ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

ΠΡΟΓΡΑΜΜΑ ΔΙΔΑΚΤΟΡΙΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Ανάπτυξη και βαθμονόμηση μεθόδων αυτόματου ελέγχου μη γραμμικών συστημάτων με χρήση τεχνικών υπολογιστικής νοημοσύνης με έμφαση στον έλεγχο μη επανδρωμένων εναέριων οχημάτων

Αριστοτέλης Σ. Καπνόπουλος

ΑΙΓΑΛΕΩ
Οκτώβριος 2023

Η υλοποίηση της διδακτορικής διατριβής συγχρηματοδοτήθηκε από την Ελλάδα και την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) μέσω του Επιχειρησιακού Προγράμματος «Ανάπτυξη Ανθρώπινου Δυναμικού, Εκπαίδευση και Δια Βίου Μάθηση», 2014-2020, στο πλαίσιο της Πράξης «Ενίσχυση του ανθρώπινου δυναμικού μέσω της υλοποίησης διδακτορικής έρευνας Υποδράση 2: Πρόγραμμα χορήγησης υποτροφιών ΙΚΥ σε υποψηφίους διδάκτορες των ΑΕΙ της Ελλάδας.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

Επιχειρησιακό Πρόγραμμα
Ανάπτυξη Ανθρώπινου Δυναμικού,
Εκπαίδευση και Διά Βίου Μάθηση
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

UNIVERSITY OF WEST ATTICA

SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

PROGRAM OF DOCTORAL STUDIES

PhD THESIS

Development and tuning of automatic control methods for nonlinear systems using computational intelligence techniques with emphasis on the control of unmanned aerial vehicles

Aristotelis Kapnopoulos

ATHENS-EGALEO
October 2023

The implementation of the doctoral thesis was co-financed by Greece and the European Union (European Social Fund-ESF) through the Operational Programme «Human Resources Development, Education and Lifelong Learning» in the context of the Act "Enhancing Human Resources Research Potential by undertaking a Doctoral Research" Sub-action 2: IKY Scholarship Programme for PhD candidates in the Greek Universities.

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Ανάπτυξη και βαθμονόμηση μεθόδων αυτόματου ελέγχου μη γραμμικών συστημάτων με χρήση τεχνικών υπολογιστικής νοημοσύνης με έμφαση στον έλεγχο μη επανδρωμένων εναέριων οχημάτων

Αριστοτέλης Σ. Καπνόπουλος

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Αλέξανδρος Αλεξανδρίδης, Καθηγητής ΠαΔΑ

ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ:
Αλέξανδρος Αλεξανδρίδης, Καθηγητής, ΠαΔΑ
Διονύσιος Κανδρής, Καθηγητής, ΠαΔΑ
Ηλίας Ζώης, Αναπληρωτής Καθηγητής, ΠαΔΑ

ΕΠΤΑΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

[illegible] Digitally signed by Alexandros Alexandridis Date: 2023.11.24 11:40:17 +02'00'

Αλέξανδρος Αλεξανδρίδης,
Καθηγητής, ΠαΔΑ

XENOFON DIONYSIOS KANDRIS Digitally signed by XENOFON DIONYSIOS KANDRIS Date: 2023.11.23 23:31:54 +02'00'

Διονύσιος Κανδρής,
Καθηγητής, ΠαΔΑ

Ilias Zois Digitally signed by Ilias Zois Date: 2023.11.23 15:36:32 +02'00'

Ηλίας Ζώης,
Αναπληρωτής Καθηγητής, ΠαΔΑ

Pantelis Malatestas Digitally signed by Pantelis Malatestas Date: 2023.11.23 16:36:50 +02'00'

Παντελής Μαλατέστας,
Καθηγητής, ΠαΔΑ

[illegible] Digitally signed by Grigorios Koulouras DN: c=GR, l=Athens, o=University of West Attica, ou=Class A - Private Key created and stored in hardware CSP, sn=Koulouras, givenName=Grigorios, serialNumber=6090016933, cn=Grigorios Koulouras, email=gregkoul@uniwa.gr Date: 2023.11.23 19:40:29 +02'00'

Γρηγόριος Κουλούρας,
Αναπληρωτής Καθηγητής, ΠαΔΑ

Dimitrios Pyromalis Digitally signed by Dimitrios Pyromalis Date: 2023.11.24 08:02:54 +02'00'

Δημήτριος Πυρομάλης,
Αναπληρωτής Καθηγητής, ΠαΔΑ

Χαράλαμπος Σαρίμβεης Digitally signed by Χαράλαμπος Σαρίμβεης Date: 2023.11.23 15:10:49 +02'00'

Χαράλαμπος Σαρίμβεης,
Καθηγητής, ΕΜΠ

Ημερομηνία εξέτασης 20/11/2023

PhD THESIS

Development and tuning of automatic control methods for nonlinear systems using computational intelligence techniques with emphasis on the control of unmanned aerial vehicles

Aristotelis Kapnopoulos

SUPERVISOR: Alex Alexandridis, Professor, University of West Attica

Examination Date 20/11/2023

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΔΑΚΤΟΡΙΚΗΣ ΔΙΑΤΡΙΒΗΣ

Ο κάτωθι υπογεγραμμένος Αριστοτέλης Καπνόπουλος του Σάββα, υποψήφιος διδάκτορας του Τμήματος Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:
«Είμαι συγγραφέας και δικαιούχος των πνευματικών δικαιωμάτων επί της διατριβής και δεν προσβάλω τα πνευματικά δικαιώματα τρίτων. Για τη συγγραφή της διδακτορικής μου διατριβής δεν χρησιμοποίησα ολόκληρο ή μέρος έργου άλλου δημιουργού ή τις ιδέες και αντιλήψεις άλλου δημιουργού χωρίς να γίνεται αναφορά στην πηγή προέλευσης (βιβλίο, άρθρο από εφημερίδα ή περιοδικό, ιστοσελίδα κ.λπ.). Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών
Αριστοτέλης Καπνόπουλος

[illegible]

ΠΕΡΙΛΗΨΗ

Στόχος της παρούσας διατριβής είναι η δημιουργία και η βέλτιστη βαθμονόμηση τεχνικών αυτόματου ελέγχου για μη γραμμικά συστήματα, με έμφαση στον έλεγχο μη επανδρωμένων εναέριων οχημάτων, χρησιμοποιώντας μεθόδους υπολογιστικής νοημοσύνης. Οι κύριες μέθοδοι υπολογιστικής νοημοσύνης που χρησιμοποιήθηκαν είναι τα νευρωνικά δίκτυα ακτινικής βάσης (Radial Basis Function Neural Networks, RBF NNs), γνωστά για την απλότητα και τη γρήγορη εκπαίδευσή τους, και η συνεργατική βελτιστοποίηση σμήνους σωματιδίων (Cooperative Particle Swarm Optimization, CPSO), αναγνωρισμένη για τα βελτιωμένα αποτελέσματα βελτιστοποίησης μέσω της συνεργασίας μεταξύ πολλαπλών σμηνών.

Ένα σημαντικό ζήτημα που αντιμετωπίστηκε σε αυτή τη διατριβή είναι το πρόβλημα του ελέγχου μη γραμμικών συστημάτων με τη χρήση μη γραμμικών μεθοδολογιών ελέγχου. Οι κύριες μεθοδολογίες ελέγχου που χρησιμοποιούνται είναι η μέθοδος backstepping και η μέθοδος ελέγχου με τη χρήση μοντέλων πρόβλεψης (Model Predictive Control, MPC). Ο έλεγχος backstepping χρησιμοποιείται για την παραγωγή ευσταθών εύρωστων ελεγκτών για τη περίπτωση συστημάτων αυστηρής ανατροφοδότησης, ενώ η μέθοδος MPC περιλαμβάνει το σχηματισμό και την επίλυση ενός προβλήματος βελτιστοποίησης σε πραγματικό χρόνο, επιτρέποντας την ακριβή πρόβλεψη της μελλοντικής συμπεριφοράς του συστήματος, αλλά και τον έλεγχο σε πολύπλοκα δυναμικά συστήματα με περιορισμούς και διαταραχές.

Τα κύρια μη γραμμικά συστήματα που διερευνώνται στην παρούσα διατριβή είναι τα μη επανδρωμένα εναέρια οχήματα (Unmanned Aerial Vehicles, UAVs), και ιδιαίτερα τα τετρακόπτερα εναέρια οχήματα (Quadrotor Unmanned Aerial Vehicles, QUAVs). Ο έλεγχος ενός τετρακόπτερου για το πρόβλημα της παρακολούθησης πηγαίας τροχιάς, αποτελεί σημαντική πρόκληση λόγω της εγγενώς μη γραμμικής και υποδιεγειρόμενης φύσης του οχήματος. Στην παρούσα διατριβή το πρόβλημα της παρακολούθησης πηγαίας τροχιάς αντιμετωπίστηκε με την ανάπτυξη ενός νέου μη γραμμικού ελεγκτή backstepping, ο οποίος ενσωματώνει νευρωνικά δίκτυα RBF. Οι ελεγκτές backstepping που έχουν τις ρίζες τους σε εξισώσεις πρώτων αρχών, αντιμετωπίζουν τη σημαντική πρόκληση του αποτελεσματικού χειρισμού εγγενών μη γραμμικοτήτων, αλλά είναι ευάλωτοι σε μη μοντελοποιημένες δυναμικές και αβεβαιότητες όταν εφαρμόζονται σε πρακτικές εφαρμογές. Για την αντιμετώπιση αυτού του προβλήματος, στην παρούσα διατριβή προτείνεται η διαμόρφωση ενός νέου ελεγκτή backstepping με ενσωματωμένα νευρωνικά δίκτυα RBF για τον χειρισμό αβεβαιοτήτων κατά την διάρκεια της παρακολούθησης πηγαίας τροχιάς του τετρακόπτερου. Με αυτόν τον τρόπο, παρέχεται μια προσέγγιση βασισμένη

σε δεδομένα για τον υπολογισμό των μη-μοντελοποιημένων αβεβαιοτήτων, με σκοπό τον επιτυχή έλεγχο του τετρακόπτερου.

Εκτός από την ανάπτυξη αποτελεσματικών στρατηγικών ελέγχου παρακολούθησης της τροχιάς ενός τετρακόπτερου, είναι εξίσου σημαντική και η κατάλληλη βαθμονόμηση των παραμέτρων ελέγχου αυτών των μεθόδων. Η δυσκολία στην βαθμονόμηση αυξάνεται όταν χρησιμοποιούνται περισσότεροι από ένας ελεγκτές για τη ρύθμιση του συστήματος, κάτι που δημιουργεί μεγάλο αριθμό παραμέτρων προς βαθμονόμηση. Για τον σκοπό αυτό, σε αυτή τη διατριβή προτείνεται ένα νέο πλαίσιο συνεργατικής βελτιστοποίησης CPSO κατάλληλο για τη βέλτιστη βαθμονόμηση των παραμέτρων ενός τετρακόπτερου συστήματος για το πρόβλημα παρακολούθησης πηγαίας τροχιάς. Το διαμορφωμένο πλαίσιο ελέγχου περιλαμβάνει δύο υποσυστήματα: έναν ελεγκτή MPC για την παρακολούθηση της θέσης και ένα σχήμα PID για τη σταθεροποίηση της στάσης του τετρακόπτερου. Ανταλλάσσοντας πληροφορίες, τα δύο σμήνη συνεργάζονται μεταξύ τους για την αποτελεσματική εξερεύνηση του χώρου αναζήτησης με σκοπό την εύρεση παραμέτρων βαθμονόμησης οι οποίες καλυτερεύουν την ικανότητα παρακολούθησης της επιθυμητής πηγαίας τροχιάς.

Συγχρόνως με την ανάπτυξη ενός αλγορίθμου για τη βέλτιστη βαθμονόμηση των παραμέτρων ελέγχου ενός τετρακόπτερου, με στόχο την αντιμετώπιση και την επίλυση προβλημάτων υψηλής διαστατικότητας που δεν είναι κυρτά, δημιουργήθηκαν δύο ακόμη νέοι συνεργατικοί αλγόριθμοι σμήνους σωματιδίων. Ειδικότερα, αναπτύχθηκαν δύο νέα πλαίσια CPSO για την αντιμετώπιση των προβλημάτων που σχετίζονται με τη διαχείριση της άεργους ροής ισχύος (Optimal Reactive Power Flow, ORPF) σε έξυπνα κατανεμημένα δίκτυα (Distributed Networks, DNs) και τη αναγνώριση των κρίσιμων παραμέτρων σε εγκαταστάσεις επεξεργασίας λυμάτων (Wastewater Treatment Plants, WWTPs). Συγκεκριμένα, διαμορφώθηκε ένα νέο συνεργατικό πλαίσιο βελτιστοποίησης και ελέγχου CPSO για την αντιμετώπιση του προβλήματος διαχείρισης ισχύος σε έξυπνα δίκτυα με υψηλό αριθμό φωτοβολταϊκών (Photovoltaic, PV) συσκευών. Επιπλέον, με σκοπό την αναγνώριση των παραμέτρων λειτουργίας σε μοντέλα εγκαταστάσεων επεξεργασίας λυμάτων διαμορφώθηκε νέο πλαίσιο CPSO το οποίο χρησιμοποιήθηκε για την επίλυση ενός μη γραμμικού προβλήματος βελτιστοποίησης.

