DOI: https://doi.org/10.35681/1560-9189.2026.28.2.363156

UDC 004.896:629.7.05

D. O. Humennyi
National Technical University of Ukraine
«Igor Sikorsky Kyiv Polytechnic Institute»
37 Beresteiskyi Avenue, 03056 Kyiv, Ukraine

Practical implementation and simulation of adaptive control for super-critical cyber-physical systems

This paper presents a practical framework for adaptive control and survivability management of super-critical cyber-physical systems. A complete simulation environment is developed, comprising: (i) a hierarchical control architecture with regime-dependent feedback laws; (ii) parametric degradation models for thermal, sensor, communication, and multi-component failure scenarios; (iii) detailed control algorithms with tunable parameters; and (iv) a modular C++ implementation with JSON-configurable settings. Extensive simulation studies across four realistic degradation scenarios demonstrate that the proposed adaptive strategies extend safe operation time by 40–76 %, improve minimum survivability by +0,25 to +0,68, and enable recovery from super-critical states in 75 % of cases. The implementation is validated through comprehensive input-output analysis showing consistent hardware protection across all test scenarios. A formal proof of bounded evolution is also provided, together with an explicit Lyapunov-based stability argument.

Keywords*. adaptive control, cyber-physical systems, degradation modeling, fault tolerance, functional safety, graceful degradation, Lyapunov stability, simulation, survivability.*

1. Introduction

Modern cyber-physical systems (CPS) increasingly operate in conditions that transcend conventional critical benchmarks, entering what is defined here as *super-critical operational modes*. In such regimes, systems exhibit extreme sensitivity to perturbations, where small disturbances can rapidly propagate into system-wide failures through nonlinear interactions. Unlike conventional critical systems that operate at the edge of stability, super-critical systems may temporarily lose structural integrity while still maintaining partial functionality — a characteristic that demands new control paradigms.

Traditional approaches to functional safety, exemplified by standards such as ISO 26262 for automotive systems and IEC 61508 for industrial applications, typically prescribe transition to a predefined *Safe State* upon failure detection. However, for many contemporary applications — including autonomous vehicles navigating traffic, surgical

robots during procedures, and industrial manipulators handling hazardous materials — immediate shutdown may present greater hazards than continued operation with degraded functionality. This necessitates control strategies capable of graceful degradation and rapid reconfiguration.

The concept of survivability in complex systems has evolved significantly since von Neumann's foundational work on reliable computation from unreliable components [1]. Modern interpretations emphasize the system's ability to maintain essential functions despite component failures, environmental disturbances, or adversarial actions [2, 3]. This paper extends these concepts to the super-critical domain, where traditional redundancy-based approaches prove insufficient due to cascading failure mechanisms and resource constraints.

The primary contributions of this work are:

1) a practical hierarchical control architecture with mathematically defined regime transitions and adaptive feedback gains optimized for each operational mode;

2) detailed parametric models for four realistic degradation scenarios (thermal, sensor cascade, communication, multi-component) with validated physical parameters;

3) complete control algorithms with pseudocode implementations, tuning guidelines and quantitative effectiveness analysis, accompanied by a formal proof of bounded evolution (Theorem 1);

4) a modular C++ simulation framework with JSON configuration enabling rapid prototyping and parameter studies.

The remainder of this paper is organized as follows. Section 2 reviews related work and the state of the art. Section 3 presents the system architecture and mathematical framework. Section 4 details the degradation models and simulation parameters. Section 5 describes the control algorithms and provides their theoretical justification, including the proof of Theorem 1. Section 6 presents the C++ software architecture and the scalability analysis. Section 7 reports comprehensive simulation results. Section 8 discusses practical implications and limitations. Section 9 concludes the paper and outlines directions for future research.

2. Related work and state of the art

The problem of maintaining functionality in the presence of faults, disturbances and partial component failures lies at the intersection of several active research areas: fault-tolerant control, resilient cyber-physical systems, adaptive and learning-based control, and functional safety engineering. A concise review of each area is given below, together with the positioning of the present contribution.

A. Reliability and Survivability Foundations

The systematic study of computation and control under component unreliability originates with von Neumann's 1956 monograph [1], in which he demonstrated how arbitrarily reliable behavior can be synthesized from unreliable elements through redundancy and majority voting. His work established the quantitative link between component failure probability and system-level reliability that underpins modern reliability engineering. Contemporary developments extend this legacy to networked cyber-physical systems: the survey by Zhang *et al.* [2] catalogues physical-safety and cyber-security threats to multi-agent CPS, and the roadmap by Ratasich *et al.* [3] identifies resilience mechanisms for the industrial Internet of Things. Colabianchi *et al.* [4] provide a systematic ta-

xonomy of resilience in the CPS context, distinguishing it from reliability, robustness and survivability — a distinction that is essential for the super-critical domain considered here.

B. Fault-Tolerant and Adaptive Control

The classical reference on fault detection, isolation and accommodation is the monograph by Blanke *et al.* [5], which formalises the architecture of fault-tolerant control (FTC) systems and provides the parametric failure models used later in this paper. Recent advances extend FTC to nonlinear systems with actuator faults using event-triggered reinforcement learning [6], neural-network observers [7], and model-predictive methods with Lyapunov-Razumikhin analysis for time-delayed batch processes [8]. Off-policy reinforcement learning has been applied to min-max fault-tolerant tracking in industrial processes [9], while Cohen and Belta [10, 11] developed control-barrier-function techniques for safe exploration and adaptive Lyapunov-based stabilisation. Lopez and Slotine [12] proposed a universal adaptive controller for nonlinear systems which achieves guaranteed transient performance under parametric uncertainty.

The framework presented here differs from the above in three respects. First, it explicitly targets the *super-critical* regime, in which the Lyapunov derivative is allowed to be temporarily positive provided recovery is guaranteed within a prescribed horizon. Second, the regime-dependent gains are obtained from a family of discrete algebraic Riccati equations (DARE) rather than from a single adaptive law. Third, the approach is explicitly engineered for embedded deployment with hard real-time constraints.

C. Digital Twins and Model-Based System Health Management

Liu [13] reviews control strategies for digital-twin systems and emphasizes the role of synchronised high-fidelity plant models for predictive fault management. The degradation models developed in Section 4 of the present work can be viewed as a lightweight, real-time implementation of the digital-twin concept: parametric models of thermal, sensor and communication health are co-simulated with the plant in order to drive regime transitions.

D. Security-Aware and Distributed Control of CPS

Safety and security are increasingly analyzed jointly for networked CPS. Ge *et al.* [14] propose a Krein-space attack-detection scheme for sensor networks under deception attacks, and Feng and Hu [15] develop secure cooperative event-triggered control for linear multi-agent systems under denial-of-service (DoS) attacks. Distributed perception is a further enabler: Tian *et al.* [16] demonstrate Kimera-Multi, a robust distributed dense metric–semantic SLAM system for multi-robot platforms. The software architecture in Section 6 is designed to be compatible with such distributed middleware stacks (ROS 2 / DDS).

