Universidad
Rey Juan Carlos

TESIS DOCTORAL

Iterative Learning Control for Precise Aircraft Trajectory Tracking

Autora:

Almudena José Buelta Méndez

Directores:

Alberto Olivares González

Ernesto Staffetti Giammaria

Programa de Doctorado en Tecnologías de la Información y las Comunicaciones

Escuela Internacional de Doctorado

2022

To my family.

Resumen extendido

Antecedentes

A través de distintas iniciativas como la europea Single European Sky ATM Research (SESAR) y la norteamericana Next Generation Air Transportation System (NextGen), los principales responsables de la gestión global de la navegación aérea están haciendo un esfuerzo de modernización para adaptarse al aumento del tráfico que previsiblemente afectará al espacio aéreo en los próximos años. Estas iniciativas pretenden posibilitar un nuevo paradigma en la gestión del tráfico aéreo (ATM, por sus siglas en inglés) consistente en operaciones basadas en trayectorias (TBO, por sus siglas en inglés). Este nuevo paradigma implica la optimización de las trayectorias que deben seguir las aeronaves según las preferencias de los agentes implicados y teniendo en cuenta tanto las tres dimensiones espaciales como la dimensión temporal. Por tanto, en la ejecución de estas trayectorias en cuatro dimensiones (4D), un retraso temporal también se considera una desviación con respecto a la trayectoria planificada, al igual que una imprecisión en el nivel de vuelo o en la posición horizontal de la aeronave.

La predictibilidad de las trayectorias —es decir, la correspondencia entre la trayectoria planificada y la volada— es un aspecto clave en la implementación de las TBO para el que se requiere gran precisión en el seguimiento de las trayectorias por parte de las aeronaves. Una mayor predictibilidad de las trayectorias seguidas por las aeronaves supone una mejor sincronización del tráfico aéreo, lo que resulta en una mejora en la seguridad, eficiencia y capacidad del sistema ATM. También permite reducir las alteraciones en la trayectoria seguida que ocurren durante el vuelo, lo que implica menos costes para las aerolíneas, además de menos emisiones y, por tanto, un menor impacto ambiental. Sin embargo, debido a múltiples factores aleatorios, como errores en la predicción del viento, la temperatura u otros factores atmosféricos (principalmente tormentas), siempre va a existir un nivel de incertidumbre que dificulte o incluso imposibilite el seguimiento de las trayectorias con la precisión requerida, lo que provoca desviaciones con respecto a la trayectoria planificada que no pueden ser compensadas por los controladores habituales instalados en las aeronaves, ya que, generalmente, usan un esquema de control por retroalimentación que solo les permite reaccionar a las perturbaciones cuando ya se han producido.

La presente tesis doctoral aborda este problema utilizando un modelo de control inteligente que proporciona un seguimiento preciso de las trayectorias en espacios con alta densidad de tráfico aéreo. En particular, el control por aprendizaje iterativo (ILC, por sus siglas en inglés) es capaz de mejorar la precisión en la trayectoria volada por una aeronave a partir de las desviaciones experimentadas por las aeronaves que previamente han efectuado una trayectoria similar. La predictibilidad de las trayectorias 4D es especialmente importante en el área de control terminal (TMA, por sus siglas en inglés), que es el espacio aéreo controlado alrededor de uno o varios aeródromos. Debido al elevado número de rutas de llegada y salida que contienen, las TMA suelen ser zonas de alta densidad de tráfico aéreo y, por tanto, son particularmente sensibles a las desviaciones espaciales o temporales de las trayectorias 4D. Además, las TMA muy concurridas son el escenario ideal para implementar el ILC, puesto que una gran cantidad de aeronaves siguen las mismas rutas con separaciones temporales breves y, en consecuencia, son más susceptibles de experimentar perturbaciones similares.

Para la aplicación del ILC al seguimiento preciso de trayectorias de aeronaves, es necesario que la información acerca de las trayectorias planificadas y efectuadas por las aeronaves previas esté disponible. Esta información será accesible en los próximos años gracias a la implementación del sistema SWIM por parte de la Organización de Aviación Civil Internacional (OACI). Este sistema comprende los estándares, infraestructuras y órganos de gobierno que permitirán el intercambio de información entre los diferentes agentes implicados en el sistema ATM, incluyendo datos de vuelo, meteorológicos y aeronáuticos.

Objetivos

El objetivo principal de esta tesis es el desarrollo de técnicas de control por aprendizaje iterativo aplicadas al seguimiento preciso de las trayectorias de aeronaves comerciales. Para su consecución, se han alcanzado los siguientes objetivos intermedios:

- Implementación de un esquema ILC basado en la optimización que mejora la precisión en el seguimiento de las trayectorias planificadas. De esta manera, una aeronave que tenga que seguir cierta trayectoria utilizará las desviaciones que ya hayan experimentado otras aeronaves idénticas que también hayan volado esa misma trayectoria para corregir las perturbaciones recurrentes que afecten a su vuelo. Estas perturbaciones pueden deberse a factores meteorológicos, como el viento o el modelo atmosférico, o a errores en el modelo del avión.
- Desarrollo de métodos para la estimación de las desviaciones experimentadas por las aeronaves con respecto a la trayectoria planificada y la predicción de dichas desviaciones en la siguiente iteración.
- Implementación de un controlador adaptativo que sirva como sistema aumentativo del controlador base de tipo proporcional-integral diseñado para simular el controlador instalado en la aeronave dedicado al seguimiento de trayectorias. Este controlador adaptativo fuerza a la aeronave a comportarse de manera similar a la de un modelo de referencia,

con lo que el modelo dinámico tiene un comportamiento redundante aunque este cambie entre iteraciones.

- Combinación de las técnicas de ILC y control adaptativo para lograr un esquema de aprendizaje transferido con el fin de mejorar la precisión en el seguimiento de las trayectorias planificadas aunque las aeronaves que ejecuten cada iteración tengan modelos dinámicos distintos.

Metodología

Debido a la imposibilidad de ensayar la metodología de aprendizaje descrita en aeronaves reales, se ha diseñado un simulador de vuelo sobre el que realizar los experimentos. Este simulador, desarrollado en MATLAB/Simulink, incluye, además del modelo dinámico de la aeronave y su controlador, la envolvente de vuelo, las distintas restricciones operativas asociadas a la trayectoria planificada, un modelo de viento horizontal y el modelo atmosférico.

Las trayectorias planificadas que deben seguir las aeronaves se han generado utilizando técnicas de control óptimo para las que se ha tenido en cuenta el modelo dinámico de la aeronave con el fin de garantizar su viabilidad. Los problemas de control óptimo se han resuelto utilizando el software DIDO, que se basa en el método pseudoespectral para calcular no solo la trayectoria óptima, sino también los indicadores necesarios para verificar la optimalidad de los resultados numéricos mediante el principio de Pontryagin.

Los métodos empleados para la estimación y predicción de las desviaciones de las aeronaves son el filtro de Kalman y la regresión con procesos gaussianos. Esta última ofrece resultados precisos incluso ante variaciones no lineales de las perturbaciones sin necesidad de conocer datos previos sobre su comportamiento.

Una vez estimadas las perturbaciones que afectan al vuelo de una aeronave, el ILC calcula, para la siguiente aeronave, un nuevo input de control, en el caso del ILC directo, o una nueva trayectoria de referencia que sirve de entrada al controlador de la aeronave, en el caso del ILC indirecto. En ambos casos, para obtener el input de control o la trayectoria de referencia que compensen las perturbaciones redundantes, es necesario resolver un problema de optimización convexo que tenga en cuenta las restricciones del modelo. En esta tesis se ha utilizado el software CPLEX para resolver dicho problema de optimización.

Para conseguir un comportamiento similar entre aeronaves con sistemas dinámicos distintos, se ha aumentado el controlador base del avión con un controlador adaptativo por modelo de referencia (MRAC, por sus siglas en inglés) que minimiza la diferencia entre la trayectoria seguida por la aeronave y la seguida por un modelo de referencia a partir de leyes adaptativas basadas en la teoría de estabilidad de Lyapunov. Si las aeronaves que deben volar una misma trayectoria planificada comparten el mismo modelo de referencia, procederán de manera similar, por lo que, con un método de control combinado ILC-MRAC, es posible transferir la trayectoria aprendida por una aeronave a otra distinta y compensar así las perturbaciones redundantes que afectan al vuelo.

Resultados

La efectividad de los métodos propuestos se ha demostrado a través de varios experimentos numéricos que simulan el comportamiento de las aeronaves a lo largo de varias iteraciones aplicando el ILC. Estos experimentos se pueden dividir en dos grupos: en el primero, se ha supuesto que todas las iteraciones las realizan aeronaves idénticas y, en el segundo, que cada vuelo puede ser ejecutado por aeronaves con modelos dinámicos distintos.

En el primer grupo, los experimentos se han llevado a cabo con un simulador de vuelo de un Airbus A320-231 realizando operaciones de ascenso y descenso continuo. Los resultados muestran que, en ambos casos, se consigue un seguimiento preciso de las trayectorias en tan solo tres iteraciones, tanto con el esquema de ILC directo como con el indirecto. Además, comparando las estimaciones y predicciones del error obtenidas mediante el filtro de Kalman y la regresión con procesos gaussianos, los experimentos muestran que esta última ofrece mejores resultados si el filtro de Kalman no está bien ajustado y resultados comparables si los parámetros del filtro de Kalman son los adecuados, lo que requiere información previa acerca del comportamiento de las perturbaciones del modelo.

En el segundo grupo, los resultados obtenidos con el MRAC al variar el modelo dinámico de un Airbus A320-231 muestran que, pese a estas variaciones, sigue una trayectoria similar a la del modelo de referencia. Esto permite que el método combinado ILC-MRAC, en el que las distintas iteraciones son ejecutadas por simuladores de vuelo de los Airbus A320-231, Boeing 767-300ER y Embraer ERJ 190-200 IGW, ofrezca buenos resultados en pocas iteraciones, tanto si una de las aeronaves realiza un aprendizaje previo a lo largo de varias iteraciones antes de transferir este aprendizaje a otra aeronave distinta, como si la transferencia se produce desde la primera iteración. Además, este esquema presenta variaciones pequeñas en el seguimiento de la trayectoria entre las distintas iteraciones.

Conclusiones

En esta tesis se han aplicado diferentes estrategias de control al problema del seguimiento preciso de trayectorias de aeronaves comerciales cuyo factor común es el uso del control por aprendizaje iterativo. Se ha empleado un sistema de ILC basado en la optimización —tanto en la estimación de las perturbaciones como en el cálculo del input de control o trayectoria de referencia que compensa estas perturbaciones para la siguiente aeronave— con el fin de mejorar la precisión en el seguimiento de la trayectoria planificada teniendo en cuenta las desviaciones sufridas por aeronaves que previamente han volado esa misma trayectoria. Este método es especialmente apropiado para el seguimiento de trayectorias en 4D en los procedimientos de salida y de llegada de los aeropuertos congestionados, pues se puede suponer que las perturbaciones, principalmente debidas a la meteorología, son similares entre vuelos consecutivos. El ILC requiere que no haya variaciones en los sistemas dinámicos que llevan a cabo cada una de las repeticiones. Para que los vuelos correspondientes a cada iteración puedan ser ejecutados por aeronaves distintas, se ha investigado un método que combina el control adaptativo con el ILC. Los resultados de los experimentos numéricos muestran la eficiencia de las técnicas

propuestas y la viabilidad de su aplicación en aeronaves comerciales, ya que no sustituyen, sino que complementan a los controladores ya instalados en dichas aeronaves.

Abstract

In this thesis, an iterative learning control method to improve precision in commercial aircraft trajectory tracking is proposed. Given a four-dimensional trajectory to be followed, the proposed method improves the system performance in following the trajectory using the spatial and temporal deviations suffered by previous aircraft to anticipate recurring disturbances, represented by weather conditions and unmodeled system dynamics, and proactively compensate for them, so that the subsequent aircraft intending to fly the same planned trajectory will follow it with greater precision than the previous ones. The iterative learning control algorithm is divided into two steps: estimation of the disturbances and model errors affecting the tracking performance, and update of the control inputs. Two different estimation methods are tested and compared: a Kalman filter, which requires careful tuning and relies on prior knowledge about the dynamics of disturbances and model errors, which are assumed to be linear, and a recursive Gaussian process regression, which estimates and predicts disturbances and model errors at a low computational cost without the linear assumption, and without relying on prior knowledge about their dynamics. As regards to the update step, both direct and indirect iterative learning control approaches are considered. Whereas the former directly updates the control input to be used in the following iteration, the latter generates a new reference signal to be fed into the underlying feedback controller of the aircraft to execute the next iteration and therefore is nonintrusive with respect to the avionics of the aircraft. Both update strategies rely on a nominal dynamic model of the aircraft, in which input and state constraints can be explicitly considered, but require the system dynamics to be repetition-invariant. To overcome this limitation, a multi-aircraft transfer learning strategy is proposed, which enables knowledge about learned trajectories to be transferred among dynamically different aircraft at each iteration. For this purpose, the baseline controller of the aircraft is augmented with a model reference adaptive controller, which forces the aircraft to behave close to a given reference model. The obtained results show a significant reduction of the trajectory tracking error in few iterations, proving the effectiveness of the iterative learning control method applied to commercial aircraft trajectory tracking. The proposed method is suitable to be used in busy terminal maneuvering areas, in which the time-based separation between aircraft is short enough to expect similar weather conditions. Higher precision in trajectory tracking implies an increase in the predictability of aircraft trajectories and an improvement in the efficiency of

the air traffic management system. Airlines can also benefit from this higher predictability by reducing the number of alterations when following their designed trajectories, which entails a reduction of costs and emissions.

Acknowledgements

Quiero comenzar estas líneas con mi más sincero agradecimiento a mis directores de tesis: Ernesto Staffetti y Alberto Olivares. Gracias de corazón por todo el tiempo y esfuerzo que habéis invertido en este trabajo, por tener siempre buenos consejos que ofrecerme y por haber sabido guiarme, ayudándome a encontrar la mejor solución posible a los retos que han ido apareciendo durante el camino. Me siento muy afortunada por haber tenido unos directores de tesis tan atentos, predispuestos a ayudar en todo lo que ha estado en vuestras manos y, además, siempre con una sonrisa. Gracias por vuestra profesionalidad, pero también por las charlas distendidas y vuestras palabras de ánimo. Gracias por todo.

Gracias también a mis compañeros del departamento por haberme acogido y acompañado durante estos años. En especial, a mis compañeros, amigos y vecinos Ana, Edu y Álex por estar siempre dispuestos a echarme una mano, a poder ser con una cerveza en la otra. Gracias, María, por acordarte de mí cuando todo esto empezó y por haber sido mi ejemplo y apoyo en esta fase final. Gracias, Mihaela, por compartir conmigo grandes momentos de excursiones y de puzzles. Además, quiero dar las gracias a mis compañeros del área Jorge, Marius y Jaime, y también a Óscar, Rebeca, Antonio y tantos otros. Sin vosotros esto no habría sido lo mismo.

I would like to thank Dr. Lyudmila Mihaylova, who supervised my stay at the University of Sheffield. Thank you for your time and your valuable contributions to this thesis. Also, many thanks to Waqas for all your help and good advice. Thank you to Jocelyn, Chen and Maddie for your pleasant chats during the breaks and for taking me to the best pubs of the city.

Tengo la inmensa suerte de contar con grandes amigos que, sin saberlo, me han ayudado a llegar hasta aquí. A las de siempre y a los que, a lo largo de los años, han ido apareciendo en mi vida para quedarse. A todos vosotros, mil gracias.

Un reconocimiento especial merecen mis padres y mi hermana. Sois la mejor familia que podría desear y no hay palabras en el mundo que expresen lo mucho que os quiero y lo agradecida que estoy por todo el apoyo y cariño que me habéis dado siempre. Gracias, Nieves, porque para mí también eres parte de mi familia. Por último, quiero dar las gracias a la persona que más cerca ha estado de mí durante todo este camino: Melón, gracias por tu ayuda, cariño, apoyo y comprensión, y por hacer que todo esto haya sido más fácil.

Contents

List of Figures

List of Tables

List of Acronyms

4D	Four-Dimensional
AIP	Spanish Aeronautical Information Publication
ATC	Air Traffic Control
ATM	Air Traffic Management
BADA	Base of Aircraft Data
CAS	Calibrated Airspeed
CCO	Continuous Climb Operation
CDO	Continuous Descent Operation
DOF	Degree of Freedom
GP	Gaussian Process
GPR	Gaussian Process Regression
ICAO	International Civil Aviation Organization
IILC	Indirect Iterative Learning Control
ILC	Iterative Learning Control
ISA	International Standard Atmosphere
LQR	Linear Quadratic Regulator
MIMO	Multi-Input Multi-Output
MRAC	Model Reference Adaptive Control
NextGen	Next Generation Air Transportation System

OCP	Optimal Control Problem
OPD	Optimized Profile Descents
PI	Proportional Integral
PID	Proportional Integral Derivative
RMSE	Root Mean Square Error
SESAR	Single European Sky ATM Research
SID	Standard Instrument Departure
STAR	Standard Terminal Arrival Route
SWIM	System Wide Information Management
TBO	Trajectory-Based Operations
TMA	Terminal Maneuvering Area
TOD	Top-Of-Descent
UAV	Unmanned Aerial Vehicle

A smart person learns from their mistakes, but a truly wise person learns from the mistakes of others.

1

Introduction

1.1 Goal and motivation

The global air navigation system is adapting to the expected increase in air traffic demand through international initiatives like the Single European Sky ATM Research (SESAR)[1] and the Next Generation Air Transportation System (NextGen)[2], whose modernization efforts are enabling a new Air Traffic Management (ATM) paradigm built on Trajectory-Based Operations (TBO), which allow for optimized trajectories based on the stakeholders' preferences and priorities. The TBO concept is based on Four-Dimensional (4D) aircraft trajectories, which consist of the three spatial dimensions and the additional component of time, meaning that any delay is considered a deviation of the trajectory as much as a level change or a change of the horizontal position. Trajectory predictability, namely the correspondence between the planned and flown trajectories, is key to the implementation of TBOs, which require high precision in aircraft trajectory tracking. Higher trajectory predictability implies better traffic synchronization, which results in an improvement in the safety, efficiency, and capacity of the ATM system. Moreover, higher trajectory predictability, by reducing the number of alterations when following the planned trajectories, entails a reduction of costs and emissions, thus lessening the environmental impact of commercial aviation. However, due to multiple random factors, such as wind and temperature forecast errors or unpredictable weather events (mainly storms), some level of uncertainty remains, hindering precision and resulting in deviations from the planned trajectory that can neither be predicted nor compensated by usual aircraft trajectory tracking controllers, which, in general, use a feedback control scheme and react to disturbances as they occur [1, Chap. 1].

[1] https://www.sesarju.eu/
[2] https://www.faa.gov/nextgen/

This thesis addresses this problem by using an intelligent control paradigm for precise aircraft trajectory tracking in air spaces with high traffic density. More specifically, the Iterative Learning Control (ILC) scheme is employed, which is able to improve the precision of the aircraft in following the trajectories taking into account the deviations experienced by previous flights. A schematic representation of the idea is shown in Figure 1.1. Improving predictability of 4D trajectories is of special importance in the Terminal Maneuvering Area (TMA), which is the controlled airspace that surrounds one or more airports. Due to the numerous arrival and departure routes TMAs contain, they generally have a high density and are therefore more sensitive to 4D trajectories' spatial or temporal deviations. Furthermore, busy TMAs represent the ideal environment to implement ILC, since many aircraft follow the same routes with short temporal separation and therefore are likely to be subject to similar recurrent disturbances.

The application of ILC to precise aircraft trajectory tracking requires information about the intended and actually flown trajectories of previous aircraft in the relevant airspace. This information is assumed here to be available, and it will indeed become accessible with the implementation of the System Wide Information Management (SWIM). This project is part of the International Civil Aviation Organization (ICAO) Global Air Navigation Plan [2, Appx. 2] and consists of the standards, infrastructure, and governance enabling information exchange among the different ATM stakeholders, including aeronautical, weather, and flight data.

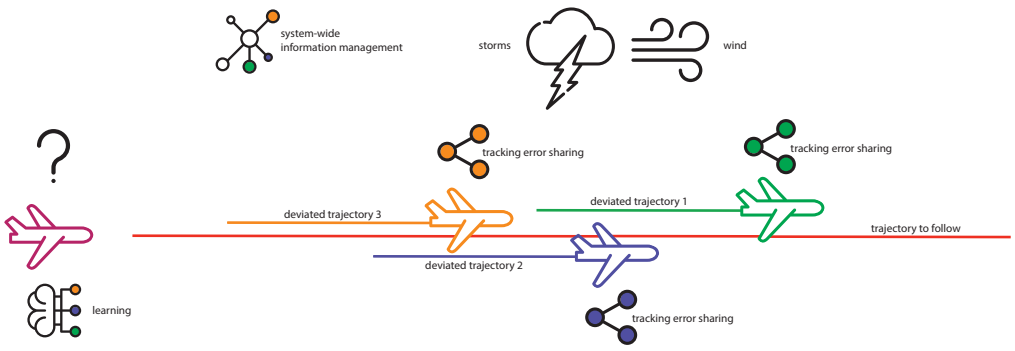


Figure 1.1: Schematic representation of the research idea developed in this thesis. Several aircraft intend to follow the same trajectory with short time separation under the effects of weather disturbances, causing them to experience deviations from the planned trajectory. Tracking errors are shared through a system-wide network with other aircraft and used to increase precision in trajectory tracking by means of ILC.

1.2 Contributions

The aim of this thesis is to apply iterative learning control to achieve high overall precision in commercial aircraft trajectory tracking in the presence of unknown disturbances, measurement noise, model uncertainties and even when the dynamic model of the aircraft changes along the iterations.

More specifically, an optimization-based ILC scheme is applied to precise trajectory tracking for commercial fixed-wing aircraft flying in realistic operational scenarios, proving the effectiveness of this approach to compensate for disturbances on these systems in the presence of operational constraints despite being much larger, less maneuverable, and having slower control response than rotary-wing small Unmanned Aerial Vehicles (UAVs), the only aerial systems in which the ILC method had already been tested.

This dissertation shows that precision in aircraft trajectory tracking can be improved by direct feed-forward adaptation of the control input in continuous climb and descent trajectories. The corresponding results have been published in the proceedings of the following international conferences:

- Buelta, A., Olivares, A., and Staffetti, E., Iterative Learning Control for Precise Aircraft Trajectory Tracking in Continuous Climb Operations, *Thirteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2019)*, 2019 [3].
- Buelta, A., Olivares, A., and Staffetti, E., Iterative Learning Control for Precise Aircraft Trajectory Tracking in Continuous Descent Approaches, *Eighth European Conference for Aeronautics and Aerospace Sciences (EUCASS)*, 2019 [4].

Additionally, a specific ILC scheme, the Indirect Iterative Learning Control (IILC), is designed, which is suitable to be implemented on commercial aircraft, since it allows for repetitive disturbances' corrections without interfering with the aircraft's existing trajectory tracking controller by simply generating, at each iteration, a new reference trajectory to be followed by the aircraft instead of a new control input. Therefore, the information about tracking errors and disturbances acquired in the previous iterations can be encapsulated into this new reference trajectory, which will be referred to as the learned reference trajectory. This means that the IILC controller can be physically implemented on a supplementary system, without changing the avionics of the aircraft. This represents a great advantage for airlines interested in adopting this technology. The results of this research activity have been published in the following journal article:

- Buelta, A., Olivares, A., and Staffetti, E., Iterative Learning Control for Precise Aircraft Trajectory Tracking in Continuous Climb and Descent Operations, *IEEE Transactions on Intelligent Transportation Systems*, 2021 [5].

The previously mentioned works assume only small linear changes between iterations in the disturbances affecting the aircraft and prior knowledge about their dynamics, which are assumed to be linear, and estimated using a Kalman filter. To overcome this limitation, a recursive Gaussian Process Regression (GPR) within the ILC framework is developed to provide estimation of nonlinear changing dynamics of the disturbances and model uncertainties affecting a flight without the need of any prior knowledge about their behavior, also predicting their values for the following aircraft performing the same trajectory. The computational cost required by the GPR is significantly reduced by using a recursive approach, in which, at each iteration, the oldest data are dismissed ant the newest observations are incorporated.

This makes it appealing for real-time control. The results of this research activity have been published in the journal article:

- Buelta, A., Olivares, A., Staffetti, E., Aftab, W., and Mihaylova, L., A Gaussian Process Iterative Learning Control for Aircraft Trajectory Tracking, *IEEE Transactions on Aerospace and Electronic Systems*, 2021 [6].

Finally, ILC requires the system dynamics to be repetition-invariant, which, in the context of this thesis, means that the same aircraft model must be employed at each iteration in tracking the planned trajectory. However, in realistic scenarios, consecutive flights along the same path are carried out by different aircraft, i.e. by different aircraft models or by the same aircraft model with different model parameters, e.g. different weight. To overcome this limitation, in this thesis, a strategy to transfer the knowledge acquired by an aircraft in following a trajectory to another aircraft to improve precision in tracking the same trajectory is devised. The key idea is that dynamically different aircraft equipped with an underlying Model Reference Adaptive Control (MRAC) are forced to behave close to the same reference model. Consequently, the knowledge acquired by an aircraft in a previous execution of the trajectory tracking task can be transferred, through a control strategy that combines ILC with MRAC, to another aircraft with a different dynamic model, to improve its precision in executing the same task.

Figure 1.2 shows the scheme of the ILC framework developed in this thesis to solve the trajectory tracking problem for commercial aircraft. The goal is to track a trajectory in the presence of unknown disturbances. This trajectory will be referred to as the desired trajectory. As shown in the figure, two main subsystems are considered: the flight simulator, which includes the aircraft model, the model of the external disturbances, and the underlying aircraft trajectory tracking controller; and the ILC, composed of an estimator of the disturbances and a control update block. The output generated by an aircraft is compared to the desired trajectory that it is intending to track, and the error is estimated by the ILC. In this thesis, two types of estimators are used: a Kalman filter and a GPR method. The estimated error is used to compute, depending on the ILC variant, a new control input (direct ILC) or a new reference trajectory (indirect ILC). In the first case (purple line in the figure), the new control input directly drives the next aircraft closer to the desired trajectory, compensating for the repetitive disturbances affecting the flight. In the second case (blue line in the figure), the new reference trajectory is fed into the aircraft trajectory tracking controller, which generates the control input. Three types of controllers are considered: a baseline controller (blue block), which is a Proportional Integral (PI) or a Linear Quadratic Regulator (LQR) PI, and an MRAC augmentation of an LQR PI controller (cyan block). The ultimate goal of the proposed MRAC-ILC framework is to transfer the knowledge acquired by a system in executing a task to a different system, so that, in the following iteration, the ILC can compensate for the recurrent disturbances even if the task is performed by a system with different model dynamics. In this thesis, this concept is referred to as transfer learning. In this case, the introduction of the adaptive control action allows to transfer the learned reference trajectory and disturbances to

a different aircraft for the following iteration, since the MRAC forces the aircraft to behave close to the same reference model, achieving repeatability in the performance of the aircraft in flying the planned trajectory. Repeatability in this context means that the system executes a task in a predefined way despite model inaccuracies and external disturbances.

To the best knowledge of the author, ILC has only been applied to commercial aircraft trajectory tracking in the works [3], [4], [5], and [6] mentioned above, and this thesis is the first to exploit the repeatable behaviour of the MRAC to transfer learning between different aircraft within the ILC framework.

This thesis has been partially supported by the grants number TRA2017-91203-EXP and RTI2018-098471-B-C33 of the Spanish Government. The general objective of the research project associated with the latter, entitled "Managing meteorological uncertainty for a more efficient air traffic system (MetATS)" is highly related to this thesis. Indeed, it is to improve the air traffic system, in terms of efficiency and safety, by integrating improved weather forecasts. Additionally, one of the specific objectives of the MetATS project is the study of learning control techniques, including iterative learning control and deep learning control, to improve the precision in tracking the planned aircraft trajectories in airspaces with high traffic density.

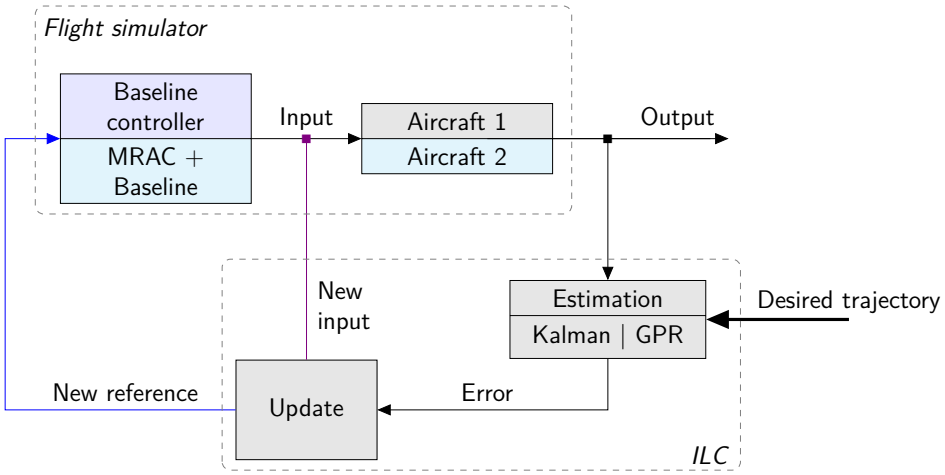


Figure 1.2: Scheme of the ILC framework. Given a trajectory to be followed, the ILC method estimates the disturbances and model errors and generates a new control input, in the direct ILC (purple line), or a new reference trajectory, in the indirect ILC (blue line). In the latter case, the new reference trajectory is fed to a baseline controller. Adaptive control allows the learned reference trajectory and estimated disturbances
to be transferred to a different aircraft (cyan blocks).

1.3 Thesis overview

This dissertation is composed of two parts. In the first part, the ILC method is applied to commercial aircraft trajectory tracking under the hypothesis that identical aircraft execute

each iteration, whereas in the second part, this hypothesis is removed and the possibility of applying the ILC algorithm when different aircraft execute each iteration is investigated.

1.3.1 Part I. Iterative learning control for aircraft trajectory tracking

The first part of the thesis comprises the research and results regarding an optimization-based ILC method and its application to commercial aircraft trajectory tracking.

In Chapter 2, the models used in this thesis are presented. They include the aircraft dynamics and operational constraints, the feedback controller design, and the atmospheric and wind models employed in the numerical simulations. Finally, the numerical optimal control method used for generating the reference trajectories is described.

In Chapter 3, the general direct and indirect ILC schemes are presented. The chapter focuses on improving the trajectory tracking performance over the iterations of a commercial aircraft subject to unknown disturbances. In the direct ILC case, precision is achieved over the iterations by pure feed-forward adaptation of the control input, whereas in the indirect approach, the ILC acts as an offline high-level controller to an underlying PI feedback controller, updating the reference trajectory rather than the control input. The experiments are carried out in a simulated environment in which identical aircraft perform continuous climb and descent operations.

The formulation of the GPR is described in Chapter 4, including the implementation of the recursive GPR with learning of the hyperparameters into the ILC algorithm. The experiments and numerical results provide a comparison between the aircraft performances achieved with different GPR techniques and the Kalman filter as estimators and predictors within the ILC algorithm.

1.3.2 Part II. Multi-aircraft transfer learning

As mentioned above, the possibility of applying the ILC paradigm to improve precision in aircraft trajectory tracking is investigated in the first part of the thesis. However, in realistic scenarios, additional drawbacks arise, since consecutive flights are usually carried out by different aircraft, so different dynamical systems perform each iteration, whereas the basic ILC scheme requires the system dynamics to be repetition-invariant. In the second part of this thesis, repeatability is achieved by using an underlying adaptive controller that copes with the changes in the system dynamics.

Chapter 5 presents the fundamental theory of direct MRAC, which forces the aircraft to behave close to a specified reference model. Then, an MRAC augmentation of an LQR PI controller is designed for a commercial aircraft and the tracking performance is tested when uncertainties are introduced in the model, achieving repeatability, even in the presence of these uncertainties.

Effective learning in the ILC framework depends on the accuracy of the system modeling; if there are any disturbances or model inaccuracies, the learning performance decreases. Chapter 6 presents a learning-based controller in which the ILC is used in combination with the MRAC, which makes the system behave in a repeatable, predefined way, remaining effective even when disturbances affect the system. Moreover, the numerical experiments show that the MRAC-ILC combination is able to transfer the learned reference trajectory to dynamically different aircraft achieving high-accuracy trajectory tracking.

Finally, in Chapter 7 some conclusions are drawn and possible future lines of research are outlined.

Part I

Iterative Learning Control for Aircraft Trajectory Tracking

2

Simulated Environment

Due to the obvious difficulty in implementing an ILC algorithm for accurate trajectory tracking using real commercial aircraft, the experiments whose results are reported in this thesis have been carried out on a realistic, simulated scenario built in MATLAB/Simulink, a proprietary programming platform developed by MathWorks[1] for numerical computation, in which dynamical systems can be modeled, simulated, and analyzed.

This simulated environment is structured in the following main blocks:

- A realistic flight simulator, which includes the aircraft model and perturbations affecting the flight, such as weather disturbances, model uncertainties, and measurement errors.
- An ILC controller that drives the aircraft to the desired trajectory, which has been previously designed.

In this chapter, the dynamic model of the aircraft used in the simulated environment is derived, as well as the atmospheric conditions and the wind model. The method to generate the desired trajectories to be followed by the aircraft is also described.

2.1 Flight simulator

The flight simulator is composed of a three Degree of Freedom (DOF) model of a commercial aircraft, an atmospheric model, and a wind model. Two sets of parameters and models are used throughout this thesis to show the robustness of the ILC controller to modeling errors and perturbations:

- The modeling equations, implemented in MATLAB to build the different algorithms, use the aircraft data obtained from the Base of Aircraft Data (BADA), and assume

[1] https://es.mathworks.com/

International Standard Atmosphere (ISA) conditions and no wind. BADA is an aircraft performance model developed and maintained by EUROCONTROL[2] with the collaboration of aircraft manufacturers and airlines, specifically designed for aircraft trajectory simulation and prediction in ATM research and development. It has two components: model specifications, which provide the theoretical models used to calculate aircraft performance parameters, and data sets, which give the aircraft-specific coefficients. More specifically, BADA Revision 3.14 has been used in this thesis [7]. The performance parameters of the aircraft models employed in the simulations can be found in Appendix A.

- In the flight simulator, implemented in Simulink, the nominal aircraft model represented by the modelling equations has been modified depending on the problem at hand. Specifically, in the numerical experiments conducted in chapters 3 and 4, model errors are introduced, which include slightly different parameters from the ones provided by BADA, a non-ISA MIL- STD-210C [8] atmosphere model, and a horizontal wind model with turbulence. Additionally, measurement errors are introduced. In the numerical experiments carried out in Chapter 5, uncertainties are considered in the aircraft model. In the numerical experiments conducted in Chapter 6, the BADA models of different commercial aircraft are employed and a horizontal wind model without turbulence is considered, as well as measurement errors.

2.1.1 Aircraft model

In this dissertation, a common 3-DOF dynamic model is considered, which describes the point variable-mass motion of the aircraft over a non-rotating flat earth model, in which the acceleration of gravity is constant and perpendicular to the surface of the earth, that is considered an approximate inertial reference frame. In particular, a symmetric flight is used. Thus, it is assumed that there is no sideslip and all the forces, namely propulsive, aerodynamic, and gravitational forces, lie in the plane of symmetry of the aircraft, which is a conventional jet airplane with fixed engines [9]. This model for aircraft performance has been extensively used in the calculation of trajectories over long-time horizons in which the focus is not on maneuvers, for which more precise assumptions would be needed, but in global performance.

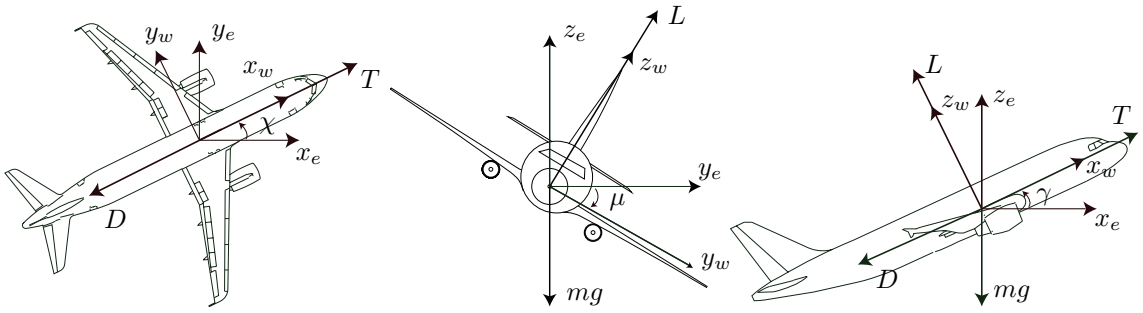
Equations of motion

The 3-DOF equations governing the translational 3D motion of the aircraft are expressed in two different coordinate systems:

- The earth reference frame (O_e, x_e, y_e, z_e), which is fixed to any point on the surface of the earth at mean sea level. The $x_e z_e$ plane is the vertical plane, with the z_e axis pointing to the center of the earth, the x_e axis pointing to a fixed direction in the horizontal plane, and the y_e axis forming a right-handed reference. Thus, it is an approximate inertial reference frame.

[2] https://www-test.eurocontrol.int/services/bada

- The wind reference frame (O_w, x_w, y_w, z_w), which moves with the airplane and is centered in any point of its plane of symmetry, typically the center of gravity. The x_w axis is coincident with the aerodynamic velocity vector of the aircraft, the z_w axis points earthward if the aircraft is in an upright orientation, and the y_w axis forms a right-handed reference.



(a) Top view **(b)** Front view **(c)** Lateral view

Figure 2.1: Aircraft state and forces.

The equations of motion of the aircraft are defined by the following system of differential equations:

$$\begin{aligned}
\dot{V}(t) &= \frac{T(t) - D(h_e(t), V(t), C_L(t)) - m(t) \cdot g \cdot \sin\gamma(t)}{m(t)}, \\
\dot{\chi}(t) &= \frac{L(h_e(t), V(t), C_L(t)) \cdot \sin\mu(t)}{m(t) \cdot V(t) \cdot \cos\gamma(t)}, \\
\dot{\gamma}(t) &= \frac{L(h_e(t), V(t), C_L(t)) \cdot \cos\mu(t) - m(t) \cdot g \cdot \cos\gamma(t)}{m(t) \cdot V(t)}, \\
\dot{x}_e(t) &= V(t) \cdot \cos\gamma(t) \cdot \cos\chi(t) + W_x(x_e(t), y_e(t), h_e(t)), \\
\dot{y}_e(t) &= V(t) \cdot \cos\gamma(t) \cdot \sin\chi(t) + W_y(x_e(t), y_e(t), h_e(t)), \\
\dot{h}_e(t) &= V(t) \cdot \sin\gamma(t), \\
\dot{m}_{climb}(t) &= -T(t) \cdot \eta(V(t)) \quad \text{or} \quad \dot{m}_{descent}(t) = -f_{min}(h_e(t)),
\end{aligned} \tag{2.1}$$

in which the three dynamic equations, relating forces to translational acceleration, are expressed in the wind reference frame, and the three kinematic equations give the translational position relative to the earth reference frame, as shown in Figure 2.1. The variables in (2.1) are defined as follows:

- The triplet (x_e, y_e, h_e) denotes the position of the center of gravity of the aircraft expressed in the earth reference frame.
- V is the true airspeed.
- χ, μ, and γ are, respectively, the heading angle, the bank angle, and the flight-path angle.

- m is the aircraft mass.
- T is the engine thrust.
- η is the fuel efficiency coefficient, and f_{min} is the minimum fuel flow corresponding to idle thrust descent conditions.
- W_x and W_y are the components of the wind vector along the horizontal axes of the earth reference frame. Wind in the vertical direction is assumed to be zero.
- The lift, $L = C_L S\hat{q}$, and the drag, $D = C_D S\hat{q}$, are the components of the aerodynamic force perpendicular and parallel to the airspeed vector, respectively. C_L and C_D are the lift and drag coefficients, respectively, parameter S is the reference wing surface area, and $\hat{q} = \frac{1}{2}\rho V^2$ is the dynamic pressure, with ρ being the air density. A parabolic drag polar $C_D = C_{D0} + C_{Di}C_L^2$ is assumed, being C_{D0} the zero-lift drag coefficient, and C_{Di} the induced drag factor.
- The constant g is the acceleration of gravity.

Flight envelope constraints

Flight envelope constraints are derived from the geometry of the aircraft, structural limitations, engine power, and aerodynamic characteristics. The performance limitations model and the parameters are obtained from BADA.

$$\begin{aligned}
&0 \leq h_e(t) \leq \min[h_{M_o}, h_u(t)], && \gamma_{min} \leq \gamma(t) \leq \gamma_{max},\\
&M(t) \leq M_{M_o}, && m_{min} \leq m(t) \leq m_{max},\\
&\dot{V}(t) \leq \bar{a}_l, && C_v V_s(t) \leq V(t) \leq V_{M_o},\\
&\dot{\gamma}(t)V(t) \leq \bar{a}_n, && 0 \leq C_L(t) \leq C_{L_{max}},\\
&T_{min}(t) \leq T(t) \leq T_{max}(t), && \mu(t) \leq \bar{\mu}.
\end{aligned} \tag{2.2}$$

In (2.2), h_{M_o} is the maximum operational altitude and $h_u(t)$ is the maximum operative altitude at a given mass, which increases as fuel is burned. $M(t)$ is the Mach number and M_{M_o} is the maximum operating Mach number. C_v is the minimum speed coefficient, $V_s(t)$ is the stall speed, V_{M_o} is the maximum operating Calibrated Airspeed (CAS), and $\bar{a}_n$ and $\bar{a}_l$ are, respectively, the maximum normal and longitudinal accelerations for civilian aircraft. Finally, $T_{min}(t)$ and $T_{max}(t)$ correspond to the minimum and maximum available thrust, respectively, and $\bar{\mu}$ corresponds to the maximum bank angle due to structural limitations.

Longitudinal dynamics

In this dissertation, ILC is used for precise trajectory tracking of the vertical profile of different trajectories, since the mission profiles for which airplanes are designed are primarily in the vertical plane. Therefore, the motion of the aircraft is limited to a vertical plane, namely with constant course and thus constant heading angle χ. Without loss of generality,

the heading angle is assumed to be zero, that is $\chi = 0$. In particular, a leveled-wing flight is considered, thus the bank angle is zero, that is $\mu = 0$. Additionally, it is assumed that there are no actions out of the vertical plane, and the wind component perpendicular to the plane of motion is zero, namely $W_y = 0$.

Therefore, the equations of motion are reduced to:

$$\begin{aligned}
\dot{V}(t) &= \frac{T(t) - D(h_e(t), V(t), C_L(t)) - m(t) \cdot g \cdot \sin\gamma(t)}{m(t)}, \\
\dot{\gamma}(t) &= \frac{L(h_e(t), V(t), C_L(t)) - m(t) \cdot g \cdot \cos\gamma(t)}{m(t) \cdot V(t)}, \\
\dot{x}_e(t) &= V(t) \cdot \cos\gamma(t) + W_x(x_e(t), h_e(t)), \\
\dot{h}_e(t) &= V(t) \cdot \sin\gamma(t), \\
\dot{m}_{climb}(t) &= -T(t) \cdot \eta(V(t)) \quad \text{or} \quad \dot{m}_{descent}(t) = -f_{min}(h_e(t)).
\end{aligned} \tag{2.3}$$

The flight envelope constraints remain the same as in (2.2), except for the one referring to the bank angle, since it is assumed to be zero.

The engine thrust, T, and the lift coefficient, C_L, are in this thesis the control variables for the aircraft, namely the control vector is $\boldsymbol{u}(t) = (T(t), C_L(t))$. The thrust is commanded by the engine throttle, and the lift coefficient can be considered linearly related to the angle of attack, which is commanded by the elevator trims. The state variables of the system (2.3) are then the true airspeed, V, the flight path angle, γ, the horizontal position, x_e, the altitude, h_e, and the aircraft mass, m. Thus, the state vector is $\mathbf{x}(t) = (V(t), \gamma(t), x_e(t), h_e(t), m(t))$.

The state variables may not be directly measurable. In Chapter 3, the ILC method is tested in this circumstance, being the output variables $\mathbf{y}(t) = (V_{IAS}(t), M(t), x_e(t), h_e(t), \dot{h}_e(t))$, where V_{IAS} is the indicated speed, and $\dot{h}_e$ is the altitude rate. V_{IAS} and M are defined as

$$\begin{aligned}
V_{IAS}(t) &= a_0 \sqrt{\frac{2}{\kappa - 1}\left(\left(\frac{p(h_e(t))}{p_0}\left[\left(1 + \frac{\kappa - 1}{2} M(t)^2\right)^{\frac{\kappa}{\kappa-1}} - 1\right] + 1\right)^{\frac{\kappa-1}{\kappa}} - 1\right)}, \\
M(t) &= \frac{V(t)}{a(h_e(t))}.
\end{aligned} \tag{2.4}$$

In (2.4), a, a_0, p, and p_0 are the local and sea-level speed of sound and pressure, respectively, and κ is the adiabatic index of air.

