Model-based design for increasing reliability and safety of autonomous systems

Italian National Ph.D. Program in Autonomous Systems

ACADEMIC DISCIPLINE: SYSTEMS AND CONTROL ENGINEERING (IINF-04/A)

Final Dissertation

Model-based Design For Increasing Reliability and Safety of Autonomous Systems

---

by

Shafqat Ali Siyyal

Administrative Headquarters:

Politecnico di Bari – Department of Electrical and Information Engineering

Hosting University:

Università Politecnica delle Marche – Department of Information Engineering

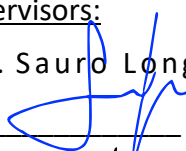
Referees:

Prof. John Ringwood

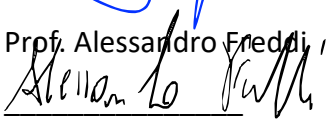
Prof. Francesco Liberati

Supervisors:

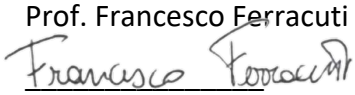
Prof. Sauro Longhi



Prof. Alessandro Freddi

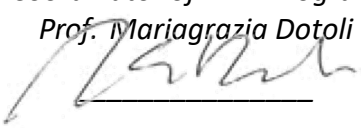


Prof. Francesco Ferracuti



Coordinator of Ph.D Program

Prof. Mariagrazia Dotoli



Course no. 38, 01/11/2022-31/10/2025

Politecnico di Bari

LIBERATORIA PER L'ARCHIVIAZIONE DELLA TESI DI DOTTORATO

Al Magnifico Rettore
del Politecnico di Bari

Il sottoscritto Shafqat Ali Siyyal_____ nato a Naushahro Feroze (EE)___ il 03/04/1996___

residente a Ancona_ in via Carlo Crivelli N.1 ___ e-mail mshafqat00@gmail.com_____

iscritto al 3° anno di Corso di Dottorato di Ricerca in AUTONOMOUS SYSTEMS_ ciclo 38_

ed essendo stato ammesso a sostenere l'esame finale con la prevista discussione della tesi dal titolo:

Model-based Design For Increasing Reliability and Safety of Autonomous Systems__

DICHIARA

1) di essere consapevole che, ai sensi del D.P.R. n. 445 del 28.12.2000, le dichiarazioni mendaci, la falsità negli atti e l'uso di atti falsi sono puniti ai sensi del codice penale e delle Leggi speciali in materia, e che nel caso ricorressero dette ipotesi, decade fin dall'inizio e senza necessità di nessuna formalità dai benefici conseguenti al provvedimento emanato sulla base di tali dichiarazioni;
2) di essere iscritto al Corso di Dottorato di ricerca AUTONOMOUS SYSTEMS_ciclo_38, corso attivato ai sensi del "*Regolamento dei Corsi di Dottorato di ricerca del Politecnico di Bari"*, emanato con D.R. n.286 del 01.07.2013;
3) di essere pienamente a conoscenza delle disposizioni contenute nel predetto Regolamento in merito alla procedura di deposito, pubblicazione e autoarchiviazione della tesi di dottorato nell'Archivio Istituzionale ad accesso aperto alla letteratura scientifica;
4) di essere consapevole che attraverso l'autoarchiviazione delle tesi nell'Archivio Istituzionale ad accesso aperto alla letteratura scientifica del Politecnico di Bari (IRIS-POLIBA), l'Ateneo archivierà e renderà consultabile in rete (nel rispetto della Policy di Ateneo di cui al D.R. 642 del 13.11.2015) il testo completo della tesi di dottorato, fatta salva la possibilità di sottoscrizione di apposite licenze per le relative condizioni di utilizzo (di cui al sito http://www.creativecommons.it/Licenze), e fatte salve, altresì, le eventuali esigenze di "embargo", legate a strette considerazioni sulla tutelabilità e sfruttamento industriale/commerciale dei contenuti della tesi, da rappresentarsi mediante compilazione e sottoscrizione del modulo in calce (Richiesta di embargo);
5) che la tesi da depositare in IRIS-POLIBA, in formato digitale (PDF/A) sarà del tutto identica a quelle **consegnate**/inviate/da inviarsi ai componenti della commissione per l'esame finale e a qualsiasi altra copia depositata presso gli Uffici del Politecnico di Bari in forma cartacea o digitale, ovvero a quella da discutere in sede di esame finale, a quella da depositare, a cura dell'Ateneo, presso le Biblioteche Nazionali Centrali di Roma e Firenze e presso tutti gli Uffici competenti per legge al momento del deposito stesso, e che di conseguenza va esclusa qualsiasi responsabilità del Politecnico di Bari per quanto riguarda eventuali errori, imprecisioni o omissioni nei contenuti della tesi;
6) che il contenuto e l'organizzazione della tesi è opera originale realizzata dal sottoscritto e non compromette in alcun modo i diritti di terzi, ivi compresi quelli relativi alla sicurezza dei dati personali; che pertanto il Politecnico di Bari ed i suoi funzionari sono in ogni caso esenti da responsabilità di qualsivoglia natura: civile, amministrativa e penale e saranno dal sottoscritto tenuti indenni da qualsiasi richiesta o rivendicazione da parte di terzi;
7) che il contenuto della tesi non infrange in alcun modo il diritto d'Autore né gli obblighi connessi alla salvaguardia di diritti morali od economici di altri autori o di altri aventi diritto, sia per testi, immagini, foto, tabelle, o altre parti di cui la tesi è composta.

Luogo e data _Bari, 12/04/2026 Firma _____

Il/La sottoscritto, con l'autoarchiviazione della propria tesi di dottorato nell'Archivio Istituzionale ad accesso aperto del Politecnico di Bari (POLIBA-IRIS), pur mantenendo su di essa tutti i diritti d'autore, morali ed economici, ai sensi della normativa vigente (Legge 633/1941 e ss.mm.ii.),

CONCEDE

- al Politecnico di Bari il permesso di trasferire l'opera su qualsiasi supporto e di convertirla in qualsiasi formato al fine di una corretta conservazione nel tempo. Il Politecnico di Bari garantisce che non verrà effettuata alcuna modifica al contenuto e alla struttura dell'opera.
- al Politecnico di Bari la possibilità di riprodurre l'opera in più di una copia per fini di sicurezza, back-up e conservazione.

Luogo e data _Bari, 12/04/2026 Firma _____

Shafqat Ali Siyyal

Model-based Design For Increasing Reliability and Safety of Autonomous Systems

Thesis submitted for the degree of Philosophiae Doctor

Italian National Ph.D. Program in Autonomous Systems
Università Politecnica delle Marche

Tutors
Prof. *Sauro Longhi*
Prof. *Alessandro Freddi*
Prof. *Francesco Ferracuti*

2026

The doctoral scholarship was funded by the European Union - Next Generation EU, Mission 4 Component [1] CUP [*D*93*C*22000850005].

Dissertation submitted for the degree of *Philosophiae Doctor*
Italian National Ph.D. Program in Autonomous Systems

Cycle:
38th

Administrative Headquarters:
Politecnico di Bari

Hosting University:
Università Politecnica delle Marche

Title:
Model-based Design For Increasing Reliability and Safety of Autonomous Systems

Ph.D Candidate:
Shafqat Ali Siyyal, Università Politecnica delle Marche (Ancona, Italy)

Tutors:
Prof. Sauro Longhi, Università Politecnica delle Marche (Ancona, Italy)
Prof. Alessandro Freddi, Università Politecnica delle Marche (Ancona, Italy)
Prof. Francesco Ferracuti, Università Politecnica delle Marche (Ancona, Italy)

Coordinator:
Prof. Engr. Mariagrazia Dotoli, Politecnico di Bari (Bari, Italy)

External Reviewers:
Prof. John Ringwood, Maynooth University (Maynooth, Ireland)
Prof. Francesco Liberati, Sapienza University of Rome (Rome, Italy)

Last version:
April 15, 2026

Abstract

This thesis presents a model-based design framework to enhance the safety and security of autonomous systems against cyber attacks. As modern autonomous systems, particularly Cyber-Physical Systems (CPS), become increasingly interconnected, they face growing threats that differ significantly from conventional IT network attacks. These control theoretic attacks allow an intelligent adversary to manipulate physical processes by compromising sensors or actuators, leading, possibly, to catastrophic failures. This strategic and adversarial nature is what distinguishes these threats from conventional system faults, which are typically uncertain and non-malicious events.

This research begins with a thorough analysis of such attacks, examining their theoretical models and studying real world case studies to understand the entire attack chain, from initial access to the manipulation of physical actuators.

To model and understand attack impacts, an antagonistic Model Predictive Control (MPC) framework is first developed. Initially, this framework explores how an attacker can destabilize a system by reformulating the controller's objective from cost minimization to maximization. To ensure the validity of this model in realistic environments, the framework is extended to a robust antagonistic MPC. By using the max-min optimization approach, this formulation accounts for worst-case environmental disturbances, guaranteeing a certain level of damage even under unfavorable scenarios. This concept is then further refined into a strategic constraint-violation attack model. Instead of simply maximizing a generic cost function, this formulation explicitly prioritizes the violation of system constraints, thereby generating attack vectors that are specifically designed to force system failure.

To counter the strategic threats modeled by this framework, this thesis develops a proactive attack mitigation framework. SoftWare Rejuvenation (SWR), a technique that resets a compromised controller to a safe, pre-defined state, as a potent mitigation and recovery strategy. In contrast to traditional periodic rejuvenation, which can be disruptive and inefficient, this research introduces a framework where rejuvenation is triggered by dedicated monitoring tools based on attack detection and prediction. Two such triggers are developed: (1) a residue based detector, utilizing a Luenberger like observer, which initiates rejuvenation upon detecting anomalies and (2) a proactive monitor based on the antagonistic MPC concept, which calculates the Time-to-Violation T^*. This novel metric represents the minimum time an attacker would need to violate system constraints from the current state. If T^* falls below a critical safety threshold, rejuvenation is preventively triggered as a crucial mitigation.

Finally, this thesis integrates these contributions into a comprehensive safety framework. This framework leverages the proposed detectors to intelligently trigger software rejuvenation, thereby providing robust defense mechanism that can effectively mitigate attacks and ensure operational safety. The efficacy and practicality of the proposed methods are validated through a numerical simulations on both highly dynamic (quadrotor) and slow dynamic (four tank system) models, showing their broad applicability in safeguarding modern autonomous systems.

All'avvenire

Acknowledgments

This PhD has been more than just an academic pursuit; it was a journey that opened doors to new experiences and relationships. I am grateful for the entire process.

I would like to extend my sincere gratitude to my supervisors: Prof. Sauro Longhi, Prof. Alessandro Freddi, and Francesco Ferracuti.

I offer special thanks to Prof. Alessandro Freddi for his constant support throughout this process. I am grateful for his kindness and accessible mentorship, which made this journey possible and manageable. I also thank Francesco Ferracuti for his academic insights and for his guidance in navigating the administrative and procedural aspects of this doctorate.

A very special thanks goes to Alessandro Baldini for his support during the initial years of my research. Thank you for helping me build the necessary background to understand complex concepts and for providing the foundation that made my later work possible.

I am also deeply grateful to my host research group at the University of Seville, Spain. My sincere thanks to Prof. José María Maestre for giving me the incredible opportunity to join his lab and collaborate with his team.

Lastly, my profound thanks go to my wife, Maira Majeed, for her encouragement and support throughout my doctoral studies.

Contents

Acknowledgments **iv**

Acronyms **xii**

Introduction

1 Overview and Motivation **2**
1.1 The rise of Autonomous Cyber Physical Systems 2
1.2 Problem Statement 3
1.3 Research Objectives 5
1.4 Thesis Outline 6

Part 1

2 Background and Literature Review **9**
2.1 Introduction 9
2.2 Real-World Cyber-Physical Attacks 9
2.2.1 Distinction between IT and OT Security 9
2.2.2 Case Study 1: Stuxnet (2010) 9
2.2.3 Case Study 2: Ukraine Power Grid (2015/2016) 10
2.2.4 Case Study 3: Havex (Dragonfly) Campaign 10
2.2.5 Case Study 4: Colonial Pipeline Ransomware (2021) . . . 11
2.2.6 Case Study 5: PIPEDREAM/INCONTROLLER (2022) . 11
2.2.7 Lessons Learned 11
2.3 CPS System Modeling 12
2.3.1 General Architecture 12
2.3.2 Mathematical System Dynamics 12
2.4 Threat Landscape and Adversary Model 14
2.4.1 Adversary Capabilities 14
2.4.2 Taxonomy of Attacks 15
2.5 Mathematical Formulation of Attacks 16
2.5.1 Actuator Side Attack Model 16
2.5.2 Sensor Side Attack Model 17
2.5.3 Combined and Stealthy Formulation 18
2.6 Defense Mechanisms in Model-Based Control Systems 18
2.7 Fundamental of Software Rejuvenation 19
2.7.1 Background and Concepts 19
2.7.2 Modes of Operation and Rejuvenation Sequence 20
2.7.3 Related Work and Limitations of SWR 21

3 System Modeling and Control Preliminaries **23**
3.1 Introduction 23
3.2 The Quadrotor System 23
3.2.1 Kinematics 24
3.2.2 Rigid Body Dynamics 24
3.2.3 Forces and Moments 25
3.2.4 Simplified Model for Control 26
3.2.5 Motor Mixer Algorithm 26
3.2.6 Linearized State-Space Model 26

3.3 The Four-Tank System . . . 29
3.3.1 Nonlinear Mathematical Model . . . 29
3.3.2 Linearized State-Space Model . . . 30
3.3.3 System Parameters and Configuration . . . 31
3.4 Fundamentals of Model Predictive Control (MPC) . . . 31
3.4.1 Mathematical Formulation of MPC . . . 32
3.4.2 System Dynamics . . . 32
3.4.3 The Cost Function . . . 32
3.4.4 Prediction Modeling . . . 33
3.4.5 Quadratic Programming Derivation . . . 33
3.4.6 The Constrained Optimization Problem . . . 34
3.5 Robust Model Predictive Control . . . 35
3.5.1 The Naive Approach . . . 36
3.5.2 The Conservative Approach . . . 37
3.6 Explicit MPC via Multi-Parametric Programming (mP-QP) . . . 40

Part 2

4 Antagonistic MPC Control 43
4.1 Introduction . . . 43
4.1.1 Operational Applications of the Framework . . . 44
4.1.2 Chapter Roadmap . . . 44
4.2 Antagonistic Control Framework . . . 45
4.2.1 Problem Formulation . . . 45
4.2.2 Solver for Antagonistic MPC . . . 45
4.2.3 Case Study: Quadrotor UAV . . . 46
4.3 Robust Antagonistic MPC . . . 50
4.3.1 Limitations of the Naive Approach . . . 50
4.3.2 Limitations of the Conservative Approach . . . 50
4.3.3 Proposed Method: Disturbance-Aware Antagonistic Controller . . . 51
4.4 Constraint-Aware Antagonistic MPC . . . 53
4.4.1 Motivation: From Maximization to Violation . . . 54
4.5 Solution Via Vertex Enumeration . . . 57
4.5.1 Geometric Interpretation of Optimal Control Vertices . . . 57
4.5.2 Example: Visualizing the Search Space and Complexity . . . 57
4.5.3 Numerical Analysis . . . 59
4.6 Discussion . . . 61

5 Condition-Based Software Rejuvenation Framework 63
5.1 Introduction . . . 63
5.1.1 Operational Limitations of Periodic SWR (The Always-On Cost) . . . 63
5.1.2 The Shift to Reactive and Proactive Triggering . . . 64
5.1.3 Chapter Contributions and Organization . . . 64
5.2 Architecture of the CB-SWR Framework . . . 64
5.3 Monitor I: Observer-Based Monitor . . . 65
5.4 Monitor II: Antagonistic MPC-based Predictor . . . 68
5.5 Logic Decision Module . . . 69
5.6 Case Study: Quadrotor Unmanned Aerial Vehicle (UAS) . . . 70
5.6.1 System Overview and Modeling . . . 70
5.6.2 Implementation of Observer-based Monitor (Rotational Observer) . . . 70
5.6.3 Implementation of Predictive Monitor (Attitude Predictor) 71
5.6.4 Simulation Setup and Results . . . 71
5.7 Case Study II: The Four-Tank System (Process Control) . . . 77

5.7.1 Implementation of Observer-based Monitor (Monitor-I) . 77
5.7.2 Implementation of Predictive Monitor (Monitor II) 78
5.7.3 Simulation Setup and Results 78
5.8 Discussion and Conclusion 83

Conclusions

6 Conclusion **85**
6.1 Summary of Research 85
6.2 Research Contributions 85
6.3 Future Work 86

List of Figures

1.1 Generic architecture illustrating the core components and integration of diverse domains within modern Cyber-Physical Systems (AI generated) . 2

2.1 A general architecture of a CPS [3] 12
2.2 Schematic of an actuator integrity attack, modified version from [3] . . . 17
2.3 Schematic of a sensor integrity attack where an adversarial signal $a_y(t)$ corrupts the measurement vector, modified version from [3] 17

3.1 Quadrotor schematic illustrating the inertial (O_E) and body (O_B) coordinate frames, gravitational force (F_g), individual rotor thrusts (f_i), and motor rotation directions in a cross configuration 25
3.2 Schematic of the quadruple tank system illustrating the flow paths. The system inputs are the pump voltages (v_1, v_2) and the system outputs are the water levels in the lower tanks (y_1, y_2). The valves distribute flow between the lower and upper tanks to create cross-coupling effects 29
3.3 Illustration of the MPC strategy, showing the measured past states, the computed future control sequence over the prediction horizon, and the predicted output tracking a reference trajectory, modified version from [82] 32
3.4 Feasible regions in the state-input space (x, u) for the SISO integrator model. The Light Red area indicates the original feasible set for the nominal system without disturbances. The Green area indicates the robust feasible set after constraint tightening. The difference between the two regions represents the safety buffer or lost feasibility required to guarantee safety against the worst-case disturbance $w \in [-1, 1]$ 39

4.1 Quadrotor inner-outer control loop architecture 47
4.2 Outer loop position tracking under nominal conditions (state in black, reference in red) 49
4.3 Cost function J_I 49
4.4 Inner loop attitude tracking under nominal conditions (state in black, reference in red) 49
4.5 Inner loop: detail of the attack (state in black, reference in red) 50
4.6 Antagonistic MPC: predicted variables (dashed line) and actual variables (solid line) 50
4.7 Comparative histogram of attack impact (damage magnitude) under stochastic wind disturbances ($N_{sim} = 100$ simulations). The Naive strategy (blue) shows high variance, achieving peak damage only under favorable wind conditions but failing when disturbances oppose the attack. In contrast, the Robust strategy (red) demonstrates a guaranteed performance floor, ensuring consistent system destabilization by mitigating the worst-case disturbance effects 53
4.8 Visual representation of the quadratic cost function relative to the operational constraints. The red markers indicate the absolute cost values at the lower and upper bound, highlighting the numerical difference between the nearest bound ($x = -2$) and the farthest bound ($x = 3$) from the current state 55

4.9 Maximization-based Method: System trajectory under the standard cost-maximization method. Following the attack initiation at $t = 3$s, the adversary targets the upper bound ($x_{max} = 3$) due to the higher potential cost, passing through the full state space rather than exploiting the closer vulnerability at the lower bound 55
4.10 Violation-Based Method: System trajectory using the proposed formulation. The adversary identifies the nearest boundary ($x_{min} = -2$) and selects inputs ($u = -2$) to force the state trajectory into the violation region immediately after $t = 3$s 56
4.11 Geometric interpretation of optimal control vertices. Visual representation of the feasible sets ($\mathcal{U}$) and cost gradients. (a) 1D DC Motor: The cost curve (blue) intersects the voltage constraints at the boundary. (b) 2D Mobile Robot: The optimization gradient (∇J) is shown directing the solution to the top-right corner of the velocity rectangle. (c) 3D Satellite: The optimal thrust vector is highlighted at the vertex of the force cube . 58
4.12 search space evolution for a single-input system ($n_u = 1$). The labels on the edges denote the specific bang-bang control action chosen at each step. The number of unique attack sequences (leaves) doubles linearly with the horizon 58
4.13 Search space evolution for multi-input system ($n_u = 2$). At each step, four distinct boundary combinations are possible (color-coded). The branching factor is quadrupled at each step, demonstrating that increasing system inputs contributes more severely to complexity than increasing the horizon 58
4.14 Trajectory comparison for 50 Monte Carlo simulations. The Vertex Enumeration solver (right) consistently identifies the global optimum, resulting in a uniform set of trajectories that maximize system deviation. The Gradient solver (left) shows inconsistent behavior, often converging to sub-optimal local maxima 59
4.15 Average damage (position error) over time. The Vertex Enumeration method (green) consistently achieves higher system deviation than the gradient-based method (red), proving it is a more reliable tool for worst-case safety analysis 60
4.16 Quantitative comparison of solver performance on the Quadrotor UAV. The dashed green line and solid red line depict the mean performance of the Vertex and Gradient solvers, respectively, while the shaded regions indicate the variance across 50 Monte Carlo runs 61

5.1 Schematic overview of the Condition-Based SWR framework. The architecture uses a dual-monitor architecture to fuse observer-based residuals and predictive safety estimates 66
5.2 Operational timeline comparison of system modes (Nominal Operation, Rejuvenation, and Recovery) during the tracking of the reference. The top timeline depicts the Periodic SWR benchmark, showing operational fragmentation and a mission completion time. The bottom timeline shows the proposed Condition-Based SWR, which triggers rejuvenation only upon detection, reducing unnecessary interruptions 74
5.3 Dual-monitor response to pitch channel attacks. The plots shows dual vertical axes: the left axis tracks the pitch residual (r_θ) and the right axis tracks the Time-to-Violation (T^*). (a) Attack scenario 2- a gradient attack evades the residual threshold (τ_r) but triggers the predictive monitor when T^* breaches τ_{safe}. Attack scenario 1- an abrupt antagonistic attack spikes the residual, triggering the reactive monitor faster than the predictive mechanism 75

5.4 The Explicit Safety Landscape visualizing the computed Time-to-Violation (T^*) metric across the roll (ϕ) and pitch (θ) state space. The vertical axis represents the minimum time required for an optimal attacker to force a constraint violation from any given state 76
5.5 Cross-sectional validation of the explicit approximation. (Top) Comparison of the T^* values computed by the exact online solver (blue solid line) versus the explicit look-up table (red dashed line) along a test trajectory ($\theta = 0$, varying ϕ). (Bottom) The absolute approximation error introduced by the Explicit MPC formulation 76
5.6 Nominal controller's performance via Monte Carlo simulation of 50. The state trajectories and control inputs confirm stable regulation to setpoints under randomized initialization, with all inputs remaining within linear saturation bounds 79
5.7 Evaluation of periodic SWR against attacks. The state evolution confirms that water levels remain bounded by the safety constraint (black dashed line), while the control inputs depict the adversarial override and the subsequent system blackout induced by the rejuvenation trigger 80
5.8 Mission timeline depiction for the periodic SWR method. The color-coded segments represent the active system state: Nominal Operation (Green), Rejuvenation (Red), and Safety Recovery (Orange), depicts the frequency of interruptions 80
5.9 System response under Condition-Based SWR. Tank levels remain stable and within safety bounds, with the framework limiting the effects of attacks and control inputs remain continuous, with SWR interventions activated only upon attack detection 81
5.10 Mission timeline for the CB-SWR framework 82
5.11 Dual-monitor response to an abrupt antagonistic attack ($t = 50$ s). The high-magnitude input causes an immediate residual spike, triggering (τ_r) faster than the predictive metric (T^*) 82
5.12 Dual-monitor response to a slow-varying gradient attack. The slowly accumulating error evades the Residue threshold (τ_r) but causes the Time-to-Violation (T^*) to decay, triggering rejuvenation when it satisfy define safety threshold (τ_{safe}) 82

List of Tables

3.1 Quadrotor Physical and Aerodynamic Parameters 28
3.2 Model Parameters for the Four-Tank System 31

5.1 SWR Downtime Values Reported in Literature 72

Acronyms

CIA Confidentiality, Integrity, and Availability. 9
CPS Cyber-Physical Systems. ii

ESO Extended State Observer. 36

ICS Industrial Control Systems. 10

KKT Karush-Kuhn-Tucker. 41

LP Linear Programming. 21
LQR Linear Quadratic Regulator. 31
LTI Linear Time-Invariant. 13

MHE Moving Horizon Estimation. 36
MIMO Multi-Input Multi-Output. 31
MPC Model Predictive Control. ii

PID Proportional-Integral-Derivative. 31
PLC Programmable Logic Controller. 10

ROA Region of Attraction. 21
RPI Robust Positively Invariant. 37

SQP Sequential Quadratic Programming. 60
SVM Support Vector Machines. 18

TWR Thrust-to-Weight Ratio. 48

UAV Unmanned Aerial Vehicle. 7

ZOH Zero-Order Hold. 13

Introduction

Chapter 1

Overview and Motivation

1.1 The rise of Autonomous Cyber Physical Systems

The last two decades have witnessed a paradigm shift in the design and operation of engineered systems, marked by the rise of CPS. Unlike traditional embedded systems, CPS are defined by the tight integration of computational algorithms and physical processes, interconnected through communication networks that enable continuous monitoring and regulation via feedback loops, where computational decisions influence physical dynamics while remaining responsive to physical changes [1]. Much as the internet transformed how humans interact with information and one another, CPS are projected to revolutionize how we interact with and control the physical world.

Conceptually, CPS represent an intersection of foundational disciplines, including embedded systems, control theory, and distributed sensor networks. Their evolution has been propelled by both technological advancements and pressing industry needs. The key technological drivers include the widespread adoption of low-cost, high-capability sensors, advances in hardware miniaturization, the availability of powerful low-power computing, and the ubiquity of wireless communication. Simultaneously, critical sectors such as aerospace, factory automation, and healthcare are increasingly demanding a more robust technological foundation to develop large-scale, safety-critical systems correctly and affordably.

These developments have extended CPS applications from micro-level smart devices to macro-level infrastructures spanning medical devices, transportation vehicles and intelligent highways, energy grids, and robotic systems. Since its introduction by [2], the CPS concept has expanded to incorporate self-properties such as autonomy and resilience, enabling systems to adapt to external influences and maintain optimal states without direct human intervention. In this vision, autonomous systems dynamically configure their own operations, optimize performances, and interact seamlessly with both human operators and other machines. Such capabilities are essential for autonomous operation in complex and dynamic environments. In (Figure 1.1), a generic architecture of CPS is illustrated, highlighting the integration of the cyber, communication, and physical domains through sensing, actuation, and feedback mechanisms [3].

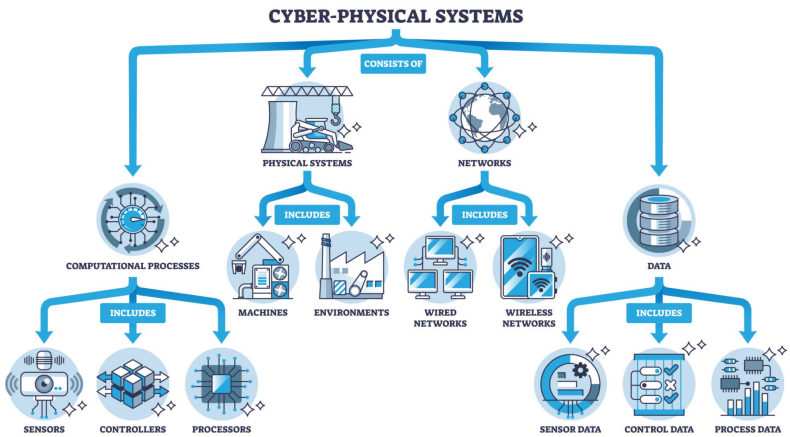


Figure 1.1: Generic architecture illustrating the core components and integration of diverse domains within modern Cyber-Physical Systems (AI generated)

A core functionality of CPS lies in their ability to coordinate spatially distributed

processes, enabling real-time automation and control across critical infrastructures such as water, electricity, and transportation networks [4]. This deep integration of computation and networking into physical processes represents a significant evolution from traditional engineered systems. Ultimately, CPS encompass the incorporation of sensing, computation, and control directly into our physical spaces. The primary goal of this paradigm is to enhance these environments, making them fundamentally safer, more efficient, more responsive, and more reliable through intelligent, automated oversight and action.

1.2 Problem Statement

The integration of computation, communication, and physical processes has positioned CPS as the foundational framework for modern infrastructure. While this convergence improves operational capabilities, it changes the risk profile of autonomous systems by invalidating the isolation assumptions in traditional safety engineering [5]. The difficulty in securing these systems arises from the difference between reliability, which addresses unintentional faults, and security, which addresses intentional attacks [6]. Classical control theory and safety standards rely on the assumptions that system disturbances are stochastic or bounded, allowing robust control techniques, such as H_∞ control or standard MPC, to retain stability against such disturbances[7]. These assumptions do not extend to cybersecurity, where attacks are intentional and strategic rather than random. As a result, adversaries with access to the control loop can exploit system dynamics to manipulate sensors and actuators. Moreover, security solutions for cyber-physical systems often rely on conventional network-based methods, such as Intrusion Detection Systems (IDS). Since these methods focus on packet inspection and protocol analysis, they are insufficient for detecting attacks rooted in system dynamics. By prioritizing data confidentiality and integrity rather than physical stability, these solutions allow adversaries to bypass network-layer protections or execute protocol-compliant attacks that destabilize the physical process. Therefore, considering these, one of the central motivation behind this thesis is the absence of unified design frameworks that model the adversary as an optimizing agent within the control loop rather than as a disturbance to be rejected. To develop a defense against such an optimizing agent, one must first define the upper bound of the system's vulnerability. It is impossible to guarantee safety without first understanding the worst-case impact an intelligent adversary can cause. This necessitates a shift from defensive to offensive modeling, viewing the problem from the attacker's perspective.

To address the vulnerabilities in autonomous systems, Lipp and Boyd proposed the concept of Antagonistic Control [8]. This framework models the worst-case impact of an intelligent aggressor by inverting the objective of the system controller. Instead of minimizing a cost function to maintain stability, the antagonistic formulation seeks an input sequence that maximizes this function to force instability. Although this approach provides a theoretical upper bound on system vulnerability, the current framework presents limitations that prevent its deployment in safety-critical environments. The primary barrier to practical implementation is the absence of application specific formulations capable of bridging the gap between abstract theory and real-world dynamics. The initial work by Lipp and Boyd validated this concept using simplified, academic two-state models. These linear abstractions fail to capture the nonlinear behaviors of modern autonomous agents (e.g., multirotors). Such systems hold complex, coupled dynamics where a malicious input on one chanel can cause cascading effects across others. The vulnerability assessment requires extending this proposal to create model specific formulations that account for physical realities rather than relying on generic, low-order models. Moreover, existing antagonistic frameworks typically neglect system uncertainties, such as external disturbances or model mismatches, by assuming perfect state knowledge. Malicious inputs calculated on nominal models become ineffective if these uncertain factors oppose the attack trajectory and including these variables transforms the standard formulation into a nested optimization problem.

This thesis addresses this complexity by proposing a formulation where the adversary maximizes damage while considering the uncertainties. This approach solves the resulting multi-layered optimization problem to guarantee certain level of damage, regardless of the unfavorable scenarios such as environmental or parametric uncertainties. Finally, the mathematical formulation of the attacker's objective often misaligned with practical goals, such as, standard antagonistic control defines success as maximizing a quadratic cost function, however, an attacker's primary aim is to violate specific physical constraints, such as forcing a drone beyond a recovery angle, rather than maximizing a numerical error value. Formulations focused solely on cost maximization may provide suboptimal solution as they may prioritize high-energy trajectories that do not cause failure while overlooking lower-cost sequences that successfully drive the system into an unsafe region. This focus on mathematical deviation leads to an underestimation of the system's time-to-failure. Further, the computational nature of the optimization problem restricts practical implementation. While standard stabilizing controllers minimize a convex cost function, the antagonistic problem requires maximizing a convex function over a convex set. This results in a non-convex optimization problem that is generally NP-hard. To address this, the proposed framework solves the problem using a vertex enumeration approach, which guarantees the identification of a valid global solution.

These worst-case scenarios provides a baseline for understanding system vulnerability, yet it is required to develop frameworks that increase safety against such intelligent adversaries. While the literature presents comprehensive research on prevention and detection mechanisms from both network and control theory perspectives, the domain of post-attack mitigation requires further study. Significant contributions exist within network security, but control-theoretic approaches for mitigation remain limited. Existing methods include the isolation of compromised components, actuator reallocation, signal correction, or completely switching off the system to avoid major damage [9], [10], [11]. Addressing this gap, this thesis focuses on post-attack solutions to ensure system resilience. This limitation requires a mitigation strategy that operates outside the standard control loop to eliminate the malicious influence. Among the available strategies, this work identifies SoftWare Rejuvenation (SWR) as a reliable recovery mechanism to be analyzed and enhanced for practical feasibility. SWR has emerged as an effective defense technique for mitigating malicious intrusions, enabling control systems to be restored to a trusted initial state [12], [13], [14]. By refreshing the runtime environment, SWR eliminates malicious modifications and potential runtime anomalies. Its feasibility in CPS relies on the physical inertia of the plant. As noted in existing literature, physical inertia guarantees that an adversary cannot destabilize a system instantaneously; it requires a finite amount of time to force the system into failure [15]. This property provides a safe window allowing the process to continue during the brief downtime required for a reset. Due to its low operational overhead and ease of deployment, SWR is suitable for securing existing systems without requiring additional architectural redesign. The state-of-the-art implementation of SWR remains on periodic schedules derived from worst-case reachability analysis to ensure the system resets before an adversary can cause an unsafe state. This preventive method reduces system availability by enforcing blind downtime and interrupting mission progress regardless of the actual threat presence. However, the size of the inertial window dictates the operational feasibility of this approach. In systems with large inertia (slow dynamics), traditional periodic SWR can be implemented easily. Conversely, for systems with low inertia (fast dynamics, e.g., multi-rotors), this safe window is incredibly brief. In these highly dynamic systems, relying on blind, periodic resets becomes highly disruptive and inefficient. These limitations require a transition to a condition-based framework that triggers rejuvenation only when necessary.

The transition to condition-based SWR requires detection capabilities that standard mechanisms cannot provide alone. Although observer-based monitors uses residual signals to identify abrupt faults, they fail to detect attacks where an adversary optimizes inputs to drift the system gradually within residual thresholds. This detection gap necessitates a dual-monitor architecture that complements the reactive observer with a predictive monitor capable of estimating the Time-to-Violation (T^*) of safety constraints. In

this architecture, the antagonistic formulation is repurposed and integrated as the core mechanism of the predictive monitor, allowing the defense to forecast the attacker's moves and trigger recovery before safety violations occur. Further, implementation of this predictive monitor introduces a computational challenge, where estimating T^* requires solving an antagonistic MPC problem to model the worst-case adversary behavior. This formulation involves maximizing a convex function over a convex set, resulting in a non-convex optimization problem. Identifying the global maximum for this scenario is computationally demanding, particularly for systems requiring control updates in milliseconds. Since solving such problems on-line exceeds the capabilities of standard hardware, explicit solutions such as Multi-Parametric Quadratic Programming (mp-QR) are required. This approach reduces the complex optimization problem into a Piecewise Affine (PWA) function, enabling real-time execution on resource-constrained hardware.

1.3 Research Objectives

The primary aim of this research is to develop a model-based design framework to increase the safety and security of autonomous CPS. Unlike traditional IT security, which prioritizes data integrity, this research addresses the specific requirements of control systems where malicious actors aim to manipulate physical processes. To achieve this, the work focuses on four specific objectives ranging from theoretical analysis to simulated validation.

Objective 1: Analysis of CPS Threat Landscapes and Real-World Attack Chains

The first objective is to conduct a analysis of the security challenges unique to CPS. While conventional IT security focuses on data breaches, ransomware, or information theft, this research investigates threats designed to manipulate physical actuators and cause system failures. This involves examining real-world case studies of industrial attacks to understand the attack chain from initial access to the final effect on the operation. This objective includes reviewing the state-of-the-art in networked control security and identifying the limitations of existing defensive methods against sophisticated, model-aware adversaries. The work also includes reviewing literature on CPS attack surfaces such as sensor and actuator level attacks since these sources describe how specific components can be targeted and help in defining the threat model used in later stages of the research. This analysis also establishes the theoretical basis for distinguishing malicious control-theoretic attacks from system faults.

Objective 2: Development of Strategic Antagonistic Control Frameworks

The second objective focuses on modeling the capabilities of an intelligent adversary through the design of an antagonistic MPC framework. This method inverts the standard control algorithm where instead of computing inputs to stabilize the system it uses the predictive feature of MPC to compute optimal control sequence that causes the system failures. This involves a two-stage development process to define the worst-case threat profile:

- **Cost Maximization:** The initial formulation explores how an attacker with perfect system knowledge can destabilize a target, such as a nonlinear quadrotor, by inverting the standard controller's objective from minimization to maximization.
- **Constraint Violation and Robustness:** The research further refines this model to create a strategic attacker that prioritizes the violation of safety constraints over simple cost maximization. This is achieved by reformulating the cost function with slack variables to explicitly prioritize constraint violations. To account for external disturbances, the problem is structured as a robust max-min optimization, ensuring the attack remains effective even in uncertain environments. Due to the non-convex

nature of this formulation, a vertex enumeration approach is used to compute the optimal attack sequence. Moreover, the application of explicit MPC allows these solutions to be computed on run without affecting the performances.

