathcal{ISS}$ as the minimal RPI set that satisfies the safety conditions (13a)-(13d). Furthermore, the advantage to use a *minimal* RPI set is the higher difference between the $\mathcal{SS}$ and $\mathcal{ISS}$ sets, so the T_{UC} can be maximized and the software refresh frequency minimized.

III. Computation of Safe Sets

This section presents our proposal to obtain $\mathcal{ISS}$ and T_{UC} based on [19]: for stable systems, a single LP computes both $\mathcal{ISS}$ and T_{UC}; for unstable systems, an iterative LP-based process is required. The computation of T_{SC} according to the corresponding safety sets is also presented.

A. Computation of $\mathcal{ISS}$ as a min-RPI set and T_{UC}

The approach of [19] assumes that the *shape* of the RPI set is arbitrarily defined a priori, i.e., r_{ISS} and matrix P_{ISS} of Eq. (12). Here, the shape of $\mathcal{ISS}$ is defined for convenience with the rows of P_{ISS} defined as the n_v eigenvectors of matrix A_{MC} and their opposites, i.e., $P^\top_{\mathrm{ISS}} = \{v^1_{A_{\mathrm{MC}}}, \ldots, v^{n_v}_{A_{\mathrm{MC}}}, -v^1_{A_{\mathrm{MC}}}, \ldots, -v^{n_v}_{A_{\mathrm{MC}}}\}$, where $v^i_{A_{\mathrm{MC}}}$ represents the i-th eigenvector of matrix A_{MC}.

Assumption 7: All eigenvectors of matrix A_{MC} are real.

Thus, $r_{\mathrm{ISS}} = 2n_v$. Note that the scaled factor α_{UC} have to be defined to ensure $\mathcal{ISS} \subseteq \mathbb{X}_{\mathrm{MC}}$, due to constraints are not considered in [19].

Furthermore, the maximum T_{UC} allowed is the value that ensures the safety constraint:

$$\mathcal{R}(\mathcal{ISS}, T_{\mathrm{UC}}; \mathrm{UC}) \subseteq \mathcal{SS}, \tag{15}$$

i.e., the attached system (Eq. (14)) must remain within set $\mathcal{SS}$ $T_{\rm UC}$ steps after departing from set $\mathcal{ISS}$. Therefore, the system state x is guaranteed to lie in $\mathcal{SS}$ even for the worst attack scenario as long as T_{UC} is computed to comply with constraint (15). To compute the set $\mathcal{T}_k \triangleq \mathcal{R}(\mathcal{ISS}, k; \mathrm{UC})$ at time step k, the following recursion holds:

$$\mathcal{T}_k = A\mathcal{T}_{k-1} \oplus B\mathbb{U}_{\rm UC} \oplus D\mathbb{W}, \tag{16}$$

with $\mathcal{T}_0 = \mathcal{ISS}$, which can be rewritten as

$$\mathcal{T}_k = A^k\mathcal{ISS} \oplus \left[\bigoplus_{j=0}^{k-1} A^j B\mathbb{U}_{\rm UC} \oplus A^j D\mathbb{W}\right]. \tag{17}$$

Taking support functions according to the directions of vectors $P_{\mathrm{SS}_i}^\top$, $\forall i = 1, \ldots, r_{\rm SS}$, equality (17) is equivalent to the support functions inequality:

$$\begin{aligned} h_{\mathcal{T}_k}(P_{\mathrm{SS}_i}^\top) &\leq h_{A^k\mathcal{ISS}}(P_{\mathrm{SS}_i}^\top) \\ &\quad + \textstyle\sum_{j=0}^{k-1}\left(h_{A^jB\mathbb{U}_{\rm UC}}(P_{\mathrm{SS}_i}^\top) + h_{A^jD\mathbb{W}}(P_{\mathrm{SS}_i}^\top)\right) \\ &\leq \rho_A^k h_{\mathcal{ISS}}(P_{\mathrm{SS}_i}^\top) \\ &\quad + \textstyle\sum_{j=0}^{k-1}\rho_A^j\left(h_{B\mathbb{U}_{\rm UC}}(P_{\mathrm{SS}_i}^\top) + h_{D\mathbb{W}}(P_{\mathrm{SS}_i}^\top)\right). \end{aligned} \tag{18}$$