2.1.2 Baseline controller

As explained in Chapter 1, one of the advantages of using IILC is that it does not interfere with the aircraft's existing feedback controller for trajectory tracking in the sense that, at each iteration, instead of a new control input, a new trajectory to be followed by the aircraft is generated. Therefore, to test the effectiveness of the IILC method with a more realistic

aircraft model, a PI feedback controller for trajectory tracking has been designed and included as part of the simulated environment used to conduct the experiments.

PI and Proportional Integral Derivative (PID) control are very common feedback control methods (see for example [10] and [11]). In this dissertation, a PI feedback controller is employed to stabilize the aircraft and drive it along a prescribed trajectory. When dealing with the aircraft's longitudinal dynamics, proportional control using the appropriate output error is usually sufficient to achieve the desired dynamic response. Integral control is then added to eliminate the steady-state error (see [12, Chap. 4]).

The longitudinal dynamics characterizes the forward speed and altitude of the aircraft. Driving them to some set values simultaneously requires feedback control of both the thrust and the lift coefficient. Two different baseline controllers have been designed:

- In Chapter 3, a PI output-feedback controller is considered as the aircraft's baseline controller. To select the most suitable output variables to provide feedback while keeping the system stable, the effect of feeding back each control variable with each output on the modal properties of the system is analyzed in [12]. The chosen strategy is to provide Mach feedback to the throttle and altitude rate feedback to the lift coefficient. To avoid excessive control effort, the speed rate is limited and, instead of direct altitude feedback, the altitude error is used to determine an appropriate value for the altitude rate.
- In Chapter 5, a MRAC design is modified to augment a baseline LQR PI state-feedback controller. The LQR with PI action regulates the speed and altitude of the aircraft under nominal conditions. In the presence of uncertainties and changing dynamics, the adaptive component provides an incremental signal to the baseline controller, restoring the desired tracking characteristics.

2.1.3 Atmosphere and wind

The aircraft model described above assumes ISA conditions, a static atmospheric model in which the air is considered an ideal gas and therefore complies with the general gas equation. Below the tropopause, the atmospheric temperature, pressure, and density depend on the altitude, and are calculated as

$$\begin{aligned} T &= T_0 - 0.0065 \cdot h_e, \\ p &= p_0 \left(1 - 22.558 \cdot 10^{-6} \cdot h_e\right)^{5.2559}, \\ \rho &= \rho_0 \left(1 - 22.558 \cdot 10^{-6} \cdot h_e\right)^{4.2559}, \end{aligned} \tag{2.5}$$

where $T_0 = 288.15\,\mathrm{K}$, $p_0 = 101325\,\mathrm{Pa}$, and $\rho_0 = 1.225\,\mathrm{kg/m^3}$ are, respectively, the temperature, pressure, and density at where the geopotential pressure altitude is zero.

Environmental disturbances need to be simulated in order to test the ILC efficiency under realistic conditions that may be faced by the aircraft during the flight. To that end, the

atmosphere model described in the military climatic standard MIL-STD-210C [8] is employed in the trajectory tracking experiments in chapters 3 and 4. In particular, the Non-Standard Day 210C block of Simulink[3] is used, which implements a portion of the climatic data of the MIL-STD-210C worldwide air environment to provide absolute temperature, pressure, density, and speed of sound data as functions of the geopotential altitude.

The wind model used in the simulations provides the combined effect of mean horizontal wind speed and Dryden turbulence using built-in MATLAB functions[4]. Wind in the vertical direction is assumed to be zero. The mean horizontal wind model in chapters 3, 4, and 6 is simulated using the U.S. Naval Research Laboratory Horizontal Wind Model routine [13], which describes the atmosphere's vector wind fields from the surface to the exobase as a function of latitude, longitude, altitude, day of year, and time of day. The empirical formulation, based on the utilization of both satellite and ground-based data sets, properly resolves the thermospheric wind patterns and is able to represent the predominant variations of middle and upper atmosphere to an acceptable degree of accuracy. The Dryden wind turbulence model in chapters 3 and 4 uses an empirical spectral representation to add speed fluctuations to the mean speed, treating the linear and angular velocity components of continuous gusts as spatially varying stochastic processes. The inputs to the Dryden model are altitude, aircraft velocity, and direction cosine matrix. Specifically, in this dissertation, the mathematical representation implemented in the Military Specification MIL-F-8785C [14] is used. Figure 2.2 shows a realization of the wind speed observed by the aircraft when following a climb trajectory.

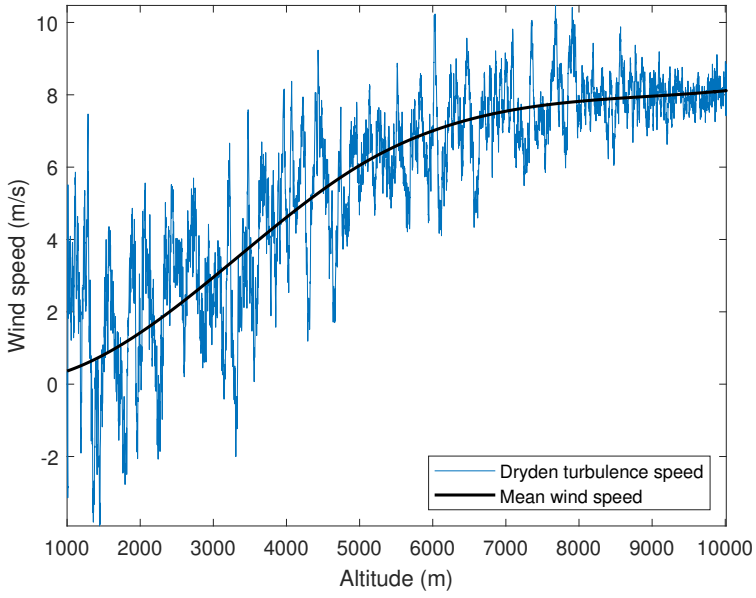


Figure 2.2: Horizontal wind speed model. The blue line represents a realization of the horizontal wind observed by the aircraft when following a climb trajectory, which includes the Dryden turbulence. The black line indicates the mean horizontal wind speed.

[3] https://es.mathworks.com/help/aeroblks/nonstandardday210c.html
[4] https://es.mathworks.com/help/aeroblks/wind.html

2.2 Trajectory generation

In the experiments presented in chapters 3 and 4 of this thesis, the trajectories to be followed are generated using optimal control techniques in which the dynamical model of the aircraft is taken into account, ensuring the feasibility of the planned trajectories. The resulting Optimal Control Problems (OCPs) are solved using DIDO [15], an optimal control software developed by Elissar Global[5] based on the pseudospectral method [16]. This application, besides the optimal control and the corresponding optimal trajectory, returns the Hamiltonian, costates, path covectors, and endpoint covectors. This information, together with classical tools such as the Pontryagin's Principle, is essential to verify the optimality of the numerical results.

In particular, the ILC paradigm is studied for Continuous Climb Operations (CCOs) and Continuous Descent Operations (CDOs), in which it is assumed that the execution of a vertical flight profile optimized to the performance of the aircraft is allowed, leading to a significant reduction of fuel consumption and environmental benefits in terms of noise and emissions. In general, CCOs are part of a Standard Instrument Departure (SID) procedure [17, Chap. 1], and CDOs are part of a Standard Terminal Arrival Route (STAR) procedure, in which case are known as Optimized Profile Descents (OPD) [18, Chap. 1]. The choice of these two types of trajectories is motivated by the fact that one of the most important requirements for successful CCOs and CDOs is predictability, thus precise trajectory tracking is essential to avoid potential conflicts between traffic flows and ensure that safety and capacity are not compromised, especially in the TMA.

2.2.1 Optimal control problem formulation

The input to DIDO is the OCP formulation in a structured format, including the system dynamics, the constraints, and the cost function as follows:

$$
\begin{aligned}
&\text{Preamble}\left\{\begin{array}{ll} \mathcal{X}\subset\mathbb{R}^{n_x}, & \mathcal{U}\subset\mathbb{R}^{n_u},\\ \boldsymbol{x}(t)=(x_1(t),\ldots,x_{n_x}(t)) & \boldsymbol{u}(t)=(u_1(t),\ldots,u_{n_u}(t)),\end{array}\right.\\
&\quad\text{Minimize}\\
&\text{Cost}\left\{\begin{array}{l} J[\boldsymbol{x}(t),\boldsymbol{u}(t),t_0,t_f]=E(\boldsymbol{x_0},\boldsymbol{x_f},t_0,t_f)+\displaystyle\int_{t_0}^{t_f}F(\boldsymbol{x}(t),\boldsymbol{u}(t),t)dt,\end{array}\right.\\
&\quad\text{Subject to:}\\
&\text{Dynamics}\left\{\begin{array}{l}\dot{\boldsymbol{x}}(t)=\boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),t),\end{array}\right.\\
&\text{Events}\left\{\begin{array}{l}\boldsymbol{e^L}\leq\boldsymbol{e}(\boldsymbol{x_0},\boldsymbol{x_f},t_0,t_f)\leq\boldsymbol{e^U},\end{array}\right.\\
&\text{Path}\left\{\begin{array}{l}\boldsymbol{h^L}\leq\boldsymbol{h}(\boldsymbol{x}(t),\boldsymbol{u}(t),t)\leq\boldsymbol{h^U}.\end{array}\right.
\end{aligned}
\tag{2.6}
$$

[5] http://www.elissarglobal.com/industry/products/

Following the structure in (2.6), the sets, functions, and variables of the OCP are:

- Preamble: $\mathcal{X}$ and $\mathcal{U}$ are the state and control domains, being $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ the state and control variables of dimensions n_x and n_u, respectively, where t stands for the time.
- Cost: J is the cost functional of the OCP, with E being the Mayer term and F the Lagrange term, namely the endpoint cost and the running cost, respectively.
- Dynamics: Function $\boldsymbol{f}(\cdot)$ captures the system dynamics.
- Events: The subscripts 0 and f, as in t_0 and t_f, correspond to the initial and final events (or endpoints), respectively. The initial and final conditions of the OCP are specified in the endpoint constraint function, $\boldsymbol{e}(\cdot)$, together with the lower and upper bounds $\boldsymbol{e^L}$ and $\boldsymbol{e}^U$.
- Path: Finally, other general constraints are captured in function $\boldsymbol{h}(\cdot)$, together with the lower and upper bounds $\boldsymbol{h^L}$ and $\boldsymbol{h}^U$.

For the sake of clarity, in the remainder of this section the notation is simplified by dropping some of the dependencies of the functions described above.

To test the optimality of the computed solution, DIDO's output allows for the analysis of the conditions that the problem must comply according to the application of the Pontryagin's Principle. The constraints on the system dynamics can be adjoined to the Lagrange term F by introducing a time-varying Lagrange multiplier vector $\boldsymbol{\lambda}$, the elements of which are called the costates of the system. This motivates the construction of the Hamiltonian, defined as

$$H(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{u}, t) = F(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^T \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t), \tag{2.7}$$

and the Lagrangian of the Hamiltonian

$$\overline{H}(\boldsymbol{\mu}, \boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{u}, t) = H(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\mu}^T \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, t), \tag{2.8}$$

where $\boldsymbol{\mu}$ is a path covector whose components μ_i, known as multipliers, must satisfy the complementary conditions given by

$$\mu_i \begin{cases} \leq 0, & \text{if} \quad h_i(\boldsymbol{x}, \boldsymbol{u}, t) = h_i^L, \\ = 0, & \text{if} \quad h_i^L < h_i(\boldsymbol{x}, \boldsymbol{u}, t) < h_i^U, \\ \geq 0, & \text{if} \quad h_i(\boldsymbol{x}, \boldsymbol{u}, t) = h_i^U, \\ \text{unrestricted}, & \text{if} \quad h_i^L = h_i^U. \end{cases} \tag{2.9}$$

Then, the adjoint equation is given by

$$-\dot{\boldsymbol{\lambda}} = \frac{\partial \overline{H}}{\partial \boldsymbol{x}}. \tag{2.10}$$

For the control function $\boldsymbol{u}$ to be optimal, the Pontryagin's Principle requires that $\boldsymbol{u}$ globally

minimizes the Hamiltonian for every $t \in [t_0, t_f]$. This is known as the Hamiltonian minimization condition, which, for the problem (2.6) described above, is stated as the nonlinear optimization problem

$$\begin{aligned} \underset{\boldsymbol{u}}{\text{minimize}} \quad & H(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{u}, t), \\ \text{subject to:} \quad & \boldsymbol{h}^{\boldsymbol{L}} \leq \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, t) \leq \boldsymbol{h}^{U}. \end{aligned} \tag{2.11}$$

The necessary optimality conditions for (2.11) are given by the stationary condition

$$\frac{\partial \overline{H}}{\partial \boldsymbol{u}} = \mathbf{0}, \tag{2.12}$$

together with the complementary conditions in (2.9), which are known as the Karush-Kuhn-Tucker conditions.

The adjoint covector $\boldsymbol{\lambda}$ must satisfy the so-called transversality conditions

$$-\boldsymbol{\lambda}(t_0) = \frac{\partial \overline{E}}{\partial \boldsymbol{x_0}}, \qquad \boldsymbol{\lambda}(t_f) = \frac{\partial \overline{E}}{\partial \boldsymbol{x_f}}, \tag{2.13}$$

where $\overline{E}$ is the endpoint Lagrangian, which is constructed as

$$\overline{E}(\boldsymbol{\nu}, \boldsymbol{x_0}, \boldsymbol{x_f}, t_0, t_f) = E(\boldsymbol{x_0}, \boldsymbol{x_f}, t_0, t_f) + \boldsymbol{\nu}^T \boldsymbol{e}(\boldsymbol{x_0}, \boldsymbol{x_f}, t_0, t_f), \tag{2.14}$$

where $\boldsymbol{\nu}$ is the endpoint covector, which satisfies the complementary conditions with respect to $\boldsymbol{e}$ analogously to (2.9). Similarly, the values of the Hamiltonian at the endpoints must satisfy the Hamiltonian value conditions

$$\mathcal{H}(\boldsymbol{\lambda}(t_0), \boldsymbol{x}(t_0), t_0) = \frac{\partial \overline{E}}{\partial t_0}, \qquad \mathcal{H}(\boldsymbol{\lambda}(t_f), \boldsymbol{x}(t_f), t_f) = \frac{\partial \overline{E}}{\partial t_f}, \tag{2.15}$$

together with the complementary conditions for the endpoint covector $\boldsymbol{\nu}$, where $\mathcal{H}$ is known as the lower Hamiltonian, which can be obtained by substituting the extremal found in (2.11) into (2.7), that is,

$$\mathcal{H}(\boldsymbol{\lambda}, \boldsymbol{x}, t) = \min_{\boldsymbol{u} \in \mathcal{U}} H(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{u}, t). \tag{2.16}$$

Finally, the Hamiltonian evolution equation is given by

$$\frac{d\mathcal{H}}{dt} = \frac{\partial \overline{H}}{\partial t}. \tag{2.17}$$

2.2.2 Continuous climb operations

In [17], the ICAO defines a CCO as:

> An aircraft operating technique enabled by airspace design, procedure design and facilitation by Air Traffic Control (ATC), allowing the execution of a flight profile optimized to the performance of the aircraft.

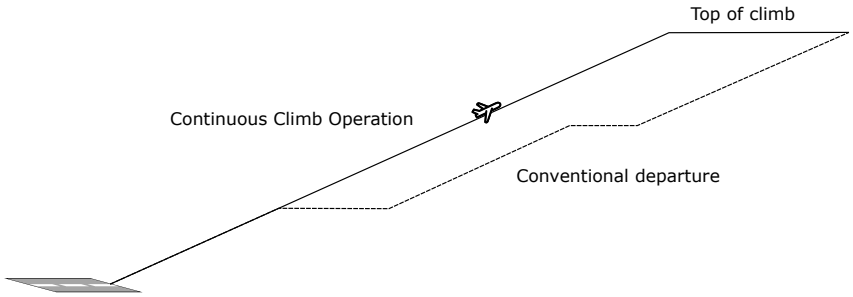


Figure 2.3: Schematic of a CCO compared to a conventional departure.

Some advantages of the CCOs are also enumerated in [17], such as more fuel efficient operations, reduction in both flight crew and controller workload through the design of procedures requiring less ATC intervention, reduction in the number of required radio transmissions, cost savings and environmental benefits through reduced fuel burn, potentially aircraft noise mitigation through thrust and height optimization, and potential authorization of operations where noise limitations would otherwise result in operations being curtailed or restricted. Figure 2.3 illustrates the difference in the vertical layout between a CCO and a conventional departure.

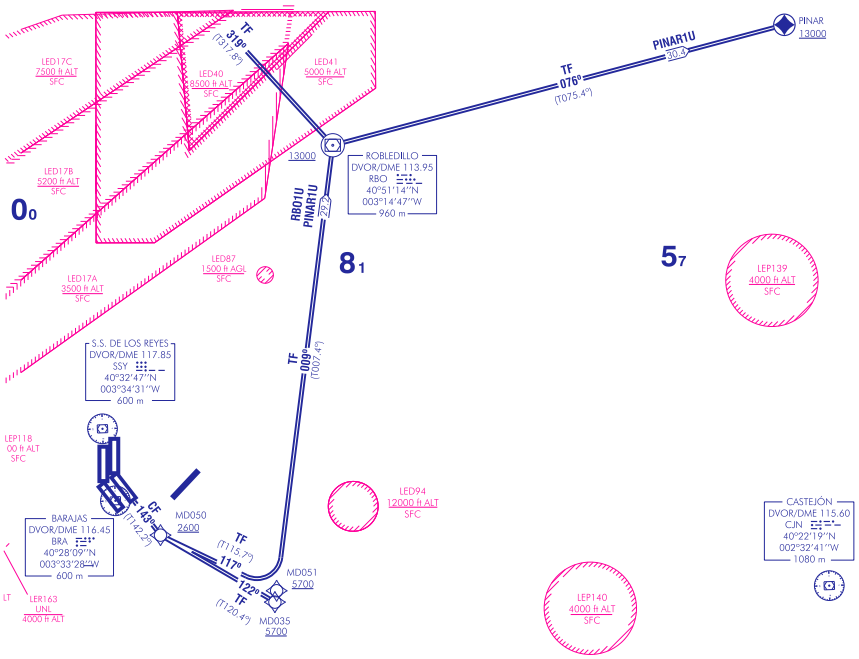


Figure 2.4: SID `PINAR1U`. Source: Spanish AIP service. Not for operational use.

In this dissertation, a CCO trajectory in the vertical plane associated with a SID procedure is generated solving an OCP in which the fuel consumption is minimized. Therefore, airspace constraints that represent the passage through the waypoints that define the SID are introduced in the trajectory generation problem together with the flight envelope constraints. The boundary conditions for the state variables are selected according to a realistic flight departing

from the Adolfo Suárez Madrid-Barajas (LEMD) airport. The trajectory generation starts up at 1000 m after takeoff and ends at 10000 m, when the aircraft reaches the cruise level. Initial velocity and path angle are selected according to standard values.

The selected SID procedure is Madrid Barajas PINAR1U, which is shown in Figure 2.4. The definition of this SID can be found in the Spanish Aeronautical Information Publication (AIP) service[6], managed by ENAIRE, where it is textually described as follows:

> To MD050 on heading $143°$M at 2600 ft or above, turn left. To MD051 at 5700 ft or above, turn left. To RBO at 13000 ft or above, turn right. To PINAR at 13000 ft or above.

Three of the points that define the SID PINAR1U are Area Navigation (RNAV) waypoints, whereas one of them is a DVOR/DME Navigational Aid (NAVAID). The geographical coordinates of these points are listed in Table 2.1.

Table 2.1: Coordinates of the waypoints and navaids of the SID PINAR1U. Source: Spanish AIP service.

Point	Type	Latitude	Longitude
MD050	RNAV waypoint	$40°25'54.0220''$N	$003°29'37.3611''$W
MD051	RNAV waypoint	$40°22'15.4740''$N	$003°19'44.9769''$W
RBO	DVOR/DME navaid	$40°51'14''$N	$003°14'47''$W
PINAR	RNAV waypoint	$40°58'49.0620''$N	$002°35'56.9980''$W

The optimal control inputs generated by DIDO that drive the aircraft along the desired CCO are depicted in Figure 2.5. The resulting path is shown in Figure 2.6, where the solid line is the propagation of the initial conditions through a Runge-Kutta (4,5) routine using a linear interpolation of the obtained optimal control inputs, proving the feasibility of the DIDO solution, whose nodes are represented by the circle markers in the figure. The path remains above the waypoints, represented by the plus sign markers, fulfilling the conditions requested in the SID procedure while performing a continuous climb vertical profile as opposed to the conventional step climb procedure. Analogously, the DIDO solutions for the speed and the flight path angle are shown in Figure 2.7 overlaid with the propagation of the initial conditions.

As mentioned before, the OCP minimizes the fuel consumption, hence maximizing the mass at the final time of the trajectory. The cost functional is then defined as

$$J[\boldsymbol{x}(t), \boldsymbol{u}(t), t_0, t_f] = -m(t_f) = -m_f, \tag{2.18}$$

where m_f is the mass at the final time t_f. Therefore, in (2.6) the endpoint cost is $E(\boldsymbol{x_0}, \boldsymbol{x_f}, t_0, t_f) = -m_f$ and the running cost is $F(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = 0$.

[6] https://ais.enaire.es/AIP/

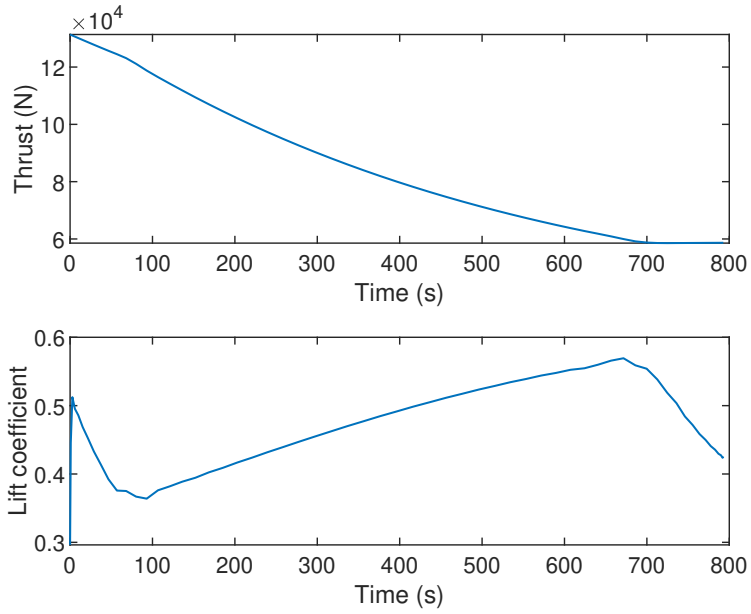


Figure 2.5: CCO optimal control inputs generated by DIDO.

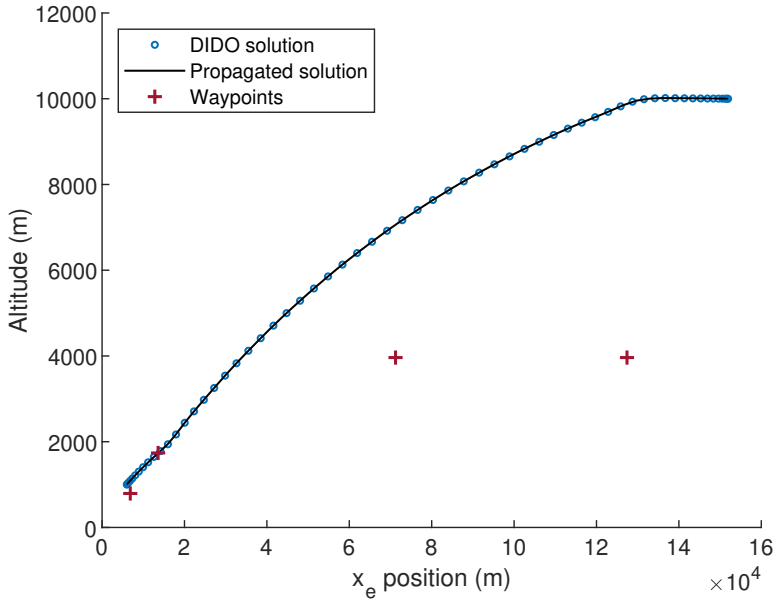


Figure 2.6: CCO path generated by DIDO (circle markers) overlaid with the propagation of the initial conditions (solid line). The plus sign markers indicate the waypoints of the SID procedure.

Following Section 2.2.1, and according to the longitudinal dynamics of the aircraft motion (2.3), the flight envelope constraints (2.2), and the cost functional (2.18), the Lagrangian of the Hamiltonian does not explicitly depend on time. Consequently, in agreement with the Hamiltonian evolution equation (2.17), the lower hamiltonian should be constant throughout the whole trajectory to comply with the Pontryagrin's Principle. In particular, this constant

value, as derived from the Hamiltonian value conditions (2.15) should be $\mathcal{H}(\boldsymbol{\lambda}, \boldsymbol{x}, t) = 0$ for every $t \in [t_0, t_f]$. Figure 2.8 proves the compliance of this condition, giving confidence about the optimality of the results.

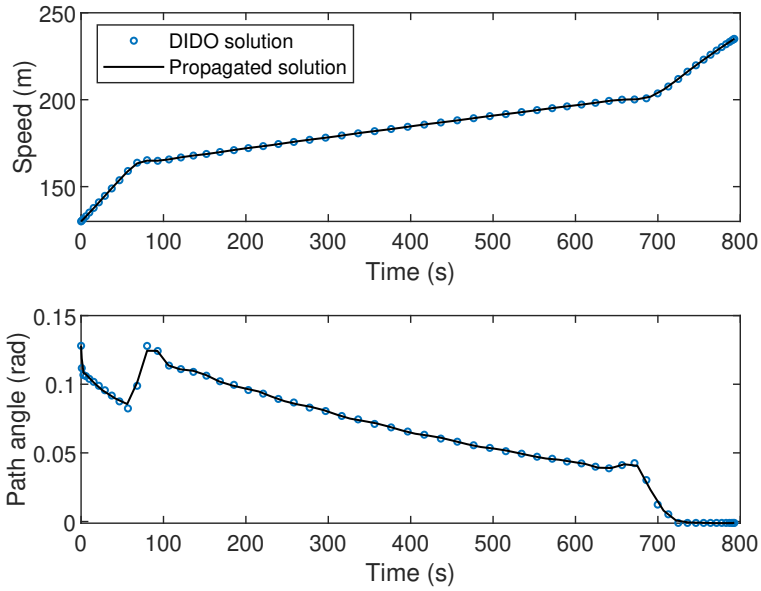


Figure 2.7: CCO speed and path angle generated by DIDO (circle markers) overlaid with the propagation of the initial conditions (solid lines).

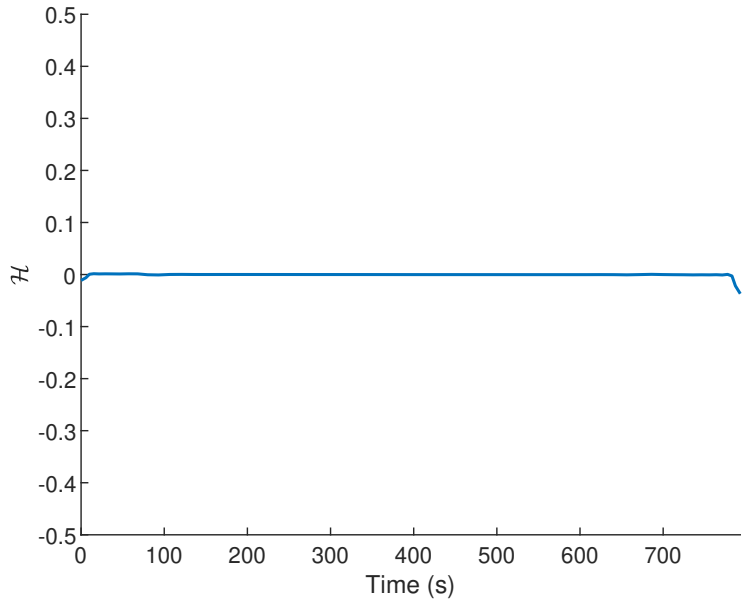


Figure 2.8: Lower Hamiltonian of the CCO path generated by DIDO.

2.2.3 Continuous descent operations

The general term CDO makes reference to the operations in which aircraft descend from the cruise altitude to the final approach fix at or near idle thrust, ideally in a low drag configuration, without level segments at low altitude, minimizing the need for high thrust levels to remain at a constant altitude and reducing the environmental impact of the flight. The european definition of a CDO, as approved by the stakeholders, is:

> An aircraft operating technique in which an arriving aircraft descends from an optimal position with minimum thrust and avoids level flight to the extent permitted by the safe operation of the aircraft and compliance with published procedures and ATC instructions.

According to [18, Chap. 1], an optimum CDO starts from the Top-Of-Descent (TOD) and uses descent profiles that reduce controller-pilot communications and segments of level flight. Furthermore, it provides for a reduction in noise, fuel burn, and emissions, while increasing the flight stability and the predictability of the flight path to both controllers and pilots. Figure 2.9 illustrates the difference in the vertical layout between a CDO and a conventional approach.

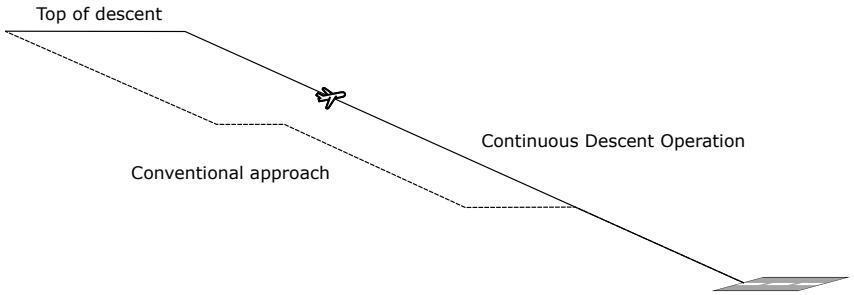


Figure 2.9: Schematic of a CDO compared to a conventional approach.

In this dissertation, a generic CDO is generated in the vertical plane solving an OCP to find the trajectory and the corresponding control inputs for which the horizontal distance is minimized. It starts at the TOD and continuously descends until reaching an altitude of $2500\,\mathrm{m}$ at a speed of $130\,\mathrm{m/s}$. The final conditions of altitude and speed of the designed trajectory are compliant with a STAR procedure to complete the landing.

Figure 2.10 shows the optimal control inputs generated by DIDO for the simulated aircraft to perform the CDO. As in the previous case, the path nodes generated by DIDO are the circle markers depicted in Figure 2.11, and the solid line is the corresponding propagation of the initial conditions through a Runge-Kutta (4,5) routine by interpolating the obtained optimal control inputs. The DIDO solution and the propagation of the speed and the flight angle are shown in Figure 2.12. These figures show the dynamical feasibility of the optimal solution generated by DIDO. The resulting CDO path continuously descends without level segments.

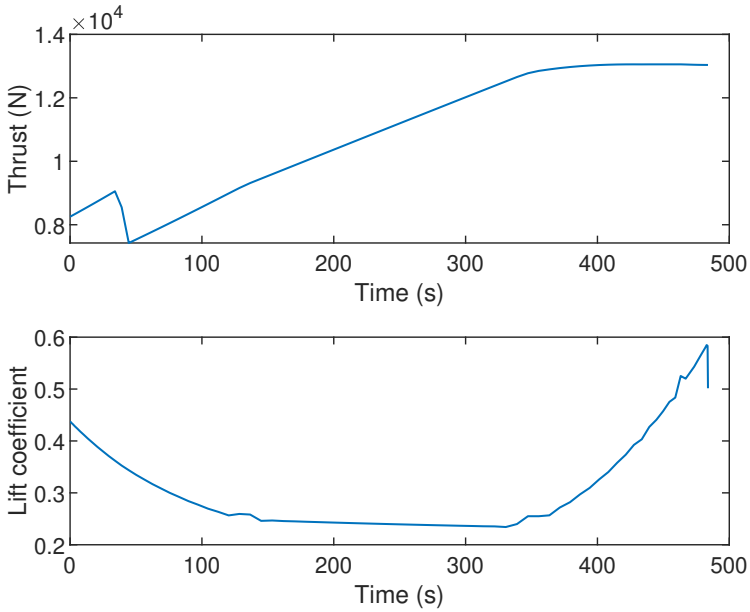


Figure 2.10: CDO optimal control inputs generated by DIDO.

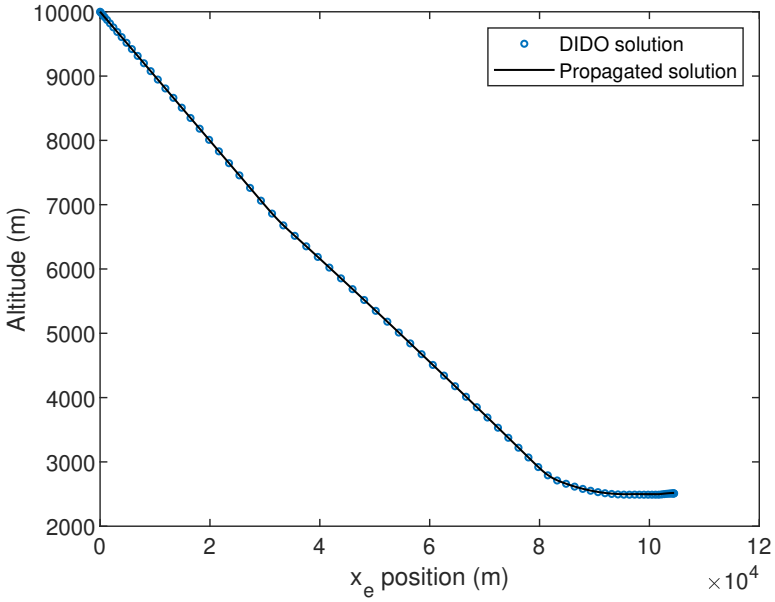


Figure 2.11: CDO path generated by DIDO (circle markers) overlaid with the propagation of the initial conditions (solid lines).

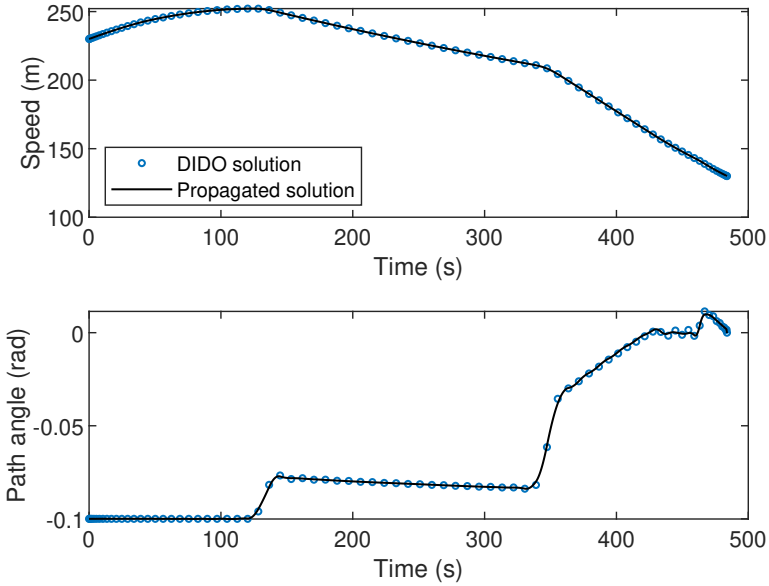


Figure 2.12: CDO speed and path angle generated by DIDO (circle markers) overlaid with the propagation of the initial conditions (solid lines).

In this case, the horizontal distance is minimized, and the cost functional is defined as

$$J[\boldsymbol{x}(t), \boldsymbol{u}(t), t_0, t_f] = x_e(t_f) = x_{ef}, \tag{2.19}$$

where x_{ef} is the horizontal distance at the final time t_f. The endpoint cost is then $E(\boldsymbol{x_0}, \boldsymbol{x_f}, t_0, t_f) = x_{ef}$, and the running cost is $F(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = 0$.

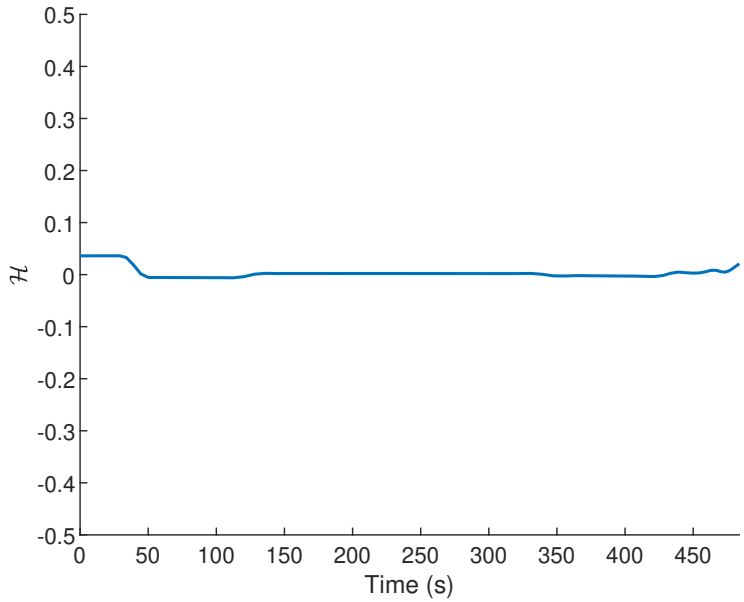


Figure 2.13: Lower Hamiltonian of the CDO path generated by DIDO.

Again, taking into account the longitudinal dynamics of the aircraft (2.3), the flight envelope (2.2), and the cost functional (2.19), the Lagrangian of the Hamiltonian does not explicitly depend on time. Therefore, following the Hamiltonian evolution equation (2.17), the lower Hamiltonian should be constant throughout the whole trajectory to comply with the Pontryagrin's Principle. In particular, $\mathcal{H}(\boldsymbol{\lambda}, \boldsymbol{x}, t) = 0$ for every $t \in [t_0, t_f]$, according to the Hamiltonian value conditions (2.15). Figure 2.13 shows that the lower Hamiltonian remains zero except for small deviations, suggesting the optimality of the provided solution.

2.3 Conclusion

In this chapter, the simulated environment in which the experiments described in the following chapters are carried out has been introduced. It includes the flight simulator and the optimal trajectories to be followed by the aircraft.

The flight simulator is composed of the aircraft model, for which the motion equations are formulated in the longitudinal plane, the flight envelope, and a baseline controller. The aircraft data have been obtained from BADA to reproduce a realistic performance of the flights. A weather model is also part of the flight simulator, which includes ISA and non-ISA atmosphere conditions, as well as a wind model composed of the combined effect of mean horizontal wind and Dryden turbulence. Finally, model and measurement errors are also introduced.

For the trajectory generation, the optimal control software DIDO, which is based on the pseudospectral method, has been used to obtain feasible optimal trajectories for CCOs and CDOs based on the pseudospectral method. The Pontryagin's Principle has been used to verify the optimality of the solutions provided by DIDO.

The simulated environment described in this chapter will be assumed to be known throughout the remaining sections of this thesis.

3

Iterative Learning Control

In this chapter, an optimization-based ILC scheme is applied to aircraft trajectory tracking in continuous climb and descent operations. Two approaches of the ILC are considered: the direct and the indirect ILC. The aim of the ILC is to compensate for recurrent disturbances, caused mainly by weather and model errors, achieving precise trajectory tracking in few iterations. In the indirect approach, the aircraft is also equipped with a feedback controller, which compensates for nonrepetitive disturbances.

This method is especially suitable for aircraft 4D trajectory tracking in departure and arrival procedures at busy airports because, due to the short time separation between flights, weather conditions and therefore disturbances can be assumed to be similar between consecutive flights, and constraints, which include waypoints and other limitations due to the procedures, can be straightforwardly taken into account.

3.1 Introduction

ILC is based on the idea that the performance of a control system that executes the same task multiple times can be improved by learning from previous executions. In particular, precision in aircraft trajectory tracking can be improved by incorporating error information from previous flights into the control law for subsequent aircraft performing the same planned trajectory. In doing so, accurate performance can be achieved with low transient tracking error despite recurrent disturbances.

The ILC paradigm emerged from industrial robotics applications to improve trajectory tracking precision of robot manipulators in repetitive tasks. The widely recognized seminal work in ILC is [19], which generated a major area of research in control systems for robotics applications. Since then, it has broadened including links with other control paradigms. In [20, Sect. 2] and references therein, non-linear ILC, robustness, adaptive schemes in ILC, the

optimal control approach to ILC, and neural-network-based ILC are introduced. In [21], the main results on ILC analysis and design are surveyed. Stability, performance, learning transient behavior, and robustness in ILC are discussed and the most popular design techniques are described. They include PD-type ILC, plant inversion methods, $\mathcal{H}_\infty$ methods, and quadratically optimal design. Extensions to time-varying, continuous-time, multivariable, and nonlinear systems are also outlined. Several textbooks on ILC are available, such as [22], which treats ILC for both linear and nonlinear systems, and [23], which focuses on real time applications. Application areas have also expanded beyond industrial robotics and ILC has recently been applied to precise trajectory tracking of aerial robots. In [24], an ILC approach is applied to precise quadrocopter trajectory tracking in the presence of model errors and other recurrent disturbances. In [25], a least-squares based learning rule is proposed to perform aggressive maneuvers with quadrocopters. Other variants of ILC have been applied to trajectory tracking for Unmanned Aerial Vehicles (UAV), see for example [26] and references therein. To the best knowledge of the authors, ILC has not been applied yet to precise aircraft trajectory tracking.

In this chapter, the optimization-based ILC method presented in [24] is applied to aircraft trajectory tracking. This ILC scheme requires the system dynamics to be repetition-invariant, which means that the same dynamical system (in this context, the aircraft) executes the same task flying the same planned trajectory at each iteration. The problem of knowledge transfer among different aircraft is addressed in Chapter 6. The ILC problem is solved in two steps, both relying on a nominal model of the aircraft, in which input and state constraints are explicitly taken into account. The first step consists in the estimation of the model errors and external disturbances affecting the flight of an aircraft along a trajectory using a time-varying Kalman filter. In the second step, optimization techniques are employed to determine an updated control input for the following aircraft, which optimally compensates for the recurrent disturbances in tracking the same trajectory. Thus, this chapter shows that precision in aircraft trajectory tracking in CCOs and CDOs can be improved using direct ILC by pure feed-forward adaptation of the control input, and that the accuracy of the trajectory tracking is limited by the level of non-repetitive disturbances.

An interesting aspect of the ILC method is that it can be made nonintrusive with respect to the trajectory tracking control system of the aircraft by calculating, at each iteration, a reference trajectory for the following aircraft rather than a control input. In this thesis, an optimization-based IILC approach is employed for this purpose. An overview of IILC can be found in [27]. It is a subclass of ILC that, unlike direct ILC, which directly alters the open-loop input to the system, modifies the reference trajectory given to a feedback controller, which then converts this trajectory into system control inputs. Thus, it can be applied to systems with underlying feedback controllers for trajectory tracking, such as aircraft, since a new reference trajectory can be computed as part of the input update and fed into the flight management system of the following aircraft. In this way, the feedback trajectory tracking control reduces nonrepetitive disturbances while the IILC is intended to reject repetitive disturbances.

3.2 ILC problem formulation

In this section, following [24], the ILC method implemented in this thesis for precise aircraft trajectory tracking is introduced. The starting point of the learning algorithm is a time-varying nonlinear model of the real dynamic system

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \\ \mathbf{y}(t) &= \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), t), \end{aligned} \tag{3.1}$$

where $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ and $\dot{\mathbf{x}}(t) \in \mathbb{R}^{n_x}$ are the state and the state's derivative, respectively, $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ is the control input, $\mathbf{y}(t) \in \mathbb{R}^{n_y}$ is the output, and $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are assumed to be continuously differentiable in $\mathbf{x}$ and $\mathbf{u}$.