Objective 3: Design of Proactive Defense Mechanisms via Software Rejuvenation

The third objective is to propose a proactive mitigation strategy based on Software Rejuvenation. Conventional SWR approaches rely on periodic schedules that interrupt mission-critical operations. This research focus on to design a condition-based SWR framework that triggers rejuvenation only when necessary. This defense method uses a dual-monitor architecture:

- **Anomaly Detection:** Implementation of a Luenberger-like observer to generate residual signals that identify deviations from nominal system behavior.
- **Predictive Safety Monitoring:** This involves using the antagonistic control logic from the defense perspective which provides a monitoring metric, Time-to-Violation (T^*), namely a prediction of the minimum time required for an attacker to force a constraint violation from the current state. Explicit MPC is employed here to enable real-time computation of safety margins, ensuring that the monitoring tool does not negatively impact system performance.

The integration of these tools allows for a decision-making module that triggers rejuvenation before a critical failure occurs, ensuring high system availability and operational safety.

Objective 4: Validation on Heterogeneous Dynamic Systems

The final objective is to validate the proposed approaches through simulation. To demonstrate applicability, the methods are tested on systems with distinct dynamic characteristics:

- **Highly Dynamic Systems:** A nonlinear model of a quadrotor is used to test the framework's response times and performance under rapid state changes.
- **Slow Dynamic Systems:** A four-tank system model serves as a benchmark for process control applications where dynamics evolve more slowly.

These simulations validate the proposed condition-based SWR and antagonistic detection metrics can increase the safety of autonomous systems against optimized, model-based attacks.

1.4 Thesis Outline

The thesis is organized into six chapters, structure to progress from theoretical foundations and system modeling to the development of adversarial control frameworks and defensive strategies.

Chapter 2: Background and Literature Review This chapter defines the context for CPS security. It presents the distinction between Information Technology (IT) and Operational Technology (OT) security and reviews real world case studies, including Stuxnet and the Ukraine Power Grid incident. The chapter defines the general dynamical model of the system and the adversary model used throughout the research. It concludes by reviewing existing defense mechanisms and introducing the fundamental concepts of SoftWare Rejuvenation (SWR).

Chapter 3: System Modeling and Control Preliminaries This chapter derives the mathematical models and control algorithms used for validation. It presents the kinematics and rigid body dynamics for the Quadrotor Unmanned Aerial Vehicle (UAV) and the mass balance equations for the Four-Tank System. Additionally, it details the formulation of standard MPC and Explicit MPC via Multi-Parametric Quadratic Programming (mp-QP).

Chapter 4: Antagonistic MPC Control This chapter proposes a framework for modeling worst-case attacks using Antagonistic MPC. It begins by formulating an attack policy based on cost maximization and refines this into a constraint-violation model designed to force system failure. The chapter addresses environmental uncertainties by proposing a robust, disturbance-aware antagonistic controller using a max-min optimization approach. Finally, it presents a vertex enumeration method to solve the resulting non-convex optimization problems.

Chapter 5: Condition-Based Software Rejuvenation Framework This chapter proposes a defensive framework for mitigating certain adversarial attacks in CPS. It defines the operational limitations of periodic SWR and introduces a condition-based triggering framework. The architecture uses two monitoring tools: an observer-based residual detector and an antagonistic MPC-based predictor for proactive safety estimation. The chapter validates the integration of these monitors through simulations on both the quadrotor and four-tank systems.

Chapter 6: Conclusion Finally, this chapter summarizes the research findings related the vulnerability of autonomous systems to model-based attacks and the efficacy of the condition-based SWR mitigation method. It also outlines the limitations of the current work and suggests directions for future research.

Part 1: Background and Preliminaries

Chapter 2

Background and Literature Review

2.1 Introduction

This chapter provides the theoretical and historical foundation necessary to understand the security challenges faced by CPS. While Chapter 1 introduced the broad objectives of the thesis, this chapter addresses the specific distinctions that make traditional IT security measures insufficient for CPS. This chapter begins by reviewing the differences between Information Technology (IT) and Operational Technology (OT) security, specifically addressing the conflicting priorities between data confidentiality and operational safety. Following this, the chapter examines real-world cyber-physical attacks to show the evolution of threat vectors from simple disruptions to complex, model-aware control attacks. This analysis provides the context for reviewing the theoretical models of adversarial control and the existing mitigation strategies, identifying the gaps that the proposed model-based design framework aims to address.

2.2 Real-World Cyber-Physical Attacks

To model and mitigate threats against autonomous systems, it is appropriate to first define the nature of the adversary and the environment in which they operate. Attacks on CPS differ from the traditional networks by targeting the intersection of cyber logic and physical dynamics.

2.2.1 Distinction between IT and OT Security

Integrating cyber systems with physical processes requires redesign of the security solutions. Although IT and OT systems share hardware and networking protocols, their operational objectives and security requirements differ. The prioritization of the Confidentiality, Integrity, and Availability (CIA) triad represents the primary distinction between these domains. IT environments prioritize confidentiality to prevent unauthorized data disclosure, regarding integrity and availability as secondary considerations. Whereas, CPS and OT prioritize availability and safety. As noted by [16], failure to ensure availability in systems such as power grids or autonomous vehicles can cause immediate physical damage. Accordingly, the CPS priority order inverts the standard IT hierarchy.

This prioritization shifts the core objective from protecting information to protecting operations [17]. Defensive mechanisms in IT often shut down systems to prevent data exfiltration. In a CPS context, a security mechanism triggering an actuator shutdown may cause more damage than the malware itself. Such operational constraints limit the utility of standard IT security measures, including aggressive encryption or automated account locking.

Threat actors and potential damages also vary between domains [18]. IT attacks are typically profit-driven, executed by criminals seeking financial gain through theft or fraud. Damages in these scenarios remain largely intangible, such as financial loss or reputational harm. While, threats to CPS often derive from ideological or strategic motivations involving nation-states targeting critical infrastructure. These attacks cause physical damage, including equipment destruction, environmental hazards and threats to human safety.

2.2.2 Case Study 1: Stuxnet (2010)

Stuxnet represents the first known cyber weapon designed to cause physical destruction to critical infrastructure. The malware specifically targeted the Natanz uranium enrichment

plant in Iran with objective of disrupting the country's nuclear program. Although the attack targeted a specific facility in Iran, the malware infected over 200,000 computers globally. This spread occurred because Stuxnet employed propagation mechanisms to bridge the air gap isolating the target network. The worm utilized four separate zero-day exploits and spread via multiple vectors, including USB drives and local network shares.

The attack sequence began when operators connected infected USB drives to engineering workstations within the facility. Stuxnet exploited a Windows shortcut vulnerability (CVE-2010-2568) to execute automatically without user interaction. It utilized stolen digital certificates to install kernel-level drivers, allowing it high system privileges and allowing it to evade antivirus detection. The malware then spread laterally across the internal network, specifically scanning for machines running Siemens Step7 software. Once established on the target Programmable Logic Controller (PLC), Stuxnet modified the control logic. These alterations adjusted centrifuge speeds beyond safe limits, causing physical degradation. Stuxnet also maintained a Remote Procedure Call (RPC) backdoor to update itself or re-infect the system if the initial components were removed. The success of this attack relied entirely on the malware's ability to maintain a persistent foothold in the controller's memory to continuously inject false commands [19], [20].

2.2.3 Case Study 2: Ukraine Power Grid (2015/2016)

In December 2015, the Sandworm group executed a coordinated cyberattack against Ukrainian power utilities, disrupting electricity distribution to approximately 230,000 customers. The intrusion began with a spear-phishing campaign targeting employees with emails containing malicious Microsoft office attachments. Enabling macros in these documents triggered the download of the BlackEnergy 3 malware.

This initial foothold allowed attackers to conduct reconnaissance and harvest credentials within the corporate IT network. They escalated privileges to access the SCADA environment, targeting the communication infrastructure between HMIs and field devices such as Remote Terminal Units (RTUs). Using compromised HMI workstations, the attackers issued unauthorized commands to open circuit breakers, physically cutting power to the grid.

To prolong the outage and complicate restoration, the attackers launched a Telephony Denial of Service (TDoS) attack against the customer call centers and deployed KillDisk malware to erase files and corrupt Master Boot Records on workstations. This case demonstrates the ability of attackers to shift from standard IT compromises to direct OT manipulation. This case highlights that a compromised controller state can be exploited to bypass safety protocols typically managed by the system [21], [22].

2.2.4 Case Study 3: Havex (Dragonfly) Campaign

The Havex campaign, attributed to the group known as Dragonfly or Energetic Bear, represents the risks posed by software supply chain compromises in Industrial Control Systems (ICS). Attackers did not target the industrial facilities directly. Instead, they compromised the websites of legitimate ICS software manufacturers. The adversaries trojanized legitimate software installers and update packages available on these vendor sites. Engineers downloading these infected drivers or updates unintentionally executed the Havex Remote Access Trojan. It allowed the attackers to bypass perimeter defenses by riding on the trust established between equipment vendors and system operators. Once inside the network, Havex used a unique scanning module based on Open Platform Communications (OPC) standards. It used this protocol to query the industrial network and map connected devices such as PLC. Unlike Stuxnet, the primary objective of Havex was gathering rather than physical destruction. The attackers collected network topology data and device specifications. This allows adversaries to identify critical nodes to prepare for future sabotage operations. This persistent presence in the trusted software layer allowed attackers to bypass perimeter defenses, maintaining a dormant state within the network while preparing for potential physical disruption [23].

2.2.5 Case Study 4: Colonial Pipeline Ransomware (2021)

The Colonial Pipeline incident of May 2021 underscores the risks associated with the convergence of IT and OT networks. Attackers from the DarkSide group gained initial access via a legacy Virtual Private Network (VPN) account that lacked Multi-Factor Authentication (MFA). Utilizing leaked credentials, they established a foothold in the corporate IT network to deploy ransomware.

Although the malware was contained within the IT environment, the company initiated a precautionary shutdown of the operational pipeline. This decision stemmed from the compromise of billing and metering systems which prevented the accurate tracking of fuel distribution. The shutdown lasted five days and halted the transport of approximately 2.5 million barrels of fuel per day, affecting 45% of the supply to the US East Coast. This incident demonstrates that corruption of the supervisory IT layer can force a physical fail-safe state even if the Programmable Logic Controllers remain untouched [24].

2.2.6 Case Study 5: PIPEDREAM/INCONTROLLER (2022)

The discovery of the PIPEDREAM toolkit made a shift toward modular malware designed to interact natively with ICS equipment. Analysis indicates that attackers gain initial administrative access to engineering workstations by exploiting a vulnerable ASRock motherboard driver (CVE-2020-15368). This technique is known as Bring Your Own Vulnerable Driver (BYOVD). Instead of using expired signatures, attackers utilize a legitimate, digitally signed driver to bypass Windows driver signing policies and load malicious kernel modules.

Once kernel-level execution is established, the toolkit interacts directly with PLC (such as Omron and Schneider Electric) using their native industrial protocols. This capability allows the attacker to scan and manipulate the controller's operational state without exploiting specific software bugs. The toolset essentially turns the controller's intended functionality against itself by sending valid commands to modify safety parameters or logic execution flow. This method renders traditional vulnerability patching ineffective because the malware abuses legitimate operational features rather than code defects [25].

2.2.7 Lessons Learned

These case studies show the specific characteristics distinguishing CPS threats from conventional IT attacks, such as:

- Attackers frequently use trusted channels to bypass conventional network based firewalls. Stuxnet used physical USB drives, Ukraine relied on employee's interaction emails and Havex compromised the trusted vendor software.
- In majority cases, intrusion begins in the enterprise layer. Attackers establish persistence and move laterally to reach the operational technology network to access physical control system.
- The reviewing of these cases show that understanding the physical process is central. Stuxnet manipulated frequency converters to damage centrifuges, while Ukraine attack targeted specific breakers and blocking the safety protocols.
- These threats target the physical limits of the system rather than data confidentiality or financial motives. The impact is measured in physical degradation, service loss or safety violations.

These observations suggest that the ability of malware to propagate laterally shows the vulnerability of flat network architectures. Once the system boundary is compromised, relaying solely on standard network-based solutions is not sufficient. Such tools analyze the packet traffic structure, which may appear legitimate to the network yet still trigger a worst case scenario.

2.3 CPS System Modeling

This section presents the modeling framework for CPS used in this work. It details the system architecture and mathematical dynamics, moving from continuous-time nonlinear models to the discrete-time linear approximations required for the analysis in subsequent chapters.

2.3.1 General Architecture

The general architecture of CPS is designed to integrate physical processes with cyber capabilities, specifically computation and communication, to enable real-time interaction with the physical environment, remote monitoring, and automated decision-making [26]. As illustrated in (Figure 2.1), this framework facilitates the interaction between the physical and digital worlds through four primary domains: the physical plant, sensors and actuators, the communication infrastructure, and the cyber system. This architecture operates through a continuous feedback loop that begins at the environment layer, where sensors measure the state of the physical plant and actuators execute physical actions. This data is transmitted via the communication layer, which ensures the seamless flow of information to the cyber system. Within this computational layer, controllers and decision logic process the incoming data to compute optimized control commands, which are then routed back to the actuators to drive the physical process.

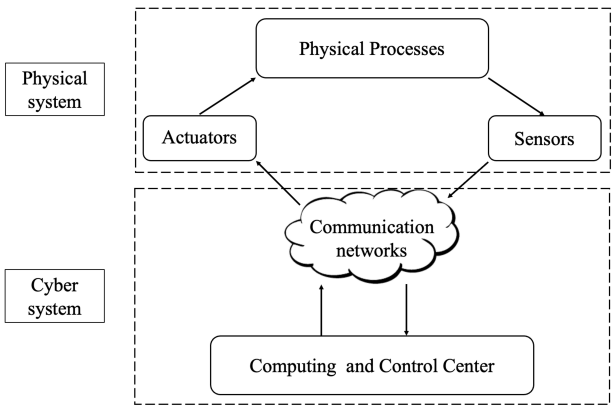


Figure 2.1: A general architecture of a CPS [3]

2.3.2 Mathematical System Dynamics

To formally analyze the impact of attack surfaces on the CPS, we consider a general continuous-time nonlinear plant. Let $x(t) \in \mathbb{R}^{n_x}$ denote the system state vector, $u(t) \in \mathbb{R}^{n_u}$ the control input, and $y(t) \in \mathbb{R}^{n_y}$ the sensor output at time t. The plant dynamics and measurement equations are described as:

$$\dot{x}(t) = f\big(x(t),\, u(t),\, w(t)\big) \tag{2.1a}$$

$$y(t) = h\big(x(t)\big) + v(t) \tag{2.1b}$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$ represents the vector field governing the physical process, and $h : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ is the observation function [27], [28]. The terms $w(t) \in \mathbb{R}^{n_w}$ and $v(t) \in \mathbb{R}^{n_y}$ represent process disturbances (such as modeling uncertainties or external perturbations) and measurement noise, respectively.

Regularity and Constraints

To guarantee the existence and uniqueness of solutions, we assume that the functions $f(\cdot)$ and $h(\cdot)$ are locally Lipschitz continuous with respect to their arguments and piecewise

continuous in time. Furthermore, operational safety constraints restrict the states and inputs to closed admissible sets. System trajectories must satisfy $x(t) \in \mathcal{X} \subset \mathbb{R}^{n_x}$ and control inputs must remain within $u(t) \in \mathcal{U} \subset \mathbb{R}^{n_u}$ for all $t \geq 0$. Depending on the specific analysis required, the disturbance and noise terms are modeled as bounded signals, such that $\|w(t)\| \leq w_{max}$ and $\|v(t)\| \leq v_{max}$ for all t [29].

Closed-Loop and Error Dynamics

The system operates under a feedback control law $\kappa(\cdot)$, which computes the control input based on available system information. Depending on the architecture, this corresponds to either full-state feedback $u(t) = \kappa(x(t))$ or output feedback $u(t) = \kappa(y(t))$ [30]. Substituting this into the system equation yields the nominal closed-loop dynamics:

$$\dot{x}(t) = f(x(t), \kappa(h(x(t)) + v(t)), w(t)) \triangleq f_{cl}(x(t), w(t), v(t)) \tag{2.2}$$

In scenarios focusing on trajectory tracking, the objective is to follow a smooth, time-varying reference trajectory $x_{ref}(t) \in \mathcal{X}$ satisfying the nominal dynamics $\dot{x}_{ref}(t) = f(x_{ref}(t), u_{ref}(t))$. Defining the tracking error as $e(t) = x(t) - x_{ref}(t)$, the error dynamics describe the deviation of the actual system from the intended behavior:

$$\dot{e}(t) = f_{cl}(x(t), u(t), w(t)) - f(x_{ref}(t), u_{ref}(t)) \tag{2.3}$$

The controller is designed such that, in the absence of adversarial interference or excessive noise, the error dynamics are asymptotically stable, ensuring $\lim_{t\to\infty} \|e(t)\| = 0$ [27], [31].

Digital Implementation

While the physical plant evolves in continuous time, the cyber components are implemented digitally. The cyber system, comprising controllers, monitors, and decision logic operates in discrete time steps $k \in \mathbb{N}$ with a sampling period T_s. The sensor measurements are sampled at discrete instances t_k, resulting $y_k = y(t_k)$. Therefore, the discrete control command $u_k = \kappa(y_k)$ is computed and applied to the continuous plant via a Zero-Order Hold (ZOH) mechanism [32]. This holds the input constant between updates:

$$u(t) = u_k \quad \forall t \in [t_k, t_{k+1}) \tag{2.4}$$

This hybrid formulation captures the realistic sampled-data nature of the CPS, where the mismatch between continuous physical evolution and discrete control updates can create vulnerabilities exploited by timing-based attacks.

Linear Approximation and Discrete-Time Models

In addition to the nonlinear formulation, this research also uses a linear representation of the system dynamics. The linearization process, detailed below, provides the foundational general model required for both the adversarial strategies and the proposed defensive mechanisms. Specifically, the predictive algorithms supporting the antagonistic attack generation and part of the safety monitors rely on this linear structure to maintain the computational tractability necessary for online optimization. This approach aligns with standard control practices, where linear models enable the use of fast, reliable solvers while maintaining performance near the operating point [33].

We define a steady-state equilibrium point (x_{eq}, u_{eq}) that satisfies the condition $f(x_{eq}, u_{eq}, 0) = 0$. The system dynamics are expressed in terms of deviation variables, defined as $\delta x(t) = x(t) - x_{eq}$ and $\delta u(t) = u(t) - u_{eq}$. By applying a first-order Taylor series expansion around this equilibrium, we obtain the Linear Time-Invariant (LTI) approximation:

$$\dot{\delta x}(t) = A_c \delta x(t) + B_c \delta u(t) \tag{2.5}$$

where the continuous system matrices $A_c \in \mathbb{R}^{n_x \times n_x}$ and $B_c \in \mathbb{R}^{n_x \times n_u}$ are the Jacobian matrices defined as:

$$A_c = \left.\frac{\partial f}{\partial x}\right|_{(x_{eq},u_{eq})}, \quad B_c = \left.\frac{\partial f}{\partial u}\right|_{(x_{eq},u_{eq})} \tag{2.6}$$

This approximation assumes that the remainder terms from the Taylor expansion are negligible, a condition that holds valid provided the deviations $\|\delta x\|$ and $\|\delta u\|$ remain sufficiently small [27]. While this local approximation is standard practice, assuming operation near a equilibrium point does present practical limitations. In highly dynamic applications, large state excursions can cause the neglected nonlinear dynamics to dominate, which may cause inaccuracies in linear state observers. However, within the context of this thesis, this assumption is practically justified because the linear model is used specifically for safety monitoring rather than nominal plant control. From a defensive perspective, the monitor is designed to safeguard the bounded, safe operational envelope around the equilibrium. If an adversarial attack forces the system far from this equilibrium, the resulting dynamic mismatch and state deviation are exactly what the monitor detects to trigger the condition-based rejuvenation framework.

To align with the sampled-data implementation, the continuous LTI model is discretized assuming a ZOH on the input over each sampling interval $[kT_s, (k+1)T_s)$. The resulting discrete-time prediction model describes the evolution of the deviation states at discrete steps k:

$$x(k+1) = Ax(k) + Bu(k) \tag{2.7}$$

In this formulation, $x(k)$ and $u(k)$ represent the discrete deviation vectors δx_k and δu_k. The discrete system matrices A and B are computed using the matrix exponential to capture the exact evolution of the linear dynamics over the sampling period T_s:

$$A = e^{A_c T_s}, \quad B = \left(\int_0^{T_s} e^{A_c \tau} d\tau \right) B_c \tag{2.8}$$

These matrices are typically computed numerically via augmented matrix exponential routines to avoid explicit numerical integration[32]. This discrete linear formulation serves as the baseline internal model for both the attack generation and some of the detection algorithms presented in subsequent chapters.

2.4 Threat Landscape and Adversary Model

In this framework, security is analyzed through an adversary model. This model defines the resources available to the attacker and categorizes the specific types of attacks they can execute.

2.4.1 Adversary Capabilities

To specify a threat model consistent with control-theoretic security analysis, we define the adversary's resources along three orthogonal dimensions: disclosure resources, disruption resources, and model knowledge [34]. This framework separates what the adversary can observe from what they can manipulate, defining the conditions required to design advanced, stealthy attacks.

- **Disclosure Resources** quantify the adversary's ability to passively obtain information about system operation without directly altering the physical evolution. This corresponds to a violation of confidentiality, where the adversary eavesdrops on sensor-to-controller or controller-to-actuator channels. Formally, we define the disclosure set $\mathcal{D}_y \subseteq \{1, \ldots, n_y\}$ and $\mathcal{D}_u \subseteq \{1, \ldots, n_u\}$ as the subsets of sensor and actuator channels, respectively, that the adversary can monitor. Access to these resources allows the attacker to collect sequences of measurements $y(t)$ and control inputs $u(t)$, enabling system identification tasks such as inferring parameters,

learning normal operating behaviors, or recording valid data sequences for future replay attacks [35].

- **Disruption Resources** define the adversary's ability to actively affect system operation, related to violations of integrity and availability. These resources are categorized based on the specific channels the adversary can compromise. Integrity or deception attacks, involve the ability to inject false data into the feedback loop; let $\mathcal{A}_y \subseteq \{1, \ldots, n_y\}$ and $\mathcal{A}_u \subseteq \{1, \ldots, n_u\}$ denote the subsets of sensor and actuator channels where the adversary can alter the transmitted signals. Alternatively, availability attacks (Denial-of-Service) involve blocking or delaying data transmission to prevent correct data from reaching its destination within the sampling period T_s. The magnitude of disruption is often constrained by a resource limit, such as $|\mathcal{A}_y| < n_y$, as the performance of resilience method often depends on the number of reliable channels available [5].
- **Model Knowledge** represents the adversary's understanding of the system dynamics. This ability dictates the complexity of the attack, ranging from limited black box knowledge to full understanding. An adversary with full knowledge has exact details of the vector fields $f(\cdot)$ and $h(\cdot)$ (or matrices A, B, C), the control law $\kappa(\cdot)$, and parameters like the sampling rate. This high level of model knowledge is the enabler for stealthy attacks, such as, by exploiting the exact mathematical model of the plant, a fully aware adversary can formulate advanced attacks, such as zero-dynamics attacks that decouple the physical impact from cyber detection, driving the state $x(t)$ to unsafe regions while maintaining sensor outputs $y(t)$ that appear nominal to the controller [36].

2.4.2 Taxonomy of Attacks

We classify cyber attacks on the CPS based on the specific component targeted within the architecture and the security property violated, namely CIA. This taxonomy integrates the resource framework discussed in the previous section with the control theoretic dynamics of the system. In this view, disclosure resources are the primary enabler for confidentiality violations, whereas disruption resources are used to violate integrity and availability. Furthermore, the level of model knowledge possessed by the adversary determines their ability to engineer advanced attacks that exploit the specific vector fields $f(\cdot)$ and $h(\cdot)$ governing the plant [37], [38], [39]. In the broader, attacks are usually categorized as follows:

Eavesdropping Attack

Eavesdropping is a passive confidentiality violation where the adversary intercepts sensor measurements or control commands without altering the physical plant dynamics. This strategy uses disclosure resources to gather operational data from the communication layer [37], [38]. The primary threat arises from the accumulation of information which facilitates system identification, parameter inference, or the recording of valid trajectories required to execute later stage deception attacks [37].

Denial of Service (DoS) Attack

A major threat to system availability arises when Denial-of-Service attacks use disruption resources to block data transmission between the plant and the cyber system [37], [38]. In the context of the sampled-data implementation, this prevents the update of control signals or measurement vectors within the required time steps. The immediate consequence is the forced dependence on stale data or return to open-loop dynamics, which can lead to instability depending on the open-loop poles of the physical process [40].

Replay Attack

The replay attack is a combined strategy involving two phases that use both disclosure and disruption resources. Initially, the adversary passively records a sequence of sensor data from a nominal operational sequence. Later, the real-time sensor feed is disconnected and replaced with this historical sequence [38], [41]. This manipulation feeds the controller with plausible measurement data while masking the actual state evolution, bypassing anomaly detectors that rely on statistical deviations [42].

Bias Injection Attack

Bias injection targets data integrity by introducing an additive offset to the sensor or actuator channels. This corruption changes the values processed by the feedback control law or the state estimator [38], [39]. Even constant or slowly varying biases can cause steady-state tracking errors and shift the system operating point close to safety boundaries, while the resulting deviations mimic natural measurement noise or process disturbances, making them invisible to a monitor unaware of the attack [38], [43].

Zero Dynamics Attack

The zero dynamic attacks require precise model knowledge. The adversary generates an attack signal matched to the system's zero dynamics so that the effects of the intrusion are (ideally) unobservable in the output channel [38], [44]. As a result, the sensor measurements indicate nominal behavior while the internal state variables diverge from the reference trajectory, allowing the attack to cause physical impact while remaining undetected [39], [44]

The specific mathematical formulation of these attacks, including the modeling of additive signals and channel selection, is detailed in the following section.

2.5 Mathematical Formulation of Attacks

To model specific attacks on subsets of components within the CPS architecture, we introduce diagonal selection matrices that map the adversary's disruption resources to specific channels [45]. Let Γ_u be a $n_u \times n_u$ diagonal matrix where its i-th diagonal element is 1 if the i-th actuator is compromised, and 0 otherwise. Similarly, let Γ_y be a $n_y \times n_y$ matrix which indicate the subset of compromised sensors. These matrices formally define the attack surface available to the adversary.

2.5.1 Actuator Side Attack Model

An actuator attack targets the system's input channel by injecting an adversarial signal $a_u(t) \in \mathbb{R}^{n_u}$. This modifies the control input actually applied to the physical plant. While the controller computes the intended command $u(t) = \kappa(y(t))$, the corrupted input becomes:

$$\tilde{u}(t) = \kappa(y(t) + \Gamma_u a_u(t)) \tag{2.9}$$

As illustrated in Figure 2.2, substituting this corrupted input into the system dynamics (Equation 2.1), the evolution of the compromised state $x(t)$ is governed by:

$$\dot{x}(t) = f(x(t), \kappa(y(t)) + \Gamma_u a_u(t), w(t)) \tag{2.10}$$

For systems that are affine in control, this adversarial input acts as a linear additive term. However, in general nonlinear systems, the attack signal alters the system's evolution through the nonlinear vector field $f(\cdot)$. This formulation captures various attack strategies; for instance, a DoS can be modeled by blocking the channel (setting $\Gamma_u a_u(t) = -u(t)$), while bias injection involves adding a constant or time-varying offset [46]. The successful execution of this intrusion depends strictly on the disruption resources targeting the channels mapped by Γ_u.

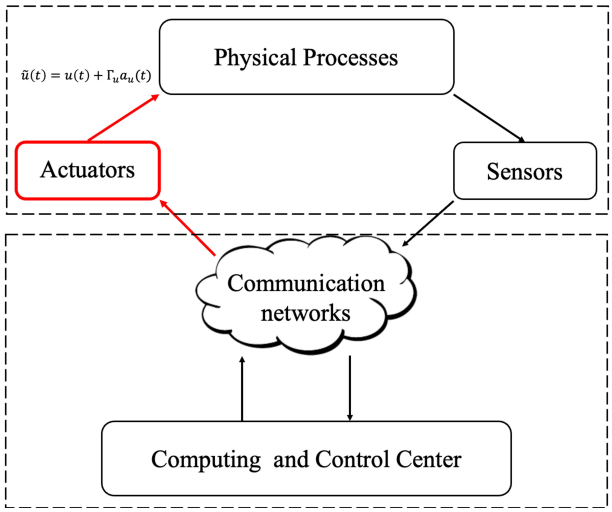


Figure 2.2: Schematic of an actuator integrity attack, modified version from [3]

2.5.2 Sensor Side Attack Model

A sensor attack targets the system's output channel by injecting an adversarial signal $a_y(t) \in \mathbb{R}^{n_y}$. This injection modifies the measurement vector provided to the cyber system. As depicted in Figure 2.3, the corrupted output $\tilde{y}(t)$ received by the controller is given by:

$$\tilde{y}(t) = y(t) + \Gamma_y a_y(t) = h(x(t)) + v(t) + \Gamma_y a_y(t) \tag{2.11}$$

The controller subsequently computes the control input based on this tampered data, such that $u(t) = \kappa(\tilde{y}(t))$. Substituting this compromised feedback law into the state equation gives the attacked closed-loop dynamics:

$$\dot{x}(t) = f(x(t), \kappa(h(x(t)) + v(t) + \Gamma_y a_y(t)), w(t)) \tag{2.12}$$

This additive signal $a_y(t)$ models deception attacks, such as bias injection, or replay attacks [47]. In a replay scenario, adversary uses disclosure resources to define an $a_y(t)$ that replaces the current real-time state measurements with valid historical data sequences recorded from previous operations.

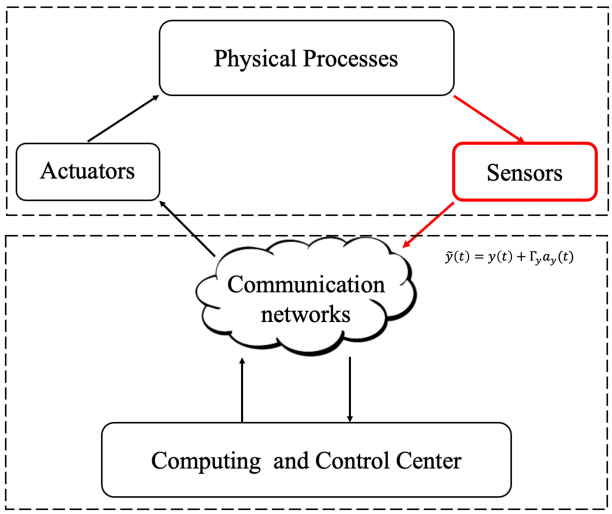


Figure 2.3: Schematic of a sensor integrity attack where an adversarial signal $a_y(t)$ corrupts the measurement vector, modified version from [3]

2.5.3 Combined and Stealthy Formulation

In advanced scenarios, the adversary may compromise both sensors and actuators ($\Gamma_u \neq 0, \Gamma_y \neq 0$), resulting in stealthy operations [6]. The adversary inputs $a_u(t)$ to drive the physical state $x(t)$ away from the nominal trajectory. To hide this deviation from the controller, the sensor injection $a_y(t)$ is designed to cancel the output error, satisfying the condition:

$$h(x(t)) + \Gamma_y a_y(t) \approx h(x(t)) \tag{2.13}$$

This offset decouples the physical system dynamics from the cyber system observations. By ensuring the received measurements mimic the nominal behavior, the adversary makes the manipulation invisible to anomaly detectors. However, the complexity of executing such an attack is significant and often serves as a practical barrier to implementation. It demands strict requirements on the adversary, necessitating precise model knowledge of the system parameters, the nonlinear functions $f(\cdot)$ and $h(\cdot)$, and the specific network configurations. Furthermore, it requires simultaneous access to specific sensor and actuator channels to perfectly cancel the dynamics, a condition that may not always be feasible in real-world environments with segmented network architectures or limited adversarial reach [5], [48].

2.6 Defense Mechanisms in Model-Based Control Systems

The literature proposes several defense mechanisms to address these cyberattacks, which can be broadly classified into:

- **Prevention mechanisms** These strategies aim to secure communication channels and computational resources to prevent the adversary from injecting attack signals $a_u(t)$ or $a_y(t)$ into the system. Fundamental solutions include the implementation of secure communication protocols that use end-to-end encryption and authentication methods, such as TLS or IEC 62443 standards, to ensure data confidentiality and integrity [49], [50]. By implementing these cryptographic measures, the system increases the computational cost for an adversary attempting to inject valid data packets. Furthermore, prevention involves minimizing the attack surface through zero-trust policies and strict access control, which restrict unauthorized communication with the controller and actuators [51]. This limits the adversary's ability to manipulate the specific channels defined by the selection matrices Γ_u and Γ_y. However, these IT-centric methods are often insufficient against adversaries who have bypassed these initial defenses or possess insider credentials.
- **Detection mechanisms** These methods identify anomalies in real-time by verifying that the actual system behavior remains consistent with the predicted behavior derived from the mathematical model. The foundation of physics-based detection is the state estimator (e.g., Kalman Filter or Luenberger observer), which computes an estimate $\hat{x}(t)$ based on the inputs $u(t)$ and measurements $y(t)$ [52]. The system generates a residual vector $r(t) = y(t) - h(\hat{x}(t))$, which triggers an alarm if its magnitude exceeds a threshold τ_{th} [45]. Specific implementations include stateless tests ($||r(t)|| > \tau_{th}$) and stateful tests like the Cumulative Sum (CUSUM) or χ^2 detectors, which aggregate residual history to identify persistent, stealthy deviations [53]. Advanced data-driven approaches uses machine learning, such as Support Vector Machines (SVM) or Recurrent Neural Networks (RNN), to learn complex normal operating patterns and detect nonlinear anomalies [52]. To address replay attacks, active detection methods such as physical watermarking are used. This involves superimposing a known random authentication signal $\Delta u(t)$ onto the control input to verify the freshness of the sensor data through the physical feedback loop [54].
- **Mitigation mechanisms** These strategies aim to maintain operational stability or ensure the system reaches a safe state following a confirmed attack. Passive

mitigation relies on robust control and fault-tolerant methods [55]. If the attack signal $a_u(t)$ is bounded, a robust controller may maintain stability without requiring dynamic reconfiguration. Active mitigation involves the dynamic reconfiguration of the control architecture or isolating the compromised components [46]. Further, advanced architectures often use Secure State Estimation to handle sensor attacks [46], [56]. Instead of discarding sensor data entirely, this method focuses on reconstructing the true state $x(t)$ despite corrupted measurements in $y(t)$. By using the physical redundancy of the sensors, these algorithms solve an optimization problem to identify the sparse attack vector that explains the deviation in the observation function $h(x)$. This provides a cleaned state estimate to the feedback law $\kappa(\cdot)$, filtering the attack from the controller. For scenarios where the primary controller cannot be trusted, switching control (such as Simplex) provide a structural defense [57], [58]. This strategy use a decision logic that monitors the plant state $x(t)$ relative to the admissible set $\mathcal{X}$. If the trajectory approaches the boundary of the safety envelope, control authority is revoked from the complex, high-performance controller and switched to a verified, simple backup controller. This guarantees that the plant state is forced back into a safe region regardless of the attack employed on the primary channel. However, limitation of these filtering and switching approaches is that they often manage the symptoms of an intrusion rather than eliminating the root compromise. Whereas, Software Rejuvenation has emerged as a post-attack solution to eliminate malicious modifications and access [15], [59], [60]. Unlike complex architectural reconfigurations, this method focuses on resetting a compromised component such as the controller, or potentially sensors and actuators to a verified safe state. In the literature, rejuvenation is increasingly viewed as a practical solution as it does not require the complete redesign of developed systems and incurs lower operational overhead compared to permanent hardware redundancy. While currently proposed primarily for rejuvenating compromised controllers, this concept can be extended to peripheral components to eliminate persistent threats and restore the integrity of the cyber-physical interface. Therefore, the remainder of this work is dedicated to modeling and analyzing software rejuvenation as a primary defense mechanism for restoring system integrity.

2.7 Fundamental of Software Rejuvenation

The mitigation methods are necessary for ensuring system safety when prevention methods fail against initial access vectors such as phishing emails, compromised USB drives, social engineering, or credentials acquired from the dark web, as detailed in the case studies section 2.2. In addition to recovery solutions, mitigation methods must also ensure the elimination of unauthorized access, including the removal of adversary footholds, revoked credentials, and disabled compromised interfaces. Among the available mitigation solution, software rejuvenation has been introduced as a robust method for counteracting software aging and cyber attacks. By refreshing the runtime system using a secure and trusted copy of the control software, a process also known as reset and restart, or rejuvenation, eliminates controller corruption, adversarial code injection, and unauthorized modifications of internal states, thereby restoring the system to a known safe configuration and preventing continued access persistence. This section details the theoretical foundations and its specific implementation within dynamical systems

2.7.1 Background and Concepts

SWR was originally introduced in 1995 by [61] to address the problem of software aging. This concept refers to failures that occur when a running program experience an unanticipated system state or suffers from performance degradation due to resource exhaustion, such as memory leaks, data corruption, and unreleased file locks. The fundamental concept of SWR involves restarting the software at a clean state, achieved

either through a complete system reboot or by restoring a verified checkpoint of the uncorrupted software image. The practical relevance of this approach has been validated by adoption in telecommunications and aviation, for instance, the Boeing 787 utilizes sequential resets to mitigate software errors that could otherwise compromise flight control systems [13], [60].

In the domain of CPS security, SWR has been adapted as a prevention and mitigation method against unmodeled or undetectable runtime attacks which can change the controller's code, data or control flow, potentially manipulating physical inputs. SWR is particularly proposed for such scenarios as it completely removes the corrupted software or data and resetting the adversary's progress. To ensure the security of the rejuvenation process itself, the architecture relies on specific hardware and software constructs proposed by [15], such as, a Hardware Root of Trust (RoT) and a Secure Execution Interval (SEI). The RoT is an isolated hardware module, such as a secure module, that is responsible for triggering the reset, ensuring that a compromised main controller cannot programmatically prevent the rejuvenation sequence. The SEI defines a specific timeframe during which all external communication interfaces are disabled, isolating the system to prevent adversaries from interfering with the restoration process. Additionally, a secondary control logic that executes immediately following a refresh and during the SEI, where its function is to drive the CPS to a known safe state before restoring communication and returning the system back to nominal operational mode.