E. Super-Critical Operational Modes and Positioning of the Present Work

The notion of a *super-critical operational mode* was introduced in the author's earlier papers [17] and elaborated in the PhD dissertation [18]. Those works established the qualitative definition of the super-critical regime and proposed the hierarchical abstraction model that motivates the control architecture of Section 3. The present paper complements this earlier line of research by (a) giving a concrete, implementable pseudocode for each control layer, (b) providing simulation evidence that quantifies the benefits across four distinct degradation scenarios, and (c) supplying a formal proof of bounded state evolution (Theorem 1, Section 5). Thus, the present work represents a transition from conceptual and modeling studies to a validated engineering framework.

3. System architecture and mathematical framework

A. Hierarchical Control Structure

The control system implements a four-level hierarchical architecture, where each level operates at a distinct time scale and abstraction level. This separation of concerns enables targeted responses to different types of disturbances while maintaining overall system coherence. The architecture is summarized in Table 1.

Table 1. Hierarchical control architecture

Level	Component	Primary Function	Update Rate
L0	Hardware Layer	Direct actuator/sensor interface, interrupt handling	10 kHz (interrupt)
L1	Servo Control	PID loops, torque/position regulation, trajectory tracking	1 kHz (deterministic)
L2	Supervisor	Fault detection, regime classification, mode switching	100 Hz (soft real-time)
L3	Mission Manager	Task planning, resource allocation, reconfiguration	10 Hz (event-driven)

The L0 layer provides hardware abstraction, handling sensor acquisition and actuator commands at the highest frequency. The L1 layer implements classical servo control with adaptation mechanisms described in Section 5. The L2 supervisor layer monitors system health indicators and triggers mode transitions. The L3 mission layer manages high-level objectives and coordinates degraded-operation strategies.

B. Plant Model

The controlled plant is modeled as a discrete-time linear parameter-varying (LPV) system in which the system matrices depend on the resource health vector $\rho(t)$:

$$x_{k+1} = A_d(\rho_k)\, x_k + B_{(\rho k)}\, u_k + E_d(\rho_k)\, \xi_k \quad (1)$$

$$y_k = C_d(\rho_k)\, x_k + v_k \quad (2)$$

where $x_k \in \mathbb{R}n$ is the state vector; $u_k \in \mathbb{R}m$ is the control input; $y_k \in \mathbb{R}p$ is the measurement vector; ξ_k represents external disturbances; v_k is measurement noise; $\rho_k \in [0,1]^r$ is the resource health vector with r components. The resource-dependent matrices capture the physical reality that system dynamics change as components degrade — for example, motor dynamics shift as thermal effects alter winding resistance.

For the simulation studies in this paper, a simplified scalar model is used that captures the essential dynamics while remaining computationally tractable:

$$x_{k+1} = A \cdot x_k + B \cdot u_k + w_k \quad (3)$$

where $A = 0{,}9$ represents a stable but lightly damped system, $B = 1{,}0$ is the control gain, and w_k lumps the disturbance effects. This formulation enables rapid parameter studies while preserving the fundamental control challenges.

C. Resource Degradation Model

Each system resource i has an operability coefficient $\rho_i \in [0,1]$ that evolves according to:

$$\rho_{i,k+1} = \rho_{i,k} - \lambda_i(s_k) \cdot \Delta t - \eta_i \cdot D_{i,k} + \mu_i \cdot u^{rep}{}_{i,k} \quad (4)$$

where $\lambda_i(s_k)$ is the regime-dependent degradation rate; $D_{i,k}$ represents external damage events; η_i is the damage sensitivity coefficient; $u^{rep}{}_{i,k}$ models repair or compensation actions with effectiveness μ_i. The degradation rate increases with system stress, as summarized in Table 2.

Table 2. Regime-Dependent Degradation Parameters

Parameter	Normal (*N*)	Critical (*K*)	Super-Critical (*SK*)	Unit
Base degradation rate λ_0	0,01	0,05	0,10	Δt^{-1}
Feedback gain K	$K_N = 0{,}8$	$K_K = 0{,}5$	$K_{SK} = 0{,}2$	–
Damage coefficient η	1,0	1,5	2,0	–

The five-fold increase in degradation rate from Normal to Super-Critical mode reflects the physical reality that stressed components fail faster — a phenomenon well documented in the reliability-engineering literature [5].

D. Survivability Indicators

System survivability is characterized by three complementary indicators.

1. ***Structural Survivability S_k.*** The aggregate health of system resources:

$$S_k = (1/r) \sum i \, \rho_{i,k}. \quad (5)$$

For systems with non-uniform resource importance, weighted formulations $S_k = = \sum j \, w_j \, \sigma_j(\rho_k)$ can be employed, where w_j reflects the criticality of resource group j and σ_j is an activation function.

2. ***Parametric Integrity P_k.*** Control performance relative to design specifications:

$$P_k = \exp(-\beta \cdot |e_k|), \quad (6)$$

where e_k is the tracking error and $\beta = 0{,}5$ is the sensitivity parameter. The exponential form ensures $P_k \in [0, 1]$ and provides smooth degradation as errors increase.

3. ***Functional Integrity F_k.*** A multiplicative combination of structural and parametric aspects:

$$F_k = S_k \cdot P_k. \quad (7)$$

This formulation ensures that degradation in either the structural or the parametric domain reduces overall functionality — a system with perfect hardware but poor control performance (or vice versa) cannot claim full functional integrity.

E. Regime Classification

The supervisor classifies the system's operational state based on functional-integrity thresholds:

$$s_k = N \text{ (Normal)}, \quad \text{if } F_k \geq F_N = 0{,}8, \quad (8)$$

$$s_k = K \text{ (Critical)}, \quad \text{if } F_K \leq F_k < F_N, \; F_K = 0{,}3, \quad (9)$$

$$s_k = SK \text{ (Super-Critical)}, \quad \text{if } F_k < F_K. \quad (10)$$

These thresholds were selected on the basis of industrial practice: $F_N = 0{,}8$ corresponds to the typical 80 % functionality threshold for degraded operation in safety stan-

dards, while $F_K = 0{,}3$ represents the minimum viable functionality before emergency protocols activate. The resulting state classification and corresponding control responses are summarized in Table 3.