Constraints on the state $\mathbf{x}(t)$, the input $\mathbf{u}(t)$, and their time derivatives are represented by

$$\mathbf{Z}\mathbf{q}(t) \preceq \mathbf{q}_{\max}, \tag{3.2}$$

where

$$\mathbf{q}(t) = \left[\mathbf{x}^T(t), \mathbf{u}^T(t), \dot{\mathbf{x}}^T(t), \dot{\mathbf{u}}^T(t), \ldots, \frac{d^m}{dt^m}\mathbf{x}^T(t), \frac{d^m}{dt^m}\mathbf{u}^T(t)\right]^T, \tag{3.3}$$

$\mathbf{Z}$ is a constant matrix of appropriate dimensions, and $\mathbf{q}_{\max} \in \mathbb{R}^{n_q}$, being n_q the total number of constraints. The inequality denoted by the symbol $\preceq$ is defined component-wise.

The goal of the proposed learning algorithm is to precisely track a feasible predefined output trajectory $\mathbf{y_d}(t)$ over a finite time interval $t \in \mathcal{T} = [t_0, t_f]$, with $t_f < \infty$. This trajectory is assumed to be feasible with respect to the nominal model. The desired output trajectory, $\mathbf{y_d}(t)$, and its corresponding state, $\mathbf{x_d}(t)$, and control, $\mathbf{u_d}(t)$, are here the result of solving an OCP based on the system dynamics (2.3). As described in Section 2.2, the OCPs formulated with the aim to generate the desired CCO and CDO trajectories are solved using a pseudospectral method.

The behavior of the system (2.1) can be represented as a linear time-varying system

$$\begin{aligned} \dot{\tilde{\mathbf{x}}}(t) &= \boldsymbol{A}(t)\tilde{\mathbf{x}}(t) + \boldsymbol{B}(t)\tilde{\mathbf{u}}(t), \\ \dot{\tilde{\mathbf{y}}}(t) &= \boldsymbol{C}(t)\tilde{\mathbf{x}}(t) + \boldsymbol{D}(t)\tilde{\mathbf{u}}(t), \quad t \in \mathcal{T}, \end{aligned} \tag{3.4}$$

where the time-dependent matrices $\boldsymbol{A}(t), \boldsymbol{B}(t), \boldsymbol{C}(t)$, and $\boldsymbol{D}(t)$ are the corresponding Jacobian matrices of the nonlinear functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ with respect to $\mathbf{x}$ and $\mathbf{u}$. The triple $(\tilde{\mathbf{u}}(t), \tilde{\mathbf{x}}(t), \tilde{\mathbf{y}}(t))$ represent small deviations from the desired output trajectory and its corresponding state and input $(\mathbf{x_d}(t), \mathbf{u_d}(t), \mathbf{y_d}(t))$, namely

$$\begin{aligned} \tilde{\mathbf{x}}(t) &= \mathbf{x}(t) - \mathbf{x_d}(t), \\ \tilde{\mathbf{u}}(t) &= \mathbf{u}(t) - \mathbf{u_d}(t), \\ \tilde{\mathbf{y}}(t) &= \mathbf{y}(t) - \mathbf{y_d}(t). \end{aligned} \tag{3.5}$$

In a real system, measurements are only available at fixed time intervals, therefore a discrete-time representation is needed, which results in the following linear, time-varying difference equations:

$$\begin{aligned} \tilde{\mathbf{x}}(k+1) &= \boldsymbol{A_D}(k)\tilde{\mathbf{x}}(k) + \boldsymbol{B_D}(k)\tilde{\mathbf{u}}(k) \\ \tilde{\mathbf{y}}(k) &= \boldsymbol{C_D}(k)\tilde{\mathbf{x}}(k) + \boldsymbol{D_D}(k)\tilde{\mathbf{u}}(k), \end{aligned} \tag{3.6}$$

where $k \in \mathcal{K} = \{0, 1, ..., N-1\}, N < \infty$, represents the discrete-time index. The desired trajectory is then represented by an N-sample sequence

$$(\mathbf{u_d}(k), \mathbf{x_d}(k+1), \mathbf{y_d}(k+1)), \quad k \in \mathcal{K}, \tag{3.7}$$

with given initial state $\mathbf{x_d}(0)$. The input and state constraints (3.2) are similarly transformed.

3.2.1 Lifted system representation

A useful way of describing the model dynamics during a single iteration is the lifted system representation, which consists of a static mapping of the finite input time series, $\tilde{\mathbf{u}}(k)$, into the corresponding output time series, $\tilde{\mathbf{y}}(k)$, for each trial [28], [29]. The deviations with respect to the desired trajectory (3.7) are then stacked vectors of all discretization time-steps, that is,

$$\begin{aligned} \boldsymbol{u} &= \left[\tilde{\mathbf{u}}^T(0), \tilde{\mathbf{u}}^T(1), \ldots, \tilde{\mathbf{u}}^T(N-1)\right]^T \in \mathbb{R}^{Nn_u}, \text{[1]} \\ \boldsymbol{x} &= \left[\tilde{\mathbf{x}}^T(1), \tilde{\mathbf{x}}^T(2), \ldots, \tilde{\mathbf{x}}^T(N)\right]^T \in \mathbb{R}^{Nn_x}, \\ \boldsymbol{y} &= \left[\tilde{\mathbf{y}}^T(1), \tilde{\mathbf{y}}^T(2), \ldots, \tilde{\mathbf{y}}^T(N)\right]^T \in \mathbb{R}^{Nn_y}. \end{aligned} \tag{3.8}$$

Using this notation, the system (3.1) can be described as

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{F}\boldsymbol{u} + \boldsymbol{d^0}, \\ \boldsymbol{y} &= \boldsymbol{G}\boldsymbol{x} + \boldsymbol{H}\boldsymbol{u}, \end{aligned} \tag{3.9}$$

where the lifted matrices $\boldsymbol{F} \in \mathbb{R}^{Nn_x \times Nn_u}$, $\boldsymbol{G} \in \mathbb{R}^{Nn_y \times Nn_x}$, and $\boldsymbol{H} \in \mathbb{R}^{Nn_y \times Nn_u}$ account for the nominal model dynamics, and vector $\boldsymbol{d^0} \in \mathbb{R}^{Nn_x}$ contains the free response of the system to the initial deviation.

Matrix $\boldsymbol{F}$ is composed of the matrices $\boldsymbol{F}_{(l,m)} \in \mathbb{R}^{n_x \times n_u}$, $1 \leq l, m \leq N$, namely

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{F}_{(1,1)} & \cdots & \boldsymbol{F}_{(1,N)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{F}_{(N,1)} & \cdots & \boldsymbol{F}_{(N,N)} \end{bmatrix}, \tag{3.10}$$

[1] Note that the notation $\mathbb{R}^{Nn_u}$ refers to the real vector space, the dimension of which is the scalar product $N \cdot n_u$. It is defined analogously in all the vectors and matrices where this notation appears throughout this dissertation.

where

$$\boldsymbol{F}(l,m) = \begin{cases} \boldsymbol{A_D}(l-1)\dots\boldsymbol{A_D}(m)\boldsymbol{B_D}(m-1), & \text{if} \quad m < l, \\ \boldsymbol{B_D}(m-1), & \text{if} \quad m = l, \\ 0, & \text{if} \quad m > l. \end{cases}$$

Matrices $\boldsymbol{G}$ and $\boldsymbol{H}$ are block-diagonal and analogously defined by

$$\boldsymbol{G}(l,m) = \begin{cases} \boldsymbol{C_D}(l), & \text{if} \quad l = m, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$\boldsymbol{H}(l,m) = \begin{cases} \boldsymbol{D_D}(l), & \text{if} \quad l = m, \\ 0, & \text{otherwise.} \end{cases}$$

Finally, vector $\boldsymbol{d^0}$ is described as

$$\boldsymbol{d^0} = \left[(\boldsymbol{A_D}(0)\tilde{\mathbf{x}}_0)^T, (\boldsymbol{A_D}(1)\boldsymbol{A_D}(0)\tilde{\mathbf{x}}_0)^T, \dots, \left(\prod_{i=0}^{N-1} \boldsymbol{A_D}(i)\tilde{\mathbf{x}}_0 \right)^T \right]^T,$$

where $\tilde{\mathbf{x}}(0) = \tilde{\mathbf{x}}_0 \in \mathbb{R}^{n_x}$ is the initial deviation.

Based on this notation, the ILC algorithm is divided into two steps: disturbance estimation and input update.

3.2.2 Disturbance estimation

In order to take into account the repetitive nature of the problem setting, the system (3.9) is rewritten as

$$\begin{aligned} \boldsymbol{x}_j &= \boldsymbol{F}\boldsymbol{u}_j + \boldsymbol{d}_j, \\ \boldsymbol{y}_j &= \boldsymbol{G}\boldsymbol{x}_j + \boldsymbol{H}\boldsymbol{u}_j, \end{aligned} \tag{3.11}$$

where the subscript j indicates the $j-$th execution of the desired task and $\boldsymbol{d}_j$ captures the repetitive disturbances along the reference trajectory, including model errors and the free response of the system to the initial deviation. The vector $\boldsymbol{d}_j$ experiences only slight random changes between iterations, which are denoted as $\boldsymbol{\omega}_j$.

Taking into account both process and measurement noises, which are captured in the random variable $\boldsymbol{\epsilon}_j$, the evolution of the learning over consecutive trials can be represented as the Kalman filter model

$$\begin{aligned} \boldsymbol{d}_j &= \boldsymbol{d}_{j-1} + \boldsymbol{\omega}_{j-1}, \\ \boldsymbol{y}_j &= \boldsymbol{G}\boldsymbol{d}_j + (\boldsymbol{G}\boldsymbol{F} + \boldsymbol{H})\boldsymbol{u}_j + \boldsymbol{\epsilon}_j, \end{aligned} \tag{3.12}$$

where both stochastic zero-mean Gaussian white noise variables, $\boldsymbol{\omega}_j \sim \mathcal{N}(0, \boldsymbol{\Omega}_j)$ and $\boldsymbol{\epsilon}_j \sim \mathcal{N}(0, \boldsymbol{M}_j)$, are trial-uncorrelated and assumed to be independent. Matrices $\boldsymbol{\Omega}_j$ and $\boldsymbol{M}_j$

represent the noise covariances and can be characterized as diagonal matrices [30, Chap. 5.2]. As mentioned in Section 2.1, in the aircraft simulation used for this study, the noise is introduced as measurement errors and variations in both the aircraft model and the wind speed, as well as the atmospheric model.

The Kalman filter estimates the current error $\boldsymbol{d}_j$ taking into account the output signals $\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_j$ from previous trials. Given the initial values of the error estimate, $\widehat{\boldsymbol{d}}_0$, and the error covariance matrix, $\boldsymbol{P}_0 = E\left[(\boldsymbol{d}_0 - \widehat{\boldsymbol{d}}_0)(\boldsymbol{d}_0 - \widehat{\boldsymbol{d}}_0)^T\right]$, the disturbance estimate is calculated as

$$\widehat{\boldsymbol{d}}_j = \widehat{\boldsymbol{d}}_{j-1} + \boldsymbol{K}_j\left(\boldsymbol{y}_j - \boldsymbol{G}\widehat{\boldsymbol{d}}_{j-1} - (\boldsymbol{GF} + \boldsymbol{H})\boldsymbol{u}_j\right), \tag{3.13}$$

where $\boldsymbol{K}_j$ is the optimal Kalman gain, which is computed as

$$\boldsymbol{K}_j = (\boldsymbol{P}_{j-1} + \boldsymbol{\Omega}_{j-1})\,\boldsymbol{G}^T\left(\boldsymbol{G}\,(\boldsymbol{P}_{j-1} + \boldsymbol{\Omega}_{j-1})\,\boldsymbol{G}^T + \boldsymbol{M}_j\right)^{-1}. \tag{3.14}$$

Knowledge of the disturbance behavior can be incorporated into the Kalman filter in the form of equality and inequality constraints to improve its performance [31]. In this case, disturbance variation over time is bounded to obtain a smooth estimation of the mean horizontal wind speed while dismissing the turbulence effect, which varies over iterations.

Constraints can be addressed projecting the unconstrained estimate of the Kalman filter onto the constraint surface. The constrained disturbance estimation can therefore be written as the following quadratic programming problem:

$$\begin{aligned} \min_{\widehat{\boldsymbol{d}}_{j,c}} \quad & (\widehat{\boldsymbol{d}}_{j,c} - \widehat{\boldsymbol{d}}_j)^T \boldsymbol{W} (\widehat{\boldsymbol{d}}_{j,c} - \widehat{\boldsymbol{d}}_j), \\ \text{subject to:} \quad & \boldsymbol{C}\widehat{\boldsymbol{d}}_{j,c} \leq \boldsymbol{c}_{max}, \end{aligned} \tag{3.15}$$

where $\boldsymbol{C}$ is a matrix of appropriate dimensions, $\boldsymbol{c}_{max}$ is the maximum allowed variation of the disturbance, and $\boldsymbol{W}$ is a positive-definite weighting matrix. Then, if the process and measurement noises are Gaussian, the maximum probability estimates are achieved by setting $\boldsymbol{W} = \boldsymbol{P}_j$. This estimated error is used as the predictive disturbance for the following iteration, that is, $\boldsymbol{d}^p_{j+1} = \widehat{\boldsymbol{d}}_{j,c}$.

Notice that, in Chapter 4, the Kalman filter is substituted by a GPR estimation and prediction of the disturbance affecting the system.

3.2.3 Input update

The learning update consists in deriving a model-based update rule to compute a new control input $\boldsymbol{u}_{j+1} \in \mathbb{R}^{Nn_u}$, which, in response to the predicted disturbance $\boldsymbol{d}^p_{j+1}$, minimizes the deviation from the nominal trajectory in the following iteration. Since this deviation, $\boldsymbol{x}_{j+1}$, is unknown, its expected value given all past measurements is considered, namely

$$E\left[\boldsymbol{x}_{j+1} | \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_j\right] = \boldsymbol{F}\boldsymbol{u}_{j+1} + \boldsymbol{d}^p_{j+1}. \tag{3.16}$$

The update rule can be expressed by means of the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{u}_{j+1}} \quad & \|\boldsymbol{F}\boldsymbol{u}_{j+1} + \boldsymbol{d}^p_{j+1}\|_\ell + \alpha\|\boldsymbol{D}\boldsymbol{u}_{j+1}\|_\ell, \\ \text{subject to:} \quad & \boldsymbol{L}\boldsymbol{u}_{j+1} \leq \boldsymbol{q}_{\max}, \end{aligned} \tag{3.17}$$

where constraints (3.2) are explicitly taken into account. The additional term $\alpha \geq 0$ and the matrix $\boldsymbol{D}$ are introduced to penalize the input or approximations of its derivatives in order to enforce the smoothness of the optimal solution. The vector ℓ-norm, with $\ell \in \{1, 2, \infty\}$, of the minimization problem (3.17) affects the result and convergence of the learning algorithm and should be chosen in accordance with the performance objectives. In the experiments herein exposed, the vector norm $\ell = 2$ has been chosen.

The update law defined in (3.17) can be formulated as a standard convex optimization problem of the form:

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & \left(\frac{1}{2}\boldsymbol{z}^T\boldsymbol{V}\boldsymbol{z} + \boldsymbol{v}^T\boldsymbol{z}\right), \\ \text{subject to:} \quad & \boldsymbol{W}\boldsymbol{z} \leq \boldsymbol{w} \quad \text{and} \quad \boldsymbol{\eta}_1 \leq \boldsymbol{z} \leq \boldsymbol{\eta}_2, \end{aligned} \tag{3.18}$$

where $\boldsymbol{z} \in \mathbb{R}^{n_z}$ represents the vector of decision variables, and vectors $\boldsymbol{v}, \boldsymbol{w}$ and matrices $\boldsymbol{V}, \boldsymbol{W}$ have appropriate dimensions. The optimization problem (3.18) is solved here using CPLEX[2] for MATLAB, which provides a tool for solving optimization or mathematical programming problems, including quadratic and quadratically constrained problems.

Additional constraints can be included to adapt the algorithm to the system requirements. For example, limiting the variation of vector $\boldsymbol{d}_j$ from one iteration to the following one by a certain percentage or a pre-specified value. Additionally, a minimum value can be defined for the state variables' deviations from the desired trajectory, updating the input only if the deviations exceed this value, and applying the same input to the next aircraft instead. The maximum deviation allowed may be different for each state variable and each part of the trajectory, e.g., in the case of a CCO or a CDO, deviations at high altitudes may be greater than near the TMA, where higher precision is required.

The scaling of the original signals $\mathbf{u}(t), \mathbf{x}(t)$, and $\mathbf{y}(t)$ in (3.1) is essential to guarantee reasonable results in the optimization problem (3.17). This scaling, exemplarily shown on the system's state $\mathbf{x}(t)$, reads as

$$\boldsymbol{x}^s = \boldsymbol{S}\boldsymbol{x}, \quad \boldsymbol{S} \in \mathbb{R}^{Nn_x \times Nn_x}, \tag{3.19}$$

The scaling matrix, $\boldsymbol{S}$, is a composition of three diagonal scaling matrices, $\boldsymbol{S} = \boldsymbol{S}_t\boldsymbol{S}_w\boldsymbol{S}_x$, with $\boldsymbol{S}_t, \boldsymbol{S}_w, \boldsymbol{S}_x \in \mathbb{R}^{Nn_x \times Nn_x}$, being $\boldsymbol{S}_x$ the state scaling matrix such that the elements of $\boldsymbol{x}^s = \boldsymbol{S}_x\boldsymbol{x}$ are within the same range of magnitude. $\boldsymbol{S}_w$ is a state weighting matrix, which

[2] https://www.ibm.com/es-es/products/ilog-cplex-optimization-studio

may be useful to give greater importance to some of the state variables over the rest. Finally, $\boldsymbol{S}_t$ modifies the scaling along the trajectory in order to put emphasis on specific parts of it.

3.2.4 Indirect approach

As mentioned in Section 3.1, one of the advantages of the indirect variant of the iterative learning algorithm is its non intrusiveness with respect to the aircraft's existing trajectory tracking controller, since a new reference trajectory is provided to the following aircraft rather than a control input. Once the updated input $\boldsymbol{u}_{j+1}$ is calculated in (3.17), the new reference trajectory is obtained by introducing this input into the lifted model (3.9), dismissing the disturbances, since they have been already taken into account in the optimization step, namely

$$\begin{aligned} \boldsymbol{x}_r &= \boldsymbol{F}\boldsymbol{u}_{j+1} + \boldsymbol{x}_d, \\ \boldsymbol{y}_r &= \boldsymbol{G}\boldsymbol{x}_r + \boldsymbol{H}\boldsymbol{u}_{j+1} + \boldsymbol{y}_d, \end{aligned} \tag{3.20}$$

where $\boldsymbol{x}_r \in \mathbb{R}^{Nn_x}$ and $\boldsymbol{y}_r \in \mathbb{R}^{Nn_y}$ are, respectively, the new reference state and output lifted vectors, and $\boldsymbol{x}_d \in \mathbb{R}^{Nn_x}$ and $\boldsymbol{y}_d \in \mathbb{R}^{Nn_y}$ are, respectively, the desired state and output vectors also in a lifted form. The reference trajectory to be fed to the system's feedback controller can be obtained from the unlifted representation form of $\boldsymbol{x}_r$ and $\boldsymbol{y}_r$.

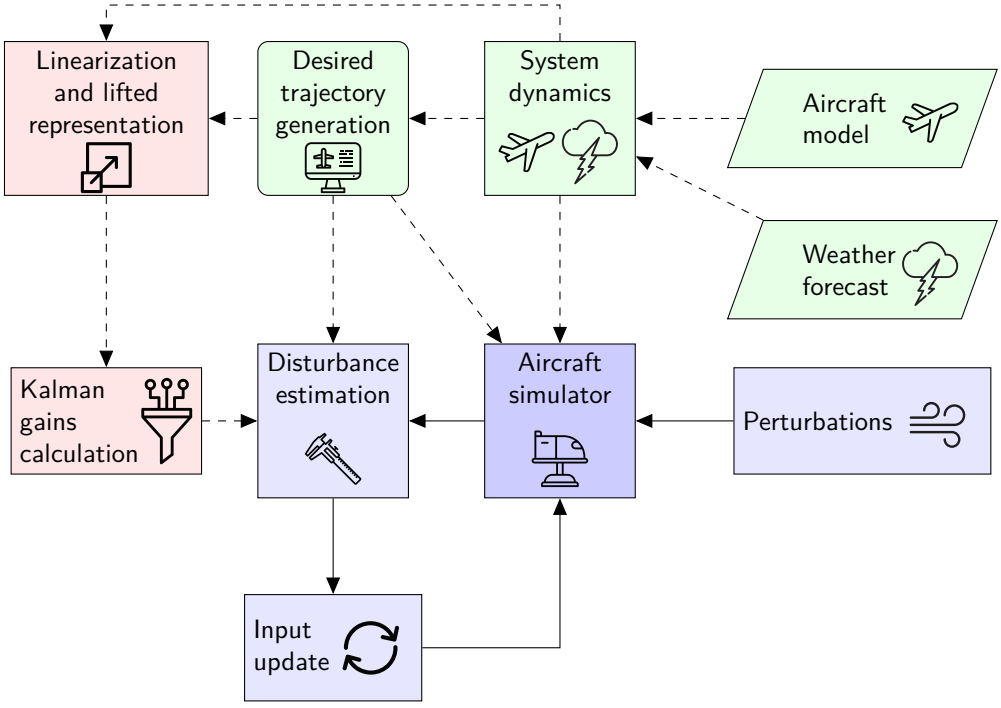


Figure 3.1: Flow chart summarizing the ILC scheme. The green blocks consist of the system dynamics and the trajectory to be followed, which are determined prior to the ILC initialization. The blocks depicted in pink color represent the additional calculations needed to initialize the algorithm, which are the lifted representation of the system and the Kalman gains. Finally, the blue blocks form the simulation and ILC stages of the iterative loop, in which the perturbed flight is simulated, prompting the estimation of the disturbances and the input update for the following aircraft.

The flowchart in figure 3.1 outlines the different elements involved in the ILC method and their connections. The ILC algorithm is summarized in Algorithm 1. The desired output trajectory, $\mathbf{y_d}$, the state, $\mathbf{x_d}$, and the corresponding input, $\mathbf{u_d}$, are loaded in Step 1 from a file generated in the trajectory planning phase. In the simulation stage, using the latest feed-forward input generated by the ILC algorithm, the corresponding reference trajectory is fed into the aircraft controller and the resulting output is generated by the flight simulator (in the case of direct ILC the input signal is directly plugged into the aircraft control). In the ILC stage, the resulting error vector is provided to a Kalman filter that estimates the tracking error. Finally, based on the estimated tracking error, the update of the learning algorithm is executed in Step 10, which determines a new control input or a reference trajectory to be tracked in the next iteration, optimally compensating for the estimated error.

Algorithm 1 Summary of the ILC algorithm

INITIALIZATION

1: Load the output trajectory to be followed and its corresponding nominal state and input $(\mathbf{x_d}(t), \mathbf{u_d}(t), \mathbf{y_d}(t))$.
2: Linearize the system about the desired trajectory and convert it into a discrete-time system.
3: Note the system as a lifted domain representation (3.8), (3.11).
4: Enter the Kalman filter initial parameters and compute the Kalman gains (3.14).

AT EACH ITERATION

• **Simulation stage**

5: Set the simulated aircraft to the initial position.
6: Enter the reference trajectory into the feedback controller, or directly the control input.
7: Simulate the flight with disturbances.

• **ILC stage**

8: Compare the output of the flight simulator with the desired trajectory.
9: Estimate the tracking error (3.13), (3.15).
10: Update the reference trajectory or the control input (3.17).

3.3 Experimental results

This section presents several numerical experiments, which show the effectiveness of the ILC method applied to tracking the trajectory of an Airbus A320-231 aircraft. These experiments are carried out in a simulated environment in which different configurations are considered, involving the direct or indirect approach of the ILC algorithm, the use of a feedback controller in the latter, the perturbations affecting the flight, and the measurable state and output variables.

Experiments 1 and 2, in which a direct ILC approach is used, are carried out in a simple scenario to show the improvement of precision in trajectory tracking in a basic configuration of the environment. Experiments 3 and 4 make use of IILC, and are performed in a more realistic

scenario, proving that tracking performance is bettered also in a more accurate picture of a real operation.

3.3.1 Direct iterative learning control

The direct ILC approach is used as summarized in Figure 3.2, where precision in trajectory tracking is improved by pure feed-forward adaptation of the control input. This ILC method is applied in two simulated experiments:

- Experiment 1: ILC for precise trajectory tracking in a CCO.
- Experiment 2: ILC for precise trajectory tracking in a CDO.

A feasible trajectory with its corresponding nominal input is the starting point of the iterative learning algorithm. In both experiments, the feasible trajectories to be followed by the simulated aircraft are generated as explained in Section 2.2. Thus, not only feasible but also optimal aircraft trajectories are obtained together with the optimal control inputs to steer the aircraft along them. In both experiments, the aircraft model and flight envelope used to generate the trajectories are those described in Section 2.1.1, assuming zero wind. The modelling errors and perturbations are introduced in the flight simulator by using different parameters as the ones provided by BADA, and including Gaussian noise in the measured variables. Additionally, Gaussian noise is also added to the input controls generated by the ILC algorithm. All state variables are assumed to be measurable. As said before, although the ILC algorithm is able to tackle 4D trajectories, in this work, for the sake of clarity, only the corresponding longitudinal trajectories are reported.

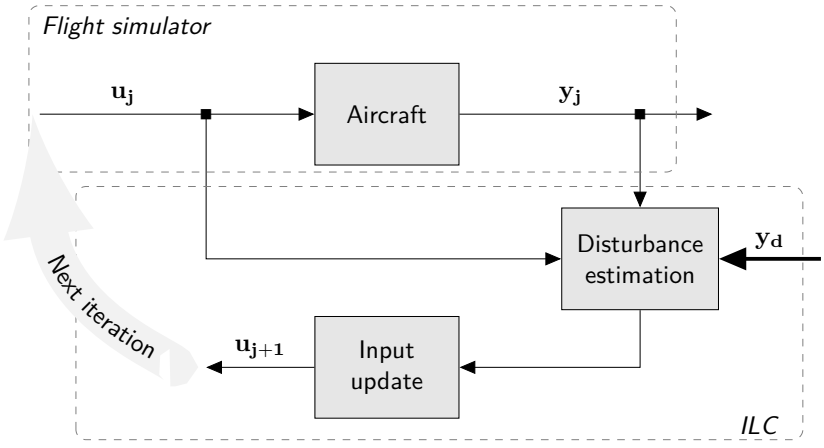


Figure 3.2: Direct ILC scheme. The simulated environment is composed of two main blocks: a flight simulator, which includes the aircraft model, and an ILC controller, composed of an estimator of the disturbances acting on the aircraft and a nonlinear programming solver that provides the updated control input for the next iteration.

Experiment 1: ILC for precise trajectory tracking in a CCO

In this section, the results of the application of the ILC scheme to precise aircraft trajectory tracking in a CCO are addressed. A state weighting matrix is introduced in the ILC algorithm to give the position state variables (h_e and x_e) greater importance than the rest, as described in (3.19). Analogously, the initial part of the trajectory is given more weight to ensure the fulfilment of the waypoints' requirements of the SID procedure to which the CCO is associated.

The desired path associated with the desired trajectory of the aircraft to perform a CCO, obtained in the trajectory planning phase, is shown in dashed black line in Figure 3.3. In the first iteration, the corresponding input is applied to the first aircraft and the resulting trajectory is generated by the flight simulator. Due to modeling and disturbance errors considered in the flight simulator, the resulting path falls far below the desired one, initiating the cruise phase more than 1000 m below the planned cruise height. The ILC scheme rapidly learns from the first executions achieving a very precise tracking of the designed CCO trajectory after only three iterations. This means that using the ILC scheme, the third aircraft is able to compensate for the recurring disturbances and follow the desired trajectory with high precision.

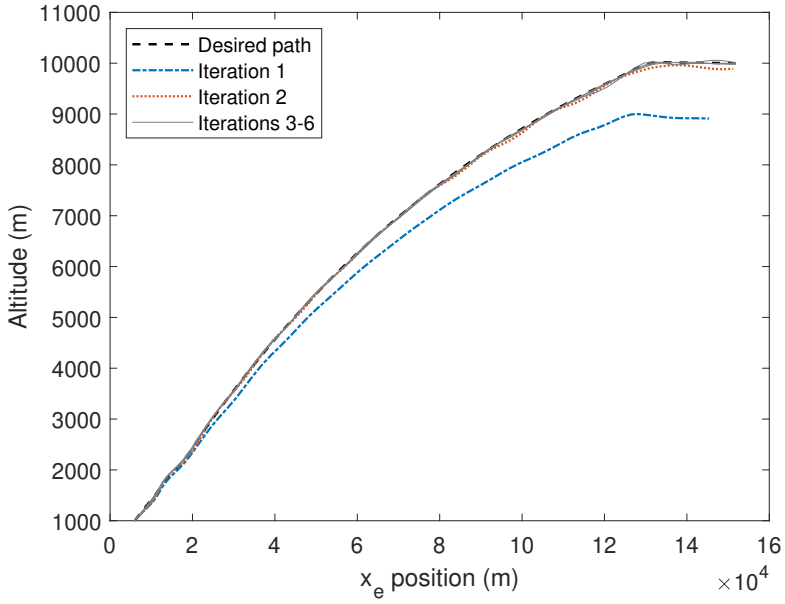


Figure 3.3: Experiment 1: Evolution of the CCO climb path $x_e - h_e$ over iterations using ILC.

As shown in Figure 3.4, the control inputs, namely the thrust and the lift coefficient, converge after few iterations. Despite this convergence, both the trajectory and the input vary over iterations due to non-repetitive disturbances affecting the flight, which can be partially compensated by the feedback trajectory tracking controller of the aircraft, as described in Section 3.3.2.

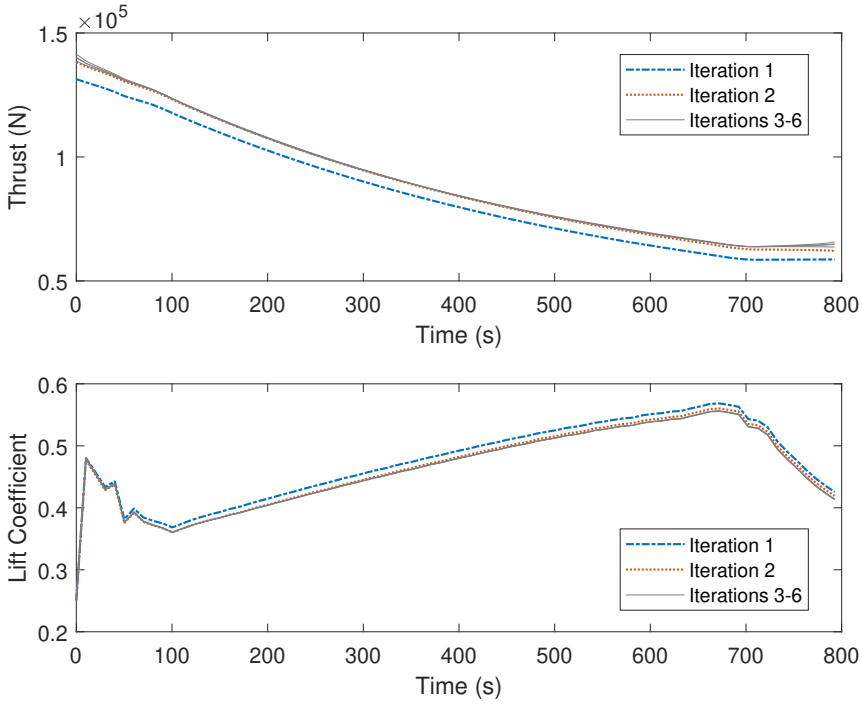


Figure 3.4: Experiment 1: Evolution of the CCO thrust and lift coefficient over iterations using ILC.

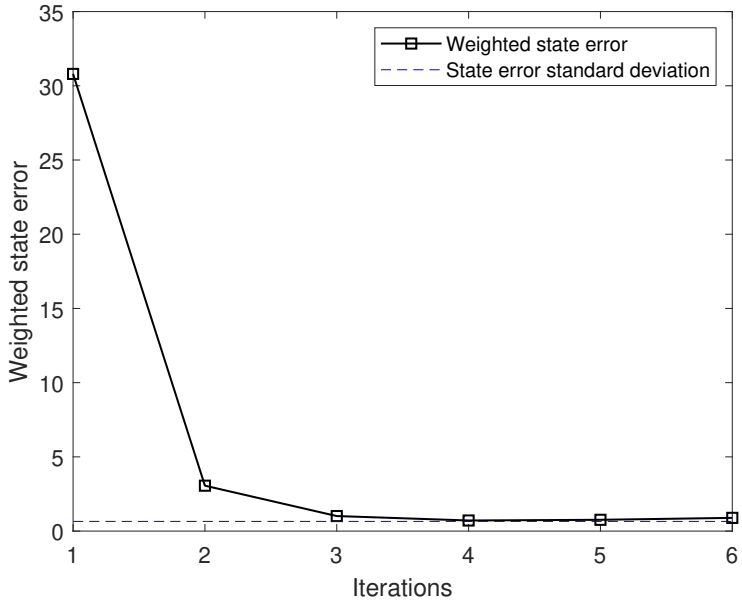


Figure 3.5: Experiment 1: Evolution of the CCO weighted state error over iterations using ILC.

The weighted state error, which is shown in Figure 3.5, allows the learning performance to be evaluated over iterations. It is calculated as follows:

$$e_{ws,j} = \|\boldsymbol{S}\boldsymbol{y}_j\|_2, \tag{3.21}$$

where $\boldsymbol{S}$ is the weighted scaling matrix of the state variables. The state error standard deviation characterizes the system noise level, scaled by $\boldsymbol{S}$. It is obtained from the variations in the trajectory when applying the same input to the aircraft several times. As expected, since the ILC scheme is intended to compensate for repetitive disturbances, the weighted state error converges to the system noise level, and not to zero, due to non-repetitive disturbances.

Experiment 2: ILC for precise trajectory tracking in a CDO

As in the previous case, the ILC algorithm applied to the tracking of the CDO includes a state weighting matrix to give greater importance to the position state variables (h_e and x_e). Here, the final part of the trajectory is given more weight to face the final approach at the correct position and speed.

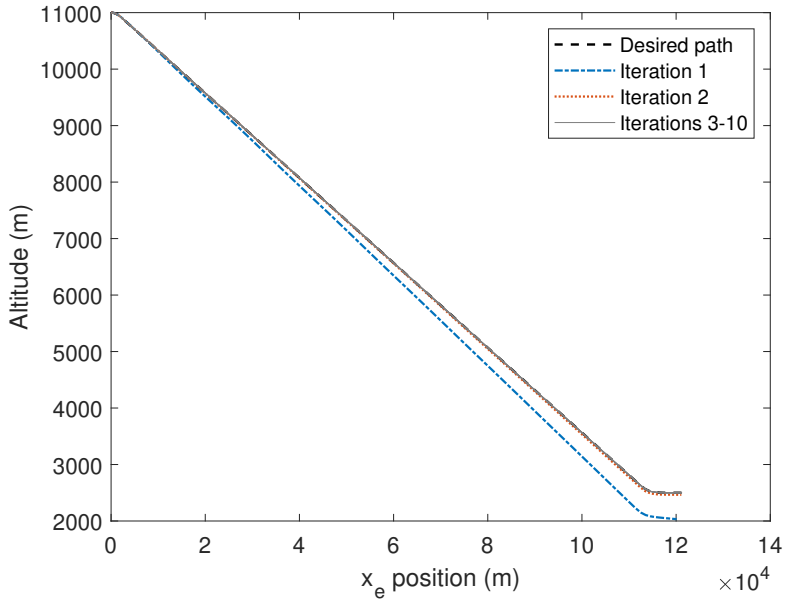


Figure 3.6: Experiment 2: Evolution of the CDO path $x_e - h_e$ over iterations using ILC.

Figure 3.6 shows the evolution of the tracking of the desired CDO path over iterations. This desired path, similar to the CDO obtained in the trajectory planning described in Section 2.2, is shown in dashed black line. The input applied in the first execution is the nominal input obtained in the desired trajectory generation, and the resulting trajectory tracked by the aircraft is generated by the flight simulator. As shown in Figure 3.6, in the first execution the path followed by the simulator falls far below the desired one due to modeling and disturbance errors, but the ILC scheme rapidly learns from the first executions, compensating for the recurring

disturbances and achieving a very precise tracking of the designed CDO trajectory after only two iterations.

As shown in Figure 3.7, the main control action is carried out by the thrust control input, which tends to converge over iterations whereas the lift coefficient input remains almost unchanged. Finally, the weighted state error depicted in Figure 3.8 shows a significant decrease of the tracking error in the first two iterations, and the convergence to the system noise level from the third iteration.

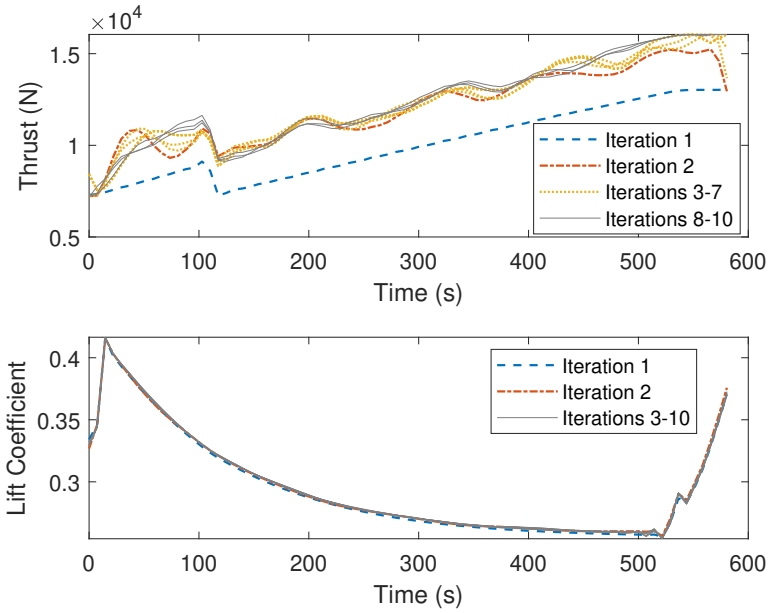


Figure 3.7: Experiment 2: Evolution of the CDO thrust and lift coefficient over iterations using ILC.

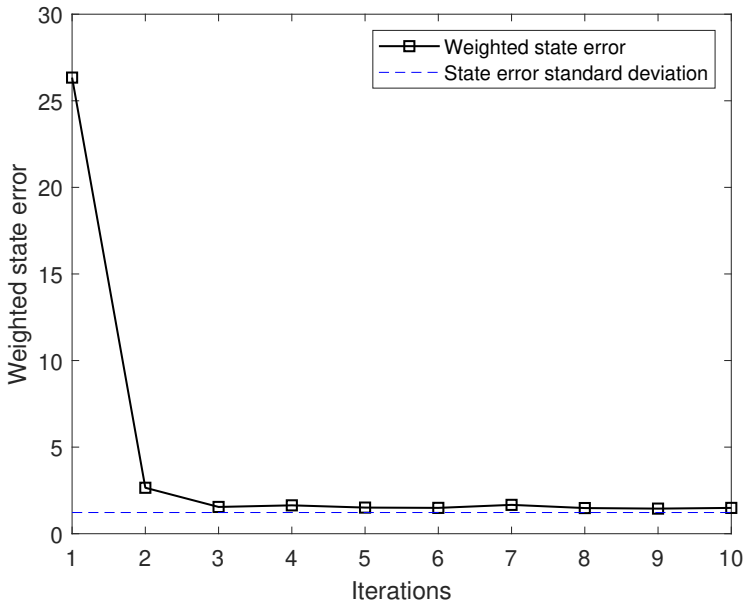


Figure 3.8: Experiment 2: Evolution of the CDO weighted state error over iterations using ILC.

3.3.2 Indirect iterative learning control

The IILC method has been applied to precise aircraft trajectory tracking in the following two simulated experiments:

- Experiment 3: IILC for precise trajectory tracking in a CCO.
- Experiment 4: IILC for precise trajectory tracking in a CDO.

The IILC approach is summarized in Figure 3.9, where the use of an ILC algorithm generates a new reference trajectory for the aircraft's underlying trajectory tracking feedback controller, instead of directly acting on the aircraft's control inputs. Here, a PI output-feedback controller is considered as the aircraft's baseline controller. As in the previous experiments, the feasible trajectories to be followed by the simulated aircraft are generated as in Section 2.2, and the aircraft model and flight envelope employed to generate the trajectories are those described in Section 2.1.1, assuming zero wind. The simulated environment in which the experiments are conducted is more accurate than in Experiments 1 and 2: a more realistic wind model is used, in which wind turbulence is considered, a constrained Kalman filter for smooth disturbance estimation is employed, which is able to reject the turbulence effect, and an aircraft model is formulated, in which some of the state variables are assumed to be not directly measurable, being the output variables the indicated airspeed, the Mach number, the horizontal and vertical position, and the altitude rate.

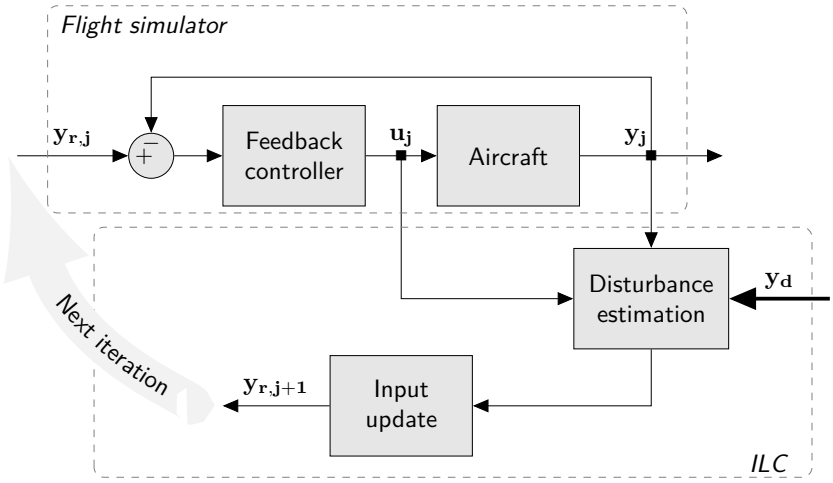


Figure 3.9: Indirect ILC scheme. The simulated environment is composed of two main blocks: a flight simulator, which includes the aircraft model and a feedback controller, and an ILC controller, composed of an estimator of the disturbances acting on the aircraft and a nonlinear programming solver that provides the updated reference trajectory for the next iteration.

Experiment 3: IILC for precise trajectory tracking in a CCO

In this experiment, the trajectory to be followed by the simulated aircraft is a CCO. In this case, the ILC algorithm is tuned to give greater importance to the part of the trajectory comprising the waypoints than the rest of it by giving more weight to these points in the scaling matrix.

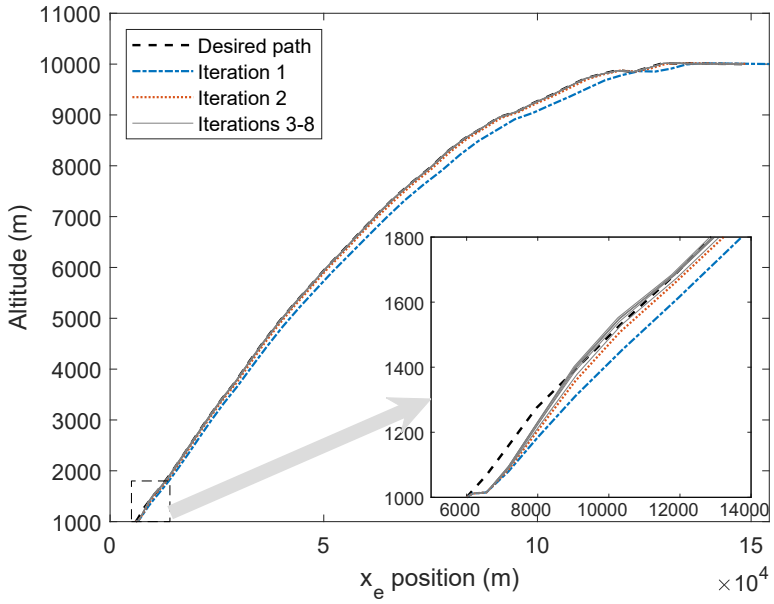


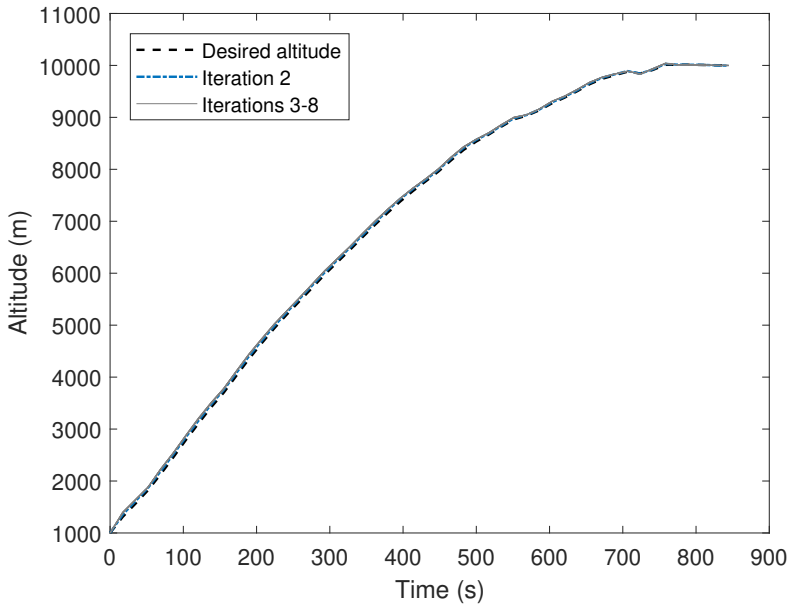
Figure 3.10: Experiment 3: Evolution of the CCO path $x_e - h_e$ over iterations using IILC.