The implementation of SWR within a feedback control loop requires a formal definition of operational states to manage the transition between performance oriented tracking and safety critical recovery. As described in the foundational literature, the operation timeline is segmented into distinct modes active during specific intervals to ensure that the uncertain control period T_{UC}, the interval where the standard controller is unavailable or untrusted, does not lead to a violation of system safety constraints. The proposed framework organizes the rejuvenation strategy into three sequential phases.

2.7.2 Modes of Operation and Rejuvenation Sequence

The implementation of SWR within a feedback control loop requires a formal definition of operational states to manage the transition between tracking and recovery. As described in the foundational literature, the system timeline is segmented into distinct modes active during specific intervals [12]. These intervals include the nominal operation duration T_{MC}, the rejuvenation interval T_{SWR} also referred to as T_{down} where the system operates in open-loop, and the recovery duration T_{SC} required to restore the system state. This segmentation ensures that the total uncertain control period T_{UC} (defined as the interval where the standard controller is unavailable or untrusted) does not lead to a violation of system safety constraints. The proposed framework organizes the rejuvenation strategy into three sequential phases based on these timing definitions.

The first phase is the nominal operation mode, often referred to as Mission Control (MC), which represents the default state where the nominal controller operates under standard network conditions. In this mode, the controller is performance oriented and designed to track a time varying reference trajectory $x_{ref}(t)$ by calculating control inputs that minimize a performance-based cost function regarding tracking error. While in nominal operation, the CPS remains connected to external networks for telemetry and command updates which exposes the system to run time cyber attacks. Operations continue in this mode for the duration T_{MC} until a specific trigger event occurs. Upon the activation of this trigger, which may be established through either a fixed time based schedule or a condition based anomaly detection mechanism, the system transitions to the SWR mode.

Following the SWR, the system enters the recovery control mode, referred to as safety controller in the literature, where a trusted, on board safety controller denoted as $\kappa_{sc}(x)$ is activated. The SC has a distinct objective compared to the nominal controller as it functions as a dedicated state regulator rather than a tracker. Its sole objective is to stabilize the plant by steering the system state $x(t)$ from its current position after the

rejuvenation towards a safe equilibrium. Unlike the fixed duration SWR mode, this phase remains active for a variable time T_{rec} and continues until the system state returns to a subset of admissible states known as the recoverable set $\mathcal{R}$. This ensures that the system is stable enough to be handed back to the high performance nominal controller. The system transitions out of this mode only when the state enters a handover region $\mathcal{R}_{ho}$, defined by a convergence threshold ϵ_{rec} such that

$$\mathcal{R}_{ho} = \{x \mid \|x - x_{ref,paused}\| \leq \epsilon_{rec}\}$$

The threshold ϵ_{rec} must be chosen such that $\mathcal{R}_{ho}$ lies within the Region of Attraction (ROA) of the nominal controller to prevent instability upon resumption. Once the condition $x(t) \in \mathcal{R}_{ho}$ is met, the system transitions back to nominal operation mode where the nominal controller $\kappa_{nom}(x)$ is reengaged, the reference trajectory resumes, and external communications are enabled.

2.7.3 Related Work and Limitations of SWR

The focus of later research moved toward applying control theory to design and enhance this time-triggered framework. [62] established a methodology using Lyapunov functions and invariant sets to mathematically define safe operating regions and derive timing bounds for the SWR schedule. In subsequent work, they provided the formal proofs for both Safety (avoiding failure) and Liveness (guaranteeing mission progress) for SWR protected systems performing trajectory tracking [63]. [64] further refined the approach using Linear Programming (LP) and polytopic sets to achieve a more computationally efficient and less conservative design of time-triggered schedules.

To improve performance, researchers have explored replacing simple feedback controllers with advanced strategies [65], [66] introduced tree-based Model Predictive Control (MPC) that proactively plans for both the upcoming reset and the possibility of an attack. More recently, [67] proposed a hybrid approach integrating Deep Reinforcement Learning (RL) to train an agent that learns a more aggressive and safe setpoint policy, enabling faster mission completion.

The robustness of the framework was further extended to address real-world complexities by providing sufficient conditions for safety under state estimation errors and disturbances and developing a secure recovery algorithm to handle persistent attacks and environmental constraints. Additionally, secure boot was integrated to defend against rootkits, with a formal schedulability analysis ensuring feasibility in real-time systems [68], [69], [70].

At the architectural level, the efficiency of SWR has been improved through micro-rejuvenation, [71] proposed a coordinated strategy for component-based systems, using a fault tree to calculate optimal time offsets for each component's schedule. In parallel, [12], [14] developed a run-time system for complex controllers like the PX4 autopilot, introducing a micro-reboot scheme that restores only the critical application process instead of the full OS. The application of these principles to more complex architectures was explored by [72] in the context of decentralized systems, presenting a procedure for agents to determine when they must communicate to ensure global safety, while using SWR for protection during these vulnerable periods.

Despite these advancements, the state-of-the-art in software rejuvenation for CPS security remains reliant on a proactive, time-triggered paradigm. This approach inherently suffers from an always-on cost model, where performance degrading resets are enforced based on a worst-case assumption about an ever present attacker. This leads to reductions in system availability, which can be as low as 64.3% for systems and significantly delayed mission progress [15]. For instance, experiments have shown that the time to complete a mission can increase from 86.63 s to 484.25 s when rejuvenation is enabled and in worst-case attack scenarios, the system may make no progress at all [14], [15]. This framework introduces a fundamental trade-off between system security and mission progress, such as research in cloud computing by [73] demonstrated that the optimal time-based schedule

for rejuvenation (to maximize availability) is in direct conflict with the optimal schedule for Moving Target Defense (to maximize security), concluding that to boost one attribute, we should compromise the other.

This shared limitation across the literature highlights a critical research gap. The assumption that run-time attacks are completely undetectable, which necessitates the proactive timer, may be overly conservative. This assumption often arises from the limitation of conventional IT solutions but fails to recognize that even stealthy attacks influence the system's physical state. As this state is governed by predictable physical laws and inertia, it presents model-based detection surface lacking in conventional approaches. This motivates our work to develop a new method for software rejuvenation that is condition-based rejuvenation which react only when there is notable evidence of a threat. By developing anomaly detection mechanisms based on physical dynamics of the system, we aim to provide security guarantees without the continuous and degrading performance penalty of the time-triggered model.

Chapter 3

System Modeling and Control Preliminaries

3.1 Introduction

The previous chapter reviewed the theoretical nature of cyber-physical attacks and the limitations of conventional IT security in operational technology environments. Defending against control-theoretic attacks requires an understanding of the underlying physical processes. Therefore, this chapter presents the mathematical preliminaries and simulation environments necessary to validate the proposed methods. The discussion focuses on the derivation of system dynamics and control algorithms used to simulate the cyber-physical environment. These mathematical models are central to the research methodology, serving as both the testbed for analyzing system degradation under adversarial manipulation and as the reference models required by the detection and recovery algorithms to forecast future system behavior.

To demonstrate the broad applicability of the proposed methods, two distinct physical models will be considered for validation. The first is a Quadrotor UAV, selected to represent systems with fast, unstable, and under-actuated dynamics. The second is a four tank System, included to represent the slow, non-linear, and coupled dynamics typical of industrial process control. By simulating these, the feasibility of the safety framework is evaluated across different operational timescales and stability margins.

The organization of this chapter is structured to provide a comprehensive modeling and control framework. Section 3.2 begins by discussing the quadrotor system, defining the kinematic and dynamic non-linear equations and deriving the linearized state-space model. Following this, Section 3.3 introduces the four-tank system and describes the differential equations related to fluid level dynamics, establishing a non-linear model for industrial process simulation. Section 3.4 then presents the fundamentals of Model Predictive Control (MPC), detailing the standard optimization formulation, cost functions, and constraints that form the basis of nominal control logic. Finally, Section 3.6 concludes the chapter with an overview of Explicit MPC and Multi-Parametric Programming, discussing their implementation methods and feasibility for real-time control.

3.2 The Quadrotor System

The quadrotor is an underactuated mechanical system equipped with four motors mounted at the ends of a rigid frame. The vehicle motion is produced by varying the rotational speeds of the individual rotors, which changes the total thrust and the body moments generated about the roll, pitch, and yaw axes. By regulating the attitude, the thrust vector is reoriented, enabling translational motion in three dimensional space while maintaining altitude and stability. In this chapter, the quadrotor is modeled as a rigid body characterized by its mass and inertia, subject to external forces and moments generated by the rotor thrusts and reaction torques. Several modeling approaches exist in the literature, including Euler Lagrange formulations [74]. Here, the Newton Euler approach is used to derive the rigid body equations of motion because it provides a direct and structured representation of translational and rotational dynamics in terms of applied forces and moments. This structure also facilitates the inclusion of actuator and aerodynamic effects through explicit force and moment models when required by the simulation and control design. The remainder of this section introduces the kinematic description, derives the rigid body dynamics, and then presents the simplified equations of motion used for simulation.

3.2.1 Kinematics

To describe the motion of the system, two distinct reference frames are used, the Earth frame $\{R\}(O, x, y, z)$, which is assumed to be inertial and the body-fixed frame $\{R_B\}(O_B, x_B, y_B, z_B)$, where O_B corresponds to the center of mass of the quadrotor. The configuration of the quadrotor is described by the linear position vector $\xi = [x, y, z]^T$ and the angular orientation vector $\eta = [\phi, \theta, \psi]^T$ representing the Euler angles (roll, pitch, and yaw). These angles are defined such that the vehicle operates in a normal flight mode, where the roll and pitch angles are bounded ($-\pi/2 < \phi < \pi/2$, $-\pi/2 < \theta < \pi/2$) to avoid singularities (gimbal lock).

The transformation of vectors from the body frame to the inertial frame is governed by the rotation matrix R, derived via the standard $Z - Y - X$ rotation sequence:

$$R = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix} \tag{3.1}$$

where, $S_{(\cdot)}$ and $C_{(\cdot)}$ denote $\sin(\cdot)$ and $\cos(\cdot)$, respectively.

The linear velocity vector in the inertial frame, v, is related to the body-frame velocity vector, v_B, via the rotation matrix R:

$$v = Rv_B \tag{3.2}$$

The angular velocity vector, $\omega = [p, q, r]^T$, comprises the instantaneous angular velocities about the body axes x_B, y_B, and z_B [74]. It is important to note that ω is distinct from the time derivative of the Euler angles, $\dot{\eta}$. However, the two are related via the transformation matrix W:

$$\omega = W\dot{\eta} \implies \dot{\eta} = W^{-1}\omega \tag{3.3}$$

The transformation matrix W is defined as:

$$W = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix} \tag{3.4}$$

3.2.2 Rigid Body Dynamics

The quadrotor is modeled as a rigid body with six degrees of freedom, subject to external forces and torques. Let m denote the total mass of the vehicle and $I \in \mathbb{R}^{3\times3}$ denote the inertia tensor. The motion of the system is described by the generalized velocity vector in the body frame, $\nu = [u, v, w, p, q, r]^T$, which contains the linear and angular velocity components.

The dynamic equations of motion are derived using the Newton-Euler formalism and can be expressed in a compact vectorial form as:

$$M\dot{\nu} + C(\nu)\nu = \tau \tag{3.5}$$

where $M \in \mathbb{R}^{6\times6}$ is the system inertia matrix. Assuming the origin of the body frame coincides with the center of mass, M is block-diagonal:

$$M = \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I \end{bmatrix} \tag{3.6}$$

Given the geometric symmetry of the quadrotor, the body axes are aligned with the principal axes of inertia. Consequently, the inertia tensor I is diagonal:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.7}$$

The matrix $C(\nu) \in \mathbb{R}^{6\times6}$ represents the Coriolis and centripetal terms, which account for the gyroscopic effects inherent to the rotating reference frame. Finally, the vector $\tau \in \mathbb{R}^6$ represents the generalized forces and moments acting on the airframe:

$$\tau = \begin{bmatrix} f_b \\ \tau_b \end{bmatrix} \tag{3.8}$$

where f_b and τ_b denote the total force and torque vectors expressed in the body frame.

3.2.3 Forces and Moments

The dominant external forces acting on the quadrotor are gravity and aerodynamic thrust. The gravitational force F_g acts on the center of mass along the negative z-axis of the inertial frame. When expressed in the body frame, this force is given by:

$$F_g^B = R^T \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{3.9}$$

Actuation is provided by four fixed-pitch propellers. The motor configuration follows a standard cross pattern, where motors $\mathcal{M}_1$ and $\mathcal{M}_3$ rotate clockwise, while $\mathcal{M}_2$ and $\mathcal{M}_4$ rotate counter-clockwise. The coordinate frames, motors, and rotation directions are illustrated in (Figure 3.1):

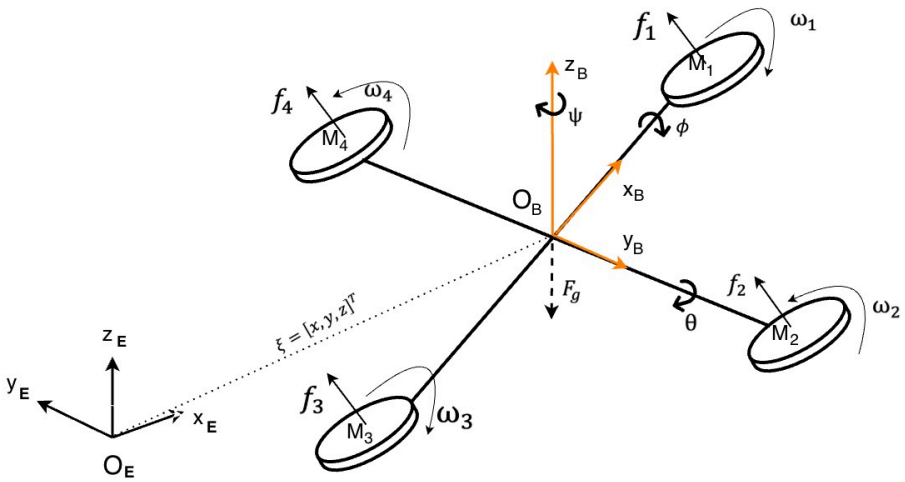


Figure 3.1: Quadrotor schematic illustrating the inertial (O_E) and body (O_B) coordinate frames, gravitational force (F_g), individual rotor thrusts (f_i), and motor rotation directions in a cross configuration

Based on blade element momentum theory, the thrust f_i generated by the i-th motor is proportional to the square of its rotational speed ω_i. This relationship is governed by $k_f = C_T \rho A r^2$, where C_T is thrust coefficient, ρ is air density, r is rotor radius, and A is disk area, allowing the relationship to be expressed as:

$$f_i = k_f \omega_i^2 \tag{3.10}$$

Similarly, the aerodynamic drag moment τ_i opposes the direction of propeller rotation. This moment is governed by the drag coefficient $k_t = C_Q \rho A r^3$, where C_Q is the drag coefficient. This defines the moment as:

$$\tau_i = k_t \omega_i^2 \tag{3.11}$$

To enhance model fidelity, gyroscopic effects resulting from the high-speed rotation of the propellers and the angular motion of the airframe are included. The gyroscopic torque

τ_{gyro} depends on the rotor inertia J_r and the residual angular velocity of the propulsion system, denoted as $\Omega_r = \omega_1 + \omega_3 - \omega_2 - \omega_4$. The resulting moment is expressed as:

$$\tau_{gyro} = J_r(\omega \times \mathbf{e}_3)\Omega_r \tag{3.12}$$

where $\mathbf{e}_3 = [0, 0, 1]^T$ represents the unit vector along the rotor axis in the body frame, and ω is the body angular velocity vector.

3.2.4 Simplified Model for Control

For the purpose of control system design, the general mathematical model is reduced via standard simplifying assumptions. The quadrotor structure is assumed to be rigid and symmetrical ($I_{xy} = I_{yz} = I_{zx} = 0$), with the center of gravity coincident with the geometric center. Furthermore, blade flapping and complex aerodynamic effects are neglected for near-hover and low-speed flight regimes.

The control inputs are mapped to a total scalar thrust, f_{thrust}, and three control torques, $\tau_\phi, \tau_\theta, \tau_\psi$. Incorporating linear and rotational damping terms to account for air resistance, the simplified equations of motion are expressed as:

$$\begin{aligned}
\ddot{x} &= \frac{1}{m}\left[(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)f_{thrust} - d_l\dot{x}\right] \\
\ddot{y} &= \frac{1}{m}\left[(\cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi)f_{thrust} - d_l\dot{y}\right] \\
\ddot{z} &= \frac{1}{m}\left[(\cos\phi\cos\theta)f_{thrust} - mg - d_l\dot{z}\right] \\
\dot{p} &= \frac{1}{I_{xx}}\left[\tau_\phi - qr(I_{zz} - I_{yy}) - d_r p\right] \\
\dot{q} &= \frac{1}{I_{yy}}\left[\tau_\theta - pr(I_{xx} - I_{zz}) - d_r q\right] \\
\dot{r} &= \frac{1}{I_{zz}}\left[\tau_\psi - pq(I_{yy} - I_{xx}) - d_r r\right]
\end{aligned} \tag{3.13}$$

where d_l and d_r represent the lumped linear and rotational drag coefficients, respectively. This set of coupled differential equations defines the plant model utilized for the derivation and stability analysis of the proposed control laws.

3.2.5 Motor Mixer Algorithm

The relationship between the control inputs $U = [f_{thrust}, \tau_\phi, \tau_\theta, \tau_\psi]^T$ and the squared motor speeds $\Omega = [\omega_1^2, \omega_2^2, \omega_3^2, \omega_4^2]^T$ is defined by the allocation matrix Γ:

$$\begin{bmatrix} f_{thrust} \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} k_f & k_f & k_f & k_f \\ 0 & -Lk_f & 0 & Lk_f \\ -Lk_f & 0 & Lk_f & 0 \\ -k_\tau & k_\tau & -k_\tau & k_\tau \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{3.14}$$

where L is the arm length, and k_f, k_τ are the aerodynamic force and torque constants derived in the previous section[75].

3.2.6 Linearized State-Space Model

The nonlinear dynamics of the quadrotor are linearized to facilitate the design of the predictive control and monitoring algorithms. We therefore, require a LTI approximation of the system. This is achieved by applying a first-order Taylor series expansion to the nonlinear vector field $f(\mathbf{x}, \mathbf{u})$ around the equilibrium point, neglecting higher-order terms. The resulting linearized dynamics are expressed in terms of the deviation variables $\delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_{eq}$ and $\delta\mathbf{u}(t) = \mathbf{u}(t) - \mathbf{u}_{eq}$:

$$\delta\dot{\mathbf{x}}(t) = A_c\delta\mathbf{x}(t) + B_c\delta\mathbf{u}(t)$$

consistent with the simplified model derived earlier, the state vector $\mathbf{x} \in \mathbb{R}^{12}$ is defined to include the inertial positions, velocities, Euler angles and body angular rates:

$$\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, p, q, r]^T$$

The control input vector $\mathbf{u} \in \mathbb{R}^4$ comprises the total thrust and the control torques as:

$$\mathbf{u} = [f_{thrust}, \tau_\phi, \tau_\theta, \tau_\psi]^T$$

The Taylor approximation is performed around a hover equilibrium $(\mathbf{x}_{eq}, \mathbf{u}_{eq})$, defined as a stationary point in space where the quadrotor maintains a constant altitude with zero linear and angular velocities. Solving the algebraic system $f(\mathbf{x}_{eq}, \mathbf{u}_{eq}) = 0$ gives the nominal operating conditions:

$$\mathbf{x}_{eq} = [x_0, y_0, z_0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$$

To maintain this hover state against gravity, the nominal input $\mathbf{u}_{eq}$ must satisfy the vertical force balance:

$$\mathbf{u}_{eq} = [mg, 0, 0, 0]^T$$

The continuous system matrices A_c and B_c correspond to the Jacobian matrices of the Taylor expansion. These are obtained by evaluating the partial derivatives of the dynamic equations with respect to $\mathbf{x}$ and $\mathbf{u}$ at the equilibrium point. Applying the small-angle approximation ($\sin\alpha \approx \alpha, \cos\alpha \approx 1$) valid near the equilibrium, the translational dynamics decouple into linear relationships and the resulting continuous system matrix $A_c \in \mathbb{R}^{12\times12}$ is defined as:

$$A_c = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{eq} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\frac{d_l}{m} & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{d_l}{m} & 0 & -g & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -\frac{d_l}{m} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{d_r}{I_{xx}} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{d_r}{I_{yy}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{d_r}{I_{zz}}
\end{bmatrix}$$

The input matrix $B_c \in \mathbb{R}^{12\times4}$, describing the control coupling of the actuators, is derived as:

$$B_c = \left.\frac{\partial f}{\partial u}\right|_{eq} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\frac{1}{m} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & \frac{1}{I_{xx}} & 0 & 0 \\
0 & 0 & \frac{1}{I_{yy}} & 0 \\
0 & 0 & 0 & \frac{1}{I_{zz}}
\end{bmatrix}$$

Given that the CPS operates with digital controllers and monitors, the continuous-time linear model must be adapted to align with the sampled-data nature of the implementation.

To achieve this, the continuous system matrices A_c and B_c are discretized assuming a ZOH on the inputs over the sampling period T_s.The resulting discrete-time system matrices, denoted as A and B, captures the exact evolution of the linear dynamics between sampling instants and later used in subsequent simulations.

The accuracy of the simulation results depends on the physical properties of the quadrotor. The parameters summarized in (Table 3.1) are used consistently across the entire research framework. These values define the rigid body properties for the full nonlinear equations of motion and serve as the constant coefficients for the linearized state-space matrices.

Table 3.1: Quadrotor Physical and Aerodynamic Parameters

Parameter	**Symbol**	**Value**	**Unit**
Mass	m	3.95	kg
X-axis Inertia	I_{xx}	0.363	$\mathrm{kg \cdot m^2}$
Y-axis Inertia	I_{yy}	0.363	$\mathrm{kg \cdot m^2}$
Z-axis Inertia	I_{zz}	0.651	$\mathrm{kg \cdot m^2}$
Arm Length	L	0.45	m
Thrust Coefficient	k_f	3.13×10^{-5}	$\mathrm{N \cdot s^2/rad^2}$
Drag Coefficient	k_τ	7.50×10^{-7}	$\mathrm{N \cdot m \cdot s^2/rad^2}$
Gravity	g	9.81	$\mathrm{m/s^2}$

3.3 The Four-Tank System

The quadruple tank process is a multivariable laboratory system that has become a benchmark in control theory [76]. The system has four interconnected water tanks, such as tank 1-4, and two pumps, as shown in (Figure 3.2). The physical arrangement is such that the tanks are placed in pairs, such as tank 3 is located above tank 1, and tank 4 is located above tank 2. The water from the upper tanks flows freely by gravity into the corresponding lower tanks, while lower tanks (1 and 2) outflow into reservoir at the bottom.

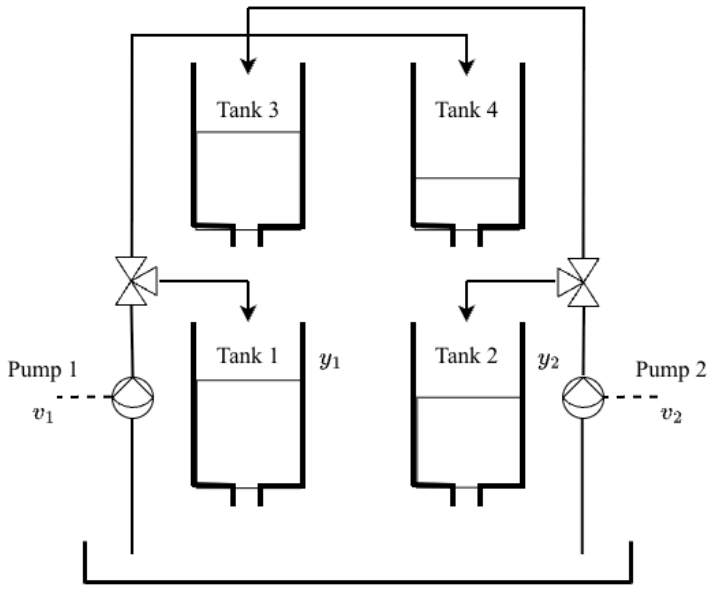


Figure 3.2: Schematic of the quadruple tank system illustrating the flow paths. The system inputs are the pump voltages (v_1, v_2) and the system outputs are the water levels in the lower tanks (y_1, y_2). The valves distribute flow between the lower and upper tanks to create cross-coupling effects

The system inputs are the voltages v_1 and v_2 applied to the two pumps. These pumps lift water from the reservoir to the tanks. A feature of this system is the use of two three-way valves that distribute the flow from each pump between an upper tank and a lower tank, where pump 1 delivers a fraction of its flow (γ_1) directly to tank 1 and the remaining fraction $(1-\gamma_1)$ to tank 4 (the cross-feed) and pump 2 delivers a fraction of its flow (γ_2) directly to tank 2 and the remaining fraction $(1-\gamma_2)$ to tank 3 (the cross-feed). The parameters $\gamma_1, \gamma_2 \in (0, 1)$ define the valve ratios that regulate flow distribution. These settings define the governing dynamics of the system. By varying these ratios, the process can be configured as minimum phase (where the pumps primarily feed the lower tanks) or non-minimum phase (where the pumps primarily feed the upper tanks), making it an ideal platform for testing safety frameworks against complex dynamic behaviors [76]. For the scope of this research, the system is configured in the Minimum Phase setting. While much of the security literature uses the non-minimum phase configuration to study stealthy, unbounded zero-dynamics attacks, analyzing those specific vulnerabilities is outside the current focus of this work. Given that this represents the initial integration of the proposed defense mechanisms with the testbed, the minimum-phase tuning was selected to reduce foundational control complexity. This configuration provides a standard baseline that allows to reliably isolate and evaluate the efficacy of the safety and rejuvenation frameworks against actuator attacks.

3.3.1 Nonlinear Mathematical Model

The mathematical model of the process is derived from mass balance equations and Bernoulli's law for flow through a hole. Let h_i denote the water level in tank i, and A_i

denote the cross-sectional area of tank i. The outlet hole of each tank has a cross-sectional area a_i. Applying the principle of conservation of mass (rate of accumulation = inflow - outflow), the dynamics for the four tanks are described by the following set of nonlinear differential equations:

$$\frac{dh_1}{dt} = -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1}v_1 \tag{3.15a}$$

$$\frac{dh_2}{dt} = -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2}v_2 \tag{3.15b}$$

$$\frac{dh_3}{dt} = -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2}{A_3}v_2 \tag{3.15c}$$

$$\frac{dh_4}{dt} = -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1}{A_4}v_1 \tag{3.15d}$$

where, g denotes the acceleration due to gravity, while k_1 and k_2 represent the pump flow constants (measured in $cm^3/V \cdot s$) that relate the input voltage to the volumetric flow rate. The term $a_i\sqrt{2gh_i}$ describes the outflow from tank i in accordance with Torricelli's law. For this model, the system states are defined as the water levels $x = [h_1, h_2, h_3, h_4]^T$, and the control inputs consist of the pump voltages $u = [v_1, v_2]^T$. Finally, the system outputs $y = [h_1, h_2]^T$, correspond to the water levels of the lower tanks as measured by level sensors.

3.3.2 Linearized State-Space Model

The nonlinear dynamics are linearized to simplify the design of the predictive control and monitoring algorithms. We require an LTI approximation of the system, by applying a first-order Taylor series expansion to the nonlinear equations around an operating point. The resulting linearized dynamics are expressed in terms of the deviation variables $\delta x(t) = x(t) - x_{eq}$ and $\delta u(t) = u(t) - u_{eq}$:

$$\delta\dot{x}(t) = A_c\delta x(t) + B_c\delta u(t)$$

The Taylor approximation is performed around a stationary equilibrium (x_{eq}, u_{eq}). Solving the algebraic system $f(x_{eq}, u_{eq}) = 0$ gives the nominal operating conditions where the inflow matches the outflow for all tanks:

$$x_{eq} = [h_1^0, h_2^0, h_3^0, h_4^0]^T$$

$$u_{eq} = [v_1^0, v_2^0]^T$$

The system matrix A_c and input matrix B_c are derived from the Jacobian of the nonlinear equations evaluated at equilibrium gives:

$$A_c = \begin{bmatrix} -\frac{1}{T_1} & 0 & \frac{A_3}{A_1T_3} & 0 \\ 0 & -\frac{1}{T_2} & 0 & \frac{A_4}{A_2T_4} \\ 0 & 0 & -\frac{1}{T_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{T_4} \end{bmatrix}, \quad B_c = \begin{bmatrix} \frac{\gamma_1 k_1}{A_1} & 0 \\ 0 & \frac{\gamma_2 k_2}{A_2} \\ 0 & \frac{(1-\gamma_2)k_2}{A_3} \\ \frac{(1-\gamma_1)k_1}{A_4} & 0 \end{bmatrix}$$

where T_i represents the time constant for tank i at the equilibrium level h_i^0, defined as:

$$T_i = \frac{A_i}{a_i}\sqrt{\frac{2h_i^0}{g}}$$

The measured output equation is given by $y(t) = C\delta x(t)$, where C selects the lower tank levels:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

3.3.3 System Parameters and Configuration

The specific parameters used for the modeling and simulation in this work are listed in (Table 3.2). As noted, the valve parameters γ_1 and γ_2 are set to 0.60. Since $\gamma_1 + \gamma_2 > 1$, the multivariable zero is located in the left half-plane, confirming the minimum phase characteristic of this model.

Table 3.2: Model Parameters for the Four-Tank System

Parameter	Symbol	Value	Unit
Cross-section of tanks	A	28	cm^2
Cross-section of outlet holes	a	0.071	cm^2
Gravitational acceleration	g	981	cm/s^2
Pump flow constant 1	k_1	3.33	cm^3/Vs
Pump flow constant 2	k_2	3.33	cm^3/Vs
Valve parameter 1	γ_1	0.60	–
Valve parameter 2	γ_2	0.60	–
Maximum pump voltage	$u_{\max}$	15	V
Maximum tank level	$h_{\max}$	50	cm

3.4 Fundamentals of Model Predictive Control (MPC)

MPC is a class of advanced algorithms that compute control actions by solving an optimization problem based on a dynamic model of the process. At each sampling instant, MPC predicts the plant's future behavior and selects inputs that satisfy performance objectives and operational limits [77].

In recent decades, this methodology has become the preferred approach for systems that must operate within strict constraints, finding adaptation in fields such as process control, automotive systems, and robotics [77]. MPC determines control actions by solving a constrained optimization problem on-line, unlike classical approaches such as Proportional-Integral-Derivative (PID) or Linear Quadratic Regulator (LQR) that rely on a fixed, pre-computed control law [78].

The MPC framework is organized around three core principles: the process model, the cost function and the receding horizon method. At a given sampling instant k, the controller uses the current system state, $x(k)$, to solve an optimal control problem over a finite prediction horizon, N [77]. This optimization produces an optimal sequence of future control inputs:

$$U = [u_{0|k}, u_{1|k}, \ldots, u_{N-1|k}]^\top$$

Although the controller calculates a trajectory for the entire horizon, it applies only the first element, $u_{0|k}$, to the system. At the subsequent sampling instant $(k+1)$, the horizon shifts forward by one step, the system state is re-measured or estimated, and the optimization problem is resolved. This iterative strategy, illustrated in (Figure 3.3) is known as the receding horizon principle [79], [80], [81]. This formulation presents several operational advantages over classical control methodologies. It replaces the reactive corrections of PID control with a predictive strategy that acts early to reduce tracking errors. In contrast to unconstrained methods like LQR, MPC directly incorporates operational limits, ensuring all control actions remain feasible. Additionally, it naturally handles Multi-Input Multi-Output (MIMO) interactions within a single optimization problem, eliminating the need for complex decentralized tuning.

Despite these significant operational advantages, MPC implementation faces certain limitations. The primary drawback is the high computational burden imposed by solving a complex optimization problem at every sampling instant, which can restrict its use in applications with extremely fast dynamics or limited processing hardware. Furthermore, the controller's performance is heavily dependent on the accuracy of the internal model and discrepancies between the model and the actual physical plant, known as plant-model

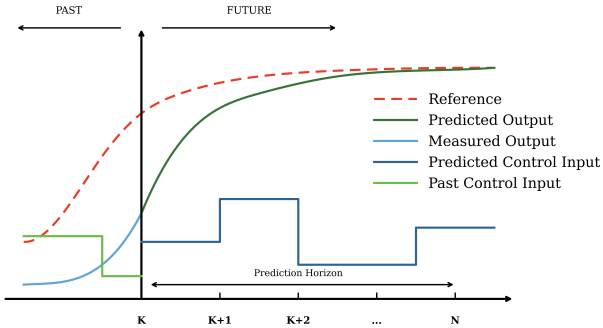


Figure 3.3: Illustration of the MPC strategy, showing the measured past states, the computed future control sequence over the prediction horizon, and the predicted output tracking a reference trajectory, modified version from [82]

mismatch can lead to suboptimal performance, steady-state errors, or even instability. Finally, the design complexity is considerably higher than classical methods, requiring precise system identification and the selection of appropriate tuning weights to balance competing control objectives [77].

3.4.1 Mathematical Formulation of MPC

The implementation of the receding horizon strategy requires a rigorous mathematical representation of the plant dynamics and a quantifiable measure of performance. This section details the formulation of the finite-horizon optimal control problem and derives the prediction matrices used to transform the dynamic system into a static optimization task. A similar formulation will be used in the next sections to design the antagonistic MPC.

3.4.2 System Dynamics

As established in Section 2.3.2, the internal prediction model used by the controller is based on the discrete-time LTI formulation:

$$x(k+1) = A\,x(k) + B\,u(k) \tag{3.16a}$$

$$y(k) = C\,x(k) \tag{3.16b}$$

where A, B, and C are the system matrices derived via the linearization and discretization process previously detailed.

For the purpose of MPC, it is necessary to distinguish between the actual system state at the current time step and the future trajectories calculated by the optimizer. We therefore introduce the notation $x_{k+i|k}$ to denote the predicted state vector at time step $k+i$, based on the information available at time k. Consistent with this notation, $x_{k|k}$ corresponds to the current measured (or estimated) state $x(k)$. This distinction is useful for the receding horizon strategy, as the controller computes a sequence of future states $\{x_{k+1|k}, \ldots, x_{k+N|k}\}$ while the actual plant exists only in the state $x(k)$ at the current time step.

3.4.3 The Cost Function

The control objective is formulated as a quadratic cost function J, which penalizes deviations of the predicted states from the reference trajectory and minimizes the control effort. The cost function, defined over the prediction horizon N, is given by:

$$J(U, x(k)) = \sum_{i=0}^{N-1} \left((x_{k+i|k} - x_{ref,k+i})^\top Q (x_{k+i|k} - x_{ref,k+i}) + u_{i|k}^\top R u_{i|k} \right) \tag{3.17}$$

where N is the prediction horizon length and $x_{ref,k+i}$ represents the desired reference state at the future time step $k+i$. In trajectory tracking scenarios, this sequence is provided by a higher-level motion planner, allowing the MPC to anticipate future path curvature. In stabilization and regulation scenarios, x_{ref} remains constant over the prediction horizon. The matrices $Q \succeq 0$ and $R \succ 0$ are the state and input weighting matrices, respectively. The term $(x - x_{ref})^\top Q(x - x_{ref})$ represents the weighted squared error of the states, while $u^\top Ru$ represents the cost of actuation. The controller seeks to find the optimal input sequence U that minimizes this total accumulated cost [81], [83].