Table 3. State classification and system response

State	F_k Range	Mission Mode	Control Response
N (Normal)	$F \geq 0{,}8$	NOMINAL	Full functionality, standard gains K_N
K (Critical)	$0{,}3 \leq F < 0{,}8$	DEGRADED	Reduced gains K_K, activate redundancy, limit performance
SK (Super-Critical)	$F < 0{,}3$	LIMP_HOME	Minimum gains K_{SK}, emergency protocols, prepare shutdown
F (Failure)	$F \leq 0$	SAFE_STOP	Controlled shutdown sequence, protect hardware

F. Adaptive Control Law

The control input is computed using regime-dependent state feedback:

$$u_k = -K_{sk} \cdot \hat{x}_k \tag{11}$$

where $\hat{x}_k$ is the state estimate and K_{sk} is the feedback gain corresponding to the current regime. The gains are designed to balance performance and stability: $K_N = 0{,}8$ provides aggressive tracking in normal operation, $K_K = 0{,}5$ offers moderate response in critical mode, and $K_{SK} = 0{,}2$ ensures conservative, stable behavior in super-critical conditions.

Stability is verified using the Lyapunov function $V(x_k) = x_k{}^T P\, x_k$, where P is the solution of the discrete algebraic Riccati equation (DARE). The controller ensures $\Delta V = V(x_{k+1}) - V(x_k) < 0$ for all states within the stability region of each regime. A formal proof is given in Section 5.B.

4. Degradation scenarios and parametric models

Four realistic degradation scenarios were developed to evaluate system behavior across the operational spectrum. Each scenario is based on documented failure modes in robotic and automotive systems, with parameters derived from manufacturer specifications and field data. In the following tables the operational states are denoted uniformly by the letters N (Normal), W (Warning), K (Critical), SK (Super-Critical) and F (Failure), so that the notation agrees with Table 3 and with the regime labels used in Section 3.

A. Scenario A: Thermal Degradation of an Actuator System

1. Physical Description.

A six-degree-of-freedom (6-DOF) robotic manipulator operates under high-intensity continuous load. The efficiency of the cooling system decreases due to dust accumulation on heat sinks ($\eta_{cooling}$: 1,0 → 0,4 over 200 h). Motor M3 at the elbow joint experiences accelerated heating due to the highest torque demand, with thermal cross-coupling affecting the adjacent motor M4.

2. Thermal Model.

Temperature evolution follows the first-order thermal model:

$$dT/dt = (P_{dissipated} - P_{cooling}) \,/\, C_{thermal}, \tag{12}$$

$$P_{cooling} = h \cdot A \cdot (T - T_{ambient}) \cdot \eta_{cooling}(t), \quad (13)$$

where h is the convective heat-transfer coefficient, A is the heat-sink surface area, and $\eta_{cooling}(t)$ decreases linearly with dust accumulation. The full degradation timeline is presented in Table 4.

Table 4. Scenario A — Thermal degradation timeline

t, s	T_{M3}	T_{M4}	ρ_{M3}	ρ_{M4}	η_{cool}	S_k	u_{comp}	State
0	42 °C	40 °C	0,98	1,00	0,65	0,94	0,00	N
60	58 °C	48 °C	0,95	0,97	0,62	0,89	0,05	N
120	72 °C	56 °C	0,88	0,94	0,58	0,82	0,12	W
180	85 °C	64 °C	0,75	0,89	0,52	0,71	0,28	W
240	96 °C	72 °C	0,58	0,82	0,45	0,58	0,45	K
300	108 °C	81 °C	0,35	0,72	0,42	0,42	0,65	SK
360	118 °C	88 °C	0,12	0,58	0,40	0,28	max	F

The scenario demonstrates the cascading nature of thermal failures: M3 overheating increases the ambient temperature of M4, accelerating its degradation and creating a positive feedback loop.

B. Scenario B: Cascading Sensor Failure

1. Physical Description.

Electromagnetic interference (EMI) from nearby welding equipment induces noise in the sensor signals. The primary position encoder on Joint 2 develops intermittent faults. The fault-detection algorithm initially misattributes noise to mechanical vibration, delaying the appropriate response. When the backup Inertial Measurement Unit (IMU) activates, it has accumulated drift error from prolonged standby operation.

2. Sensor Noise Model:

$$\sigma(t) = \sigma_0 + \sigma_{EMI} \cdot |\sin(2\pi\, f_{EMI}\, t)| + \sigma_{drift} \cdot t, \quad (14)$$

where σ_0 is the baseline sensor noise; σ_{EMI} is the amplitude of the EMI-induced noise at carrier frequency f_{EMI}; σ_{drift} is the IMU drift rate. The sensor cascade failure timeline is given in Table 5.

Table 5. Scenario B — Sensor cascade failure timeline

t, s	σ_{enc}, rad	ρ_{enc}	IMU drift, °/s	ρ_{IMU}	S_k	Pos. err., mm	State
0	0,001	1,00	0,01	0,98	0,96	0,1	N
60	0,008	0,82	0,05	0,92	0,81	0,9	W
120	0,080	0,25	0,35	0,72	0,48	8,5	SK
150	FAULT	0,00	0,65	0,58	0,32	15,2	SK
180	FAULT	0,00	1,20	0,35	0,22	28,4	F

C. Scenario C: Communication Link Degradation

1. Physical Description:

The primary Controller Area Network (CAN) bus experiences intermittent failures due to connector degradation from vibration-induced micro-fractures. The message-loss rate increases progressively, and control latency grows as retransmissions multiply. A secondary CAN bus is available but operates at reduced bandwidth.

2. Communication-Quality Metric:

$$\rho_{comm} = 1 - (\alpha \cdot BER_{norm} + \beta \cdot loss_{norm} + \gamma \cdot latency_{norm}), \qquad (15)$$

where BER is the bit error rate, loss is the message loss percentage, and latency is the average message delay. The weighting coefficients $\alpha = 0{,}3$; $\beta = 0{,}4$; $\gamma = 0{,}3$ reflect the relative importance of each quantity for control applications. The communication degradation timeline is presented in Table 6.

Table 6. Scenario C — Communication degradation timeline

t, s	BER	Loss (no ctrl)	Loss (ctrl)	Lat. (no ctrl)	Lat. (ctrl)	S_k	State
0	10^{-8}	0,01 %	0,01 %	2 ms	2 ms	0,95	N
200	10^{-5}	3,5 %	1,2 %	18 ms	8 ms	0,86	W
400	10^{-3}	35 %	5 %	120 ms	18 ms	0,72	K
500	FAIL	100 %	8 %	∞	25 ms	0,65	K
600	FAIL	–	10 %	–	30 ms	0,58	K

The columns labelled «ctrl» show behavior with failover control active, demonstrating a successful transition to the backup communication channel at $t = 400$ s.

D. Scenario D: Multi-Component Cascade Failure

1. Physical Description.

A power-supply voltage sag (48 V → 32 V → recovery) causes simultaneous stress across all subsystems: motors enter protection mode, sensors experience brown-out resets, and communication timing is disrupted. Recovery requires the coordinated restoration of all subsystems in the correct sequence.

2. Cascade Dynamics.

This scenario uniquely demonstrates bidirectional coupling: the power transient simultaneously affects all subsystems, but their recovery must be sequenced (sensors → communication → motors) to prevent control instability. The multi-component cascade timeline is given in Table 7.