By definition, $h_{A\mathcal{R}}(z) = h_{\mathcal{R}}(A^\top z)$, so the support of a mapping satisfies $A^k\mathcal{R} \subseteq \rho_A^k\mathcal{R}$ and $h_{\mathcal{R}}(A^\top z) \leq \rho_A h_{\mathcal{R}}(z)$.

Recall (15) and let $\mathcal{T}_{T_{\rm UC}} \triangleq \mathcal{R}(\mathcal{ISS}, T_{\rm UC}; \mathrm{UC})$. Taking support functions, we have:

$$h_{\mathcal{T}_{T_{\rm UC}}}(P_{\mathrm{SS}_i}^\top) \leq q_{\mathrm{SS}_i}, \forall i = 1, \ldots, r_{\rm SS}, \tag{19}$$

where P_{SS_i} represents the i-th row of matrix $P_{\rm SS}$. The left-hand side of (19) can be upper bounded using the inequality (18), providing a means to compute T_{UC}. In particular, considering the geometric series $\sum_{j=0}^{k-1}\rho_A^j$, the corresponding constraint becomes:

- If $\rho_A \neq 1$, $\forall i = 1, \ldots, r_{\rm SS}$:

$$\rho_A^{T_{\rm UC}} h_{\mathcal{ISS}_i} + \frac{1-\rho_A^{T_{\rm UC}}}{1-\rho_A}\left(h_{B\mathbb{U}_{uc_i}} + h_{D\mathbb{W}_i}\right) \leq q_{\mathrm{SS}_i} \tag{20}$$

- If $\rho_A = 1$, $\forall i = 1, \ldots, r_{\rm SS}$:

$$h_{\mathcal{ISS}_i} + T_{\rm UC}\left(h_{B\mathbb{U}_{uc_i}} + h_{D\mathbb{W}_i}\right) \leq q_{\mathrm{SS}_i} \tag{21}$$

where $h_{\mathcal{ISS}_i} \triangleq h_{\mathcal{ISS}}(P_{\mathrm{SS}_i}^\top)$, $h_{B\mathbb{U}_{uc_i}} \triangleq h_{B\mathbb{U}_{\rm UC}}(P_{\mathrm{SS}_i}^\top)$ and $h_{D\mathbb{W}_i} \triangleq h_{D\mathbb{W}}(P_{\mathrm{SS}_i}^\top)$. Note that $h_{B\mathbb{U}_{uc_i}}$ can be replaced by $\alpha_{UC}h_{B\mathbb{U}_i}$, leading to a explicit link between α_{UC} and T_{UC}.

From this point, the rest of the approach changes depending on the value of ρ_A and three cases can be considered.

1) Stable systems where $\rho_A = 1$: Constraint (21) preserves linearity with respect to $T_{\rm UC}$ and the following LP computes simultaneously $\mathcal{ISS}$ as a min-RPI set and an admissible $T_{\rm UC}$:

$$\begin{aligned} \mathbb{P}: \max_{\substack{\{T_{\rm UC}, c_i, d_i, \xi^i, \nu^i, h_{\mathcal{ISS}_j}, \\ h_{B\mathbb{U}_{uc_j}}, h_{D\mathbb{W}_j}, \psi_{\rm ISS}^j, \psi_{B\mathbb{U}_{uc}}^j, \psi_{\mathbb{W}}^j\} \\ \forall i \in \{1,\ldots,r_{\rm ISS}\}, j \in \{1,\ldots,r_{\rm SS}\}}} & \beta_1 \sum_{i=1}^{r_{\rm ISS}}(c_i + d_i) \\ + \beta_2 \sum_{j=1}^{r_{\rm SS}}(h_{\mathcal{ISS}_j} + h_{B\mathbb{U}_{uc_j}} + h_{\mathbb{W}_j}) & + \beta_3 T_{\rm UC} \end{aligned} \tag{22}$$

