Adaptive Control Class – Mini-Project 2

Adaptive Control Lyapunov Functions and Adaptive Control Barrier Functions with Concurrent Learning

Department of Aerospace engineering

Semester: Spring 2026 Due Date: 15 Khordad 1405

Based on Chapters 4 and 5 of "Adaptive and Learning-Based Control of Safety-Critical Systems" by Max Cohen and Calin Belta

Abstract

This mini-project explores the design of adaptive controllers for nonlinear systems with parametric uncertainties, combining stability and safety guarantees. You will develop Adaptive Control Lyapunov Functions (aCLFs) and Adaptive Control Barrier Functions (aCBFs) for a MIMO nonlinear system, enhance parameter estimation via concurrent learning, and construct robust adaptive CBFs to safeguard against residual estimation errors. All controllers are implemented as quadratic programs (QPs) in MATLAB/Simulink. A central theme is the trade-off between adaptation speed, excitation, and safety robustness.

1 Project Overview

This mini-project builds upon the CLF/CBF framework from Mini-Project 1 by incorporating parametric model uncertainties and adaptive mechanisms. The core tasks are:

1. Model a MIMO nonlinear system with structured parametric uncertainties, validated from the literature.
2. Design an Adaptive CLF (aCLF) and formulate the certainty-equivalence minimum-norm QP with adaptive parameter update.
3. Improve parameter convergence using **concurrent learning** (CL); compare convergence with and without CL, and analyze the minimum eigenvalue of the history stack.
4. Design an Adaptive CBF (aCBF) that uses the parameter estimates to enforce safety constraints via a QP.

5. Design a **Robust Adaptive CBF (raCBF)** that incorporates a robustness margin to guarantee safety despite estimation errors.

6. Perform extensive simulations comparing aCBF vs. raCBF, and discuss the influence of concurrent learning on safety and performance.

2 Learning Objectives

By completing this project, you will:

- Formulate control-affine systems with linearly parameterized uncertainties.
- Derive and implement aCLF-based QP controllers with stability-based parameter adaptation.
- Understand and apply concurrent learning to accelerate parameter convergence without relying solely on persistency of excitation.
- Evaluate the information content of recorded data via the minimum eigenvalue of the data matrix.
- Integrate adaptive estimates into CBF constraints to obtain aCBF-QP safety filters.
- Robustify aCBFs by adding margins that account for parameter error bounds, leading to raCBFs.
- Critically compare the safety guarantees and conservatism of aCBF vs. raCBF.

3 Project Tasks and Detailed Instructions

3.1 Task 1: System Modeling with Parametric Uncertainties

3.1.1 System Selection

Choose a MIMO nonlinear system that can be written in the control-affine form with linearly parameterized uncertainties:

$$\dot{x} = f_0(x) + F(x)\theta + g(x)u, \tag{1}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $\theta \in \Theta \subset \mathbb{R}^p$ is a vector of unknown constant parameters, $f_0 : \mathbb{R}^n \to \mathbb{R}^n$ and $F : \mathbb{R}^n \to \mathbb{R}^{n\times p}$ are known functions, and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ is the known input matrix. The output is not explicitly required unless trajectory tracking is considered.

3.1.2 Model Documentation

Provide:

- Explicit expressions for $f_0(x)$, $F(x)$, $g(x)$, and the true value of θ used in simulations.
- Physical units and typical values.
- Verification that the system is controllable and that the uncertainties enter in a linearly parameterized manner.
- A Simulink model `Uncertain_System.slx` that accepts u and θ and outputs $\dot{x}$.

Deliverable for Task 1: Simulink model and a report section with complete system description, parameter values, and validation (e.g., open-loop response with known θ).

3.2 Task 2: Adaptive Control Lyapunov Function (aCLF) Design

3.2.1 CLF and Regressor Construction

Select a Control Lyapunov Function $V(x)$ for the nominal (or certainty-equivalent) system. Compute:

$$\omega(x) = \frac{\partial V}{\partial x} f_0(x), \tag{2}$$

$$\phi(x) = \left(\frac{\partial V}{\partial x} F(x)\right)^\top \in \mathbb{R}^p, \tag{3}$$

$$L_g V(x) = \frac{\partial V}{\partial x} g(x) \in \mathbb{R}^{1\times m}. \tag{4}$$