Figure 3.10 shows the evolution of the tracking of the desired path to perform the CCO obtained in the trajectory planning phase. The desired path is drawn as a dashed black line. In the first iteration, the reference trajectory fed into the flight simulator corresponds to the altitude and Mach number obtained in the trajectory planning phase. After each execution, the reference trajectory is updated and the simulated aircraft flies along the paths plotted in the figure.

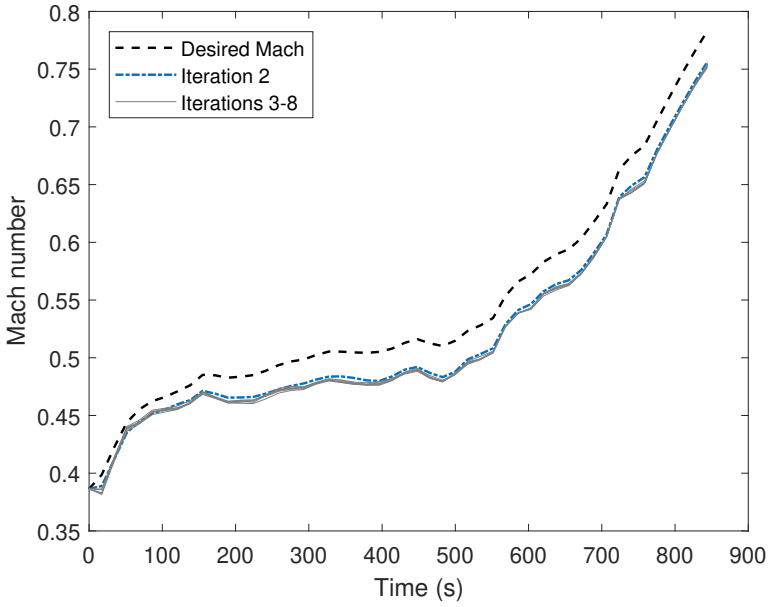
As shown in Figure 3.10, the path followed by the aircraft in the first iteration when performing the CCO remains below the desired one until it reaches the cruise level, because of modeling and disturbance errors considered in the flight simulator. Due to the importance of flying through the waypoints with precision, the initial part of the path is shown magnified in the box on the right-hand side of the figure, where it can be observed that the aircraft tracks the desired path with precision from the second iteration. The deviation corresponding to the first portion of the path is due to the dynamics of the feedback controller.

It has already been mentioned that the control paradigm proposed here is nonintrusive with respect to the underlying aircraft feedback controller since it calculates, at each iteration, a reference trajectory for the following aircraft rather than a control input to track the desired

trajectory precisely. As explained in Section 2.1.2, the reference trajectory is commanded to the feedback controller of the flight simulator by feeding it with the reference altitude and Mach number, which are depicted in Figure 3.11, showing that the speed is adjusted in pursuit of precision in the aircraft's position, while the commanded altitude barely varies along the iterations.



(a) Reference CCO altitude over iterations.



(b) Reference CCO Mach number over iterations.

Figure 3.11: Experiment 3: Evolution of the CCO reference trajectory over iterations using IILC.

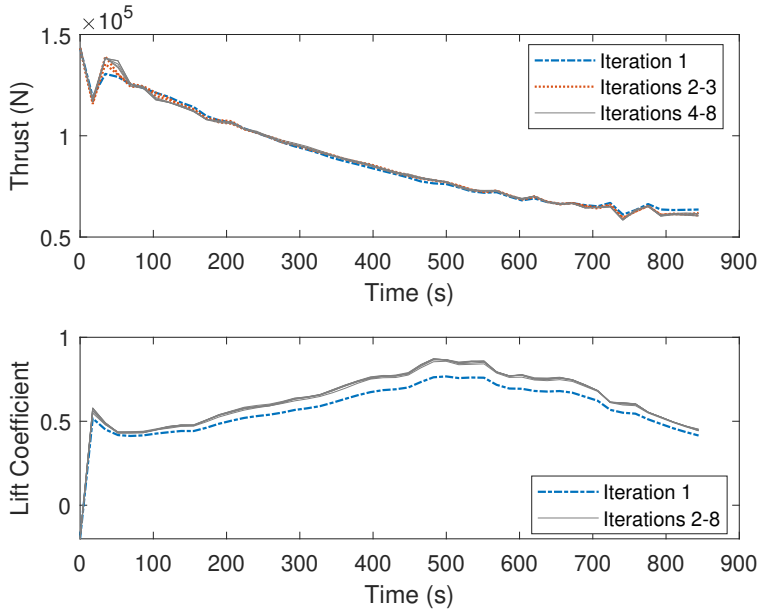


Figure 3.12: Experiment 3: Evolution of the CCO thrust and lift coefficient over iterations using IILC.

As shown in Figure 3.12, the reference Mach number in the CCO, depicted in Figure 3.11b, is reduced mainly by increasing the lift coefficient input. It can be seen that the control inputs tend to converge in few iterations. As in the previous experiments, slight variations remain both in the trajectory and the input due to nonrepetitive disturbances that are not completely compensated by the baseline controller, although these variations are smaller than in Experiments 1 and 2.

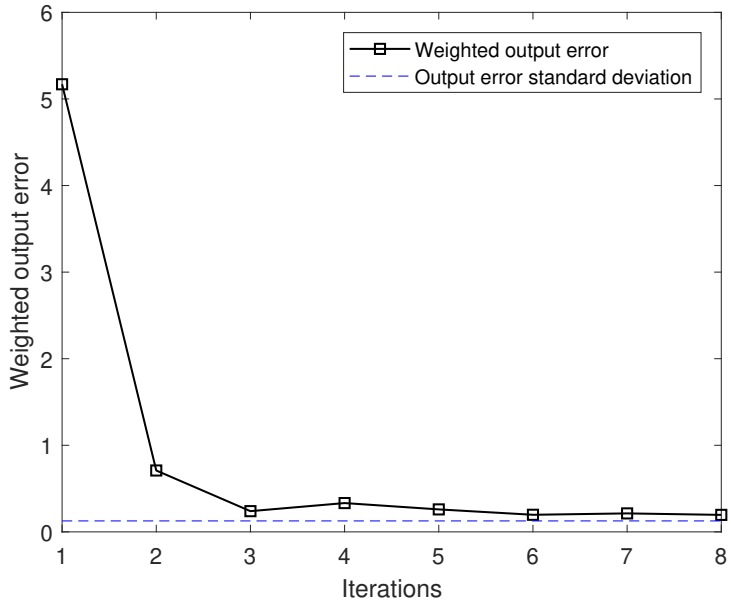


Figure 3.13: Experiment 3: Evolution of the CCO weighted output error over iterations using IILC.

The weighted output error shown in Figure 3.13 is calculated, analogously to the weighted state error in (3.21), as

$$e_{wo,j} = \|\boldsymbol{S}_o \boldsymbol{y}_j\|_2, \tag{3.22}$$

where $\boldsymbol{S}_o$ is the weighted scaling matrix of the output variables. The output error standard deviation characterizes the system noise level, scaled by $\boldsymbol{S}_o$.

Although the planned trajectory is feasible in calm wind conditions, it is not when adding wind perturbations. Thus, precision in position is prioritized over precision in the other state variables, and the initial part of the trajectory is given more importance, penalizing other segments. Therefore, the weighted output error is significantly reduced but does not converge to zero, although it remains near the output error standard deviation, which can be viewed as a lower bound for the achievable tracking error.

Experiment 4: IILC for precise trajectory tracking in a CDO

In this experiment, the trajectory to be followed by the simulated aircraft is a CDO. To ensure that the aircraft reaches the final conditions with precision, the ILC algorithm is tuned to give greater importance to the final part of the trajectory than the rest of it.

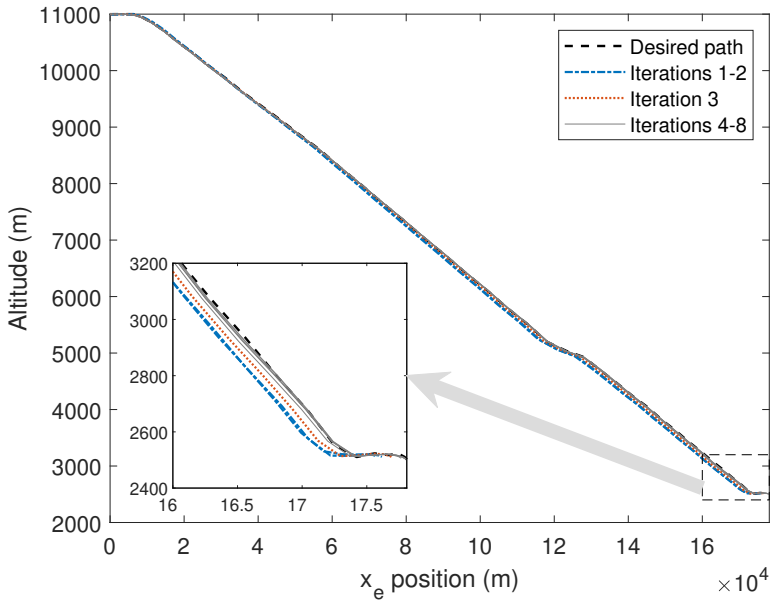
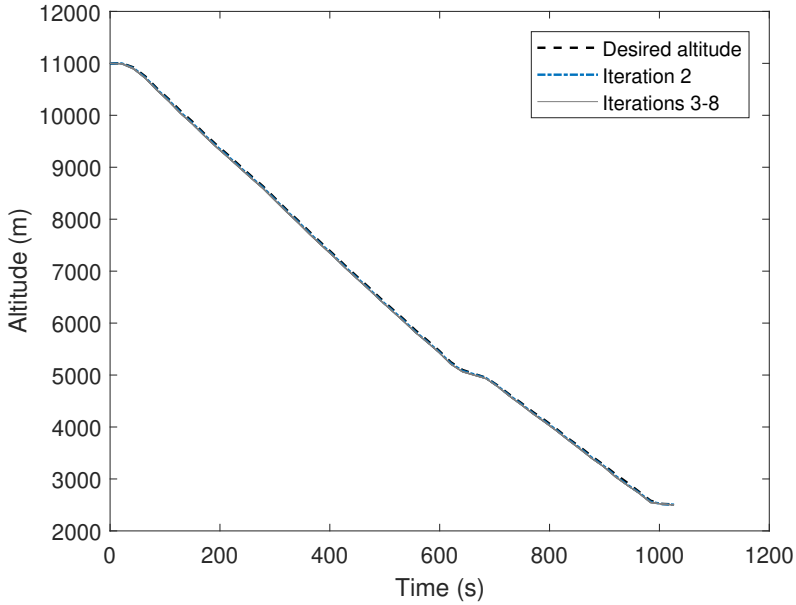


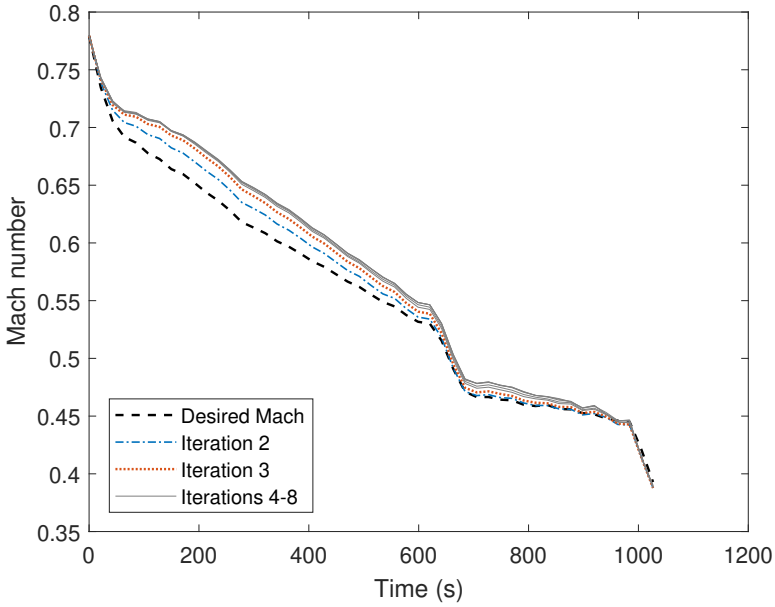
Figure 3.14: Experiment 4: Evolution of the CDO path $x_e - h_e$ over iterations using IILC.

Applying the desired CDO trajectory as the reference trajectory and feeding the flight simulator with the corresponding altitude and Mach number results into the deviations shown in blue in Figure 3.14, where the dashed black line represents the desired trajectory. In this case, it is important that the final part of the path be followed precisely to undertake the approach phase of the flight. This part is shown magnified in the box on the left-hand side

of the figure. Again, the IILC scheme rapidly learns from the first executions, compensating for the recurring disturbances and achieving very precise tracking of the final segment of the CDO trajectory after only two iterations.



(a) Reference CDO altitude over iterations.



(b) Reference CDO Mach number over iterations.

Figure 3.15: Experiment 4: Evolution of the CDO reference trajectory over iterations using IILC.

As in Experiment 3, the IILC computes a new reference trajectory at each iteration and the reference altitude and Mach number are fed into the feedback controller of the flight simulator. These reference variables are depicted in Figure 3.15 for the CDO case and, as in the CCO experiment, the speed is adjusted to achieve the required precision in the aircraft's position, while the commanded altitude remains almost unaltered along the iterations.

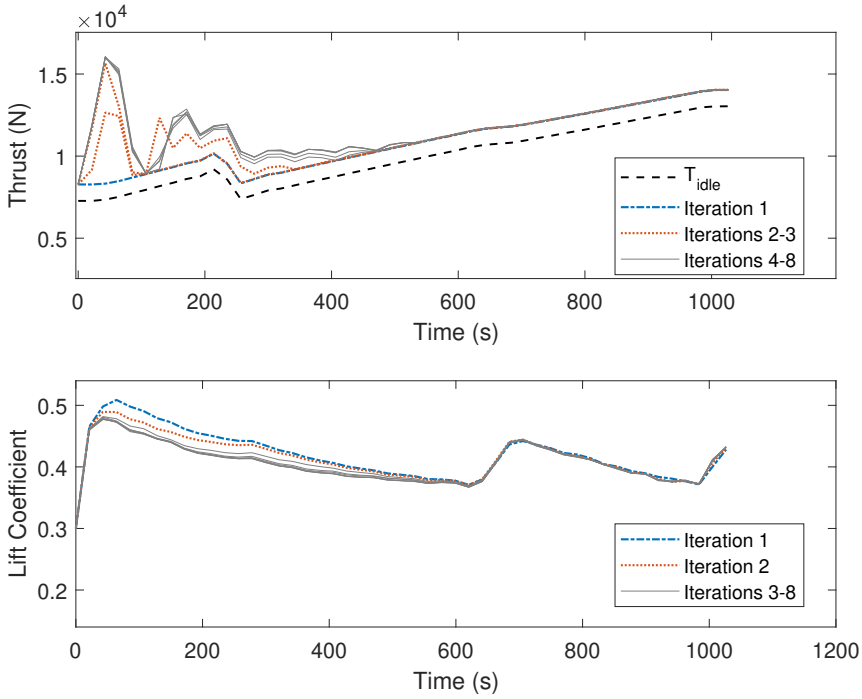


Figure 3.16: Experiment 4: Evolution of the CDO thrust and lift coefficient over iterations using IILC.

As shown in Figure 3.16, the main control action is carried out by the thrust control input, which increases although remaining near idle, while the lift coefficient is reduced. The initial bump in the thrust is due to the dynamics of the feedback controller.

Figure 3.17 shows the evolution of the weighted output error along the iterations. The weighted output error does not converge to zero since the planned trajectory becomes unfeasible when wind perturbations are integrated into the simulation. Furthermore, the limitations in the thrust control to remain near idle make it more difficult to guide the aircraft along the desired trajectory. Other state variables have been penalized in pursuit of precision in the position of the aircraft, especially in the final segments of the trajectory, where aircraft speed has been also prioritized. Despite the aforementioned limitations, the weighted output error is reduced by 70% in only three iterations.

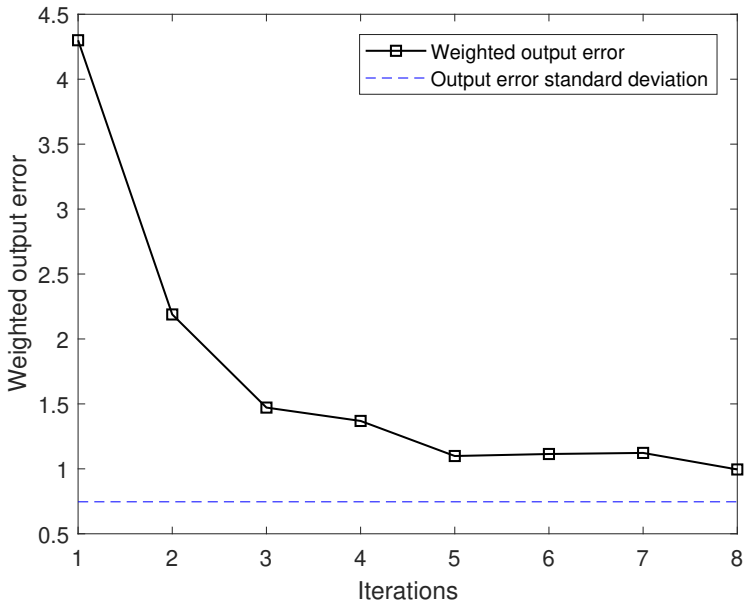


Figure 3.17: Experiment 4: Evolution of the CDO weighted state error over iterations using IILC.

3.4 Conclusion

The experiments carried out in this chapter prove that the optimization-based ILC method is very effective in compensating for the recurrent disturbances affecting the aircraft performance. Precise trajectory tracking is achieved in few iterations in continuous climb and descent operations using both the direct and the indirect ILC approaches. Moreover, in the latter, a new reference trajectory is generated instead of a control input, making it nonintrusive with respect to the aircraft's underlying trajectory tracking controller, and therefore suitable for real operation. In both cases optimality is pursued in both the estimation of the disturbances and the updating of the new input or reference trajectory, and the learning behavior can be adapted depending on the flight phase and precision requirements by assigning different weights to the variables and segments of the trajectory.

4

Gaussian Process Regression Applied to ILC

This chapter proposes a recursive Gaussian process regression in combination with an optimization-based iterative learning control algorithm to estimate and predict disturbances and model uncertainties affecting a flight. As exposed in the previous chapter, the ILC algorithm is divided into two steps: estimation of the model error and disturbances affecting the flight, and update of the control inputs so that the subsequent aircraft intending to fly the same planned trajectory will follow it with greater precision than the previous ones. In this chapter, the first step is undertaken by a recursive Gaussian process regression method, which estimates and predicts perturbations and model errors with no need for prior knowledge about their dynamics and with low computational cost. This method is tested on a simulated commercial aircraft performing a CCO and compared to an iterative learning algorithm using a Kalman filter estimator in a similar scenario.

4.1 Introduction

Most ILC approaches rely on the system model as a basis of the learning algorithm, making its performance very sensitive to model uncertainties. In [24], the estimation step is performed using a Kalman filter, as shown in Chapter 3. The Kalman filter approach to recursive estimation has seen extensive use in aerospace applications since its introduction in this area [32] and continues to form the basis for many new filtering approaches to the state estimation problem [33], [34]. However, the underlying model dynamics must be known for it to be possible to tune the Kalman filter, and it is suitable only for slight linear changes in the unmodeled dynamics. Enhanced variants of the Kalman filter, such as the extended Kalman filter, would allow nonlinear changing dynamics of the disturbances to be estimated, but would also require some knowledge of its behavior. In the new approach presented in this chapter, the Kalman filter is replaced with a recursive GPR method, aiming to provide an alternative ILC approach in which

no prior knowledge of the disturbances is needed, meaning that no prior tuning is required, and which is capable of estimating the disturbances even if they vary between iterations, despite not knowing the dynamics of their variations.

Gaussian processes provide a nonparametric statistical learning of nonlinear dynamic systems from noisy data, which are characterized by covariance functions that usually have a set of hyperparameters [35]. Regression methods based on Gaussian processes are used in many areas of application, such as machine learning [36], [37], signal processing [38], and control engineering [39], [40].

The main drawback of Gaussian processes is the considerable computational cost when working with large data sets. Multiple approaches have been proposed in order to reduce the computational burden, for example in [41], [42], and [43], where the training data is reduced to a number of so-called inducing points. Another difficulty is the determination of the hyperparameters, which are usually learned by optimizing the marginal likelihood [44].

In [45], an online procedure for updating Gaussian process parameters is proposed, in which the regression is performed on a set of basis vectors with low computational and memory demands, simultaneously learning the hyperparameters. This approach is adapted in this thesis to be included in the estimation and prediction steps of the ILC algorithm. To do so, the sequential nature of ILC is taken into account in the time series prediction of the disturbances affecting the flight, assuming that the behavior of these disturbances is more correlated to the recent past observations than to the distant past ones. Therefore, the location of the basis vectors is updated at each iteration replacing the oldest one with the new data estimation.

The main contributions of this chapter are threefold:

(i) A novel framework for precise aircraft trajectory tracking is introduced, which includes a GPR within an ILC scheme.

(ii) This learning GPR estimates its hyperparameters online. The disturbances and model uncertainties affecting a flight are also estimated without the need for any prior knowledge about them.

(iii) The developed framework is validated and tested over a range of examples with different uncertainties.

The proposed framework is flexible and has significant advantages compared to the Kalman filter used as an estimator in the previous chapter, since it learns from the data. In contrast, Kalman filters need to be correctly tuned using a priori knowledge of the dynamics to be estimated, and it is assumed that they remain unchanged between iterations in the prediction step. The computational cost required by the GPR is significantly reduced by using a recursive approach, which is updated at each iteration dismissing the oldest data and incorporating the newest observations. This makes it appealing for real-time control.

4.2 Gaussian process regression

4.2.1 Problem formulation

For the GPR, it is assumed that a set of data $\mathcal{D} = \{(\boldsymbol{v}_1, y_1), \ldots, (\boldsymbol{v}_n, y_n)\}$ is drawn from the noisy process

$$y_i = g(\boldsymbol{v}_i) + \epsilon,$$

where the column vectors $\boldsymbol{v}_i \in \mathbb{R}^{n_v}$ are the inputs, $y_i \in \mathbb{R}$ are the observations or outputs, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is a zero-mean Gaussian noise with variance σ^2. A Gaussian Process (GP) is used to infer the latent function g from the set of n past observations $\mathcal{D}$, also called training data. For $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ being either the training or the testing input data vectors, the GP is completely defined by a mean function $\mu(\boldsymbol{v}_i) \equiv E\left[g(\boldsymbol{v}_i)\right]$, which specifies the expected output value, with $E[\cdot]$ being the mathematical expectation operation, and a positive semi-definite covariance function $k(\boldsymbol{v}_i, \boldsymbol{v}_j) \equiv \mathrm{COV}\left\{g(\boldsymbol{v}_i), g(\boldsymbol{v}_j)\right\}$, which specifies the covariance between pairs of inputs $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ and is often called a kernel. Typical examples are the zero-mean function $\mu(\boldsymbol{v}_i) = 0$ and the squared exponential kernel

$$k(\boldsymbol{v}_i, \boldsymbol{v}_j) = \alpha^2 \exp\left(-\frac{1}{2}(\boldsymbol{v}_i - \boldsymbol{v}_j)^T \boldsymbol{\Lambda}^{-1}(\boldsymbol{v}_i - \boldsymbol{v}_j)\right),$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix of the characteristic length-scales for each input dimension, which determine the length of the "wiggles" in the function, and α^2 is the variance of the latent function g. Such parameters of the mean and covariance functions together with the noise variance σ^2 are called the hyperparameters of the GP and are here grouped in the vector $\boldsymbol{\eta}$.

4.2.2 Prediction

Based on the training data $\mathcal{D} = \{(\boldsymbol{v}_1, y_1), \ldots, (\boldsymbol{v}_n, y_n)\}$, the GP framework described above can be used to predict the function values $\boldsymbol{g}_* = [g(\boldsymbol{v}_{n+1}), \ldots, g(\boldsymbol{v}_{n+m})]^T$ at the arbitrary inputs $\boldsymbol{v}_* = [\boldsymbol{v}_{n+1}, \ldots, \boldsymbol{v}_{n+m}]^T$. Given this observations, the collection of functions $\boldsymbol{g} = [g(\boldsymbol{v}_1), \ldots, g(\boldsymbol{v}_n)]^T$ at the training inputs $\boldsymbol{v} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]^T$ follows a multivariate Gaussian distribution with mean function $\boldsymbol{\mu}(\boldsymbol{v}) = [\mu(\boldsymbol{v}_1), \ldots, \mu(\boldsymbol{v}_n)]^T$ and covariance matrix $\boldsymbol{K}(\boldsymbol{v}, \boldsymbol{v}) \in \mathbb{R}^{n \times n}$, the $(i, j)-$th element of which is $\boldsymbol{K}_{ij} = k(\boldsymbol{v}_i, \boldsymbol{v}_j)$.

The joint distribution of the observed data $\boldsymbol{y}$ and the predicted function values $\boldsymbol{g}_*$ is given by

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{g}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}(\boldsymbol{v}) \\ \boldsymbol{\mu}(\boldsymbol{v}_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{v}, \boldsymbol{v}) + \sigma^2 \boldsymbol{I} & \boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v})^T \\ \boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}) & \boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}_*) \end{bmatrix}\right), \tag{4.1}$$

where $\boldsymbol{\mu}(\boldsymbol{v}_*) = [\mu(\boldsymbol{v}_{n+1}), \ldots, \mu(\boldsymbol{v}_{n+m})]^T$, $\boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}) \in \mathbb{R}^{m \times n}$ has the $(i, j)-$th element defined as $\boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v})_{ij} = k(\boldsymbol{v}_{n+i}, \boldsymbol{v}_j)$, and $\boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}_*) \in \mathbb{R}^{m \times m}$ has the $(i, j)-$th element defined as $\boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}_*)_{ij} = k(\boldsymbol{v}_{n+i}, \boldsymbol{v}_{n+j})$.

Thus, the predictive distribution conditioned by the data set $\mathcal{D}$ is given by a Gaussian distribution with mean and variance

$$\begin{aligned}\boldsymbol{\mu}_g(\boldsymbol{v}_*) &= \boldsymbol{\mu}(\boldsymbol{v}_*) + \boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v})^T(\boldsymbol{K}(\boldsymbol{v}, \boldsymbol{v}) + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{\mu}(\boldsymbol{v})),\\ \boldsymbol{\sigma}_g^2(\boldsymbol{v}_*) &= \boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}_*) - \boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v})^T(\boldsymbol{K}(\boldsymbol{v}, \boldsymbol{v}) + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}).\end{aligned} \tag{4.2}$$

4.2.3 Recursive Gaussian process with learning

The GP prediction depends on the inverse of $\boldsymbol{K}(\boldsymbol{v}, \boldsymbol{v})$, which scales with $\mathcal{O}(n^3)$. For large data sets this is computationally unfeasible. The recursive GPR proposed in [45] aims to perform all calculations on a sparse representation of the GP formed by a set of $\ell \ll n$ so-called basis vectors, relying on this basis vectors for estimating the latent function and learning the hyperparameters on-line from data.

On-line regression

The basis vectors are located at $\boldsymbol{v} \equiv [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_\ell]^T$, store local estimates $\boldsymbol{g} \equiv g(\boldsymbol{v})$ of the latent function, and are updated on-line with n_t new observations, $\boldsymbol{y}_t$, at inputs $\bar{\boldsymbol{v}}_t \equiv [\boldsymbol{v}_{t,1}, \ldots, \boldsymbol{v}_{t,n_t}]^T$ and time steps $t = 0, 1, \ldots$. The basis vectors are fixed in number and location for each time step. Assuming known hyperparameters, the goal is to calculate recursively the posterior distribution $p(\boldsymbol{g}|\boldsymbol{y}_{1:t})$, with $\boldsymbol{y}_{1:t} \equiv [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t]^T$, by updating the prior distribution of $\boldsymbol{g}$ from the distribution on the previous step $t-1$, namely $p(\boldsymbol{g}|\boldsymbol{y}_{1:t-1})$.

For deriving a recursive algorithm, the desired posterior distribution is expanded according to

$$p(\boldsymbol{g}|\boldsymbol{y}_{1:t}) = \int c_t \cdot p(\boldsymbol{y}_t|\boldsymbol{g}, \bar{\boldsymbol{g}}_t) \cdot p(\bar{\boldsymbol{g}}_t|\boldsymbol{g}) \cdot p(\boldsymbol{g}|\boldsymbol{y}_{1:t-1}) d\bar{\boldsymbol{g}}_t, \tag{4.3}$$

where $\bar{\boldsymbol{g}}_t = [g(\boldsymbol{v}_{t,1}), \ldots, g(\boldsymbol{v}_{t,n_t})]^T$, and c_t is a normalization constant.

Learning

If the hyperparameters, $\boldsymbol{\eta}$, are unknown, they are estimated simultaneously with the values of the latent function at the basis vectors. This is done calculating a joint posterior distribution $p(\boldsymbol{z}_t|\boldsymbol{y}_{1:t})$, where $\boldsymbol{z}_t^T \equiv [\boldsymbol{g}^T, \boldsymbol{\eta}_t^T]$ is the joint hidden state.

A detailed explanation of the recursive GP with the learning algorithm can be found in [45].

4.3 Recursive GPR estimation and prediction in ILC

The goal is to estimate the disturbances, $\boldsymbol{d}_j$, affecting the aircraft at each iteration of the task, that is to say at each intent of flying the planned trajectory, and, ultimately, to predict the disturbances in the following iteration, $\boldsymbol{d}_{j+1}$.

The Kalman filter estimation proposed in [24] assumes that the disturbances are almost constant, that is to say

$$\boldsymbol{d}_j = \boldsymbol{d}_{j-1} + \boldsymbol{\omega}_{j-1}, \tag{4.4}$$

where $\boldsymbol{\omega}_j$ is a white noise, namely a zero-mean Gaussian variable.

GPR allows for non-linear changes in the disturbances between iterations. The system dynamics is separated into two components, a known function of the state and control input, $\boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{u}_j)$, and an unknown function, $\boldsymbol{g}$, which represents an unknown, deterministic dynamics that is not captured by the a priori model $\boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{u}_j)$. The function $\boldsymbol{g}$ depends on τ_j, which is the time elapsed between the first and the $j-$th iteration, to capture the evolution of the unknown dynamics along time. The state and output at each iteration are therefore modeled as

$$\begin{aligned} \boldsymbol{x}_j &= \boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{u}_j) + \boldsymbol{g}(\tau_j), \\ \boldsymbol{y}_j &= \boldsymbol{h}(\boldsymbol{x}_j, \boldsymbol{u}_j) + \boldsymbol{\epsilon}_j, \end{aligned} \tag{4.5}$$

where, using the lifted representation described in Section 3.2.1, $\boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{u}_j) = \boldsymbol{F}\boldsymbol{u}_j$, the disturbances are captured in the unknown function $\boldsymbol{g}(\tau_j)$, and $\boldsymbol{h}(\boldsymbol{x}_j, \boldsymbol{u}_j) = \boldsymbol{G}\boldsymbol{x}_j + \boldsymbol{H}\boldsymbol{u}_j$. In this chapter, it is assumed that full-state information is available and, without loss of generality, $\boldsymbol{y}_j = \boldsymbol{x}_j + \boldsymbol{\epsilon}_j$.

Finally, the disturbance is obtained in the following format:

$$\boldsymbol{d}_j = \boldsymbol{y}_j - \boldsymbol{F}\boldsymbol{u}_j = \boldsymbol{g}(\tau_j) + \boldsymbol{\epsilon}_j. \tag{4.6}$$

The recursive GPR is applied separately to each single element $d_j^i = g^i(\tau_j) + \epsilon_j^i$ of the lifted vector $\boldsymbol{d}_j$, with $i = 1, 2, \ldots, N \cdot n_x$, where N is the number of time-steps in which the system dynamics is discretized and n_x is the number of state variables, i.e., each state variable at each time-step of the discretization of the trajectory is treated as an independent output and assumed uncorrelated to any other state variable, or the same one at any other time-step. Conversely, for each output, the cross-covariance between the basis vectors and the new observations is taken into account.

For the sake of clarity, the superindex i indicating the $i-$th element of vectors $\boldsymbol{d}_j$ and $\boldsymbol{y}_j$ will be omitted in the remainder of this section. The unknown function value, g^i, corresponding to d_j^i, will be denoted by g. Analogously, the standard deviation of ϵ_j^i will be denoted by σ. In this chapter, the recursive GPR approach is formulated to estimate a single element of $\boldsymbol{d}_j$, now denoted as d_j. The estimation of the remaining elements of $\boldsymbol{d}_j$ is computed analogously.

The approach presented in [45] is adapted here to the ILC algorithm and modified to incorporate the new observations to the basis vectors. Therefore, the basis vectors are fixed in number, but their location changes at each iteration, replacing the first one with the new data estimation. Since the function g depends solely on the time between the first and the current iterations, the basis vectors at the $j-$th iteration are the values of τ corresponding to the ℓ prior iterations, that is to say adding the subindex j to the notation in Section 4.2.3,

$\boldsymbol{v}_j = [\tau_{j-\ell}, \ldots, \tau_{j-1}]^T$. Similarly, $\boldsymbol{g}_j = [g(\tau_{j-\ell}), \ldots, g(\tau_{j-1})]^T$ is the vector of local estimates of the latent function at the ℓ basis vectors $\boldsymbol{v}_j$.

Although the observations are made on the output data, $\boldsymbol{y}_j$, the estimated and predicted values are the unmodeled dynamics and disturbances $\boldsymbol{d}_j$ affecting the system. The indirect observation of each element d_j becomes explicit in (4.6), as the corresponding element of $\boldsymbol{d}_j = \boldsymbol{y}_j - \boldsymbol{F}\boldsymbol{u}_j$.

Estimation

As explained in Section 4.2.3, the algorithm recursively estimates the values of the latent function and simultaneously learns the hyperparamenters. As such, no prior knowledge of the disturbances is needed. This is achieved by calculating a joint posterior Gaussian distribution $p(\boldsymbol{z}_j|\boldsymbol{y}_{1:j})$, where $\boldsymbol{z}_j^T \equiv [\boldsymbol{g}_j^T, \boldsymbol{\eta}_j^T]$ is the joint hidden state with mean and covariance

$$\boldsymbol{\mu}_j^z \equiv \begin{bmatrix} \boldsymbol{\mu}_j^g \\ \boldsymbol{\mu}_j^\eta \end{bmatrix}, \quad \boldsymbol{C}_j^z \equiv \begin{bmatrix} \boldsymbol{C}_j^g & \boldsymbol{C}_j^{g\eta} \\ \boldsymbol{C}_j^{\eta g} & \boldsymbol{C}_j^\eta \end{bmatrix}, \tag{4.7}$$

being $\boldsymbol{\mu}_j^g$ and $\boldsymbol{C}_j^g$ the mean and covariance of $\boldsymbol{g}_j$, $\boldsymbol{\mu}_j^\eta$ and $\boldsymbol{C}_j^\eta$ the mean and covariance of the hyperparameters, and $\boldsymbol{C}_j^{g\eta}$ and $\boldsymbol{C}_j^{\eta g}$ the cross-covariance matrices at iteration j.

The starting point for this calculation is a joint prior distribution $p(\boldsymbol{z}_{j-1}|\boldsymbol{d}_{1:j-1})$ at iteration $j-1$, which is updated with the new observation d_j. To incorporate the new input at iteration j, it is necessary to infer the latent function $\bar{g}_j = g(\tau_j)$. The state-space model incorporating the hyperparameters is given by

$$\begin{bmatrix} \boldsymbol{z}_{j-1} \\ \bar{g}_j \end{bmatrix} = \boldsymbol{A}_j(\boldsymbol{\eta}_j - 1)\boldsymbol{z}_{j-1} + \boldsymbol{w}_j, \tag{4.8}$$

where

$$\boldsymbol{A}_j(\boldsymbol{\eta}_{j-1}) = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{J}_j(\boldsymbol{\eta}_{j-1}) & \boldsymbol{0} \end{bmatrix},$$

and the noise $\boldsymbol{w}_j$ is Gaussian with mean and covariance

$$\boldsymbol{\mu}_j^w \equiv \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ b(\boldsymbol{\eta}_{j-1}) \end{bmatrix}, \quad \boldsymbol{C}_j^w \equiv \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & B(\boldsymbol{\eta}_{j-1}) \end{bmatrix}. \tag{4.9}$$

The kernel functions in $\boldsymbol{J}_j(\boldsymbol{\eta}_{j-1}) \equiv \boldsymbol{K}(\tau_j, \boldsymbol{v})\boldsymbol{K}(\boldsymbol{v}, \boldsymbol{v})^{-1}$, in $B(\boldsymbol{\eta}_{j-1}) \equiv \boldsymbol{K}(\tau_j, \tau_j) - \boldsymbol{J}_j(\boldsymbol{\eta}_{j-1})\boldsymbol{K}(\boldsymbol{v}, \tau_j)$, and in $b(\boldsymbol{\eta}_{j-1}) \equiv \mu(\tau_j) - \boldsymbol{J}_j(\boldsymbol{\eta}_{j-1})\mu(\boldsymbol{v})$ are calculated using the hyperparameters $\boldsymbol{\eta}_{j-1}$.

The model (4.8) is nonlinear with respect to the hyperparameters $\boldsymbol{\eta}_{j-1}$, but, for a given hyperparameter, the model is linear and the prediction inducing the desired joint distribution

$p(\boldsymbol{z}_{j-1}, \bar{g}_j|\boldsymbol{d}_{1:j-1})$ can be performed exactly using a Kalman predictor. Here, a collection of s sigma points $\hat{\boldsymbol{\eta}}_i$ is selected and given weights $\boldsymbol{\omega}_i$, $i = 1, \ldots, s$, using the unscented transform [46] constrained so that the hyperparameters are positive [47]. A Kalman predictor is applied to each sigma point obtaining the joint Gaussian distribution $p(\boldsymbol{z}_{j-1}, \bar{g}_j|\boldsymbol{d}_{1:j-1})$ with mean and covariance

$$\begin{aligned} \boldsymbol{\mu}_j^p &= \sum_{i=1}^{s} \boldsymbol{\omega}_i \boldsymbol{\mu}_i^p, \\ \boldsymbol{C}_j^p &= \sum_{i=1}^{s} ((\boldsymbol{\mu}_i^p - \boldsymbol{\mu}_j^p)(\boldsymbol{\mu}_i^p - \boldsymbol{\mu}_j^p)^T + \boldsymbol{C}_i^p), \end{aligned} \tag{4.10}$$

where

$$\begin{aligned} \boldsymbol{\mu}_i^p &\equiv \boldsymbol{A}_j(\hat{\boldsymbol{\eta}}_i) \begin{bmatrix} \boldsymbol{\mu}_{j-1}^g + \boldsymbol{C}_{j-1}^{g\eta}(\boldsymbol{C}_{j-1}^{\eta})^{-1}(\hat{\boldsymbol{\eta}}_i - \boldsymbol{\mu}_{j-1}^{\eta}) \\ \hat{\boldsymbol{\eta}}_i \end{bmatrix} + \boldsymbol{\mu}_j^w(\hat{\boldsymbol{\eta}}_i), \\ \boldsymbol{C}_i^p &\equiv \boldsymbol{A}_j(\hat{\eta}_i) \begin{bmatrix} \boldsymbol{C}_{j-1}^g - \boldsymbol{C}_{j-1}^{g\eta}(\boldsymbol{C}_{j-1}^{\eta})^{-1}\boldsymbol{C}_{j-1}^{\eta g} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \boldsymbol{A}_j(\hat{\boldsymbol{\eta}}_i)^T + \boldsymbol{C}_t^w(\hat{\boldsymbol{\eta}}_i). \end{aligned} \tag{4.11}$$

The incorporation of the new observation, d_j, is performed in two steps, in which the joint distribution is decomposed into an observed and an unobserved part, namely

$$p(\boldsymbol{z}_j|\boldsymbol{d}_{1:j}) = \int p(\boldsymbol{g}_j, \boldsymbol{\eta}_j^-|\sigma, \bar{g}_j) \cdot p(\sigma, \bar{g}_j|\boldsymbol{d}_{1:j}) d\bar{g}_j,$$

where $\boldsymbol{\eta}_j^-$ is the vector of all the hyperparameters except σ. As such, $\boldsymbol{\mu}_j^p$ and $\boldsymbol{C}_j^p$ can be expressed as

$$\boldsymbol{\mu}_j^p \equiv \begin{bmatrix} \boldsymbol{\mu}_{j-1}^u \\ \boldsymbol{\mu}_j^o \end{bmatrix}, \quad \boldsymbol{C}_j^z \equiv \begin{bmatrix} \boldsymbol{C}_{j-1}^u & \boldsymbol{C}_j^{uo} \\ \boldsymbol{C}_j^{ou} & \boldsymbol{C}_j^o \end{bmatrix}, \tag{4.12}$$

where $\boldsymbol{o}^T = [\sigma, \bar{g}_j^T]$ is the observable state with mean $\boldsymbol{\mu}_j^o$ and covariance $\boldsymbol{C}_j^o$, $\boldsymbol{u}_{j-1}^T = [\boldsymbol{g}_j^T, (\boldsymbol{\eta}_j^-)^T]$ is the unobservable state with mean $\boldsymbol{\mu}_{j-1}^u$ and covariance $\boldsymbol{C}_{j-1}^u$, and $\boldsymbol{C}_j^{uo}$ and $\boldsymbol{C}_j^{ou}$ are the cross-covariance matrices between the observable and unobservable states.

The observable state can be directly updated and, assuming that the observed state and the observations are jointly Gaussian, the conditional distribution $p(\sigma, \bar{g}_j|\boldsymbol{d}_{1:j})$ will have mean and covariance

$$\begin{aligned} \boldsymbol{\mu}_j^e &= \boldsymbol{\mu}_j^o + \boldsymbol{C}_j^{od}(C_j^d)^{-1}(d_j - \mu_j^d), \\ \boldsymbol{C}_j^e &= \boldsymbol{C}_j^o - \boldsymbol{C}_j^{od}\left[\boldsymbol{C}_j^{od}(C_j^d)^{-1}\right]^T, \end{aligned} \tag{4.13}$$

where $\mu_j^d = E[\bar{g}_j]$, $C_j^d = \mathrm{COV}[\bar{g}_j] + \mathrm{VAR}[\sigma] + E[\sigma]^2$, and $\boldsymbol{C}_j^{od} = \mathrm{COV}[\boldsymbol{o}_j, \bar{g}_j]$ are elements of $\boldsymbol{\mu}_j^o$ and $\boldsymbol{C}_j^o$. The unobservable part is then updated as a Gaussian distribution $p(\boldsymbol{g}_j, \boldsymbol{\eta}_j^-|\sigma, \bar{g}_j)$ with mean and covariance

$$\begin{aligned} \boldsymbol{\mu}_j^u &= \boldsymbol{\mu}_{j-1}^u + \boldsymbol{C}_j^{uo}(\boldsymbol{C}_j^o)^{-1}(\boldsymbol{\mu}_j^e - \boldsymbol{\mu}_j^o), \\ \boldsymbol{C}_j^u &= \boldsymbol{C}_{j-1}^u + \boldsymbol{C}_j^{uo}(\boldsymbol{C}_j^o)^{-1}(\boldsymbol{C}_j^e - \boldsymbol{C}_j^o)\left[\boldsymbol{C}_j^{uo}(\boldsymbol{C}_j^o)^{-1}\right]^T. \end{aligned} \tag{4.14}$$

Finally, the mean and covariance of the the joint posterior distribution $p(\boldsymbol{z}_j|\boldsymbol{d}_{1:j})$ with

updated basis vectors and hyperparameters is obtained combining and rearranging (4.13) and (4.14) as $\boldsymbol{z}_j^T = [(\boldsymbol{g}_j^-)^T, \bar{g}_j^T, \boldsymbol{\eta}_j^T]$, where $\boldsymbol{g}_j^-$ indicates the latent function at the basis vectors except the basis vector corresponding to the iteration $j - \ell$. This is done to incorporate the latent function estimation of the new observations, $\bar{g}_j$, into the basis vectors while keeping a fixed number of them, since the most recent data will become more significant than the oldest in the following iterations. The updated basis vectors will be then $\boldsymbol{v}_{j+1} = [\tau_{j-\ell+1}, \ldots, \tau_j]$.