subject to, for all $i \in \{1, \ldots, r_{\rm ISS}\}$, $j \in \{1, \ldots, r_{\rm SS}\}$

$$c_i \leq P_{\mathrm{ISS}_i} A_{\rm MC}\xi^i \tag{23a}$$
$$P_{\rm ISS}\xi^i \leq c + d \tag{23b}$$
$$d_i \leq P_{\mathrm{ISS}_i}\nu^i \tag{23c}$$
$$P_{D\mathbb{W}}\nu^i \leq q_{D\mathbb{W}} \tag{23d}$$
$$h_{\mathcal{ISS}_j} \leq P_{\mathrm{SS}_j}\psi_{\rm ISS}^j \tag{23e}$$
$$P_{\rm ISS}\psi_{\rm ISS}^j \leq c + d \tag{23f}$$
$$h_{B\mathbb{U}_{uc_j}} \leq P_{\mathrm{SS}_j}\psi_{B\mathbb{U}_{uc}}^j \tag{22g}$$
$$P_{B\mathbb{U}_{uc}}\psi_{B\mathbb{U}_{uc}}^j \leq q_{B\mathbb{U}_{uc}} \tag{22h}$$
$$h_{D\mathbb{W}_j} \leq P_{\mathrm{SS}_j}\psi_{D\mathbb{W}}^j \tag{22i}$$
$$P_{D\mathbb{W}}\psi_{D\mathbb{W}}^j \leq q_{D\mathbb{W}} \tag{22j}$$
$$h_{\mathcal{ISS}_j} + T_{\rm UC}(h_{B\mathbb{U}_{uc_j}} + h_{D\mathbb{W}_j}) \leq q_{\mathrm{SS}_j} \tag{22k}$$
$$T_{\rm UC} \in \mathbb{N} \tag{22l}$$

where $\beta_1 \gg \beta_2 \gg \beta_3 > 0$ are tuning parameters with a difference of at least one order of magnitude for computing the three objectives at once, namely, i) computing the min-RPI for $\mathcal{ISS}$; ii) computing the support functions of sets $\mathcal{ISS}$, $B\mathbb{U}_{UC}$ and $D\mathbb{W}$; and iii) computing the maximum value allowed for $T_{\rm UC}$; $\xi^i \in \mathbb{R}^{n_x}$, $\nu^i \in \mathbb{R}^{n_x}$, $\psi_{\rm ISS}^j \in \mathbb{R}^{n_x}$, $\psi_{B\mathbb{U}_{uc}}^j \in \mathbb{R}^{n_x}$ and $\psi_{\mathbb{W}}^j \in \mathbb{R}^{n_x}$ are some auxiliary variables; and mapped sets of inputs and disturbances are defined as $B\mathbb{U}_{UC} = \{x \in \mathbb{R}^{n_x} \mid P_{B\mathbb{U}_{uc}}x \leq q_{B\mathbb{U}_{uc}}\}$ and $D\mathbb{W} = \{x \in \mathbb{R}^{n_x} \mid P_{D\mathbb{W}}x \leq q_{D\mathbb{W}}\}$.