The derivative of V along trajectories is

$$\dot{V}(x,u,\theta) = \omega(x) + \phi(x)^\top \theta + L_g V(x) u. \tag{5}$$

3.2.2 Certainty-Equivalence aCLF QP

Given a current parameter estimate $\hat{\theta}(t)$, the **aCLF constraint** is

$$\omega(x) + \phi(x)^\top \hat{\theta} + L_g V(x) u \le -\gamma V(x), \tag{6}$$

where $\gamma > 0$. (Again, note that this formulation is partially different with that introduced in the class.) The **minimum-norm controller** is obtained by solving

$$\begin{aligned} \min_{u\in\mathbb{R}^m} \quad & \|u\|^2 \\ \text{s.t.} \quad & \omega(x) + \phi(x)^\top \hat{\theta} + L_g V(x) u \le -\gamma V(x). \end{aligned} \tag{7}$$

This is a QP with $H = I_m$, $f = 0$, $A = L_g V(x)$, $b = -\omega(x) - \phi(x)^\top \hat{\theta} - \gamma V(x)$.

3.2.3 Parameter Update Law

After solving the QP, compute the **constraint violation**

$$e(x, u, \hat{\theta}) = \omega(x) + \phi(x)^\top \hat{\theta} + L_g V(x) u + \gamma V(x). \tag{8}$$

Note that if the QP is feasible, $e \leq 0$; equality holds when the constraint is active. The parameter estimate is updated using:

$$\dot{\hat{\theta}} = \Gamma \phi(x), \tag{9}$$

where $\Gamma = \Gamma^\top \succ 0$ is the adaptation gain matrix.

3.2.4 Simulation Requirements

- Implement the aCLF-QP controller in a MATLAB function `u = aCLF_Controller(x, theta_hat, gamma, Gamma)`.
- In Simulink, use an integrator for $\hat{\theta}$ with the update law (9).
- Simulate stabilization from several initial conditions and with initial parameter guesses $\hat{\theta}(0)$ different from the true θ.
- Plot the evolution of $\|x(t)\|$, $\|u(t)\|$, and the parameter estimation error $\|\tilde{\theta}(t)\|$.
- **Without persistent excitation**, parameter convergence may be slow or partial; document this behavior.

Deliverable for Task 2: MATLAB functions, Simulink model `aCLF_ClosedLoop.slx`, and a report section with aCLF derivation, QP formulation, update law derivation, and baseline adaptive performance plots.

3.3 Task 3: Concurrent Learning for Parameter Estimation

The slow convergence of the update law (9) motivates the use of **concurrent learning (CL)**. The idea is to augment the update law with information from a history stack of recorded data points (x_j, u_j), $j = 1, \ldots, N$.

3.3.1 History Stack Management

At runtime, select and store a set of points x_j that are sufficiently spaced and informative. For each stored point, compute L_j and Y_j.

3.3.2 Concurrent Learning Update Law

The concurrent learning update law is

$$\dot{\hat{\theta}} = \Gamma \left[\phi(x) + \gamma_c \sum_{j=1}^{N} L_j^T (Y_j - L_j \hat{\theta}) \right], \tag{10}$$

where $\gamma_c > 0$ is a learning gain.

3.3.3 Minimum Eigenvalue Analysis

The convergence rate of the parameter estimates is closely related to the matrix

$$\Phi = \sum_{j=1}^{N} L_j^T L_j \in \mathbb{R}^{p\times p}. \tag{11}$$

If the minimum eigenvalue $\lambda_{\min}(\Phi)$ is large, the recorded data provides rich information and accelerates convergence. Your analysis should:

- Compute and plot $\lambda_{\min}(\Phi(t))$ as the history stack is built during the simulation.
- Correlate parameter convergence speed with $\lambda_{\min}(\Phi)$.
- Compare the evolution of $\|\tilde{\theta}(t)\|$ with and without the CL term.

3.3.4 Simulation Requirements

- Implement a history stack of size $N \geq p$ (at least the number of unknown parameters). Describe the criteria for adding/replacing points (e.g., based on maximizing the determinant or minimum eigenvalue).
- Run the aCLF with CL and generate the following plots:
  1. Parameter estimates $\hat{\theta}(t)$ vs. true θ.
  2. Norm of estimation error $\|\tilde{\theta}(t)\|$.
  3. Minimum eigenvalue $\lambda_{\min}(\Phi(t))$.
  4. Comparison of $\|\tilde{\theta}(t)\|$ between stability-only (Task 2) and CL-based updates.