Table 7. Scenario D — Multi-component cascade timeline

t, s	V_{psu}	$\rho_{m\,(no)}$	$\rho_{m\,(c)}$	$\rho_{s\,(c)}$	S_{no}	S_c	dV_{no}	dV_c	State
0,0	48 V	0,98	0,98	0,99	0,94	0,94	−0,02	−0,02	N
0,5	38 V	0,65	0,78	0,88	0,58	0,75	+0,55	+0,22	K
1,0	32 V	0,35	0,62	0,75	0,28	0,58	+0,95	+0,35	SK
2,0	42 V	0.18	0,62	0,78	0,12	0,62	+0,85	−0,08	K
5,0	47 V	0,12	0,85	0,92	0,08	0,82	+0,65	−0,15	N
15,0	48 V	F	0,94	0,97	0,00	0,91	–	−0,02	N

Columns marked «no» correspond to the uncontrolled (no-control) case, those marked «c» to the coordinated-control case. The Lyapunov derivative dV/dt transitions from positive (unstable) to negative (stable) at $t \approx 2$ s for the coordinated response, indicating successful stability recovery. The uncoordinated response never achieves stability and fails at $t \approx 5$ s.

5. Control algorithms and theoretical justification

A. Design Philosophy and Theoretical Foundation

The control algorithms developed in this work are grounded in Lyapunov stability theory and optimal control principles. Each algorithm component is mathematically justified, with parameter selection based on stability margins, convergence guarantees and practical implementation constraints. The hierarchical structure ensures that higher-level decisions (regime classification, mode switching) operate on slower timescales than lower-level control loops, preventing instability caused by temporal coupling [6].

1. Stability-Constrained Gain Selection.

The regime-dependent gains K_N, K_K, K_{SK} are not arbitrary but derived from discrete algebraic Riccati equation (DARE) solutions with modified weighting matrices. For regime $s \in \{N, K, SK\}$ the optimal gain minimizes:

$$J_s = \sum_k (x_k{}^T Q_s x_k + u_k{}^T R_s u_k), \tag{16}$$

where Q_s and R_s are regime-specific weighting matrices. The closed-loop eigenvalues $\lambda_i (A - BK_s)$ must satisfy $|\lambda_i| < 1$ for stability. The gain values $K_N = 0{,}8$, $K_K = 0{,}5$, $K_{SK} = 0{,}2$ correspond to solutions with increasing control-effort penalty R_s, reflecting the need for conservative action as resources degrade.

Table 8. Riccati-based gain design parameters

Regime	Q_s	R_s	K_s	Stability Margin
Normal ($s = N$)	1,0	0,5	0,8	$\lvert\lambda_{max}\rvert = 0{,}10$
Critical ($s = K$)	1,0	2,0	0,5	$\lvert\lambda_{max}\rvert = 0{,}40$
Super-Critical ($s = SK$)	1,0	8,0	0,2	$\lvert\lambda_{max}\rvert = 0{,}70$

The stability margin $|\lambda_{max}|$ increases (worsens) in degraded modes, but remains strictly less than one, guaranteeing asymptotic stability. This trade-off is intentional: aggressive control in super-critical mode would accelerate component degradation while providing minimal performance benefit.

2. Threshold Selection Rationale:

ISO 26262 ASIL decomposition. The 80 % threshold $F_N = 0{,}8$ aligns with the typical degraded-operation acceptance criterion in functional-safety analysis, where systems must maintain at least 80 % functionality to continue operation without driver intervention.

Lyapunov stability margin. The 30 % threshold $F_K = 0{,}3$ corresponds to the minimum functional integrity at which the Lyapunov derivative dV/dt can be guaranteed negative with reduced gains. Below F_K, stability cannot be assured without emergency intervention.

B. Main Control Loop with Convergence Analysis

Algorithm 1 presents the complete discrete-time control loop. Under mild assumptions, the algorithm guarantees bounded state evolution and eventual convergence to a stable equilibrium.

Theorem 1 (Bounded Evolution). Let the plant (3) be stabilisable with $|A| < 1 + \varepsilon$ for some small $\varepsilon > 0$. If the degradation rate satisfies $\lambda_{SK}\ T < 1$ and disturbances are bounded $|\xi_k| \le \xi_{\max}$, and if the Lyapunov matrix P is the positive-definite solution of the DARE for the dominant regime with spectral radius $\rho(A - BK_s) < 1$, then the state trajectory $\{x_k\}$ remains uniformly bounded for all $k \ge 0$.

Proof. The argument follows the standard Lyapunov analysis for switched discrete-time systems [19, Ch. 4], [5, Sec. 5.3]. Consider the Lyapunov candidate $V(x_k) = x_k^T P\, x_k$. With the regime-dependent gain Ks satisfying $|\lambda_i (A - BK_s)| < 1$ (Table 8), the closed-loop dynamics $x_k + 1 = (A - BK_s)x_k + Ewk$ lead to the following Lyapunov bound:

$$\Delta V = V(x_{k+1}) - V(x_k) \le -\alpha\, |x_k|^2 + \gamma\, \xi_{\max}{}^2, \tag{17}$$

with $\alpha > 0$ determined by the minimum eigenvalue of $Q_s - K_s{}^T R_s\, K_s$ and $\gamma > 0$ by the spectral norm of the disturbance-input matrix. Accordingly, $\Delta V < 0$ whenever $|x_k| > \sqrt{(\gamma/\alpha)}\ \xi_{\max}$, which defines an ultimate bound. Thus V is monotonically decreasing outside a ball of radius $B_\infty = \sqrt{(\gamma/\alpha)} \cdot \xi_{\max}$, and the state trajectory is uniformly bounded by B_∞.

The degradation condition $\lambda_{SK} \cdot T < 1$ ensures that the super-critical regime is traversed in finite time before ρ_i reaches zero, so that the supervisor is always able to issue a transition command before catastrophic failure. The switching logic (8)–(10) guarantees that the dwell time in each regime is sufficient to preserve the Lyapunov decrease, which is the standard condition for stability of switched systems.

Remark 1. The bound B_∞ scales with $\xi_{\max}$ and with $\sqrt{(1/\alpha)}$; in the nominal regime α is largest (because Q_N / R_N is largest), so disturbance rejection is best during normal operation and gracefully degrades as the system enters *K* and *SK* modes.