Prediction

To predict the latent function at iteration $j+1$, the process is repeated to calculate $\boldsymbol{\mu}_{j+1}^p$ as in (4.10), using the joint posterior distribution, $p(\boldsymbol{z}_j|\boldsymbol{d}_{1:j})$, obtained in the previous steps. The predicted disturbance will be then

$$d_{j+1}^p = E[\bar{g}_{j+1}], \tag{4.15}$$

where $E[\bar{g}_{j+1}]$ can be extracted from $\boldsymbol{\mu}_{j+1}^p$.

Input update

Repeating the previous process for each element of $\boldsymbol{d}_j$, the vector $\boldsymbol{d}_{j+1}^p$ is built, which contains the predicted values of the disturbances affecting all the state variables at every time-step in which the trajectory has been discretized. Following [24], a new control input, $\boldsymbol{u}_{j+1}$, is calculated optimally compensating for the predicted disturbance by solving the constrained optimization problem

$$\begin{aligned} \min_{\boldsymbol{u}_{j+1}} \quad & \|\boldsymbol{F}\boldsymbol{u}_{j+1} + \boldsymbol{d}_{j+1}^p\| + \alpha\|\boldsymbol{D}\boldsymbol{u}_{j+1}\|, \\ \text{subject to:} \quad & \boldsymbol{L}\boldsymbol{u}_{j+1} \leq \boldsymbol{q}_{\max}, \end{aligned} \tag{4.16}$$

where the system's constraints (3.2) are explicitly taken into account. The additional term $\alpha \geq 0$ and the matrix $\boldsymbol{D}$ are introduced to penalize the input or approximations of its derivatives pursuing smoothness of the optimal solution. The update law in (4.16) can be expressed as a standard convex optimization problem, as explained in Section 3.2.3.

4.4 Experimental results

To show the effectiveness of the GPR for estimation in ILC, it has been tested in a simulated experiment consisting of the precise trajectory tracking of an Airbus A320-231 aircraft in a CCO, generated as explained in Section 2.2. The aircraft model and flight envelope employed to generate the trajectories are those described in Section 2.1.1, assuming zero wind. The simulated environment in which the experiments are carried out includes a realistic wind model, in which wind turbulence has been introduced, as well as model and measurement errors.

The experiments are conducted using the GPR and the Kalman filter methods for the estimation and prediction steps of the ILC in similar conditions and with different parameters to compare their performance.

Ten experiments are performed consisting of 30 iterations each. At each iteration, the same aircraft intends to fly the CCO trajectory. The time between iterations is one hour. The figures show the median of the ten experiments of:

- the Root Mean Square Error (RMSE) of the scaled disturbance estimation at each iteration,
- the RMSE of the scaled disturbance prediction at each iteration for the following one, and
- the weighted state error, which is calculated as in (3.21), that is, $e_{ws,j} = ||\boldsymbol{S}\boldsymbol{y}_j||_2$.

The black dashed horizontal line in figures 4.1 to 4.6 is the output error standard deviation, which characterizes the system noise level. It is obtained from the variations in the trajectory when applying the same input to the aircraft several times.

Experiment 1: Comparison between different GPR approaches

A comparison between the GPR method using all the available data, the GPR using only data from last 10 iterations, and the recursive GPR using 10 basis vectors is shown in figures 4.1, 4.2, and 4.3, respectively. The training data used to define the basis vectors is obtained from the first 10 iterations as well as the initial guess of the hyperparameters. The non-recursive GPR methods are implemented using MATLAB's embedded function for Gaussian process regression. As shown in the figures, after some iterations, the recursive GPR achieves, in both estimation and prediction, an RMSE similar to, and in some iterations even lower than the non-recursive GPR approach. All three methods exhibit a similar behavior when implemented into the ILC algorithm in terms of the weighted state error. The peaks observed at iterations 12 and 15 are caused by a change in the trend of the mean horizontal wind speed. Moreover, the difference between the wind speed values at these iterations and the previous ones is greater than in any other pair of consecutive iterations.

Table 4.1 shows the mean execution times of the Kalman filter and the non-recursive and recursive GPR approaches, observed on a standard laptop computer with a 1.6 GHz Intel Core i5 processor and 16 GB RAM. In the same table, the corresponding mean weighted state error after three iterations is also reported, showing that the non-recursive GPR approach is faster than the recursive GPR in reducing the tracking error when the training data are obtained from the previous 5 or 10 iterations. In both approaches, the weighted state error converges to its minimum value when using training data from more than 20 past iterations. The execution time of the recursive GPR, although higher than that of the Kalman filter, is significantly lower than the execution time of the non-recursive GPR method. Note that the computational effort increases as the iterations take place and more data are available.

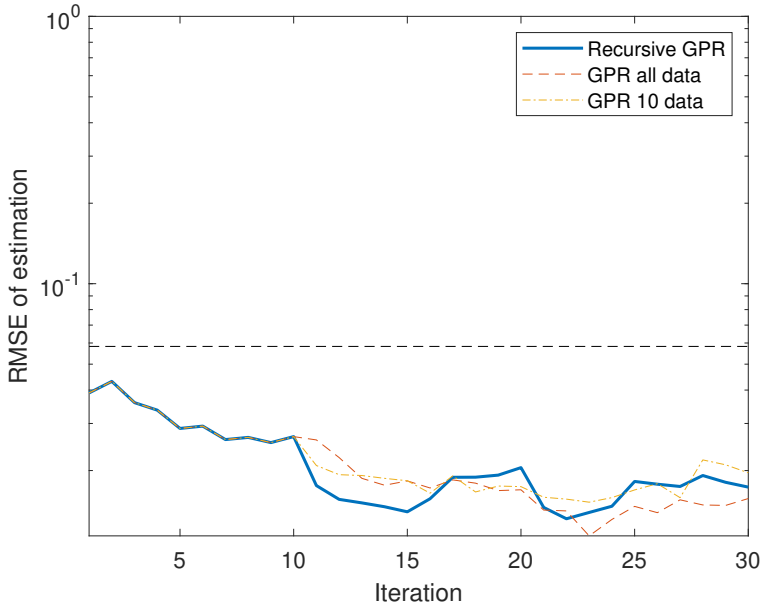


Figure 4.1: Experiment 1: Comparison of the RMSE of the estimation obtained with different GPR approaches.

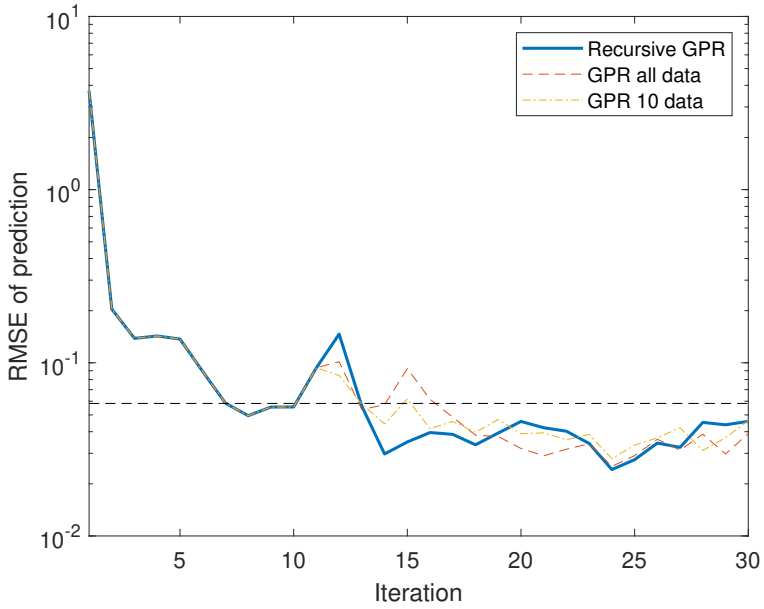


Figure 4.2: Experiment 1: Comparison of the RMSE of the prediction obtained with different GPR approaches.

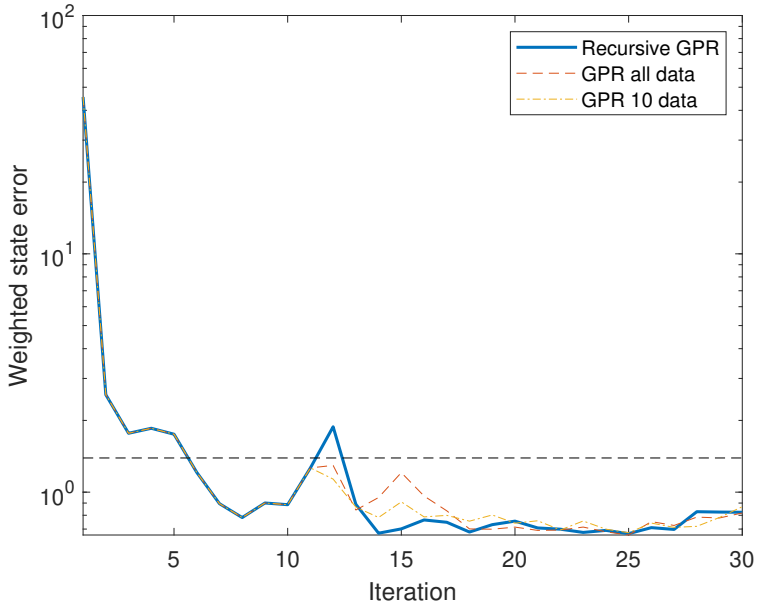


Figure 4.3: Experiment 1: Comparison of the weighted state error obtained with different GPR approaches.

	Training data	Mean computation time (s)	Mean e_w after 3 iterations
Non-recursive GPR	5 iterations	1.5316	0.8840
	10 iterations	2.2788	0.9615
	20 iterations	3.0928	0.7063
	50 iterations	3.3923	0.7005
Recursive GPR	5 iterations	0.1186	1.0891
	10 iterations	0.1284	1.0162
	20 iterations	0.1633	0.7551
	50 iterations	0.7150	0.7697
Kalman filter		0.0003	1.3606

Table 4.1: Mean computation times per iteration and weighted state error observed using a standard laptop computer.

Experiment 2: Comparison between the GPR and the Kalman filter

As explained above, one of the advantages of using the GPR is that no previous knowledge about the system dynamics is needed, in contrast to the Kalman filter, which requires the noise and measurement covariance matrices to be set. The recursive GPR for estimation and prediction is compared here to the Kalman filter's performance in the ILC method in three different cases: with $\boldsymbol{\Omega} = 0.1\boldsymbol{M}$, with $\boldsymbol{\Omega} = \boldsymbol{M}$, and with $\boldsymbol{\Omega} = 10\boldsymbol{M}$, where $\boldsymbol{\Omega}$ is the covariance matrix of the process noise and $\boldsymbol{M}$ is the covariance matrix of the measurement noise.

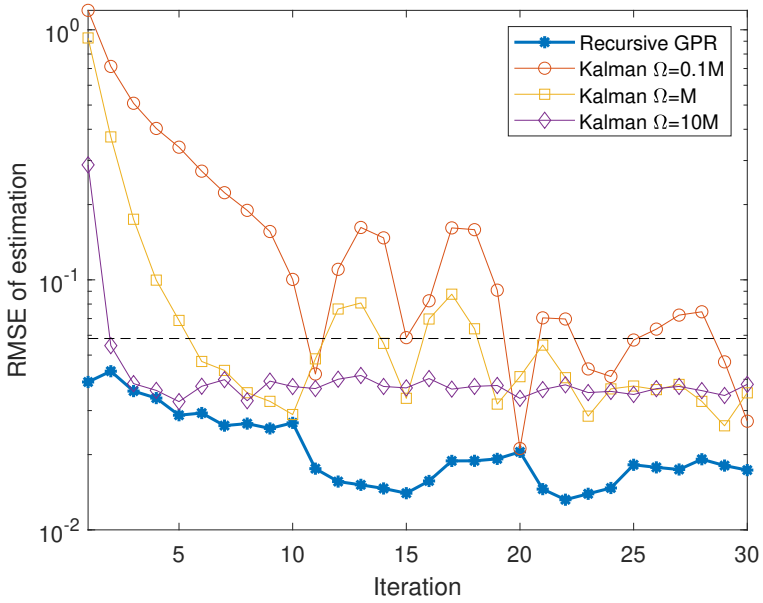


Figure 4.4: Experiment 2: RMSE of the estimation obtained with the recursive GPR and the Kalman filter.

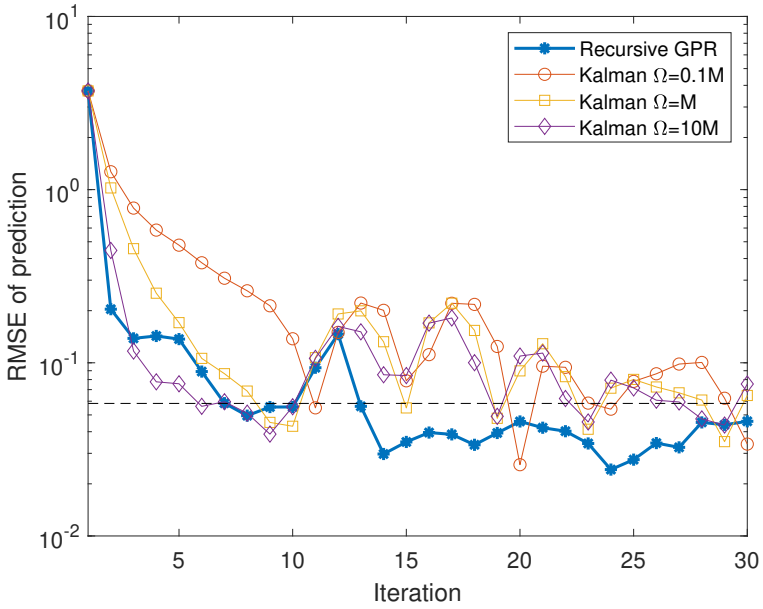


Figure 4.5: Experiment 2: RMSE of the prediction obtained with the recursive GPR and the Kalman filter.

Figures 4.4 and 4.5 show that the recursive GPR approach achieves better estimations and predictions of the disturbances than any of the Kalman filters. In the first iterations, the recursive GPR and the Kalman filter with $\boldsymbol{\Omega} = 10\boldsymbol{M}$ are comparable and much faster than the Kalman filter with $\boldsymbol{\Omega} = \boldsymbol{M}$ and $\boldsymbol{\Omega} = 0.1\boldsymbol{M}$. In terms of the weighted state error, Figure 4.6 shows that the recursive GPR and the Kalman filter with $\boldsymbol{\Omega} = 10\boldsymbol{M}$ are comparable in the first iterations and that only in the last few iterations all three methods return similar results.

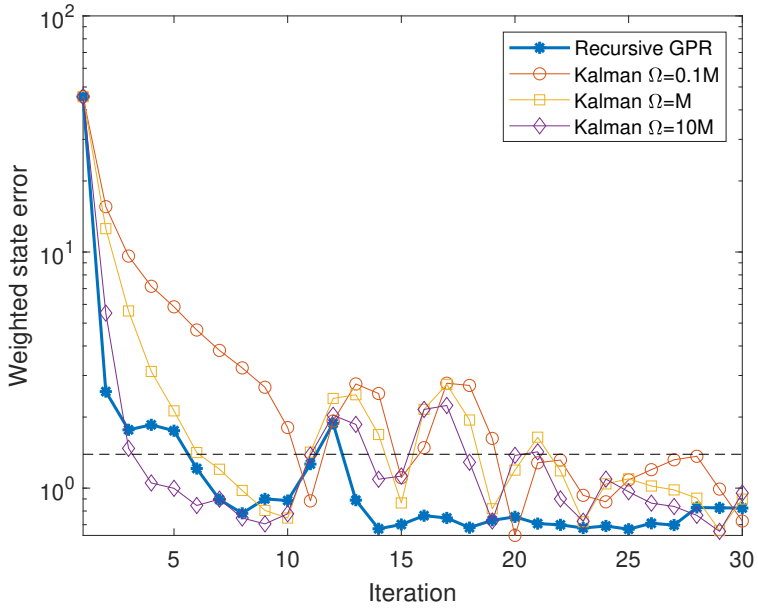


Figure 4.6: Experiment 2: Weighted state error obtained with the recursive GPR and the Kalman filter.

Figure 4.7 shows an enlarged view of the last part of the path described by the aircraft in the first iterations using the GPR and the Kalman filter with $\boldsymbol{\Omega} = \boldsymbol{M}$. In the first iteration, the inputs fed to the aircraft are the nominal ones obtained in the trajectory generation where disturbances are not taken into account, and the resulting path is clearly deviated from the desired one. In the following iterations, the ILC method amends these deviations. Figure 4.7 shows that with the GPR estimation and prediction, the convergence to the desired path is faster than with the Kalman estimator until reaching the best achievable accuracy, which is represented by the path that corresponds to iteration 25. It can be seen that the error between the actual and the desired paths is significantly reduced, but is not zero due to the aircraft's operational limits and the system's noise, which cannot be accurately predicted.

Finally, Figure 4.8 shows the control inputs generated by the ILC method when the estimation and prediction steps are performed using the GPR and the Kalman filter. It can be observed that, as in Figure 4.7, with the GPR, the control inputs converge faster to the ones that drive the aircraft to the most precise achievable tracking of the trajectory (iteration 25) than those generated when using the Kalman filter.

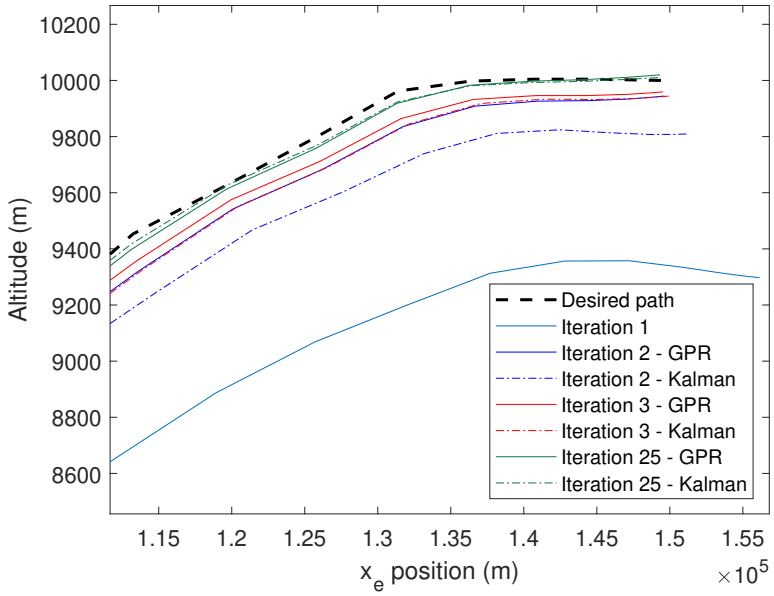


Figure 4.7: Experiment 2: Enlarged view of the path $x_e - h$ described by the aircraft using the GPR and the Kalman predictions of the disturbances.

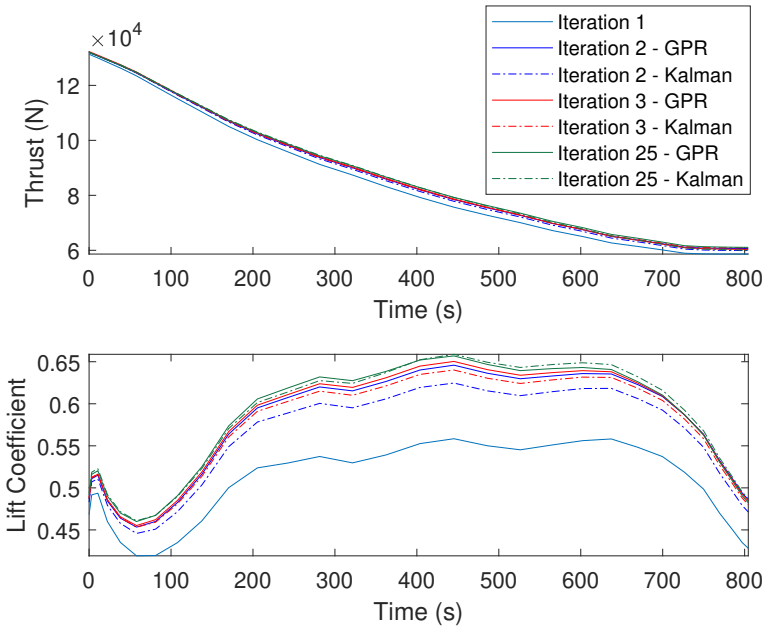


Figure 4.8: Experiment 2: Evolution of the thrust and lift coefficient over iterations using the GPR and the Kalman predictions of the disturbances.

4.5 Conclusion

Estimation and prediction are key for precision and fast convergence of ILC, since the update of the inputs given to the aircraft is based on the predicted perturbations and the estimated model errors that cause deviations from the planned trajectory. In this chapter, GPR has been implemented as the estimation and prediction step of an optimization-based ILC method to compensate for the disturbances affecting a sequence of identical aircraft performing the same trajectory. In order to reduce the high computational effort required by GPR, a recursive approach has been used, which performs the regression on a set of points that are updated at each iteration to dismiss old data in favor of the most recent measurements, simultaneously learning the hyperparameters.

With no prior knowledge of the behavior of the disturbances, the experiments show that recursive GPR provides much better estimations and predictions in the first iterations compared to the Kalman filter with $\boldsymbol{\Omega} = 0.1\boldsymbol{M}$ and $\boldsymbol{\Omega} = \boldsymbol{M}$, reaching a mean improvement of 80% and 67%, respectively, in the weighted state error along the first five iterations, therefore achieving faster and more precise tracking of the aircraft trajectory. The mean improvement across all 30 iterations is 65% and 48%, respectively. The results obtained by both methods are comparable only when the Kalman filter is better tuned, with $\boldsymbol{\Omega} = 10\boldsymbol{M}$, with a mean improvement of 22%, which means collecting some prior knowledge about the dynamics of the disturbances. Additionally, recursive GPR allows for estimation and prediction of nonlinear dynamics.

Part II

Multi-Aircraft Transfer Learning

5

MRAC Adaptive State-Feedbak Control

In the previous chapters, the possibility of applying the ILC paradigm to improve precision in aircraft trajectory tracking has been investigated. However, in this context, additional difficulties arise, since consecutive flights are carried out by different aircraft, so different dynamical systems perform each iteration, whereas the basic ILC scheme requires the system dynamics to be repetition-invariant. This drawback is overcome by using an underlying adaptive controller that forces the aircraft to behave close to a specified reference model. In particular, an MRAC is combined with the aircraft's preexisting trajectory tracking controller, in this case an LQR PI controller, enhancing its performance through direct adaptation.

5.1 Introduction

An adaptive control system can be described as a control system that has the ability to adjust control design parameters online based on inputs received by the plant in order to accommodate system uncertainty [48]. Research in adaptive control was initiated in the early 1950s, motivated by the interest in designing new advanced autopilots for high-performance aircraft that operated in a wide range of speeds and altitudes, experiencing large parametric variations [49]. Up to the present time, research in adaptive control remains highly active. Applications of adaptive control include aerospace engineering [50], [51], [52], [53], [54] and other fields such as automotive engineering, robotics, and mechatronics [55], [56], [57], [58]. More recent robust adaptive control methods include the adaptive loop recovery [59] and $\mathcal{L}_1$ adaptive control [60].

MRAC is one of the most popular approaches to adaptive control. It was initially proposed in [61], and the Lyapunov stability theory led to the development of improved MRAC schemes [62]. An MRAC system consists of a controller that, using an adaptive law, estimates its parameters online, based on the system output and on a reference model that determines

the trajectory to follow according to a commanded input. The adaptive controller uses the estimated parameters to compute a control input that forces the system output to follow the desired external commands, even when it operates in the presence of uncertainties and model errors.

Several textbooks on MRAC are available, such as [48], which provides a foundation in the MRAC control with basic well-accepted adaptive control techniques, as well as more recent robust adaptive control methods, and [63], which deals with the fundamental topics of robust and adaptive control with special focus on the aerospace applications field. Some recent publications on novel MRAC-based techniques are [64], which develops a derivative-free MRAC law, and [65], which presents an MRAC approach for systems with unstructured uncertainty based on the gradient-based and the recursive least-squares methods. Recent aerospace applications of MRAC are [66], [67], and [68].

This chapter presents the fundamental theory of direct MRAC, which estimates the control gains using available measurements, as opposed to the indirect approach, in which the gains are approximated by solving system design equations that relate the plant uncertainties to the known signals in the system. In this thesis, parametric matched uncertainty and unknown bounded disturbance are assumed. Parametric matched uncertainties are those in which the uncertainty enters the system through the same channel as the control, so the control does not have to overcome integrator transients in order to counteract its effects. In other words, there exists a control input capable of completely cancelling the uncertainty out if the adaptation is perfect [69]. A generic block diagram of an MRAC controller is shown in Figure 5.1. The main components of the MRAC architecture are:

- The plant, which can be a linear or nonlinear system affected by uncertainty or unmodelled dynamics.
- The reference model, which determines a desired response of an adaptive control system to a command input.
- The controller, designed to provide overall system performance and stability for a nominal plant without uncertainty. In this thesis, an LQR PI controller augmented with an adaptive controller is used for two reasons: to enable adaptive control without replacing the aircraft's existing controller, and because the adaptive augmentation control design is generally more robust than a fully adaptive control design. The LQR PI acts as a baseline controller which simulates the aircraft's underlying trajectory tracking controller, often designed with proportional and integral feedback connections. The MRAC augments the nominal control action provided by the baseline controller by adding an adaptive control action in the presence of system uncertainties and changing dynamics.
- The adaptive law, which updates the adaptive parameters so that the tracking error converges to an as small as possible value. To ensure system stability, the adaptive laws are designed using Lyapunov stability theory.

The problem of tracking a command requires the control input to be able to force the output tracking error, namely the difference between the output and the external command, to asymptotically become sufficiently small as time increases, keeping all the signals in the closed-loop control system uniformly bounded. If asymptotic tracking is not feasible, the aim of the controller is to achieve uniform ultimate boundedness of the tracking error. As shown in Figure 5.1, the MRAC is designed to enable the output to track an external command using adaptive gains. The adaptive law updates the adaptive gains based on the tracking error between the model output and the reference model output.

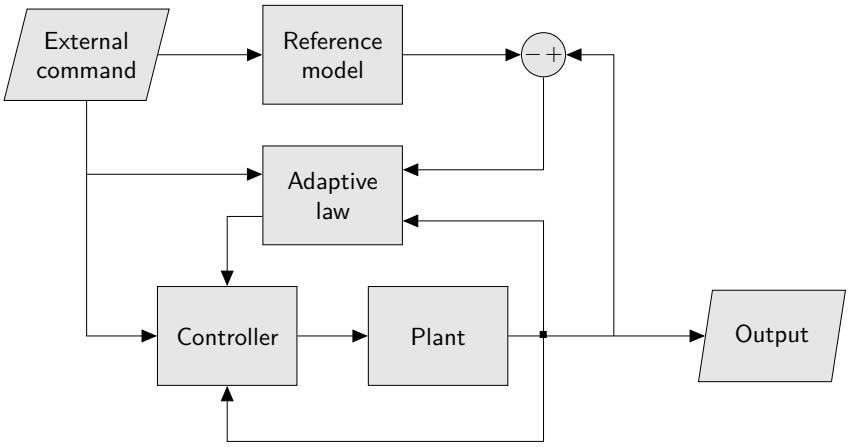


Figure 5.1: MRAC block diagram.

5.2 MRAC problem formulation

In the following sections, the direct MRAC design proposed in [63, Chap. 9] is introduced.

The uncertain and changing system dynamics are assumed to be a Multi-Input Multi-Output (MIMO) nonlinear system in the form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{\Lambda}(\boldsymbol{u} + \boldsymbol{f}(\boldsymbol{x})), \tag{5.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the system state, $\boldsymbol{u} \in \mathbb{R}^m$ is the control input, and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ is the known control matrix. $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{\Lambda} \in \mathbb{R}^{m\times m}$ are unknown constant matrices, being $\boldsymbol{\Lambda}$ a control input uncertainty, which is a diagonal matrix, the elements of which are strictly positive. It is assumed that the pair $(\boldsymbol{A}, \boldsymbol{B}\boldsymbol{\Lambda})$ is controllable. The nonlinear vector function $\boldsymbol{f}(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^m$ represents the matched uncertainty that can be parameterized as

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\Theta}^T \boldsymbol{\Phi}(\boldsymbol{x}), \tag{5.2}$$

where $\boldsymbol{\Theta} \in \mathbb{R}^{N\times m}$ is a constant, unknown matrix, and $\boldsymbol{\Phi}(\boldsymbol{x}) = (\varphi_1(\boldsymbol{x}), \ldots, \varphi_N(\boldsymbol{x}))^T \in \mathbb{R}^N$ is a known regressor vector, which is a basis vector whose components are assumed to be Lipschitz-continuous in $\boldsymbol{x}$.

The aim is to design a MIMO state feedback adaptive control law such that the system state $\boldsymbol{x}$ follows a reference model specified by

$$\dot{\boldsymbol{x}}_{\boldsymbol{m}} = \boldsymbol{A_m x_m} + \boldsymbol{B_m r}, \tag{5.3}$$

where $\boldsymbol{x_m}(t) \in \mathbb{R}^n$ is the state vector of the reference model, $\boldsymbol{A_m} \in \mathbb{R}^{n \times n}$ is a known Hurwitz matrix, $\boldsymbol{B_m} \in \mathbb{R}^{n \times m}$ is known, and $\boldsymbol{r}(t) \in \mathbb{R}^m$ is the external bounded reference input.

5.3 MRAC architecture

If matrices $\boldsymbol{A}$ and $\boldsymbol{\Lambda}$ where known, an ideal fixed-gain control law could be defined that perfectly canceled out the uncertainty and enabled $\boldsymbol{x}(t)$ to follow the state reference $\boldsymbol{x_m}(t)$, as

$$\boldsymbol{u} = \boldsymbol{K_x^T x} + \boldsymbol{K_r^T r} - \boldsymbol{\Theta}^T \boldsymbol{\Phi}(\boldsymbol{x}), \tag{5.4}$$

getting the ideal closed-loop system

$$\dot{\boldsymbol{x}} = (\boldsymbol{A} + \boldsymbol{B\Lambda K_x^T})\boldsymbol{x} + \boldsymbol{B\Lambda K_r^T r}. \tag{5.5}$$

Comparing the ideal closed-loop system to the reference model, the ideal control gains $\boldsymbol{K_x}$ and $\boldsymbol{K_r}$ must satisfy the model matching conditions

$$\begin{aligned} \boldsymbol{A} + \boldsymbol{B\Lambda K_x^T} &= \boldsymbol{A_m}, \\ \boldsymbol{B\Lambda K_r^T} &= \boldsymbol{B_m}. \end{aligned} \tag{5.6}$$

If $\boldsymbol{A_m}$ and $\boldsymbol{B_m}$ have the same structures as those of $\boldsymbol{A}$ and $\boldsymbol{B\Lambda}$, respectively, or if $\boldsymbol{B\Lambda}$ is a square and invertible matrix, then there exist $\boldsymbol{K_x}$ and $\boldsymbol{K_r}$ that satisfy the model matching conditions. Assuming they exist, the following control law is defined

$$\boldsymbol{u} = \hat{\boldsymbol{K}}_{\boldsymbol{x}}^T \boldsymbol{x} + \hat{\boldsymbol{K}}_{\boldsymbol{r}}^T \boldsymbol{r} - \hat{\boldsymbol{\Theta}}^T \boldsymbol{\Phi}(\boldsymbol{x}), \tag{5.7}$$

where $\hat{\boldsymbol{K}}_{\boldsymbol{x}} \in \mathbb{R}^{n \times m}$, $\hat{\boldsymbol{K}}_{\boldsymbol{r}} \in \mathbb{R}^{m \times m}$, and $\hat{\boldsymbol{\Theta}} \in \mathbb{R}^{N \times m}$ are the estimates of the ideal unknown matrices $\boldsymbol{K_x}$, $\boldsymbol{K_r}$, and $\boldsymbol{\Theta}$, respectively.

Substituting (5.7) into (5.1), the closed-loop system model can be expressed as

$$\dot{\boldsymbol{x}} = (\boldsymbol{A} + \boldsymbol{B\Lambda}\hat{\boldsymbol{K}}_{\boldsymbol{x}}^T)\boldsymbol{x} + \boldsymbol{B\Lambda}\left(\hat{\boldsymbol{K}}_{\boldsymbol{r}}^T \boldsymbol{r} - (\hat{\boldsymbol{\Theta}} - \boldsymbol{\Theta})^T \boldsymbol{\Phi}(\boldsymbol{x})\right). \tag{5.8}$$

Defining the state tracking error as $\boldsymbol{e}(t) = \boldsymbol{x}(t) - \boldsymbol{x_m}(t)$, and taking into account the matching conditions (5.6), the closed-loop tracking error dynamics can be formulated as

$$\begin{aligned}\dot{\boldsymbol{e}} &= \left(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Lambda}\hat{\boldsymbol{K}}_{\boldsymbol{x}}^T\right)\boldsymbol{x} + \boldsymbol{B}\boldsymbol{\Lambda}\left(\hat{\boldsymbol{K}}_{\boldsymbol{r}}^T\boldsymbol{r} - (\hat{\boldsymbol{\Theta}} - \boldsymbol{\Theta})^T\boldsymbol{\Phi}(\boldsymbol{x})\right) - \boldsymbol{A_m}\boldsymbol{x_m} - \boldsymbol{B_m}\boldsymbol{r} \\ &= \left(\boldsymbol{A_m} + \boldsymbol{B}\boldsymbol{\Lambda}(\hat{\boldsymbol{K}}_{\boldsymbol{x}} - \boldsymbol{K}_{\boldsymbol{x}})^T\right)\boldsymbol{x} - \boldsymbol{A_m}\boldsymbol{x_m} + \boldsymbol{B}\boldsymbol{\Lambda}(\hat{\boldsymbol{K}}_{\boldsymbol{r}} - \boldsymbol{K}_{\boldsymbol{r}})^T\boldsymbol{r} - \boldsymbol{B}\boldsymbol{\Lambda}(\hat{\boldsymbol{\Theta}} - \boldsymbol{\Theta})^T\boldsymbol{\Phi}(\boldsymbol{x}) \\ &= \boldsymbol{A_m}\boldsymbol{e} + \boldsymbol{B}\boldsymbol{\Lambda}\left[(\hat{\boldsymbol{K}}_{\boldsymbol{x}} - \boldsymbol{K}_{\boldsymbol{x}})^T\boldsymbol{x} + (\hat{\boldsymbol{K}}_{\boldsymbol{r}} - \boldsymbol{K}_{\boldsymbol{r}})^T\boldsymbol{r} - (\hat{\boldsymbol{\Theta}} - \boldsymbol{\Theta})^T\boldsymbol{\Phi}(\boldsymbol{x})\right].\end{aligned} \tag{5.9}$$

Let $\widetilde{\boldsymbol{K}}_{\boldsymbol{x}} = \hat{\boldsymbol{K}}_{\boldsymbol{x}} - \boldsymbol{K}_{\boldsymbol{x}}$, $\widetilde{\boldsymbol{K}}_{\boldsymbol{r}} = \hat{\boldsymbol{K}}_{\boldsymbol{r}} - \boldsymbol{K}_{\boldsymbol{r}}$, and $\widetilde{\boldsymbol{\Theta}} = \hat{\boldsymbol{\Theta}} - \boldsymbol{\Theta}$ be the estimation errors of the adaptive gains. Then, the tracking error dynamics is expressed as

$$\dot{\boldsymbol{e}} = \boldsymbol{A_m}\boldsymbol{e} + \boldsymbol{B}\boldsymbol{\Lambda}\left[\widetilde{\boldsymbol{K}}_{\boldsymbol{x}}^T\boldsymbol{x} + \widetilde{\boldsymbol{K}}_{\boldsymbol{r}}^T\boldsymbol{r} - \widetilde{\boldsymbol{\Theta}}^T\boldsymbol{\Phi}(\boldsymbol{x})\right]. \tag{5.10}$$

Lyapunov's methods provide sufficient conditions for stability of a nominal trajectory without requiring explicit knowledge of the system solutions. In particular, the direct method, also called Lyapunov theorem on local stability, presents sufficient conditions for appraising uniform stability, which are expressed in terms of a locally positive-definite function $V(\boldsymbol{x}) > 0$, often referred to as a Lyapunov function candidate. Lyapunov's direct method states that the origin is uniformly stable if the time derivative along the system trajectories is locally negative-semidefinite $\left(\dot{V}(\boldsymbol{x}) \leq 0\right)$. If V is negative-definite $\left(\dot{V}(\boldsymbol{x}) < 0\right)$, then in addition to being stable, the system trajectories will approach the origin asymptotically. Additionally, according to Barbashin–Krasovskii–LaSalle theorem, if a radially unbounded Lyapunov function can be found, then the local uniform (asymptotic) stability properties from Lyapunov's direct method become global. An overview of Lyapunov stability theory is given in [63, Chap. 8], which includes the proofs of both theorems.

To derive the adaptive laws, the following globally radially unbounded quadratic Lyapunov candidate function is chosen, in which the positive-definite adaptation rate matrices $\boldsymbol{\Gamma_x} = \boldsymbol{\Gamma_x}^T > \boldsymbol{0}$, $\boldsymbol{\Gamma_r} = \boldsymbol{\Gamma_r}^T > \boldsymbol{0}$, and $\boldsymbol{\Gamma_\Theta} = \boldsymbol{\Gamma_\Theta}^T > \boldsymbol{0}$ are introduced:

$$V(\boldsymbol{e}, \widetilde{\boldsymbol{K}}_{\boldsymbol{x}}, \widetilde{\boldsymbol{K}}_{\boldsymbol{r}}, \widetilde{\boldsymbol{\Theta}}) = \boldsymbol{e}^T\boldsymbol{P}\boldsymbol{e} + \mathrm{Tr}\left(\left[\widetilde{\boldsymbol{K}}_{\boldsymbol{x}}^T\boldsymbol{\Gamma_x}^{-1}\widetilde{\boldsymbol{K}}_{\boldsymbol{x}} + \widetilde{\boldsymbol{K}}_{\boldsymbol{r}}^T\boldsymbol{\Gamma_r}^{-1}\widetilde{\boldsymbol{K}}_{\boldsymbol{r}} + \widetilde{\boldsymbol{\Theta}}^T\boldsymbol{\Gamma_\Theta}^{-1}\widetilde{\boldsymbol{\Theta}}\right]\boldsymbol{\Lambda}\right), \tag{5.11}$$

where the symbol Tr denotes the trace of a square matrix and $\boldsymbol{P} = \boldsymbol{P}^T > \boldsymbol{0}$ solves the algebraic Lyapunov equation

$$\boldsymbol{P}\boldsymbol{A_m} + \boldsymbol{A_m}^T\boldsymbol{P} = -\boldsymbol{Q}, \tag{5.12}$$

for some $\boldsymbol{Q} = \boldsymbol{Q}^T > \boldsymbol{0}$. Then, the time derivative of V is evaluated along the trajectories of (5.10) obtaining

$$\begin{aligned}\dot{V}(\boldsymbol{e}, \widetilde{\boldsymbol{K}}_{\boldsymbol{x}}, \widetilde{\boldsymbol{K}}_{\boldsymbol{r}}, \widetilde{\boldsymbol{\Theta}}) = \dot{\boldsymbol{e}}^T\boldsymbol{P}\boldsymbol{e} + \boldsymbol{e}^T\boldsymbol{P}\dot{\boldsymbol{e}} \quad &+ 2\,\mathrm{Tr}\left(\left[\widetilde{\boldsymbol{K}}_{\boldsymbol{x}}^T\boldsymbol{\Gamma}_{\boldsymbol{x}}^{-1}\dot{\hat{\boldsymbol{K}}}_{\boldsymbol{x}} + \widetilde{\boldsymbol{K}}_{\boldsymbol{r}}^T\boldsymbol{\Gamma}_{\boldsymbol{r}}^{-1}\dot{\hat{\boldsymbol{K}}}_{\boldsymbol{r}} + \widetilde{\boldsymbol{\Theta}}^T\boldsymbol{\Gamma}_{\boldsymbol{\Theta}}^{-1}\dot{\hat{\boldsymbol{\Theta}}}\right]\boldsymbol{\Lambda}\right)\\ = \quad -\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} \quad &+ \left[2\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}\boldsymbol{\Lambda}\widetilde{\boldsymbol{K}}_{\boldsymbol{x}}^T\boldsymbol{x} + 2\,\mathrm{Tr}\left(\widetilde{\boldsymbol{K}}_{\boldsymbol{x}}^T\boldsymbol{\Gamma}_{\boldsymbol{x}}^{-1}\dot{\hat{\boldsymbol{K}}}_{\boldsymbol{x}}\boldsymbol{\Lambda}\right)\right]\\ &+ \left[2\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}\boldsymbol{\Lambda}\widetilde{\boldsymbol{K}}_{\boldsymbol{r}}^T\boldsymbol{r} + 2\,\mathrm{Tr}\left(\widetilde{\boldsymbol{K}}_{\boldsymbol{r}}^T\boldsymbol{\Gamma}_{\boldsymbol{r}}^{-1}\dot{\hat{\boldsymbol{K}}}_{\boldsymbol{r}}\boldsymbol{\Lambda}\right)\right]\\ &+ \left[-2\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}\boldsymbol{\Lambda}\widetilde{\boldsymbol{\Theta}}^T\boldsymbol{\Phi}(\boldsymbol{x}) + 2\,\mathrm{Tr}\left(\widetilde{\boldsymbol{\Theta}}^T\boldsymbol{\Gamma}_{\boldsymbol{\Theta}}^{-1}\dot{\hat{\boldsymbol{\Theta}}}\boldsymbol{\Lambda}\right)\right],\end{aligned} \tag{5.13}$$

where (5.12) is used. Utilizing the trace property $\mathrm{Tr}(\boldsymbol{C}\boldsymbol{D}^T) = \boldsymbol{D}^T\boldsymbol{C}$ for two generic matrices $\boldsymbol{C}$ and $\boldsymbol{D}$, then

$$\begin{aligned}\dot{V}(\boldsymbol{e}, \widetilde{\boldsymbol{K}}_{\boldsymbol{x}}, \widetilde{\boldsymbol{K}}_{\boldsymbol{r}}, \widetilde{\boldsymbol{\Theta}}) = &-\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} + 2\,\mathrm{Tr}\left(\widetilde{\boldsymbol{K}}_{\boldsymbol{x}}^T\left[\boldsymbol{\Gamma}_{\boldsymbol{x}}^{-1}\dot{\hat{\boldsymbol{K}}}_{\boldsymbol{x}} + \boldsymbol{x}\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}\right]\boldsymbol{\Lambda}\right)\\ &+ 2\,\mathrm{Tr}\left(\widetilde{\boldsymbol{K}}_{\boldsymbol{r}}^T\left[\boldsymbol{\Gamma}_{\boldsymbol{r}}^{-1}\dot{\hat{\boldsymbol{K}}}_{\boldsymbol{r}} + \boldsymbol{r}\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}\right]\boldsymbol{\Lambda}\right)\\ &+ 2\,\mathrm{Tr}\left(\widetilde{\boldsymbol{\Theta}}^T\left[\boldsymbol{\Gamma}_{\boldsymbol{\Theta}}^{-1}\dot{\hat{\boldsymbol{\Theta}}} - \boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}\right]\boldsymbol{\Lambda}\right).\end{aligned} \tag{5.14}$$

Therefore, selecting the adaptive laws as

$$\begin{aligned}\dot{\hat{\boldsymbol{K}}}_{\boldsymbol{x}} &= -\boldsymbol{\Gamma}_{\boldsymbol{x}}\boldsymbol{x}\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B},\\ \dot{\hat{\boldsymbol{K}}}_{\boldsymbol{r}} &= -\boldsymbol{\Gamma}_{\boldsymbol{r}}\boldsymbol{r}\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B},\\ \dot{\hat{\boldsymbol{\Theta}}} &= \boldsymbol{\Gamma}_{\boldsymbol{\Theta}}\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B},\end{aligned} \tag{5.15}$$

it follows that the time derivative of V becomes globally negative-semidefinite

$$\dot{V}(\boldsymbol{e}, \widetilde{\boldsymbol{K}}_{\boldsymbol{x}}, \widetilde{\boldsymbol{K}}_{\boldsymbol{r}}, \widetilde{\boldsymbol{\Theta}}) = -\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} \leq 0. \tag{5.16}$$

The Lyapunov function V can be viewed as an "energy-like" function, thus if the derivative of the energy function is not positive-definite, then the system energy dissipates and, consequently, every trajectory is bounded and exists globally for all time. According to Lyapunov's direct method, (5.16) implies global uniform stability of the origin, as well as uniform boundedness of $\boldsymbol{e}$, $\widetilde{\boldsymbol{K}}_{\boldsymbol{x}}$, $\widetilde{\boldsymbol{K}}_{\boldsymbol{r}}$, and $\widetilde{\boldsymbol{\Theta}}$. The parameter estimates $\hat{\boldsymbol{K}}_{\boldsymbol{x}}$, $\hat{\boldsymbol{K}}_{\boldsymbol{r}}$, and $\hat{\boldsymbol{\Theta}}$ are then uniformly bounded, and, since $\boldsymbol{r}$ is bounded and $\boldsymbol{A}_{\boldsymbol{m}}$ is Hurwitz, then $\boldsymbol{x}_{\boldsymbol{m}}$ and $\dot{\boldsymbol{x}}_{\boldsymbol{m}}$ are bounded. Therefore, the system state $\boldsymbol{x}$ is uniformly bounded and the control input $\boldsymbol{u}$ is bounded as well, which implies that $\dot{\boldsymbol{x}}$ an thus $\dot{\boldsymbol{e}}$ are bounded.