In this problem, the maximization of the first objective subject to constraints (23a) to (23d) to find $\mathcal{ISS}$ corresponds to the LP problem of [19]. It comes from the RPI condition $A_{\rm MC}x(k) + Dw(k) \in \mathcal{ISS}$, $\forall x \in \mathcal{ISS}, w \in \mathbb{W}$, which is equivalent to $c(q_{\rm ISS}) + d \leq b(q_{\rm ISS})$, where support functions have been taken: $b_i(q_{\rm ISS}) \triangleq h_{\mathcal{ISS}}(P_{\mathrm{ISS}_i}^\top)$, $c_i(q_{\rm ISS}) \triangleq h_{A\mathcal{ISS}}(P_{\mathrm{ISS}_i}^\top)$ and $d_i \triangleq h_{D\mathbb{W}}(P_{\mathrm{ISS}_i}^\top)$, for $i \in \{1, \ldots, r_{\rm ISS}\}$. Further details can be found in [19].

On the other hand, maximizing the second objective subject to constraints (23e) to (22j) allows finding vectors of support functions to sets $\mathcal{ISS}$, $B\mathbb{U}_{UC}$ and $D\mathbb{W}_j$, which are required for constraints (22k, 22l) to compute $T_{\rm UC}$ when $\rho_A = 1$.

2) Stable systems where $\rho_A < 1$: The left-hand sides (LHS) of (20) and (21) are composed by the same support functions weighted by different coefficients. As for the coefficients of $h_{\mathcal{ISS}_i}$, if $\rho_A < 1$, $\rho_A^{T_{\rm UC}} \in (0, \rho_A]$, $\forall T_{UC} \in \mathbb{N}$ in (20), so it can be established that $\rho_A^{T_{\rm UC}} < 1$, where 1 is the corresponding coefficient in (21). Also, if $\rho_A < 1$, the coefficients of $\left(h_{B\mathbb{U}_{uc_i}} + h_{D\mathbb{W}_i}\right)$ of (20) is always lower than or equal to $T_{\rm UC}$, which is the corresponding coefficient of (21). Therefore, the LHS of (21) represents an upper bound for the LHS of (20) and the T_{UC} computed by the LP (22) will be suitable, but may not be the maximum feasible value, for (20) may hold for greater $T_{\rm UC}$ values than those of constraint (22k) of the LP. To find the maximum T_{UC} value, constraint (20) can be iterated after solving the LP (22) using the resulting support function values $h_{\mathcal{ISS}}$, $h_{B\mathbb{U}_{uc}}$ and $h_{D\mathbb{W}_j}$. In particular, T_{UC} should be increased from the value returned by the LP until the last one that complies with the constraint is found.

3) Unstable systems ($\rho_A > 1$): If $\rho_A > 1$, the relation between constraints (20) and (21) is the opposite of the previous case ($\rho_A < 1$). Therefore, the LP (22) has to be initially computed but the obtained T_{UC} represents an upper bound $\overline{T_{UC}}$. To find the maximum T_{UC}, constraint (20) has to be iterated as in the previous case of $\rho_A < 1$, but starting with $T_{UC} = \overline{T_{UC}}$ and decreasing it until the constraint is held. The last value represents the maximum feasible T_{UC}. Thus, the number of iterations required will be $\overline{T_{UC}}$ at most.

B. Computation of T_{SC}

The *safe set* is defined as the T_{SC}*-steps stabilizable set*, which represents the set of states for which exists an admissible control sequence that drives the system to the invariant set $\mathcal{ISS}$ in T_{SC} steps with an admissible evolution. Therefore, T_{SC} can be obtained following Algorithm 1, which is based on the recursion for computing the n-steps stabilizable set. However, note that instead of having T_{SC} as input to define $\mathcal{SS}$, the algorithm does the opposite: it computes T_{SC} starting from $\mathbb{X}_{SC} = \mathcal{SS}$ as the T_{SC}*-steps stabilizable set*.