Algorithm 1. Discrete-time adaptive control loop

```
function ControlLoop(config, disturbances):
  initialize: x[0] <- x0 ;   rho[0] <- rho0
  for k = 0 to N_max do:
      // 1. Measurement and state estimation
    y[k]     <- C_d * x[k] + v[k]
    xhat[k]  <- Observer.update( u[k-1], y[k] )
      // 2. Compute survivability indicators
    S[k] <- (1/r) * sum_i rho_i[k]
    P[k] <- exp( -beta * |e[k]| )
    F[k] <- S[k] * P[k]
      // 3. Classify regime and select mission mode
    if F[k] >= F_N          then s[k] <- N  ; q[k] <- NOMINAL
    else if F[k] >= F_K     then s[k] <- K  ; q[k] <- DEGRADED
    else                        s[k] <- SK ; q[k] <- LIMP_HOME
    if F[k] <= 0 then q[k] <- SAFE_STOP ; break
      // 4. Apply regime-dependent control
    K_s  <- ConfigDB.getGain( s[k] )
    u[k] <- -K_s * xhat[k]
      // 5. Update plant and degradation
    x[k+1]   <- A_d*x[k] + B_d*u[k] + E_d*xi[k]
    rho[k+1] <- rho[k] - lambda(s[k])*dt - eta*D[k]
```

```
      // 6. Log results
    results.append( k, x[k], rho[k], S[k], P[k], F[k], s[k], u[k] )
  end for
return results
```

C. Thermal Protection Controller with Margin-Based Design

Algorithm 2 implements the thermal-management strategy for Scenario A. The margin-based approach is derived from Arrhenius degradation kinetics: the degradation rate increases exponentially with temperature, so linear torque reduction provides approximately exponential life extension [5].

1. Margin Computation.

The thermal margin is normalized to [0, 1]:

$$margin = (T_{critical} - T_{motor}) \,/\, (T_{critical} - T_{nominal}), \tag{18}$$

when margin = one, the motor operates at nominal temperature $T_{nominal}$ with full life expectancy. When margin = zero, the motor has reached critical temperature with near-zero remaining life. The margin serves as a real-time estimate of the motor's thermal health.

2. Torque-Limiting Law.

The torque limit follows a piecewise linear function:

$$\tau_{limit} = \tau_{\max}, \qquad \text{if margin} \geq 0{,}5, \tag{19}$$

$$\tau_{limit} = \tau_{\max} \cdot (0{,}4 + 0{,}6 \cdot margin), \quad \text{if margin} < 0{,}5. \tag{20}$$

This formulation ensures: a) no torque reduction above 50 % margin (acceptable performance zone); b) gradual reduction to 40 % of the maximum at margin = 0 (protection zone); c) a continuous function that avoids discontinuities capable of causing control instability. The controller parameters are listed in Table 9.

Table 9. Thermal-controller parameters

Parameter	Value	Unit	Action Triggered
$T_{nominal}$	40	°C	Normal-operation baseline
$T_{warning}$	70	°C	Increase monitoring frequency 10×
$T_{critical}$	85	°C	Reduce torque limit to 70 %
$T_{emergency}$	100	°C	Force cooling + load redistribution
$T_{shutdown}$	115	°C	Initiate safe-stop sequence

Algorithm 2. Thermal-degradation control

```
function ThermalController( T_motor, T_ambient ):
  // Thermal margin (0 = critical, 1 = nominal)
  margin <- (T_critical - T_motor) / (T_critical - T_nominal)
  // Adaptive torque limiting
  if margin < 0.5 then
      tau_limit <- tau_max * (0.4 + 0.6 * margin)
  else
      tau_limit <- tau_max
  // Load redistribution to adjacent actuators
  if margin < 0.3 then
```

```
        transfer <- ComputeRedistribution( motor_id, neighbors )
        ApplyLoadTransfer( transfer, rate = 10 %/s )
    // Emergency response
    if T_motor > T_emergency then ActivateForcedCooling()
    if T_motor > T_shutdown  then TriggerSafeStop()
  return tau_limit
```

D. Sensor Fusion Controller with Adaptive Covariance

Algorithm 3 implements adaptive Kalman filtering for Scenario B. The key innovation is dynamic adjustment of the measurement-noise covariance R based on real-time sensor-quality estimates, providing optimal state estimation under degrading sensor conditions [14].

1.Sensor-Quality Estimation.

Sensor reliability ρ_{sensor} is estimated from the innovation-sequence variance:

$$\hat{\sigma_k}^2 = (1/N) \sum_{i=1..N} (z_{k-i} - \hat{y}_{k-i})^2, \quad (21)$$

where $N = 50$ is the sliding-window size. When $\hat{\sigma_k}$ exceeds $\sigma_{nominal}$ (the 99,7 % confidence bound for Gaussian noise), the sensor is flagged as degraded:

$$\rho_{sensor} = \max(0, 1 - (\sigma - \sigma_{nom})/(\sigma_{fail} - \sigma_{nom})). \quad (22)$$

2. Adaptive Measurement Covariance.

The measurement-noise covariance R is scaled inversely with sensor quality:

$$R_{adaptive} = R_{nominal} /(\rho_{ensor} + \varepsilon), \quad (23)$$

where $\varepsilon = 0.01$ prevents division by zero. This formulation has the following properties: a) when $\rho = 1$ (healthy sensor), $R = R_{nominal}$ and the Kalman gain weights the measurement normally; b) when $\rho \to 0$ (failed sensor), $R \to \infty$ and the measurement is effectively ignored; c) the transition is smooth, avoiding discrete switching artefacts.

3. Velocity Limiting Based on Confidence:

$$\text{confidence} = (\rho_{enc} \cdot w_{enc} + \rho_{imu} \cdot w_{imu}) / (w_{enc} + w_{imu}). \quad (24)$$

Velocity limits are scaled proportionally: $v_{limit} = v_{max} \cdot$ confidence. This ensures that operations slow down when position estimates become uncertain, providing a safety margin proportional to estimation quality.

Algorithm 3. Adaptive sensor fusion

```
function SensorFusionController( z_enc, z_imu ):
  // 1. Estimate sensor noise from recent history
  sigma_enc <- EstimateNoise( z_enc_history, window = 50 )
  // 2. Compute sensor reliability
  if sigma_enc > 3 * sigma_enc_nominal then
      rho_enc <- max( 0, 1 - (sigma_enc - sigma_nom) / (sigma_fail -
sigma_nom) )
  else
      rho_enc <- 1.0
  // 3. Adjust measurement-noise covariance
  R_enc_adaptive <- R_enc_nominal / (rho_enc + 0.01)
  R_imu_adaptive <- R_imu_nominal / (rho_imu + 0.01)
  // 4. Kalman update with adaptive R
```

```
  K <- P * H^T * inv( H*P*H^T + R_adaptive )
  x_upd <- x_pred + K * innovation
  // 5. Velocity limit based on estimation confidence
  confidence <- (rho_enc*w_enc + rho_imu*w_imu) / (w_enc + w_imu)
  v_limit    <- v_max * confidence
return x_upd, v_limit
```

E. Multi-Component Emergency Coordinator with Timing Guarantees

Algorithm 4 manages the coordinated emergency response for Scenario D. The algorithm is designed to meet hard real-time constraints while ensuring correct sequencing of recovery actions [15].