Deliverable for Task 3: Updated `aCLF_Controller_CL.m`, Simulink model, and a report section discussing the CL algorithm, history stack management, eigenvalue analysis, and comparative plots.

3.4 Task 4: Adaptive Control Barrier Function (aCBF) Design

Now consider a safety constraint $h(x) \geq 0$ as in Mini-Project 1, but the dynamics contain the unknown parameter θ. The derivative of h is

$$\dot{h}(x,u,\theta) = L_f h(x) + \underbrace{\big(L_F h(x)\big)\theta}_{\text{uncertain}} + L_g h(x) u. \tag{12}$$

Define the regressor for the barrier:

$$\phi_h(x) = \big(L_F h(x)\big)^\top \in \mathbb{R}^p. \tag{13}$$

3.4.1 Certainty-Equivalence aCBF Constraint

Using a parameter estimate $\hat{\theta}$, the **aCBF constraint** is

$$L_f h(x) + \phi_h(x)^\top \hat{\theta} + L_g h(x) u \geq -\gamma_B h(x), \tag{14}$$

with $\gamma_B > 0$.

3.4.2 aCBF Safety Filter QP

Given a nominal controller u_{nom} (e.g., the aCLF controller), the safety filter solves

$$\begin{aligned} \min_{u \in \mathbb{R}^m} \quad & \|u - u_{\text{nom}}\|^2 \\ \text{s.t.} \quad & L_f h(x) + \phi_h(x)^\top \hat{\theta} + L_g h(x) u \geq -\gamma_B h(x). \end{aligned} \tag{15}$$

Note that this QP requires another estimation of $\hat{\theta}$ appropriate for safety, while u_{nom} used the previous estimation of $\hat{\theta}$.

3.4.3 Simulation Requirements

- Implement a safety-critical scenario where the nominal aCLF controller would violate $h(x) \geq 0$.
- Show that the aCBF filter intervenes and maintains safety, even though the parameter estimates are initially inaccurate.
- Observe and discuss any temporary constraint violations when $\hat{\theta}$ is far from the true value.
- Implement multiple barrier functions if appropriate (e.g., joint limits and obstacle avoidance).

Deliverable for Task 4: MATLAB functions `aCBF_Constraint.m`, `aCBF_SafetyFilter.m`, Simulink model `aCBF_ClosedLoop.slx`, and a report section documenting the aCBF formulation and simulation results.

3.5 Task 5: Robust Adaptive Control Barrier Function (raCBF) Design

To robustly guarantee safety at all times, we design a **Robust Adaptive CBF (raCBF)**.

3.5.1 Robustness Margin

Assume that a known bound on the parameter error is available: $\|\tilde{\theta}\| \leq \bar{\theta}$ (e.g., enforced via parameter projection or from initial knowledge). Then we have

$$|\phi_h(x)^\top \tilde{\theta}| \leq \|\phi_h(x)\| \, \bar{\theta}. \tag{16}$$

The worst-case effect of the uncertainty on $\dot{h}$ is therefore a reduction by at most $\|\phi_h(x)\|\bar{\theta}$. To compensate, we tighten the CBF constraint:

$$L_f h(x) + \phi_h(x)^\top \hat{\theta} + L_g h(x) u \geq -\gamma_B h(x) + \|\phi_h(x)\|\,\bar{\theta}. \tag{17}$$

This ensures that even under the worst-case parameter mismatch, the true derivative satisfies $\dot{h} \geq -\gamma_B h(x)$.

3.5.2 raCBF QP Formulation

$$\begin{aligned} \min_{u \in \mathbb{R}^m} \quad & \|u - u_{\text{nom}}\|^2 \\ \text{s.t.} \quad & L_f h(x) + \phi_h(x)^\top \hat{\theta} + L_g h(x) u \geq -\gamma_B h(x) + \|\phi_h(x)\|\,\bar{\theta}. \end{aligned} \tag{18}$$

3.5.3 Comparison with aCBF

- Simulate the **same** safety-critical scenario using both the aCBF (Task 4) and the raCBF (Task 5).
- Compare the following:
  1. Minimal value of $h(x(t))$ (how close the system gets to violating safety).
  2. Control effort and conservatism (the raCBF may apply more aggressive corrections).
  3. Feasibility of the QP over time.
- Discuss the trade-off: raCBF provides guaranteed safety but is more conservative, whereas aCBF is less restrictive but may fail if parameter error is large.