3.4.4 Prediction Modeling

To solve the optimization problem, it is necessary to express the future states explicitly as a function of the current state $x(k)$ and the future control sequence. By iterating the discrete-time state-space model forward in time, the recursive prediction equations can be written as:

$$x_{k+1|k} = Ax_{k|k} + Bu_{0|k} \tag{3.18a}$$

$$\begin{aligned} x_{k+2|k} &= Ax_{k+1|k} + Bu_{1|k} \\ &= A^2x_{k|k} + ABu_{0|k} + Bu_{1|k} \end{aligned} \tag{3.18b}$$

$$\vdots$$

$$x_{k+i|k} = A^i x_{k|k} + \sum_{j=0}^{i-1} A^{i-1-j} Bu_{j|k} \tag{3.18c}$$

This formulation given by [79] allows the predicted states and control inputs to be stacked into the vectors $\mathbf{X}$ and $\mathbf{U}$, defined as

$$\mathbf{X} = \begin{bmatrix} x_{k+1|k} \\ x_{k+2|k} \\ \vdots \\ x_{k+N|k} \end{bmatrix} \in \mathbb{R}^{Nn_x}, \quad \mathbf{U} = \begin{bmatrix} u_{k|k} \\ u_{k+1|k} \\ \vdots \\ u_{k+N-1|k} \end{bmatrix} \in \mathbb{R}^{Nn_u} \tag{3.19}$$

The relationship between the predicted state trajectory, the current state, and the future control sequence can then be expressed in compact form as

$$\mathbf{X} = \Psi x(k) + \Theta \mathbf{U} \tag{3.20}$$

where the prediction matrices $\Psi \in \mathbb{R}^{Nn_x \times n_x}$ and $\Theta \in \mathbb{R}^{Nn_x \times Nn_u}$ are given by:

$$\Psi = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \; \Theta = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix} \tag{3.21a}$$

3.4.5 Quadratic Programming Derivation

To use standard Quadratic Programming (QP) solvers, the cost function defined in Eq. 3.17 must be reformulated in terms of the stacked decision vector $\mathbf{U}$ [84]. First, we define the block-diagonal weighting matrices $\mathcal{Q} \in \mathbb{R}^{Nn_x \times Nn_x}$ and $\mathcal{R} \in \mathbb{R}^{Nn_u \times Nn_u}$, which represent the cumulative weights over the prediction horizon:

$$\mathcal{Q} = \begin{bmatrix} Q & 0 & \cdots & 0 \\ 0 & Q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q \end{bmatrix}, \quad \mathcal{R} = \begin{bmatrix} R & 0 & \cdots & 0 \\ 0 & R & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R \end{bmatrix} \tag{3.22}$$

Similarly, let $\mathbf{X}_{\text{ref}}$ denote the reference trajectory vector extended over the prediction horizon, matching the dimensions of the predicted state vector:

$$\mathbf{X}_{\text{ref}} = \begin{bmatrix} x_{\text{ref},k+1}^T & x_{\text{ref},k+2}^T & \cdots & x_{\text{ref},k+N}^T \end{bmatrix}^T \tag{3.23}$$

Using these definitions, the cost function $J(\mathbf{U}, x(k))$ can be written in compact vector form as

$$J(\mathbf{U}, x(k)) = (\mathbf{X} - \mathbf{X}_{\text{ref}})^T \mathcal{Q} (\mathbf{X} - \mathbf{X}_{\text{ref}}) + \mathbf{U}^T \mathcal{R} \mathbf{U} \tag{3.24}$$

Substituting the prediction model $\mathbf{X} = \Psi x(k) + \Theta \mathbf{U}$ into the cost function gives:

$$J(\mathbf{U}, x(k)) = (\Psi x(k) + \Theta \mathbf{U} - \mathbf{X}_{\text{ref}})^T \mathcal{Q} (\Psi x(k) + \Theta \mathbf{U} - \mathbf{X}_{\text{ref}}) + \mathbf{U}^T \mathcal{R} \mathbf{U} \tag{3.25}$$

Let E_k denote the tracking error term independent of the future control input, defined as

$$E_k = \Psi x(k) - \mathbf{X}_{\text{ref}} \tag{3.26}$$

The cost function can then be expressed as

$$\begin{aligned} J(\mathbf{U}, x(k)) &= (\Theta \mathbf{U} + E_k)^T \mathcal{Q} (\Theta \mathbf{U} + E_k) + \mathbf{U}^T \mathcal{R} \mathbf{U} \\ &= \mathbf{U}^T \Theta^T \mathcal{Q} \Theta \mathbf{U} + 2 E_k^T \mathcal{Q} \Theta \mathbf{U} + E_k^T \mathcal{Q} E_k + \mathbf{U}^T \mathcal{R} \mathbf{U} \end{aligned} \tag{3.27}$$

By grouping the quadratic and linear terms with respect to $\mathbf{U}$, and discarding the constant term $E_k^T \mathcal{Q} E_k$ which does not affect the location of the minimum, the standard QP objective form is obtained:

$$J(\mathbf{U}) = \frac{1}{2} \mathbf{U}^T H \mathbf{U} + \mathbf{f}^T \mathbf{U} \tag{3.28}$$

where the Hessian matrix H and the gradient vector $\mathbf{f}$ are given by:

$$H = 2 \left(\Theta^T \mathcal{Q} \Theta + \mathcal{R} \right) \tag{3.29a}$$

$$\mathbf{f} = 2 \Theta^T \mathcal{Q} \left(\Psi x_k - \mathbf{X}_{\text{ref}} \right) \tag{3.29b}$$

This derivation explicitly relates the system dynamics and tuning weights to the static optimization problem solved at each time step k. Since $Q \succeq 0$ and $R \succ 0$, the Hessian matrix H is positive definite, ensuring a convex optimization problem with a unique global minimum.

3.4.6 The Constrained Optimization Problem

At each sampling instant k, the MPC controller solves a constrained optimization problem to find the optimal control sequence U. While the cost function derived in Eq. (3.28) drives the system toward performance goals (such as tracking a trajectory), the optimization must satisfy the physical and safety limitations of the system. These constraints consist of actuator saturation (input constraints) and safety boundaries (state constraints). To include these limits into the QP solver, we must express them in terms of the decision variable U. The constraints for a single time step are defined as:

$$u_{min} \leq u(k) \leq u_{max}, \quad x_{min} \leq x(k) \leq x_{max} \tag{3.30}$$

where $u_{min}, u_{max} \in \mathbb{R}^{n_u}$ represent the actuator limits (e.g., rotor thrust in the quadrotor example), and $x_{min}, x_{max} \in \mathbb{R}^{n_x}$ represent the safety limits on states (a constraint on the 4-tanks system, e.g. "max allowed level"). Since the decision vector U stacks the control inputs for the entire prediction horizon N, these single-step constraints must be replicated N times to apply to every predicted step. We formulate the stacked bound vectors $\mathcal{U}_{min}$ and $\mathcal{U}_{max}$ using the kronecker product operator $\otimes$, which repeats the limit vectors N times:

$$\mathcal{U}_{min} = \mathbf{1}_N \otimes u_{min}, \quad \mathcal{U}_{max} = \mathbf{1}_N \otimes u_{max} \tag{3.31}$$

where $\mathbf{1}_N$ is a column vector of ones of length N. This allows us to write the input constraints for the entire horizon as $\mathcal{U}_{min} \leq U \leq \mathcal{U}_{max}$. Similarly, the state constraints must be satisfied for all predicted states X. We define the stacked state bounds $\mathcal{X}_{min} = \mathbf{1}_N \otimes x_{min}$ and $\mathcal{X}_{max} = \mathbf{1}_N \otimes x_{max}$. However, the solver optimizes U, not X, therefore, we substitute the prediction model equation $X = \Psi x(k) + \Theta U$ into the inequality to express the state limits directly in terms of the control inputs:

$$\mathcal{X}_{min} \leq \Psi x(k) + \Theta U \leq \mathcal{X}_{max} \tag{3.32}$$

Standard QP algorithms generally require constraints to be cast in the canonical inequality form $A_{ineq}x \leq b_{ineq}$. To map the physical constraints to this format, the double-sided box constraints must be decomposed into independent single-sided linear inequalities and rearranged to isolate the decision variable U on the left-hand side. We first address the input constraints defined by $U_{min} \leq U \leq U_{max}$. This double-sided inequality is split into an upper bound $I \cdot U \leq U_{max}$ and a lower bound $U \geq U_{min}$. To satisfy the canonical less than or equal to requirement, the lower bound is multiplied by negative one, resulting in $-I \cdot U \leq -U_{min}$.

A similar procedure is applied to the state constraints $X_{min} \leq \Psi x(k) + \Theta U \leq X_{max}$. Since the optimization variable is U, the term dependent on the current state, $\Psi x(k)$, must be moved to the right-hand side of the inequality. The upper limit transforms directly into $\Theta U \leq X_{max} - \Psi x(k)$. For the lower limit, we again reverse the inequality direction, transforming $X_{min} \leq \Psi x(k) + \Theta U$ into $-\Theta U \leq -X_{min} + \Psi x(k)$.

Finally, we stack these linear relations to form a single matrix inequality. By separating the constant physical limits from the terms varying with the current state $x(k)$, we obtain the structured affine constraint formulation:

$$GU \leq S + Ex(k) \tag{3.33a}$$

$$\begin{bmatrix} I \\ -I \\ \Theta \\ -\Theta \end{bmatrix} U \leq \begin{bmatrix} U_{\max} \\ -U_{\min} \\ X_{\max} \\ -X_{\min} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\Psi \\ \Psi \end{bmatrix} x(k) \tag{3.33b}$$

where G is constant matrix, S is the vector of constant constraints, and E is the state feedback matrix. The MPC algorithm thus solves the following quadratic program at every time step:

$$\min_{U} \quad \frac{1}{2}U^\top HU + f^\top U \tag{3.34a}$$

$$\text{subject to} \quad GU \leq S + Ex(k) \tag{3.34b}$$

solving this problem produces the optimal sequence $U = [u_{0|k}, u_{1|k}, \ldots, u_{N-1|k}]^\top$. Consistent with the receding horizon principle, only the first element $u(k) = u_{0|k}$ is applied to the physical plant. The remaining predicted elements are discarded, and the entire optimization process is repeated at the next sampling instant $k+1$ using the updated state measurement [79], [81].

3.5 Robust Model Predictive Control

In the standard MPC formulation, the controller was derived under the assumption of a deterministic system model. However, in realistic CPS, this assumption is an idealization that rarely holds in practice. Operational environments are subject to uncertainty arising from sensor noise, unmodeled dynamics, and exogenous disturbances.To account for these factors, the system model must be extended to a LTI system with additive bounded uncertainty:

$$x(k+1) = Ax(k) + Bu(k) + Dw(k)$$

where $w(k) \in \mathbb{R}^{n_w}$ represents the uncertainty acting on the system at time step k, and $D \in \mathbb{R}^{n_x \times n_w}$ is the disturbance distribution matrix. While D is often used to map external disturbances, it can also be structured to embed modeling and linearization errors. Following the methodology of Chen and Patton [85], the term $Dw(k)$ can serve as a representation of the mismatch between the nonlinear plant and the linearized model. In this framework, the matrix D (also often denoted as E in literature) is used to map the nonlinear residuals, such as the unmodeled exponential terms in a chemical reactor's Arrhenius dynamics onto the system states. This effectively transforms state-dependent nonlinearities into a bounded pseudo-disturbance, allowing the linear controller to remain mathematically tractable while accounting for the errors introduced during the linearization process [85]. We assume the disturbance vector $w(k)$ is bounded within a compact, convex set $\mathbb{W}$:

$$w(k) \in \mathbb{W}, \quad \forall k \geq 0$$

The set $\mathbb{W}$ denotes the bounded uncertainty set used in the robust MPC formulation. It captures both exogenous disturbances (e.g., wind gusts, sensor noise) and parametric model mismatches (e.g., linearization errors). The combined set $\mathbb{W}$ is typically constructed using Minkowski sums of the individual uncertainty sets, followed by a convex-hull approximation to ensure tractability. In the domain of robust control, addressing this bounded uncertainty is approached through different frameworks, ranging from estimating the disturbance to optimizing against its worst-case realization. The following subsections outline the two standard uncertainty methods: the Naive approach and the Conservative approach.

3.5.1 The Naive Approach

The naive approach simplifies the robust control problem by treating the uncertain elements as deterministic, relying on a best-estimate of the disturbance [86], [87]. Rather than assuming the disturbance to be simply zero, this framework actively estimates the disturbance sequence $\hat{w}_{i|k}$ using real-time data and system observables. In the existing literature, obtaining this estimate is typically achieved through various sophisticated techniques, including Extended State Observer (ESO), Moving Horizon Estimation (MHE) or learning-based predictors to forecast environmental dynamics [88], [89], [90]. Once this estimate $\hat{w}$ is obtained, the controller proceeds to solve the optimization problem accordingly. The optimization problem uses this estimate to compute a control input that counteracts the estimated disturbance. The mathematical formulation solves the following minimization problem at each time step k:

$$\min_{U} \quad J = \sum_{i=0}^{N-1} \left(x_{k+i|k}^{\top} Q x_{k+i|k} + u_{k+|k}^{\top} R u_{k+i|k} \right) \tag{3.35a}$$

$$\text{subject to} \quad x_{k+i+1|k} = A x_{k+i|k} + B u_{k+i|k} + D \hat{w}_{k+i|k} \tag{3.35b}$$

$$x_{\min} \leq x_{k+i|k} \leq x_{\max} \tag{3.35c}$$

$$u_{\min} \leq u_{k+i|k} \leq u_{\max} \tag{3.35d}$$

$$\hat{w}_{k+i|k} \in \mathcal{W} \tag{3.35e}$$

$$x_{0|k} = x_{init} \tag{3.35f}$$

here, the constraint $\hat{w}_{k+i|k} \in \mathcal{W}$ is a parameter provided by the chosen estimation method, transforming the uncertain dynamics into standard deterministic linear equalities. While computationally efficient, this formulation is insufficient for modeling a worst-case scenario. Specifically, because this approach neglects the prediction error covariance, it offers no theoretical guarantees for constraint satisfaction when the actual disturbance deviates from the estimate $\hat{w}_{k+i|k}$. Therefore, the system remains vulnerable to infeasibility during sudden, unpredicted environmental shifts, giving it unsuitable for safety-critical applications requiring strict robustness.

3.5.2 The Conservative Approach

The conservative approach, widely recognized as Robust Model Predictive Control (RMPC), addresses the limitation of safety risks in the naive method by explicitly incorporating the bounded uncertainty set $\mathcal{W}$ into the optimization problem. Instead, optimizing for a single best-estimate trajectory, this framework seeks a control policy that guarantees system safety and feasibility for every possible disturbance realization $w \in \mathcal{W}$.

The assumption is a worst-case design framework wherein controller must ensure that constraints are satisfied and cost function is minimized, even if the disturbance sequence acts in the most antagonistic manner possible against the system objectives. The optimization problem becomes a semi-infinite programming problem, as it must hold for an infinite number of possible disturbance realizations within the set. To address this computationally, the approach is generally categorized into two formulations:

A. Constraint-Tightening Method (Robustness via Constraint Restrictions) The controller minimizes the cost of the nominal trajectory while tightening the constraints that are robust to worst-case uncertainty.

B. Min–Max Formulation (Robustness via Worst-Case Cost) The min–max formulation addresses disturbances by explicitly modeling their impact within the optimization problem and minimizing for the worst-case cost over all admissible uncertainties, rather than modifying the original constraints.

Further details are provided as follows:

3.5.2.1 Constraint-Tightening Method

The first formulation under the conservative approach ensures safety by explicitly restricting the search space for the controller. The principle is constraint tightening, which decomposes the system dynamics into a deterministic nominal trajectory $(\bar{x}, \bar{u})$ optimized for performance, and an additive error state that captures the effect of disturbances. By tightening the constraints on the nominal trajectory, we create a safety buffer that absorbs the impact of uncertainties.

Theoretical Foundation: The RPI Set The theoretical guarantee of safety in this framework relies on the concept of a Robust Positively Invariant (RPI) set, often referred to as a tube in Robust MPC literature[91]. Let $e(k) = x(k) - \bar{x}(k)$ denote the error between the actual and nominal states. A set Ω is robustly invariant for the error dynamics if, once the error enters the set, it remains inside it for all future time steps and all admissible disturbances $w(k) \in \mathcal{W}$. Combining the nominal trajectory with this error set gives the concept of the tube. Formally, the tube $\mathcal{T}(k)$ at time step k is defined as the Minkowski sum of the nominal state and the invariant set:

$$\mathcal{T}(k) \triangleq \bar{x}(k) \oplus \Omega = \{\bar{x}(k) + e \mid e \in \Omega\}$$

Mathematically, safety is guaranteed if this entire tube $\mathcal{T}(k)$ remains within the physical system limits. If the nominal controller respects these tightened bounds, the actual system is guaranteed to remain safe indefinitely.

Implementation: Worst-Case Reachability While the RPI set provides the theoretical basis for infinite-time safety, computing the exact minimal RPI set for high-dimensional systems can be computationally intensive. Therefore, in this work, we uses a finite-horizon reachability approach that computes the exact worst-case tightening margins required for the specific prediction horizon N. Instead of a fixed invariant tube, we calculate the worst-case reachable set of the error term at each step of the horizon. The tightening margin M_i for prediction step i is defined as the maximum possible accumulated deviation

caused by the disturbance sequence $[w(0), \ldots, w(i-1)]$. This is computed by solving a linear optimization problem (LP) for each constraint dimension:

$$M_i = \max_{w \in \mathcal{W}} \left(\sum_{j=0}^{i-1} A^{i-1-j} D w(j) \right)$$

This formulation, solved via standard LP (e.g., linprog), identifies the specific disturbance sequence within the bounds $w_{min} \leq w \leq w_{max}$ that maximizes the violation. These computed margins are then used to tighten the physical box constraints x_{min} and x_{max} dynamically over the horizon.

Using these computed margins M_i, the physical constraints are tightened to $\bar{x}_{min,i} = x_{min} + M_i$ and $\bar{x}_{max,i} = x_{max} - M_i$. The MPC then solves the standard minimization problem for the nominal trajectory:

$$\min_{\bar{U}} \quad J(\bar{U}, \bar{x}(k)) \tag{3.36a}$$

$$\text{subject to} \quad \bar{x}_{k+i+1|k} = A\bar{x}_{k+i|k} + B\bar{u}_{k+i|k} \tag{3.36b}$$

$$x_{\min} + M_i \leq \bar{x}_{k+i|k} \leq x_{\max} - M_i \tag{3.36c}$$

$$u_{\min} \leq \bar{u}_{k+i|k} \leq u_{\max} \tag{3.36d}$$

$$\bar{x}(k|k) = x_{init} \tag{3.36e}$$

By applying that the nominal plan $\bar{x}$ stays inside these tighter limits, we ensure that the actual system x, which includes the accumulated disturbance effects, effectively stays within the theoretical "safety tube" and never violates the original physical constraints.

Numerical Illustration To visualize the concept of constraint tightening, consider a simple Single-Input Single-Output (SISO) integrator model:

$$x_{k+1} = x(k) + u(k) + w(k)$$

The system is subject to physical constraints where the state is limited to $x \in [-5, 5]$, the input to $u \in [-2, 2]$, and the additive disturbance to $w \in [-1, 1]$.

(Figure 3.4) shows the feasible region for this system in the state-input space (x, u). The red region represents the original feasible set where the nominal system would satisfy constraints if disturbances were ignored ($w = 0$). Note that the corners of the square box are cut off, this is because extreme combinations of state and input (e.g., $x = 5, u = 1$) would drive the next state x_{k+1} outside the safe limits. The green region represents the tightened, robust feasible set. To ensure robust safety, the controller must account for worst-case disturbances. For example, if the system were to target the upper limit of $x_{k+1} = 5$ while a disturbance pushes the state by $w_k = +1$, the resulting state would be 6, causing a violation. To prevent this, the nominal controller must restrict its target to a maximum of $5 - 1 = 4$. Similarly, the lower nominal bound adjusts to -4. Therefore, the tightened feasible region shrinks, creating a safety buffer that guarantees the physical system remains within $[-5, 5]$ regardless of the disturbance realization. This reduced region provides a visual example of the set subtraction operation $\mathcal{X} \ominus \mathcal{W}$.

This method is conservative. As shown, by confining the optimization to the restricted green subspace dictated by low-probability, worst-case events, the controller sacrifices feasible trajectories that lie near the physical boundaries (the red area). Despite these limitations, the approach offers a decisive computational advantage. The mathematically intensive set-theoretic operations required to determine the robust invariant sets and tightening margins are performed entirely offline. This reduces the online computational burden to solving a standard QP problem, ensuring that the robust safety guarantees do not compromise the real-time feasibility of the control loop.

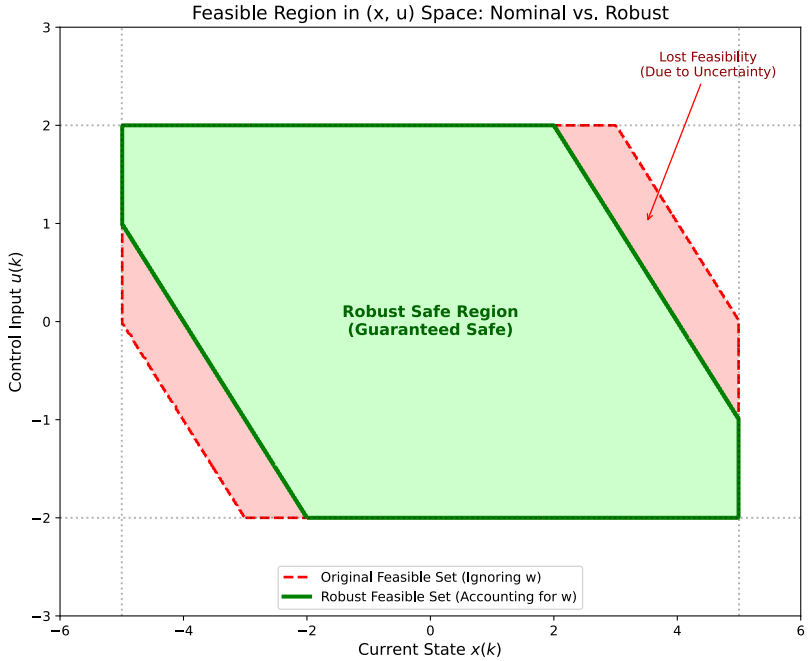


Figure 3.4: Feasible regions in the state-input space (x, u) for the SISO integrator model. The Light Red area indicates the original feasible set for the nominal system without disturbances. The Green area indicates the robust feasible set after constraint tightening. The difference between the two regions represents the safety buffer or lost feasibility required to guarantee safety against the worst-case disturbance $w \in [-1, 1]$

3.5.2.2 Min-Max Formulation (Robustness via Worst-Case Cost)

The limitation of the constraint-tightening approach discussed previously lies in its decoupling of safety from performance. By focusing solely on reducing the feasible region to ensure constraint satisfaction, the controller optimizes a nominal cost function that essentially ignores the presence of disturbances. This creates a gap where a control trajectory might be safe (feasible) but inefficient. To account this, a Min-Max formulation that incorporates uncertainty directly into the objective function is being proposed [79], [92]. Unlike constraint tightening, which works primarily with the constraints, this method modifies the optimization formulation itself to ensure robustness along with performance.

This framework models the control problem as a zero-sum game between the controller (minimizer) and the nature/disturbance (maximizer). Instead of optimizing a single nominal trajectory, the controller seeks a control policy that minimizes the performance index under the worst-case disturbance realization. The general min-max optimization problem is defined as:

$$J^*(x(k)) = \min_{U} \max_{\mathbf{W} \in \mathcal{W}} J(x(k), U, W) \tag{3.37}$$

where, $\mathbf{W}$ represent the disturbance sequences over the prediction horizon N, $W = [w_{0|k}^\top, \ldots, w_{N-1|k}^\top]^\top \in \mathcal{W}^N$ and expanding the cost function over the prediction horizon N gives:

$$J(U, x(k), W) = \sum_{i=0}^{N-1} \left(x_{k+i|k}^\top Q x_{k+i|k} + u_{k+i|k}^\top R u_{k+i|k} \right) \tag{3.38}$$

subject to uncertain dynamics and constraints defined previously. This formulation forces the optimizer to select control inputs that minimize the cost under worst-case disturbance realizations.

Solving this nested min-max problem directly poses an implementation challenge. Standard optimization algorithms generally require the problem to follow a specific

canonical structure (such as a Quadratic Program) to guarantee convergence and computational speed. The nested maximization operator in Eq: 3.37 violates this structure, as it adds a logical search operation within the cost function, preventing the use of efficient, gradient-based QP solvers which require explicit matrix-defined constraints.

To solve this, we reformulate the bilevel optimization into a single-level constrained minimization problem using the epigraph form. We introduce a scalar auxiliary variable γ and reformulate the inner maximization into the constraint set. The objective is formulated to minimize the nominal cost plus the maximum possible excess cost caused by the disturbance:

$$\min_{U,\gamma} \quad J(U, x(k), \mathbf{0}) + \gamma \tag{3.39a}$$

subject to

$$\gamma \geq J(U, x(k), W) - J(U, x(k), \mathbf{0}), \quad \forall W \in \mathbb{W}^N \tag{3.39b}$$

the condition that γ must simultaneously upper-bound the cost difference for every valid disturbance sequence. While this condition theoretically imposes an infinite number of constraints because it must hold for every possible point w within the continuous uncertainty set $\mathcal{W}^N$, however, the coming section details how this can be reduced to a finite, problem using vertex enumeration. Note that the detailed algebraic expansion of the cost difference and the specific vertex enumeration algorithm will be derived in Section 4.3.3 and Section 4.5, respectively, where we use this robust framework to design the proposed antagonistic controller.

3.6 Explicit MPC via Multi-Parametric Programming (mP-QP)

The standard MPC formulation derived in the previous section relies on solving a Quadratic Programming (QP) problem online at every sampling instant. While effective, this repetitive online computation can be infeasible for systems operating under real-time constraints and limited computational resources. In the context of this work, specifically for the predictive safety mechanisms which will be presented in Chapter 5, it is necessary to compute the optimal control structure without the computational delay of an iterative solver. To address these limitations, explicit MPC is used, whereby the online optimization problem is solved offline as a multi-parametric Quadratic Program (mp-QP), giving a piecewise-affine (PWA) state-feedback control law. Instead of considering the state vector $x(k)$ as a fixed numerical value, it is treated as a vector of parameters. This approach shifts the computational burden offline, allowing the optimal control solution to be pre-calculated as an explicit function of the state variables. To derive the multi-parametric formulation, we consider the standard quadratic cost function $J(U)$ derived earlier where the objective function is given by:

$$J(U) = \frac{1}{2} U^\top H U + f^\top U$$

In the standard QP derivation, the linear gradient term f represents the relationship between the current state and the cost defined as:

$$f = 2\Theta^\top \mathcal{Q}(\Psi x(k) - X_{ref})$$

For the purpose of deriving the explicit control law, we consider the regulation problem where the reference state is the origin (i.e., $X_{ref} = 0$). This formulation is sufficient for the safety monitoring objectives of this thesis, as the proposed detectors operate on the deviation of the system from its desired equilibrium. Therefore, this assumption implies no loss of generality, as general tracking problems can be equivalently reformulated as regulation problems by defining $x(k)$ as the error state vector. Under this assumption, the linear term simplifies to a direct linear function of the current state $x(k)$:

$$f(x) = 2\Theta^\top \mathcal{Q} \Psi x(k)$$

We now substitute this explicit definition of $f(x)$ back into the cost function term $f^\top U$. Using the property of the matrix transpose $(ABC)^\top = C^\top B^\top A^\top$, we can rewrite the scalar product as:

$$\begin{aligned} f^\top U &= (2\Theta^\top \mathcal{Q}\Psi x(k))^\top U \\ &= x(k)^\top (2\Psi^\top \mathcal{Q}^\top \Theta) U \end{aligned}$$

Since the weighting matrix $\mathcal{Q}$ is diagonal and symmetric ($\mathcal{Q} = \mathcal{Q}^\top$), this term can be expressed compactly by introducing a new state feedback weighting matrix F:

$$F = 2\Theta^\top \mathcal{Q}\Psi$$

Substituting F into the cost equation, the linear term becomes $(Fx(k))^\top U$, or equivalently $x(k)^\top F^\top U$. This transformation explicitly shows that the cost function is quadratically dependent on U and linearly dependent on the parameter vector $x(k)$.

Finally, the MPC optimization problem can be recast as finding the optimal control sequence U^* for a given parameter x. The constant terms in the constraints are separated from the state-dependent terms (in Equation 3.34b), giving the complete multi-parametric Quadratic Programming (mp-QP) formulation:

$$\min_U \quad \frac{1}{2} U^\top H U + x^\top F^\top U \tag{3.40a}$$

$$\text{subject to} \quad GU \leq S + Ex \tag{3.40b}$$

The constraints define a polyhedral set of feasible parameters and decision variables. The goal of multi-parametric programming is to define the solution $U^*(x)$ for all states x within a bounded polyhedral set $\mathbb{X} \subseteq \mathbb{R}^{n_x}$.

The Solution via Piecewise Affine Control The solution to the mp-QP problem is derived using the Karush-Kuhn-Tucker (KKT) optimality conditions. As shown in the foundational work on explicit MPC [79], the space of feasible states $\mathbb{X}$ can be partitioned into a finite number of polyhedral regions known as Critical Regions (CR_i). Within each critical region, a specific set of inequality constraints remains active. Because the KKT conditions for a fixed set of active constraints are linear, the relationship between the optimal input U^* and the state parameter x is affine. This results in an explicit control law $u_{mpc}(x)$ defined as a Piecewise Affine (PWA) function:

$$u_{mpc}(x) = \begin{cases} K_1 x + c_1 & \text{if } x \in CR_1 \\ K_2 x + c_2 & \text{if } x \in CR_2 \\ \vdots & \vdots \\ K_M x + c_M & \text{if } x \in CR_M \end{cases} \tag{3.41}$$

where M is the total number of regions. Each region CR_i is a convex polyhedron defined by linear inequalities representing its geometric boundaries:

$$CR_i = \{x \in \mathbb{R}^{n_x} \mid H_i x \leq k_i\}$$

Here, H_i and k_i are matrices computed offline that define the hyperplane boundaries of the i-th region. This formulation reduces the online control task to a point location problem, identifying which region contains the current state $x(k)$ (i.e., satisfying $H_i x \leq k_i$) and evaluating the corresponding affine function.

Part 2: Attack and Defense Frameworks

Chapter 4

Antagonistic MPC Control

4.1 Introduction

The operational landscape of modern infrastructure is undergoing a transition. From irrigation networks and automated factories to national power grids, control systems are migrating toward SCADA architectures. This integration of computation, networking, and physical processes has improved the efficiency and allowed for remote manageability but it has simultaneously affected the isolation that once protected these systems [93]. While traditional IT based security focuses on protecting data confidentiality, integrity, and availability, the security of CPS is tied to the preservation of physical stability and operational safety [94]. In a standard IT network, an attack might result in data theft or service denial, in a CPS, a successful attack can lead to complete physical failures, damaging critical infrastructure and endangering human lives [95]. Existing literature in control theory has majorly studied the resilience of systems against stochastic faults and non-malicious disturbances. However, treating intelligent cyber-attacks as mere random noise or passive faults underestimates the adversary's capability. As noted by [34], an intelligent attacker with knowledge of system dynamics can design malicious inputs that are far more destructive than random signal injections. This knowledge need not be available, a adversary can acquire system parameters and topology passively through network sniffing or by employing data-driven system identification techniques to learn the underlying dynamics before launching an active attack [96], [97]. Moreover, current security research has largely focused on defensive monitoring to detecting these attacks, such as false data injection or DoS after they occur. This reactive approach leaves a critical gap in our understanding the true extent of potential vulnerabilities. Rather than asking "How do we detect an intrusion?", this chapter takes an adversarial perspective, assuming an attacker has already accessed the system then "what is the maximum damage they can cause, and how quickly can they destabilize the system?"

To address this, this chapter introduce the antagonistic MPC framework. Unlike traditional regulators (e.g., LQR or standard MPC) that minimize a cost function J to maintain equilibrium and ensure stability, the antagonistic framework inverts this objective [8]. It functions as an anti-controller, seeking to identify optimal input sequences that maximize the deviation from the desired state over a prediction horizon N. While this objective shares similarities with reachability analysis, which computes the set of all attainable states from a given initial condition, antagonistic MPC differs fundamentally in its approach. Reachability analysis typically determines if an unsafe state is reachable, providing a binary safety verification or a static envelope of operation [98]. In contrast, antagonistic MPC dynamically computes the optimal trajectory to violate safety constraints, revealing not just the possibility of failure, but the specific, worst-case attack sequence required to achieve it. To clarify the scope of this research, the focus of this thesis is restricted to servo tracking and regulatory control loops. In these systems, the nominal control objective is defined as the minimization of a baseline cost function (such as tracking error or control effort). Control systems designed for baseline performance maximization (e.g., gives maximization or renewable energy harvesting) fall outside the immediate scope of this work. The application of the proposed methodologies to these performance maximization objectives is considered a valuable direction for future research.

This framework models the worst-case behavior of an intelligent adversary who has successfully compromised the control loop and is fundamentally distinguished from stochastic or static interference methods by its predictive ability and optimization-theoretic formulation. The specific capabilities enabling this advanced adversarial behavior are

such as:

- the attacker predicts how the system will react to an attack N steps into the future, using the model dynamics rather than reacting blindly;
- the attacker calculates inputs that are valid (respecting actuator limits) yet destructive, or inputs that steer the system toward states where recovery becomes difficult;
- the attack is not a static but a dynamically computed trajectory that evolves in real-time as the system state changes.

4.1.1 Operational Applications of the Framework

The antagonistic MPC framework is formulated to model the behavior of an intelligent adversary but its operational value extends beyond theoretical threat modeling. By explicitly computing worst-case trajectories and quantifying the maximum possible deviation, the framework enables defenders to analyze vulnerabilities and time-critical failure modes. In this sense, the method serves as a dual-use tool, such as adversarial in formulation, yet essential for proactive safety assessment.

1. **Vulnerability Monitoring (Real-Time Safety Assessment)** From a defender's perspective, solving the antagonistic control problem provides a metric for real-time safety. By calculating the maximum possible damage ($p^\star$) an attacker could cause from the current state over a future horizon N, we can quantify the system's instantaneous vulnerability.
    - If $p^\star$ exceeds a safety threshold, a warning can be issued.
    - Further, this allows us to derive the Time-to-Violation ($T^\star$). By solving the problem for different horizons, we can determine the minimum time an attacker needs to force the system into a critical failure state. If $T^\star$ is large, operators have time to react; if $T^\star$ is small, automated mitigation is required immediately.
2. **Security Assessment (A Prior Defense Allocation)** This framework can be used offline to identify critical weak points. A typical CPS consists of numerous sensors and actuators. An adversarial aims to gain control of a few subsystems as possible to remain stealthy. By simulating antagonistic attacks on different subsets of actuators, we can determine which components allow an attacker to cause the most damage. This insight allows system architects to prioritize security resources by hardening the most critical subsystems while allocating fewer resources to components where an attacker has limited authority.

4.1.2 Chapter Roadmap

The remainder of this chapter is organized as follows: Section 4.2 sets up the mathematical formulation of the antagonistic controller, defining the optimization problem and provides numerical validation through applying it on quadrotor case study. Section 4.3 addresses the limitations of deterministic models in uncertain environments by using a max-min game theoretic approach to guarantee optimal solution. Section 4.4 reformulates the objective, shifting the focus from simple cost maximization to prioritize violation of safety boundaries. Section 4.5 details the computational solution to these non-convex optimization problems using a vertex enumeration method, showing how the global worst-case attack vectors are identified at the vertices of the control polytope. Finally, the chapter concludes in Section 4.6 by discussing how these offensive control method serve as the foundation for the predictive methods developed in subsequent chapters.

4.2 Antagonistic Control Framework

This section introduces the adversary's control problem, building on the system dynamics of Section 2.3.2 and the predictive control scheme described in Section 3.4.

4.2.1 Problem Formulation

We consider the discrete-time LTI system under the influence of an intelligent adversary. We assume the adversary has successfully breached the control loop and possesses full authority over the input vector $u \in \mathbb{R}^{n_u}$ (or a specific subset thereof). Unlike the nominal controller, which solves a convex minimization problem to regulate the state $x(k)$ to the reference trajectory, the antagonistic controller operates with an inverse objective. The adversary uses the system model (A, B) to compute a sequence of inputs that maximizes the deviation of the system states from the reference trajectory over a finite prediction horizon N. Let $U = [u_{0|k}^\top, u_{1|k}^\top, \dots, u_{N-1|k}^\top]^\top \in \mathbb{R}^{Nn_u}$ denote the sequence of malicious control inputs computed at time step k. The antagonistic cost function over the prediction horizon is defined as:

$$J(U, x(k)) = \sum_{i=0}^{N-1} (x_{k+i|k} - x_{ref,k+i})^\top Q (x_{k+i|k} - x_{ref,k+i}) \tag{4.1}$$

where $Q \succeq 0$ is the state weighting matrix. The antagonistic optimization problem is formally stated as:

$$\max_{U} \quad J(U, x(k)) \tag{4.2a}$$

$$\text{s.t.} \quad x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k} \tag{4.2b}$$

$$u_{min} \leq u_{k+i|k} \leq u_{max} \tag{4.2c}$$

$$x_{\min} \leq x_{i|k} \leq x_{\max}, \tag{4.2d}$$

$$x_{0|k} = x_{init} \tag{4.2e}$$

By respecting these hard constraints, the control signals remain within valid actuator ranges and prevent the immediate triggering of low-level saturation alarms.

4.2.2 Solver for Antagonistic MPC

To solve the optimization problem using standard numerical tools, we use the compact matrix formulation established in Section 3.4.5. By expressing the objective function in terms of the decision variable U and substituting the prediction model, the problem is cast as a Quadratic Program (QP). Since maximizing J is equivalent to minimizing $-J$, the formulation becomes:

$$\min_{U} \quad \tfrac{1}{2} U^\top \bar{H} U + \bar{f}^\top U \tag{4.3a}$$

$$\text{s.t.} \quad GU \leq S + Ex(k)) \tag{4.3b}$$

Here, the constraint matrices G, S, and E are exactly similar derived in nominal formulation, ensuring the attack respects the same physical limitations as the nominal controller. To distinguish the antagonistic objective from the nominal case, we denote the modified Hessian and gradient as $\bar{H}$ and $\bar{f}$:

$$\bar{H} = -2(\Theta^\top \mathcal{Q} \Theta), \quad \bar{f} = -2\Theta^\top \mathcal{Q}(\Psi x(k) - \mathbf{X}_{ref}) \tag{4.4}$$

where $\mathcal{Q}$ is the compact block-diagonal state-weighting matrix over the prediction horizon. Note that the input-weighting matrix $\mathcal{R}$ is omitted (equivalently, set to zero) in this formulation, since the adversary does not seek to penalize control effort, while a non-zero $\mathcal{R}$ could be included to model a stealthy or resource-constrained attacker seeking

to minimize their footprint, we set it to zero to simulate a worst-case adversary that prioritizes maximum state deviation regardless of the control effort required.

The distinction in this formulation arises from the spectral properties of the Hessian matrix $\bar{H}$. In the nominal MPC formulation, the Hessian is positive definite ($H \succ 0$), ensuring strict convexity and the existence of a unique global minimum. Whereas, the inversion of the objective function in the antagonistic formulation gives $\bar{H}$ symmetric negative semi-definite. This shift changes the optimization landscape from a convex minimization to a concave minimization problem and the objective function no longer possesses a unique interior minimum; instead, the optimal solution is pushed toward the boundary of the feasible set, specifically located at one of the vertices (extreme points) of the constraint polytope.