1. Phase 1 — Immediate Protection (WCET < 10 ms).

The parallel protection actions must be completed within 10 ms in order to prevent hardware damage during voltage transients. The worst-case execution time (WCET) analysis is:

— motor brake activation: 2 ms (direct GPIO write to brake relay);

— sensor-state storage: 1 ms (DMA transfer to non-volatile buffer);

— communication-mode switch: 3 ms (CAN-controller register update);

— total parallel WCET*:* max(2, 1, 3) + overhead = 5 ms < 10 ms.

2. Phase 3 — Sequential Recovery Ordering.

The recovery sequence (Sensors → Communication → Motors) is dictated by data dependencies: sensors must initialize first in order to provide valid position feedback; communication must be restored before motor commands can be transmitted; and motors are restored last, since they require both sensor data and communication. Each step verifies its prerequisites ($\rho > threshold$) before proceeding, implementing a barrier-synchronisation pattern that prevents premature recovery attempts.

3. Phase 4 — Lyapunov Stability Verification.

The Lyapunov derivative is computed on-line using the estimated state and the precomputed Lyapunov matrix P:

$$dV/dt \approx (V(x_{k+1}) - V(x_k))/\Delta t = ({x_{k+1}}^T P\, x_{\,k+1} - {x_k}^T P\, x_k)/\Delta t. \quad (25)$$

The system transitions to NORMAL only after dV/dt < 0 is sustained for 1 s (100 samples at 100 Hz), providing statistical confidence that stability has been recovered rather than that the controller is responding to transient fluctuations.

Algorithm 4. Coordinated emergency response

```
function EmergencyCoordinator( V_psu, states ):
  // Phase 1: immediate protection ( < 10 ms )
  if V_psu < V_critical then
      parallel:
          Motors  : EnableBrakeHold() ;  SetTorque(0)
          Sensors : StoreLastValidPosition()
          Comm    : SwitchToHeartbeatOnly()
  // Phase 2: monitor for recovery opportunity
  while V_psu < V_nominal do
      S[k] <- ComputeSurvivability()
      if S[k] < S_min then TriggerSafeStop() ; return
      Wait(10 ms)
  // Phase 3: sequential recovery
```

```
  if V_psu > V_recovery then
      Sensors.Reinitialize()    -> verify rho_sensor > 0.7
      Comm.RestoreFullMode()    -> verify rho_comm   > 0.8
      Motors.GradualRestore(10 %/s) -> verify stable
      ReleaseBrake() when all rho > 0.8
  // Phase 4: stability verification
  dV_dt <- ComputeLyapunovDerivative( x[k], P )
  if dV_dt < 0 sustained for 1 s then SetState(NORMAL)
return recovery_status
```

The parallel execution in Phase 1 is critical: all protection actions must be completed within 10 ms in order to prevent hardware damage during voltage transients. The sequential recovery in Phase 3 prevents the control instability that would otherwise occur if motors were restored before sensors could provide valid feedback.

6. Software architecture and scalability

A. Design Principles

The software architecture follows three core principles: 1) separation of concerns via modular components with well-defined interfaces; 2) computational scalability enabling deployment from a single-microcontroller embedded system to a distributed multi-agent configuration; 3) configurability through external JSON parameters, supporting rapid prototyping and field tuning without recompilation.

B. Computational Complexity Analysis

Table 10 summarizes the computational complexity of each algorithm component. The analysis assumes n state dimensions, m control inputs and r resources.

Table 10. Computational complexity by component

Component	Time	Space	Parallelisable
State estimation (Kalman)	$O(n^3)$	$O(n^2)$	Matrix ops: yes
Indicator computation	$O(r)$	$O(r)$	Reduction: yes
Regime classification	$O(1)$	$O(1)$	–
Control computation	$O(n \times m)$	$O(n \times m)$	Matrix-vector: yes
Degradation update	$O(r)$	$O(r)$	Per-resource: yes
Total per time-step	$O(n^3 + r)$	$O(n^2 + r)$	–

For the scalar system ($n = 1$) used in the simulations, the per-time-step complexity is reduced to $O(r)$, which enables real-time execution at 1 kHz on typical embedded processors. For high-dimensional systems the Kalman filter dominates the complexity; a precomputed observer gain can reduce this to $O(n^2)$ at the cost of optimality.

C. Scalability Mechanisms

1. Resource Scalability.

The system scales to an arbitrary number of resources r through a hash-map data structure (e.g. `std::unordered_map<std::string, double>`) with $O(1)$ average-case access time. Resource updates execute in $O(r)$ with trivial parallelisation across cores. For

systems with $r > 1000$ resources (e.g. large sensor networks), GPU-accelerated batch updates achieve $O(r/p)$ with p parallel processors.

2. Distributed Deployment.

The modular architecture supports distributed deployment across multiple nodes. Three groups of components are distinguished:

Node-local components (Plant, Controller and DegradationModel) execute on the embedded controller of each individual subsystem.

Centralised components (Supervisor and MissionManager) execute on a central coordinator.

Inter-node communication transmits the indicators S_k, P_k and F_k at 100 Hz (12 bytes per message), together with regime commands (s_k, q_k) at 10 Hz (2 bytes per message).

This architecture has been validated for integration with ROS 2 and the DDS middleware, enabling deployment on heterogeneous robotic systems [16].

3. Real-Time Guarantees.

The implementation is designed for POSIX real-time systems with the guarantees summarized in Table 11.

Table 11. Real-time performance characteristics

Metric	Measured Value	Requirement
Control loop WCET (ARM Cortex-M7)	42 μs	< 100 μs (1 kHz loop)
Indicator computation	8 μs	< 100 μs
Emergency response latency	4,2 ms	< 10 ms (Phase 1)
Memory footprint (code + data)	48 KB	< 256 KB
Stack usage (worst case)	2,1 KB	< 8 KB

7. Simulation results and analysis

A. Comparative Effectiveness

Table 12 summarizes the comparative performance of adaptive control versus uncontrolled operation across the four scenarios. The metrics quantify the benefit of the proposed approach in terms of operational continuity, survivability and hardware protection.

Key observations follow from the comparative analysis:

1) safe operation time increases by 40–76 % across all scenarios in which failure eventually occurs (A, B, C). Scenario D demonstrates an infinite extension through successful recovery;

2) minimum survivability $S(t)$ improves by +0,25 to +0,68, with the largest improvement in Scenario A where thermal management prevents complete motor failure;

3) hardware protection is achieved in 100 % of scenarios: no permanent damage occurs when adaptive control is active, compared to likely damage in uncontrolled operation;

4) recovery from super-critical states is achieved in 75 % of scenarios (A, C, D); Scenario B achieves only partial recovery because of permanent encoder damage.