To prove asymptotic stability of the tracking error, Barbalat's lemma is employed, which essentially states that, if a time-dependent function tends to a limit and if its time derivative is uniformly continuous, then the derivative tends to zero. The proof of Barbalat's lemma can be found in [63, Chap. 8].

The second time derivative of V is

$$\ddot{V}(\boldsymbol{e}, \widetilde{\boldsymbol{K}}_{\boldsymbol{x}}, \widetilde{\boldsymbol{K}}_{\boldsymbol{r}}, \widetilde{\boldsymbol{\Theta}}) = -2\boldsymbol{e}^T \boldsymbol{Q} \dot{\boldsymbol{e}},$$

which is uniformly bounded, since $\dot{\boldsymbol{e}}$ is also uniformly bounded. Therefore, $\dot{V}$ is uniformly continuous in t. Additionally, since $V \geq 0$ and $\dot{V} \leq 0$, the Lyapunov function tends to a limit. Applying Barbalat's lemma

$$\lim_{t \to \infty} \dot{V}(\boldsymbol{e}, \widetilde{\boldsymbol{K}}_{\boldsymbol{x}}, \widetilde{\boldsymbol{K}}_{\boldsymbol{r}}, \widetilde{\boldsymbol{\Theta}}) = 0, \tag{5.17}$$

and, consequently, the state tracking error is asymptotically stable with $\boldsymbol{e}(t) \to \boldsymbol{0}$ as $t \to \infty$, whereas the adaptive gains remain uniformly bounded, as proven in [63, Chap. 9].

The individual components of the MRAC architecture, as depicted in Figure 5.1, and their design equations, are summarized in Table 5.1, which is an extract from Table 9.3 of [63, Chap. 9].

Table 5.1: MRAC architecture components.

Plant dynamics	$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{\Lambda}(\boldsymbol{u} + \boldsymbol{\Theta}^T \boldsymbol{\Phi}(\boldsymbol{x}))$	(5.1), (5.2)
Reference model	$\dot{\boldsymbol{x}}_{\boldsymbol{m}} = \boldsymbol{A}_{\boldsymbol{m}} \boldsymbol{x}_{\boldsymbol{m}} + \boldsymbol{B}_{\boldsymbol{m}} \boldsymbol{r}$	(5.3)
Adaptation laws	$\dot{\hat{\boldsymbol{K}}}_{\boldsymbol{x}} = -\boldsymbol{\Gamma}_{\boldsymbol{x}} \boldsymbol{x} \boldsymbol{e}^T \boldsymbol{P} \boldsymbol{B}$ $\dot{\hat{\boldsymbol{K}}}_{\boldsymbol{r}} = -\boldsymbol{\Gamma}_{\boldsymbol{r}} \boldsymbol{r} \boldsymbol{e}^T \boldsymbol{P} \boldsymbol{B}$ $\dot{\hat{\boldsymbol{\Theta}}} = \boldsymbol{\Gamma}_{\boldsymbol{\Theta}} \boldsymbol{\Phi}(\boldsymbol{x}) \boldsymbol{e}^T \boldsymbol{P} \boldsymbol{B}$	(5.15)
Control law	$\boldsymbol{u} = \hat{\boldsymbol{K}}_{\boldsymbol{x}}^T \boldsymbol{x} + \hat{\boldsymbol{K}}_{\boldsymbol{r}}^T \boldsymbol{r} - \hat{\boldsymbol{\Theta}}^T \boldsymbol{\Phi}(\boldsymbol{x})$	(5.7)

5.4 Robustness of MRAC

As explained in the previous section, MRAC is used in an adaptive control system to track a reference command signal by estimating the control gains so as to cancel out the unwanted effect of the system uncertainty. Asymptotic tracking can be achieved if the uncertainty has known functional characteristics, although its parameters may be uncertain. However, if the functional characteristics are also uncertain, the bounds on the adaptive parameters cannot be established by the Lyapunov stability analysis, causing MRAC to be generally non-robust, since unmodeled dynamics, unstructured uncertainty, and exogenous disturbances often cannot be fully captured in the model of the system. All these effects can produce closed-loop dynamics that can lead to instability when MRAC is used in an adaptive controller [48].

5.4.1 Parameter drift

Parameter drift occurs when a disturbance term causes an adaptive parameter to grow unbounded. Consider the system in (5.1), affected by an unknown bounded time-dependent disturbance $\boldsymbol{\xi}(t) \in \mathbb{R}^n$, $\|\boldsymbol{\xi}(t)\| \leq \xi_{\max}$, with known and constant upper bound $\xi_{\max} \geq 0$,

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{\Lambda}(\boldsymbol{u} + \boldsymbol{f}(\boldsymbol{x})) + \boldsymbol{\xi}(t), \tag{5.18}$$

and the reference model

$$\dot{\boldsymbol{x}}_{\boldsymbol{m}} = \boldsymbol{A}_{\boldsymbol{m}}\boldsymbol{x}_{\boldsymbol{m}} + \boldsymbol{B}_{\boldsymbol{m}}\boldsymbol{r}. \tag{5.19}$$

Proceeding analogously to Section 5.3 yields to the tracking error dynamics

$$\dot{\boldsymbol{e}} = \boldsymbol{A}_{\boldsymbol{m}}\boldsymbol{e} + \boldsymbol{B}\boldsymbol{\Lambda}\left[\widetilde{\boldsymbol{K}}_{\boldsymbol{x}}^T\boldsymbol{x} + \widetilde{\boldsymbol{K}}_{\boldsymbol{r}}^T\boldsymbol{r} - \widetilde{\boldsymbol{\Theta}}^T\boldsymbol{\Phi}(\boldsymbol{x})\right] + \boldsymbol{\xi}(t). \tag{5.20}$$

Stability of the adaptive system is analyzed by the Lyapunov's direct method. Using the Lyapunov function candidate in (5.11), the time derivative of V is

$$\begin{aligned}\dot{V}(\boldsymbol{e}, \widetilde{\boldsymbol{K}}_{\boldsymbol{x}}, \widetilde{\boldsymbol{K}}_{\boldsymbol{r}}, \widetilde{\boldsymbol{\Theta}}) = &- \boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} + 2\,\mathrm{Tr}\left(\widetilde{\boldsymbol{K}}_{\boldsymbol{x}}^T\left[\boldsymbol{\Gamma}_{\boldsymbol{x}}^{-1}\dot{\widetilde{\boldsymbol{K}}}_{\boldsymbol{x}} + \boldsymbol{x}\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}\right]\boldsymbol{\Lambda}\right) \\ &+ 2\,\mathrm{Tr}\left(\widetilde{\boldsymbol{K}}_{\boldsymbol{r}}^T\left[\boldsymbol{\Gamma}_{\boldsymbol{r}}^{-1}\dot{\widetilde{\boldsymbol{K}}}_{\boldsymbol{r}} + \boldsymbol{r}\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}\right]\boldsymbol{\Lambda}\right) \\ &+ 2\,\mathrm{Tr}\left(\widetilde{\boldsymbol{\Theta}}^T\left[\boldsymbol{\Gamma}_{\boldsymbol{\Theta}}^{-1}\dot{\widetilde{\boldsymbol{\Theta}}} - \boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}\right]\boldsymbol{\Lambda}\right) + 2\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{\xi}(t).\end{aligned} \tag{5.21}$$

Introducing the adaptive laws as in (5.15), yields

$$\dot{V}(\boldsymbol{e}, \widetilde{\boldsymbol{K}}_{\boldsymbol{x}}, \widetilde{\boldsymbol{K}}_{\boldsymbol{r}}, \widetilde{\boldsymbol{\Theta}}) = -\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} + 2\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{\xi}(t) \leq -\lambda_{\min}(\boldsymbol{Q})\|\boldsymbol{e}\|^2 + 2\|\boldsymbol{e}\|\lambda_{\max}(\boldsymbol{P})\xi_{\max}, \tag{5.22}$$

where $\lambda_{\min}(\boldsymbol{Q})$ and $\lambda_{\max}(\boldsymbol{P})$ are the minimum and maximum eigenvalues of matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$, respectively. Consequently, $\dot{V} \leq 0$ if

$$\|\boldsymbol{e}\| \geq 2\frac{\lambda_{\max}(\boldsymbol{P})}{\lambda_{\min}(\boldsymbol{Q})}\xi_{\max} = e_0. \tag{5.23}$$

The Lyapunov stability analysis provides no information on the bound of the adaptive gains, which can potentially become unbounded in certain situations, even though the tracking error norm remains finite at all times. This phenomenon, known as the parameter drift, shows that MRAC is not robust to bounded disturbances.

5.4.2 Modifications of the MRAC design for robustness

Some well-established design modifications and techniques to enforce robustness of MRAC are the dead-zone method, the projection method, the σ-modification and the e-modification. The dead-zone method and the projection method are common robust modification schemes based on the principle of limiting the adaptive parameters, whereas the σ-modification and the

e-modification are based on the principle of adding damping mechanisms in the adaptive laws to bound the adaptive parameters. In this section, the dead-zone and projection methods are described, since they are the ones chosen to enforce robustness in the experiments conducted in this thesis.

Dead-zone method

The dead-zone method stops the adaptation process when the tracking error norm falls below a certain threshold [70]. This method assures uniform ultimate boundedness of the estimation error of the adaptive gains, exemplarily shown on $\widetilde{\boldsymbol{\Theta}}$, preventing adaptive systems from attempting to reduce the noise instead of the tracking error, which can lead to a parameter drift.

The dead-zone method for an adaptive law as in (5.15) is stated as

$$\dot{\hat{\boldsymbol{\Theta}}} = \begin{cases} \boldsymbol{\Gamma}\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}, & \text{if } \|\boldsymbol{e}\| > e_0, \\ \boldsymbol{0}, & \text{if } \|\boldsymbol{e}\| \leq e_0, \end{cases} \tag{5.24}$$

where e_0 is a prescribed adaptation threshold, which is generally selected by trial-and-error taking into consideration that too large values would prematurely terminate the adaptation process, whereas too small values would be ineffective in preventing a parameter drift. The proof can be found in [48, Chap. 9].

Projection method

The aim of the modifications for robustness is to continuously modify the adaptive laws (5.15) in order to maintain the negative semi-definiteness of the Lyapunov function time derivative (5.21) while keeping the adaptive parameters $\hat{\boldsymbol{K}}_{\boldsymbol{x}}$, $\hat{\boldsymbol{K}}_{\boldsymbol{r}}$, and $\hat{\boldsymbol{\Theta}}$ uniformly bounded in time. The projection method achieves these design objectives by constraining the adaptive parameters to remain within a convex set [71], [72].

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable convex function, and consider the convex subset $\boldsymbol{\Omega}_\delta = \{\boldsymbol{\theta} \in \mathbb{R}^n : f(\boldsymbol{\theta}) \leq \delta\}$, with $0 \leq \delta \leq 1$. A Lipschitz-continuous version of the projection operator is defined as

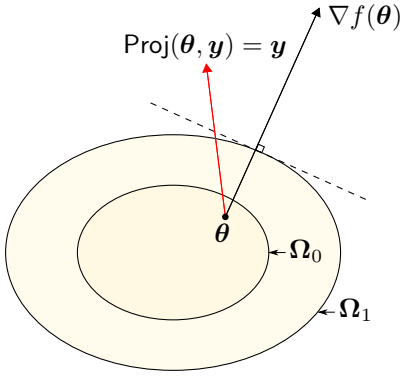
$$\text{Proj}(\boldsymbol{\theta}, \boldsymbol{y}) = \begin{cases} \boldsymbol{y}, & \text{if } f(\boldsymbol{\theta}) < 0, \\ \boldsymbol{y}, & \text{if } f(\boldsymbol{\theta}) \geq 0 \text{ and } \nabla f^T\boldsymbol{y} \leq 0, \\ \boldsymbol{y} - \frac{\boldsymbol{\Gamma}\nabla f \nabla f^T}{\|\nabla f\|_\Gamma^2}\boldsymbol{y} f(\boldsymbol{\theta}), & \text{if } f(\boldsymbol{\theta}) \geq 0 \text{ and } \nabla f^T\boldsymbol{y} > 0, \end{cases} \tag{5.25}$$

where $\boldsymbol{\Gamma} \in \mathbb{R}^{n\times n}$ is a constant symmetric positive-definite matrix and $\|\nabla f\|_\Gamma^2 = \nabla f^T\boldsymbol{\Gamma}\nabla f$.

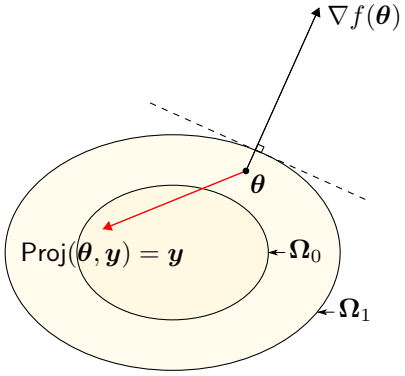
The projection operator permits the normal operation of the MRAC adaptation process provided that the adaptive parameters remain inside a predefined convex set. If the adaptive

parameters get closer to the boundary of the convex set, the projection operator changes the adaptation law to redirect them back into the convex set.

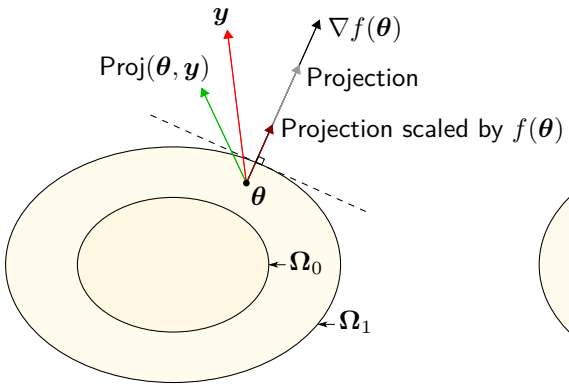
Figure 5.2 illustrates this concept, where $\mathbf{\Gamma}$ is set to be the identity matrix for the sake of ease of exposition. In this figure, $\mathbf{\Omega}_0 = \{\boldsymbol{\theta} : f(\boldsymbol{\theta}) \leq 0\}$ and $\mathbf{\Omega}_1 = \{\boldsymbol{\theta} : f(\boldsymbol{\theta}) \leq 1\}$. If $\boldsymbol{\theta}$ belongs to the convex set $\mathbf{\Omega}_0$, as in Figure 5.2a, the projection operator does not alter the vector $\boldsymbol{y}$. If $\boldsymbol{\theta}$ belongs to the annulus set $0 \leq f(\boldsymbol{\theta}) \leq 1$, and the parameter $\boldsymbol{y}$ points towards the inside of the set $\mathbf{\Omega}_1$, as in Figure 5.2b, no action is taken, since $\boldsymbol{y}$ will be driven further inside the predefined set. If $\boldsymbol{\theta}$ belongs to the annulus set $0 \leq f(\boldsymbol{\theta}) \leq 1$, and the parameter $\boldsymbol{y}$ points outwards of the predefined set $\mathbf{\Omega}_1$, as in Figure 5.2c, parameter $\boldsymbol{y}$ must be modified to prevent $\boldsymbol{\theta}$ from exiting the set. To do so, a projection of $\boldsymbol{y}$ normal to the boundary is scaled by $f(\boldsymbol{\theta})$ and subtracted from $\boldsymbol{y}$. Finally, if the parameter belongs to the boundary of the convex set $\mathbf{\Omega_1}$, as in Figure 5.2d, the result of the projection is a vector tangent to the boundary. For an arbitrary positive-definite matrix $\mathbf{\Gamma}$, similar sketches can be drawn.



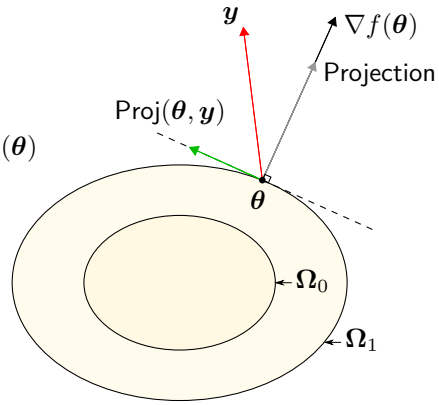
(a) If $f(\boldsymbol{\theta}) < 0$, no change is needed for $\boldsymbol{y}$.



(b) If $f(\boldsymbol{\theta}) \geq 0$ and $\nabla f^T \boldsymbol{y} \leq 0$, no change is needed for $\boldsymbol{y}$.



(c) If $0 \leq f(\boldsymbol{\theta}) < 1$ and $\nabla f^T \boldsymbol{y} > 0$, the projection operator subtracts a vector scaled by $f(\boldsymbol{\theta})$ and normal to the boundary from $\boldsymbol{y}$.



(d) If $f(\boldsymbol{\theta}) = 1$ and $\nabla f^T \boldsymbol{y} > 0$, the projection operator results in a vector tangent to the boundary.

Figure 5.2: Different actions of the projection operator in the projection-based adaptive law. In this figure, $\mathbf{\Omega}_0 = \{\boldsymbol{\theta} : f(\boldsymbol{\theta}) \leq 0\}$ and $\mathbf{\Omega}_1 = \{\boldsymbol{\theta} : f(\boldsymbol{\theta}) \leq 1\}$

.

For a robust projection-based MRAC design, $\boldsymbol{\theta}$ is the result of the integration of $\boldsymbol{y}$. The convex function $f(\boldsymbol{\theta})$ is defined as

$$f(\boldsymbol{\theta}) = \frac{(\epsilon_\theta + 1)\boldsymbol{\theta}^T\boldsymbol{\theta} - \theta_{\max}^2}{\epsilon_\theta \theta_{\max}^2}, \tag{5.26}$$

where $\theta_{\max}$ is the norm bound imposed on vector $\boldsymbol{\theta}$, and $\epsilon_\theta > 0$ is an a priori chosen projection tolerance bound [73, App. B].

The projection-based adaptive law is then defined as

$$\dot{\hat{\boldsymbol{\Theta}}} = \text{Proj}\left(\hat{\boldsymbol{\Theta}}, \boldsymbol{\Gamma}_{\boldsymbol{\Theta}} \boldsymbol{\Phi} \boldsymbol{e}^T \boldsymbol{P} \boldsymbol{B}\right). \tag{5.27}$$

Given an adaptive parameter $\hat{\boldsymbol{\Theta}}$ of a MIMO system, and being $\hat{\boldsymbol{\Theta}}_j$ the j-th column of $\hat{\boldsymbol{\Theta}}$, the projection operator Proj$(\hat{\boldsymbol{\Theta}}_j, \dot{\hat{\boldsymbol{\Theta}}}_j)$ guarantees uniform boundedness of the adaptive gains column-wise, ensuring that the columns $\hat{\boldsymbol{\Theta}}_j$ of the adaptive time-dependent parameter matrix $\hat{\boldsymbol{\Theta}}(t)$ do not exceed their prespecified bounds. The projection-based adaptive law also contributes to the negative semi-definiteness of the Lyapunov function. The proof of these statements can be found in [63, Chap. 11.4].

5.5 Implementation of the MRAC

An LQR PI + MRAC controller is designed to be implemented into the flight simulator of a commercial aircraft, as described in Chapter 2. The LQR baseline controller with PI action stabilizes the aircraft dynamics and regulates the speed and altitude of the aircraft under nominal conditions, and the MRAC acts as an augmentation control signal to mitigate the system uncertainties. The extended architecture used in this thesis is identical to the one proposed in [63, Chap. 11.5]. This augmentation approach enables the use of adaptive control improving the performance of the aircraft's underlying feedback controller, in this case the LQR PI baseline controller, instead of replacing it.

5.5.1 LQR PI feedback controller

The design of the LQR PI baseline controller is described in [63, Sec. 10.2]. In this section, only the extension of the state vector to include the integral of the error is reported.

As in (5.1), a MIMO nonlinear uncertain system is considered in the form

$$\begin{aligned} \dot{\boldsymbol{x}}_{\boldsymbol{p}} &= \boldsymbol{A}_{\boldsymbol{p}} \boldsymbol{x}_{\boldsymbol{p}} + \boldsymbol{B}_{\boldsymbol{p}} \boldsymbol{\Lambda} \left(\boldsymbol{u} + \boldsymbol{f}(\boldsymbol{x}_{\boldsymbol{p}})\right), \\ \boldsymbol{y} &= \boldsymbol{C}_{\boldsymbol{p}} \boldsymbol{x}_{\boldsymbol{p}}, \end{aligned} \tag{5.28}$$

with

$$\boldsymbol{f}(\boldsymbol{x}_{\boldsymbol{p}}) = \boldsymbol{\Theta}^T \boldsymbol{\Phi}(\boldsymbol{x}_{\boldsymbol{p}}), \tag{5.29}$$

where $\boldsymbol{x_p} \in \mathbb{R}^{n_p}$ is the system state vector, $\boldsymbol{u} \in \mathbb{R}^m$ is the control input, and $\boldsymbol{y_p} \in \mathbb{R}^m$ is the system regulated output. $\boldsymbol{B_p} \in \mathbb{R}^{n_p \times m}$ and $\boldsymbol{C_p} \in \mathbb{R}^{m \times n_p}$ are known and constant matrices, $\boldsymbol{A_p} \in \mathbb{R}^{n_p \times n_p}$ is an unknown and constant matrix, and $\boldsymbol{\Lambda} \in \mathbb{R}^{m \times m}$ is a constant diagonal unknown matrix with positive diagonal elements. The pair $(\boldsymbol{A_p}, \boldsymbol{B_p\Lambda})$ is assumed to be controllable. $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^m$ is the matched uncertainty, where $\boldsymbol{\Theta} \in \mathbb{R}^{N \times m}$ is the matrix of unknown and constant parameters, and $\boldsymbol{\Phi}(\boldsymbol{x_p}) \in \mathbb{R}^N$ is a known regressor vector, whose components are locally Lipschitz-continuous functions of $\boldsymbol{x_p}$.

The control goal is to design a control input, $\boldsymbol{u}$, such that the system regulated output tracks any bounded time-varying command, $\boldsymbol{r}(t) \in \mathbb{R}^m$, with bounded errors and in the presence of the system uncertainties $\{\boldsymbol{A_p}, \boldsymbol{\Lambda}, \boldsymbol{\Theta}\}$.

Introducing the system output tracking error, $\boldsymbol{e_y} = \boldsymbol{y}(t) - \boldsymbol{r}(t)$, into (5.28) yields to the extended open-loop dynamics

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{\Lambda}\left(\boldsymbol{u} + \boldsymbol{\Theta}^T\boldsymbol{\Phi}(\boldsymbol{x_p})\right) + \boldsymbol{B_m}\boldsymbol{r}, \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x}, \end{aligned} \tag{5.30}$$

in which the integrated output tracking error

$$\boldsymbol{e_{yI}}(t) = \int_0^t \boldsymbol{e_y}(\tau)d\tau \tag{5.31}$$

is appended to the state vector, whose dimension becomes $n = n_p + m$. The extended state vector is $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{e_{yI}^T} & \boldsymbol{x_p^T} \end{pmatrix}^T \in \mathbb{R}^n$. Integral control plays an important role in control system design because it achieves asymptotic regulation of a given output eliminating the residual steady-state error. The extended open-loop system matrices in (5.30) are then

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{0}_{m\times m} & \boldsymbol{C_p} \\ \boldsymbol{0}_{n_p\times m} & \boldsymbol{A_p} \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} \boldsymbol{0}_{m\times m} \\ \boldsymbol{B_p} \end{pmatrix}, \quad \boldsymbol{B_m} = \begin{pmatrix} -\boldsymbol{I}_{m\times m} \\ \boldsymbol{0}_{n_p\times m} \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} \boldsymbol{0}_{m\times m} & \boldsymbol{C_p} \end{pmatrix}. \tag{5.32}$$

Since nominal operating conditions are assumed for the extended system, namely $\boldsymbol{\Lambda} = \boldsymbol{I}_{m\times m}$ and $\boldsymbol{\Theta} = \boldsymbol{0}_{N\times m}$, the control law is

$$\boldsymbol{u}_{bl} = -\boldsymbol{K_x^T}\boldsymbol{x} = -\boldsymbol{K_I}\boldsymbol{e_{yI}} - \boldsymbol{K_P}\boldsymbol{x_p}, \tag{5.33}$$

in which the gain matrix $\boldsymbol{K_x}$ is partitioned into the integral gain $\boldsymbol{K_I}$ and the proportional gain $\boldsymbol{K_P}$ and designed using an optimal control technique, namely an LQR. The LQR is one of the most widely used control design methods in aerospace applications, since it allows the desired performance to be achieved while minimizing the control effort.

5.5.2 MRAC augmentation of the baseline controller

The tracking performance of the LQR PI tracking controller may deteriorate in the presence of the system uncertainties. An augmented adaptive controller is designed in order to restore the expected behavior. The design entails defining the reference model, formulating the tracking dynamics, and determining the adaptive laws. The design of the augmented adaptive controller is explained in [63, Sec. 10.3]. The main steps are reported here for convenience.

Given a reference Hurwitz matrix $\boldsymbol{A_m}$ and an unknown positive-definite diagonal constant matrix $\boldsymbol{\Lambda}$, it is assumed that there exists a constant and possibly unknown gain matrix $\boldsymbol{K_x} \in \mathbb{R}^{n\times m}$, such that

$$\boldsymbol{A_m} = \boldsymbol{A} + \boldsymbol{B\Lambda K_x^T}. \tag{5.34}$$

The reference model representative of the baseline closed-loop system dynamics is

$$\begin{aligned} \dot{\boldsymbol{x}}_{\boldsymbol{m}} &= \boldsymbol{A_m x_m} + \boldsymbol{B_m r}, \\ \boldsymbol{y_m} &= \boldsymbol{C x_m}, \end{aligned} \tag{5.35}$$

so that the extended system dynamics in (5.30) can be expressed as

$$\dot{\boldsymbol{x}} = \boldsymbol{A_m x} + \boldsymbol{B\Lambda}\left(\boldsymbol{u} - \boldsymbol{K_x^T x} + \boldsymbol{\Theta}^T\boldsymbol{\Phi}(\boldsymbol{x_p})\right) + \boldsymbol{B_m r}. \tag{5.36}$$

The control input is then the sum of the baseline component, $\boldsymbol{u_{bl}}$, and its adaptive augmentation, $\boldsymbol{u_{ad}}$,

$$\boldsymbol{u} = -\boldsymbol{K_x^T x} + \boldsymbol{u_{ad}} = \boldsymbol{u_{bl}} + \boldsymbol{u_{ad}}. \tag{5.37}$$

Substituting (5.37) into the system dynamics (5.36) yields

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A_m x} + \boldsymbol{B\Lambda}\left(\boldsymbol{u_{ad}} + \left(\boldsymbol{I}_{m\times m} - \boldsymbol{\Lambda}^{-1}\right)\boldsymbol{u_{bl}} + \boldsymbol{\Theta}^T\boldsymbol{\Phi}(\boldsymbol{x_p})\right) + \boldsymbol{B_m r} \\ &= \boldsymbol{A_m x} + \boldsymbol{B\Lambda}\left(\boldsymbol{u_{ad}} + \bar{\boldsymbol{\Theta}}^T\bar{\boldsymbol{\Phi}}(\boldsymbol{u_{bl}}, \boldsymbol{x_p})\right) + \boldsymbol{B_m r}, \end{aligned} \tag{5.38}$$

where the regressor vector and the matrix of unknown parameters are redefined as

$$\bar{\boldsymbol{\Phi}}(\boldsymbol{u_{bl}}, \boldsymbol{x_p}) = \begin{pmatrix} \boldsymbol{u_{bl}^T} & \boldsymbol{\Phi}^T(\boldsymbol{x_p}) \end{pmatrix}^T, \quad \bar{\boldsymbol{\Theta}} = \begin{pmatrix} \boldsymbol{K_u^T} & \boldsymbol{\Theta}^T \end{pmatrix}^T, \tag{5.39}$$

with $\boldsymbol{K_u^T} = \boldsymbol{I}_{m\times m} - \boldsymbol{\Lambda}^{-1}$. The adaptive control component is designed to compensate for the system uncertainty

$$\boldsymbol{u_{ad}} = -\hat{\bar{\boldsymbol{\Theta}}}^T\bar{\boldsymbol{\Phi}}(\boldsymbol{u_{bl}}, \boldsymbol{x_p}), \tag{5.40}$$

where $\hat{\bar{\boldsymbol{\Theta}}} \in \mathbb{R}^{(N+n)\times m}$ is the matrix of adaptive parameters. Substituting (5.40) into (5.38) yields

$$\dot{\boldsymbol{x}} = \boldsymbol{A_m x} - \boldsymbol{B\Lambda}\widetilde{\boldsymbol{\Theta}}^T\bar{\boldsymbol{\Phi}}(\boldsymbol{u_{bl}}, \boldsymbol{x_p}) + \boldsymbol{B_m r}, \tag{5.41}$$

with $\widetilde{\boldsymbol{\Theta}} = \hat{\bar{\boldsymbol{\Theta}}} - \bar{\boldsymbol{\Theta}}$.

Given the state tracking error $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x_m}$, the tracking error dynamics are

$$\dot{\boldsymbol{e}} = \boldsymbol{A_m e} - \boldsymbol{B\Lambda\widetilde{\Theta}}^T\bar{\boldsymbol{\Phi}}. \tag{5.42}$$

Considering the positive-definite adaptation rate matrix $\boldsymbol{\Gamma_{\bar{\Theta}}} = \boldsymbol{\Gamma_{\bar{\Theta}}^T} > \boldsymbol{0}$, the adaptive law is obtained through the Lyapunov function candidate

$$V(\boldsymbol{e}, \widetilde{\boldsymbol{\Theta}}) = \boldsymbol{e}^T\boldsymbol{Pe} + \mathrm{Tr}\left(\widetilde{\boldsymbol{\Theta}}^T\boldsymbol{\Gamma}_{\bar{\boldsymbol{\Theta}}}^{-1}\widetilde{\boldsymbol{\Theta}}\boldsymbol{\Lambda}\right), \tag{5.43}$$

where $\boldsymbol{P} = \boldsymbol{P}^T > \boldsymbol{0}$ is the solution of the algebraic Lyapunov equation

$$\boldsymbol{PA_m} + \boldsymbol{A_m^T P} = -\boldsymbol{Q}, \tag{5.44}$$

for some $\boldsymbol{Q} = \boldsymbol{Q}^T > \boldsymbol{0}$. The time derivative of V is then

$$\begin{aligned}\dot{V}(\boldsymbol{e}, \widetilde{\boldsymbol{\Theta}}) &= -\boldsymbol{e}^T\boldsymbol{Qe} - 2\boldsymbol{e}^T\boldsymbol{PB\Lambda\widetilde{\Theta}}^T\bar{\boldsymbol{\Phi}} + 2\,\mathrm{Tr}\left(\widetilde{\boldsymbol{\Theta}}^T\boldsymbol{\Gamma}_{\bar{\boldsymbol{\Theta}}}^{-1}\dot{\widetilde{\boldsymbol{\Theta}}}\boldsymbol{\Lambda}\right) \\ &= -\boldsymbol{e}^T\boldsymbol{Qe} + 2\,\mathrm{Tr}\left(\widetilde{\boldsymbol{\Theta}}^T\left[\boldsymbol{\Gamma}_{\bar{\boldsymbol{\Theta}}}^{-1}\dot{\widetilde{\boldsymbol{\Theta}}} - \bar{\boldsymbol{\Phi}}\boldsymbol{e}^T\boldsymbol{PB}\right]\boldsymbol{\Lambda}\right).\end{aligned} \tag{5.45}$$

To prove the uniform ultimate boundedness of $(\boldsymbol{e}, \widetilde{\boldsymbol{\Theta}})$, the adaptive laws are selected as

$$\dot{\widetilde{\boldsymbol{\Theta}}} = \boldsymbol{\Gamma}_{\bar{\boldsymbol{\Theta}}}\bar{\boldsymbol{\Phi}}(\boldsymbol{u_{bl}}, \boldsymbol{x_p})\boldsymbol{e}^T\boldsymbol{PB}, \tag{5.46}$$

leading to

$$\dot{V}(\boldsymbol{e}, \widetilde{\boldsymbol{\Theta}}) = -\boldsymbol{e}^T\boldsymbol{Qe} \leq 0. \tag{5.47}$$

The adaptive laws in (5.46) can be written in terms of the original parameters as

$$\begin{aligned}\dot{\hat{\boldsymbol{K}}}_{\boldsymbol{u}} &= \boldsymbol{\Gamma_u u_{bl} e^T PB}, \\ \dot{\hat{\boldsymbol{\Theta}}} &= \boldsymbol{\Gamma_\Theta \Phi(x_p) e^T PB},\end{aligned} \tag{5.48}$$

where $\boldsymbol{\Gamma_u}$ and $\boldsymbol{\Gamma_\Theta}$ are the rates of adaptation for the uncertainties corresponding to $\boldsymbol{x}$ and $\boldsymbol{\Phi}(\boldsymbol{x_p})$, with $\boldsymbol{\Gamma_{\bar{\Theta}}} = \begin{pmatrix} \boldsymbol{\Gamma_u} & \boldsymbol{0}_{n\times m} \\ \boldsymbol{0}_{N\times m} & \boldsymbol{\Gamma_\Theta} \end{pmatrix}$. The total control input can be also expressed as

$$\boldsymbol{u} = \boldsymbol{u}_{bl} + \boldsymbol{u}_{ad} = -\boldsymbol{K}_{\boldsymbol{x}}^T\boldsymbol{x} + \left[-\hat{\boldsymbol{K}}_{\boldsymbol{u}}^T\boldsymbol{u}_{bl} - \hat{\boldsymbol{\Theta}}\boldsymbol{\Phi}^T(\boldsymbol{x_p})\right]. \tag{5.49}$$

The components of the MRAC augmentation of the baseline controller and their design equations are summarized in Table 5.2, which is an extract from Table 10.2 of [63, Chap. 10]. The overall control block diagram is shown in Figure 5.3.

Table 5.2: MRAC augmentation of a baseline LQR PI controller architecture components.

Plant dynamics	$\dot{\boldsymbol{x}}_{\boldsymbol{p}} = \boldsymbol{A}_{\boldsymbol{p}}\boldsymbol{x}_{\boldsymbol{p}} + \boldsymbol{B}_{\boldsymbol{p}}\boldsymbol{\Lambda}(\boldsymbol{u} + \boldsymbol{\Theta}^T\boldsymbol{\Phi}(\boldsymbol{x}_{\boldsymbol{p}}))$	(5.28), (5.29)
Extended dynamics	$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{\Lambda}(\boldsymbol{u} + \boldsymbol{\Theta}^T\boldsymbol{\Phi}(\boldsymbol{x}_{\boldsymbol{p}})) + \boldsymbol{B}_{\boldsymbol{m}}\boldsymbol{r}$	(5.30)
Reference model	$\dot{\boldsymbol{x}}_{\boldsymbol{m}} = \boldsymbol{A}_{\boldsymbol{m}}\boldsymbol{x}_{\boldsymbol{m}} + \boldsymbol{B}_{\boldsymbol{m}}\boldsymbol{r}$	(5.35)
Adaptation laws	$\dot{\hat{\boldsymbol{K}}}_{\boldsymbol{u}} = \boldsymbol{\Gamma}_{\boldsymbol{u}}\boldsymbol{u}_{bl}\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}$ $\dot{\hat{\boldsymbol{\Theta}}} = \boldsymbol{\Gamma}_{\boldsymbol{\Theta}}\boldsymbol{\Phi}(\boldsymbol{x}_{\boldsymbol{p}})\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}$	(5.48)
Control laws	$\boldsymbol{u} = \boldsymbol{u}_{bl} + \boldsymbol{u}_{ad}$ $\boldsymbol{u}_{bl} = -\boldsymbol{K}_{\boldsymbol{x}}^T\boldsymbol{x}$ $\boldsymbol{u}_{ad} = -\hat{\boldsymbol{K}}_{\boldsymbol{u}}^T\boldsymbol{u}_{bl} - \hat{\boldsymbol{\Theta}}\boldsymbol{\Phi}^T(\boldsymbol{x}_{\boldsymbol{p}})$	(5.49)

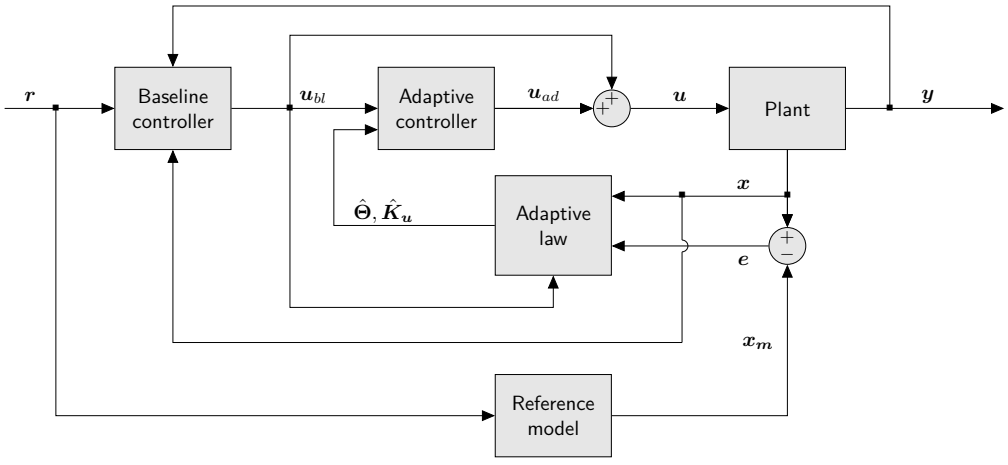


Figure 5.3: Block diagram of MRAC augmentation of a baseline controller.

5.6 Experimental results

The following experiment is conducted, in which the proposed MRAC augmentation of an LQR PI baseline controller is applied to an Airbus A320-231 (A320) flight simulator, which includes the dead zone modification to prevent parameter drift, and the projection operator to constrain the adaptive parameters. The adaptive augmentation approach followed allows the preexisting baseline controller to be retained, avoiding a complete redesign of the control system to introduce the adaptive control action. The equations of motion of the A320 aircraft are those described in Section 2.1.1.

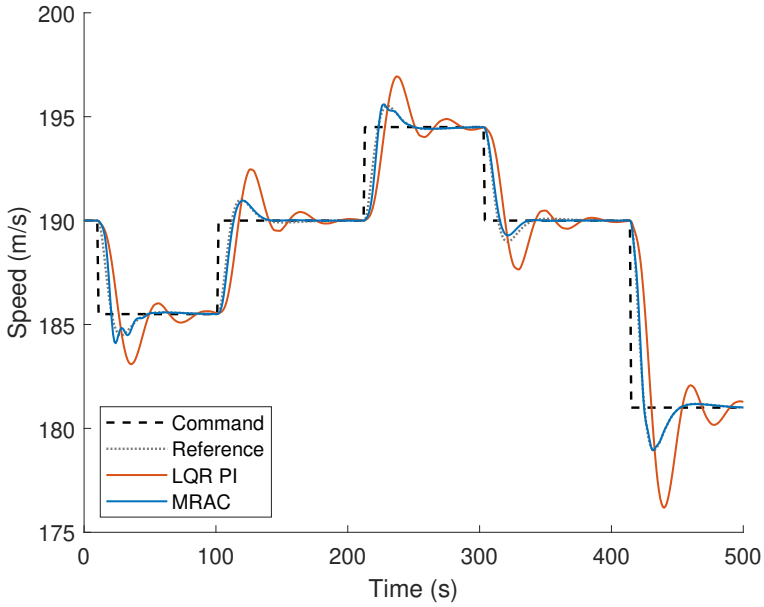
Experiment: Trajectory tracking performance of an MRAC augmentation of an LQR PI baseline controller

The aim of the experiment is to compare the performance of the MRAC augmentation of the LQR PI baseline controller with the performance of the same controller without MRAC augmentation when tracking a series of step-input commands of different magnitudes for both the speed and altitude of the aircraft, which are the variables regulated by the LQR PI controller. The control inputs are the lift coefficient and the thrust. The reference model for the MRAC augmentation scheme is selected to represent the closed-loop system dynamics of the baseline controller, obtained by substituting the baseline controller (5.33) into the system (5.30), with $\mathbf{\Lambda} = \boldsymbol{I}_{m\times m}$ and $\mathbf{\Theta} = \mathbf{0}_{N\times m}$. Therefore, if no uncertainties are considered, the responses of the reference and actual systems coincide. The goal of the MRAC augmentation of the baseline LQR PI controller is that the output variables of the aircraft track a command signal $\mathbf{r}(t) = (V_{cmd}(t), h_{e,cmd}(t))$. Therefore, the output variables considered are $\mathbf{y}(t) = (V(t), h_e(t))$. The extended state vector of the aircraft controller is $\mathbf{x}(t) = (V_I(t), h_{eI}(t), V(t), \gamma(t), h_e(t))$, where $V_I(t)$ and $h_{eI}(t)$ are the integrated output tracking errors of the speed and the altitude, respectively, defined as in (5.31). Matched uncertainties are introduced in the aircraft model to degrade the performance of the baseline controller and test the capacity of the MRAC to recover the closed-loop performance of the reference model.

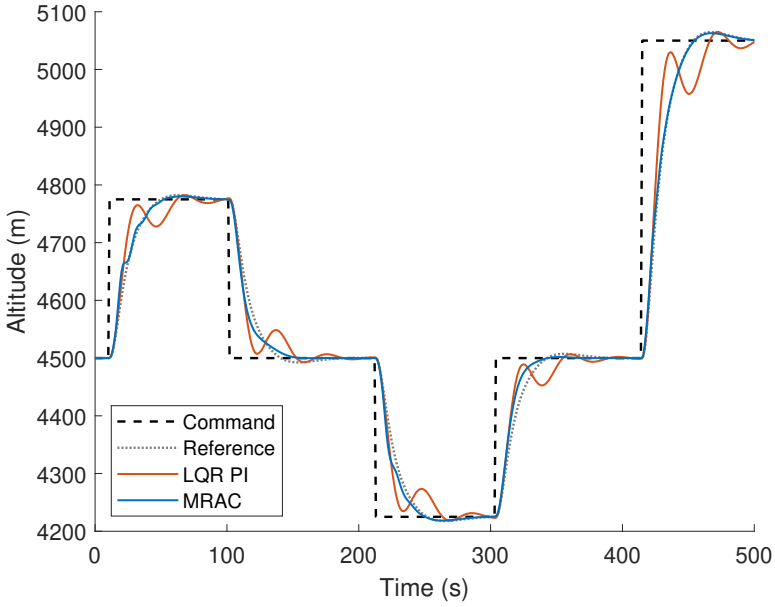
Figure 5.4 depicts the tracking performance of the aircraft when matched uncertainties are introduced in the model. The black dashed line represents the commanded speed and altitude, the dotted grey line is the output of the reference model, and the orange and blue lines represent the output signals when only the baseline controller is active, and when it is augmented with the adaptive controller, respectively. The figure shows that the perturbations introduced in the model destabilize the baseline controller. The MRAC recovers the desired reference closed-loop performance, tracking a similar trajectory as the reference model.