Minimizing a concave quadratic function over a polytope is known to be an NP-hard problem. The KKT conditions, which standard Active-Set or Interior-Point solvers rely on, are necessary but not sufficient for global optimality in non-convex problems. Standard convex QP solvers (e.g., OSQP, qpOASES) typically fail to converge or terminate upon detecting the indefinite Hessian. General nonlinear solvers (e.g., IPOPT) may converge to a local minimum (a local worst-case attack), but cannot guarantee the global worst-case scenario.

However, for the specific scope of real-time safety assessment where the horizon N is short, the attacker can use standard solvers to find locally optimal attacks. Even a sub-optimal solution that lies on the boundary of the constraint set represents a high-impact deviation capable of destabilizing the system.

4.2.3 Case Study: Quadrotor UAV

In this section, we adapt the general antagonistic MPC framework to the specific case of a Quadrotor UAV. This application validates the theoretical concepts on a fast-dynamic system where the time window for a successful attack is critically short. The content of this section originated in the conference publication [99].

4.2.3.1 Attack Strategy and Implementation Logic

The proposed attack strategy relies on the assumption that the adversary has breached the communication channel between the onboard controller and the actuators, gaining the ability to manipulate the rotational speeds of all four motors. We assume the attacker can passively intercept the telemetry data (system states and nominal control inputs) and has a limited time window to cause the damage before detection mechanisms, such as watchdogs, trigger a failsafe. The attacker's objective is to maximize physical destabilization in the shortest possible time.

To achieve this while remaining stealthy during the setup phase, the attack is structured into two distinct stages: Delivery and Affect.

In the first stage, stage I - Delivery, the adversary passively intercepts the communication without modifying the control signals. The antagonistic MPC runs in a shadow mode, using the system's linearized model to iteratively predict the damage achievable over the prediction horizon N_p. To quantify this, we introduce a damage performance index, J_I, which represents the predicted alignment of the thrust vector relative to gravity at the end of the horizon. Mathematically, this index assesses whether the attacker can force the drone into a tilt where recovery is physically impossible:

$$J_I = \cos\left(\min\left(|\phi_{k+N_p|k}|, \frac{\pi}{2}\right)\right) \cdot \cos\left(\min\left(|\theta_{k+N_p|k}|, \frac{\pi}{2}\right)\right) \tag{4.5}$$

This index estimates the effectiveness of the attack if carried out. In a stable hover, $J_I = 1$, while a value of 0 indicates a 90-degree tilt around either the x-axis or y-axis. The system remains in this passive delivery stage until J_I drops below a critical threshold, indicating that a constraint violation is feasible within the current horizon.

Once this threshold is crossed, the system transitions to stage II - Attack. The antagonistic MPC actively overrides the nominal control inputs with the optimized

malicious sequence. If the computational burden of solving the optimization problem at every step exceeds the sampling capabilities, the attacker applies the pre-calculated input sequence from the previous receding horizon solution, ensuring continuous destabilization even under computational constraints.

4.2.3.2 Simulation Setup

To validate the antagonistic framework, we use a simulation environment where the plant is modeled using the nonlinear rigid body dynamics described in Chapter 3. The simulation framework, illustrated in Figure 4.1, models a realistic commercial autopilot setup relying on a cascaded control structure. As depicted in the block diagram, the system consists of distinct modules for reference generation, position control (Outer Loop), and attitude control (Inner Loop), ensuring model separation between the fast rotational dynamics and the slower translational dynamics [100].

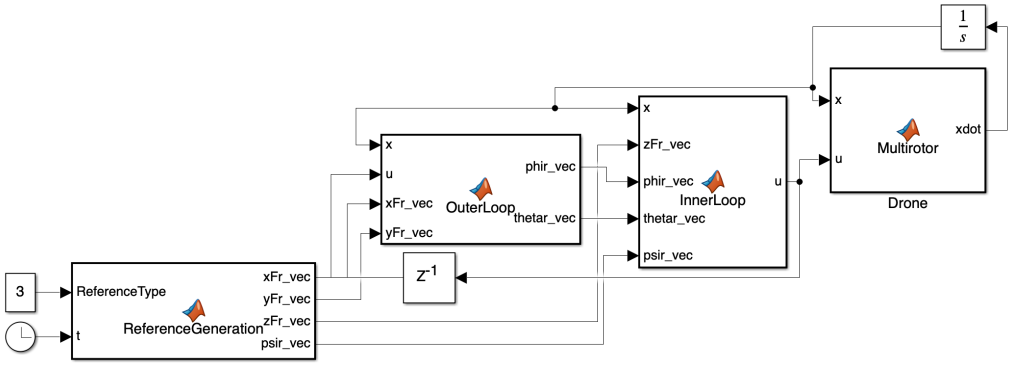


Figure 4.1: Quadrotor inner-outer control loop architecture

The nominal control used for the simulation follows the standard inner-loop/outer-loop approach shown in the figure. The *ReferenceGeneration* block produces the desired trajectory (waypoints). These references are provided to the *outerLoop*, operating at 40 Hz, which is responsible for position tracking. It calculates the necessary attitude references (roll ϕ_r and pitch θ_r) to minimize the position error (x_F, y_F) using a pole placement technique that stabilizes the translational error dynamics. Given the position references x_{F_r}, y_{F_r}, the auxiliary control inputs v_x, v_y are calculated as:

$$v_x = \ddot{x}_{F_r} - \alpha_{x1}(\dot{x}_F - \dot{x}_{F_r}) - \alpha_{x0}(x_F - x_{F_r}) \tag{4.6a}$$

$$v_y = \ddot{y}_{F_r} - \alpha_{y1}(\dot{y}_F - \dot{y}_{F_r}) - \alpha_{y0}(y_F - y_{F_r}) \tag{4.6b}$$

These auxiliary inputs are then converted into attitude references for the inner loop via inversion of the translational dynamics:

$$\begin{bmatrix} \phi_r \\ \theta_r \end{bmatrix} = \frac{m}{u_z} \begin{bmatrix} \sin\psi & -\cos\psi \\ \cos\psi & \sin\psi \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{4.7}$$

These reference angles are then passed to the *innerLoop*, which operates at a higher frequency of 400 Hz. The inner loop is designed based on the small angle approximation $(\phi \approx 0, \theta \approx 0)$ for simplicity. It computes the required virtual control inputs, the commanded thrust magnitude f_{thrust} and the control body torque vector $\boldsymbol{\tau} = [\tau_\phi, \tau_\theta, \tau_\psi]$, needed to track the altitude and attitude trajectories. This is achieved through a feedback linearization law that compensates for gyroscopic effects and nonlinearities, combined with a pole placement stabilizer for the rotational error. The control inputs are defined as:

$$f_{thrust} = \frac{m}{\cos\phi\cos\theta}\,(g + v_z) \tag{4.8a}$$

$$\tau = J\,T(\eta)^{-1}\Big(-\dot{T}(\eta)\,\omega + T(\eta)J^{-1}(\omega \times J\omega) + v_\eta\Big) \tag{4.8b}$$

where, m denotes the vehicle mass, g is the gravitational acceleration, J represents the inertia matrix, ω is the body angular velocity vector, and $T(\eta)$ is the kinematic transformation matrix relating Euler rates to body angular velocities. The terms v_z and v_η represent the linear stabilizing feedback designed to track the references z_{F_r} and $\eta_r = [\phi_r, \theta_r, \psi_r]^\top$:

$$v_z = \ddot{z}_{F_r} - \alpha_{z1}(\dot{z}_F - \dot{z}_{F_r}) - \alpha_{z0}(z_F - z_{F_r})$$

$$v_\eta = \ddot{\eta}_r - \alpha_{\eta 1}(T(\eta)\omega - \dot{\eta}_r) - \alpha_{\eta 0}(\eta - \eta_r)$$

The control gains $(\alpha_x, \alpha_y, \alpha_z, \alpha_\eta)$ used in this simulation are taken from [101], which details the tuning procedure to ensure asymptotic stability. Finally, the drone block (*Multirotor*) receives these inputs, simulating the physical response of the system. For the purpose of the antagonistic control formulation, the optimization constraints are defined directly on the operational envelope of these virtual inputs. This abstraction provides a clearer definition of the control authority available to the attacker. The physical limits of the actuators restrict the maximum torque achievable around each axis, such that $|\tau_\phi| \leq 2.5$ Nm, $|\tau_\theta| \leq 2.5$ Nm, and $|\tau_\psi| \leq 0.5$ Nm. Additionally, the vertical control limit is bounded by a maximum Thrust-to-Weight Ratio (TWR) of 2.5, ensuring the total thrust T remains within the range $0 \leq T \leq 2.5 \cdot mg$.

The antagonistic controller is designed to override the nominal inner loop commands. It targets the fast dynamics of the vehicle with a prediction horizon of $N_p = 10$ steps and a control horizon of $N_u = 4$ steps, using the linearized state-space model presented in Chapter 3 for prediction. Unlike the nominal controller which aims for stability, the adversary's objective is inverted. We define the weighting matrix Q to penalize only the attitude angles; this choice is motivated by the sensitivity of the rotational dynamics, where even slight deviations in roll (ϕ) and pitch (θ) can trigger an abrupt loss of stability and lead to complete system failure. The state vector is ordered as $\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, p, q, r]^T$, therefore, the roll ($\phi$) and pitch ($\theta$) angles correspond to the 7th and 8th components of the state vector, respectively. The state weighting matrix $Q \in \mathbb{R}^{12\times 12}$ is therefore defined as a diagonal matrix with positive scalars q_ϕ and q_θ at these indices and zeros elsewhere:

$$Q = \text{diag}([0, 0, 0, 0, 0, 0, q_\phi, q_\theta, 0, 0, 0, 0])$$

where q_ϕ and q_θ are positive scalars set to $(q_\phi = q_\theta = 1)$ for this study. The attacker ignores the cost of control effort $(R = 0_{4\times 4})$, as the goal is to use the actuators up to their maximum limits. The optimization problem maximizes the weighted sum of the squared pitch and roll angles over the prediction horizon N_p:

$$J = \sum_{i=0}^{N-1} \left(q_\phi \phi_{k+i|k}^2 + q_\theta \theta_{k+i|k}^2 \right) \tag{4.9}$$

This configuration directs all available adversarial control toward overturning the drone, ignoring yaw and position states to maximize the immediate impact on attitude stability.

4.2.3.3 Simulation Results

The efficacy of the proposed antagonistic framework is evaluated in a waypoint tracking scenario. The quadrotor is tasked with hovering and tracking a reference trajectory provided by the outer loop controller. As shown in (Figure 4.2) the outer loop position tracking confirms that the system maintains stable flight and accurately tracks the reference trajectory under nominal conditions. Throughout the simulation, the attacker is active in the system but initially operates in Stage I (Delivery), passively sniffing telemetry data without injecting malicious signals.

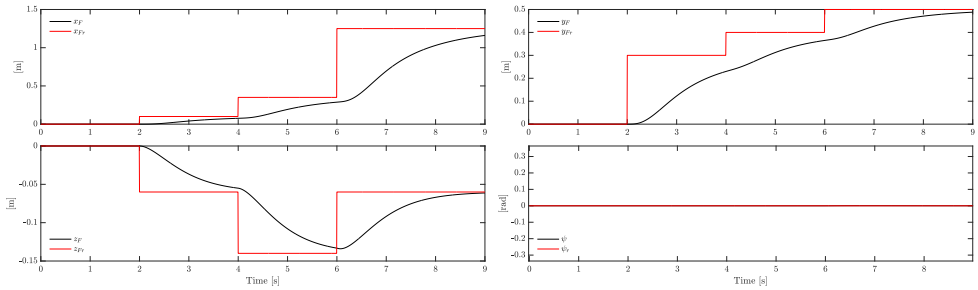


Figure 4.2: Outer loop position tracking under nominal conditions (state in black, reference in red)

The transition from passive monitoring to active attack is not arbitrary but governed by the damage performance index J_I, as defined in Equation 4.5. This index serves as a real-time metric of the system's vulnerability, quantifying how easily the drone can be destabilized given its current state. The attacker monitors this index at every time step, waiting for a specific maneuver where the drone's natural dynamics lower the energy barrier required for destabilization. The trigger threshold is configured to $J_I < \cos(55°)$, representing the point where a catastrophic tilt becomes mathematically feasible within the prediction horizon. As shown in the (Figure 4.3), this condition is met and at that instant, the antagonistic MPC computes the inputs to satisfy the attack delivery condition, specifically tilting the drone beyond the point of recovery is achievable, and the system immediately transitions to Stage II of attack.

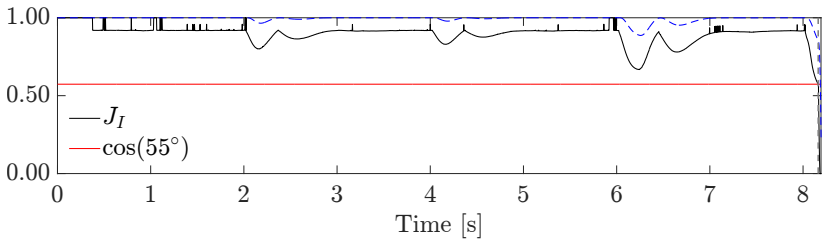


Figure 4.3: Cost function J_I

Once the attack is triggered, the antagonistic MPC overrides the nominal virtual control inputs. The impact on the system's attitude dynamics is immediate. Prior to the attack, the inner-loop states, specifically pitch and roll, closely track the references generated by the outer loop, maintaining stable flight. However, once the attack initiates, the adversary maximizes the deviation of these angles. As shown in (Figure 4.4) and detailed in (Figure 4.5), the simulation results show a rapid divergence, with the roll angle reaching 90° in approximately 0.035 seconds. This extreme attitude change rotates the thrust vector perpendicular to the gravity vector. As a result, the vertical component of the thrust f_{thrust} drops to zero, causing the altitude control to be physically impossible and the response of outer loop confirms the impact of the attack.

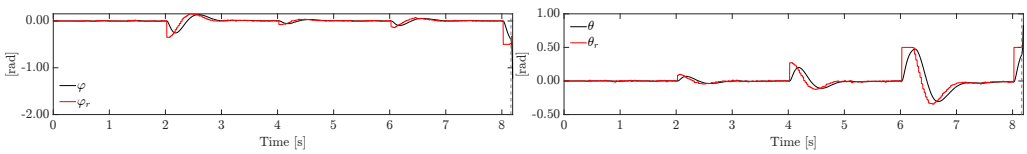


Figure 4.4: Inner loop attitude tracking under nominal conditions (state in black, reference in red)

Further, referring back to (Figure 4.2), the position deviation appears marginal in the initial time interval following the trigger, this is due to the translational inertia of the vehicle. However, the loss of vertical lift guarantees a crash, regardless of the outer loop's corrective attempts, validating the attacker's objective to cause maximum physical disruption in the shortest possible time.

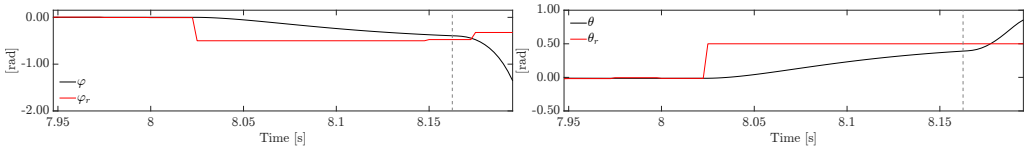


Figure 4.5: Inner loop: detail of the attack (state in black, reference in red)

The behavior of the antagonistic controller is further analyzed by observing the control inputs generated during the attack. Unlike random jamming or simple maximum-throttle attacks, the antagonistic MPC computes an optimal sequence of inputs that exploits the coupling of the system dynamics.

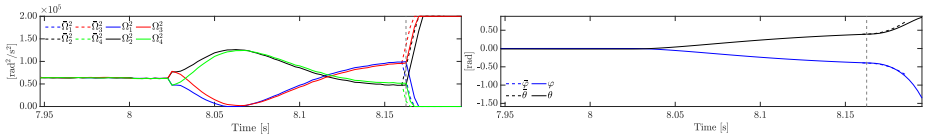


Figure 4.6: Antagonistic MPC: predicted variables (dashed line) and actual variables (solid line)

As (Figure 4.6) shows the aggressive control inputs and validates the internal model used by the adversary. Upon activation, the controller drives the body torques, particularly the roll torque τ_ϕ immediately to the saturation limit. The control profile shows the behavior of switching between extremes to maximize the angular acceleration.

4.3 Robust Antagonistic MPC

In the previous formulations, the adversarial controller was derived under the assumption of a deterministic system model. However, as discussed in Section 3.5, real-world operational environments are subject to bounded uncertainty $w(k) \in \mathbb{W}$. This uncertainty arises from sensor noise, unmodeled dynamics, and exogenous disturbances. To ensure the attack remains effective under these unfavorable conditions, the adversarial framework must be extended to account for uncertain behaviors. This section evaluates the applicability of the standard robust control methods defined in Chapter 3, specifically the naive and conservative approaches and highlights their limitations when repurposed for adversarial objectives.

4.3.1 Limitations of the Naive Approach

As detailed in Section 3.5.1, the naive approach simplifies the control problem by relying on a real-time estimate of the disturbance, $\hat{w}$. While computationally efficient, this formulation is unsafe in a safety-critical context (From attacker's point of view). In a standard control context, a disturbance is typically viewed as a force pushing the system away from a target. However, in an adversarial context where the target is a state deviation, a uncertain disturbance could act in opposition to the attacker's goal. Since the naive approach optimizes against a single estimate $\hat{w}$, it does not guarantee optimal solution, if the actual disturbance deviates from this prediction. Therefore, an attack optimized for $\hat{w}$ may fail if the environment saves the system, making this approach insufficient for guaranteeing damage.

4.3.2 Limitations of the Conservative Approach

To address the risks associated with estimation errors, the conservative approach incorporates the entire bounded uncertainty set $\mathbb{W}$ into the optimization formulation. Instead of optimizing for a single trajectory, this framework seeks a policy that holds for every possible disturbance realization $w \in \mathbb{W}$. As discussed in Chapter 3.5, this is generally categorized into two formulations; **Constraint-Tightening Method-** where

robustness is achieved via restricting the feasible search space, **Min–Max Formulation-** where, robustness is achieved via realizing the uncertainty directly into the cost function.

While the constraint-tightening method ensures safety by explicitly restricting the search space to a safe tube (invariant set), but it shows a limitation when viewed from an attacker's perspective. The framework is explicitly designed to guarantee invariance (keeping the state inside a safe set). An attacker, however, aims to guarantee violation (forcing the state outside the safe set) and must use the full capacity of the actuators to maximize damage. Therefore, standard constraint tightening cannot be simply recasted to address this, as it restricts the controller's capacity to maintain safety margins.

In comparison, the Min–Max formulation addresses disturbances by explicitly modeling their impact within the optimization problem (cost function) rather than modifying the constraints. This formulation is more suitable for adversarial goals. However, the standard defensive formulation minimizes the worst-case cost. To create an optimal attack framework, this formulation must be inverted where the adversary must maximize the cost while assuming the disturbance acts to minimize it (resisting against the attacker's goal). This necessitates a dedicated adversarial formulation. In the following section, we derive the Disturbance-Aware Antagonistic Controller by reformulating the Min-Max approach to guarantee certain level of damage in least favorable conditions.

4.3.3 Proposed Method: Disturbance-Aware Antagonistic Controller

This section presents a disturbance aware antagonistic controller derived from the robust min–max concepts discussed previously. Unlike the standard defensive robust MPC, which seeks to minimize the worst-case cost to ensure safety, our objective here is inverted where we seek to maximize the attack impact. However, simply maximizing the nominal cost function is insufficient, since in real-world operation the system is subject to uncertainty, such as uncertain external disturbances, which may counteract the malicious inputs. To ensure the attack remains effective even under these least favorable conditions from attacker's point of view, the controller is explicitly designed to be robust against such uncertainties.

To guarantee a certain level of damage despite uncertainties, we formulate the problem as a bilevel max-min optimization. This structure models the interaction as a zero-sum game between two competing agents, where the attacker selects the input sequence to maximize the system's deviation and violation while the disturbance is assumed to act as an opposing agent that may minimize the cost, representing the worst-case scenario for the attacker. The optimization problem is formally defined as:

$$J^*(x(k)) = \max_{U \in \mathcal{U}} \min_{W \in \mathcal{W}} J(U, x(k), W) \tag{4.10}$$

subject to the standard system dynamics and input constraints. Solving this nested optimization problem directly is computationally intractable due to the coupling between inputs and disturbances. To address this, the problem is simplified by isolating the effect of the disturbance from the nominal system behavior. The cost function is decomposed by adding and subtracting the disturbance-free nominal term $J(x(k), U, \mathbf{0})$. This algebraic manipulation breaks the objective function into a deterministic nominal component and a disturbance-dependent residual:

$$\max_{U} \Big(J(U, x(k), \mathbf{0}) + \min_{W} \left(J(U, x(k), W) - J(U, x(k), \mathbf{0}) \right) \Big) \tag{4.11}$$

This reformulated objective consists of two distinct components. The first is the nominal term, $(J(U, x(k), \mathbf{0}))$, which accounts for the damage caused by the attack on the ideal, noise-free system. The second component accounts for the effect of disturbances through a minimization over w. By minimizing this residual term, the formulation explicitly considers the worst-case scenario in which the disturbance most strongly counteracts and reduces the attack's impact.

To solve the inner minimization explicitly, the residual term is expanded using the compact prediction matrices (Ψ, Θ, Ξ) established in Chapter 3. The state trajectory

vector X is composed of a nominal component (dependent on the initial state and control input) and a uncertain component (dependent on the disturbance):

$$X = \Psi x(k) + \Theta U + \Xi w$$

The objective function $J(U, x(k), W)$ over the horizon is defined as a standard quadratic form:

$$J(U, x(k), W) = X^\top \mathcal{Q} X + U^\top \mathcal{R} U$$

Substituting the expanded state equation into the cost function gives:

$$J(U, x(k), W) = (\Psi x(k) + \Theta U + \Xi w)^\top \mathcal{Q}(\Psi x(k) + \Theta U + \Xi w) + U^\top \mathcal{R} U$$

To simplify the expansion, let the nominal trajectory be denoted as $X_{nom} = \Psi x(k) + \Theta U$ The cost function becomes:

$$J(U, x(k), W) = (X_{nom} + \Xi w)^\top \mathcal{Q}(X_{nom} + \Xi W) + U^\top \mathcal{R} U$$

Expanding the quadratic term gives:

$$J(U, x(k), W) = X_{nom}^\top \mathcal{Q} X_{nom} + W^\top \Xi^\top \mathcal{Q} \Xi W + 2W^\top \Xi^\top \mathcal{Q} X_{nom} + U^\top \mathcal{R} U$$

Substituting X_{nom} back into the expression gives the full expansion:

$$J(U, x(k), W) = (\Psi x(k) + \Theta U)^\top \mathcal{Q}(\Psi x(k) + \Theta U) + U^\top \mathcal{R} U + W^\top \Xi^\top \mathcal{Q} \Xi W + 2W^\top \Xi^\top \mathcal{Q}(\Psi x(k) + \Theta U)$$

The nominal cost $J(x(k), U, \mathbf{0})$ corresponds to the scenario where the disturbance is zero ($W = \mathbf{0}$). Setting $W = \mathbf{0}$ in the expression above eliminates the terms containing Ξ, leaving only:

$$J(x(k), U, \mathbf{0}) = (\Psi x(k) + \Theta U)^\top \mathcal{Q}(\Psi x(k) + \Theta U) + U^\top \mathcal{R} U$$

Subtracting the nominal cost from the full cost function cancels out the terms purely dependent on Ψ and Θ, isolating the residual effect of the uncertainty:

$$J(x(k), U, W) - J(x(k), U, \mathbf{0}) = W^\top (\Xi^\top \mathcal{Q} \Xi) W + 2W^\top \Xi^\top \mathcal{Q}(\Psi x(k) + \Theta U) \tag{4.12}$$

where the first term represents the energy of the disturbance weighted by the system dynamics, and the second term captures the linear interaction between the disturbance and the planned control trajectory.

4.3.3.1 Case Study: Robustness Validation under Uncertain Disturbance

To validate the theoretical advantages of the proposed disturbance-aware antagonistic controller, we simulated a comparative analysis against a baseline Naive attacker. While the previous sections established the attack mechanism in ideal conditions, this case study introduces stochastic environmental disturbances to evaluate the reliability of the attack when the adversary cannot perfectly predict external disturbances.

The validation is performed using a Monte Carlo approach consisting of $N_{sims} = 120$ independent simulation runs. To isolate the effect of the control method, the quadrotor parameters and initial conditions are identical to those used previously. However, specific modifications were introduced. Unlike the deterministic scenarios, random wind gusts are injected into the system dynamics through a specifically defined disturbance distribution matrix D. For this case study, D is structured as a mapping of a six-dimensional uncertainty vector directly into the system's dynamic channels, specifically, identity blocks are placed at the indices corresponding to the linear velocity and angular rate states ($D_{4:6,1:3} = I_3$ and $D_{10:12,4:6} = I_3$). This configuration models the uncertainty as physical external forces that accelerate the drone's translation and rotation. These disturbances are injected with an intensity ($w_{max} = 1.0$) to simulate turbulent conditions. To ensure a fair comparison, the same random wind sequence is applied to both the

naive and robust methods during each simulation iteration. The naive method solves the optimization assuming zero disturbance ($w = 0$), while the robust method uses the max-min formulation to anticipate the worst-case sequence of disturbance.

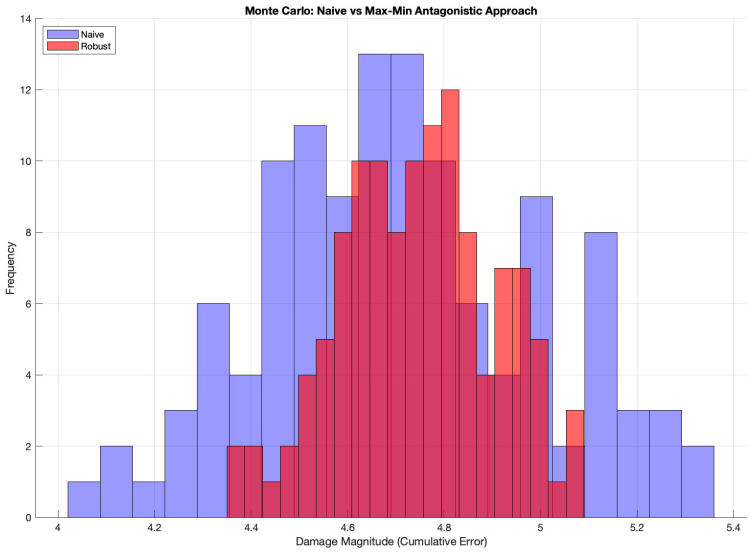


Figure 4.7: Comparative histogram of attack impact (damage magnitude) under stochastic wind disturbances ($N_{sim} = 100$ simulations). The Naive strategy (blue) shows high variance, achieving peak damage only under favorable wind conditions but failing when disturbances oppose the attack. In contrast, the Robust strategy (red) demonstrates a guaranteed performance floor, ensuring consistent system destabilization by mitigating the worst-case disturbance effects

The results of the Monte Carlo analysis are visualized in (Figure 4.7), which presents the distribution of the attack impact for both strategies. In this analysis, attack severity is quantified by the Damage Magnitude, defined as the cumulative Euclidean norm of the roll and pitch attitude deviations over the attack duration, serving as a direct metric of the induced instability.

The histogram illustrates a trade-off between the variance of the attack impact and its guaranteed severity. The naive strategy, represented by the blue distribution, shows a high degree of inconsistency. Its wide variance indicates that performance is dependent on environmental luck, the distribution reaches the highest peak damage only when random wind gusts coincidentally align with the attack vector, but it also suffers from significant failures (the left tail) where opposing winds dampen the attack's impact. In comparison, the robust strategy (red distribution) maintains a tighter clustering around the mean and, most importantly, raises the lower bound of performance. By solving the inner minimization problem, the robust controller eliminates the low-damage failures seen in the naive case, ensuring that even under the worst-case wind conditions, the damage never drops below a sufficient threshold. This confirms that while the naive approach may gives higher damage in favorable conditions, the disturbance-aware controller provides a necessary mathematical guarantee of destabilization regardless of environmental uncertainty.

4.4 Constraint-Aware Antagonistic MPC

The antagonistic control framework presented in section 4.2 shows the potential to destabilize system dynamics by maximizing the objective function. However, relying on cost maximization presents limitations when the adversary's primary goal is to force a specific operational safety violation. This formulation focuses only on the maximization of the convex cost function $J(U, x(k))$ under the assumption that maximizing state

deviation is optimal for adversarial impact, thereby neglecting the explicit targeting of state constraints.

The limitation of the previous approach is that optimality in terms of cost magnitude does not necessarily equate to optimality in terms of safety violation. A controller maximizing a quadratic cost function primarily seeks states with large deviations from the equilibrium. However, system failure often occurs at specific boundaries (e.g., maximum tilt angle or rotor saturation) that may not necessarily correspond to the regions of highest quadratic cost. If a specific safety violation occurs at a state with a lower cost magnitude, a maximization-based attacker may fail to identify it. This creates a strategic blind spot, the maximization strategy is inherently biased toward actions that gives higher cumulative numerical costs. To address this, a constraint-targeted attack strategy is required, one that moves beyond simple cost maximization to prioritize the direct violation of operational safety limits. The main results of this sections are derived from the study [102].

4.4.1 Motivation: From Maximization to Violation

To address the limitations of pure cost maximization, we reformulate the optimization problem to explicitly encourage constraint violations by introducing a nonnegative slack variable and its associated penalty. While slack variables are traditionally used in optimization to manage infeasibility by softening constraints, this framework uses them to serve the adversarial objective by quantifying and enabling the direct violation of state limits. The core of this reformulation involves introducing an auxiliary variable α and a slack variable vector $s \in \mathbb{R}^{n_x}$, recasting the optimization as the minimization problem defined as:

$$\min_{\alpha,\, s} \quad \alpha + \rho^\top s \tag{4.13a}$$

$$\text{s.t.} \quad \alpha \geq J(U, x(k)) \tag{4.13b}$$

$$x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k} \tag{4.13c}$$

$$u_{min} \leq u_{k+i|k} \leq u_{max} \tag{4.13d}$$

$$x_{min} - s_{k+i|k} \leq x_{k+i|k} \leq x_{max} + s_{k+i|k} \tag{4.13e}$$

$$s_{k+i|k} \geq 0 \tag{4.13f}$$

$$x_{0|k} = x_{init} \tag{4.13g}$$

In this formulation, the constraint $\alpha \geq J(U, x(k))$, which refers to the concave objective function, ensures that α captures the upper bound of the cost function. The reformulation lies in the modified state constraints $x_{min} - s_{k+i|k} \leq x_{k+i|k} \leq x_{max} + s_{k+i|k}$ with slack condition $s \geq 0$ and the penalty vector ρ, which function as tunable design parameters for the attack. The slack variable s represents the magnitude of the constraint violation, where setting element-wise lower bounds allows the adversary to force the optimizer toward solutions that include specific degrees of violation. Meanwhile, the penalty vector p determines the relative cost of violating a constraint, where a high penalty discourages slack usage for specific states to keep constraints tight, while lower penalties on target states allow the optimizer to selectively violate them. This configuration provides the adversary the flexibility to prioritize critical states for exploitation, a feature absent in the pure maximization framework.

To demonstrate the practical necessity of this approach, consider a one-dimensional integrator system defined by $x(k+1) = x(k) + u(k)$, subject to the constraints $x \in [-2, 3]$ and $u \in [-2, 2]$. The adversarial objective is to maximize the quadratic cost $J = (x - x_{ref})^2$ with $Q = 1$ and a reference $x_{ref} = -1$. At the start of the attack ($t = 3$s), the system is at the steady state $x = -1$.

As illustrated in (Figure 4.8): the quadratic nature of the cost function introduces a asymmetry relative to the bounds.The cost to reach the lower bound ($x_{min} = -2$) is distance-based: $J(-2) = (-2 - (-1))^2 = 1.0$, while the cost to reach the upper bound ($x_{max} = 3$) is significantly higher: $J(3) = (3 - (-1))^2 = 16.0$.

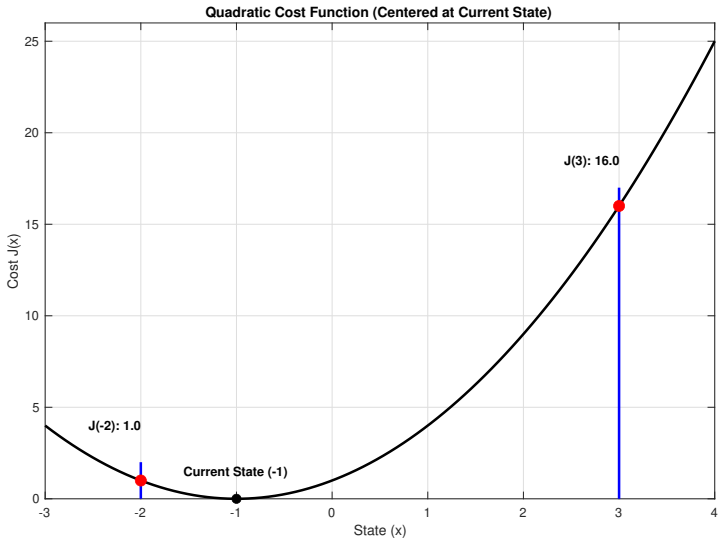


Figure 4.8: Visual representation of the quadratic cost function relative to the operational constraints. The red markers indicate the absolute cost values at the lower and upper bound, highlighting the numerical difference between the nearest bound ($x = -2$) and the farthest bound ($x = 3$) from the current state

Under a pure maximization method as shown in (Figure 4.9) the optimizer greedily chooses the maximizer that gives the highest possible cost. Recognizing that the reward at the upper bound far outweighs the reward at the lower bound, the controller drives the state upward toward $x = 3$. This behavior forces the system to go through the full state space, ignoring the physically closer boundary at $x = -2$. The strategy prioritizes cost magnitude over the violation of constraints. Moreover, the attacker spends more time and control effort to reach the distant boundary, delaying the potential for violation compared to the alternative approach.

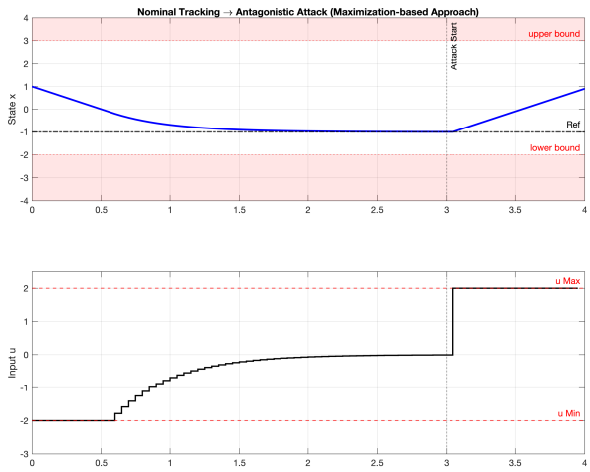


Figure 4.9: Maximization-based Method: System trajectory under the standard cost-maximization method. Following the attack initiation at $t = 3$s, the adversary targets the upper bound ($x_{max} = 3$) due to the higher potential cost, passing through the full state space rather than exploiting the closer vulnerability at the lower bound

The violation-based approach (Figure 4.10) changes this behavior by introducing slack variables s and penalizing their magnitude. By incorporating the concept of violation directly into the optimization, the solver is no longer driven solely by the quadratic cost values. Instead, it identifies the lower bound ($x = -2$) as the point to exploit.

Upon attack initiation at $t = 3$s, the control input switches to $u_{min} = -2$. This solution takes the system towards nearest boundary x_{min} and driving the state directly into the violation region (indicated by the red shading). The slack variables quantify this breach, allowing the solver to return a solution that targets the nearest system vulnerability. This result confirms that the proposed method identifies and exploits the closest constraint regardless of the nominal cost, leading to a more immediate breach of system limits compared to the maximization strategy.

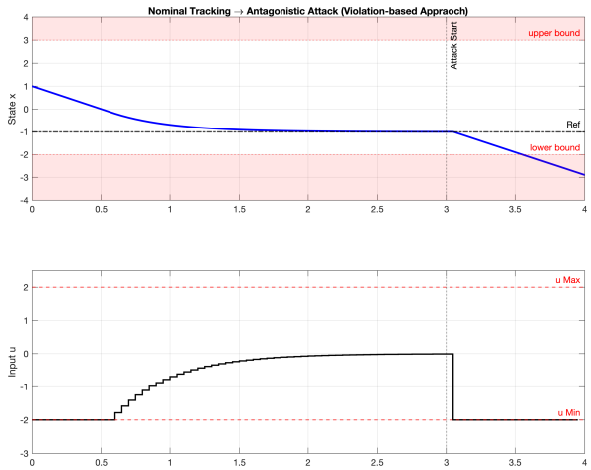


Figure 4.10: Violation-Based Method: System trajectory using the proposed formulation. The adversary identifies the nearest boundary ($x_{min} = -2$) and selects inputs ($u = -2$) to force the state trajectory into the violation region immediately after $t = 3$s

4.5 Solution Via Vertex Enumeration

To enable the practical solution of the attacker's maximization problem, we must resolve the computational intractability of the maximization. Standard gradient-based solvers cannot guarantee global optimality for maximizing a convex function, as they may get trapped in local optima (which are actually minima in convex landscapes). However, we can exploit the geometric properties of the cost function [103].

Theorem 4.5.1 (*Bauer Maximum Principle*)

Let $f : S \to \mathbb{R}$ be a convex function defined on a non-empty, compact, convex set S. Then, the global maximum of f over S is attained at an extreme point (vertex) of S. [104]. □

In the context of our antagonistic formulation, the objective function representing the system deviation and cost is convex with respect to the control input sequence U. Therefore, the optimal attack strategy that maximizes the cost is necessarily a bang-bang sequence, where every control input in the prediction horizon takes a value strictly at its upper or lower bound, rather than an intermediate value.