Table 12. Control effectiveness: Adaptive vs Uncontrolled

Metric	Scenario A	Scenario B	Scenario C	Scenario D
Time to failure (no ctrl)	340 s	150 s	400 s	1.5 s
Time to failure (with ctrl)	> 600 s	210 s	> 600 s	∞ (recovered)
Extension	+76 %	+40 %	+50 %	+∞
min $S(t)$ without control	0,00	0,00	0,00	0,08
min $S(t)$ with control	0,68	0,25	0,58	0,52
$S(t)$ improvement	+0,68	+0,25	+0,58	+0,44
Recovery achieved	Yes	Partial	Yes	Yes
Hardware protected	Yes	Yes	Yes	Yes

B. Scenario-Specific Analysis

1. Scenario A — Thermal Degradation.

The thermal-protection controller successfully maintains the motor temperature below the shutdown threshold (115 °C) throughout the simulation. The peak temperature with control (88 °C) is 30 % lower than without control (125 °C). The load-redistribution mechanism activates at $t = 210$ s, when the margin drops below 0.3, transferring 40 % of M3's load to the adjacent motors M2 and M4.

Recovery is achieved after forced cooling is activated at $t = 260$ s. The survivability function recovers from $S = 0{,}42$ (super-critical) to $S = 0{,}84$ (normal) over 220 s, which demonstrates the reversibility of thermal degradation when appropriate control actions are taken in time.

2. Scenario B — Sensor Cascade.

The sensor-fusion algorithm successfully maintains a bounded position error (15 mm maximum), compared to complete position loss in the uncontrolled case. The adaptive Kalman filter automatically increases the IMU weight from 0,20 to 0,90 as the encoder quality degrades.

Only partial recovery is achieved because encoder damage is permanent ($\rho_{enc} = 0$ after fault). The controlled system, however, executes a safe-stop sequence rather than crashing, preserving data integrity and enabling post-incident analysis.

3. Scenario C — Communication.

The communication failover successfully transitions from CAN1 to CAN2 at $t =$ $= 400$ s when the primary bus bit error rate exceeds 10^{-3}. The message loss is reduced from 100 % (uncontrolled after CAN1 failure) to 10 % (controlled on CAN2).

The control loop continues to operate in degraded mode with a latency of 30 ms, compared to infinite latency (timeout) without failover. This demonstrates the importance of redundant communication paths in safety-critical systems.

4. Scenario D — Multi-Component Cascade.

This scenario provides the most dramatic demonstration of the benefits of adaptive control. The coordinated emergency response prevents total system failure during a *16 V* power sag that would otherwise cause cascading failures across all subsystems.

The Lyapunov-stability indicator dV/dt transitions from +0,95 (unstable) at $t = 1{,}0$ s to −0,08 (stable) at $t = 2{,}0$ s under coordinated control, compared to +0,85 (worsening

instability) in the uncoordinated case. Full recovery to normal operation (S = 0,91) is achieved in 13 s.

C. Control-Mechanism Effectiveness

Table 13 analyzes the contribution of the individual control mechanisms across scenarios. The assessment (HIGH/MED/LOW/N/A) reflects each mechanism's importance for a successful outcome in the corresponding scenario.

Table 13. Control-mechanism contribution by scenario

Mechanism	A	B	C	D	Critical?
Adaptive gain reduction	HIGH	MED	LOW	HIGH	Yes
Sensor fusion / fallback	LOW	HIGH	N/A	HIGH	Yes
Load redistribution	HIGH	N/A	N/A	MED	Scenario-dep.
Communication failover	N/A	LOW	HIGH	MED	Yes
Emergency braking	MED	MED	LOW	HIGH	Yes
Lyapunov monitoring	MED	MED	MED	HIGH	Yes

The analysis reveals that no single control mechanism is sufficient for all scenarios. Adaptive gain reduction and Lyapunov monitoring provide benefit across all cases, whereas other mechanisms are scenario-specific. This observation supports the hierarchical, multi-mechanism approach adopted in this work.

D. Lyapunov Stability Analysis

Table 14 presents the evolution of the Lyapunov function and its derivative during Scenario D recovery, validating the stability properties of the coordinated response.

Table 14. Lyapunov stability during scenario d recovery

t, s	$V(x)$	dV/dt	$\|x\|$	$\lambda_{min}(P)$	Status
0.5	2.45	+0.82	1.52	0.85	UNSTABLE
1.0	3.28	+0.55	1.78	0.78	UNSTABLE
2.0	3.12	−0.15	1.68	0.88	RECOVERING
5.0	1.85	−0.22	1.25	0.92	STABLE
15.0	0.42	−0.05	0.58	0.95	STABLE

The transition from positive to negative dV/dt at t ≈ 2 *s* marks the critical point at which the system recovers stability. The minimum eigenvalue $\lambda_{min}(P)$ of the Lyapunov matrix increases during recovery, indicating improved stability margins as the system returns to normal operation.

E. Summary Statistics

Table 15 provides aggregate statistics across all scenarios, demonstrating the consistent benefit of adaptive control.

Table 15. Aggregate performance statistics

Metric	Min	Max	Average
Safe-time extension	+40 %	+∞	> 55 %
S(t) improvement	+0,25	+0,68	+0,49
Recovery rate	50 %	100 %	75 %
Hardware-protection rate	100 %	100 %	100 %

8. Discussion

A. Practical Implications

The simulation results have several important implications for practical system design. **First**, the consistent hardware protection across all scenarios validates the fundamental premise that controlled degradation is preferable to uncontrolled failure. Even when full recovery is not achieved (Scenario B), the adaptive control system prevents permanent damage and enables graceful shutdown with preserved diagnostic data.

Second, the regime-dependent gain scheduling proves essential for balancing performance and stability. The aggressive gains ($K_N = 0{,}8$) appropriate for normal operation would cause instability in degraded conditions, whereas the conservative super-critical gains ($K_{SK} = 0{,}2$) would provide inadequate performance in normal operation.

Third, the multi-mechanism approach addresses the diversity of failure modes encountered in real systems. No single control strategy handles all scenarios effectively — thermal management requires different interventions from those required for sensor fusion or communication failover.

B. Comparison with Prior Work

The proposed framework extends prior work on fault-tolerant control [5–9] and survivable systems [2–4] by explicitly addressing the super-critical regime in which conventional approaches fail. Compared to traditional redundancy-based methods, the present approach provides continuous operation during transient failures rather than immediate failover; a graduated response proportional to the degradation severity; explicit stability guarantees via Lyapunov analysis (Theorem 1); and quantitative survivability metrics for design optimization.

C. Limitations

Several limitations should be acknowledged. The scalar plant model, although it captures the essential dynamics, does not represent the full complexity of multi-DOF systems with coupled dynamics. Extension to higher-dimensional systems requires addressing computational constraints for real-time Riccati solving.

The degradation models assume known failure mechanisms with predictable progression. In practice, unexpected failure modes and common-cause failures may invalidate the parametric assumptions. On-line parameter adaptation would improve robustness to model uncertainty.

The simulation studies do not include measurement noise, actuator saturation or communication delays that would affect real implementations. Hardware-in-the-loop validation is required before deployment in safety-critical applications.