Ένα άλλο κρίσιμο πρόβλημα που αντιμετωπίστηκε σε αυτή τη διατριβή σχετίζεται με τη μοντελοποίηση και τον έλεγχο μη γραμμικών χρονικά μεταβαλλόμενων συστημάτων. Σε αυτό το πρόβλημα, η δυσκολία δεν αφορά απλώς τη χρήση γραμμικών ή μη γραμμικών μοντέλων, αλλά αναφέρεται στο γεγονός ότι το μοντέλο που χρησιμοποιείται πρέπει να είναι προσαρμοστικό, ώστε να είναι σε θέση να παρακολουθεί τις αλλαγές στη δυναμική του συστήματος. Στην παρούσα διατριβή παρουσιάζεται ένα νέο μη γραμμικό πλαίσιο ελέγχου στο οποίο ενσωματώνονται

προσαρμοστικά μοντέλα νευρωνικών δικτύων. Συγκεκριμένα, αναπτύχθηκε ένα ολοκληρωμένο πλαίσιο μη γραμμικού προσαρμοστικού ελέγχου, για την εξασφάλιση καλής απόδοσης ελέγχου σε διάφορες περιοχές λειτουργίας. Ο νόμος ελέγχου του συστήματος κλειστού βρόχου αποδεικνύεται ασυμπτωτικά ευσταθής με τη χρήση της θεωρίας ευστάθειας Lyapunov. Στο πλαίσιο αυτό διεξάγονται δύο αναλύσεις υποθέσεων, που αφορούν ένα μη γραμμικό αυτοπαλινδρομούμενο εξωγενές σύστημα (Nonlinear Autoregressive Exogenous, NARX) και έναν χημικό αντιδραστήρα συνεχούς ροής με ανάδευση (Continuous Stirred Tank Reactor, CSTR).

Οι στρατηγικές που παρουσιάζονται στην παρούσα διατριβή αξιολογούνται σε μια σειρά αναλύσεων υποθέσεων συμπεριλαμβανομένων προσομοιωμένων σεναρίων. Η αποτελεσματικότητα αυτών των προτεινόμενων μεθόδων παρουσιάζεται μέσω συγκρίσεων με άλλες προσεγγίσεις που αναφέρονται στη βιβλιογραφία.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Αυτόματος Έλεγχος, Υπολογιστική Νοημοσύνη

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: αυτόματος έλεγχος, βαθμονόμηση, έλεγχος backstepping, νευρωνικά δίκτυα ακτινικών συναρτήσεων βάσης, παρακολούθηση πηγαίας τροχιάς, συνεργατικές μεθευρετικές μέθοδοι, συνεργατικός αλγόριθμος σμήνους σωματιδίων, τετρακόπτερο, υπολογιστική νοημοσύνη

ABSTRACT

The objective of this dissertation is to develop and fine-tune automatic control techniques for nonlinear systems, with a focus on unmanned aerial vehicle control, through the application of computational intelligence methods. Specifically, this research focuses on utilizing radial basis function networks (RBFNs), known for their simplicity and fast training, and cooperative particle swarm optimization (CPSO), recognized for its improved optimization results through collaboration among multiple swarms.

An important issue faced in this dissertation is the problem of controlling nonlinear systems by utilizing nonlinear control methodologies, primarily backstepping control and model predictive control. Backstepping control offers robustness, and stability for non-strict feedback systems, whereas the model predictive control (MPC) method involves formulating and solving an optimization problem at discrete time steps, enabling accurate prediction of future system behavior and control in complex dynamic systems with constraints and disturbances.

The main nonlinear systems that are investigated in this dissertation are unmanned aerial vehicles (UAVs), with a specific focus on quadrotor vehicles. Controlling the quadrotor, especially concerning trajectory tracking, presents a significant challenge due to its inherently nonlinear and underactuated nature, characterized by intercoupled terms. In this thesis the trajectory tracking problem was addressed by developing a new nonlinear backstepping controller which integrates RBF neural networks. Backstepping controllers are based on first-principles equations to face the significant challenge of effectively handling inherent nonlinearities, but are vulnerable to unmodeled dynamics and uncertainties in practical applications. To tackle this challenge, the thesis proposes a novel solution which integrates a backstepping controller with RBF networks for handling uncertainties during quadrotor trajectory tracking, thus offering a data-driven approximation for handling unmodeled uncertainties.

In addition to developing an effective tracking control strategy for a quadcopter, it is equally important to properly tune its control parameters, especially when more than one controller is used for regulating the system. To this end, in this thesis, a novel CPSO optimization framework is designed for optimizing the tuning parameters of a quadrotor trajectory tracking control scheme. The control framework included two subsystems: an MPC controller for position tracking and a PID scheme for attitude stabilization. This approach involves collaborative optimization of the numerous controllers tuning parameters, resulting in improved tracking performance, enhanced robustness and efficient optimization within reasonable timeframes.

In tandem with the development of an algorithm for the optimal tuning of a quadcopter's control parameters, two additional cooperative particle swarm algorithms were also devised to address and resolve high-dimensional non-convex problems. To this end, two novel CPSO frameworks were formulated to address the problems related to optimal reactive power flow (ORPF) management in smart distribution grids and critical parameter identification in WWTPs. To be more specific a CPSO optimization and control framework was designed in order to tackle the reactive power flow (RPF) problem of photovoltaic-heavy distribution networks. Furthermore, in response to the estimation of critical parameters challenges faced in wastewater treatment processes (WWTPs), a new CPSO-identification framework was proposed that can be used for solving a nonlinear optimization problem.

This thesis also addresses another crucial issue concerning the modeling and control of nonlinear time-varying systems. In this context, the challenge lies not only in choosing between linear and nonlinear models but, more importantly, in ensuring that the model employed can adapt its parameters so as to effectively track changes in the system's dynamics. In this thesis, a new nonlinear control framework is presented in which adaptive neural network models are incorporated. A comprehensive framework for nonlinear adaptive control is developed, ensuring satisfactory control performance across various operation regions. The control law of the closed-loop system is proven to be asymptotically stable using Lyapunov stability theory. Two case studies are conducted within this framework, involving a nonlinear autoregressive exogenous (NARX) system and a time-varying continuous stirred tank reactor (CSTR).

The strategies presented in this dissertation are evaluated across a range of case studies, including simulated scenarios. The effectiveness of these proposed schemes is showcased through comparisons with other approaches documented in the bibliography.

SUBJECT AREA: Automatic Control, Computational Intelligence

KEYWORDS: automatic control, backstepping, computational intelligence, cooperative metaheuristics, cooperative particle swarm optimization, model predictive control, quadrotor, radial basis function neural networks, trajectory tracking, tuning

Στους γονείς μου, Σάββα και Γεωργία.

List of Publications

Journal publications:

A. Kapnopoulos, A. Alexandridis "*A cooperative particle swarm optimization approach for tuning an MPC-based quadrotor trajectory tracking scheme*", Aerospace Science and Technology, 127 (2022), pp 107725. (*Impact Factor: 5.6*)

M. Papadimitrakis, A. Kapnopoulos, S. Tsavartzidis, A. Alexandridis "*A cooperative PSO algorithm for Volt-VAR optimization in smart distribution grids*", Electric Power Systems Research, 212 (2022), pp. 108618. (*Impact Factor: 3.9*)

A. Kapnopoulos, C. Kazakidis A. Alexandridis "*Quadrotor trajectory tracking based on backstepping control and radial basis function neural networks*", Results in Control and Optimization, 14 (2024) 10335.

Conference publications:

I. Kalogeropoulos, T. Protoulis, I. Kordatos, A. Kapnopoulos, P.L. Zervas, H. Sarimveis, A. Alexandridis "*An integrated PSO-DMC framework for the identification and control of wastewater treatment plants"* 1st International Conference on Sustainable Chemical and Environment Engineering, Rethymno, Greece (2022)

T. Protoulis, I. Kalogeropoulos, I. Kordatos, A. Kapnopoulos, P.L. Zervas, H. Sarimveis, A. Alexandridis, "*An Identification and Control Framework for Optimizing the Energy Consumption of a Wastewater Treatment Plant*", IEEE 6th International Conference and workshop in Obuda on Electrical and Power Engineering (CANDO EPE), Budapest, Hungary (2023)

Papers under preparation:

Kapnopoulos, A. Alexandridis, *"Nonlinear Model Predictive Control Based for an Online Adjustable RBF Neural Network"* (υπό συγγραφή)

Contents

Contents

List of figures

List of Tables

Abbreviations

ΠαΔΑ	Πανεπιστήμιο Δυτικής Αττικής
UniWA	University of West Attica
ΕΜΠ	Εθνικό Μετσόβιο Πολυτεχνείο
NTUA	National Technical University of Athens
ΤΗΗΜ	Τμήμα Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών
NNs	Neural networks
RBF	Radial Basis Function
PSO	Particle Swarm Optimization
SFM	Symmetric Fuzzy Means
DNs	Distributed Networks
ORPF	Optimal Reactive Power Flow
WWTPs	Wastewater Treatment Plants
MPC	Model Predictive Control
UAV	Unmanned Aerial Vehicle

Forward

Θα ήθελα να ευχαριστήσω κάποιους ανθρώπους, των οποίων η συμβολή υπήρξε καθοριστική για την ολοκλήρωση αυτής της διατριβής.

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα Καθηγητή μου Αλέξανδρο Αλεξανδρίδη. Η εμπιστοσύνη που έδειξε στις δυνατότητες μου, οι επιστημονικές του συμβολές και η συνεχής καθοδήγηση και στήριξη του, ήταν από τους βασικούς λόγους ολοκλήρωσης της παρούσας διατριβής.

Επίσης οφείλω να ευχαριστήσω τον Καθηγητή Διονύσιο Κανδρή και τον Αναπληρωτή Καθηγητή Ηλία Ζώη για την συμμετοχή τους στην τριμελή συμβουλευτική επιτροπή και τις πολύτιμες συμβουλές τους κατά την εκπόνηση της διατριβής.

Ευχαριστώ ακόμα θερμά τα νυν και πρώην μέλη του εργαστηρίου Τηλεπικοινωνιών, επεξεργασίας σήματος και Ευφυών συστημάτων, Δέσποινα Καραμιχαηλίδου, Μύρων Παπαδημητράκη, Νικό Γιαμαρέλο, Θεόδωρο Πρωτούλη, Κορδάτο Ιωάννη και Μάριο Στογιάννο για τη συμπαράσταση και την φιλική τους συνεργασία όλα αυτά τα χρόνια.

Επιπρόσθετα νιώθω την ανάγκη να ευχαριστήσω όλα τα άτομα που με στήριξαν στο ταξίδι του διδακτορικού μου (Χρήστο, Μιχάλη, Εύη, Παναγιώτη, Έλενα, κ.α.) και ιδιαιτέρως τον φίλο μου Στάθη που με τιμά με τη φιλία του.

Τέλος, θα ήθελα να ευχαριστήσω ιδιαίτερα τους γονείς μου Σάββα Καπνόπουλο και Γεωργία Μεϊντάνη, και τον αδελφό μου Νικόλαο Καπνόπουλο για τη διαρκή στήριξη, δύναμη και κουράγιο που μου παρείχαν στη προσπάθεια αυτή.

1. Introduction

Automatic control refers to the application of different methods and strategies to govern and manipulate systems or processes without direct human intervention. In the dynamic field of engineering and technology, computational intelligence techniques play an essential role in enhancing the efficiency of the control task. These techniques harness the power of artificial intelligence and machine learning to enable systems to adjust, improve and make decisions in intricate and constantly evolving environments. For example, in industrial automation, fuzzy logic controllers are utilized to effectively manage variables like temperature and pressure within production processes, securing the optimal product quality and uniformity [1], [2]. Similarly, in autonomous vehicles, deep learning neural networks are employed to process real-time data and make critical decision, such as identifying and classifying objects (e.g., pedestrians, traffic signs, other vehicles) predicting their behavior, and planning a safe path for vehicle to follow [3], [4].

The precision of system modeling is essential for enabling successful regulation results, as computational modeling aids in designing, analyzing, and optimizing control systems for enhanced efficiency. In last decades, there has been a significant research focus on modeling Artificial Neural Networks, commonly referred to as Neural Networks (NNs) which are powerful mathematical tools within the realm of computational intelligence, capable of modeling unknown systems using input-output data exclusively. The construction of a neural network model generally consists of two key phases: initially defining the network's architecture, which includes specifying the quantity of hidden layers and neurons, and subsequently, optimizing the network's parameters associated with neurons and synaptic weights through an optimization algorithm that minimizes discrepancies between the actual system outputs and the network's predictions based on a training dataset. During this training process, the network internalizes the correlations between input and output variables.

While most traditional neural network training methodologies heavily rely on optimization algorithms that demand substantial computational resources and time, radial basis function networks [5] stand out as a distinct architecture offering notable advantages, including a simpler structure, quicker training algorithms, and enhanced approximation capabilities. Due to their advantageous characteristics, RBF networks have seen widespread use in modeling [6]–[8] and control [9], [10].