We solve the problem by transforming the continuous maximization over the feasible input space $\mathcal{U}^N$ into a discrete search over a finite set of vertices. We define the input vertex set $\mathcal{V}_\mathcal{U}$ as the cartesian product of the control bounds over the prediction horizon N. For a system with n_u inputs subject to box constraints $[u_{\min}, u_{\max}]$, the set of vertices consists of sequences where every element is at a boundary:

$$\mathcal{V}_\mathcal{U} \subset \{u_{\min}, u_{\max}\}^{N \cdot n_u} \tag{4.14}$$

4.5.1 Geometric Interpretation of Optimal Control Vertices

To better understand the structure of the solution space, we begin by examining the geometry of the constraints at a single time instant. The Bauer Maximum Principle implies that the gradient of the convex cost function pushes the optimal control input toward the corners of the feasible set. The (Figure 4.11) gives this concept from three standard control system architectures. In the simplest case of a one-dimensional DC motor subject to voltage limits, the convex cost curve touches the constraint boundary, confirming that the maximum lies strictly at u_{min} or u_{max}. Extending this to a two-dimensional mobile robot with linear and angular velocity constraints, the feasible set forms a rectangular polytope. The gradient of the objective function (∇J) aligns with the active constraints, locating the global maximum at a specific vertex of this geometry. Similarly, for a three-dimensional satellite system where inputs represent thruster forces, the feasible set forms a polytope (cube), and the optimal thrust vector aligns with the vertex that maximizes the projection of the cost gradient.

4.5.2 Example: Visualizing the Search Space and Complexity

We now extend this analysis to the full prediction horizon N, where the combinatorial nature of the search becomes visible. To intuitively get the structure of these optimal attack vectors and the associated computational challenge, we visualize the search space as a decision tree. Consider a simplified single-input system ($n_u = 1$) subject to normalized constraints $u \in \{-1, +1\}$. As shown in (Figure 4.12), the search space expands as a binary tree. At each time step k, the attacker must choose between the extreme boundaries (labeled -1 or +1). The total number of unique attack inputs corresponds to the leaf nodes at the bottom of the tree. For a short horizon of $N = 3$, this results in $2^3 = 8$ distinct sequences, which is computationally trivial.

However, the curse of dimensionality becomes severely apparent when the number of control inputs increases. The complexity of the method is governed by the cardinality of the input vertex set V_U, given by $|V_U| = 2^{N \cdot n_u}$, meaning the computational load doubles with every unit increase in the horizon or the number of inputs. While, (Figure 4.13)

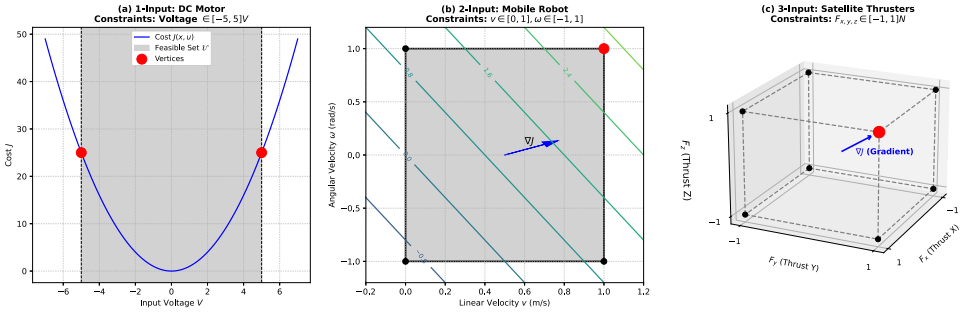


Figure 4.11: Geometric interpretation of optimal control vertices. Visual representation of the feasible sets ($\mathcal{U}$) and cost gradients. (a) 1D DC Motor: The cost curve (blue) intersects the voltage constraints at the boundary. (b) 2D Mobile Robot: The optimization gradient (∇J) is shown directing the solution to the top-right corner of the velocity rectangle. (c) 3D Satellite: The optimal thrust vector is highlighted at the vertex of the force cube

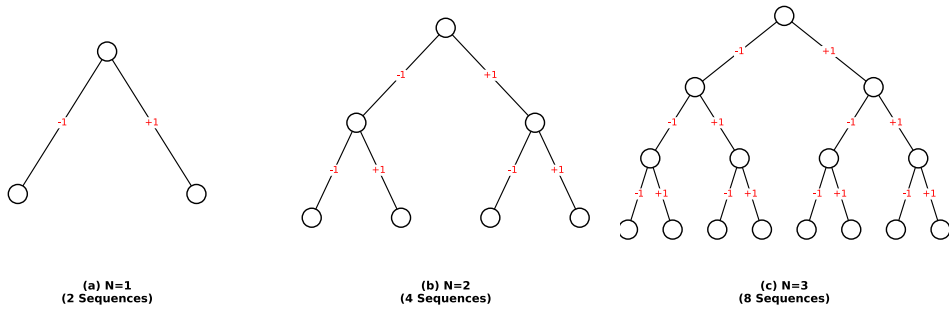


Figure 4.12: search space evolution for a single-input system ($n_u = 1$). The labels on the edges denote the specific bang-bang control action chosen at each step. The number of unique attack sequences (leaves) doubles linearly with the horizon

illustrates this impact for a two-input system ($n_u = 2$). Here, the input vector at each step has four possible combinations (i.e., $[-1, -1]$, $[+1, +1]$). As seen in panel (c) of (Figure 4.13), extending this two-input system to $N = 3$ results in an explosion to 64 distinct vertex sequences.

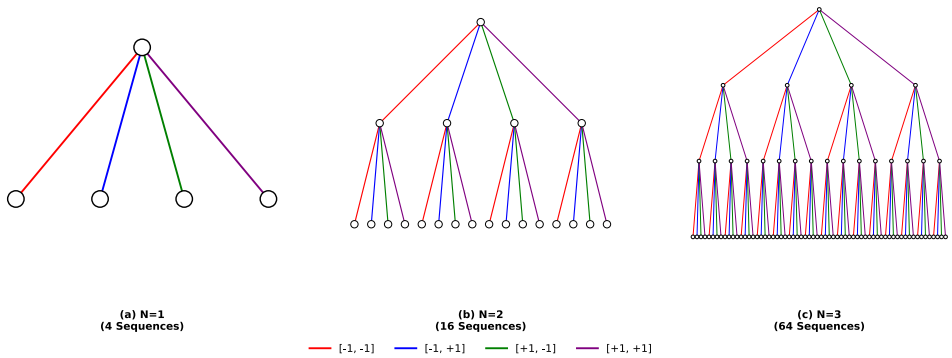


Figure 4.13: Search space evolution for multi-input system ($n_u = 2$). At each step, four distinct boundary combinations are possible (color-coded). The branching factor is quadrupled at each step, demonstrating that increasing system inputs contributes more severely to complexity than increasing the horizon

Unlike approximation methods (such as gradient ascent) which might settle for a suboptimal local maximum, this approach guarantees the identification of the absolute

worst-case attack. The exhaustive nature of the tree search ensures that the resulting safety baseline is mathematically precise.

4.5.3 Numerical Analysis

To validate the theoretical advantage of vertex enumeration over standard gradient-based methods (e.g., fmincon), we conduct a comparative simulation. This analysis uses a monte carlo method to rigorously evaluate solver performance across randomized initial conditions, ensuring that the results are not artifacts of a specific starting state.

A. Example of a Double-Integrator Model We applied the antagonistic controller to a fundamental double integrator system ($n_x = 2, n_u = 1$). The continuous dynamics are discretized with a sampling time of $T_s = 0.01$s, resulting in the following state-space matrices ($x_{k+1} = Ax_k + Bu_k$):

$$A = \begin{bmatrix} 1 & 0.01 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 5 \times 10^{-5} \\ 0.01 \end{bmatrix}$$

To show the limitations of gradient-based solvers in non-symmetric cost landscapes, we impose asymmetric control constraints:

$$u_{min} = -10, \quad u_{max} = 8$$

The simulation is setup with a prediction horizon of $N = 6$. For the vertex enumeration solver, this results in a search space of $2^6 = 64$ vertices at each time step. While computationally more intensive than a single gradient descent iteration but it guarantees convergence to the global solution under all conditions. We performed $N_{sims} = 50$ Monte Carlo simulations. In each run, the system was initialized with a random state within the bounds $x_1 \in [-0.8, 0.8]$ m and $x_2 \in [-0.2, 0.2]$ m/s. Both solvers were subjected to identical initial conditions for a direct pair-wise comparison.

The resulting state trajectories in (Figure 4.15) illustrate a clear difference. The vertex enumeration solver (green) consistently produces the same solution by selecting the control input with larger magnitude ($|u_{min}| = 10 > |u_{max}| = 8$), gives the same optimal solution across all simulation runs. In comparison, the gradient-based solver (red) produces inconsistent trajectories across many runs, indicating convergence to local optima. As a result, the solver does not guarantee convergence to the global optimum and may become trapped in suboptimal solutions.

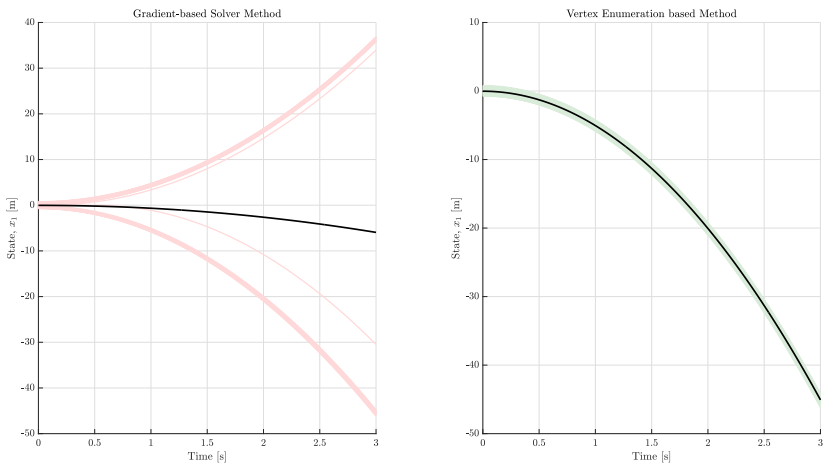


Figure 4.14: Trajectory comparison for 50 Monte Carlo simulations. The Vertex Enumeration solver (right) consistently identifies the global optimum, resulting in a uniform set of trajectories that maximize system deviation. The Gradient solver (left) shows inconsistent behavior, often converging to sub-optimal local maxima

The difference in performance is measured using the average state x_1 deviation error over time, as shown in (Figure 4.15). The vertex enumeration method gives a consistently higher mean error (green dashed line) than the gradient-based approach (red solid line). The narrow spread of trajectories shows that the vertex method gives consistent worst-case solutions. In comparison, the wider spread produced by the gradient-based solver shows that, while it can occasionally identify high-damage trajectories, convergence to the worst-case scenario is not guaranteed always.

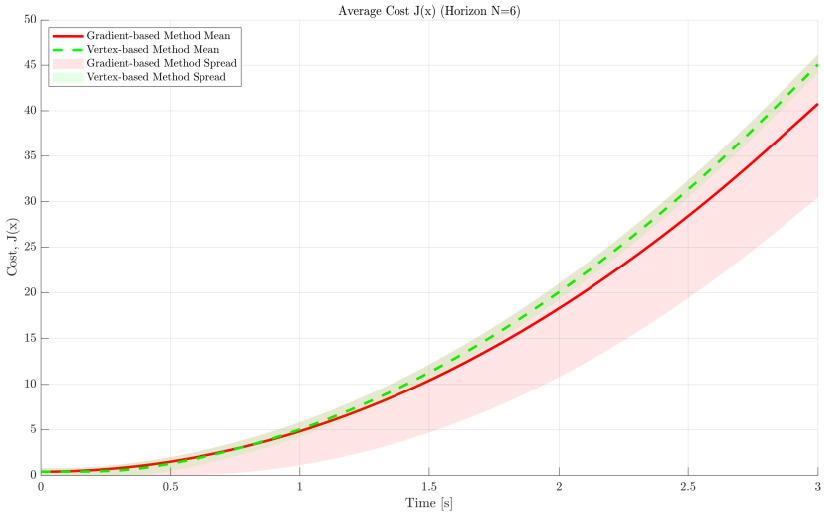


Figure 4.15: Average damage (position error) over time. The Vertex Enumeration method (green) consistently achieves higher system deviation than the gradient-based method (red), proving it is a more reliable tool for worst-case safety analysis

A. Example of a Quadrotor Model The proposed vertex enumeration method is evaluated on the quadrotor UAV framework from Section 3.2, which provides a high-dimensional test case. The system model (nonlinear rigid body dynamics), control architecture (cascaded inner-outer loop), and state definitions ($x \in \mathbb{R}^{12}$) remain identical to the setup described previously. However, while the preliminary analysis relied on a standard gradient-based solvers, this validation focuses on quantifying the performance gap between that gradient-based approach and the proposed vertex enumeration method.

For this comparative analysis, the antagonistic controller was configured to maximize the deviation of the roll angle (ϕ, state x_7). We define the control input vector $U \in \mathbb{R}^4$ as consisting of the total thrust and the three body moments:$U = [f_{thrust}, \tau_\phi, \tau_\theta, \tau_\psi]^\top$. While the thrust component must respect physical limitations ($f_{thrust} \in [0, u_{max}]$), we applied the asymmetric bounds specifically to the roll moment channel ($\tau_\phi \in [-15, 10]$). This asymmetry creates a trap for optimization algorithms: a local maximum exists at the upper bound (10), but the global maximum (worst-case damage) lies at the lower bound (-15). The simulation parameters were chosen to also challenge the computational tractability of the method. A prediction horizon of $N = 3$ was selected, which, given the quadrotor's four control inputs, generates a search space of $|V_U| = (2^4)^3 = 4,096$ vertices. At every sampling step ($T_s = 0.05$s), the Vertex solver evaluates all 4,096 sequences to guarantee the identification of the global optimum, while the gradient solver uses standard Sequential Quadratic Programming (SQP) optimization. To ensure the results are robust to state-dependency, 50 monte carlo simulations were conducted with randomized initial Roll and Pitch angles (± 0.15 rad).

The results, presented in (Figure 4.16), illustrates a divergence in solver reliability. As shown in the trajectory plots, the vertex enumeration solver manages the dimensionality of the 12-state system. It consistently identifies that the larger negative control ($|-15| > |10|$) offers the maximum destabilizing cost. Therefore, it gives a consistently same vector of trajectories (green dashed line) that drive the system to instability, with the roll angle

exceeding -45. In comparison, the gradient-based solver is less consistent, indicated by the wide red shaded region. In many iterations, the solver fails to escape the local optimum or saddle point near the equilibrium, resulting in conservative attack trajectories that underestimate the system's true vulnerability.

The difference is further measured in the left panel of (Figure 4.16) which plots the average absolute error, defined as the magnitude of the roll angle's deviation from the stable hover equilibrium ($\phi = 0$). The dashed green line represents the mean trajectory of the vertex enumeration method, showing an upper bound on the damage. In comparison, the solid red line depicts the mean performance of the gradient-based solver. The gap between these two means illustrates the estimation error introduced by using sub-optimal solvers. Furthermore, the shaded regions denote the spread of the monte carlo simulations. The narrow green shaded area confirms the deterministic nature of the vertex approach, whereas the wide red shaded area visually captures the high variance and unreliability of the gradient method. By verifying the control vertices, the proposed method shows the absolute worst-case capability of the attacker, providing a verified baseline for safety certification that standard optimization cannot guarantee.

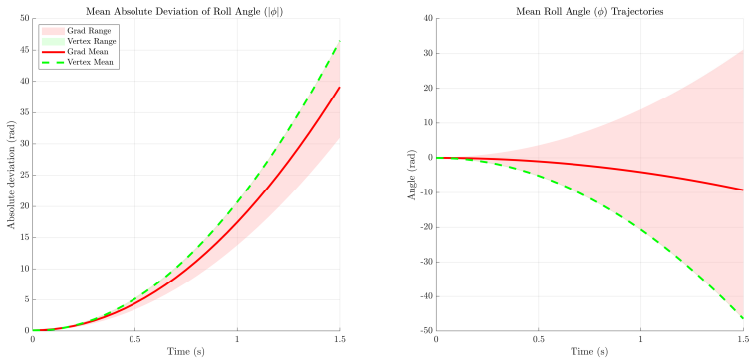


Figure 4.16: Quantitative comparison of solver performance on the Quadrotor UAV. The dashed green line and solid red line depict the mean performance of the Vertex and Gradient solvers, respectively, while the shaded regions indicate the variance across 50 Monte Carlo runs

4.6 Discussion

This chapter introduced the antagonistic MPC framework, shifting the perspective from traditional defensive regulation to an adversarial standpoint. Unlike standard resilience studies that treat threats as stochastic faults or random noise , this framework models an intelligent adversary capable of using the system's internal dynamics to cause instability. By inverting the standard control objective, the antagonistic controller computes input sequences that maximize the deviation from the reference trajectory.

Further, development of this framework shows the limitations of a naive deterministic approach when applied to realistic CPS environments subject to uncertainty. As shown in the comparative analysis, a disturbance can act in opposition to an attack and saving the system. To guarantee the attack's efficacy, the formulation was extended to a robust antagonistic MPC using a max-min approach. By modeling the interaction as a zero-sum game where nature acts as a minimizer against the adversary's maximization, this method ensures a guaranteed lower bound of damage regardless of environmental disturbance.

Moreover, this work shows that optimality in terms of cost magnitude does not equate to optimality in terms of safety violation. The study presents that a pure maximization method introduces a bias where the solver naturally targets states with the highest numerical cost (often distant boundaries), creating a blind spot regarding closer, lower-cost vulnerabilities. Therefore, an attacker focused solely on maximizing J might overlook an immediate opportunity of causing failure to the system. The proposed constraint-aware formulation reformulate the problem by introducing slack variables to penalize the distance

to the nearest violation rather than the equilibrium, forcing the optimizer to prioritize the violation of safety limits over the accumulation of quadratic cost. Finally, the solution of these antagonistic problems presents a computational challenge, as minimizing a concave function is known to be NP-hard. Standard gradient-based solvers shows suboptimal solutions, frequently converging to local optima (conservative attacks) rather than the true worst-case scenario. Therefore, based on the optimization problem, the global optimum lies at the vertices of the constraint polytope, leading to a bang-bang control method. The implemented vertex enumeration solver uses this geometry to guarantee the identification of the global worst-case attack sequence.

From a defensive perspective, this framework transitions from a theoretical threat model to a practical tool for safety assessment. By computing the optimal attack trajectory, the defender can quantify the instantaneous vulnerability through the Maximum Achievable Damage, denoted as p^*. Formally, for a current state $x(k)$, p^* is the optimal value of the attacker's maximization problem:

$$p^*(x(k)) \triangleq \max_U J(U, x(k))$$

This metric provides a real-time quantification of the worst-case system deviation. Moreover, this predictive capability allows for the derivation of the Time-to-Violation (T^*). We formally define T^* as the minimum prediction step i at which the system state leaves the safe set $\mathcal{X}_{safe}$ under the optimal attack sequence:

$$T^*(x(k)) = \min\{i \in \{1, \dots, N\} \mid x_{k+i|k} \notin \mathcal{X}_{safe}\}$$

If the system remains safe throughout the entire horizon, we define $T^* = N$. This metric provides a consistent measure of system safety and supports both offline identification of critical actuator subsystems and online vulnerability monitoring. By measuring the available safety margin, T^* enables the proactive defense strategy introduced in the next chapter, where it serves as the trigger for automated Software Rejuvenation.

Chapter 5

Condition-Based Software Rejuvenation Framework

5.1 Introduction

Software Rejuvenation (SWR) offers a reliable fail-safe mechanism for countering runtime cyber-attacks in CPS by periodically resetting compromised control components to a known, trustworthy state. As reviewed in Chapter 2, the predominant approach in the literature uses time-triggered (periodic) rejuvenation schedules. A considerable research effort has been devoted in optimizing these schedules, such as, ranging from formal safety guarantees derived via Lyapunov analysis [62], [63] to advanced strategies integrating tree-based MPC [65] and Deep Reinforcement Learning [67], aiming to manage the performance overhead resulting from resetting a running controller.

However, the underlying concept of these approaches remains proactive and static, where rejuvenation is executed based on conservative approach that assumes worst-case scenario, irrespective of the actual presence or absence of an attack. This design choice is driven by the defensive assumption that sophisticated adversaries can engineer stealthy attacks capable of evading conventional intrusion detection until physical damage is irreversible. While this strategy guarantees a bounded time-to-recovery, it introduces operational penalties on modern autonomous systems that function in highly dynamic, resource-constrained, and mission-critical environments.

5.1.1 Operational Limitations of Periodic SWR (The Always-On Cost)

The fixed worst-case scheduling of periodic SWR results in an always-on overhead. Since the system cannot reliably distinguish benign operation from compromise, it must repeatedly interrupt the control loop, degrading both availability and performance even under nominal conditions. As a result, these interruptions affect the system in three primary aspects:

1. **Availability** The system spends a significant portion of its operating time in a rejuvenation and recovery state. Quantitative measurements from the literature [15] shows the severity of this limitation, with availability dropping to approximately 64.3% under realistic periodic policies.
2. **Mission Latency** Control interruptions cause delays in trajectory execution and experiments on real, nonlinear quadrotor systems validates that the trajectory completion times can increase from a baseline of 86.63 s (no rejuvenation) to 484.25 s when periodic resets are enabled [14], [15].
3. **Transient Control Instability** For dynamical systems, a controller reset is not just a pause and it results in a loss of internal state history, including integral windup terms, estimator covariance matrices, and MPC warm-start solutions. Frequent resets erase the controller's internal memory, causing transient jerk and tracking errors immediately post-rejuvenation. In extreme cases, when tight schedules are required to counter fast-acting attacks, this repeated loss of control authority can cause the system instability or incapable of making forward progress.

This introduces a well-established trade-off: schedules designed to maximize security via high-frequency resets inevitably compromise both system availability and stability [73]. In fast-dynamic systems such as the quadrotor model described in Chapter 3 where control updates occur at rates exceeding 100 Hz, this trade-off is significant, often making periodic rejuvenation impractical for deployment in real-world scenarios.

5.1.2 The Shift to Reactive and Proactive Triggering

The limitations of periodic SWR arise from handling CPS threats similar to traditional IT attacks, where detection surfaces are largely digital and exploitable for stealth. However, this analogy overlooks the fundamental attribute of CPS: Physicality.

In CPS environments, adversaries can only achieve effects by influencing the physical system. Any tampering with sensors or actuators produces observable deviations in the system state, which evolves according to established dynamical models and is bounded by mechanical inertia. These predictable physical behaviors provide a model-driven basis for anomaly detection. This chapter extends the antagonistic MPC framework proposed in Chapter 4 by repurposing the formalization of the attacker's optimal strategy for defensive use. We introduce a Condition-Based Software Rejuvenation (CB-SWR) framework that shifts away from clock-based triggers to evidence-based decisions. This methodology uses a dual-monitor framework to detect range of attacks, consisting of two complementary components:

- **Monitor I (Luenberger-like Observer-based Monitor)** A residual-based detector that identifies ongoing anomalies where the observed physical response is inconsistent with the expected dynamic model.
- **Monitor II (Antagonistic MPC-based Predictor)** A safety estimator that calculates the Time-to-Violation (T^*), identifying scenarios where the system is currently stable but is drifting toward safety margins.

The integration of these monitors ensures that rejuvenation occurs only upon the detection of an attack or when safety margins drop below a safety threshold. This logic eliminates unnecessary resets during safe operation while maintaining formal safety bound.

5.1.3 Chapter Contributions and Organization

This chapter formally introduces the condition-based SWR framework, a defense strategy designed to balance operational availability with safety. The core contribution is a dual-monitor architecture that combines a reactive observer and a predictive monitor. By using these monitors to trigger rejuvenation only when a specific threat is verified, the framework eliminates the unnecessary downtime associated with traditional periodic schedules. To show its flexibility, the method is validated on two platforms: a highly dynamical model (Quadrotor System) and a slower, stable Four-Tank process control system. The main results of this chapter have also been presented in a manuscript currently under review for the IFAC conference [105].

The chapter is organized to progress from theoretical design to numerical validation. Sections 5.2 through 5.5 define the system architecture, the mathematical derivation of the monitors, and the decision logic. Section 5.6 details the first case study on the Quadrotor UAV, focusing on protection against attitude instability. Section 5.7 extends this validation to the Four-Tank System, testing the framework against hydraulic overflow threats in a process control environment. Finally, Section 5.8 summarizes the experimental findings.

5.2 Architecture of the CB-SWR Framework

This section details the proposed dual-monitor framework, which serves as the core methodology for triggering SWR. The architecture, illustrated in (Figure 5.1), formalizes the parallel monitoring loop that operates alongside the standard closed-loop control system. The framework is structurally divided into two primary domains, the operational domain (the physical plant and its immediate controllers) and the monitoring domain (the diagnostic tools). This separation allows for a modular defense strategy where the monitoring logic can be deployed flexibly either embedded onboard for autonomous operation or hosted on an edge computer for computationally intensive tasks without changing the fundamental control structure.

The operational domain, depicted on the right-hand side of (Figure 5.1), shows the physical system and its immediate control execution logic. At the core is the plant, representing the physical system (e.g., a Quadrotor or Tank system) which evolves according to non-linear dynamics. Operating within this environment is an attacker, defined in the threat model of Chapter 4 as an intelligent adversary capable of injecting malicious inputs into the control, with the specific aim of destabilizing the system or violating safety constraints. To maintain operational integrity against such threats, the framework uses a module of trusted components acting as a secure gateway for control execution. This module contains the nominal controller, which is responsible for standard operation, alongside a safety controller, a simplified and trusted backup unit activated during the recovery phase. The switching between these control modes is managed by logic decisor 2, a local execution mechanism that receives the binary rejuvenation signal ($SWR \in \{0, 1\}$) from the monitoring domain and physically switches the system input between the nominal and safety controllers to execute the recovery.

The characteristic of this proposed CB-SWR framework is its architectural flexibility, allowing for implementation suited to specific mission requirements rather than being constrained by computational limitations. The framework supports both configurations, such as, a remote/edge-based setup and a fully autonomous onboard integration. In a remote configuration, the monitoring components are hosted off-board, where they receive state telemetry $y(t)$ via a wireless link, compute safety metrics, and transmit the binary rejuvenation command (SWR) back to the onboard controller. This architecture is particularly relevant for centralized command-and-control scenarios where ground stations supervise multiple agents or where telemetry logging is required for post-mission analysis.

In comparison, the framework is equally capable of fully autonomous, onboard implementation, which ensures operational self-reliance even in communication-denied environments. While predictive monitoring schemes dependent on optimization-based methods are computationally intensive for embedded processors, this research overcomes this limitation. As detailed later in Section, the proposed method reformulates the optimization problem as a Multi-Parametric Quadratic Program (mp-QP). By applying standard multi-parametric techniques, the explicit form of the control law is computed offline, transforming the complex online calculation into a solution that functions similarly to a look-up table. This algorithmic reduction allows the computationally expensive predictive monitor to be executed directly on standard onboard hardware with minimal processing overhead. As a result, the choice between offboard and onboard deployment is determined solely by the operational context (e.g., operations in communication-denied environments, bandwidth constraints, or the need for centralized supervision) rather than the computational capacity of the autonomous platform.

5.3 Monitor I: Observer-Based Monitor

The first layer of the defense framework is designed to detect reactive, high-magnitude anomalies where the system's behavior diverges from its mathematical model. This component uses a state observer functioning as a residual generator.

Let the nominal, discretized closed-loop dynamics of the system be denoted by the function $f_{cl}(\cdot)$. This function represents the one-step evolution of the system state under the nominal controller $u_{nom}(k) = \kappa(x(k))$, such that in an attack-free scenario formulation

$$x(k+1) = f(x(k), \kappa(x(k))) \triangleq f_{cl}(x(k))$$

We design a Luenberger-like observer to generate a state estimate $\hat{x}(k)$ running in parallel to the physical system. The observer dynamics are given by:

$$\hat{x}(k+1) = f_{cl}(\hat{x}(k)) + L(y(k) - \hat{y}(k)) \tag{5.1}$$

where $\hat{x}(k) \in \mathbb{R}^n$ is the estimated state vector, $L \in \mathbb{R}^{n \times n}$ is the observer gain matrix, and assuming full state feedback, the estimated output is $\hat{y}(k) = \hat{x}(k)$.

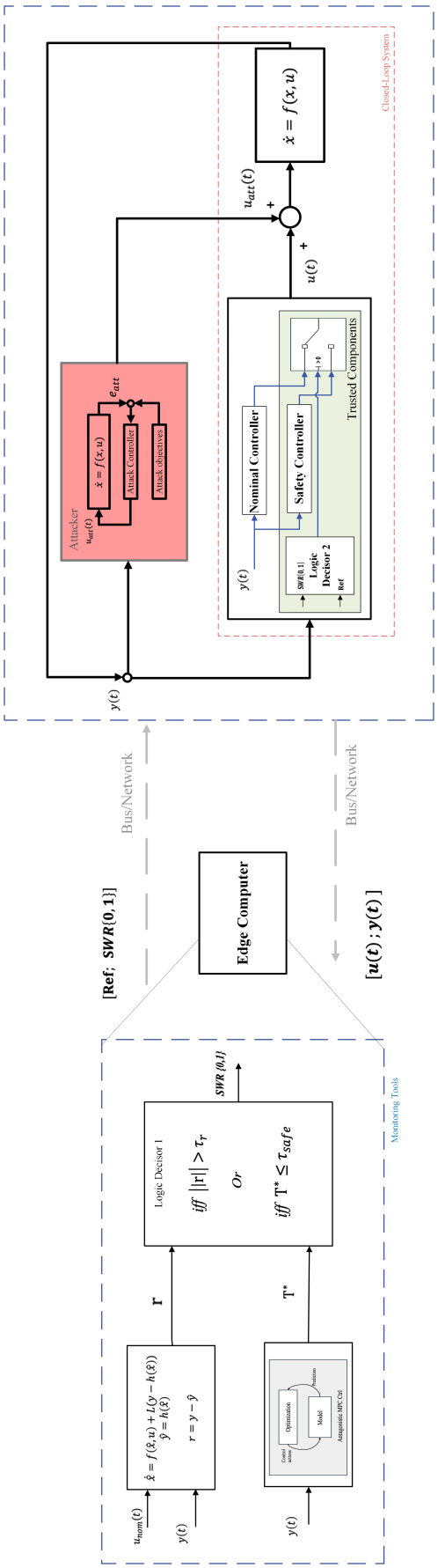


Figure 5.1: Schematic overview of the Condition-Based SWR framework. The architecture uses a dual-monitor architecture to fuse observer-based residuals and predictive safety estimates

The diagnostic signal, residual is defined as the difference between the measured physical state and the observer's estimate:

$$r(k) \triangleq x(k) - \hat{x}(k) \tag{5.2}$$

To analyze the convergence properties of this residual, we derive its time evolution by substituting the system dynamics and the attack model into Equation 5.2. The dynamics of the residual $r(k+1)$ are governed by:

$$r(k+1) = x(k+1) - \hat{x}(k+1)$$

Substituting the attack-injected system dynamics $x(k+1) = f(x(k), u(k))$ and the observer dynamics, we obtain:

$$r(k+1) = f_{cl}(x(k)) - f_{cl}(\hat{x}(k)) - Lr(k) + w_a(k)[f(x(k), u_{att}(k)) - f_{cl}(x(k)) \tag{5.3}$$

The behavior of the residual is determined by the presence of the attack indicator $w_a(k) \in \{0, 1\}$.

Case 1: Nominal Operation ($w_a(k) = 0$) In the absence of attacks, the second term in Equation 5.3 vanishes. The residual dynamics reduce to the autonomous error system:

$$r(k+1) = f_{cl}(x(k)) - f_{cl}(\hat{x}(k)) - Lr(k)$$

For small estimation errors, using a first-order Taylor series approximation $f_{cl}(x) - f_{cl}(\hat{x}) \approx A_{cl}(x - \hat{x})$, where $A_{cl} = \frac{\partial f_{cl}}{\partial x}$, the dynamics can be linearized as:

$$r(k+1) \approx (A_{cl} - L)r(k)$$

Proposition 5.3.1 (*Observer Stability*)

If the observer gain matrix L is selected such that the eigenvalues of $(A_{cl} - L)$ lie strictly inside the unit circle, the error dynamics are asymptotically stable. Consequently, in the absence of attacks and noise,

$$\lim_{k \to \infty} \|r(k)\| = 0$$

□

This proposition ensures that under normal operating conditions, any initial discrepancy between the observer and the plant decays over time, preventing false positives due to initialization errors.

Case 2: Attack Scenario ($w_a(k) = 1$) When an attack is active, the adversary overrides the nominal control input with $u_{att}(k)$ or injects false data. The expression enclosed in the square brackets in Equation 5.3 represents the mismatch between the physical system's evolution under attack and the observer's expectation of nominal behavior:

$$\Delta_{att}(k) = f(x(k), u_{att}(k)) - f_{cl}(x(k))$$

This term acts as a non-vanishing external disturbance on the error dynamics. Even if the observer is stable, this forcing function drives the residual $r(k)$ away from the origin.

Based on the dynamics derived above, the detection logic relies on the magnitude of the residual vector. Since real-world systems are subject to measurement noise and model uncertainties, the residual will not be identically zero even in safe conditions. Therefore, a static threshold τ_r is introduced to distinguish between nominal noise and

active attacks.The monitor flags an anomaly and triggers the decision logic if the residual violates this bound:

$$\text{Trigger}(k) = \begin{cases} 1 & \text{if } \|r(k)\| > \tau_r \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

where $\|\cdot\|$ denotes the Euclidean norm. The threshold τ_r is a tunable parameter selected to balance the trade-off between sensitivity (detecting small attacks) and robustness (avoiding false alarms due to sensor noise). This reactive mechanism is effective against dumb or brute-force attacks that cause immediate, large-scale deviations in the system trajectory. The selection of τ_r is a statistical trade-off. It is generally tuned to strictly upper-bound the residual induced by sensor noise and model uncertainties during nominal operation.

5.4 Monitor II: Antagonistic MPC-based Predictor

While the observer-based monitor is effective at detecting existing discrepancies caused by attacks, it is inherently reactive. To ensure comprehensive safety, the defense framework must also be capable of anticipating future risks before physical limits are violated.

The second component of the framework, monitor II, addresses this by proactively assessing the system's vulnerability in real-time. It computes the Time-to-Violation (T^*), a novel safety metric that quantifies the minimum time an intelligent adversary would require to drive the system from its current state $x(k)$ into an unsafe region $\mathcal{X}_{unsafe}$, assuming they have full control access and perfect model knowledge. This metric represents a worst-case countdown, providing the decision logic with a precise window of opportunity to initiate rejuvenation before recovery becomes physically impossible.

To estimate T^*, we repurpose the antagonistic MPC framework originally developed in Chapter 4 as an attack strategy. In that context, the framework was used to find optimal inputs to maximize system damage. Here, we invert the perspective and use the same mathematical formulation from a defensive standpoint to simulate the worst-case evolution of the system. To ensure online computational feasibility, the nonlinear dynamics are approximated by the LTI model derived (via linearization and discretization):

$$x_{i+1|k} = Ax_{i|k} + Bu_{i|k} \tag{5.5}$$

where $x_{i|k}$ denotes the predicted state at step i given information at time k, and (A, B) are the discrete-time system matrices. We define the unsafe region $\mathcal{X}_{unsafe}$ as the set of states where state variables (e.g., roll or pitch angles) exceed their physical safety limits:

$$\mathcal{X}_{unsafe} = \{x \in \mathbb{R}^{n_x} \mid |x_i| > X_{limit,\ i}\} \tag{5.6}$$

where x_i is the i-th component of the state vector and $X_{limit,\ i}$ represents the corresponding upper/lower safety constraint.

This monitor is based on solving a sequence of finite-horizon optimization problems. For a given prediction horizon N, we seek to compute the worst-case trajectory is determined by solving the constraint-aware antagonistic MPC problem previously defined in Equation 4.13.

Remark 5.4.1 (*Vertex Enumeration Solution*)

As discussed in Chapter 4, maximizing a convex function (such as the quadratic cost) over a convex polytope is a non-convex optimization problem, for which standard gradient-based methods are generally sub-optimal. However, since the optimal solution is attained at a vertex of the feasible set, we use the vertex enumeration method. This approach searches over the finite set of input sequences generated by the vertices of the control set $\mathcal{U}$, thereby guaranteeing attainment of the global optimum. □

The monitor iteratively solves formulation defined in Equation 4.13 for increasing horizons $N = 1, \ldots, T_{max}$. The Time-to-Violation is formally defined as the smallest

horizon N where the computed worst-case trajectory $\{x^*_{j|k}\}_{j=1}^{N}$ intersects the unsafe region:

$$T^*(x(k)) = \min\{N \in [1, T_{max}] \mid \exists j \leq N,\ x^*_{j|k} \in \mathcal{X}_{unsafe}\} \tag{5.7}$$

If no violation is found within the maximum look-ahead horizon T_{max}, the system is considered safe for the foreseeable future, and we set $T^* = T_{max}$.

The computed T^* value is fed into the decision logic to enable preventive action. Unlike the observer-based monitor which triggers on past anomalies, this monitor triggers based on future vulnerability. The rejuvenation sequence is initiated if the estimated T^* falls below a critical safety margin τ_{safe}:

$$\text{Trigger}(k) = \begin{cases} 1 & \text{if } T^*(x(k)) \leq \tau_{safe} \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

This logic ensures that the system resets itself while it is still controllable, effectively mitigating the attack before the adversary can drive the system into the unsafe region.