Figure 5.5 represents the norm of the state tracking error, which is calculated as the norm of the difference between the states of the plant model and the reference model, namely as $\|\mathbf{x}-\mathbf{x_m}\|_2$, being $\mathbf{x}$ the extended state of the actual aircraft model and $\mathbf{x_m}$ the extended state

of the reference aircraft model. The figure shows that the state tracking error is significantly reduced when the MRAC is active.



(a) Performances in tracking speed changes.



(b) Performances in tracking altitude changes.

Figure 5.4: Output tracking performances of an Airbus A320 aircraft model with matched uncertainties, using the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation.

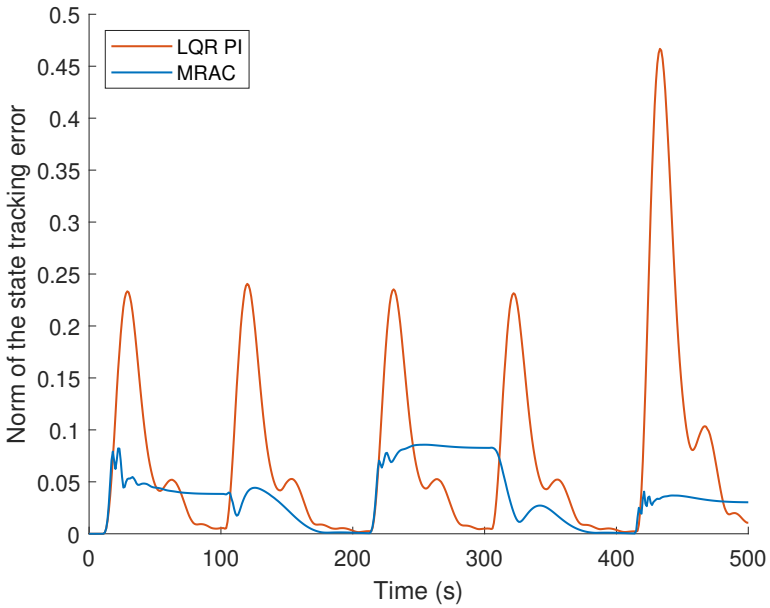


Figure 5.5: Norms of the state tracking errors observed on an Airbus A320 aircraft model with matched uncertainties, using the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation.

5.7 Conclusion

The results of the experiment described in this chapter show that the MRAC augmentation of an LQR PI baseline controller achieves a much more precise tracking of the desired trajectory when matched uncertainties are introduced in the aircraft model, as compared to the performance of the aircraft when the only control action is the one provided by the baseline controller. To prevent the parameter drift that these uncertainties may cause, robustness modifications are also included in the MRAC controller. In this way, the aircraft performance shows a similar behavior as the reference model, even in the presence of model uncertainties and external disturbances. This augmented controller will be employed in the next chapter in combination with ILC to achieve transfer learning among different aircraft.

6

MRAC–ILC Multi-Aircraft Transfer Learning

Consecutive flights operating in a TMA are generally carried out by different aircraft, so different dynamical systems perform each iteration. In Chapter 5, the MRAC augmentation of an underlying baseline controller was applied to a simulated Airbus A320 aircraft, showing accurate trajectory tracking even if model uncertainties are introduced in the nominal model. This result is due to the fact that MRAC is able to recover the closed-loop performance of the system, forcing it to behave close to a reference model. The results obtained in the previous chapters are the base of the multi-aircraft transfer learning framework proposed here, in which the MRAC is used within the ILC algorithm. The essence of this framework is that knowledge acquired by an aircraft equipped with an MRAC in following a trajectory in previous iterations of the ILC algorithm can be directly transferred to a different aircraft to improve precision in the following iterations of the ILC algorithm, as long as the latter is equipped with an MRAC with the same reference model.

6.1 Introduction

The aim of this chapter is to design a control framework for precise trajectory tracking based on ILC which is capable of improving the aircraft performance over iterations, and in addition, is able to transfer the learned trajectories to different aircraft, making them behave in a predictable way even in the presence of unknown disturbances and changing dynamics.

The combination of adaptive control and learning control has been explored in other works. In [74], a model reference adaptive ILC strategy is presented for single-input single-output linear time-invariant robot manipulators with unknown parameters, performing repetitive tasks, through a discrete-type parametric adaptation law that refines the transient response from iteration to iteration. ILC is combined with an MRAC strategy in [75] to design a control parameter adaptive law, in which the parameters of the control systems at each iteration are

adapted based on the results of previous iterations. In [76], a reinforcement learning-based flight control system is designed to improve the transient response performance of a closed-loop model reference adaptive control system. A framework of $\mathcal{L}_1$ adaptive feedback control with parallel ILC is proposed in [77], where the combination of $\mathcal{L}_1$ and ILC is successfully applied to a simulated system. In this work, ILC ensures that the plant uncertainty is sufficiently small for the $\mathcal{L}_1$ controller, which is in charge of precision motion control, to compensate for parametric uncertainties. In [78], experimental results are reported of a serial framework of $\mathcal{L}_1$ adaptive control and ILC for quadrotor trajectory tracking under changing dynamics. The knowledge acquired by the ILC algorithm over iterations is transferred across different systems, since the $\mathcal{L}_1$ adaptive controller copes with the underlying changes in the system dynamics.

In this chapter, a serial MRAC-ILC architecture is proposed, in which the MRAC augmentation of the LQR PI baseline controller described in Chapter 5 is used as an underlying controller to achieve robust repeatable behaviour, whereas the ILC acts as a high-level adaptation scheme and compensates for repetitive disturbances. This approach is based on the $\mathcal{L}_1$-ILC framework presented in [79], in which it is proven that this framework enables transfer learning between dynamically different systems, where learned experience of one system serves to another different system to achieve high accuracy trajectory tracking. Different control methods have already been applied to transfer learning tasks, such as [80], where feedback linearization and adaptive control techniques are used to transfer controllers between systems that have similar control structure, and [81], in which the relationship between the dynamics of heterogeneous UAVs is used to improve the tracking performance of the target UAV by learning from the tracking errors made by the training UAV.

6.2 Combined MRAC-ILC framework

As said earlier, the objective of the MRAC-ILC framework is to force the controlled system to behave in a repeatable and reliable way, while achieving precise trajectory tracking despite the presence of unknown disturbances and changing dynamics. The MRAC makes the system performance repeatable, forcing it to behave close to a reference model, even if it is affected by model uncertainties and disturbances. However, the MRAC by itself may not suffice to drive the system along a desired trajectory if the model errors and perturbations affecting the system move it away from this trajectory. In this case, the ILC gradually compensates for repetitive disturbances improving the tracking performance of the controlled system over multiple executions of the task.

The scheme of the MRAC-ILC framework is shown in Figure 6.1. The MRAC augmentation of the baseline controller can be regarded as the closed-loop underlying controller, whereas the ILC acts as a higher-level controller. This way, the function of the former controller to achieve repeatability is decoupled from the task of the latter of improving the tracking performance.

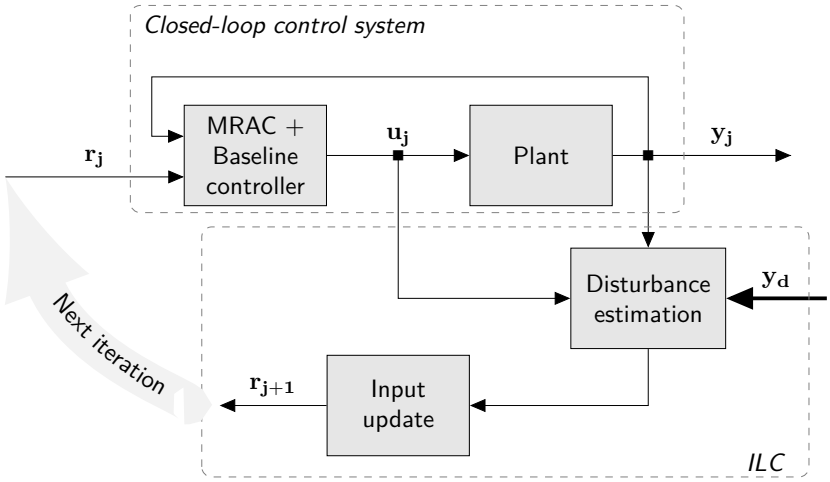


Figure 6.1: MRAC-ILC block diagram. Two main subsystems can be distinguished: the closed-loop control system represented by the MRAC augmentation of the baseline controller, which makes the system behave in a repeatable predefined way, and the ILC, which improves the tracking performance over iterations.

6.2.1 MRAC-ILC architecture

It is assumed that the uncertain and changing system dynamics, the plant in Figure 6.1, is a MIMO system described by the following equations:

$$\begin{aligned} \dot{\mathbf{x}}_\mathbf{p} &= \boldsymbol{A_p}\mathbf{x_p} + \boldsymbol{B_p}\boldsymbol{\Lambda}\left(\mathbf{u} + \boldsymbol{\Theta}^T\boldsymbol{\Phi}(\mathbf{x_p})\right), \\ \mathbf{y} &= \boldsymbol{C_p}\mathbf{x_p}, \end{aligned} \tag{6.1}$$

where $\mathbf{x_p} \in \mathbb{R}^{n_p}$ is the system state vector, $\mathbf{u} \in \mathbb{R}^m$ is the control input, and $\mathbf{y_p} \in \mathbb{R}^m$ is the system regulated output. $\boldsymbol{B_p} \in \mathbb{R}^{n_p \times m}$ and $\boldsymbol{C_p} \in \mathbb{R}^{m \times n_p}$ are known and constant matrices, $\boldsymbol{A_p} \in \mathbb{R}^{n_p \times n_p}$ is an unknown and constant matrix, and $\boldsymbol{\Lambda} \in \mathbb{R}^{m \times m}$ is a constant diagonal unknown matrix with positive diagonal elements. The pair $(\boldsymbol{A_p}, \boldsymbol{B_p}\boldsymbol{\Lambda})$ is assumed to be controllable. The matched uncertainty is composed of the matrix of unknown and constant parameters $\boldsymbol{\Theta} \in \mathbb{R}^{N \times m}$, and a known regressor vector $\boldsymbol{\Phi}(\mathbf{x_p}) \in \mathbb{R}^N$, whose components are locally Lipschitz-continuous functions of $\mathbf{x_p}$.

Consider the extended system (5.30) described in Section 5.5.1

$$\begin{aligned} \dot{\mathbf{x}} &= \boldsymbol{A}\mathbf{x} + \boldsymbol{B}\boldsymbol{\Lambda}\left(\mathbf{u} + \boldsymbol{\Theta}^T\boldsymbol{\Phi}(\mathbf{x_p})\right) + \boldsymbol{B_m}\mathbf{r}, \\ \mathbf{y} &= \boldsymbol{C}\mathbf{x}, \end{aligned}$$

where $\mathbf{x} = \begin{pmatrix} \boldsymbol{e}_{\boldsymbol{yI}}^T & \mathbf{x_p}^T \end{pmatrix}^T \in \mathbb{R}^n$ is the extended state vector, with $\boldsymbol{e_{yI}}(t) = \int_0^t \boldsymbol{e_y}(\tau)d\tau$, and $\mathbf{r}$ is the commanded reference trajectory fed to the controller. The matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{B_m}$, and $\boldsymbol{C}$ are given in (5.32).

Consider the total control law (5.49)

$$\mathbf{u} = \mathbf{u}_{bl} + \mathbf{u}_{ad} = -\boldsymbol{K}_{\boldsymbol{x}}^T \mathbf{x} + \left[-\hat{\boldsymbol{K}}_{\boldsymbol{u}}^T \mathbf{u}_{bl} - \hat{\boldsymbol{\Theta}} \boldsymbol{\Phi}^T(\mathbf{x_p})\right],$$

where the adaptive gains $\hat{\boldsymbol{K}}_{\boldsymbol{u}}$ and $\hat{\boldsymbol{\Theta}}$ are updated following the adaptive laws in (5.48). This controller forces the system to behave as the reference model

$$\begin{aligned} \dot{\mathbf{x}}_\mathbf{m} &= \boldsymbol{A}_m \mathbf{x}_\mathbf{m} + \boldsymbol{B}_m \mathbf{r}, \\ \mathbf{y}_\mathbf{m} &= \boldsymbol{C} \mathbf{x}_\mathbf{m}, \end{aligned} \tag{6.2}$$

with

$$\boldsymbol{A}_{\boldsymbol{m}} = \boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{K}_{\boldsymbol{x}}^T. \tag{6.3}$$

Once repeatability is achieved by the adaptive controller, the goal of the ILC is to provide a commanded reference signal that enables the system to track a desired output trajectory, $\mathbf{y_d}(t)$, which is defined over a finite-time interval and is assumed to be feasible with respect to the nominal dynamics of the plant controlled by the MRAC augmentation of the baseline controller, the closed-loop system in Figure 6.1. The technique for ILC described in Chapter 3 and summarized here can be applied to this system. It is assumed that an approximate model of the system is known in the form

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \\ \mathbf{y}(t) &= \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), t). \end{aligned} \tag{6.4}$$

The output, state, and input signals are discretized and the deviations with respect to the desired output trajectory and its corresponding state and input, $\tilde{\boldsymbol{y}}(t) = \mathbf{y}(t) - \mathbf{y_d}(t)$, $\tilde{\boldsymbol{x}}(t) = \mathbf{x}(t) - \mathbf{x_d}(t)$, and $\tilde{\boldsymbol{u}}(t) = \mathbf{u}(t) - \mathbf{u_d}(t)$, are then lifted as stacked vectors of all discretization time-steps (3.8), obtaining the triple of lifted vectors $(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y})$.

The lifted representation of the system is given in (3.11) as

$$\begin{aligned} \boldsymbol{x}_j &= \boldsymbol{F}\boldsymbol{u}_j + \boldsymbol{d}_j, \\ \boldsymbol{y}_j &= \boldsymbol{G}\boldsymbol{x}_j + \boldsymbol{H}\boldsymbol{u}_j, \end{aligned}$$

where the subscript j indicates the $j-$th execution of the desired task, $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{H}$ are constant matrices derived from the nominal model, and $\boldsymbol{d}_j$ captures the repetitive disturbances along the trajectory. This disturbances are initially unknown and need to be estimated based on measurements from previous iterations to compute the predicted disturbances for the following iteration, $\boldsymbol{d}_{j+1}^p$.

The updated input that minimizes the deviation from the desired trajectory in the following iteration, $\boldsymbol{u}_{j+1}$, is computed solving an optimization problem. The update rule considered in

(3.17) can be expressed as

$$\begin{aligned} \min_{\boldsymbol{u}_{j+1}} \quad & \|\boldsymbol{F}\boldsymbol{u}_{j+1} + \boldsymbol{d}_{j+1}^{p}\|_{\ell} + \alpha\|\boldsymbol{D}\boldsymbol{u}_{j+1}\|_{\ell}, \\ \text{subject to:} \quad & \boldsymbol{L}\boldsymbol{u}_{j+1} \leq \boldsymbol{q}_{\max}, \end{aligned}$$

where $\alpha \geq 0$ and $\boldsymbol{D}$ are introduced to penalize the input or approximations of its derivatives in order to enforce the smoothness of the optimal problem solution. Different cost functions can be used, which may include terms penalizing other signals such as the input change from iteration to iteration.

The new reference trajectory is obtained as in (3.20), namely

$$\begin{aligned} \boldsymbol{x}_r &= \boldsymbol{F}\boldsymbol{u}_{j+1} + \boldsymbol{x}_d, \\ \boldsymbol{y}_r &= \boldsymbol{G}\boldsymbol{x}_r + \boldsymbol{H}\boldsymbol{u}_{j+1} + \boldsymbol{y}_d, \end{aligned}$$

where $\boldsymbol{x}_r \in \mathbb{R}^{Nn_x}$ and $\boldsymbol{y}_r \in \mathbb{R}^{Nn_y}$ are, respectively, the new reference state and output lifted vectors, and $\boldsymbol{x}_d \in \mathbb{R}^{Nn_x}$ and $\boldsymbol{y}_d \in \mathbb{R}^{Nn_y}$ are, respectively, the desired state and output vectors also in a lifted form. The reference trajectory, $\mathbf{r}_{j+1}$ to be fed to the MRAC augmentation of the baseline LQR PI controller in the following iteration can be obtained from the unlifted representation form of $\boldsymbol{x}_r$ and $\boldsymbol{y}_r$.

6.2.2 Transfer learning

The introduction of the MRAC in ILC allows learning to be transferred among different systems. The goal of transfer learning between two different systems, usually referred to as training and target system, is to improve the performance of the target system by transferring to it the knowledge acquired by the training system. In the context of this chapter, the knowledge to be transferred includes the learned reference trajectory and the estimated disturbances.

The concept of transfer learning among systems in ILC has been introduced in [78] and [79], in which an $\mathcal{L}_1$ adaptive controller is used as the underlying trajectory tracking controller of quadrotors with different dynamics, showing that the trajectories learned by a quadrotor in previous iterations of the ILC algorithm can be transferred to another quadrotor to reduce the tracking error in the following iterations of the ILC algorithm. In this chapter, instead of an $\mathcal{L}_1$ adaptive controller, an MRAC is employed. The main differences between these adaptive controllers are the specifications of the control problem they are applied to, which in the $\mathcal{L}_1$ adaptive controller explicitly includes the robustness specification. Therefore, it decouples adaptation from robustness and only compensates for uncertainties that relay within a bandwidth determined by a bandwidth-limited linear filter [73, Chap. 1].

To achieve precision in trajectory tracking for a given desired trajectory in the presence of uncertain environment conditions, the ILC optimizes the system performance over multiple executions of the task. Nevertheless, system dynamics may keep changing after learning, and ILC has the limitation of not being able to generalize previously learned tasks to new

dynamical systems. The MRAC makes systems designed with the same reference model behave in a repeatable predefined way, even in the presence of unknown and changing disturbances, making the learned trajectories exchangeable without further modifications.

The transfer learning scheme is shown in Figure 6.2. Given two different closed-loop control systems, both equipped with an MRAC augmentation of a baseline controller with the same reference model, the objective of the second control system is to precisely track a desired trajectory, $\mathbf{y_d}$, based on the experience gained by the first control system. At iteration j, a previously computed commanded reference signal, $\mathbf{r}_j$, is fed to the adaptive controller of system 1, which generates a control action that, due to unmodelled dynamics and external disturbances, drives the system through an output signal $\mathbf{y}_{1,j}$ that may be deviated from the desired trajectory. The ILC estimates this deviation and optimally generates a new reference signal for the following iteration, $\mathbf{r}_{j+1}$. Since the adaptive controllers of the systems are designed using the same reference model, the signal $\mathbf{r}_{j+1}$ can be directly introduced into the controller of control system 2. The resulting output signal $\mathbf{y}_{2,j+1}$ incorporates the corrections determined by the ILC given the experience gathered by control system 1, achieving high performance on tracking the desired trajectory.

In summary, this transfer learning architecture achieves high-accuracy in trajectory tracking despite model uncertainties and disturbances. If the system dynamics changes over iterations, the learned trajectories can be transferred to dynamically different systems, which thus achieve similar performances in tracking a given trajectory without needing to relearn it.

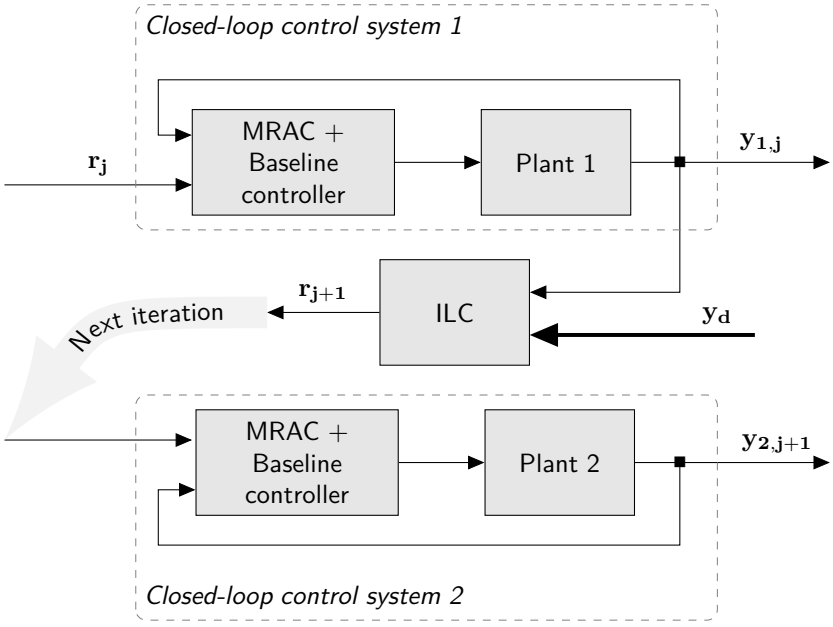


Figure 6.2: MRAC-ILC transfer learning block diagram. The closed-loop systems 1 and 2 are controlled by an MRAC augmentation of their baseline controllers with the same reference model. The ILC improves the trajectory tracking performance of system 2 by learning from the deviations suffered by system 1.

6.3 Experimental results

In this section, the results of the application of the MRAC-ILC framework to different trajectory tracking problems that involve different commercial aircraft are described.

Three different aircraft are considered in the numerical experiments, namely an Airbus A320-231 (A320), a Boeing 767-300ER (B767), and an Embraer ERJ 190-200 IGW (E195), which belong to different classes, being a mid-sized aircraft, a large-to-mid-sized aircraft, and a mid-to-light-sized aircraft, respectively. The equations of motion and the flight envelope constraints of the aircraft are those described in Section 2.1. The performance parameters of each aircraft have been obtained from BADA, and are listed in Appendix A. Note that, although in this chapter the three aircraft have identical controllers, the parameters of the LQR PI controller and those of its MRAC augmentation can be tuned to fit each aircraft's dynamics, as long as the reference model of the MRAC is the same.

Marking reference to Figure 6.1, the aim of the aircraft's underlying controller, namely the MRAC augmentation of the baseline LQR PI controller, is that two components of the output variables track those of a command signal $\mathbf{r}(t) = (V_{cmd}(t), h_{e,cmd}(t))$. Therefore, the output variables considered by the underlying controller are $\mathbf{y_{uc}}(t) = (V(t), h_e(t))$. The extended state vector of the underlying aircraft controller is $\mathbf{x_{uc}}(t) = (V_I(t), h_{eI}(t), V(t), \gamma(t), h_e(t))$, where $V_I(t)$ and $h_{eI}(t)$ are the integrated output tracking errors of the speed and the altitude, respectively, defined as in (5.31).

The reference signal is computed at each iteration by the ILC in such a way that all the components of the output of the plant follow a desired trajectory $\mathbf{y_d}(t) = (V_d(t), \gamma_d(t), x_{e,d}(t), h_{e,d}(t), m_d(t))$. At each iteration, the ILC estimates the deviations of the actual output trajectory of the plant from the desired output trajectory using a Kalman filter. The ILC takes into account all the state variables of the plant model described in Section 2.1.1, namely $\mathbf{x}(t) = (V(t), \gamma(t), x_e(t), h_e(t), m(t))$, which are assumed to be directly measurable, thus the output vector of the plant used by the ILC is $\mathbf{y}(t) = \mathbf{x}(t)$.

The following errors are used to evaluate the performance of both the underlying controller and the ILC:

- The state tracking error measures the performance of the MRAC augmentation of the LQR PI in following the reference model. The norm of the state tracking error at each time step is calculated as the norm of the difference between the states of the plant model and the reference model, namely as $\|\mathbf{x_{uc}} - \mathbf{x_m}\|_2$, being $\mathbf{x_m}$ the extended state of the reference aircraft model.

- The weighted state error characterizes the performance of the MRAC-ILC framework in following the desired trajectory at each iteration, scaled by a weighting matrix $\boldsymbol{S}$, that is

$$e_{ws,j} = \|\boldsymbol{S}\boldsymbol{y}_j\|_2,$$

where $\boldsymbol{y}_j$ is the lifted output error vector of the ILC at iteration j, defined as in (3.8). The weighted scaling matrix, $\boldsymbol{S}$, is used to assign larger weights to the output errors of the position variables of the aircraft to achieve better precision in these output variables.

The following four trajectory tracking experiments are carried out.

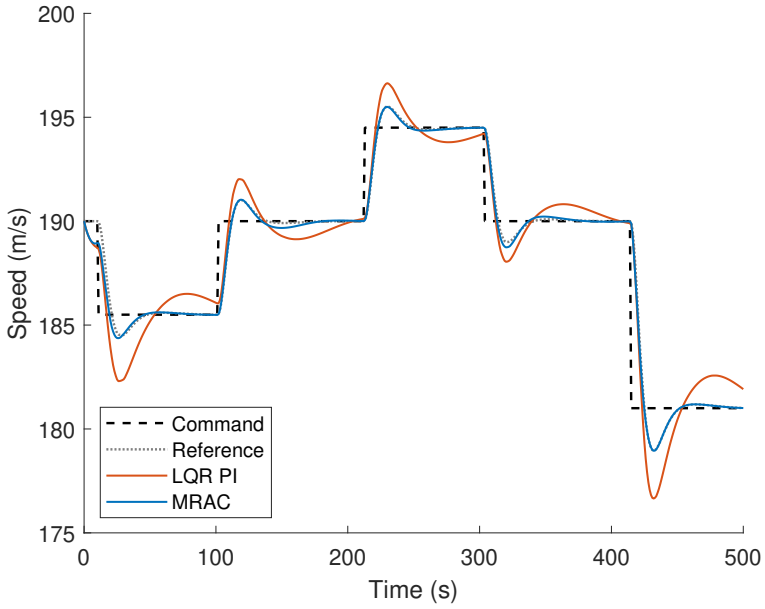
Experiment 1: MRAC augmentation of a baseline LQR PI controller with mismatch between reference and plant models

The goal of this experiment is to demonstrate the capability of the MRAC to force different aircraft to behave as a common reference model. The reference model in the MRAC controller is that of an A320 aircraft with a baseline LQR PI controller. The plant model is that of a B767 aircraft, which is assumed to be not affected by external disturbances. Since external disturbances do not affect the plant, only the first iteration of the ILC is executed. This amounts to not using the ILC controller and testing only the closed-loop control system in Figure 6.1. The reference signal $\mathbf{r}_1$ is composed of a series of steps of different magnitude for both the speed and altitude for the B767 aircraft.

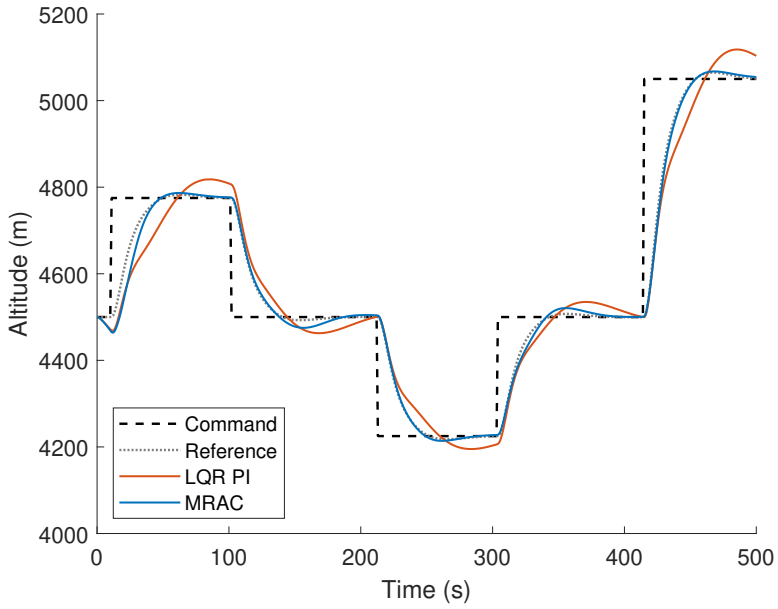
In Figure 6.3, the black dashed lines represent the reference speed and altitude to be tracked, the dotted gray lines represent the outputs of the reference model, which is that of an A320 aircraft, and the blue lines represent the outputs of the plant model, which is that of a B767 aircraft, using the MRAC augmentation of the baseline LQR PI controller. Finally, the orange lines represent the outputs of the plant model when only the baseline LQR PI controller is employed. As expected, since the baseline controller is designed for an A320 aircraft, its tracking performance on the plant model, which is a B767 aircraft, shows significant deviations when the MRAC is not used. On the contrary, when the MRAC is used, its tracking performance on the plant model is similar to the performance of the baseline LQR PI controller on the reference model.

Figure 6.4 shows the norms of the state tracking errors observed at each time step using the MRAC augmentation of the baseline LQR PI controller and the LQR PI baseline controller without MRAC augmentation. The norm of the state tracking error confirms that the state tracking error is considerably higher when only the baseline controller is active, as compared to the state tracking error when using the MRAC augmentation of the baseline controller.

The results of this experiment serve as the basis for the following experiments, in which external disturbances affecting the plant are considered. Therefore, the whole MRAC-ILC framework is applied, in which the ILC iteratively compensates for the external disturbances and the MRAC allows the iterations to be carried out by different aircraft.



(a) Performances in tracking speed changes.



(b) Performances in tracking altitude changes.

Figure 6.3: Experiment 1: Output tracking performances of the plant using the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation. The reference model of the MRAC is an A320 aircraft and the plant model a B767 aircraft.

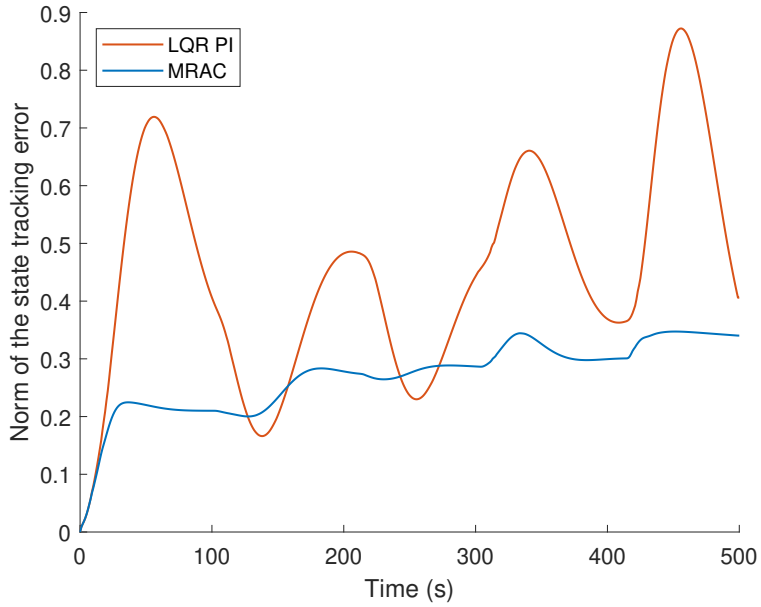


Figure 6.4: Experiment 1: Norms of the state tracking errors observed using the MRAC augmentation of a baseline LQR PI controller and the LQR PI baseline controller without MRAC augmentation. The reference model of the MRAC is an A320 aircraft and the plant model a B767 aircraft.

Experiment 2: MRAC-ILC with transfer learning between two aircraft

The goal of this experiment is to test the effectiveness of the whole MRAC-ILC framework in Figure 6.2 when two different aircraft perform each iteration of the tracking task. The reference model in the MRAC is that of an A320 aircraft with a baseline LQR PI controller. In this experiment, the plant model is assumed to be affected by external disturbances. Specifically, horizontal wind and measurement errors are introduced. Since external disturbances affect the plant, several iterations of the ILC are executed. This amounts to using the whole MRAC-ILC framework in Figure 6.2.

To make the scenario of this experiment more realistic, since successive flights are, in general, operated by different aircraft, the plant model is assumed to change during the iterations of the ILC. Specifically, the plant model is that of an A320 aircraft for the first nine iterations of the ILC. Afterwords, it changes to that of a B767 aircraft for the last eleven iterations of the ILC.

The ultimate goal of this experiment is to prove that the deviations from the desired output trajectory experienced by the plant due to external disturbances can be amended by the ILC over iterations, learning from the deviations observed in previous executions of the tracking task, even if they are performed by two different aircraft.

In this experiment, the desired trajectory to be followed by the plant, $\mathbf{y_d}(t)$, is a feasible trajectory for the A320 aircraft in the absence of external disturbances. The corresponding desired speed and altitude components are the command signals for the MRAC augmentation

of the baseline LQR PI at the first iteration, namely $\mathbf{r}_1(t) = (V_d(t), h_{e,d}(t))$. In the subsequent iterations, the command signals are updated by the ILC.

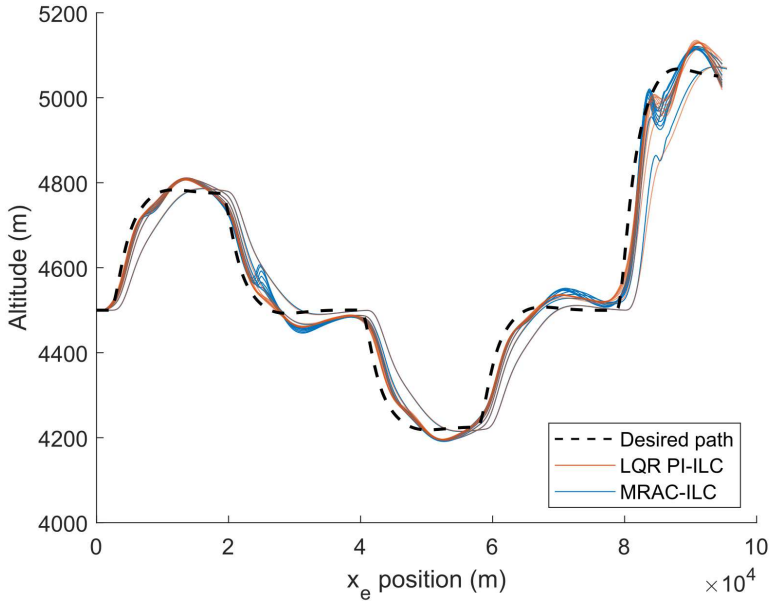


Figure 6.5: Experiment 2: Evolution of the path $x_e - h_e$ of the plant from iteration 1 to 9, before the plant model switch, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation.

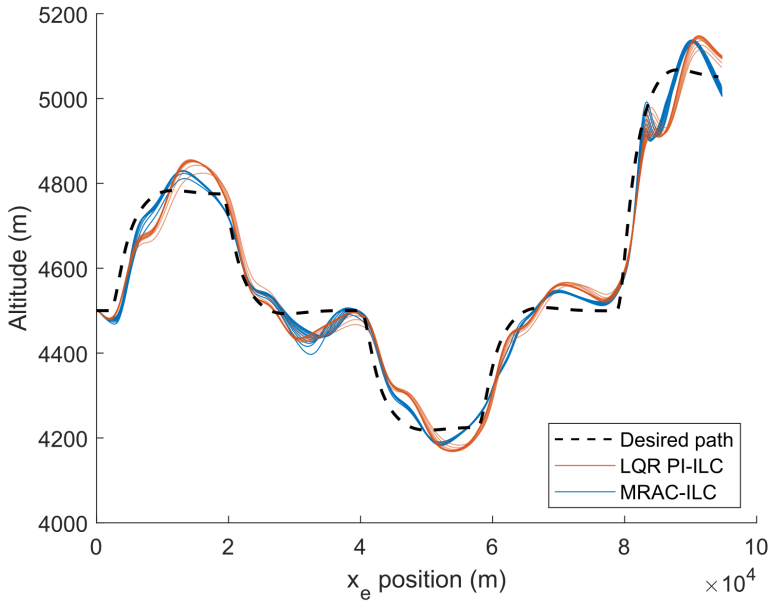


Figure 6.6: Experiment 2: Evolution of the path $x_e - h_e$ of the plant from iteration 10 on, after the plant model switch, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation.

Figures 6.5 and 6.6 show the evolution of the path $x_e - h_e$ over iterations using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation. In Figure 6.5 only the results of the first nine iterations are reported, for which both control schemes show a similar behavior, since there is a concordance between the reference and the plant models. Figure 6.6 shows the results obtained using both schemes from iteration 10 on, when the plant model switches to that of a B767. It can be seen that the path followed by the aircraft differs depending on the control strategy, remaining closer to the desired path when the baseline controller is augmented with the MRAC.

Figures 6.7 and 6.8 show the evolution of the altitude and the speed of the plant over the first nine iterations and the remaining eleven iterations, respectively, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation. As shown in Figure 6.7, the plant exhibits a similar behavior in the first iterations when the plant and the reference models coincide. At the first iteration, the tracking of the speed is quite accurate, but due to the greater importance given to precision in position than precision in speed in the ILC algorithm, the precision in tracking the desired speed is sacrificed in favor of the precision in tracking both the horizontal position and the altitude. It can be observed in Figure 6.7 that only the latter tends to the desired altitude over the iterations, whereas the speed remains below the desired value. It can be observed in Figure 6.8 that from iteration 10, the behavior in speed an altitude tracking differ depending on the control strategy. The flown altitude is closer to the desired one when using the MRAC augmentation of a baseline LQR PI controller than without MRAC augmentation, and the speed is similar in both cases, since precision in the position variables takes precedence over precision in the rest of the variables.

Figure 6.9 shows the evolution of the weighted state error over iterations using the ILC in combination with the MRAC augmentation of the baseline LQR PI controller and the LQR PI baseline controller without MRAC augmentation. Using the MRAC-ILC framework, the initial output tracking error caused by the external disturbances is reduced in a few iterations of the ILC. The same occurs if the ILC is used in combination with the baseline LQR PI controller. This is due to the fact that during the first nine iterations there is a concordance between reference and plant models, which in both cases is that of an A320 aircraft. However, after iteration 10, in which the plant model switches to that of a B767 aircraft, and there is a mismatch between reference model and plant model, using the MRAC-ILC framework, a small increase of the output tracking error is observed after the plant model switch. On the contrary, using the LQR PI-ILC combination, a significantly larger output tracking error is observed after the plant model switch. It is interesting to point out that in both cases the ILC remains effective after the plant model switch, and the learning process continues until its convergence to the lower bound for the achievable weighted state error. The only difference is that the MRAC-ILC framework is faster than the LQR PI-ILC combination, which requires more iterations to converge.

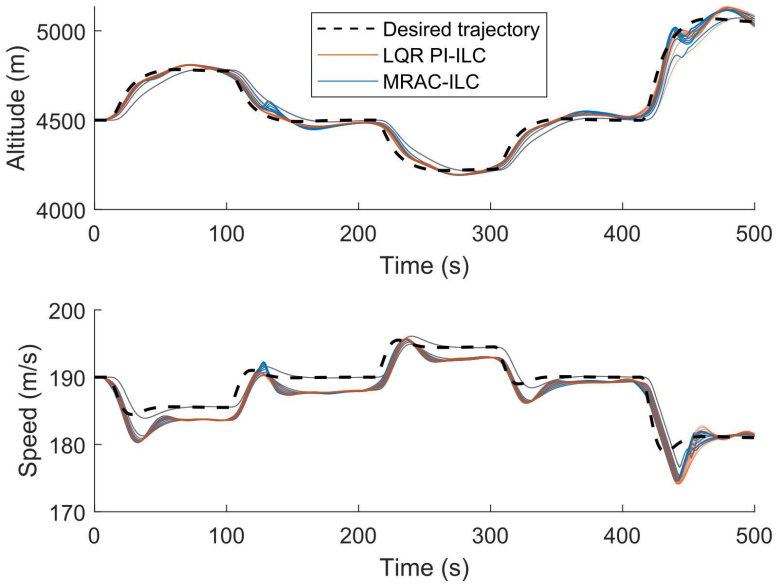


Figure 6.7: Experiment 2: Time evolution of the speed and altitude of the plant from iteration 1 to 9, before the plant model switch, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation.

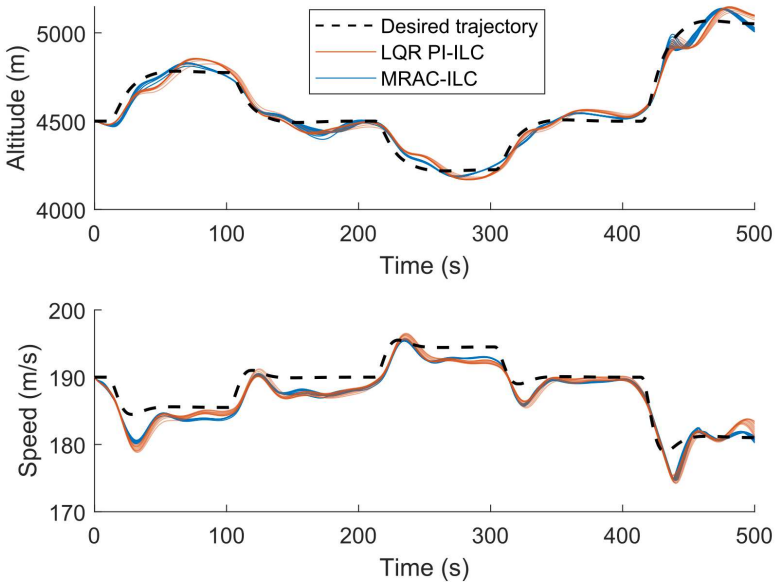


Figure 6.8: Experiment 2: Time evolution of the speed and altitude of the plant from iteration 10 on, after the plant model switch, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation.

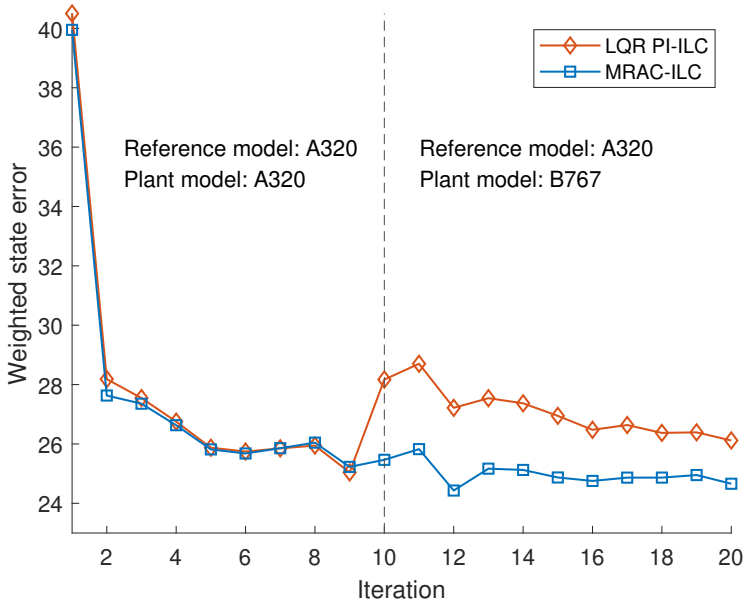


Figure 6.9: Experiment 2: Evolution of the weighted state error over iterations, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI baseline controller without MRAC augmentation. During the first nine iterations there is a concordance between reference and plant models, which in both cases is that of an A320. At iteration 10 the plant model switches to that of a B767.

Experiment 3: MRAC-ILC with transfer learning among multiple aircraft with no mismatch in the first iterations

In this experiment, the whole MRAC-ILC framework in Figure 6.2 is tested when several different aircraft perform each iteration of the tracking task. As in Experiment 2, the reference model in the MRAC is that of an A320 aircraft with a baseline LQR PI controller, and the plant model is assumed to be affected by external disturbances, namely horizontal wind and measurement errors.

To make the scenario of this experiment even more realistic than that of Experiment 2, since successive flights are, in general, operated by different aircraft having different weights, both the plant model and its weight are assumed to change randomly during the iterations of the ILC. Specifically, the plant model is that of an A320 aircraft for the first nine iterations of the ILC. Afterwords, at each iteration it randomly changes to that of an A320, a B767, or an E195 with a random weight within their performance limits. Table 6.1 shows the sequence of aircraft and their weights performing each iteration. It can be observed in this table that from iteration 11 to 12 and from iteration 16 to 17 only the plant weight changes.

The ultimate goal of this experiment is to prove that the deviations from the desired output trajectory experienced by the plant due to external disturbances can be amended by the ILC over iterations, learning from the deviations observed in previous executions of the tracking task, even if they are performed by a random sequence of three different aircraft having random

weights within their performance limits.

The desired trajectory and the command vector in the first iteration are the same as in Experiment 2. Both the evolution of the path $x_e - h_e$, and the time evolution of the speed and altitude of the plant in the first nine iterations, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation are similar to those described in figures 6.5 and 6.7 of Experiment 2, respectively. For the sake of brevity, the corresponding figures are omitted in the description of this experiment.

Iteration	Aircraft	Mass (t)
10	E195	39.23
11	A320	59.60
12	A320	54.16
13	E195	32.90
14	B767	157.59
15	E195	40.63
16	B767	129.54
17	B767	134.37
18	A320	62.30
19	B767	182.82
20	E195	28.95

Table 6.1: Experiment 3: Random sequence of aircraft and the corresponding random weights used at each iteration in the MRAC-ILC framework.