RBF network training algorithms can be categorized into two groups: one with a predetermined hidden neuron count, often time-consuming as they necessitate a trial-and-error process to determine the correct number of neurons, and another with structure selection mechanisms. The k-means [11] clustering algorithm, belonging to the first category, is a popular choice for selecting the centers coordinates in the hidden layer. The latter category includes algorithms that simultaneously determine network and structure parameters [12], [13], but many of them involve extensive computational requirements. In any case, once the centers are known the rest of the network's parameters are defined with the use of linear least squares.

In recent years, the fuzzy-means (FM) algorithm [14] has emerged as a highly effective training method for RBF networks, distinguished by its ability to successfully define the right number of hidden nodes and compute model parameters. Its's notable advantages include its speed, as it requires only a single pass of training examples, its consistency in results since it doesn't rely on random initializations, and its capacity to simultaneously determine both network structure and parameters, thus minimizing the need for iterative trials. The wide range of applications [15]–[17] of the FM algorithm showcases the efficiency and versatility that it offers in modeling and controlling systems.

While the right integration and utilization of computational intelligence techniques are pivotal for the successful modeling of systems, it is equally important to select capable and efficient control methods to achieve optimal outcomes. In practice, many systems exhibit nonlinear behavior, making linear control strategies inadequate in dealing with the inherent nonlinear traits of these systems. As a result, nonlinear control methods become indispensable tools, enabling the design of regulators that can ensure stability, enhance performance, and preserve robustness in challenging and dynamically changing operational conditions. Despite the numerous advantages offered by some nonlinear control methods, such as having the capacity to deliver robust control solution in the face of uncertainties and disturbances, manage non-smooth system behavior, handle nonlinear interactions, incorporate constraints and adapt to dynamic conditions, the design of nonlinear controllers for a specific regulation problem remains a complex and demanding task.

Backstepping control is a nonlinear control method known for its ability to effectively stabilize complex nonlinear strict-feedback systems, making it especially valuable in practical applications. It is based on decomposing the problem in several steps, where in each step a virtual controller is designed to stabilize the former system, so that in the end all the subsystems are stable. Backstepping control's key features lie in its robustness against uncertainties and disturbances, precise tracking of desired trajectories, adaptability to dynamic conditions. For this reason, many researchers have used controlling schemes based on the backstepping strategy for various regulation tasks.

Indeed, a wide variety of backstepping-based controllers have been introduced to address system regulation challenges. In [18] six integral backstepping controllers are designed for the full autonomous flight of quadrotor, and their effectiveness has been demonstrated through real flight experiments. Van et al. in [19] devise an adaptive backstepping nonsingular fast terminal sliding controller for tracking control of robot manipulators. The controller showcases superior tracking performance when tested on the PUMA560 robot compared to other controllers. Wen and Changyun [20] develop two robust adaptive algorithms by using backstepping approaches for uncertain nonlinear systems, incorporating a Nussbaum function to handle input saturation while ensuring global stability and adjustable transient performance. Simulation tracking results on spring-mass system signify the controller’s performance in handling input saturation.

Another important nonlinear control methodology is model predictive control, widely used in various fields [21]–[24]. The MPC method relies on the simple idea of using an explicit dynamic model of the system to predict the effect of future actions on the output. The control actions are determined through an optimization procedure, with the objective of minimizing the predicted error. It offers advantages such as the ability to handle complex, multivariable systems, address constraints on inputs and states, and handle disturbances and uncertainties. MPC’s predictive behavior allows it to make informed control moves based on future predictions, making it effective for both tracking desired trajectories and regulating tasks. Over the last decades, MPC methodologies have gained substantial traction within both the academic sphere [25], [26] and the industry [27], resulting in multiple of successful applications [28].

A multitude of MPC methodologies have been proposed in the literature in a variety of systems, with a common characteristic among the methodologies being the formulation of a real-time optimization problem that requires resolution. In [29] an MPC controller is devised which utilizes obstacle ship trajectory prediction models based on RBF networks. The method is tested with real port-data assessing trajectory safety and efficiency when compared with other control techniques. In [30] a MPC scheme is introduced that integrates direct and indirect neural control methods by employing an RBF model that accelerates optimization problem-solving. The controller’s performance is evaluated on a nonlinear inverted pendulum on cart. In another interesting publication [31] an adaptive framework for robust nonlinear MPC is designed that enhances aircraft control capabilities under adverse conditions and disturbances. The controller is tested successfully on a real unmanned aerial system. Xia et al. propose a finite-state MPC strategy tailored specifically for permanent-magnet brushless DC motors to reduce commutation torque ripple. Simulation results verify the method’s ability to minimize commutation torque ripple in both dynamic and steady-state conditions.

Another crucial task regarding MPC optimization is related with the stability of the control systems. The literature on MPC stability for dynamic systems, both linear and nonlinear, is extensive [25], [32], [33]. Popular methodologies in the literature fall into two main categories: those employing a cost function as a Lyapunov candidate function, including terminal constraint [34], infinite output prediction horizon [34], terminal cost function [35], and terminal constraint set methods [36]; and those relying on state decrease in specific norms [37].

Considering the critical significance of effectively controlling nonlinear systems, it becomes evident that relying on nonlinear controllers presents certain advantages. Indeed, the control of nonlinear systems poses a challenging endeavor due to their complex and often unpredictable behaviors. In practice, nearly all real-world systems exhibit some kind of nonlinear characteristic, making it vital to navigate the nonlinear attributes. These systems can exhibit abrupt changes, discontinuities, and intricate interactions among variables, rendering conventional linear control methods insufficient for accurately capturing their dynamics.

In this thesis, the central emphasis lies in the in-depth investigation of nonlinear dynamics and control strategies within the domain of unmanned aerial vehicles. Over the past few decades, vertical takeoff and landing UAVs have garnered significant attention. Among the noteworthy UAVs in the family of rotary-wing aircrafts [38], [39], the quadrotor, or quadcopter, holds a prominent position. A quadrotor helicopter is a vehicle that features four propellers, enabling a reduction in the size of each rotor while maintaining or even increasing the total load capacity, compared to single-rotor helicopters. This design provides the quadrotor with advantages related with high maneuverability, precise movements in limited space, the ability for stationary flight (hovering) and the capacity for vertical take-off and landing. The ample set of abilities that the quadrotor possesses has led to a growing implementation in several industries such as surveillance, rescue, photography, forest patrolling, and agricultural quality inspection [40]–[43]. The escalating practical utility of quadrotors has prompted researchers towards an in-depth study regarding their design, modeling and control [44], [45].

Examining the dynamics of a quadrotor vehicle through the modeling of their differential equations plays a crucial role in analyzing and governing these versatile aerial vehicles. Within this context, two traditional frameworks, namely Lagrangian and Newton-Euler frameworks, are commonly employed. The Lagrangian formalism [46], [47] approaches the dynamics from the perspective of energy, expressing the system in terms of generalized coordinates and Lagrangian functions. On the other hand, the Newton-Euler formalism [48], [49] at specific point in space and time, providing a spatial perspective of the dynamics. In the present thesis the Newton-Euler formalism is selected due to its inherent advantages of providing a more direct approach to calculating forces and torques, facilitating real time control. Moreover, the system dynamics are represented using the state space framework.

The control of a quadrotor is not a simple task, primarily because it inherently embodies characteristics of instability, under actuation, nonlinearity, and a statue as a multiple-input-multiple-output (MIMO) nonlinear system with strongly couple dynamic terms. Amidst this intricate context, the task of controlling a quadrotor for trajectory tracking becomes even more challenging. It requires meticulous control of the quadrotor's dynamic behavior to achieve accurate and robust tracking performance.

Early attempts to quadrotor flight control utilize techniques such as the proportional-integral-derivative (PID) controller, the linear quadratic regulator (LQR) and the H-infinity controller (H_∞). Because of their simplicity in implementation and flexibility in parameter tuning, PID-based control strategies have been widely adopted by researchers for developing autonomous quadrotor flight systems [50]. The LQR controller constitutes another standard control scheme for quadrotors, which has been applied for trajectory planning [51] and stability control [52]. In [53] a comparison between the performance of PID and LQR controllers is conducted, including experimental results. As far as the application of H_∞ for quadrotor control is concerned, Wang et al. [54] designed a tracking controller with the use of variation-based linearization to track a reference trajectory. Raffo et al. [55], [56] proposed a method for solving the path tracking problem by using and H_∞ controller that achieves path following in the presence of external disturbance and model uncertainties. Simulation results from both works show the robustness ability of the H_∞ controllers.

Although the mentioned control strategies display advantages in terms of simplicity and implementation, they heavily rely on linearized system models. However, given the underactuated and inherently nonlinear nature of quadrotors, linearization leads to imprecise models that only approximate its behavior in regions near the equilibrium points. As a response to the challenges, especially in achieving accurate trajectory tracking posed by the complex dynamics of quadrotors, researchers have increasingly adopted nonlinear control methods to improve the stability and controllability.

Feedback linearization constitutes a classical way in controlling nonlinear systems. The method involves a transformation of the nonlinear control system into a linear one through control input and state transformation. Das et al. [57] proposed a two-loop approach nonlinear controller with an outer PD loop and an inner feedback linearizing controller loop that deals with the coupling dynamics problem of the quadrotor. In [58] a controller is designed that combines control strategies, including feedback linearization, to cope with the nonlinear dynamics of the system, resulting in satisfactory attitude performance. In [59], an effective nonlinear controller is proposed, designed to handle both position tracking and attitude stabilization. This controller is successfully implemented on an actual quadrotor through a backstepping-feedback linearization approach. This method accounts for physical parameter challenges and underwent outdoor experimentation for validation. While feedback linearization control overcomes the capabilities of linear controllers, it shares some limitations with them. Precise modeling is required, and it can't efficiently handle external disturbances.

Sliding mode control (SMC) consists of a useful tool when dealing with quadrotor and nonlinear systems in general. The main idea of the SMC is to design control strategies that compel the system's state trajectory to adhere near to a predefined manifold in the state space. The first applications of the control method were studied by Sira-Ramirez [60] for attitude control of a small helicopter. In [61] a model-free based controller is combined along with an SMC controller to solve the position and attitude trajectory problem for a quadrotor. In another interesting publication [62] the advantages from the utilization of an adaptive SMC quadrotor neural network controller for attitude and position control are showcased. In [63] a trajectory tracking designing process was combined with a cuckoo search algorithm for reducing the power consumption of a quadrotor that uses a terminal SMC. Tang et al. [64] propose a fault-tolerant terminal SMC for robust trajectory tracking control of a quadrotor under disturbances, parametric uncertainties and actuator faults.

While the SMC algorithm can effectively handle the nonlinearities present in quadrotors, it is very sensitive to chattering phenomena which could be caused by unmodeled dynamics.

Backstepping constitutes a control technique which offers advantages such as improved robustness against chattering phenomena resulting from unmodeled dynamics while effectively addressing the system's nonlinearity. In [65], a trajectory tracking controller is proposed, utilizing a backstepping method with robust compensation theory. [66] presents a backstepping adaptive controller by combining it with a prescribed performance function that guarantees quadrotor trajectory tracking performance. In [67], a command filter-based backstepping control approach with input saturation is employed to address the quadrotor's vehicle trajectory tracking problem. A trajectory tracking fault tolerant controller is applied in [68] by using a sliding mode backstepping approach. In [69] a backstepping-based quadrotor controller is designed for path following while addressing challenges related with unknown vehicle parameters and external disturbances.

Though the aforementioned backstepping controllers effectively handle nonlinearities, they rely on models derived from first principles equations, making them susceptible to unmodeled dynamics and uncertainties when applied to real-world systems. Neural networks, as black box modeling techniques, are frequently used in conjunction with backstepping methodologies, to produce robust controllers that can handle uncertainties. In [70], C. Nicol et.al present an early attempt which utilizes NN in a quadrotor for handing model uncertainties. Specifically, a neural based quadrotor control scheme is introduced capable to attenuate modeling error and wind disturbance through NN weight adaptation. In another interesting publication [71], the authors utilize a neural network-based backstepping control scheme for quadrotor position and yaw angle desired setpoint tracking. [72] introduces a novel backstepping design scheme for the quadrotor, based on an NN for modelling uncertainties. In [73], a nonlinear controller based on backstepping and PID employs an NN to identify unmodelled dynamics. [74] focuses in solving the finite time tracking problem of a quadrotor by utilizing an NN-based control scheme for compensating actuator faults and external disturbances. [75] introduces a robust terminal sliding mode approach for quadrotor attitude and position control, integrating an RBF NN to mitigate the influence of external disturbances. The effectiveness of these NN models within the broader control scheme riles heavily on their modeling accuracy, which in turn depends on the architecture and training algorithm used.

Selecting RBF networks trained with the FM algorithm as the neural network modeling approach constitutes a highly effective choice, thanks the ability of the FM algorithm to deliver RBF networks with enhanced accuracy within shorter computational duration. This has resulted in the successful integration of the FM algorithm into various nonlinear control schemes [10], [15], [16], [30], [76].