5.5 Logic Decision Module

The final component of the monitoring domain is the logic decision module. Its primary function is to combine the results from the observer-based (Monitor I) and the antagonistic-based (Monitor II) to generate a single binary signal that controls the software rejuvenation sequence.

To prioritize system safety, the module primarily operates on a fail-safe OR logic. This configuration ensures that the defense proposal is activated if either monitor detects an abnormality, minimizing the risk of a missed attack. However, the framework is designed to be flexible to different operational requirements. In scenarios where system availability is top priority or where high measurement noise might lead to frequent false alarms, the decision rule can be switched to an AND logic. This configuration requires consensus between both monitors (i.e., both the residual and the predictive safety metric must be violated) before triggering a reset, thereby favoring operational continuity and robustness against false positives over maximum sensitivity.

Let $SWR(k) \in \{0, 1\}$ denote the binary trigger signal at time step k, where 1 initiates the rejuvenation process. The decision logic is formally defined as:

$$SWR(k) = \begin{cases} 1 & \text{if } (T^*(x(k)) \leq \tau_{safe}) \circledast (\|r(k)\| \geq \tau_r) \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

where $\circledast$ represents the logical operator ($\vee$ for safety-critical OR logic, $\wedge$ for availability-critical AND logic). In the context of this work, we prioritize the safety-critical formulation ($\vee$), where τ_{safe} represents the critical safety time margin and τ_r is the residual threshold.

This integration creates a robust protection scheme that addresses different types of threats by using the strengths of both monitors:

- High-magnitude anomalies that cause immediate, large deviations in the system state are instantly identified by the observer-based monitor ($\|r(k)\| \geq \tau_r$). This ensures a rapid response to brute-force attacks without waiting for predictive calculations.
- While, subtle, slow-varying deviations that are designed to stay within the residual noise floor are captured by the predictive monitor. While these attacks may not trigger the residual threshold immediately, they gradually degrade the safety margin until the predicted Time-to-Violation drops below the critical limit ($T^*(x(k)) \leq \tau_{safe}$).

By relying on both indicators, this structure prevents missed detections while avoiding unnecessary interruptions during standard operation.

5.6 Case Study: Quadrotor Unmanned Aerial Vehicle (UAS)

To validate the operation of the proposed condition-based software rejuvenation framework, we apply the methodology to a quadrotor UAS. The quadrotor represents a canonical benchmark for safety-critical CPS due to its fast, unstable dynamics and high sensitivity to actuator deviations.

5.6.1 System Overview and Modeling

The mathematical model of the quadrotor follows the rigid body dynamics with six degrees of freedom derived in detail in Chapter 3. For this case study, we uses the full 12-state vector $x \in \mathbb{R}^{12}$:

$$x = [p;\ v;\ \eta;\ \omega] \tag{5.10}$$

where p is the position, v is the linear velocity, $\eta = [\phi, \theta, \psi]^\top$ represents the Euler angles (roll, pitch, yaw), and ω is the angular velocity in the body frame.

While the physical plant's simulation uses the full nonlinear model, the predictive monitor (Monitor II) uses the linearized, discrete-time LTI approximation ($x_{k+1} = Ax_k + Bu_k$) to ensure prediction based optimization problem remains computationally feasible in real-time.

5.6.2 Implementation of Observer-based Monitor (Rotational Observer)

For the reactive detection layer, instead of estimating the full state vector, the observer is specifically designed to target the quadrotor's rotational subsystem. This design choice is motivated by the fact that the vehicle's stability is dependent on attitude control where deviations in roll (ϕ) or pitch (θ) directly influence the translational dynamics, leading to catastrophic failure.

We construct a residual generator for the angular momentum vector $J\omega$. Applying the observer principle, the dynamics of the observer state $z \in \mathbb{R}^3$ are given by:

$$\dot{z} = -k_r\omega - \omega \times J\omega + F_2 u_{\text{nom}} + H\big(J\omega - z\big) \tag{5.11a}$$

$$r = J\omega - z \tag{5.11b}$$

where u_{nom} is the nominal control input computed by the flight controller, $H \in \mathbb{R}^{3\times3}$ is a Hurwitz gain matrix designed to ensure error convergence, and F_2 is the control allocation matrix mapping motor speeds to body-frame moments. The residual vector $r = [r_\phi, r_\theta, r_\psi]^\top$ represents the instantaneous mismatch between the expected and actual torque acting on the airframe.

An important aspect of the monitor's implementation is the selection of the threshold τ_r to minimize false positives caused by sensor noise and model uncertainties (e.g., aerodynamic drag approximations). We determine these thresholds experimentally using a statistical approach. Monte carlo simulations were conducted under nominal, attack-free conditions to estimate the baseline noise profile of the residual. Following the 3-Sigma (3σ) rule, the threshold for each axis $i \in \{\phi, \theta, \psi\}$ is set as:

$$\tau_i = |\mu_i| + 3\sigma_i$$

where μ_i and σ_i are the mean and standard deviation of the residual signal under nominal operation. This establishes a reliable statistical boundary that captures 99.73% of nominal behavior, ensuring that alarms are triggered only by statistically significant deviations signals an attack.

Remark 5.6.1 (*Stealthy Attack Subspace*)

It is important to note that the observer monitors torque mismatches. Therefore, an attack input u_{att} remains stealthy *to this monitor if it lies in the kernel of the moment*

mapping matrix, i.e.,

$$u_{\text{att}} \in \ker(F_2)$$

Physically, this corresponds to an attack that scales the thrust of all four rotors equally, thereby changing the total lift without inducing any rotational moment. □

5.6.3 Implementation of Predictive Monitor (Attitude Predictor)

The predictive monitor is specifically configured to counter adversaries who attempt to evade the observer-based monitor. While they flags high-magnitude violations, an intelligent attacker may purposely constrain their inputs to remain within the residual noise threshold ($\|r\| < \tau_r$), thereby bypassing the first line of defense.

However, even with these constrained inputs, the adversary's optimal strategy is to target the system's most critical states, such as the fast rotational dynamics. By injecting small but optimally directed deviations into the roll (ϕ) and pitch (θ) channels, the adversary drives the system states toward their physical constraints. The objective is to maximize the accumulated deviation at the boundary, ensuring that when the physical limit is violated, the failure is sudden and catastrophic, leaving the system with insufficient control authority to recover. Therefore, monitor-II counter this, by solving the maximization problem to show these violations before they become irreversible ($T^* < \tau_{safe}$).

To compute the Time-to-Violation (T^*), we formulate the maximization problem to specifically target the attitude states. The cost function J is weighted to penalize deviations in roll and pitch:

$$J(U, x(k)) = x_k^\top Q x_k = q_\phi \phi_k^2 + q_\theta \theta_k^2$$

where $Q \succeq 0$ is a diagonal weighting matrix with non-zero entries only for the roll and pitch states. The unsafe region $\mathcal{X}_{unsafe}$ is defined by the physical flight envelope limits:

$$\mathcal{X}_{unsafe} = \{x \in \mathbb{R}^{12} \mid |\phi| \geq \phi_{lim} \vee |\theta| \geq \theta_{lim}\}$$

To ensure the antagonistic MPC-based predictor operates within real-time constraints, we avoid solving the optimization problem online at every time step, instead, we use an Explicit MPC formulation. Theoretically, as discussed in 3.6, the explicit solution to a constrained linear MPC problem takes the form of a Piecewise Affine (PWA) control law defined over polyhedral regions. However, practically, we implement a numerical approximation of the explicit solution via state-space discretizations.

Instead of storing the PWA regions directly, the state space (defined by roll ϕ and pitch θ) is discretized into a high-resolution grid. For every discrete point $x(k)$ in this grid, the MPC optimization problem is solved offline. This effectively samples the optimal explicit control law $u^*(x)$ and the resulting Time-to-Violation (T^*) across the entire operational domain. This process encodes the continuous MPC dynamics into a static safety lookup table. This allows the online controller to approximate the explicit MPC solution through fast bilinear interpolation, avoiding the computational overhead of online optimization.

5.6.4 Simulation Setup and Results

The proposed framework is validated using a simulation environment developed in MATLAB & Simulink. This section details the operational parameters, the implementation of the rejuvenation logic, the specific attack profiles designed to evaluate the monitoring system, and the formulation of the periodic benchmark used for performance comparison.

To ensure precise synchronization between the physical plant dynamics and the defensive logic, the simulation is configured with a unified sampling architecture. The non-linear quadrotor plant, the flight control stack (Inner/Outer loops), and the proposed

dual-monitor framework are all executed at a sampling step of $T_s = 10^{-3}s$. This high-frequency operation ensures that the predictive monitor captures fast transient behaviors and that the Time-to-Violation metric (T^*) relies on an internal discrete model perfectly matched to the execution rate.

The mission profile assigns the quadrotor a reference trajectory consisting of a 5 s vertical ascent to 10 m, followed by execution of a 30 m × 30 m square path at constant altitude. The system's operational envelope is constrained by physical actuator saturation and safety limits. The maximum available torque is limited to $\tau_{\phi,\theta}^{max} = 2.5$ Nm and $\tau_{\psi}^{max} = 0.5$ Nm, with a maximum TWR of 2.5 and attitude constraints defined as $\mathcal{X}_{unsafe} = \{x \in \mathbb{R}^{12} \mid |\phi| \geq 90° \vee |\theta| \geq 90°\}$.

Software rejuvenation is modeled as a state machine that transitions the system state between mission, rejuvenation, and recovery modes. An important parameter in this framework is the downtime duration, denoted as T_{down}, which represents the interval where the nominal controller is offline. This parameter is dependent on the hardware architecture and the specific restoration method used. As summarized in (Table 5.1), this duration can vary, ranging from standard OS reboots (≈ 1.5 s) to optimized method such as checkpoint restoration or micro-reboots (≈ 30 ms). Recent advancements have shown that these lower downtime values are practically feasible even for highly dynamic systems like quadrotors. To maintain consistency with these state-of-the-art fast recovery methods, this simulation considers a downtime of $T_{SWR} = 50$ ms. During this interval, the actuators simulate a temporary loss of control by holding zero thrust (free fall). Upon the completion of SWR, the system enters the recovery mode. In this phase, the system isolates the outer loop and engages a safety controller mode. This mode use the inner loop to stabilize the attitude dynamics around a hover equilibrium (i.e., $\phi_{ref} = 0, \theta_{ref} = 0$). This ensures the drone arrests any rotational momentum caused during the downtime or attack. Control is handed back to the fully cascaded nominal controller only once the attitude error converges within a pre-defined safety region, ensuring a smooth resumption of the mission.

Table 5.1: SWR Downtime Values Reported in Literature

Downtime Value	System / Controller	Type of Downtime	Source Article
3 ms	PX4 Flight Controller	Optimized snapshot restore	[106]
10–50 ms	PX4 Quadrotor (SITL)	Rollback variation	[72]
∼15 ms	PX4 Autopilot (SITL)	Rollback	[12]
∼20 ms	PX4 Autopilot (SITL)	Checkpoint	[14]
30 ms	PX4 / JMAVSim	Micro-reboot	[107]
100 ms	Quadrotor (Simulation)	Software Refresh	[62]
390 ms	3DOF Helicopter (ZedBoard)	Full Reboot Time	[15]
1.5 s	PX4 Flight Controller	Simple / Full Reboot	[106]

To evaluate the dual-monitor detection performance, the system is subjected to two different classes of attacks targeting the pitch/roll dynamics.

Attack scenario 1- It represents a worst-case, optimized adversary that solves an antagonistic optimization problem online to compute a control input maximizing the deviation of the pitch or roll angle from equilibrium:

$$J(U, x(k)) = \sum_{i=0}^{N-1} \left[q_\phi \phi_{k+i|k}^2 + q_\theta \theta_{k+i|k}^2 \right]$$

where ($q_\phi = q\theta = 1$). This formulation gives abrupt, high-magnitude control inputs attempting to destabilize the system in the minimum possible time. It validate the responsiveness and sensitivity of the observer-based monitor.

Attack Scenario 2- It represents a model-free attack designed to evade residual-based detection. The adversary injects a signal into the pitch/roll moment channel that grows linearly over time, modeled as a ramp function:

$$u(t) = u_{nom}(t) + \alpha(t - t_{start})$$

the slope α is selected to be sufficiently small such that the resulting residual $r(k)$ remains below the detection threshold τ_r for an extended period. This effectively masks the attack from the observer until the accumulated error forces the system near its physical limits, validating the necessity of the predictive monitor.

To benchmark the performance of the proposed condition-based SWR method, we compare it against the restart-based method proposed by [15]. Unlike fixed-interval approaches, this baseline uses real-time reachability analysis to dynamically compute the maximum safe operational window (λ) at each time step. This method guarantees that the system remains within the recoverable region $\mathcal{R}$ despite potential adversarial actions. Under the defined mission profile and safety constraints, this reachability-driven approach gives a dynamic periodicity with a mean restart interval of 0.5673 s and a standard deviation of 0.0506 s.

To evaluate the operational result of the rejuvenation on mission performance, the system's operational timeline was analyzed under both frameworks. This analysis focuses on availability, defined as the proportion of time the system spends in nominal operation versus the downtime caused by the rejuvenation and recovery processes. (Figure 5.2) presents a comparative visualization of the system modes over the mission duration, segmented into three distinct phases: Nominal operation mode (green), where the flight controller follows the trajectory, Rejuvenation mode (red), where the system undergoes the reset process with a fixed delay of $T_r = 50$ ms, and Recovery mode (orange), where the safety controller stabilizes the plant until the state error converges to the handover threshold.

The upper timeline illustrates the operational profile of the periodic SWR strategy. Driven by the conservative, reachability-driven constraints derived in the previous section, this benchmark triggers a system reset approximately every 0.57 seconds. While this frequency guarantees safety throughout the operation, but at the cost of reduced operational availability. This scheduling results in fragmentation of the operational timeline and the system is frequently trapped in a cycle of resetting and recovering, leaving minimal windows for the nominal controller to track the reference trajectory. Therefore, this always-on overhead creates a cumulative latency that extends the total mission completion time to 42.6 seconds. The system spends a significant portion of its uptime to regain stability rather than progressing along the desired path.

In comparison, the proposed framework maintains continuous availability for the majority of the mission. By shifting from a time-triggered to an event-triggered, the framework eliminates the blind restarts feature of the periodic approach. The timeline shows that rejuvenation is triggered only upon the detection of verified threats. This reduction in unnecessary interruptions allows the quadrotor to complete the reference trajectory in 29.0 seconds. This represents 31.9% reduction in total mission duration compared to the periodic benchmark. The results validate that continuous resetting is not a requisite for guaranteed safety if accurate, model-based attack detection mechanisms are used.

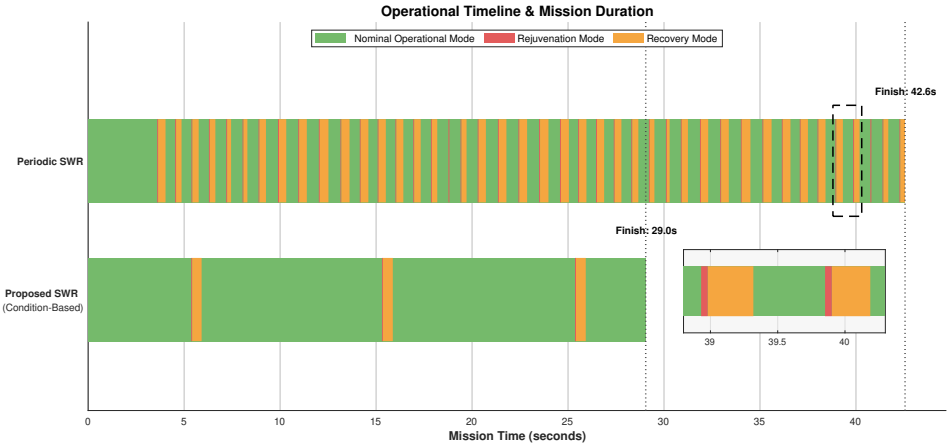


Figure 5.2: Operational timeline comparison of system modes (Nominal Operation, Rejuvenation, and Recovery) during the tracking of the reference. The top timeline depicts the Periodic SWR benchmark, showing operational fragmentation and a mission completion time. The bottom timeline shows the proposed Condition-Based SWR, which triggers rejuvenation only upon detection, reducing unnecessary interruptions

In addition to validate the dual-monitor framework, the system was subjected to two distinct attack profiles targeting the pitch (θ). (Figure 5.3) depicts the complementary response of the framework to these threats using a dual-axis format, where the left vertical axis defines the magnitude of the reactive pitch residual (r_ϕ) (orange trace), while the right vertical axis tracks the predictive Time-to-Violation (T^*) in seconds (purple trace).

In attack Scenario 1, an abrupt antagonistic attack is injected to cause immediate destabilization at $t = 5\ s$ shown in (Figure 5.3b). This high-magnitude input creates a sharp spike in the residual, allowing the observer-based monitor to trigger a rejuvenation almost instantaneously. This response is faster than the predictive horizon update, showing the necessity of the reactive layer for mitigating abrupt inputs while the predictive layer addresses slow-varying, long-horizon threats.

In attack scenario 2, a gradient attack is injected shown in (Figure 5.3a). The slowly accumulating error evades the residual-based detector, as the residual r_θ remains consistently below the anomaly threshold ($\tau_r \approx 0.09$). However, the predictive monitor identifies the drifting safety margin. As the estimated Time-to-Violation (T^*) decays below the safety horizon ($\tau_{safe} = 0.2$ s), the system triggers rejuvenation before a physical violation occurs.

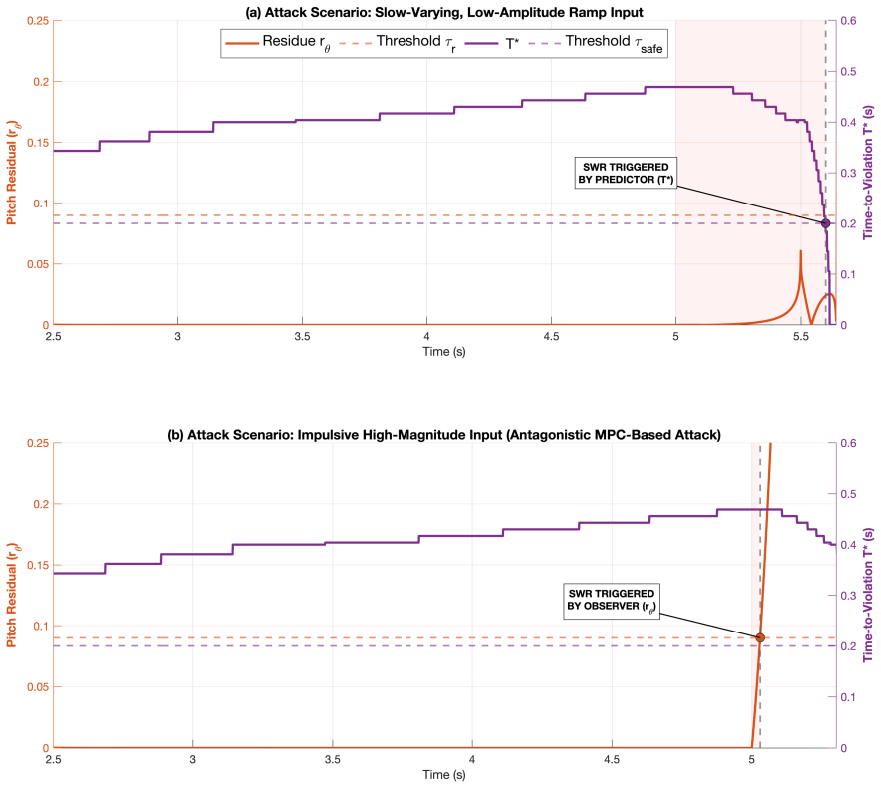


Figure 5.3: Dual-monitor response to pitch channel attacks. The plots shows dual vertical axes: the left axis tracks the pitch residual (r_θ) and the right axis tracks the Time-to-Violation (T^*). (a) Attack scenario 2- a gradient attack evades the residual threshold (τ_r) but triggers the predictive monitor when T^* breaches τ_{safe}. Attack scenario 1- an abrupt antagonistic attack spikes the residual, triggering the reactive monitor faster than the predictive mechanism

To ensure the practical feasibility of the proposed predictive monitor for real-world autonomous systems subject to timing constraints, the computational efficiency and precision of the explicit MPC formulation were validated. To enable real-time execution, the predictive monitor uses the Explicit MPC formulation discussed in Section 3.6. We computed the T^* lookup table by discretizing the state space, specifically the roll (ϕ) and pitch (θ) angles, within the safety bounds of $\pm\pi/3$ rad. The antagonistic optimizer was executed offline for each grid point to compute the minimum time to violation, resulting in the static safety map. The visualization of this pre-computed dataset is presented in (Figure 5.4), which depicts the geometric representation of the pre-computed safety index T^* of the quadrotor. The vertical axis represents the Time-to-Violation (T^*) in seconds, while the horizontal plane represents the system's attitude states. The landscape depicts a pyramid-like structure where the peak (Red zone, $T^* \approx 0.55$ s) is related to the stable equilibrium at the origin (hover condition). This indicates maximum robustness against attacks. As the system states deviate from the origin towards the safety boundaries ($|\phi|, |\theta| \to 90°$), the available safety margin decreases monotonically (Blue zone). This smooth gradient is important for the monitor's performance, as it ensures that the T^* metric provides a continuous warning signal as the drone approaches a critical state.

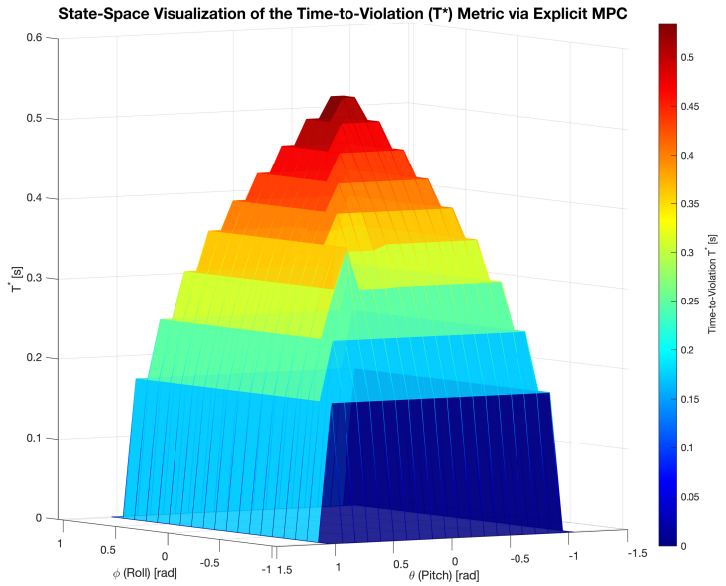


Figure 5.4: The Explicit Safety Landscape visualizing the computed Time-to-Violation (T^*) metric across the roll (ϕ) and pitch (θ) state space. The vertical axis represents the minimum time required for an optimal attacker to force a constraint violation from any given state

Since the online system relies on linear interpolation between these pre-computed grid points, it is essential to verify that this approximation does not introduce inaccuracies. (Figure 5.5) presents a cross-sectional validation of the explicit solution. This plot compares the exact T^* computed via the computationally expensive online solver (blue solid line) against the approximated value retrieved from the look-up table (red dashed line) along a test slice of the state space. The results confirm that the explicit surface captures the non-smooth dynamics of the safety function. The approximation error is explicitly quantified. The maximum absolute error remains strictly bounded below 0.012 s. Given the system's sampling rate, this discretization error is negligible and does not compromise the reliability of the rejuvenation trigger.

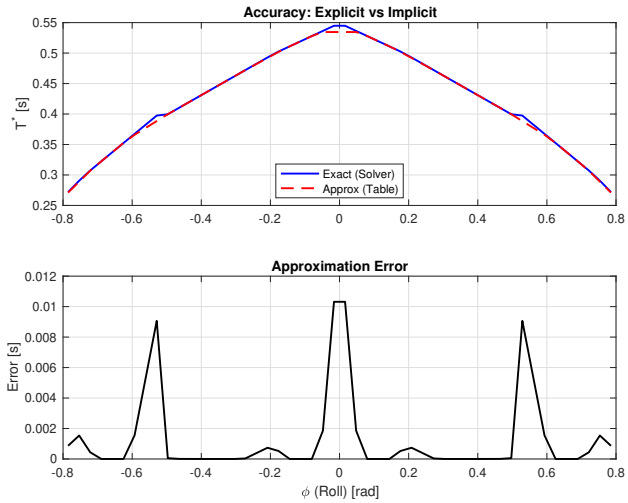


Figure 5.5: Cross-sectional validation of the explicit approximation. (Top) Comparison of the T^* values computed by the exact online solver (blue solid line) versus the explicit look-up table (red dashed line) along a test trajectory ($\theta = 0$, varying ϕ). (Bottom) The absolute approximation error introduced by the Explicit MPC formulation

Finally, the computational overhead was benchmarked to justify the shift to explicit

MPC. The standard implicit MPC (online optimization) required an average computation time of 163 ms per step. This latency is two orders of magnitude higher than the available sampling time (1 ms), computing the implicit method infeasible for real-time implementation. In comparison, the explicit MPC approach commutated an average execution time of approximately 0.11 ms. This represents a speedup factor of 1403×, confirming that the proposed predictive monitor can operate comfortably within the embedded real-time constraints without causing control latency.

5.7 Case Study II: The Four-Tank System (Process Control)

To validate the operation of the proposed condition-based SWR framework in the domain of process control, we apply the methodology to the Four-Tank System. This benchmark represents a typical Industrial Control Systems (ICSs) defined by multivariable cross-coupling and slower time constants. The primary safety objective in this case study is to ensure operational continuity and prevent critical constraint violations, such as tank overflows or dry-out conditions, even in the presence of malicious actuator attacks.

The physical plant is modeled using the nonlinear dynamics derived in Chapter 3. For this validation, the system is configured in the minimum phase setting where the pumps primarily feed the lower tanks, but the cross-coupling flows from the upper tanks remain significant enough to introduce complex interdependencies between the control inputs and the system states.

5.7.1 Implementation of Observer-based Monitor (Monitor-I)

To implement the reactive detection layer (Monitor I), we use a Luenberger-like nonlinear observer design that mirrors the mass-balance dynamics of the plant. The observer state vector is defined as the estimates of the water levels in all four tanks:

$$z = [\hat{h}_1, \hat{h}_2, \hat{h}_3, \hat{h}_4]^\top$$

While the model estimates the full state, the physical system is considered with sensors only at the lower tanks (Tank 1 and Tank 2). Therefore, the residual vector $r(t)$ is limited to the deviation in these two observable outputs:

$$r(t) = \begin{bmatrix} r_1(t) \\ r_2(t) \end{bmatrix} = \begin{bmatrix} h_1(t) - \hat{h}_1(t) \\ h_2(t) - \hat{h}_2(t) \end{bmatrix}$$

The observer dynamics are derived by applying the nonlinear equations of motion to the estimated states, augmented by a linear correction term derived from the residuals. Let $L \in \mathbb{R}^{4\times 2}$ denote the observer gain matrix, where L_{ij} represents the gain applied to the j-th residual to correct the i-th state. The specific dynamics for the observer are given by:

The estimates for the lower tanks are driven by the inflows from the upper tanks and the pumps, corrected directly by their own sensor residuals:

$$\frac{d\hat{h}_1}{dt} = -\frac{a_1}{A_1}\sqrt{2g\hat{h}_1} + \frac{a_3}{A_1}\sqrt{2g\hat{h}_3} + \frac{\gamma_1 k_1}{A_1}v_1 + L_{11}r_1 + L_{12}r_2 \tag{5.12a}$$

$$\frac{d\hat{h}_2}{dt} = -\frac{a_2}{A_2}\sqrt{2g\hat{h}_2} + \frac{a_4}{A_2}\sqrt{2g\hat{h}_4} + \frac{\gamma_2 k_2}{A_2}v_2 + L_{21}r_1 + L_{22}r_2 \tag{5.12b}$$

since the upper tanks (h_3, h_4) are not equipped with sensors, their estimates cannot be corrected directly. Instead, the observer relies on the cross-coupling in the error dynamics. For instance, an error in $\hat{h}_1$ implies an error in the inflow coming from Tank 3. The gain coefficients L_{31} and L_{42} propagate the visible errors from the lower tanks upstream to

correct the unmeasured upper states:

$$\frac{d\hat{h}_3}{dt} = -\frac{a_3}{A_3}\sqrt{2g\hat{h}_3} + \frac{(1-\gamma_2)k_2}{A_3}v_2 + L_{31}r_1 + L_{32}r_2 \tag{5.13a}$$

$$\frac{d\hat{h}_4}{dt} = -\frac{a_4}{A_4}\sqrt{2g\hat{h}_4} + \frac{(1-\gamma_1)k_1}{A_4}v_1 + L_{41}r_1 + L_{42}r_2 \tag{5.13b}$$

This structure ensures that even if an attack targets the unmeasured dynamics (e.g., a disturbance affecting Tank 3's outflow), the effect will eventually propagate to Tank 1, appear in residual r_1, and subsequently correct the estimate $\hat{h}_3$ via gain L_{31}.

The detection logic monitors the magnitude of the available residuals. We define the attack detection condition as:

$$\mathcal{D}(t) = \begin{cases} 1 & \text{if } |r_1(t)| > \tau_1 \vee |r_2(t)| > \tau_2 \\ 0 & \text{otherwise} \end{cases}$$

where τ_1 and τ_2 are static thresholds derived from the 3σ noise profile of the level sensors under nominal operating conditions.

5.7.2 Implementation of Predictive Monitor (Monitor II)

To estimate the Time-to-Violation (T^*) for the Four-Tank process, we use the antagonistic MPC formulation specifically configured to target the hydraulic constraints of the system. The safety boundaries are strictly defined by the physical geometry of the tanks. The unsafe set $\mathcal{X}_{unsafe}$ defined as the state space regions where the water level exceeds the defined tank height or falls below the given requirements:

$$\mathcal{X}_{unsafe} = \{x \in \mathbb{R}^4 \mid \exists i \in \{1, \ldots, 4\},\ h_i \geq H_{max}\}$$

The antagonistic optimization problem is set up to find the input sequence U that drives the system into $\mathcal{X}_{unsafe}$ in the minimum number of steps. The cost function J_N is weighted to prioritize volume accumulation in the lower tanks (which are subject to the direct pump inflows). We define the maximization objective as:

$$J_N(\mathbf{U}, x(k)) = \sum_{i=0}^{N-1} \left[q_1 \left(h_{1,k+i|k} - h_1^{eq}\right)^2 + q_2 \left(h_{2,k+i|k} - h_2^{eq}\right)^2 \right]$$

where $h_{1,k+i|k}$ and $h_{2,k+i|k}$ denote the predicted levels of the lower tanks at step i, and h_i^{eq} represents their nominal equilibrium setpoints. The weighting coefficients $q_1, q_2 > 0$ are tuned to focus the attack search on the tank closest to its physical limit. The solver uses the vertex enumeration method to identify the worst-case inputs that accelerate this violation.

5.7.3 Simulation Setup and Results

The proposed safety framework is validated through numerical simulations in the MATLAB environment. The physical plant is simulated using the nonlinear dynamics, while the controller and monitors operate at a discrete sampling interval. The control objective is to regulate the water levels of the lower tanks to the setpoints $[h_1^{eq}, h_2^{eq}] = [8, 6]^\top$ cm. The nominal MPC is tuned with a prediction horizon of $N = 10$ steps. The cost function weights are selected to prioritize tracking error minimization in the lower tanks while maintaining control effort:

$$Q = \text{diag}([10, 10, 1, 1]), \quad R = \text{diag}([1, 1])$$

where the diagonal entries of Q correspond to the state vector $x = [h_1, h_2, h_3, h_4]^\top$, giving higher penalties to the lower tanks, while penalizing both pump voltages equally. The constraints are set to the physical limits of the actuators ($0 \leq v_i \leq 15$ V).

The implementation of the defense mechanisms and the adversary models in this case study directly parallels the validation setup used for the Quadrotor. The framework uses the identical finite state machine architecture for SWR, managing the transitions between operational, rejuvenation, and recovery modes based on monitor triggers to ensure consistent logic across different physical platforms. To ensure consistency across validation domains, the attack strategy mirrors the methodology applied to the Quadrotor. We evaluate two distinct adversarial profiles, used here to target liquid level constraints: **Attack Scenario 1-** analogous to the UAV's worst-case attack, this adversary solves the antagonistic optimization problem to drive the pump inputs (v_1, v_2) to saturation. The objective is to force a tank overflow ($h > H_{max} = 20$ cm) in minimum time. **Attack Scenario 2-** this injects a slowly varying ramp signal. The slope is tuned to maintain the tracking residual below the observer's detection threshold, gradually filling the tank to the limit while masking the intrusion.

In the following sections, attack scenario 1 is used to evaluate the operational efficiency of the SWR frameworks, while attack scenario 2 is used to validate the sensitivity of the dual-monitor detection logic.

To define a performance baseline, the nominal controller was validated through Monte Carlo analysis. (Figure 5.6) presents the results of 50 simulation runs under randomized initial conditions. The state trajectories show that the controller drives the lower tank levels to their equilibrium targets (8 cm and 12 cm), showing convergence. The resulting control inputs respect the saturation limits, confirming that the linear MPC design provides a basis for the process regulation.

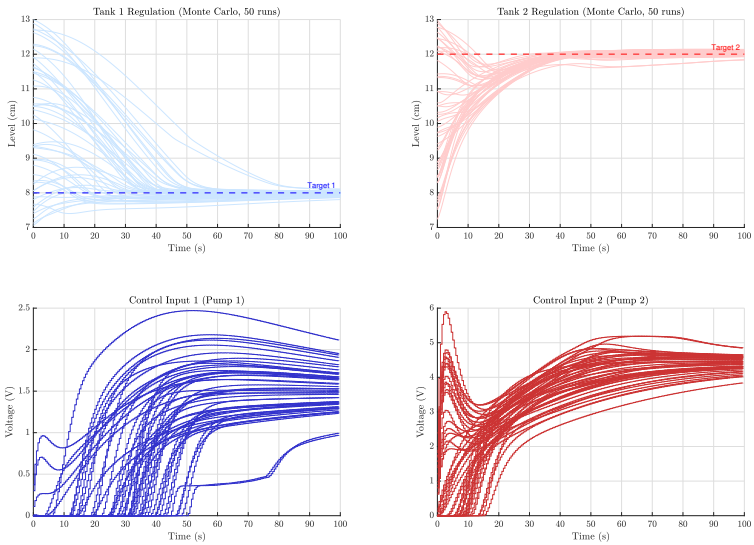


Figure 5.6: Nominal controller's performance via Monte Carlo simulation of 50. The state trajectories and control inputs confirm stable regulation to setpoints under randomized initialization, with all inputs remaining within linear saturation bounds

Following the nominal validation, the fail-safe capability of the benchmark periodic SWR framework was evaluated against active antagonistic attacks. Two separate attack instances were injected at distinct time intervals: $t = 150$ s and $t = 350$ s. In both cases, the adversary overrode the control inputs to force an immediate overflow. The (Figure 5.7) depicts the system response during these windows. At $t = 150$ s, an attack targeting pump 2 caused the water level in tank 2 to rise sharply. However, the periodic schedule successfully triggered a rejuvenation event before the level violated the safety limit. During the subsequent blackout (rejuvenation mode), the malicious input was eliminated, and the trusted safety controller stabilized the plant (recovery mode) before handing control back to the nominal MPC. A similar attack sequence was executed at

$t = 350$ s, this time targeting tank 1, confirming that the periodic strategy guarantees physical safety by containing the attack's impact within the safe envelope.

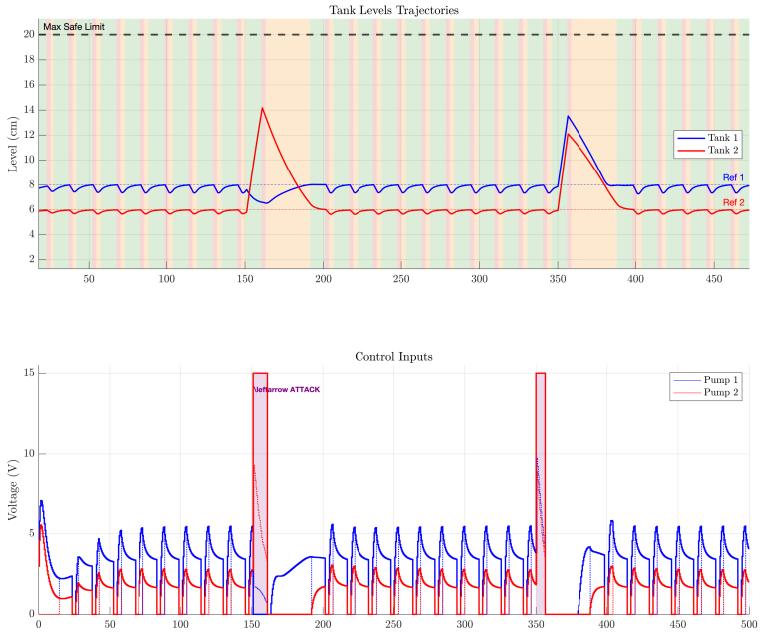


Figure 5.7: Evaluation of periodic SWR against attacks. The state evolution confirms that water levels remain bounded by the safety constraint (black dashed line), while the control inputs depict the adversarial override and the subsequent system blackout induced by the rejuvenation trigger

While periodic SWR confirms the safety guarantees, the operational cost of this strategy is depicted in the mission timeline presented in (Figure 5.8). The timeline shows the system's mode transitions over the entire simulation duration, where the green regions represent nominal operation, red represents rejuvenation (system blackout), and orange represents recovery. As shown in the plot, the timeline is heavily fragmented. Driven by the conservative reachability analysis (provided by [15]) required to prevent overflows, the system is forced to rejuvenate frequently, even during the long intervals of benign operation (e.g., $t = 0$ to 150 s). This always-on fragmentation significantly reduces the duty cycle, forcing the controller to spend a substantial portion of its operating time recovering from self-induced blackouts rather than maintaining steady-state regulation. This contrast highlights the fundamental limitation of the periodic approach, while it ensures guaranteed safety against potential attacks, it does so at the expense of continuous system availability.