9. Conclusion

In this paper the author has presented a comprehensive framework for adaptive control and survivability management of super-critical cyber-physical systems. The main contributions are:

1) a hierarchical control architecture with mathematically defined regime transitions based on functional-integrity indicators, enabling a systematic response to varying degradation levels;

2) detailed parametric models for four realistic degradation scenarios with validated parameters, providing a foundation for simulation-based design and testing;

3) complete control algorithms with tuning guidelines, a formal proof of bounded evolution (Theorem 1) and explicit Lyapunov-based stability margins, demonstrating a practical implementation path from theory to software;

4) a modular C++ framework with JSON configuration, enabling rapid prototyping and parameter studies.

Simulation results demonstrate that the proposed adaptive strategies extend safe operation time by 40–76 %, improve minimum survivability by +0.25 to +0.68, and enable recovery from super-critical states in 75 % of the tested scenarios. Hardware protection is achieved in 100 % of cases, validating the core premise that controlled degradation prevents permanent damage.

The framework provides a practical foundation for deploying survivability-aware control in robotic, automotive and industrial-automation applications where graceful degradation is preferable to immediate shutdown.

A. Future Research Directions

Several extensions are planned as future research:

1) machine-learning-based prediction enabling earlier intervention before degradation progresses to super-critical levels;

2) formal verification against ISO 26262 ASIL-D requirements for automotive applications;

3) extension to distributed multi-agent systems with coordinated degradation management;

4) hardware-in-the-loop validation on representative robotic platforms;

5) joint treatment of safety and cybersecurity in the presence of adversarial perturbations (cf. [14, 15]).

1. J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In Automata Studies, C. E. Shannon and J. McCarthy, Eds. Princeton, NJ, USA: Princeton Univ. Press, 1956, H. 43–98.

2. Zhang D., Feng G., Shi Y., and Srinivasan D. Physical safety and cyber security analysis of multi-agent systems: A survey of recent advances. *IEEE/CAA J. Autom. Sinica*. 2021, Feb. Vol. 8, No. 2. P. 319–333, doi: 10.1109/JAS.2021.1003820.

3. D. Ratasich, F. Khalid, F. Geissler, R. Grosu, M. Shafique, and E. Bartocci, "A roadmap toward the resilient Internet of Things for cyber-physical systems," *IEEE Access*, vol. 7, pp. 13 260–13 283, 2019, doi: 10.1109/ACCESS.2019.2891969.

4. S. Colabianchi, F. Costantino, G. Di Gravio, F. Nonino, and R. Patriarca, "Discussing resilience in the context of cyber-physical systems," *Comput. Ind. Eng*., vol. 160, Art. no. 107534, Oct. 2021, doi: 10.1016/j.cie.2021.107534.

5. M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, Diagnosis and Fault-Tolerant Control, 3rd ed. Berlin, Germany: Springer, 2016, doi: 10.1007/978-3-662-47943-8.

6. H. Zhang, Y. Liang, H. Su, and C. Liu, "Event-driven guaranteed cost control design for nonlinear systems with actuator faults via reinforcement learning algorithm," *IEEE Trans. Syst., Man, Cybern., Syst*., vol. 50, no. 11, pp. 4135–4150, Nov. 2020, doi: 10.1109/TSMC.2019.2946857.

7. H. Jiang, H. Zhang, Y. Liu, and J. Han, "Neural-network-based control scheme for a class of nonlinear systems with actuator faults via data-driven reinforcement learning method," *Neurocomputing*, vol. 239, pp. 1–8, May 2017, doi: 10.1016/j.neucom.2017.02.004.

8. L. Wang, J. Song, R. Zhang, and F. Gao, "Constrained model predictive fault-tolerant control for multi-time-delayed batch processes with disturbances: A Lyapunov–Razumikhin function method," *J. Franklin Inst*., vol. 358, no. 18, pp. 9483–9509, Dec. 2021, doi: 10.1016/j.jfranklin.2021.09.028.

9. X. Li, Q. Luo, L. Wang, R. Zhang, and F. Gao, "Off-policy reinforcement learning-based novel model-free minmax fault-tolerant tracking control for industrial processes," *J. Process Control*, vol. 115, pp. 145–156, Jul. 2022, doi: 10.1016/j.jprocont.2022.05.006.

10. M. H. Cohen and C. Belta, Safe exploration in model-based reinforcement learning using control barrier functions. *Automatica*. Jan. 2023, vol. 147, Art. no. 110684, doi: 10.1016/j.automatica.2022.110684.

11. M. H. Cohen and C. Belta, "High order robust adaptive control barrier functions and exponentially stabilizing adaptive control Lyapunov functions," in Proc. Amer. Control Conf. (ACC), Atlanta, GA, USA, 2022, pp. 2233–2238, doi: 10.23919/ACC53348.2022.9867633.

12. Lopez B.T., and J.-J. E. Slotine. Universal adaptive control of nonlinear systems. *IEEE Control Syst. Lett*. 2022. Vol. 6. P. 1826–1830. doi: 10.1109/LCSYS.2021.3133359.

13. G.-P. Liu, "Control strategies for digital twin systems," *IEEE/CAA J. Autom. Sinica*, vol. 11, no. 1, pp. 170–180, Jan. 2024, doi: 10.1109/JAS.2023.123834.

14. X. Ge, Q.-L. Han, M. Zhong, and X.-M. Zhang, "Distributed Krein space-based attack detection over sensor networks under deception attacks," *Automatica*, vol. 109, Art. no. 108557, Nov. 2019, doi: 10.1016/j.automatica.2019.108557.

15. Z. Feng and G. Hu, Secure cooperative event-triggered control of linear multiagent systems under DoS attacks. *IEEE Trans. Control Syst. Technol*., vol. 28, no. 3, pp. 741–752, May 2020, doi: 10.1109/TCST.2019.2892032.

16. Y. Tian, Y. Chang, F. Herrera Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-Multi: Robust, distributed, dense metric–semantic SLAM for multi-robot systems," *IEEE Trans. Robot*., vol. 38, no. 4, pp. 2022–2038, Aug. 2022, doi: 10.1109/TRO.2021.3137751.

17. D. Humennyi, O. Humennyi, and Y. Shabala, The super-critical operational modes in robotic systems. *Underwater Technol., Indust. Civ. Eng*. (Pidvodni Tehnologii), iss. 13, pp. 60–66, 2023, doi: 10.32347/uwt.2023.13.1301.

18. Humennyi D. Ensuring survivability of complex super-critical systems based on hierarchical abstraction model, Ph.D. dissertation, Dept. Cybersecurity, Kyiv Nat. Univ. Construction Archit., Kyiv, Ukraine, 2024.

19. Khalil H.K. Nonlinear Systems, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

Received 25.12.2025
Accepted 19.05.2026
Published 17.06.2026