RBF networks have been used in conjunction with backstepping to oversee a range of systems. In [77], an RBF-backstepping scheme was applied to regulate the equilibrium position of an electrohydraulic elastic manipulator, even in scenarios with variable stiffness. In another instance [78], within an aerial work platform vehicle, a backstepping control approach was implemented, incorporating RBF networks to reduce tracking errors in work platforms and effectively dampen vibrations. Surprisingly though, the literature is very scarce on using integrated RBF-backstepping networks for quadrotor trajectory tracking. In the few relative publications, the training procedure involves only the network weights [72], while the centers of the RBF network are selected arbitrarily; this could result in a subpar model for the uncertainties in terms of accuracy, bearing a detrimental effect on the performance of the overall control schemes

Furthermore, another way to enhance the performance of quadrotor trajectory tracking control schemes, involves the application of MPC methodology. In [79], an MPC control law is responsible for position control while a feed-forward controller guarantees the quadrotor's stabilization. In [80], a robust MPC controller is implemented in real flight scenarios. In [81] a state space error predictive controller is utilized for trajectory tracking, while an H_∞ controller stabilizes the attitude of the quadrotor. In [82] the control system is split into two subsystems: one for path following using a state-space predictive controller, and the other for attitude stabilization using a non-model based active disturbance rejection (ADCR) controller with a linear extended state observer strategy.

Irrespective of the chosen control method, effective parameter tuning is a crucial practical task, as inadequately selected controller parameters can impede the performance of a quadrotor. Unfortunately, although the MPC method has been highly successful for quadrotor control, it is often associated with a cumbersome parameter tuning procedure. This can be attributed to the high number of operational parameters that need to be selected, which increases further when the MPC method is coupled with different techniques that carry their own parameters, as is often the case in quadrotor control. Under such circumstances, performing the tuning procedure by trial and error is not effective and a mathematical optimization problem needs to be formulated and solved. Unfortunately, when viewing the MPC tuning as an optimization procedure, the high number of design variables is not the only obstacle to be encountered; the mixed integer and continuous nature of the design variables, combined with a complicated, nonlinear and multimodal objective function make the use of conventional optimization methods inadequate.

Metaheuristic search methods constitute a class of optimization algorithms that are better equipped to overcome such difficulties and provide better quality solutions to the tuning problem. By relying on stochastic search, metaheuristic search methods reduce the risk of getting trapped in local minima, while the generation and evolution of multiple solutions bestows increased exploration capabilities. This is due to their ability to solve problems that are too difficult or time-consuming to solve using traditional methods. Among the popular metaheuristic techniques are genetic algorithms (GA) [83], which emulate natural selection to evolve potential solutions over multiple generations; and simulated annealing (SA) [84], which emulates the annealing process in metallurgy, iteratively exploring neighboring solutions and gradually approaching near-optimal or optimal solutions. Another well-known technique is the ant colony optimization (ACO) [85] which is inspired by the behavior of ants in finding food, where artificial ants navigate solution spaces and deposit pheromones to guide the search towards best solutions. However, these metaheuristic approaches, exhibit some significant disadvantages including slow convergence, sensitivity to parameters and susceptibility to local optima.

Particle swarm optimization (PSO) [86] is another simple yet effective metaheuristic optimization method which belongs to the family of swarm intelligence; it relies on simulating the social structure of flocking birds and has been used successfully for tuning diverse control schemes [87] and optimization [88].

Not surprisingly, PSO has been utilized extensively for solving the quadrotor control tuning optimization problem. In [89], a PSO tuning process was used in order to design four decentralized PID controllers achieving stabilization for the quadrotor's altitude and attitude. In [90], a PSO algorithm is used to tune RBF neural networks responsible for adjusting the parameters of a PID controller coupled to the quadrotor. Yacef F. et.al in [91], implement a PSO-based tuning strategy to an integral back-stepping controller in order to achieve height and angle stabilization. PSO has also been used to optimize the control parameters in quadrotor path tracking problems. In [92] a path planning solution is presented for the inspection of a photovoltaic farm based on a PSO - Bezier curve algorithm. In [93], the PSO algorithm optimizes the parameters of an ADCR strategy for the sake of following a desired path. In the particular work the PSO-ADCR methodology is compared with the simple ADCR showing the improvement in settling time, overshoot and desired tracking error. In [94], a heterogenous comprehensive learning PSO is utilized to optimize the parameters of a quadrotor saturation-based controller in order to perform three-dimensional trajectories in space. Mahmoodabadi et al. in [95], used a multi-objective high exploration PSO algorithm to tune the membership functions of a quadrotor fuzzy controller based on the LQR methodology.

While simple variants of PSO have proven effective for optimizing quadrotor control strategies, they do have limitations, including constrained exploration of solution spaces, vulnerability to local optima, and decreased efficiency in high-dimensional environments. Therefore, the modularity of the respective optimization problem could be better addressed by cooperative PSO techniques. CPSO works by dividing a complex optimization problem into smaller subproblems and then utilizing multiple sub-swarms of particles to explore these sub-problems simultaneously. These sub-swarms collaborate by periodically sharing information. This cooperative approach favors diverse exploration and leverages the strengths of both local and global search resulting in high-quality-solutions. Numerous works [96]–[98] have highlighted the effectiveness of employing cooperative techniques in dealing with complex optimization problems and achieving success. While cooperative techniques have demonstrated their effectiveness in addressing complex optimization problems, it’s noteworthy that, to the best of the author’s knowledge, there is a relative scarcity of research on the application of cooperative techniques for tuning quadrotor control schemes.

Another important aspect of MPC control is the choice of the predictive model. The majority of MPC methodologies in both literature and industrial applications predominantly rely on linear models, with the Dynamic Matrix Control (DMC) [99] being the most popular among them. Its widespread acceptance can be attributed to its usage of a linear step response model, simplifying parameter determination, and the application of linear or quadratic objective function, streamlining the optimization process. However, in practical industrial scenarios, nonlinear processes are prevalent. Linear model-based controllers perform well primarily within the vicinity of their delivered linear models, but this operation region can become severely constrained when substantial nonlinearity is present.

Neural networks provide an interesting approach to modeling nonlinear systems and have been employed as models in MPC control methodologies. A recent overview regarding neural network modeling approaches for model predictive control is provided by Ren et al.

[100]. Specifically, Lanzetti et al. [101] propose a data-driven modeling approach by implementing a recurrent neural network within an MPC controller for managing systems commonly encountered in industrial applications. In [102] the authors introduce an MPC method for precise tracking in repetitive batch processes, by utilizing neural networks to identify system dynamics. Kang et al. [103] introduce an event-triggered MPC approach for robotic manipulators by combining neural network-based predive modeling, global learning for optimization and event-triggered solving to enhance tracking control on a real-world robot.

Introducing non-linear models into an MPC controlled system can help resolve problems associated with non-linearity in systems. However, it doesn't address the case of time-varying systems, where the use of static MPC framework can lead to poor controller performance or even instability. To address the issues arising in the control of time-varying processes, various adaptive tuning methods have been proposed. Early efforts regarding real-time adaptation or process models in MPC schemes can be witnessed in [104]–[106].

It should be noted that all the aforementioned approaches rely on linear models, but in recent years, there has been substantial advancement in the development non-linear adaptive MPC methods. Hedjar in [107], introduces an adaptive neural network MPC controller to address parameter variations and uncertainties in nonlinear systems improving tracking performance. In [108] the authors propose an online backpropagation NN to effectively control forging process, addressing their inherent time variance and nonlinearity. Carughi et al. [109] present an online neural network-enhanced MPC methodology for a UAV, improving trajectory tracking and handling actuator faults.

Another approach in neural-network MPC methods involves the use of RBF models as adaptive modeling techniques. The motivation behind this, is that RBF models offer enhanced adaptability and modeling accuracy, making them a promising choice for adaptive control systems. However, there have been relatively few studies conducted on the development of non-linear adaptive MPC methodologies. In [110] the authors introduce a control methodology for processes with multiple steady states, using an MPC framework with process dynamics modeled by RBF NNs. The approach is successfully applied to a challenging CSTR reactor with three steady state points. In another publication [111], a real time MPC method with self-organizing RBF NNs for nonlinear systems is proposed, which demonstrates effective tracking and disturbance rejection characteristics. Han et al. [112] propose a self-organizing recurrent RBF NN improving modeling accuracy with a spikingbased growing and pruning algorithm and adaptive learning. The designed scheme is applied to control dissolved oxygen concentration in a WWTPs, with simulation results demonstrating improved model fitting and control performance compared to existing methods.

Motivated by the aforementioned discussion, this thesis aims to contribute to the domains of automatic control and computational intelligence, addressing the challenges and gaps previously outlined. The primary contributions include:

- A novel approach for harnessing the CPSO framework has been developed to address complex optimization and control challenges in diverse applications. This CPSO algorithm is designed to efficiently optimize and control systems facing different issues, as demonstrated in two distinct applications. Firstly, the CPSO framework was applied to optimize and control distributed networks heavily reliant on photovoltaic systems. The scheme leverages the power of multiple swarms to optimize distinct zones within the network, effectively managing the growing complexity of the optimization problem as photovoltaic systems become more integrated. The CPSO algorithm combines exploration capabilities with zone-based exploitation for candidate solutions, making it well-suited for the challenges posed by these networks. Its effectiveness was assessed in reducing voltage deviations and minimizing real power losses in an IEEE distribution grid under various load and generation profiles. Additionally, the new CPSO framework was employed to identify critical system parameters in a first principles model of a WWTPs. By solving a nonlinear optimization problem based on a system identification scheme, it successfully estimated crucial parameters in a modified benchmark simulation model No.1 (BSM1). The performance of the CPSO algorithm was compared with two other PSO algorithms, specifically in the task of estimating the kinetic and stoichiometric coefficients of the modified BSM1 model.
- A new control strategy is proposed for solving the quadrotor trajectory tracking problem by utilizing a backstepping-RBF network framework. The proposed backstepping controller is designed so as to ensure Lyapunov stability for the closed-loop system. It combines a first-principles-based dynamic model of the quadrotor, and an RBF network which provides a data-driven approximation of unmodelled uncertainties of any type. The RBF network is trained using the FM algorithm, enhancing modeling accuracy and thereby enabling improved tracking performance in the presence of unmodeled dynamics. This methodology was

successfully tested in two different trajectory tracking scenarios. The simulation results against a classical backstepping control approach confirm the superiority of the proposed method.

- A novel comprehensive framework is introduced, employing cooperative PSO to produce an optimized quadrotor trajectory-tracking MPC controller. To be more specific, the employed framework comprises two control subsystems, namely an MPC controller for dealing with the path following problem and a PID scheme for dealing with attitude stabilization. The two different control subsystems require a large number of tuning parameters, which are optimized effectively by a cooperative PSO scheme, employing multiple swarms in order to handle the different components of the solution vector. In this case, two different swarms are used for the MPC and PID tuning parameters, respectively; though each swarm controls a different set of parameters, they ultimately work together towards bestowing improved trajectory tracking abilities to the integrated control framework. The proposed approach is evaluated through a series of experiments employing a number of different trajectories, while also testing the method’s robustness by applying it in trajectories different than the ones used for tuning; performance is compared to different metaheuristic search methods through statistical testing.
- A new nonlinear MPC strategy is developed that employs discrete adaptive dynamic RBF models for predicting the future behavior of nonlinear time-varying systems. The model adaption is achieved through the adaptive fuzzy means algorithm, which provides the benefit of dynamically adjusting both the network’s structure and its synaptic weights. The adaptive nonlinear MPC approach is thoughtfully based on Lyapunov stability theory by invoking the monotonicity with respect to time of the MPC cost function. In this sense, the stability of the closed-loop system is ensured. The effectiveness of this approach is illustrated using two different nonlinear systems: a NARX system and a time-varying CSTR reactor. In the case of the CSTR, the methodology is applied to control a reactor with three equilibrium points. The results demonstrate that the proposed controller successfully controls the CSTR across its entire operating range.

The structure of this doctoral thesis is as follows:

Chapter two introduces the key computational intelligence tools utilized throughout the thesis, highlighting the architecture of RBF neural networks and the swarm-based metaheuristic approach of cooperative PSO. Additionally, it demonstrates an application of the CPSO framework for optimizing and controlling photovoltaic-heavy distribution networks. Simulation results across diverse scenarios validate the efficacy of CPSO when compared to alternate approaches.

Chapter three introduces the primary control methods utilized in this thesis for nonlinear system control, i.e., the backstepping and model predictive control methods.

Chapter four initiates with an introduction to UAVs and quadrotor dynamics and control, subsequently delving into the quadrotor trajectory tracking control problem.

Chapter five of the PhD dissertation introduces two trajectory tracking control approaches for addressing the quadrotor trajectory tracking problem. It includes the induction of a Backstepping neural network control scheme and the development of a cooperative PSO controller tuning scheme. The effectiveness of these methods is demonstrated through a series of simulations and case studies.