Algorithm 1 Computation of T_{SC}

0: **Initialization:** $\mathcal{S} \leftarrow \mathcal{ISS}$, $k \leftarrow 1$
0: **Step 1:** $\mathcal{S} \leftarrow (A_{SC}^{-1}(\mathcal{S} \ominus D\mathbb{W})) \cap \mathcal{SS}$
0: **if** $\mathcal{S} \neq \mathcal{SS}$ **then**
0: $k \leftarrow k + 1$
0: GO TO Step 1
0: **else**
0: $T_{SC} \leftarrow k$
0: **end if**=0

IV. Results

The algorithm is tested using the linear model of the lab-scale microgrid HyLab [20] with sample time $T_S = 0.5$ min:

$$x(k+1) = x(k) + \begin{bmatrix} 8.136 & 5.958 \\ -15.2886 & 0 \end{bmatrix} u(k) + \begin{bmatrix} 5.958 \\ 0 \end{bmatrix} w(k), \quad (24)$$

where the state variables are the state of charge of the battery $SOC(k)$ and the metal hydrides level of the storage tank $MHL(k)$, both measured in percentage (%), i.e., $x = [SOC,\ MHL]^\top$; inputs are $u = [P_{H_2}\ P_{grid}]^\top$, with the former representing the power of the electrolyzer ($P_{H_2}(k) \leq 0$) and the power of the fuel cell ($P_{H_2}(k) > 0$), and the latter the power exchange with the grid, which is positive when power is imported and negative when exported; finally, disturbances are given by $w(k) = P_{net}(k)$, which corresponds to the difference between the power produced by a renewable energy source and the power demanded by the consumers.

State constraints are given by $x_{max} = [40,\ 40]^\top$ and $x_{min} = [-40,\ -40]^\top$. Regarding inputs signals, they have to be within the intervals $P_{H_2}(k) \in [-0.9, 0.9]$ kW and $P_{grid}(k) \in [-2.5, 2]$ kW. Likewise, the disturbance is considered to lie in the set $w(k) \in [-0.1995, 0.2732]$, which has been generated based on solar generation and demand data from 2020 from the Spanish National Electricity Network (SNEN) webpage.[1]

[1]https://demanda.ree.es/visiona/peninsula/demanda/total

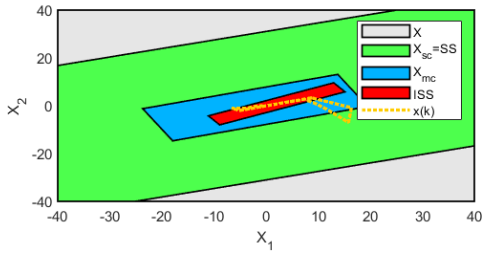

Fig. 3: Admissible state and safe sets for the HyLab model, including also the system evolution for the simulation.

A. Computation of safe sets

The control laws for SC and MC are

$$K_{SC} = \begin{bmatrix} 0.0103 & -0.0290 \\ 0.0293 & 0.0131 \end{bmatrix}, K_{MC} = \begin{bmatrix} 0.0066 & -0.0173 \\ 0.0155 & 0.0066 \end{bmatrix},$$

which correspond to discrete LQR designed using $Q_{SC} = 10I$, $Q_{MC} = 2.5I$ and $R_{SC} = R_{MC} = \text{diag}(5000, 8000)$, where I is the unit matrix of the corresponding dimensions. The corresponding closed-loop matrices A_{SC} and A_{MC} can be obtained using (10) and (7), respectively.

Safe sets and admissible state sets for safety and mission control mode are depicted in Fig. 3, where $\mathcal{SS} = \mathbb{X}_{SC}$ and $\mathcal{ISS}$ is computed using the LP of (22) since $\rho_A = 1$. The reduced input constraint set for the uncertain control period, $\mathbb{U}_{UC} = \alpha_{UC}\mathbb{U}$, has been defined for $\alpha_{UC} = 0.15$ to guarantee $\mathcal{ISS} \subseteq \mathcal{X}_{MC}$. Likewise, $T_{UC} = 5 = 2.5$ min, which is also computed with the same LP as $\mathcal{ISS}$. Assuming $T_{SR} = 1 = 0.5$ min, the remaining time steps for MC is $T_{MC} = 4 = 2$ min. Finally, $T_{SC} = 12 = 6$ min.