Deliverable for Task 5: MATLAB function `raCBF_SafetyFilter.m`, Simulink model `raCBF_ClosedLoop.slx`, and a report section with the raCBF derivation, comparison plots, and discussion.

3.6 Task 6: Comprehensive Analysis and Discussion

Your handwritten report must contain a thorough discussion including:

- **Effect of Concurrent Learning**: Quantify how CL improves parameter convergence and how this impacts both stabilization (aCLF) and safety (aCBF/raCBF).
- **Eigenvalue Analysis**: Interpret the time evolution of $\lambda_{\min}(\Phi)$ and relate it to the excitation richness of the trajectory.
- **aCBF vs. raCBF**: Under what conditions does the aCBF fail to keep the system safe? How much conservatism does the raCBF introduce? Provide simulation evidence.
- **Influence of $\bar{\theta}$**: For the raCBF, discuss the choice of the error bound. What happens if the bound is too small (violation) or too large (overly conservative)?
- **Practical Considerations**: Robustness to measurement noise, selection of history stack points, and computational aspects of the QP.

4 Deliverables Summary

Table 1: Project Deliverables

Deliverable	Format	Details
Handwritten Report	PDF scan	Maximum 10 pages, single-sided, legible.
Video Presentation	MP4 file	5 minutes (strict), explaining code and key results.
MATLAB/Simulink Code	`.m` and `.slx` files	All files listed below, archived with folder structure.

5 Code Organization

All simulations must be executable from a single `Main.m` script. The required file structure is:

```
Main.m                          # Master script
System_Parameters.m             # True parameters, bounds, etc.
Dynamics.m                      # f0(x), F(x), g(x) for uncertain system
Regressors.m                    # Computes phi(x), omega(x), LgV, etc.
aCLF_Controller.m               # Task 2: aCLF QP (stability-based update)
aCLF_Controller_CL.m            # Task 3: aCLF with concurrent learning
aCBF_SafetyFilter.m             # Task 4: aCBF safety filter
raCBF_SafetyFilter.m            # Task 5: robust aCBF safety filter
HistoryStack.m                  # Management of recorded data
Plot_Results.m                  # Generate all required plots
Uncertain_System.slx            # Simulink model of open-loop system
aCLF_ClosedLoop.slx             # Closed-loop with adaptive CLF
aCLF_CL_ClosedLoop.slx          # Closed-loop with CL
aCBF_ClosedLoop.slx             # Closed-loop with aCBF safety filter
raCBF_ClosedLoop.slx            # Closed-loop with raCBF safety filter
README.txt                      # Instructions to run
```

6 Report Formatting Guidelines

Your handwritten report should be organized as follows:

```
1. Introduction
2. System Description with Parametric Uncertainties
3. Adaptive CLF Design and Baseline Performance
4. Concurrent Learning for Parameter Estimation
```

```
   4.1 Algorithm Description and History Stack Management
   4.2 Convergence Analysis and Minimum Eigenvalue
   4.3 Comparison: Stability-only vs. Concurrent Learning
5. Adaptive CBF Design and Safety Filtering
6. Robust Adaptive CBF: Design and Comparison with aCBF
7. Comprehensive Discussion
8. Conclusions
9. References (optional)
```

7 References

1. Cohen, M., & Belta, C. (2023). *Adaptive and Learning-Based Control of Safety-Critical Systems*. Springer.

2. Chowdhary, G., Mühlegg, M., & Johnson, E. (2013). Exponential parameter and tracking error convergence guarantees for adaptive controllers without persistency of excitation. *International Journal of Control*, 87(8), 1583–1603.

3. Ames, A. D., Xu, X., Grizzle, J. W., & Tabuada, P. (2017). Control barrier function based quadratic programs for safety critical systems. *IEEE TAC*, 62(8), 3861–3876.

4. Taylor, A. J., Dorobantu, V. D., Leung, K., & Ames, A. D. (2020). Episodic Learning with Control Lyapunov Functions for Uncertain Robotic Systems. *IEEE/RSJ IROS*.

5. Isaly, A. D., Patil, O. S., Sanfelice, R. G., & Dixon, W. E. (2021). Adaptive safety with multiple barrier functions using integral concurrent learning. *ACC*.