Chapter six of the thesis presents and analyzes a nonlinear MPC strategy incorporating adaptive RBF models. The NMPC controller’s efficiency is evaluated through two case studies: a nonlinear NARX system and a time-varying CSTR system. Simulation results demonstrate the controller’s effectiveness.

Finally, the conclusion drawn from the completion of the thesis are presented in Chapter seven.

2. Computational Intelligence methods

2.1 Introduction to Computational Intelligence

Computational intelligence (CI) [113] refers to a field of study that explores the design and development of intelligent systems capable of performing complex tasks by simulating human-like cognitive processes. CI offers an arsenal of tools that can be applied in diverse scientific fields for dealing with problems that would have been difficult to solve with conventional approaches. CI mainly focuses on a number of core families of methods, which include artificial neural networks, swarm intelligence, artificial immune systems, fuzzy systems, evolutionary computation [114]–[117].

The objective of CI is to develop systems that learn from experience, adjust to evolving environments, make decisions and solve problems without the need for explicit programming. Through the utilization of extensive datasets and diverse computational techniques, these systems demonstrate intelligent behavior akin to humans such as pattern recognition, decision-making and problem-solving.

This chapter presents the primary computational intelligence methods utilized within the scope of this thesis.

2.2 Radial Basis Function Neural Networks

Artificial neural networks (ANNs) constitute a standard machine learning technique; essentially, they are considered very effective for the modeling of highly nonlinear systems or processes, and during the last 20 years they have been extensively used for the realization [10] of such models in order to design novel control schemes.

Radial basis function neural networks (RBFNNs) [117] constitute a popular machine learning technique with various applications in nonlinear system identification and system control. Due to their simple structure and improved accuracy, RBFNNs are widely considered for modeling nonlinear dynamics.

RBF networks are characterized by a specific architectural design, consisting of a single hidden layer that is directly connected to the output layer. This linear connection contributes to advantages in terms of training speed and efficiency compared to classical multilayer perceptron (MLPs).

This section introduces the fundamental formulation of the RBF NNs and describes a specific algorithm namely the symmetric fuzzy means for the training of the network, which is applied in the context of this thesis.

2.2.1 RBF Network structure

An RBF network is a simple feedforward neural network comprising of a single hidden layer. A typical RBF NN is shown in Figure 1. The input layer is followed by the hidden layer, where a nonlinear transformation takes place. More specifically, each hidden neuron utilizes a radial basis function centered around a center vector $\boldsymbol{c_l}$. In this work the modified thin plate spline function is used as the nonlinear radial basis function:

$$\boldsymbol{\varphi_l(k) = \mu_l\big(x(k)\big)^2 \cdot log\big(\mu_l\big(x(k)\big) + 1\big)} \tag{1}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^{N\times 1}$ denotes the *k*th input vector and $\varphi_l\colon \mathbb{R}^{N\times 1} \to \mathbb{R}$ is the output value of the *l*th hidden neuron and the activity $\mu_l(\boldsymbol{x}(k))$ of the *l*th node is the Euclidean distance between the *k*th input vector and the *l*th center vector given by

$$\boldsymbol{\mu_l\big(x(k)\big) = \|x(k) - c_l\| = \sqrt{\sum_{i=1}^{N} (x(k) - c_l)^2}\ ,\ \ k = 1, \dots, K} \tag{2}$$

where K is the total number of the data set and $\boldsymbol{c_l} \in \mathbb{R}^{\mathrm{N}\times 1}$ indicates the center vector of the *l*th hidden neuron.

For each data input and each node, an activation function value is calculated. The hidden node responses for the *k*th can be written as:

$$\boldsymbol{z(k) = \big[\, \varphi\big(\mu_1(x(k))\big), \varphi\big(\mu_2(x(k))\big), \dots, \varphi\big(\mu_L(x(k))\big)\big]} \tag{3}$$

The output of each hidden neuron $\boldsymbol{z}_l$ is multiplied by a synaptic weight $\boldsymbol{w_l}$ and then propagated towards the output layer, which consists of a linear combination of weighted nonlinear hidden basis functions. The final output of a multi-input and single-output (MISO) RBF neural network, can be described as:

$$\boldsymbol{\hat{y}(k) = \sum_{l=1}^{L} w_l(k) \cdot \varphi(\mu_l(x(k)))} \tag{4}$$

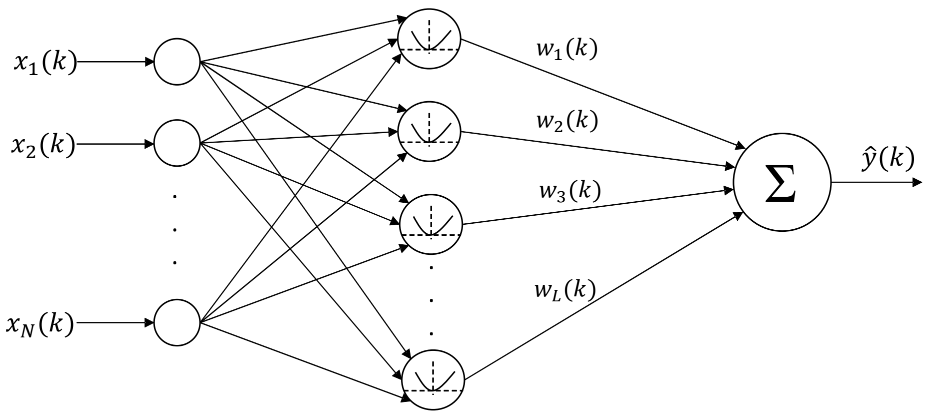

Figure 1: Typical structure of an RBF network

where L is the total number of hidden neurons and w_l is the synaptic weight between each hidden and output neuron.

Having the RBF centers fixed in the hidden layer, the synaptic weights are typically calculated using linear regression:

$$\boldsymbol{w^T} = \boldsymbol{Y^T} \cdot \boldsymbol{Z} \cdot (\boldsymbol{Z^T} \cdot \boldsymbol{Z})^{-\mathbf{1}} \tag{5}$$

where $\boldsymbol{Z} \in \mathbb{R}^{K \times L}$ is a matrix containing the hidden layer outputs, and $\boldsymbol{Y} \in \mathbb{R}^{K \times 1}$ is a vector that includes the target values.

Thus, establishing the hidden node centers is a vital step in designing of an RBF NN, as it holds significant importance for network's functionality and performance.

2.2.2 Training of RBF Networks - The Fuzzy Means Algorithm

The k-means algorithm [11] constitutes a standard approach in determining the coordinates of the RBF centers with a predefined number of centers. However, since the appropriate number of centers is often unknown in advance, it necessitates an arduous trial-end-error process to determine the optimal number. The fuzzy means algorithm [14] is highly suitable for addressing this task, because it adopts a fuzzy clustering approach in order to determine the node centers.

Consider for a system with N normalized input variables the universe of discourse (domain) of each variable x_i, $i = 1, \dots, N$ a number s of triangular 1-D fuzzy sets:

$$\boldsymbol{T_i} = \{\boldsymbol{A_{i,1}}, \boldsymbol{A_{i,2}}, \cdots, \boldsymbol{A_{i,s}}\}, \;\; \boldsymbol{i} = \mathbf{1}, \mathbf{2}, \dots, \boldsymbol{N} \tag{6}$$

where each fuzzy set $A_{i,j}$ is characterized by a center $\alpha_{i,j}$ $i = 1, \dots, N$ and $j = 1, \dots, s$ and its corresponding width $\delta\alpha_i$ which for a symmetric input partition i.e., $\delta\alpha_i = \delta\alpha_1 = \delta\alpha_2 = ,\cdots, = \delta\alpha_N = \delta a$ and can be fully described as:

$$\boldsymbol{A_{i,j}} = \{\boldsymbol{a_{i,j}}, \boldsymbol{\delta\alpha}\} \tag{7}$$

Each fuzzy set $\boldsymbol{A}_{i,j}$ is described by its own membership function regarding the input $\boldsymbol{x}_i(k)$ and is defined as:

$$\boldsymbol{\mu_{A_{i,j}}(x_i(k))} = \begin{matrix} \mathbf{1} - \dfrac{|\boldsymbol{x_i(k) - a_{i,j}}|}{\boldsymbol{\delta\alpha}}, & \boldsymbol{if\ x_i(k)} \in [\boldsymbol{a_{i,j} - \delta\alpha, a_{i,j} + \delta\alpha}] \\ \mathbf{0}, & \textbf{otherwise} \end{matrix} \tag{8}$$

By partitioning the entire input space using the fuzzy principles, we can generate a total of S multidimensional fuzzy subspaces $\boldsymbol{A}^m$, where $m = 1, \dots, S$

$$\boldsymbol{S} = \prod_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{s_i} \tag{9}$$

A compact way to represent each multidimensional subspace can be defined as:

$$\boldsymbol{A^m} = [\boldsymbol{A_1^m, A_2^m, \dots, A_N^m}]\,,\ \ \boldsymbol{m = 1, 2, \dots, S} \tag{10}$$

A subset of the above fuzzy subspaces depicts the candidate RBF centers, that are chosen so as to cover the data input space in a uniform way. The selection of the right subspace is based on the idea of a multidimensional membership function $\mu_{\boldsymbol{A}^m}(\boldsymbol{x}(k))$ defining a hypersphere of an input vector $\boldsymbol{x}(k)$ to fuzzy subspace $\boldsymbol{A}^m$:

$$\boldsymbol{\mu_{A^m}(x(k))} = \begin{cases} \mathbf{1} - \boldsymbol{rd^m(x(k))}, & \boldsymbol{if}\quad \boldsymbol{rd^m(x(k))} \le \mathbf{1} \\ \mathbf{0}, & \textbf{otherwise} \end{cases} \tag{11}$$

where $rd^m(x(k))$ denotes the Euclidean distance between the fuzzy subspace A^m with the input vector $x(k)$:

$$\boldsymbol{rd^m(x(k))} = \frac{\left[\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} (\boldsymbol{a_{i,j}^m - x_i(k)})^{\mathbf{2}}\right]^{\mathbf{1}/\mathbf{2}}}{\left[\boldsymbol{\Sigma_{i=1}^N (\delta\alpha)^2}\right]^{\mathbf{1}/\mathbf{2}}} \tag{12}$$

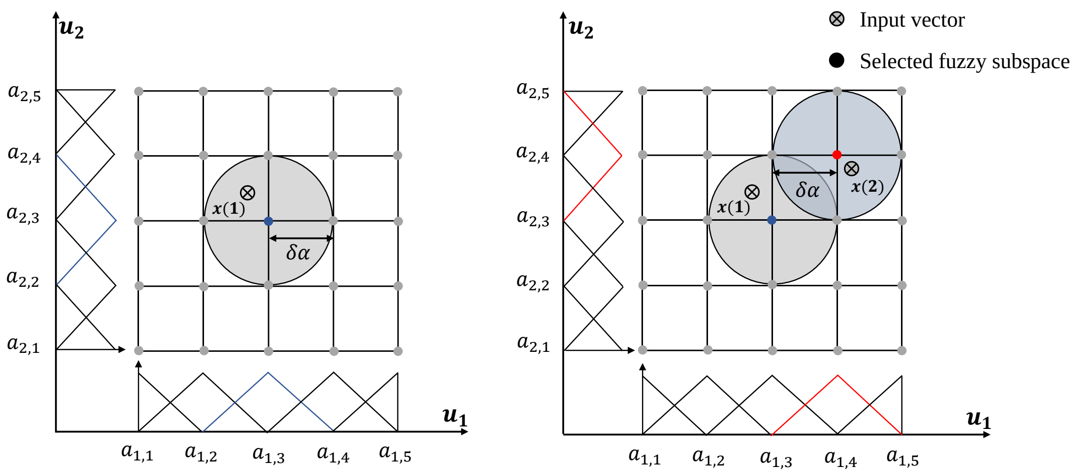

Figure 2: A 2-D input space fuzzy partition with a circular membership function

An overall view of the fuzzy means algorithm and determination of the closest fuzzy subspace to an input vector $\boldsymbol{x}$ is given in Figure 2 in which an example of a symmetric fuzzy partitioning for a 2-D input space is considered.

It is important to state here that only a single pass of the data is needed to determine both the number and location of RBF centers. This results to a fast-non-iterative procedure to find a subset of the subspaces, ensuring that all input data points are covered by at least one fuzzy subspace. Consequently, the resulting RBF NN primarily depends on the number of fuzzy sets, which can be determined efficiently through an exhaustive search within a narrow range.

2.3 Standard Particle Swarm Optimization

PSO [86] is a simple yet effective metaheuristic optimization method which belongs to the family of swarm intelligence (SI). The algorithm simulates the social structure of flocking birds flying in formation. To be more specific, the algorithm encodes a population of possible solutions known as particles, which 'fly' through the search space by taking into account the personal best location they have visited, as well as the global best solution achieved.

During each iteration, the particles move towards the direction of their own personal best solution found so far, as well as in the direction of the global best position found by the entire swarm. Each particle of the swarm is characterized by its position $x_i(t)$, its velocity

$v_i(t)$ and its best previous position in the search space $y_i(t)$, where $\hat{y}(t)$ denotes the best-known position of the entire swarm.