B. Simulation results

Figs. 3 and 4 shows the effects of two external attacks to test the software rejuvenation strategy. In the first one, the attacker sets the input values as zero, whereas in the second one, it tries to drive the system to unsafe states by setting the input signals to their maximum values. Here, the disturbance signal is generated based on data from October 1st, 2020, from 9 am to 1 pm. Note that while the software refresh is taking place, input signals keep the same value as in the last instant of mission control mode.

The system remains in $\mathcal{ISS}$ during all time steps when there are no attacks, so that the SC mode does not need to be activated. Also, it remains within $\mathcal{SS}$ while being attacked and the SC mode is successfully activated after finishing the software refresh, and the state is taken back to the $\mathcal{ISS}$ in less than the bounded time T_{SC}.

As can be seen, the design of safe sets is conservative, resulting in a relatively high software refresh frequency. The main reason is that the disturbances set has been estimated by getting data from an entire year. Likewise, the time required for the safety controller to take the system back to $\mathcal{ISS}$ is lower than the bounded time T_{SC}. Less conservatism and a lower frequency of refreshes can be attained by using data from a shorter period to build the disturbance set, e.g., from the previous weeks.

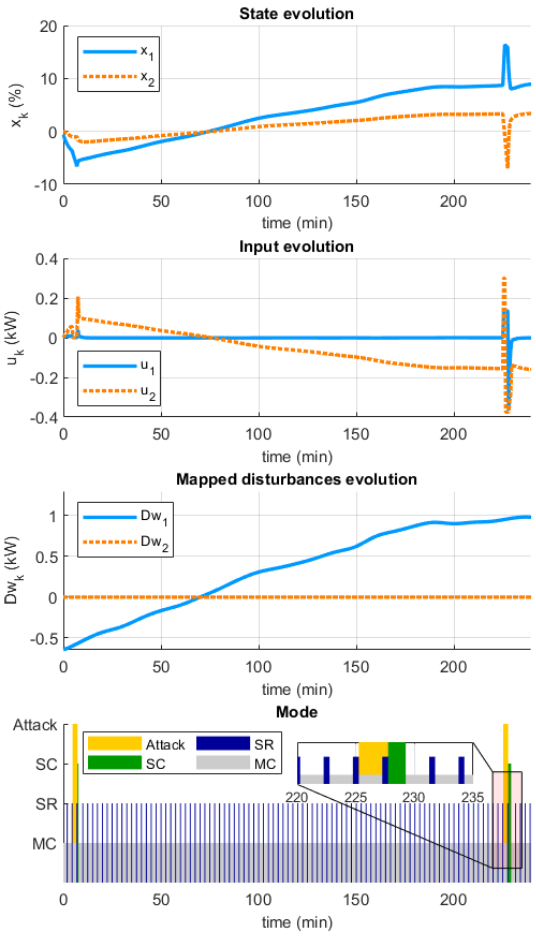

Fig. 4: Simulation with two external attacks.

V. Conclusion

This article extends the software rejuvenation strategy into the discrete-time setting to ensure system safety without the need for attack detection mechanisms. Leveraging the results of [19] and considering stable systems, a one-step LP to compute a *inner safe set* satisfying software rejuvenation constraints and an admissible T_{UC}. Also, an equivalent iterative process based on LPs has been presented for unstable systems. Both approaches are especially useful for high-dimensional systems, even when the computation of the maximum number of time steps T_{SC} is performed using classical set-theoretic tools. However, the designed algorithms are only suitable in the case of the matrix A that defines the system dynamics only has real eigenvalues. Also, the performance of the algorithm has been verified via simulation.

Extensions of this work will consider systems with complex eigenvalues and model predictive control (MPC) strategies for MC and SC controllers. Finally, connections with reinforcement learning will also be explored in this context.