Figure 6.10 shows the evolution of the path $x_e - h_e$ of the plant, from iteration 10 and on, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation. Since the plant model switches at each iteration, the path flown by the aircraft experiences variations. Nevertheless, it can be observed that these variations are smaller and the actual path of the plant remains closer to the desired path when the MRAC augmentation is active.

Figure 6.11 shows the time evolution of the altitude and the speed of the plant, from iteration 10 and on, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation. Greater importance is given in the ILC algorithm to precision in position than precision in speed. Therefore, the precision in tracking the desired speed is sacrificed in favor of the precision in tracking the horizontal position and the altitude. As a consequence, as it can be observed in this figure, at each iteration, the altitude is driven closer to the desired value, whereas the speed remains below the desired values. As in the previous figure, although there is variability in the flown trajectories due to the random change in the plant model at each iteration, the actual altitude is closer to the desired one for the most part of the trajectory when using the MRAC augmentation of a baseline LQR PI controller than without MRAC augmentation, whereas the actual speed is similar in both cases.

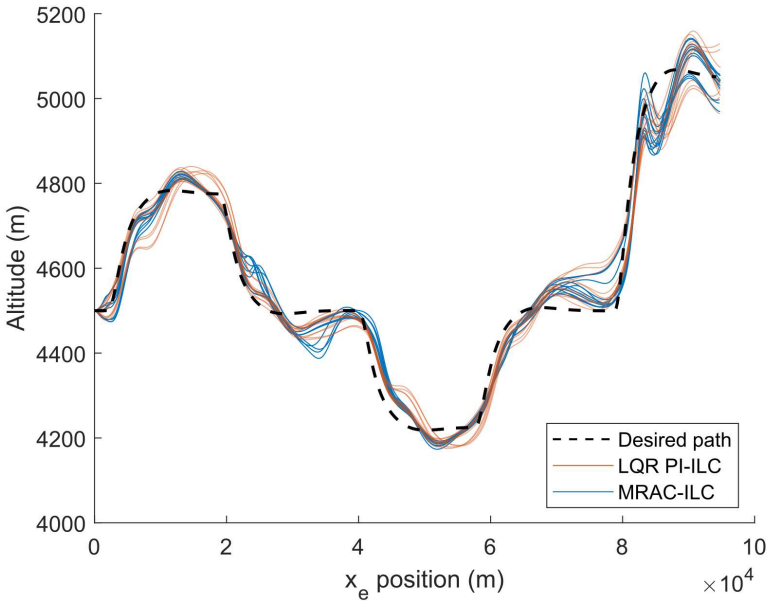


Figure 6.10: Experiment 3: Evolution of the path $x_e - h_e$ of the plant from iteration 10 on, when the plant model randomly switches at each iteration, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation.

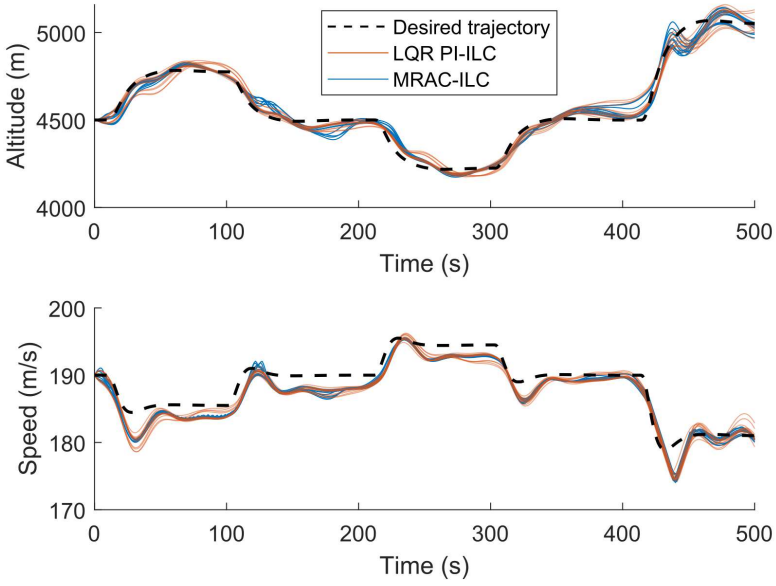


Figure 6.11: Experiment 3: Time evolution of the speed and altitude of the plant from iteration 10 on, when the plant model randomly switches at each iteration, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation.

Finally, Figure 6.12 shows the evolution of the weighted state error over iterations using the ILC in combination with the MRAC augmentation of the baseline LQR PI controller and the LQR PI baseline controller without MRAC augmentation. It can be observed that the initial output tracking error is reduced in a few iterations in a similar trend when using the ILC with and without the MRAC augmentation of the baseline controller, since there is a concordance between the reference model and the plant model, which is that of an A320 aircraft in both cases. After iteration 10, when the plant model randomly switches at each iteration, the weighted state error experiences variations with both control strategies. However, these variations are more pronounced when using the LQR PI-ILC than with the MRAC-ILC, in particular when the B767 aircraft is randomly selected, since the performance characteristics of this aircraft are substantially different from those of both the A320 and the E195 aircraft.

It is important to point out that, despite the variations in the trajectory tracking performance, a trend can be observed in which the weighted state error converges to the lower bound with both control strategies. The advantage of the MRAC-ILC combination over the LQR PI-ILC framework is that the former remains closer to this lower bound throughout all the iterations. The smooth variations in the weighted state error achieved by the MRAC-ILC framework make it the ideal control scheme for real implementation.

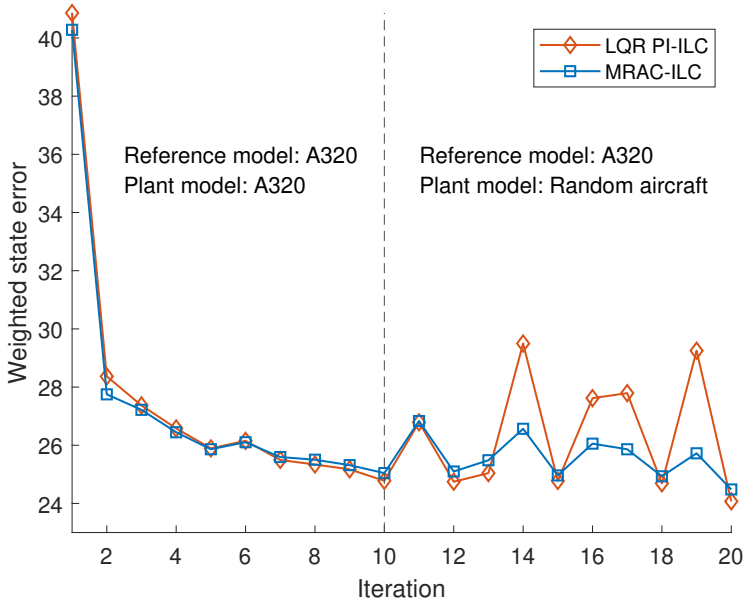


Figure 6.12: Experiment 3: Evolution of the weighted state error over iterations, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI baseline controller without MRAC augmentation. During the first nine iterations there is a concordance between reference and plant models, which in both cases is that of an A320. From iteration 10 the plant model randomly switches at each iteration.

Experiment 4: MRAC-ILC with transfer learning among multiple aircraft with mismatch from the first iteration

In this experiment, the whole MRAC-ILC framework in Figure 6.2 is tested again when several different aircraft perform each iteration of the tracking task. As in Experiment 3, the reference model in the MRAC is that of an A320 aircraft with a baseline LQR PI controller, and the plant model is assumed to be affected by external disturbances, namely horizontal wind and measurement errors. To make the scenario of this experiment fully realistic, since successive flights are, in general, operated by different aircraft having different weights, both the plant model and its weight are assumed to change randomly from the very beginning along the iterations of the ILC. Specifically, at each iteration it randomly changes to that of an A320, a B767, or an E195 with a random weight within their performance limits. Table 6.2 shows the sequence of aircraft and their weights performing each iteration. The desired trajectory and the command vector in the first iteration are the same as in Experiment 3.

The ultimate goal of this experiment is to prove that the deviations from the desired output trajectory experienced by the plant due to external disturbances can be amended by the ILC over iterations, learning from the deviations observed in previous executions of the tracking task, even if they are performed from the very beginning by a random sequence of three different aircraft having random weights within their performance limits.

Iteration	Aircraft	Mass (t)
1	B767	173.29
2	B767	109.39
3	A320	55.66
4	B767	135.64
5	E195	48.94
6	E195	51.69
7	A320	69.14
8	A320	54.70
9	B767	147.97
10	A320	43.60
11	B767	130.47
12	A320	55.79
13	B767	118.17
14	E195	43.88
15	B767	134.38
16	B767	142.70
17	B767	162.24
18	B767	118.82
19	E195	48.87
20	E195	40.72

Table 6.2: Experiment 4: Random sequence of aircraft and the corresponding random weights used at each iteration in the MRAC-ILC framework.

Figure 6.13 shows the evolution of the path $x_e - h_e$ of the plant along all 20 iterations using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation. Since the plant model switches at each iteration, starting from the first one, the path flown by the aircraft experiences variations. As it occurred with Experiment 3, it can be observed that these variations are smaller and the actual path of the plant remains closer to the desired path when the MRAC augmentation is active. In addition, since there is not a series of initial iterations with concordance between reference and plant models as in Experiment 3, before switching the plant model, the path followed by the aircraft at the first iterations is more precise with the MRAC than without it. After the first iteration, the path followed by the aircraft gets closer to the desired path at each iteration with both control strategies. However, this progressive approach towards the desired path is faster with the MRAC-ILC scheme than with the LQR PI-ILC scheme.

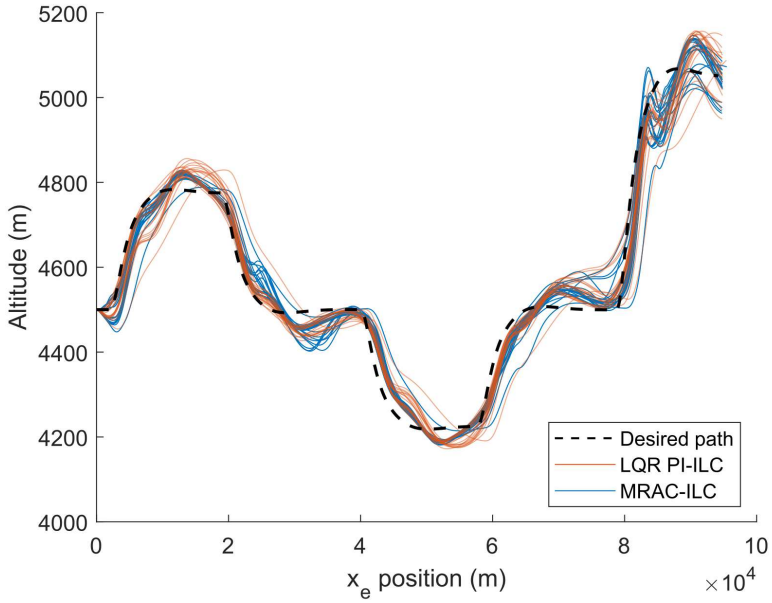


Figure 6.13: Experiment 4: Evolution of the path $x_e - h_e$ of the plant, the model of which randomly switches at each iteration, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation.

Figure 6.14 shows the time evolution of the altitude and the speed of the plant along all 20 iterations using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation. As in the previous experiments, greater importance is given in the ILC algorithm to precision in position than precision in speed. Therefore, the precision in tracking the desired speed is sacrificed in favor of the precision in tracking the horizontal position and the altitude. As a consequence, the speed is driven below the desired values despite being quite accurate in the first iteration, so the altitude can get closer to the desired values at each iteration. As in the previous figure, although there is variability in the flown trajectories due to the random change in the plant

model at each iteration, the actual altitude is closer to the desired one for the most part of the trajectory when using the MRAC augmentation of a baseline LQR PI controller than without MRAC augmentation, whereas the actual speed is similar in both cases, although it experiences more variability with the LQR PI-ILC scheme.

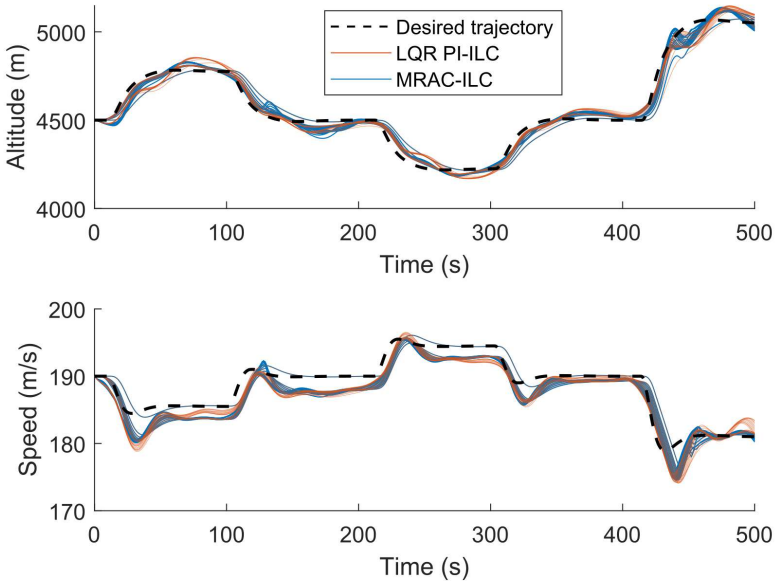


Figure 6.14: Experiment 4: Time evolution of the speed and altitude of the plant, the model of which randomly switches at each iteration, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI controller without MRAC augmentation.

Finally, Figure 6.15 shows the evolution of the weighted state error over iterations using the ILC in combination with the MRAC augmentation of the baseline LQR PI controller and the LQR PI baseline controller without MRAC augmentation. It can be observed that the initial weighted state error is lower when using the ILC with the MRAC augmentation of the baseline controller, since there is no concordance between the reference model and the plant model. In addition, the reduction of this error is faster with the MRAC-ILC scheme than with the LQR PI-ILC. The weighted state error experiences variations with both control strategies. However, these variations are more pronounced when using the LQR PI-ILC than with the MRAC-ILC, in particular when the B767 aircraft is randomly selected, since the performance characteristics of this aircraft are substantially different from those of both the A320 and the E195 aircraft.

Statistical analysis of the performance of the MRAC-ILC. To characterize statistically the tracking performance of the ILC used in combination with the MRAC augmentation of the baseline LQR PI controller, the experiment described above is repeated 30 times and the average and the standard deviation of the weighted state error are computed. The results are compared with those obtained using the ILC in combination with the LQR PI baseline controller without MRAC augmentation.

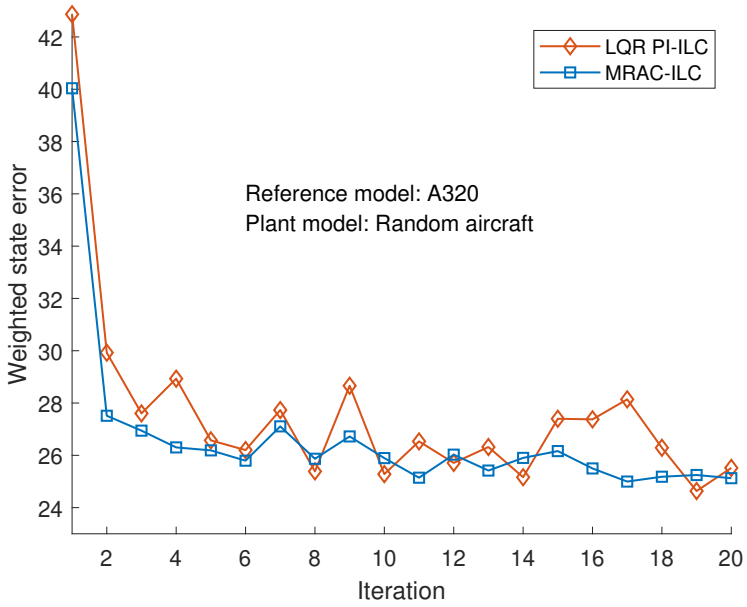


Figure 6.15: Experiment 4: Evolution of the weighted state error over iterations, using the ILC in combination with the MRAC augmentation of a baseline LQR PI controller and the LQR PI baseline controller without MRAC augmentation. The plant model randomly switches at each iteration.

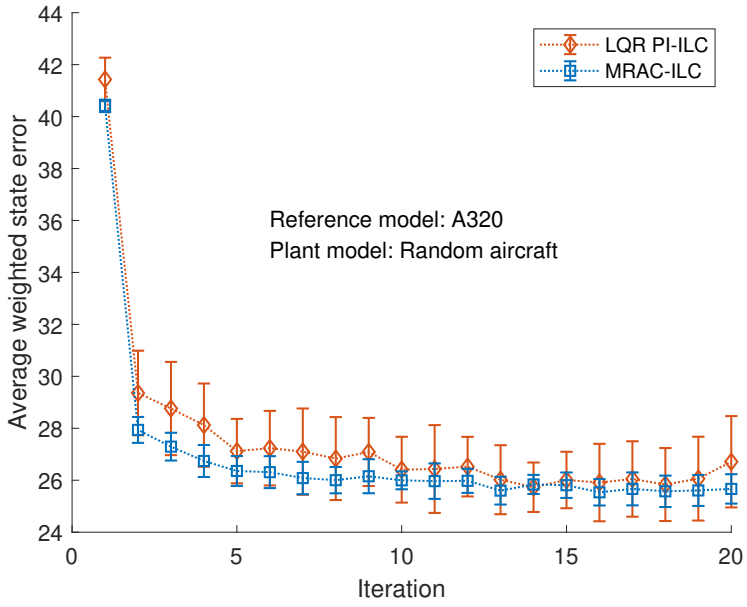


Figure 6.16: Experiment 4: Evolution of the average weighted state error and its standard deviation across 30 sets of iterations, using the LQR PI baseline controller and the MRAC adaptive controller. The plant randomly switches at each iteration.

Figure 6.16 shows the evolution of the average weighted state error and its standard deviation calculated across the 30 sets of 20 iterations with the ILC used in combination with the MRAC augmentation of the baseline LQR PI controller and the LQR PI controller without MRAC augmentation. Since there is not a series of initial iterations with concordance between reference and plant models before switching the plant model, the LQR PI-ILC is slower than the MRAC-ILC, reaching the convergence level several iterations after the MRAC-ILC, in average. The variations with respect to the average error are noticeably higher with the LQR PI-ILC approach. On the contrary, the MRAC-ILC scheme shows much smaller standard deviations consistently along all iterations.

These results demonstrate that the proposed MRAC-ILC scheme successfully reduces the external deviations affecting the flights by learning from previous executions of the same trajectory. Each iteration can be performed by different aircraft, with different performance parameters and weights, and the knowledge acquired can still be transferred to the following aircraft with small variations in the tracking error. As a consequence, predictability of trajectory tracking is improved, making the proposed architecture suitable for real operation, where, in general, different aircraft perform a planned trajectory in the presence of disturbances.

6.4 Conclusion

The results of the experiments show that the proposed MRAC-ILC framework is able to improve trajectory tracking by learning from previous iterations and, at the same time, transfer the knowledge acquired in the previous iterations among dynamically different aircraft, since the MRAC forces them to behave in a similar way, whereas the ILC improves the tracking performance even in the presence of external disturbances.

Comparing the performance through iterations of the aircraft equipped only with the baseline LQR PI controller and the aircraft in which the controller is augmented with the MRAC, it is clear that the latter achieves a more precise tracking of the reference trajectory, and, at the same time, shows smoother variations in the output error, increasing trajectory predictability.

7

Conclusions and Future Work

In this thesis, different control strategies in which ILC is the common factor have been applied to precise aircraft four-dimensional trajectory tracking. An optimization-based iterative learning approach has been employed to improve the precision of the aircraft in following a planned trajectory, taking into account the spatial and temporal deviations experienced by other aircraft in following the same trajectory. Optimality is pursued in both the estimation of the disturbances and in the calculation of the input update, which optimally compensates for the recurring disturbances. The learning behavior can be adapted depending on the flight phase and precision requirements by assigning different weights to the variables and segments of the trajectory. This method is especially suitable for aircraft trajectory tracking in departure and arrival procedures of busy airports because, due to the short time separation between flights, weather conditions, and therefore, disturbances can be assumed to be similar between consecutive flights, and operational constraints, which include waypoints and other limitations due to the procedures, can be straightforwardly taken into account. The simulated environment in which the experiments have been carried out includes a flight simulator of a commercial aircraft flying in the vertical plane, and weather disturbances such as wind and non-standard atmosphere conditions.

Both direct and indirect ILC approaches have been tested, proving to be very effective in compensating for recurrent disturbances affecting the flights, caused mainly by weather and model errors, achieving precise trajectory tracking in few iterations of continuous climb and descent operations performed by the same aircraft model. The direct ILC approach updates, at each iteration, the control input to be applied by the following aircraft, whereas the indirect ILC approach generates a new reference trajectory rather than a control input, which is fed to the aircraft's underlying feedback controller, being thus nonintrusive with respect to the aircraft's existing trajectory tracking controller.

Estimation and prediction are key for precision and fast convergence of the ILC, since the update of the inputs given to the aircraft is based on the predicted perturbations and the estimated model errors that cause deviations from the planned trajectory. In this thesis, two strategies have been implemented, and their performances have been compared, as the estimation and prediction step of the ILC method: the Kalman filter, and the GPR. In order to reduce the high computational effort required by the GPR, a recursive approach has been used, which performs the regression on a set of points that are updated at each iteration to dismiss old data in favor of the most recent measurements, simultaneously learning the hyperparameters.

To extend the ILC approach to a more realistic scenario in which consecutive flights are carried out by different aircraft, an MRAC augmentation of the underlying baseline controller has been applied to the simulated aircraft, forcing the system to behave close to a reference model despite the presence of disturbances and unmodeled dynamics. Therefore, repeatability is achieved by the MRAC, and the ILC can act as a high-level controller to improve the tracking performance of the aircraft. This behavior enables dynamically different systems to share the learned reference trajectories, making the MRAC-ILC suitable as a multi-aircraft transfer learning framework.

7.1 Conclusions

According to the obtained results, the following main conclusions can be drawn:

(i) ILC approaches successfully compensate for repetitive disturbances affecting identical aircraft following the same planned trajectory, improving tracking precision. In particular, the IILC does not interfere with the aircraft's existing trajectory tracking controller, and can be implemented without changing the avionics of the aircraft.

(ii) With no prior knowledge of the behavior of the disturbances, the experiments show that the recursive GPR provides much better estimations and predictions in the first iterations compared to the Kalman filter, therefore achieving faster and more precise tracking of the aircraft trajectory. The results obtained by both methods are comparable only when the Kalman filter is correctly tuned, which means collecting some prior knowledge about the dynamics of the disturbances. Additionally, the recursive GPR allows for estimation and prediction of nonlinear dynamics.

(iii) The MRAC forces the aircraft to behave close to a given reference model, making its performance predictable even in the presence of system uncertainties and changing dynamics. In addition, the MRAC augmentation of a baseline controller enables the use of the adaptive control enhancing the aircraft's underlying feedback controller performance without replacing it.

(iv) The MRAC-ILC architecture serves as a multi-aircraft transfer learning framework that enables to transfer the learned reference trajectories among dynamically different aircraft.

This control strategy decouples the task of rejecting disturbances and handling changing dynamics, executed by the MRAC, from the task of improving trajectory tracking, accomplished by the ILC, achieving high-accuracy tracking performance even if consecutive iterations are executed by different aircraft and in the presence of unknown dynamics.

Overall, it can be concluded that the ILC techniques proposed in this dissertation are able to improve the tracking performance of commercial aircraft in following their planned trajectories, taking into account the deviations suffered by previous flights. Trajectory predictability is thus enhanced, facilitating the implementation of TBOs in busy airports and reducing the number of alterations when following the planned trajectories, which entails an increase of the efficiency of the ATM system, resulting in a reduction of costs and emissions.

7.2 Future work

The MRAC-ILC architecture for commercial aircraft trajectory tracking enables the knowledge about tracking errors and disturbances to be transferred among dynamically different aircraft under the assumption that they follow the same desired trajectory. However in operational scenarios, different aircraft in general follow different trajectories. Therefore the first and foremost extension of this architecture would be devising a strategy that allows this knowledge to be transferred even when it is acquired by different aircraft following different trajectories. Preliminary research indicates that this problem could be solved by including in the process learning the mapping from desired trajectories to the inputs that make the aircraft track these trajectories in the relevant environment. In this way, the error in following new desired trajectories is reduced [82].

Another possible extension of the MRAC-ILC architecture for commercial aircraft trajectory tracking could be removing the assumption of symmetric flight in the vertical plane and consider flights in the three dimensional space.

In this thesis, the MRAC-ILC architecture has been tested in trajectory tracking tasks in which the aircraft model is linearized at a single operating point. Therefore, another possible extension could be to incorporate gain scheduling, in which a set of parameters for the baseline and adaptive controllers are designed for several model linearizations associated with several operating points of the aircraft, which reflect several flight phases and conditions, such as different altitudes, speeds, and weights.

In this thesis, the A320 aircraft has been chosen as the reference model in the MRAC-ILC scheme. However, the effect of the choice of the reference model on the performance of the MRAC-ILC scheme has not been studied. Therefore, the problem of establishing what reference model is most suitable, given the trajectory to be tracked and the aircraft involved in the trajectory tracking, will be subject of future research.

In this thesis, the MRAC has been used in combination with the ILC to transfer the learned reference trajectories among different aircraft. However, other adaptive controllers could be

employed, such as the $\mathcal{L}_1$ adaptive controller, which has already been tested in combination with the ILC in quadrotor trajectory tracking giving good results [79]. The study of this and other adaptive controllers will be subject of future research.

Appendices

A

Aircraft Performance Parameters

This appendix aims to summarize the BADA performance parameters of the aircraft simulated in this thesis, i.e. an Airbus A320-231 (A320), a Boeing 767-300ER (B767), and an Embraer ERJ 190-200 IGW (E195). For more details and further information on performance models, performance limitations or airline procedure models, the reader is referred to BADA Revision 3.14 [7] and posterior versions.

Aircraft		A320	B767	E195
Mass				
Reference mass	m_{ref} [t]	64	154.59	44
Minimum mass	$m_{\min}$ [t]	39	90.011	28.667
Maximum mass	$m_{\max}$ [t]	77	186.88	52.29
Maximum payload mass	m_{pyld} [t]	21.5	43.799	13.933
Flight envelope				
Maximum operating speed (CAS)	V_{MO} [kn]	350	360	320
Maximum operating Mach number	M_{MO}	0.82	0.86	0.82
Maximum operating altitude	h_{MO} [ft]	41000	43100	41000
Maximum altitude at MTOW and ISA	$h_{\max}$ [ft]	33295	36502	35815
Weight gradient on maximum altitude	G_{w} [ft/kg]	0.4325	0.11823	0.40345
Temperature gradient on maximum altitude	G_{t} [ft/K]	-313.6	-41.12	-185.5
Aerodynamics				
Reference wing surface area	S [m^2]	122.6	283.35	92.53
Parasitic drag coefficient:				
▪ Cruise	$C_{\text{D0,CR}}$	0.026659	0.021112	0.026806
▪ Initial climb	$C_{\text{D0,IC}}$	0.023	0.01612	0
▪ Take off	$C_{\text{D0,TO}}$	0.033	0.027554	0
▪ Approach	$C_{\text{D0,AP}}$	0.038	0.030382	0

Aircraft		A320	B767	E195
▪ Landing	$C_{D0,LD}$	0.096	0.093121	0
▪ Landing gear	$C_{D0,\Delta\ LDG}$	0.038	0.017	0
Induced drag coefficient:				
▪ Cruise	$C_{Di,CR}$	0.038726	0.042118	0.042012
▪ Initial climb	$C_{Di,IC}$	0.044	0.053848	0
▪ Take off	$C_{Di,TO}$	0.041	0.047458	0
▪ Approach	$C_{Di,AP}$	0.0419	0.048029	0
▪ Landing	$C_{Di,LD}$	0.0371	0.039805	0
Stall speed (CAS):				
▪ Cruise	$V_{stall,CR}$ [kn]	140.5	167	139
▪ Initial climb	$V_{stall,IC}$ [kn]	118	138	139
▪ Take off	$V_{stall,TO}$ [kn]	112.1	124	112
▪ Approach	$V_{stall,AP}$ [kn]	105.1	122	98
▪ Landing	$V_{stall,LD}$ [kn]	101.3	118	95
Engine thrust				
Maximum climb thrust coefficients:	$C_{Tc,1}$ [N]	142310	322410	104450
	$C_{Tc,2}$ [ft]	51680	56718	50404
	$C_{Tc,3}$ [1/ft^2]	0.56809E-10	0.13638E-10	0.58971E-10
Thrust temperature coefficients:	$C_{Tc,4}$ [K]	10.138	9.5535	9.8814
	$C_{Tc,5}$ [1/K]	0.8871E-2	0.37598E-2	0.91048E-2
Low altitude descent thrust coefficient	$C_{Tdes,low}$	0.10847	0.055988	0.91048E-2
High altitude descent thrust coefficient	$C_{Tdes,high}$	0.13603	0.064359	0.1233
Transition altitude for calculation of descent thrust	$H_{p,des}$ [ft]	29831	26418	31368
Approach thrust coefficient	$C_{Tdes,app}$	0.15749	0.12475	0.083934
Landing thrust coefficient	$C_{Tdes,ld}$	0.39566	0.32981	0.13921
Reference descent speed (CAS)	$V_{des,ref}$ [kn]	310	310	300
Reference descent Mach number	$M_{des,ref}$	0.78	0.80	0.7
Fuel flow				
Thrust specific fuel consumption coefficients:	$C_{f,1}$ [kg/(min kN)]	0.75882	0.7422	0.68322
	$C_{f,2}$ [kn]	2938.5	2060.5	972.76
Descent fuel flow coefficients:	$C_{f,3}$ [kg min^{-1}]	8.9418	15.902	11.383
	$C_{f,4}$ [ft]	93865	145380	90044
Cruise fuel flow correction coefficient	C_{fcr}	0.96358	0.90048	1.0013

Table A.1: BADA Revision 3.14 performance parameters of Airbus A320-231 (A320), Boeing 767-300ER (B767), and Embraer ERJ 190-200 IGW (E195) aircraft.

References

Bibliography

[1] A. Tewari, *Advanced Control of Aircraft, Spacecraft and Rockets*. Wiley, 2011.

[2] International Civil Aviation Organization, "Global air navigation plan," ICAO, Tech. Rep. 9750-AN/963, 2016.

[3] A. Buelta, A. Olivares, and E. Staffetti, "Iterative learning control for precise aircraft trajectory tracking in continuous climb operations," in *Proceedings of the Thirteenth USA/Europe Air Traffic Management Research and Development Seminar*, Vienna, Austria, June 2019.

[4] ——, "Iterative learning control for precise aircraft trajectory tracking in continuous descent approaches," in *Proceedings of the 8th European Conference for Aeronautics and Aerospace Sciences*, Madrid, Spain, July 2019.

[5] ——, "Iterative learning control for precise aircraft trajectory tracking in continuous climb and descent operations," *IEEE Transactions on Intelligent Transportation Systems*, no. In press, 2021.

[6] A. Buelta, A. Olivares, E. Staffetti, W. Aftab, and L. Mihaylova, "A Gaussian process iterative learning control for aircraft trajectory tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 6, pp. 3962–3973, 2021.

[7] V. Mouillet, "User manual for the Base of Aircraft Data (BADA) Revision 3.14," EUROCONTROL Experimental Centre, Brétigny, France, Tech. Rep., 2017.

[8] U.S. Department of Defense, "Global climatic data for developing military products," U.S. Federal Government, Tech. Rep. U.S. Military Specification MIL-STD-210C, 1987.

[9] D. G. Hull, *Fundamentals of Airplane Flight Mechanics*. Springer-Verlag, 2007.

[10] B. Huang, B. Lu, and Q. Li, "A proportional–integral-based robust state-feedback control method for linear parameter-varying systems and its application to aircraft," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 233, no. 12, pp. 4663–4675, 2019.

[11] M. Hadi and N. Abdul, "Reducing the effect of the atmospheric disturbance on longitudinal flight control system usage PID controller," *International Journal of Computer Applications*, vol. 127, no. 13, pp. 27–31, 2015.

[12] M. Peters and M. A. Konyak, "The engineering analysis and design of the aircraft dynamics model for the FAA Target Generation Facility," Federal Aviation Administration, Tech. Rep. 99162-01, 2012.

[13] D. P. Drob, J. T. Emmert, J. W. Meriwether, J. J. Makela, E. Doornbos, M. Conde, G. Hernandez, J. Noto, K. A. Zawdie, S. E. McDonald, J. D. Huba, and J. H. Klenzing, "An update to the Horizontal Wind Model (HWM): The quiet time thermosphere," *Earth and Space Science*, vol. 2, no. 7, pp. 301–319, 2015.

[14] U.S. Department of Defense, "The flying qualities of piloted airplanes," U.S. Federal Government, Tech. Rep. U.S. Military Specification MIL-F-8785C, 1980.

[15] M. I. Ross, *A Primer on Pontryagin's Principle in Optimal Control*. Collegiate Publishers, 2015.

[16] Q. Gong, F. Fahroo, and I. M. Ross, "Spectral algorithm for pseudospectral methods in optimal control," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 3, pp. 460–471, 2008.

[17] International Civil Aviation Organization, "Continuous Climb Operations (CCO) manual," ICAO, Tech. Rep. Doc. 9993 AN/495, 2012.

[18] ——, "Continuous Descent Operations (CDO) manual," ICAO, Tech. Rep. Doc 9931 AN/476, 2010.

[19] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 123–140, 1984.

[20] K. L. Moore, M. Dahleh, and S. P. Bhattacharyya, "Iterative learning control: A survey and new results," *Journal of Robotic Systems*, vol. 9, no. 5, pp. 563–594, 1992.

[21] D. A. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, pp. 2039–2114, 2006.

[22] J.-X. Xu and Y. Tan, *Linear and Nonlinear Iterative Learning Control*. Springer, 2003.

[23] J.-X. Xu, S. K. Panda, and T. H. Lee, *Real-time Iterative Learning Control: Design and Applications*. Springer, 2009.

[24] A. P. Schoellig, F. L. Mueller, and R. D'Andrea, "Optimization-based iterative learning for precise quadrocopter trajectory tracking," *Autonomous Robots*, vol. 33, no. 1-2, pp. 103–127, 08 2012.

[25] O. Purwin and R. D'Andrea, "Performing and extending aggressive maneuvers using iterative learning control," *Robotics and Autonomous Systems*, vol. 59, no. 1, pp. 1–11, 2011.

[26] X. Liang, M. Zheng, and F. Zhang, "A scalable model-based learning algorithm with application to UAVs," *IEEE Control Systems Letters*, vol. 2, no. 4, pp. 839–844, 2018.

[27] Y. Wang, F. Gao, and F. J. Doyle, "Survey on iterative learning control, repetitive control, and run-to-run control," *Journal of Process Control*, vol. 19, no. 10, pp. 1589–1600, 2009.

[28] R. Tousain, E. van der Meche, and O. Bosgra, "Design strategy for iterative learning control based on optimal control," in *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228)*, Orlando, FL, USA, December 2001.

[29] B. Bamieh, J. B. Pearson, B. A. Francis, and A. Tannenbaum, "A lifting technique for linear periodic systems with applications to sampled-data control," *Systems & Control Letters*, vol. 17, no. 2, pp. 79–88, 1991.

[30] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, 2002.

[31] D. Simon, "Kalman filtering with state constraints: A survey of linear and nonlinear algorithms," *IET Control Theory Applications*, vol. 4, no. 8, pp. 1303–1318, 2010.

[32] C. E. Hutchinson, "The Kalman filter applied to aerospace and electronic systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-20, no. 4, pp. 500–504, 1984.

[33] J. Dunik, O. Straka, M. Simandl, and E. Blasch, "Random-point-based filters: Analysis and comparison in target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 2, pp. 1403–1421, 2015.

[34] H. Bonyan Khamseh, S. Ghorbani, and F. Janabi-Sharifi, "Unscented Kalman filter state estimation for manipulating unmanned aerial vehicles," *Aerospace Science and Technology*, vol. 92, pp. 446–463, 2019.

[35] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[36] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models," *Autonomous Robots*, vol. 27, pp. 75–90, 2008.

[37] T. Van Gestel, J. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle, "Bayesian framework for least-squares support vector machine classifiers, Gaussian processes, and kernel Fisher discriminant analysis," *Neural Computation*, vol. 14, pp. 1115–1147, 2002.

[38] F. Perez-Cruz, S. Van Vaerenbergh, J. Murillo-Fuentes, M. Lázaro-Gredilla, and I. Santamaria, "Gaussian processes for nonlinear signal processing: An overview of recent advances," *IEEE Signal Processing Magazine*, vol. 30, pp. 40–50, 2013.

[39] G. Pillonetto, "A new kernel-based approach to hybrid system identification," *Automatica*, vol. 70, pp. 21–31, 2016.

[40] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith, "Dynamic systems identification with Gaussian processes," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 11, no. 4, pp. 411–424, 2005.

[41] N. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse Gaussian process methods: The informative vector machine," in *Advances in Neural Information Processing Systems 15. Proceedings of the 2002 Conference*, Vancouver, BC, Canada, December 2002.

[42] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems 18. Proceedings of the 2005 Conference*, Vancouver, BC, Canada, December 2005.

[43] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems 13. Proceedings of the 2000 Conference*, Denver, CO, USA, December 2000.

[44] S. Brahim-Belhouari and A. Bermak, "Gaussian process for nonstationary time series prediction," *Computational Statistics and Data Analysis*, vol. 47, pp. 705–712, 2004.

[45] M. F. Huber, "Recursive Gaussian process: On-line regression and learning," *Pattern Recognition Letters*, vol. 45, pp. 85–91, 2014.

[46] S. J. Julier, "The scaled unscented transformation," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, Anchorage, AK, USA, May 2002.

[47] R. Kandepu, L. Imsland, and B. Foss, "Constrained state estimation using the unscented Kalman filter," in *Proceedings of the 2008 Mediterranean Conference on Control and Automation*, Ajaccio, France, July 2008.

[48] N. Nguyen, *Model Reference Adaptive Control - a Primer*. Springer, 2018.

[49] K. J. Aström and B. Wittenmark, *Adaptive Control*. Dover Publications, 2008.

[50] J. Chen, J. Wang, and W. Wang, "Robust adaptive control with control structure modification for aircraft," *Journal of Aerospace Engineering*, vol. 32, no. 3, pp. 1–11, 2019, Article number 04019019.

[51] H. Yoon and P. Tsiotras, "Adaptive spacecraft attitude tracking control with actuator uncertainties," *The Journal of the Astronautical Sciences*, vol. 56, no. 2, pp. 251–268, 2008.

[52] J. Lee, D.-E. Kang, and C. Park, "Geometric robust adaptive control for satellite attitude tracking with reaction wheels," *Acta Astronautica*, vol. 179, pp. 238–252, 2021.

[53] L. Zhou, S. Xu, H. Jin, and H. Jian, "A hybrid robust adaptive control for a quadrotor UAV via mass observer and robust controller," *Advances in Mechanical Engineering*, vol. 13, no. 3, pp. 1–11, 2021.

[54] A. N. Ouda, "A robust adaptive control approach to missile autopilot design," *International Journal of Dynamics and Control*, vol. 6, pp. 1239–1271, 2018.

[55] O. Yechiel, G. Israeli, and H. Guterman, "Direct adaptive control using a neuro-evolutionary algorithm for vehicle speed control," in *in Proceedings of the 2018 IEEE International Conference on the Science of Electrical Engineering*, Eilat, Israel, December 2018.

[56] T. Hsia, "Adaptive control of robot manipulators - A review," in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3, San Francisco, CA, USA, April 1986.

[57] J. Li, Y. Wang, Z. Liu, X. Jing, and C. Hu, "A new recursive composite adaptive controller for robot manipulators," *Space: Science and Technology*, vol. 2021, pp. 1–7, 2021, Article ID 980142.

[58] N. Nguyen, M. Bright, and D. Culley, "Feedback adaptive flow control of air injection in compressor," in *In proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, CA, USA, August 2005.

[59] A. J. Calise and T. Yucelen, "Adaptive loop transfer recovery," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 3, pp. 807–815, 2012.

[60] C. Cao and N. Hovakimyan, "Design and analysis of a novel $\mathcal{L}_1$ adaptive control architecture with guaranteed transient performance," *IEEE Transactions on Automatic Control*, vol. 53, no. 2, pp. 586–591, 2008.

[61] H. Whitaker, J. Yamron, and A. Kezer, "Design of model-reference adaptive control systems for aircraft," Massachusetts Institute of Technology. Instrumentation Laboratory, Tech. Rep. R-164, 1958.

[62] P. Parks, "Lyapunov redesign of model reference adaptive control systems," *IEEE Transactions on Automatic Control*, vol. 11, no. 3, pp. 362–367, 1966.

[63] E. Lavretsky and K. Wise, *Robust and Adaptive Control With Aerospace Applications*. Springer, 2013.

[64] T. Yucelen and A. J. Calise, "Derivative-free model reference adaptive control," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 4, pp. 933–950, 2011.

[65] N. Nguyen, "Least-squares model-reference adaptive control with Chebyshev orthogonal polynomial approximation," *Journal of Aerospace Information Systems*, vol. 10, no. 6, pp. 268–286, 2013.

[66] N. Nguyen, C. Hanson, J. Burken, and J. Schaefer, "Normalized optimal control modification and flight experiments on NASA F/A-18 aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 4, pp. 1061–1075, 2017.

[67] S. S.-M. Swei and N. T. Nguyen, "Adaptive disturbance torque estimation for orbiting spacecraft using recursive least-squares methods," *Journal of Aerospace Information Systems*, vol. 14, no. 2, pp. 92–102, 2017.

[68] J. Chen, J. Wang, and W. Wang, "Model reference adaptive control for a class of aircraft with actuator saturation," in *in Proceedings of the 37th Chinese Control Conference*, Wuhan, China, July 2018.

[69] W. S. Levine, *The Control Systems Handbook. Control System Advanced Methods*. CRC Press, 2011.

[70] K. S. Narendra and B. B. Peterson, "Bounded error adaptive control," in *in Proceedings of the 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, Albuquerque, NM, USA, December 1980.

[71] J.-B. Pomet and L. Praly, "Adaptive nonlinear regulation: estimation from the Lyapunov equation," *IEEE Transactions on Automatic Control*, vol. 37, no. 6, pp. 729–740, 1992.

[72] P. Ioannou and B. Fidan, *Adaptive Control Tutorial*. SIAM, 2006.

[73] N. Hovakimyan and C. Cao, $\mathcal{L}_1$ *Adaptive Control Theory: Guaranteed Robustness with Fast Adaptation*. SIAM, 2010.

[74] A. Tayebi, "Model reference adaptive iterative learning control for linear systems," *International Journal of Adaptive Control and Signal Processing*, vol. 20, no. 9, pp. 475–489, 2006.

[75] S. Jingzhuo and W. Huang, "Model reference adaptive iterative learning speed control for ultrasonic motor," *IEEE Access*, vol. 8, pp. 181815–181824, 2020.

[76] B. Yuksek and G. Inalhan, "Reinforcement learning based closed-loop reference model adaptive flight control system design," *International Journal of Adaptive Control and Signal Processing*, vol. 35, no. 3, pp. 420–440, 2021.

[77] B. Altın and K. Barton, "Robust iterative learning for high precision motion control through $\mathcal{L}_1$ adaptive feedback," *Mechatronics*, vol. 24, no. 6, pp. 549–561, 2014.

[78] K. Pereida, R. R. P. R. Duivenvoorden, and A. P. Schoellig, "High-precision trajectory tracking in changing environments through $\mathcal{L}_1$ adaptive feedback and iterative learning," in *in Proceedings of the IEEE International Conference on Robotics and Automation*, Marina Bay Sands, Singapore, May 2017.

[79] K. Pereida, D. Kooijman, R. R. P. R. Duivenvoorden, and A. P. Schoellig, "Transfer learning for high-precision trajectory tracking through $\mathcal{L}_1$ adaptive feedback and iterative learning," *International Journal of Adaptive Control and Signal Processing*, vol. 33, no. 2, pp. 388–409, 2019.

[80] G. Chowdhary, T. Wu, M. Cutler, and J. P. How, "Rapid transfer of controllers between UAVs using learning-based adaptive control," in *in Proceedings of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013.

[81] Z. Chen, X. Liang, and M. Zheng, "Knowledge transfer between different UAVs for trajectory tracking," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4939–4946,

2020.

[82] K. Pereida, M. K. Helwa, and A. P. Schoellig, "Data-efficient multirobot, multitask transfer learning for trajectory tracking," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1260–1267, 2018.