For a fitness function $\boldsymbol{f}$ to be minimized and a swarm consisting of $\boldsymbol{s}$ particles of dimensionality equal to $\boldsymbol{n}$, the following equations describe how the particles update their current position and velocity:

$$\boldsymbol{v_{i,j}(t+1) = w \cdot v_{i,j}(t) + c_1 \cdot r_{1,i} \cdot [y_{i,j}(t) - x_{i,j}(t)] + c_2 \cdot r_{2,i} \cdot [\hat{y}_j(t) - x_{i,j}(t)]} \tag{13}$$

$$\boldsymbol{x_i(t+1) = x_i(t) + v_i(t+1)} \tag{14}$$

where $v_{i,j}$, $i = 1,2,\dots,s$, $j = 1,2,\dots,n$ is the velocity for the jth dimension of the ith particle, c_1, c_2 denote the acceleration coefficients, *w* the inertia coefficient and $r_{1,i}$, $r_{2,i}$ are two random values sampled from a uniform distribution in the range [0,1].

Equation (13) consists of three parts. The first is the momentum or inertia part, that reflects the particle's tendency to maintain its current motion. The second is the cognitive part which represents the ability of the particle to reflect on its behavior and follow the best personal position found in the past. The last part is the social one, which depicts the particles' tendency to follow the optimal position found by their neighbors.

The personal best position of each particle is updated using the following equation,

$$\boldsymbol{y_i(t+1) = \begin{cases} y_i(t), & if\ f(x_i(t+1) \geq f(y_i(t)) \\ x_i(t+1), & if\ f(x_i(t+1) < f(y_i(t)) \end{cases}} \tag{15}$$

And the global best position of the swarm is defined as:

$$\boldsymbol{\hat{y}_i(t+1) = \arg\min_{y_i} f\big(y_i(t+1)\big), \quad 1 \leq i \leq s} \tag{16}$$

There are various variants of the PSO algorithm that incorporate different elements, such as different initialization techniques [118], constrictions factors [119], inertial weights [120] and cooperative particle partitions [121].

Specifically, in the latter case the inclusion of cooperative particle partitions in the PSO scheme proves to be particularly valuable, fostering collaboration among particles and greatly improving its performance in complex optimization problems where the search space is large or exhibits multiple sub-optimal solutions.

2.4 Cooperative Particle Swarm Optimization

The concept of cooperation between candidate solutions of a population has been applied in various metaheuristics, such as evolutionary algorithms and PSO [122], [123]. The CPSO-S framework [124] emerged as the first instance of cooperative behaviour within the PSO category, by splitting a solution vector of n parts into exactly n 1-D particles; a generalized version of this approach is introduced in [121] where the solution vector is split into N_c swarms with $N_c \leq n$, where n is the length of the solution vector.

It is obvious that a prerequisite for the implementation of cooperative approaches is the assortment of the n design variables in N_c swarms, which in turn indicate the presence of P_k individual distinct swarms, $k = 1,2, \dots, Nc$.

For each swarm P_k, the particle position $x_{i,j}^{k}(t)$ and velocity $v_{i,j}^{k}(t)$ are updated according to:

$$\boldsymbol{v_{ij}^k(t+1) = wv_{ij}^k(t) + c_1 r_{1,i}(t)\left[y_{ij}^k(t) - x_{ij}^k(t)\right] + c_2 r_{2,i}(t)\left[\hat{y}_i^k(t) - x_{ij}^k(t)\right]} \tag{17}$$

$$\boldsymbol{x_{ij}^k(t+1) = x_{ij}^k(t) + v_{ij}^k(t+1)} \tag{18}$$

where $y_{ij}^{k}(t)$ stands for the best personal position of particle i, in dimension j for swarm k at iteration t and $\hat{y}_i^k(t)$ denotes the global best position vector of particle i for swarm k at iteration t; $r_{1,i}(t)$ and $r_{2,i}(t)$ are randomly sampled numbers from a uniform distribution in the range [0 1], while c_1, c_2 denote the acceleration coefficients and w the inertia coefficient. In order to control the exploration-exploitation trade-off, a velocity clamping constant v_{max}^{k} is employed to regulate the particle positions in the range $[-v_{max}^{k}\ v_{max}^{k}]$.

As each swarm P_k contains a distinctive part of the original solution vector, the right cooperation between the swarms' agents is essential in order to calculate the fitness function for the overall optimization problem. This task is feasible by utilizing a context vector $\boldsymbol{x}_{P_k}$ formed by linking each of the particle positions of swarm P_k with the global best positions of the remaining swarms:

$$\boldsymbol{x_{P_k} = \left[\hat{y}^1, \dots, \hat{y}^{k-1}, x_i^k, \hat{y}^{k+1}, \dots, \hat{y}^{Nc}\right]} \tag{19}$$

This means that despite the fact that the network is partitioned into several distinct sub-swarms, the fitness function evaluation for each swarm's individual particle is estimated using the whole design vector**.** After forming the context vector $\boldsymbol{x}_{P_k}$, the fitness function evaluation $f(\boldsymbol{x}_{P_k})$ takes place by utilizing the objective function *f* with respect to swarm *k.* A problem often encountered by most PSO-based schemes is stagnation, which is related to the problem of particles being trapped in suboptimal solutions during the optimization process. In the case of CPSO where multiple swarms are used, the impairing effects caused by stagnation can be magnified [96]. This phenomenon, which limits the space exploration capabilities of the particles, can be tackled by introducing a resetting mechanism. The mechanism is activated when one of the swarms converges into a small region of the search space, while the global best position for one or more of the complementary swarms' changes. When these two conditions are fulfilled simultaneously, the particles of the stagnating swarm are reset to new random positions, thus allowing it to discover new areas

Algorithm 1 Cooperative particle swarm optimization algorithm

```
s: Swarm size population
Iter: Maximum number of iterations
c1, c2 w, $P_k$, $v_{max}^k$: PSO configuration parameters
1: Initialize the particles $x_i{}^k$ for all swarms $k$ at random positions
2:Calculate fitness $f(\boldsymbol{x}_{P_k})$ and set global bests for all swarms $\hat{y}^k(0)$, $k$=1, 2, …, $N_c$
3:   For = 1: Iter Do:
4:     For $k$ = 1: $N_c$
5:         If stagnation criterion is met for kth swarm, Then
6:             reset particles $x_i{}^k(t)$
7:         End
8:         For =1: s Do:
9:             Calculate fitness $f(\boldsymbol{u}_{P_k})$ and $y_i{}^k(t)$
10:        End for
11:        Calculate global best $\hat{y}^k(t)$ for swarm $k$
12:        For $i$ =1: s Do:
13:            For $j$ =1: $N_{P_k}$ Do:
14:              Update velocity $v_{ij}^k(t+1)$
15:              Update particle's position $x_{ij}^k(t+1)$
16:          End for
17:        End for
18:    End for
19:  End for
```

of the search space, which may be better suited to the new global best positions achieved by the other swarms. More details on the implementation of the resetting mechanism in CPSO schemes can be found in [96]. The resetting mechanism is expected to increase the effectiveness of the method, since the optimization problem contains multiple local minima.

The pseudocode for the proposed CPSO framework is given in Algorithm 1.

2.4.1 A CPSO based framework for optimization and control of photovoltaic heavy distribution networks

The goal of the proposed approach is the optimization and control of photovoltaic heavy distribution networks. by employing multiple swarms that can optimize different DN zones. Specifically, a reactive power optimization method for PV-heavy DNs based on CPSO is introduced. The proposed scheme employs multiple swarms to optimize different zones of the DN, where each zone contains design variables that are interrelated with respect to the optimization objective. In order to assign efficiently the design variables to the different swarms, a technique is used based on the Girvan-Newman community detection algorithm [125]. The proposed method is assessed for voltage deviation minimization, as well as the minimization of real power losses for an IEEE distribution grid, under various load and generation profiles.

2.4.1.1 Basic concepts and optimization problem formulation

The main task within the ORPF problem is to ensure that the bus voltage magnitudes stay within operational limits. This is especially needed in PV-heavy distribution grids where cloud coverage can obscure specific grid areas, thus rendering traditional control means with a grid-wide effect unsuitable. The communication infrastructure of the smart grid paradigm has allowed the inclusion of PV inverters as reactive power control devices.

Within the context of PV-heavy smart distribution grids featuring controllable inverters, a challenging problem revolves around the optimization of voltage deviations and real power losses. This optimization task focuses on adjusting a set of control variables, while adhering to several constraints. The control variables, denoted by u, signify the active power curtailment (APC) and reactive power injection for individual PV installations. A detailed presentation of the voltage deviation minimization problem is beyond the scope of this thesis and more details can be found in the original publication [126].

Two distinct optimization objectives are examined: one aims to reduce voltage deviations from a predefined nominal value at each buts, while the other focuses on minimizing real power losses throughout the system. The control variable vector u contains the APC and reactive power injection Q of each PV installation:

$$\boldsymbol{u} = \left[\boldsymbol{Q_1}, \boldsymbol{APC_1}, \boldsymbol{Q_2}, \boldsymbol{APC_2}, \dots \boldsymbol{Q_{N_{PV},}} \boldsymbol{APC_{N_{PV}}}\right] \tag{20}$$

In summary, the described problem is a non-convex constrained optimization challenge falling within the NP-hard category. The inclusion of a realistic modeling of the inverter's reactive power compensation (RPC) capability as a function of its active power curtailment percentages APC (as opposed to treating it as a bounded variable) adds to the total constraint complexity, while the fairly large number of design variables (2 per PV installation) contribute to high dimensionality.

2.4.1.2 Cooperative PSO for partitioned electrical networks

Since the objective of interest is to minimize the voltage deviations of the grid by optimizing the active and reactive power of distributed generators, a strategy is proposed based on portioning the grid according to its voltage sensitivity. This approach draws inspiration from the Girvan and Newman algorithm, a community detection method for complex networks that doesn't require a predefined number of communities (hereby referred to as "partitions") [127]. The algorithm accepts a weighted adjacency matrix (or edge-weight matrix) that corresponds to the undirected graph structure of the electrical grid and generates the optimal partitions $C^* = \left\{C_1^*, C_2^*, \dots, C_{N_c}^*\right\}$. A detailed presentation of the Girvan and Newman algorithm is beyond the scope of this thesis and more details can be found in the original publication [127].

The Girvan-Newman algorithm can be applied to the community detection problem, in order to yield the optimal partitions. These partitions will contain buses that are highly coupled among themselves with respect to voltage fluctuations. This information, obtained from the Girvan Newman algorithm, assists in organizing PV devices within highly interconnected network zones, enhancing smart grid voltage optimization. By utilizing information about the underlying electrical dynamics of the DN, the Girvan Newman algorithm yields the voltage-decoupled zones C^*, which in turn indicate the swarms P_k, $k = 1, 2, \dots, Nc$, where N_c is the total number of swarms:

$$\boldsymbol{P_k} = \left\{\boldsymbol{Q_1}, \boldsymbol{APC_1}, \boldsymbol{Q_2}, \boldsymbol{APC_2}, \dots, \boldsymbol{Q_{N_{P_k}}}, \boldsymbol{APC_{N_{P_k}}}\right\} \tag{21}$$

where, N_{P_k} is the total number of PV installations that reside in network partition C_i^* (it is assumed that $N_{P_k} \geq 1$ for every C_i^*). For each swarm P_k, the particle position $x_{ij}^k(t)$ and velocity $v_{ij}^k(t)$ are updated according to equations (17)-(18).

Moreover, as each swarm P_k contains a distinctive part of the original $\boldsymbol{u}$ vector (**20**), the right cooperation between the swarms' agents is essential in order to calculate the fitness function for the overall optimization problem. This task is feasible by utilizing a context vector $\boldsymbol{u}_{P_k}$ formed by linking each of the particle positions of swarm *P_k* with the global best positions of the remaining swarms:

$$\boldsymbol{u}_{\boldsymbol{P}_{\boldsymbol{k}}} = \left[\, \hat{\boldsymbol{y}}^{\boldsymbol{1}}, \dots, \hat{\boldsymbol{y}}^{\boldsymbol{k-1}}, \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{k}}\,, \hat{\boldsymbol{y}}^{\boldsymbol{k+1}}, \dots, \hat{\boldsymbol{y}}^{\boldsymbol{Nc}} \right] \tag{22}$$

The cooperative PSO approach presents three important features with respect to the ORPF problem. Firstly, the fitness function is evaluated after updating each part of the solution vector that corresponds to the respective swarm particles, resulting in finer-grained credit assignment. This addresses the classic "two steps forward - one step back" problem often encountered by PSO schemes, where a solution vector update improves one part of the solution vector but impairs another. This phenomenon is especially evident for the case of a network with voltage decoupled zones, where a part of a solution vector that corresponds to a specific zone may quickly converge to an unsatisfactory local minimum, thus limiting the whole swarm's exploration capabilities throughout the search space.

The second advantage is related to the increase in the solution diversity offered by CPSO [121]. Indeed, in the CPSO case, each solution vector becomes a combination of several particles which belong to different swarms; thus, the overall solution diversity is increased, because different personal and global best solutions are used for updating each particle, depending on the swarm it belongs to.