References

[1] D. Kushner, "The real story of Stuxnet," *IEEE Spectrum*, vol. 3, no. 50, pp. 48–53, 2013.

[2] A. Bindra, "Securing the power grid: Protecting smart grids and connected power systems from cyberattacks," *IEEE Power Electronics Magazine*, vol. 4, no. 3, pp. 20–27, 2017.

[3] M. Uma and G. Padmavathi, "A survey on various cyber attacks and their classification.," *IJ Network Security*, vol. 15, no. 5, pp. 390–396, 2013.

[4] P. Cheng, L. Shi, and B. Sinopoli, "Guest editorial special issue on secure control of cyber-physical systems," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 1, pp. 1–3, 2017.

[5] Y. Z. Lun, A. D'Innocenzo, F. Smarra, I. Malavolta, and M. D. Di Benedetto, "State of the art of cyber-physical systems security: An automatic control perspective," *Journal of Systems and Software*, vol. 149, pp. 174–216, 2019.

[6] F. Pasqualetti, F. Dörfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE transactions on automatic control*, vol. 58, no. 11, pp. 2715–2729, 2013.

[7] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *6th ACM international symposium on Mobile ad hoc networking and computing*, pp. 46–57, 2005.

[8] M. Arroyo, H. Kobayashi, S. Sethumadhavan, and J. Yang, "FIRED: frequent inertial resets with diversification for emerging commodity cyber-physical systems," *CoRR*, vol. abs/1702.06595, 2017. Available online: https://arxiv.org/abs/1702.06595.

[9] F. Abdi, C. Y. Chen, M. Hasan, S. Liu, S. Mohan, and M. Caccamo, "Guaranteed physical security with restart-based design for cyber-physical systems," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pp. 10–21, IEEE, 2018.

[10] R. Romagnoli, B. H. Krogh, and B. Sinopoli, "Design of software rejuvenation for CPS security using invariant sets," in *2019 American Control Conference (ACC)*, pp. 3740–3745, IEEE, 2019.

[11] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton, "Software Rejuvenation: Analysis, module and applications," in *25th international symposium on fault-tolerant computing. Digest of papers*, pp. 381–390, IEEE, 1995.

[12] D. Cotroneo, R. Natella, R. Pietrantuono, and S. Russo, "A survey of software aging and rejuvenation studies," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 10, no. 1, pp. 1–34, 2014.

[13] K. M. M. Aung and J. S. Park, "Software Rejuvenation approach to security engineering," in *International Conference on Computational Science and Its Applications*, pp. 574–583, Springer, 2004.

[14] F. Gruber and M. Althoff, "Computing safe sets of linear sampled-data systems," *IEEE Control Systems Letters*, vol. 5, no. 2, pp. 385–390, 2020.

[15] S. V. Rakovic, E. C. Kerrigan, K. I. Kouramas, and D. Q. Mayne, "Invariant approximations of the minimal robust positively invariant set," *IEEE Transactions on automatic control*, vol. 50, no. 3, pp. 406–410, 2005.

[16] S. V. Raković, B. Kouvaritakis, and M. Cannon, "Equi-normalization and exact scaling dynamics in homothetic tube model predictive control," *Systems & Control Letters*, vol. 62, no. 2, pp. 209–217, 2013.

[17] I. M. Mitchell, J. Yeh, F. J. Laine, and C. J. Tomlin, "Ensuring safety for sampled data systems: An efficient algorithm for filtering potentially unsafe input signals," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 7431–7438, IEEE, 2016.

[18] K. P. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 7130–7135, IEEE, 2018.

[19] P. Trodden, "A one-step approach to computing a polytopic robust positively invariant set," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 4100–4105, 2016.

[20] M. Pereira, D. Limon, D. M. de la Peña, L. Valverde, and T. Alamo, "Periodic economic control of a nonisolated microgrid," *IEEE Transactions on industrial electronics*, vol. 62, no. 8, pp. 5247–5255, 2015.