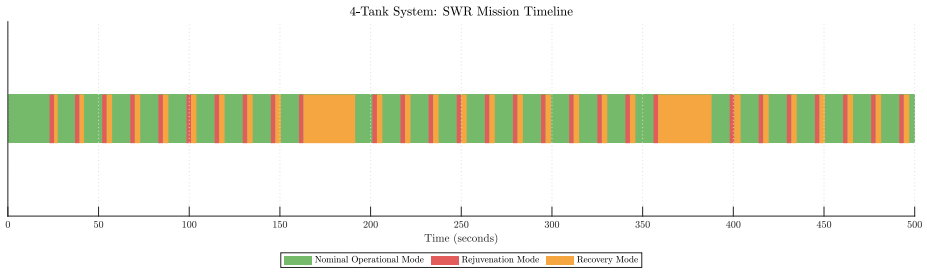


Figure 5.8: Mission timeline depiction for the periodic SWR method. The color-coded segments represent the active system state: Nominal Operation (Green), Rejuvenation (Red), and Safety Recovery (Orange), depicts the frequency of interruptions

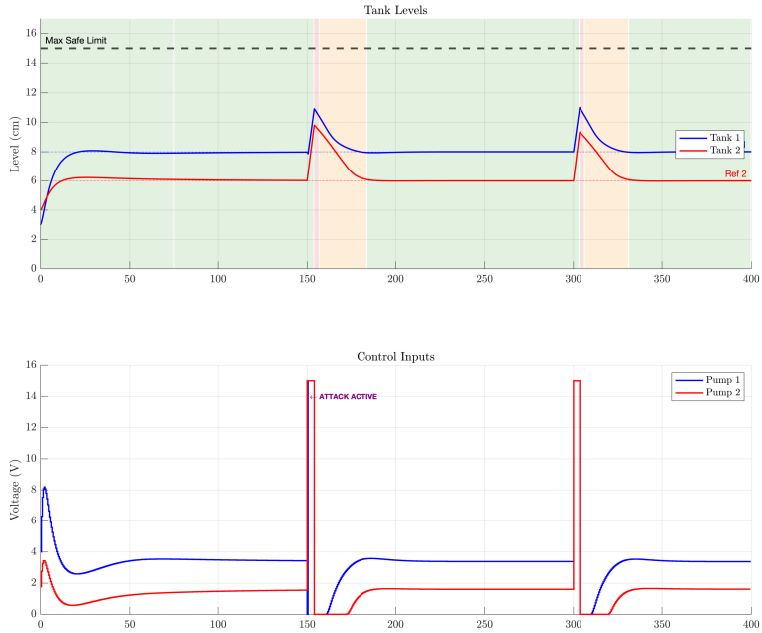


Figure 5.9: System response under Condition-Based SWR. Tank levels remain stable and within safety bounds, with the framework limiting the effects of attacks and control inputs remain continuous, with SWR interventions activated only upon attack detection

The proposed condition-based SWR framework is analyzed using the comparative simulation results shown in (Figure 5.9). For this comparison, the system is subjected to attack scenario 1 (antagonistic). This profile is selected as the primary test case because its abrupt nature presents the most immediate threat to physical safety, requiring rapid rejuvenation. In comparison to the periodic benchmark, where the control signal was continuously fragmented, the CB-SWR framework maintains uninterrupted nominal operation during the attack-free intervals. When the antagonistic attacks are injected at $t = 150$ s and $t = 300$ s, the adversary overrides Pump 1 to its saturation limit to force an overflow in tank 1. As the water level deviates from the reference and accelerates toward the constraint, the monitors identify the threat. The system triggers rejuvenation immediately upon detection, creating a fail-safe before the tank level can violate the constraints even without a conservative, always-on schedule.

The operational efficiency of this approach is further depicted in the mission timeline (Figure 5.10). Unlike the fragmentation observed in the periodic baseline, the CB-SWR timeline is majorly green (in nominal operational mode), indicating that the system maintains full availability for the majority of the duration. The rejuvenation (Red) and recovery (Orange) sequences appear only as distinct events strictly correlated with the attack instances. By eliminating unnecessary resets, the proposed framework resolves the trade-off between safety and availability, ensuring the process remains both secure against unexpected threats and operationally efficient.

To validate the dual-monitor architecture under the specific dynamics of the process control, the system was subjected to two distinct threat profiles defined in the setup. (Figure 5.11) illustrates the system's response to attack scenario 1 (abrupt antagonistic) attack injected at $t = 50$ s. The high-magnitude input causes a steephow to, instantaneous divergence between the plant and the model, forcing the residual (red trace) to spike vertically and cross the statistical anomaly threshold (τ_r) at $t \approx 50.5$ s. This immediate detection confirms that the reactive layer remains the primary defense against brute-force shocks, triggering the fail-safe before the predictive monitor requires a horizon update.

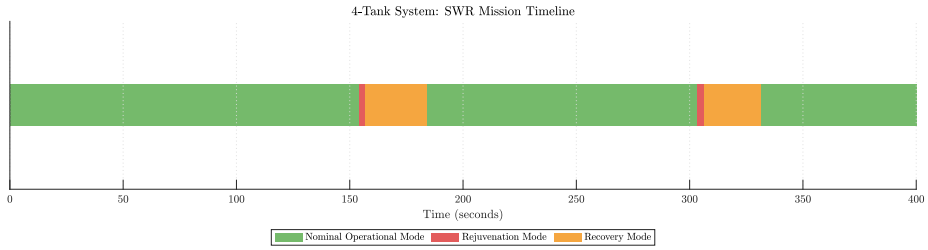


Figure 5.10: Mission timeline for the CB-SWR framework

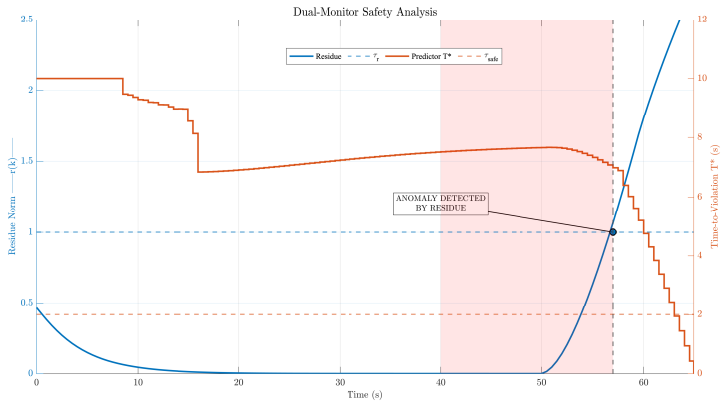


Figure 5.11: Dual-monitor response to an abrupt antagonistic attack ($t = 50$ s). The high-magnitude input causes an immediate residual spike, triggering (τ_r) faster than the predictive metric (T^*)

In comparison, the attack scenario 2 (slow varying gradient) attack shown in (Figure 5.12) shows the role of the predictive layer. Here, a slow-varying ramp signal is injected, to cause state drift while keeping the residual error consistently below the detection threshold (τ_r). While the reactive monitor remains blind to this accumulation, the predictive monitor identifies the near violation. As the water level drifts toward the overflow limit, the Time-to-Violation metric (T^*) decays, dropping below the safety horizon (τ_{safe}). This triggers the rejuvenation sequence well before physical safety is compromised, validating the framework's ability to intercept low-observable threats that bypass traditional anomaly detection.

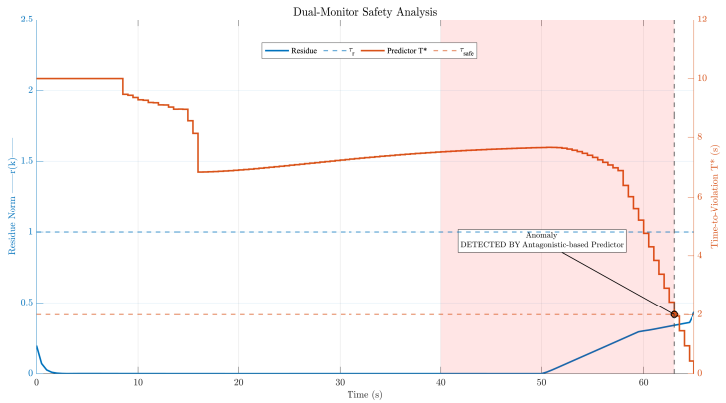


Figure 5.12: Dual-monitor response to a slow-varying gradient attack. The slowly accumulating error evades the Residue threshold (τ_r) but causes the Time-to-Violation (T^*) to decay, triggering rejuvenation when it satisfy define safety threshold (τ_{safe})

5.8 Discussion and Conclusion

This chapter presented the condition-based SWR framework as a method to address the trade-off between safety and availability in periodic rejuvenation schedules. The methodology shifts from time-triggered execution to evidence-based decision, by using the physical invariants of CPS to detect the nominal operation from active compromise. The core contribution is the dual-monitor architecture which fuses reactive and predictive detection. The Luenberger-like observer functions as a residual generator to identify anomalies where the system state diverges from the model. Complementing this, the Antagonistic MPC-based predictor introduces the Time-to-Violation (T^*) metric to quantify future vulnerability. Validation on the Quadrotor UAS platform shows that the framework reduced total mission duration by 31.9% compared to the periodic benchmark by eliminating the overhead of preventative resets. Furthermore, the implementation of Explicit MPC shows feasibility for real-time operation. Extensions to the Four-Tank process control system verified the applicability of the framework to prevent constraint violations such as hydraulic overflows without the timeline fragmentation observed in periodic method.

Despite these results, the work presents specific limitations. First, the reliability of the detection logic depends on the accuracy of the system model, which may be compromised by discrepancies between the mathematical model and the physical plant, including unmodeled external disturbances or parameter uncertainties. Secondly, the threat model is limited to actuator-level attacks; stealthy strategies such as sensor spoofing or false data injection that might mislead the monitor are not addressed in this iteration. Finally, the results presented are preliminary and rely on numerical simulation. While the concept of software rejuvenation is validated in safety-critical commercial hardware, evidenced by the Boeing 787 sequential resets for flight control software [60], the maintenance of high-availability telecommunications systems at AT&T Bell Labs [108], and automated container management in cloud environments like AWS and Google Cloud [109], [110] but its application to the specific dynamics of CPS requires further analysis. Therefore, validation on a physical testbed is required to assess real-world computational overhead and timing constraints.

In conclusion, Condition-Based SWR offers an alternative to fixed-interval schedules. By integrating physics-based anomaly detection with predictive safety estimation, the framework ensures autonomous systems remain secure against control-theoretic attacks while maintaining the performance required for mission execution.

Conclusions

Chapter 6

Conclusion

6.1 Summary of Research

This thesis presented a model-based design framework to address the security challenges inherent to autonomous CPS. The research began by establishing the critical distinction between IT and OT security, while traditional networks prioritize data confidentiality, CPS security must prioritize operational availability and physical safety.

A comprehensive analysis of real-world cases, including Stuxnet, Havex, and PIPEDREAM, revealed a consistent kill chain in modern cyber-physical attacks. As detailed in the case studies, attackers often gain initial access via enterprise layers (e.g., phishing or compromised vendor software), propagate laterally to the operational network, and establish persistence on the controller. These adversaries do not always attack immediately; they sit and wait for the optimal timing to manipulate physical dynamics, often masking their presence until the system is most vulnerable.

To model the worst-case impact of such persistent, intelligent adversaries, this research uses the antagonistic MPC framework. While initial research studied the cost-maximization method, the research refined this into a constraint-violation attack model. This formulation validated that an optimal attacker aims not just to increase error, but to actively force the system across physical safety boundaries, such as a tank overflow or critical rotor inclination. Furthermore, to ensure the validity of these attack models in realistic, uncertain environments, the framework was extended to a robust antagonistic framework. By using a max-min optimization approach, the model accounts for worst-case environmental disturbances, ensuring that the computed attack trajectories remain effective even when external uncertainties oppose the adversary's objective. To address the computational challenge of this formulation, specifically, the non-convex nature of maximizing a convex cost function, a vertex enumeration method was used. This solver guarantees the identification of the global optimum, ensuring that the worst-case impact is accurately quantified rather than underestimated.

To counteract these persistent threats, the research identified SWR as a potent mitigation strategy, as real-world malware (like PIPEDREAM) relies on modifying logic or maintaining a resident state in memory, SWR offers a unique advantage; by completely wiping the runtime environment and re-initializing from a trusted root of trust, it eliminates all malicious modifications and adversarial footholds. However, the state-of-the-art periodic SWR was found to be operationally inefficient, causing an "always-on" availability cost regardless of the actual threat level. Therefore, this thesis proposed a condition-based SWR framework. This novel architecture shifts SWR from a periodic, time-triggered to a condition based framework using a dual-monitor architecture.

The framework uses a reactive Luenberger-like Observer (Monitor I) designed to detect abrupt, high-magnitude attacks; when an antagonist attempts to force a violation rapidly, the sudden deviation between the estimated and actual states creates a spike in the residual signal ($r(t)$), triggering immediate rejuvenation. Complementing this is a proactive antagonistic predictor (Monitor II), designed to detect slow-varying gradient like attacks that evade residual thresholds. By repurposing the antagonistic control model as a defense tool, this monitor calculates the Time-to-Violation (T^*), a metric that forecasts the response of a abrupt or slowly acting adversary, allowing the system to trigger rejuvenation process before a safety violation becomes inevitable.

6.2 Research Contributions

The key contributions of this dissertation to the field of CPS security are:

- While antagonistic control is a known theoretical concept, this thesis advanced its application by implementing it on realistic nonlinear systems (Quadrotor UAV and Four-Tank System). Furthermore, the work refined the attack objective from generic cost maximization to a constraint-violation model, providing a rigorous method to quantify the worst-case physical impact an adversary can cause.
- The thesis contributed a novel methodology by repurposing the adversarial framework from an offensive into a defensive monitoring tool. By solving the antagonistic problem online at each instant, the system computes the worst-case trajectory and the Time-to-Violation (T^*). This transforms the attacker's advantage into a safety metric for the defender.
- The research filled a gap in the literature by proposing a condition-based alternative to periodic rejuvenation. By integrating the dual-monitor system, the framework ensures that the effective yet operationally intensive mitigation of SWR is only deployed when a credible threat is detected, thereby optimizing the trade-off between security and availability.
- To enable the online computation of the complex antagonistic metrics (T^* and P^*), the framework uses Explicit MPC (mp-QP). This contribution shows that complex, model-based safety checks can be executed in real-time on embedded systems without prohibitive latency.

6.3 Future Work

While this thesis proposes a framework for CPS safety, the following areas are identified for future research to address current limitations:

- **Experimental Validation and Hardware-in-the-Loop (HIL):** The current validation relies on numerical simulations but future work must focus on Hardware-in-the-Loop (HIL) implementation to test the framework under real-world constraints. Specifically, the feasibility of SWR needs to be evaluated on physical embedded controllers to quantify the actual overhead of the rejuvenation cycle and the robustness of the dual monitors against real-world sensor noise and network latency.
- **Defense Against Stealthy Sensor Integrity Attacks:** The current framework is effective against actuator attacks. However, sophisticated sensor-side integrity attacks (such as replay attacks or zero-dynamics attacks) where the adversary masks their presence remain a challenge. Future work would integrate analytical redundancy to cross-validate sensor data against physical predictions, and Moving Target Defense (MTD) to dynamically alter system parameters. These additions would detect stealthy injections that attempt to hide their physical impact, providing a comprehensive defense against adversaries with high operational knowledge.
- **Extension to Peripheral Components (Edge SWR):** Currently, the rejuvenation focus is primarily on the central controller. However, as seen in complex attacks, threats can persist in firmware. Future research should investigate extending SWR to smart sensors and actuators. This involves developing protocols to reset peripheral firmware to eliminate threats at the edge, requiring new control strategies to handle the temporary loss of actuation or sensing during the reset phase.

References

[1] Greer, C., Burns, M., Wollman, D., and Griffor, E., "Cyber-physical systems and internet of things," 2019.

[2] Lee, E. A., "Cyber physical systems: Design challenges," *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pp. 363–369, 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:7870743.

[3] Duo, W., Zhou, M., and Abusorrah, A., "A survey of cyber attacks on cyber physical systems: Recent advances and challenges," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 5, pp. 784–800, 2022.

[4] Mangharam, R. and Pajic, M., "Distributed control for cyber-physical systems," 2013.

[5] Teixeira, A., Shames, I., Sandberg, H., and Johansson, K. H., "A secure control framework for resource-limited adversaries," *Automatica*, vol. 51, pp. 135–148, 2015.

[6] Smith, R. S., "Covert misappropriation of networked control systems: Presenting a feedback structure," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 82–92, 2015.

[7] Chen, J. and Shi, Y., "Stochastic model predictive control framework for resilient cyber-physical systems: Review and perspectives," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2207, p. 20 200 371, 2021.

[8] Lipp, T. and Boyd, S., "Antagonistic control," *Systems & Control Letters*, vol. 98, pp. 44–48, 2016.

[9] Dibaji, S. M., Pirani, M., Flamholz, D. B., Annaswamy, A. M., Johansson, K. H., and Chakrabortty, A., "A systems and control perspective of cps security," *Annual reviews in control*, vol. 47, pp. 394–411, 2019.

[10] Casola, V., De Benedictis, A., Mazzocca, C., and Montanari, R., "Designing secure and resilient cyber-physical systems: A model-based moving target defense approach," *IEEE Transactions on Emerging Topics in Computing*, vol. 12, no. 2, pp. 631–642, 2022.

[11] Flammini, F. et al., *Resilience of cyber-physical systems*. Springer, 2019.

[12] Romagnoli, R., Krogh, B. H., De Niz, D., Hristozov, A. D., and Sinopoli, B., "Software rejuvenation for safe operation of cyber–physical systems in the presence of run-time cyberattacks," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 4, pp. 1565–1580, 2023.

[13] Alonso, J., Bovenzi, A., Li, J., Wang, Y., Russo, S., and Trivedi, K., "Software rejuvenation - do it & telco industries use it?" In *Proceedings - 23rd IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW 2012*, 2012, pp. 299–304. DOI: 10.1109/ISSREW.2012.96.

[14] Romagnoli, R., Krogh, B. H., Niz, D. D., Hristozov, A. D., and Sinopoli, B., "Runtime system support for cps software rejuvenation," *IEEE Transactions on Emerging Topics in Computing*, vol. 11, pp. 594–604, 3 Jul. 2023. DOI: 10.1109/TETC.2023.3267899.

[15] Abdi, F., Chen, C. Y., Hasan, M., Liu, S., Mohan, S., and Caccamo, M., "Guaranteed physical security with restart-based design for cyber-physical systems," in *Proceedings - 9th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2018*, Institute of Electrical and Electronics Engineers Inc., Aug. 2018, pp. 10–21. DOI: 10.1109/ICCPS.2018.00010.

[16] Harkat, H., Camarinha-Matos, L. M., Goes, J., and Ahmed, H. F., "Cyber-physical systems security: A systematic review," *Computers & Industrial Engineering*, vol. 188, p. 109 891, 2024. DOI: 10.1016/j.cie.2024.109891.

[17] Krotofil, M. and Gollmann, D., "Industrial control systems security: What is happening?" In *Proceedings of the 11th IEEE International Conference on Industrial Informatics (INDIN)*, IEEE, 2013, pp. 670–675. DOI: 10.1109/INDIN.2013.6622966.

[18] Barreto, C., Schwartz, G., and Cardenas, A. A., "Cyber-risk: Cyber-physical systems versus information technology systems," in *Safety, Security and Privacy for Cyber-Physical Systems*, ser. Lecture Notes in Control and Information Sciences, vol. 486, Springer, 2021, pp. 319–336. DOI: 10.1007/978-3-030-65048-3_14.

[19] Lüders, S., "Stuxnet and the impact on accelerator control systems," in *Proceedings of the 2nd International Particle Accelerator Conference (IPAC2011)*, San Sebastián, Spain: JACoW, 2011, pp. 2932–2934.

[20] Karnouskos, S., "Stuxnet worm impact on industrial cyber-physical system security," in *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, (Often cited regarding "Myth and Reality" of Stuxnet), IEEE, 2011, pp. 4490–4494. DOI: 10.1109/IECON.2011.6120048.

[21] Lee, R. M., Assante, M. J., and Conway, T., "Analysis of the cyber attack on the ukrainian power grid," E-ISAC and SANS Institute, Defense Use Case, Mar. 2016. [Online]. Available: https://www.cisa.gov/sites/default/files/publications/E-ISAC_SANS_Ukraine_DUC_5.pdf.

[22] Industry Report, *Ukraine power grid cyberattack and us susceptibility: Cybersecurity implications of smart grid advancements in the us*, Whitepaper/Report, Accessed: 2025, 2016.

[23] Institute, S., "The impact of dragonfly malware on industrial control systems," SANS Industrial Control Systems Security Blog, Tech. Rep., 2014, Havex/Dragonfly Campaign Analysis.

[24] StealthLabs. "Leading us gasoline pipeline hit by ransomware attack." Accessed: 2025. [Online]. Available: https://www.stealthlabs.com/news/leading-us-gasoline-pipeline-hit-by-ransomware-attack/.

[25] TXOne Networks. "Analysis of the pipedream local exploit." Technical analysis of CVE-2020-15368 and INCONTROLLER toolkit. [Online]. Available: https://www.txone.com/blog/.

[26] González-Nalda, P., Etxeberria-Agiriano, I., Calvo, I., and Otero, M. C., "A modular cps architecture design based on ros and docker," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 11, no. 4, pp. 949–955, 2017.

[27] Khalil, H. K. and Grizzle, J. W., *Nonlinear systems*. Prentice hall Upper Saddle River, NJ, 2002, vol. 3.

[28] Sontag, E. D., *Mathematical control theory: deterministic finite dimensional systems*. Springer Science & Business Media, 2013, vol. 6.

[29] Simon, D., *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[30] Åström, K. J. and Murray, R., *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2021.

[31] Slotine, J.-J. E., Li, W., et al., *Applied nonlinear control*. Prentice hall Englewood Cliffs, NJ, 1991, vol. 199.

[32] Åström, K. J. and Wittenmark, B., *Computer-controlled systems: theory and design*. Courier Corporation, 2013.

[33] Schwenzer, M., Ay, M., Bergs, T., and Abel, D., "Review on model predictive control: An engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, pp. 1327–1349, 2021.

[34] Teixeira, A., Sou, K. C., Sandberg, H., and Johansson, K. H., "Secure control systems: A quantitative risk management approach," *IEEE Control Systems*, vol. 35, pp. 24–45, 1 Feb. 2015. DOI: 10.1109/MCS.2014.2364709.

[35] Li, Y., Yang, Y., Zhao, Z., Zhou, J., and Quevedo, D. E., "Deception attacks on remote estimation with disclosure and disruption resources," *IEEE Transactions on Automatic Control*, vol. 68, no. 7, pp. 4096–4112, 2022.

[36] Park, G., Shim, H., Lee, C., Eun, Y., and Johansson, K. H., "When adversary encounters uncertain cyber-physical systems: Robust zero-dynamics attack with disclosure resources," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, 2016, pp. 5085–5090.

[37] Teixeira, A., Sou, K. C., Sandberg, H., and Johansson, K. H., "Secure control systems: A quantitative risk management approach," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 24–45, 2015. DOI: 10.1109/MCS.2014.2364709.

[38] Teixeira, A., Shames, I., Sandberg, H., and Johansson, K. H., "A secure control framework for resource-limited adversaries," *Automatica*, vol. 51, pp. 135–148, 2015. DOI: 10.1016/j.automatica.2014.10.067.

[39] Pasqualetti, F., Dörfler, F., and Bullo, F., "Attack detection and identification in cyber-physical systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 11, pp. 2715–2729, 2013. DOI: 10.1109/TAC.2013.2266831.

[40] De Persis, C. and Tesi, P., "Input-to-state stabilizing control under denial-of-service," *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 2930–2944, 2015. DOI: 10.1109/TAC.2015.2416924.

[41] Mo, Y. and Sinopoli, B., "Secure control against replay attacks," in *Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing*, 2009, pp. 911–918. DOI: 10.1109/ALLERTON.2009.5394956.

[42] Mo, Y., Chabukswar, R., and Sinopoli, B., "Detecting integrity attacks on scada systems," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1396–1407, 2014. DOI: 10.1109/TCST.2013.2280899.

[43] Mo, Y. and Sinopoli, B., "False data injection attacks in control systems," in *First Workshop on Secure Control Systems (SCS), CPS Week*, Stockholm, Sweden, 2010.

[44] Teixeira, A., Shames, I., Sandberg, H., and Johansson, K. H., "Revealing stealthy attacks in control systems," in *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing*, 2012, pp. 1806–1813. DOI: 10.1109/Allerton.2012.6483441.

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

[46] Fawzi, H., Tabuada, P., and Diggavi, S., "Secure estimation and control for cyber-physical systems under adversarial attacks," *IEEE Transactions on Automatic control*, vol. 59, no. 6, pp. 1454–1467, 2014.

[47] Shoukry, Y., Nuzzo, P., Puggelli, A., Sangiovanni-Vincentelli, A. L., Seshia, S. A., and Tabuada, P., "Secure state estimation for cyber-physical systems under sensor attacks: A satisfiability modulo theory approach," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 4917–4932, 2017.

[48] Teixeira, A., Shames, I., Sandberg, H., and Johansson, K. H., "Revealing stealthy attacks in control systems," in *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2012, pp. 1806–1813.

[49] Sandberg, H., Amin, S., and Johansson, K. H., "Cyberphysical security in networked control systems: An introduction to the issue," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 20–23, 2015.

[50] Kholidy, H. A., "Autonomous mitigation of cyber risks in the cyber–physical systems," *Future Generation Computer Systems*, vol. 115, pp. 171–187, 2021.

[51] Teixeira, A., Sou, K. C., Sandberg, H., and Johansson, K. H., "Secure control systems: A quantitative risk management approach," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 24–45, 2015.

[52] Ding, D., Han, Q.-L., Ge, X., and Wang, J., "Secure state estimation and control of cyber-physical systems: A survey," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 176–190, 2021. DOI: 10.1109/TSMC.2020.3041121.

[53] Ali, R. F., Muneer, A., Dominic, P. D., Ghaleb, E. A., and Al-Ashmori, A., "Survey on cyber security for industrial control systems," in *2021 International Conference on Data Analytics for Business and Industry (ICDABI)*, IEEE, 2021, pp. 630–634.

[54] Ma, L., Chu, Z., Yang, C., Wang, G., and Dai, W., "Recursive watermarking-based transient covert attack detection for the industrial cps," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1709–1719, 2023.

[55] Tantouris, N. M., "Robust fault tolerant detection and isolation techniques for actuator failures in dynamical systems," 2017.

[56] Pajic, M. et al., "Robustness of attack-resilient state estimators," in *2014 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, IEEE, 2014, pp. 163–174.

[57] Sha, L., "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, p. 20, 2001.

[58] Bak, S., Chivukula, D. K., Adekunle, O., Sun, M., Caccamo, M., and Sha, L., "The system-level simplex architecture for improved real-time embedded system safety," in *2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, IEEE, 2009, pp. 99–107.

[59] Huang, Y. and Kintala, C., "Software rejuvenation: Analysis, module and applications," Tech. Rep.

[60] Arroyo, M. A., Ziad, M. T. I., Kobayashi, H., Yang, J., and Sethumadhavan, S., "Yolo: Frequently resetting cyber-physical systems for security," in *Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2019*, SPIE, vol. 11009, 2019, pp. 166–183.

[61] Huang, Y., Kintala, C., Kolettis, N., and Fulton, N., "Software rejuvenation: Analysis, module and applications," in *Twenty-Fifth International Symposium on Fault-Tolerant Computing. Digest of Papers*, 1995, pp. 381–390. DOI: 10.1109/FTCS.1995.466961.

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

[63] Romagnoli, R., Krogh, B. H., and Sinopoli, B., "Safety and liveness of software rejuvenation for secure tracking control," in *2019 18th European Control Conference (ECC)*, IEEE, 2019, pp. 2215–2220.

[64] Arauz, T., Maestre, J. M., Romagnoli, R., Sinopoli, B., and Camacho, E. F., "A linear programming approach to computing safe sets for software rejuvenation," *IEEE Control Systems Letters*, vol. 6, pp. 1214–1219, 2022. DOI: 10.1109/LCSYS.2021.3090448.

[65] Arauz, T., Maestre, J. M., Quevedo, D., and Camacho, E. F., "Tree-based model predictive control strategy for software rejuvenation," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, IEEE, 2022, pp. 1124–1129.

[66] Arauz, T., Maestre, J. M., Chanfreut, P., Quevedo, D. E., and Camacho, E. F., "Open and closed-loop predictive control strategies for software rejuvenation," *IEEE Transactions on Emerging Topics in Computing*, vol. 13, pp. 330–340, 2 2025. DOI: 10.1109/TETC.2024.3481997.

[67] Chen, A., Mitsopoulos, K., and Romagnoli, R., "Reinforcement learning-based optimal control and software rejuvenation for safe and efficient uav navigation," in *2023 62nd IEEE Conference on Decision and Control (CDC)*, IEEE, 2023, pp. 7527–7532.

[68] Romagnoli, R., Krogh, B. H., and Sinopoli, B., "Robust software rejuvenation for cps with state estimation and disturbances," in *2020 American Control Conference (ACC)*, IEEE, 2020, pp. 1241–1246.

[69] Romagnoli, R., Griffioen, P., Krogh, B. H., and Sinopoli, B., "Software rejuvenation under persistent attacks in constrained environments," in *IFAC-PapersOnLine*, vol. 53, Elsevier B.V., 2020, pp. 4088–4094. DOI: 10.1016/j.ifacol.2020.12.2437.

[70] Banerjee, V., Hounsinou, S., Olufowobi, H., Hasan, M., and Bloom, G., "Secure reboots for real-time cyber-physical systems," in *Proceedings of the 4th Workshop on CPS & IoT Security and Privacy*, 2022, pp. 27–33.

[71] Paroli, L., Botarelli, T., Carnevali, L., and Vicario, E., "A compositional approach to coordinated software rejuvenation of component-based systems," in *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, 2024, pp. 593–604.

[72] Griffioen, P., Romagnoli, R., Krogh, B. H., and Sinopoli, B., *Secure Networked Control for Decentralized Systems via Software Rejuvenation*. 2020. DOI: 10.0/Linux-x86-64.

[73] Torquato, M., Maciel, P., and Vieira, M., "Software rejuvenation meets moving target defense: Modeling of time-based virtual machine migration approach," in *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, 2022, pp. 205–216.

[74] Sabatino, F., *Quadrotor control: Modeling, nonlinearcontrol design, and simulation*, 2015.

[75] Beard, R. W. and McLain, T. W., *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

[76] Johansson, K. H., "The quadruple-tank process: A multivariable laboratory process with an adjustable zero," *IEEE Transactions on control systems technology*, vol. 8, no. 3, pp. 456–465, 2002.

[77] Darby, M. L. and Nikolaou, M., "Mpc: Current practice and challenges," *Control Engineering Practice*, vol. 20, no. 4, pp. 328–342, 2012.

[78] Okasha, M., Kralev, J., and Islam, M., "Design and experimental comparison of pid, lqr and mpc stabilizing controllers for parrot mambo mini-drone," *Aerospace*, vol. 9, no. 6, p. 298, 2022.

[79] Camacho, E. F. and Bordons, C., "Constrained model predictive control," in *Model predictive control*, Springer, 2007, pp. 177–216.

[80] Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O., "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[81] Borrelli, F., Bemporad, A., and Morari, M., *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[82] Simon, D., *Model predictive control in flight control design stability and reference tracking*. Linkopings Universitet (Sweden), 2014.

[83] Kerrigan, E. C. and Maciejowski, J. M., "Designing model predictive controllers with prioritised constraints and objectives," in *Proceedings. IEEE International Symposium on Computer Aided Control System Design*, IEEE, 2002, pp. 33–38.

[84] Boyd, S. and Vandenberghe, L., *Convex optimization*. Cambridge university press, 2004.

[85] Chen, J. and Patton, R. J., *Robust model-based fault diagnosis for dynamic systems*. Springer Science & Business Media, 2012, vol. 3.

[86] Pannocchia, G., "Robust disturbance modeling for model predictive control with application to multivariable ill-conditioned processes," *Journal of Process Control*, vol. 13, no. 8, pp. 693–701, 2003.

[87] Krupa, P., Zanon, M., and Bemporad, A., "Learning disturbance models for offset-free reference tracking," *IEEE Transactions on Automatic Control*, 2025.

[88] Chen, W.-H., Yang, J., Guo, L., and Li, S., "Disturbance-observer-based control and related methods—an overview," *IEEE Transactions on industrial electronics*, vol. 63, no. 2, pp. 1083–1095, 2015.

[89] Waslander, S. and Wang, C., "Wind disturbance estimation and rejection for quadrotor position control," in *AIAA Infotech@ Aerospace conference and AIAA unmanned... Unlimited conference*, 2009, p. 1983.

[90] Li, T., Xing, H., Hashemi, E., Taghirad, H. D., and Tavakoli, M., "A brief survey of observers for disturbance estimation and compensation," *Robotica*, vol. 41, no. 12, pp. 3818–3845, 2023.

[91] Köhler, J., Müller, M. A., and Allgöwer, F., "A novel constraint tightening approach for nonlinear robust model predictive control," in *2018 Annual American control conference (ACC)*, IEEE, 2018, pp. 728–734.

[92] De La Pena, D. M., Alamo, T., Ramirez, D., and Camacho, E., "Min-max model predictive control as a quadratic program," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 263–268, 2005.

[93] Ten, C.-W., Liu, C.-C., and Manimaran, G., "Vulnerability assessment of cybersecurity for scada systems," *IEEE Transactions on Power Systems*, vol. 23, no. 4, pp. 1836–1846, 2008.

[94] Cárdenas, A. A., Amin, S., Lin, Z.-S., Huang, Y.-L., Huang, C.-Y., and Sastry, S., "Attacks against process control systems: Risk assessment, detection, and response," in *Proceedings of the 6th ACM symposium on information, computer and communications security*, 2011, pp. 355–366.

[95] Humayed, A., Lin, J., Li, F., and Luo, B., "Cyber-physical systems security—a survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1802–1831, 2017.

[96] Abdalmoaty, M. R., Anand, S. C., and Teixeira, A. M., "Privacy and security in network controlled systems via dynamic masking," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 991–996, 2023.

[97] Kim, J. and Lavaei, J., "System identification from partial observations under adversarial attacks," *arXiv preprint arXiv:2504.00244*, 2025.

[98] Kwon, C. and Hwang, I., "Reachability analysis for safety assurance of cyber-physical systems against cyber attacks," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 2272–2279, 2017.

[99] Cavanini, L., Felicetti, R., Ferracuti, F., Freddi, A., Longhi, S., and Siyyal, S. A., "Antagonistic model predictive control for quadrotor cyber attacks," in *2024 20th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, IEEE, 2024, pp. 1–6.

[100] Baldini, A., Felicetti, R., Freddi, A., Longhi, S., and Monteriù, A., "Modeling and control of a telescopic quadrotor using disturbance observer based control," in *2022 30th Mediterranean Conference on Control and Automation (MED)*, IEEE, 2022, pp. 396–402.

[101] Baldini, A., Felicetti, R., Freddi, A., Longhi, S., and Monteriu, A., "Actuator fault tolerant control of variable pitch quadrotor vehicles," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 4095–4102, 2020.

[102] Siyyal, S. A., Maestre, J. M., Freddi, A., and Longhi, S., "A constraints-aware antagonistic controller with disturbance-adaptive attacks," *IFAC-PapersOnLine*, vol. 59, no. 11, pp. 13–18, 2025.

[103] Simion, R., "Convex polytopes and enumeration," *Advances in Applied Mathematics*, vol. 18, no. 2, pp. 149–180, 1997.

[104] Rockafellar, R. T., "Convex integral functionals and duality," in *Contributions to nonlinear functional analysis*, Elsevier, 1971, pp. 215–236.

[105] Siyyal, S. A., Freddi, A., Maestre, J. M., Romagnoli, R., Baldini, A., and Longhi, S., "Condition-based software rejuvenation framework for cyber-physical systems using a dual-monitor architecture," Manuscript under review for the 22nd IFAC World Congress, 2026, 2026.

[106] Arroyo, M., Kobayashi, H., Sethumadhavan, S., and Yang, J., "Fired: Frequent inertial resets with diversification for emerging commodity cyber-physical systems," Feb. 2017. [Online]. Available: http://arxiv.org/abs/1702.06595.

[107] Romagnoli, R., Krogh, B. H., Niz, D. de, and Sinopoli, B., "Software rejuvenation for secure tracking control," Oct. 2018. [Online]. Available: http://arxiv.org/abs/1810.10468.

[108] Alonso, J., Bovenzi, A., Li, J., Wang, Y., Russo, S., and Trivedi, K., "Software rejuvenation: Do it & telco industries use it?" In *2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops*, IEEE, 2012, pp. 299–304.

[109] Alonso, J., Matias, R., Vicente, E., Maria, A., and Trivedi, K. S., "A comparative experimental study of software rejuvenation overhead," *Performance Evaluation*, vol. 70, no. 3, pp. 231–250, 2013.

[110] Avritzer, A., Cotroneo, D., Huang, Y., and Trivedi, K., "Software aging and rejuvenation: A genesis," in *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, IEEE, 2020, pp. 319–320.