Lastly, the third advantage refers to the robustness of the algorithm, even when applied in networks with weakly voltage-decoupled zones; CPSO consolidates the partitioned design variables in one design vector at the end of every iteration, thus taking into account any inter-zonal effects.

2.4.1.3 Setup

The IEEE 123-bus distribution system [128] is selected as a suitable testbed for the simulation studies. Its large scale can accommodate a high number of PV installations, which warrants the application of cooperative optimization methods. As described in the previous section, the network partitions are determined through the utilization of the Girvan-Newman algorithm. A large number of PVs are considered in order to create a high dimensional search space; therefore, 20 installations are placed throughout the grid, spanning capacities from 140-280 kW; their technical specifications are shown in Table 1. These capacities represent a typical PV penetration level of a Greek mainland distribution grid, and were chosen as so. Each inverter has a nominal power that is +10% higher than its corresponding installed PV capacity, as is usual practice. The inverters can curtail the generated PV power and control the power factor of the injected power in the grid.

To simulate different DN states, two different scenarios are created. Both are regarded as snapshot scenarios (i.e., static), and are used in order to infer statistical conclusions for the performance of the proposed method. The scenarios are outlined below:

Scenario 1 is used to assess performance for the VDM objective and represents the phenomenon of partial cloudiness, resulting in severe undervoltage. The main challenge in this scenario is to bring the voltage of each bus as close as possible to the nominal value. The scenario information is shown on Table 2.

Scenario 2 is employed to evaluate the performance of the RPLM objective assuming full solar irradiance, which results in overvoltage in certain buses of the grid. In this particular

Table 1: PV installation specifications

#	Bus	PV Capacity (kW)	Power Rating (kVA)	Zone	#	Bus	PV Capacity (kW)	Power Rating (kVA)	Zone
1	6	140	155	C_1	11	100	280	310	C_3
2	10	140	155	C_1	12	119	280	310	C_3
3	117	140	155	C_1	13	109	280	310	C_3
4	27	180	200	C_2	14	111	280	310	C_3
5	26	180	200	C_2	15	78	280	310	C_4
6	41	180	200	C_2	16	88	280	310	C_4
7	45	180	200	C_2	17	92	280	310	C_4
8	50	180	200	C_2	18	82	280	310	C_4
9	55	140	155	C_1	19	21	180	200	C_2
10	68	280	310	C_3	20	63	280	310	C_3

scenario, the primary challenge involves ensuring that the voltage at each bus within the network adheres to operational constraints, while simultaneously minimizing power losses. Table 3 presents detailed information regarding the scenario.

In order to illustrate the effectiveness of the proposed method, three competing schemes are introduced: the first scheme is a randomly partitioned CPSO scheme (RPCPSO), where the respective swarms are not assigned according to the Girvan-Newman partition, but randomly. The second scheme, which has been proposed in study [129], formulates a decentralized optimization problem for each network partition, and solves each problem independently using a PSO algorithm with adaptive weights. The scheme is hereby referred to as "decentralized PSO" (dPSO) in this study, and its main difference in comparison to the proposed method is the decentralization of zones, namely the absence of inter-zone communication. The third scheme applies a standard centralized PSO algorithm [130] to the original problem.

The tuning parameters for each one of the competing schemes are shown in Table 3; they were selected based on indicative values found in the literature [131]–[133], in conjunction with a trial-and-error procedure. In order to ensure fairness of comparison, the standard

Table 2: Snapshot scenario information

Scenario	Slack bus nominal voltage (p.u.)	Average irradiance percent per zone				Load multiplier per zone			
		C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4
1	1.00	100	80	50	50	1	1	1	1
2	1.02	100	100	100	100	1.6	1	0.4	0.4

Table 3: Tuning parameters for all methods

Scheme	Swarm size	Stall iterations	Coefficients c_1, c_2	Function Tolerance	Inertia type	Inertia range
RPCPSO / CPSO	30	40	1.2	10^{-6}	Exponential	[1, 0.75]
PSO	90	40	1.49	10^{-6}	Adaptive	[1.1, 0.1]
dPSO	30	40	1.49	10^{-6}	Adaptive	[1.1, 0.1]

Table 4: Scenario 1: Statistical results for VDM objective

	Objective value average	**Objective value standard deviation**	**Best Objective value**	***p*-value**	**Average function evaluations**[1]
CPSO	0.7006	0.0603	0.6241	-	5260
RPCPSO	0.8904	0.1482	0.7574	1.67E-04	5120
PSO	0.9792	0.2746	0.7810	1.89E-04	4960
dPSO	2.2109	0.2295	1.7594	1.71E-25	6520

[1] convergence to the 1st decimal

Table 5: Scenario 2: Statistical results for RPLM objective

	Objective value average	**Objective value standard deviation**	**Best Objective value**	***p*-value**	**Average function evaluations**[2]
CPSO	0.01035	0.00072	0.00918	-	8520
RPCPSO	0.01164	0.00173	0.00961	1.67E-02	8340
PSO	0.01230	0.00202	0.01027	2.23E-05	7120
dPSO	0.04307	0.00230	0.04027	5.543E-37	9240

[2] convergence to the 3rd decimal

PSO algorithm was allowed a larger swarm size, namely three times the one used by the rest of the algorithms; this way, the number of total objective function evaluations across all methods was similar, and their performance comparable.

2.4.1.4 Results and discussion

Since the three competing schemes are based on stochastic search, multiple runs are needed to properly assess their performance. To be more specific, a total number of 20 runs for each scenario is performed, starting from different randomly chosen initial particle positions in each run. In order to reach valid conclusions regarding the statistical superiority of the proposed scheme, a *t*-test between CPSO and each one of its rivals has been applied for scenarios 1-2.

The null hypothesis is that the results produced by the two competing methodologies are generated by populations with the same mean. Tables 4-5 depicts the average and standard deviation values, as well as the best value for the objective function from the 20 runs,

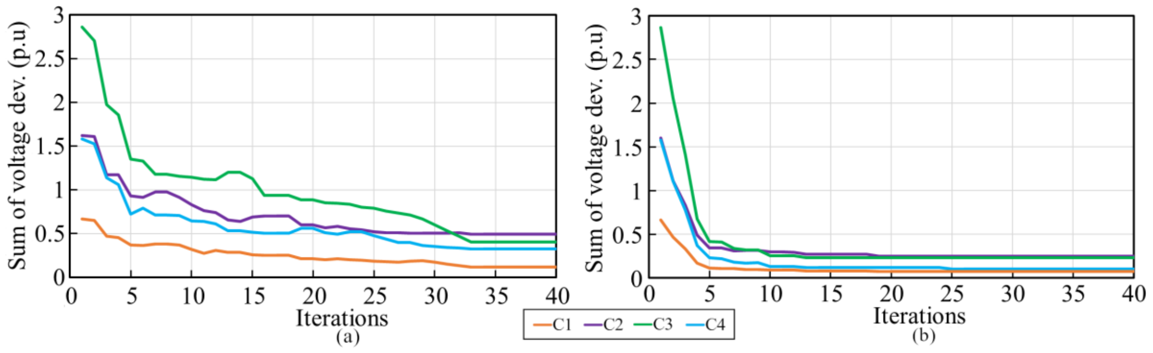

Figure 3: Convergence curves for each individual network zone for the best PSO (a) and CPSO (b) runs on scenario 1.

together with the *p*-value corresponding to the *t*-test and the average number of function evaluations of each method for scenarios 1 and 2, respectively. Regarding scenario 1, CPSO achieves 21%, 28% and 68% lower average objective value (sum of voltage deviations) in comparison to RPCPSO, PSO and dPSO, respectively. Similar performance is recorded for scenario 2 where the objective of RPLM (total power losses in MW) is addressed: an 11%, 16% and 76% improvement is achieved over RPCPSO, PSO and dPSO, respectively. The superiority of CPSO is also confirmed when comparing the best runs of each method: in scenario 1, CPSO achieves a 17%, 20% and 64% improvement over RPCPSO, PSO and dPSO, respectively, while in scenario 2, CPSO scores 5%, 11% and 77% improvements over RPCPSO, PSO and dPSO, respectively. The statistical significance of these results is established by the *t*-test with a confidence interval of over 98%, as indicated by the produced *p*-values.

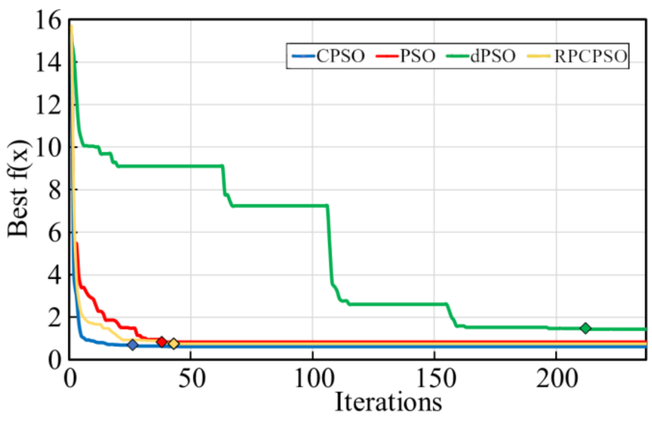

Figure 4: Convergence graph for the best runs of scenario 1. Coloured rectangles denote convergence to the 1st decimal. Note that the convergence curve of dPSO corresponds to the successive minimization of the four zone-based optimization problems

In addition, the superior performance of the proposed method is consistent, i.e., the method converges around the same solution for each different run. This is indicated by the low value of standard deviation for both scenarios, and is testimony to the increased search space exploitation capabilities inherent to the CPSO algorithm. Lastly, it should be emphasized that the proposed method achieves the aforementioned performance improvements with a reasonable computational burden, as indicated by the number of average objective function evaluations: when compared to dPSO, CPSO exhibits consistently lower computational requirements in both scenarios. It should be noted that CPSO is surpassed by standard PSO in this aspect, but on the other hand it manages to greatly outperform the latter in terms of optimization performance.

There exist multiple reasons for the superior statistical performance of CPSO in scenarios 1 & 2. First of all, since the RPCPSO approach has randomly assigned each design variable in the cooperative swarms, the underlying topology of the network is not utilized. Therefore, it fails to properly explore the available search space, in contrast to the proposed CPSO approach where the network partitioning information is incorporated. Next, as mentioned earlier, standard PSO suffers from the "two steps forward - one step back" problem and cannot effectively explore the available search space. This can be shown by examining the convergence curves of each individual zone for PSO and CPSO in an example run, as illustrated in Figure 3. Here (fig. 3a), it is confirmed that zone convergence for PSO is not monotonic, meaning that some updates may improve some parts of the solution vector, but worsen others, as reflected in the performance of each zone; this phenomenon is especially highlighted at iterations 10-15, where improvement of zone C_3 is temporarily impaired. CPSO's zone-based approach alleviates this problem as shown by the monotonicity of zone convergence (fig. 3b), and is further confirmed by examining the overall convergence characteristics, as shown Figure 4: Here, considering the first 30 iterations, CPSO achieves a rapid improvement in objective value, in contrast to PSO, which appears to stall multiple times over the same period. Moreover, CPSO achieves superior exploitation characteristics, since it converges to the 1st decimal much earlier than PSO. The dPSO approach also sufficiently exploits the search space of each of the four zone-based optimization problems, which appear as distinct "steps" on the convergence curve; the quick convergence of each problem to the 1st decimal this observation. It should be noted that, one hand, dPSO retains the important practical advantage of complete decentralization [134], which CPSO and PSO lack. On the other hand, dPSO exhibits worse exploration capabilities out of the other three schemes. This can be attributed to the fact that as the algorithm progresses from one zone-based problem to the next, it cannot account for inter-zone effects, which are strong for the test case selected in this work. The result is an overall deterioration of optimization performance.

2.4.2 A CPSO based framework for parameter identification in wastewater treatment plant modelling

The objective of this approach is to create an accurate parameter identification system for a WWTP using a modified BSM1 model. To achieve this, a customized CPSO framework is utilized in order to successfully estimate the values of critical parameters in a detailed first principles model of the process, by solving a nonlinear optimization problem which is based on a system identification scheme. The resulting model can subsequently be applied for energy-efficient optimization by integrating it into control schemes. The proposed CPSO is assessed through comparisons with two other PSO algorithms in the task of estimating the kinetic and stoichiometric coefficients of the modified BSM1 model.

2.4.2.1 Basic concepts and problem formulation

The BSM1 model [135] is a well-known first principles model that is being widely used to simulate the dynamic behavior of WWTPs. In this study, the generic BSM1 model was customized to match the structural and operational aspects of a real WWTP in Greece, with further refinement using real sensor data. The BSM1 model encompasses many critical parameters significantly affecting the entire process. Specifically, it includes 14 kinetic and 5 stoichiometric parameters, each one of them closely related to a certain chemical process.

The proper adaptation of BSM1 to a specific WWTP requires an accurate estimation of the parameters that exist in the model. To achieve this, a system identification scheme is proposed that is based on formulating and solving a nonlinear optimization problem. The objective of this optimization problem is to minimize the mean squared error (MSE) between the data collected from the actual plant and the simulated data produced by the adapted BSM1 model, as described by the following equation:

$$\boldsymbol{J} = \frac{\boldsymbol{1}}{\boldsymbol{N}} \cdot \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} (\boldsymbol{y_r} - \boldsymbol{\hat{y}})^{\boldsymbol{2}} \tag{23}$$

where *N* corresponds to the total amount of data collected, y_r represents the data collected from the actual plant and $\hat{y}$ denotes the simulated data produced by the adapted model.

The formulated nonlinear optimization problem aims to minimize equation (**23**) by selecting the appropriate values of the aforementioned parameters, which are present in the modified BSM1 model. However, a number of undesired properties are observed in this problem, including high dimensionality and multimodality, rendering the use of standard optimization solvers inappropriate; thus, it is necessary to employ a solver capable of effectively handling such properties.

2.4.2.2 Cooperative PSO for system identification

Standard PSO could be used to solve this nonlinear, high-dimensional optimization problem. In this case, the design variables of the formulated optimization problem are grouped into one unique swarm, while the PSO solver concurrently seeks to accurately estimate their values. This particular property of the PSO approach, in conjunction with the high number of parameters associated with the particular optimization problem, significantly increases the difficulty of approaching a satisfying solution and thus of estimating the values of the adapted BSM1 model parameters accurately.

To overcome the problems that appear in the standard PSO methodology, a customized cooperative PSO [136] approach is designed for solving the formulated nonlinear optimization problem. In this customized CPSO solver, the design variables of the optimization problem are separated into several distinct swarms, which are formed by taking advantage of the correlations that exist between them. Each swarm, then, explores the search space independently trying to approach the best possible solution regarding its own design variables, while sharing information in terms of its position with the other swarms.

In this customized cooperative PSO methodology, each swarm includes only some of the design variables and in order to evaluate the objective function, the particles of each swarm are combined with the best particles of the other swarms. In this way, a context vector is created and the value of the objective function can be directly obtained. Moreover, the proposed cooperative approach for solving the formulated nonlinear optimization problem exhibits several advantages compared to the standard PSO algorithm. The CPSO solver optimizes the design variables separately by grouping them into distinct swarms, thus avoiding the two steps forward – one step back problem that appears in the PSO methodology. In addition to this, exploration of the search space by each swarm independently, while simultaneously exchanging information regarding their best positions with the rest of the swarms, significantly enhances the diversity of the final solution.

Remark: Once the CPSO-based identification is completed, the obtained BSM1 model can be used for different purposes involving optimization and control of WWTPs. One potential use is to conduct the necessary step tests on the identified BSM1 model, for developing step response predictive models in an economic dynamic matrix control (EDMC) scheme. The EDMC controller aims to minimize an objective function, optimizing energy consumption and operating costs while adhering to environmental constraints. Further elaboration on the EDMC controller is outside the scope of this thesis; interested readers can refer to the original publication [137] for additional details.

2.4.2.3 Set up

In order to deploy the developed identification scheme, two months of data have been collected by the BSM1 model using the nominal values of the parameters, as they are defined in [136]. The values of 12 state variables measured by specific sensors in the investigated plant, have been stored with a sampling period of 15 minutes, which is suitable for designing WWTP automatic control schemes [138]. Then, the data are provided to the CPSO solver in order to approximate the values of the 14 kinetic and 5 stoichiometric parameters that are present in the modified BSM1 model.

For comparison purposes, the employed CPSO solver was tested against standard PSO and a modified PSO methodology [139]. The algorithms are individually tuned using literature suggestions in conjunction with a trial-and-error procedure. Due to the inherent stochastic nature of the algorithms, 15 runs are executed for each algorithm and the superiority of CPSO is validated by running a two-sample t test.

2.4.2.4 Results and Discussion

As mentioned previously, the most important advantage of the proposed CPSO methodology for solving the nonlinear optimization problem is its ability to separate the design variables into several distinct swarms, by taking advantage of the correlations that exist between them. In particular, in the formulated optimization problem, the 8 parameters associated with the heterotrophic phenomena form the first swarm, while the 5 of them linked with the autotrophic phenomena comprise the second swarm. The 3 parameters affecting the hydrolysis procedure are included in the third swarm, and the unique parameter influencing the ammonification process forms the fourth swarm. Finally, the remaining 2 parameters that are closely related to the COD in the biomass of the plant are grouped into the fifth swarm.

Table 6 summarizes the results. As it can be seen, the performance of CPSO is superior in terms of the best or the average solution with a statistical significance higher than 90%, while it also manages to produce more consistent results, as indicated by the standard deviation values. The proposed CPSO approach manages to estimate the kinetic and stoichiometric coefficients of the modified BSM1 model with high accuracy and specifically accomplishes a value for the average objective function equal to $6.8 \cdot 10^{-4}$. In Figure 5a and 5b, the results of the identified model versus the original one are shown for the ammonia and the dissolved oxygen concentration of the last aerobic tank of the second line, respectively, while Figure 6 depicts the evolution of the 5 distinct swarms during the identification procedure. The presented results have been obtained using validation data that have not been used in the identification process. It is clear, that the identification procedure is successful, as it produces a highly accurate dynamic model. This result is attributed to CPSO, which manages to obtain an accurate estimation of the critical process parameters.

Table 6: Performance Metrics

Algorithm	**Fitness Average**	**Fitness standard deviation**	**Best Fitness**	**Average Function Evaluations**	**p-value**
CPSO	0.00068	0.00014	0.00041	2348	-
Standard PSO	0.00223	0.00244	0.00045	2726	0.0311
Modified PSO	0.00106	0.00076	0.00044	2535	0.0885

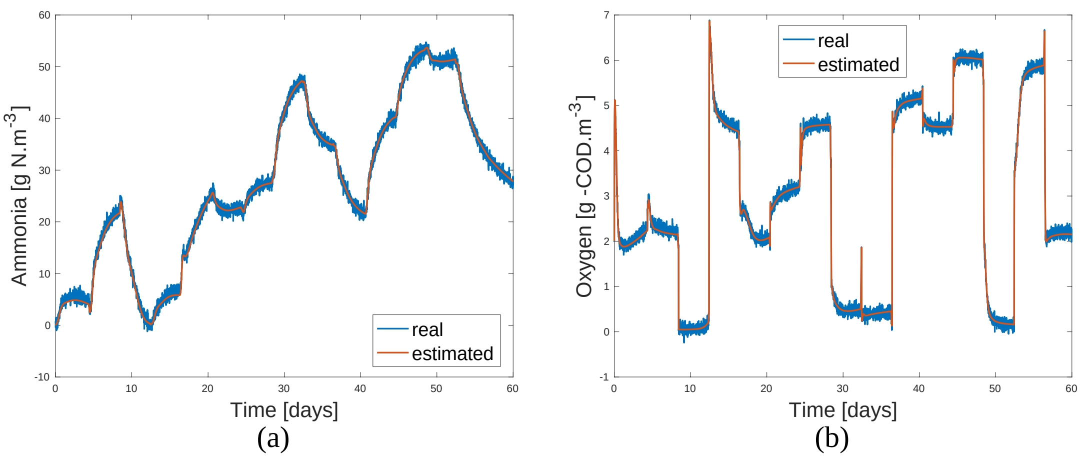

Figure 5: Results of the identified model for (a) Ammonia Concentration, (b) Dissolved Oxygen Concentration

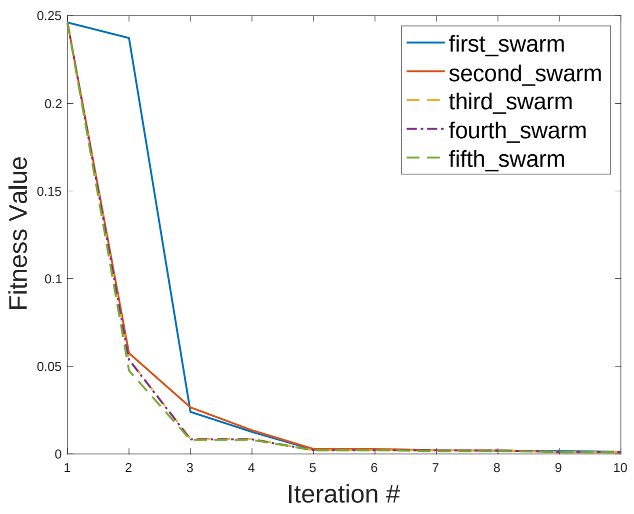

Figure 6: Evolution of the fitness values for each of the 5 swarms

3. Nonlinear control methods

3.1 Introduction to nonlinear control methods

In the field of control engineering, both linear and nonlinear control methods play pivotal roles in governing and manipulating dynamic systems in various fields, including engineering robotics, energy management and biological systems

Linear control methods [140] relying on linear system theory, accurately represent systems with minor deviations from equilibrium using linear equations. These methods, including PID and LQR controllers along with state-space control techniques, perform well for relatively simple systems. However, their limitations arise when dealing with complex systems exhibiting nonlinearity and uncertainty. In such real-world scenarios, linear control approaches may struggle in ensuring satisfactory performance and stability, thus underscoring the necessity of nonlinear control approaches.

Nonlinear control methods [141] on the other hand, are designed to handle systems that exhibit nonlinear behavior. These methods are essential because they recognize that the behavior of certain systems cannot be accurately described solely by linear equations. To address this limitation, they utilize sophisticated mathematical techniques capable of capturing the intricate nonlinear dynamics of the system.

Specifically, nonlinear control excels in handling complex systems [142] accurately modeling nonlinearities [143], and providing improved stability [144] particularly for systems with significant deviations from equilibrium. Its robustness against uncertainties [145] and the ability to adapt in system and environmental conditions make it an essential tool for controlling inherently nonlinear systems, chaotic systems, and other complex dynamic systems.

In the upcoming chapter, a concise yet comprehensive theoretical overview of notable nonlinear control methodologies will be presented.

3.2 Backstepping control method

Backstepping control is a prominent nonlinear method, gaining significant attention in the control literature [141], [146]. Unlike conventional control approaches, which struggle with highly nonlinear systems, backstepping provides a systematic and recursive strategy to stabilize such systems. Backstepping control is a recursive design technique that links the choice of a control Lyapunov function with the design of a feedback controller, guaranteeing global asymptomatic stability for strict feedback systems.

The core principle of backstepping involves breaking down the control problem into simpler task, each focusing on an individual state variable. Through the design of feedback control laws for each variable, backstepping guides the system's trajectory toward the desired state, ensuring overall stability and convergence.

3.2.1 Integrator Backstepping

Consider the special case of integrator backstepping given by the following control system:

$$\dot{\boldsymbol{\xi}} \quad = \quad \boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\eta} \tag{24}$$

$$\dot{\boldsymbol{\eta}} \quad = \quad \boldsymbol{u} \tag{25}$$

Where $[\xi^T, \eta^T] \in \mathbb{R}^{n+1}$ is the state and $u \in \mathbb{R}$ is the control input. The functions $\boldsymbol{f} : D \to \mathbb{R}^n$ and $g : D \to \mathbb{R}^n$ are smooth in a domain $D \subset \mathbb{R}^n$ that contains $\boldsymbol{\xi} = 0$ and $f(0) = 0$.

Theorem 1 (Backstepping theorem)

Considering the control system (24)-(25) *with smooth vector fields f and g with $f(0) = 0$ and $g(0) = 0$. Let $\eta = \varphi(\xi)$ be a stabilizing state feedback law for the subsystem* (24). *Consider that $\boldsymbol{V_1}(\xi)$ is Lyapunov function such that*

$$\frac{\partial \boldsymbol{V_1}}{\partial \boldsymbol{\xi}} \cdot [\boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\varphi}(\boldsymbol{\xi})] \quad \leq \quad -\boldsymbol{W}(\boldsymbol{\xi}) \tag{26}$$

where $W(\xi)$ is a positive definite. Then the state feedback control law given by:

$$\boldsymbol{u} = \frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{\xi}} \cdot [\boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\eta}] - \frac{\partial \boldsymbol{V_1}}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{g}(\boldsymbol{\xi}) - \boldsymbol{\kappa} \cdot [\boldsymbol{\eta} - \boldsymbol{\varphi}(\boldsymbol{\xi})], \qquad \boldsymbol{\kappa} > \boldsymbol{0} \tag{27}$$

stabilizes the origin $(\xi, \eta) = (0,0)$ of the system (24)-(25) *with the Lyapunov function given by:*

$$\boldsymbol{V_2}(\boldsymbol{\xi}, \boldsymbol{\eta}) \quad = \quad \boldsymbol{V_1}(\boldsymbol{\xi}) + \frac{\boldsymbol{1}}{\boldsymbol{2}} \cdot [\boldsymbol{\eta} - \boldsymbol{\varphi}(\boldsymbol{\xi})]^{\boldsymbol{2}} \tag{28}$$

Proof: The goal is to design a state feedback control law $\boldsymbol{u}$ that stabilizes the origin ($\boldsymbol{\xi} = 0$, $\eta = 0$). We assume that the functions f and g are known. We also assume $f(0) = 0$ and $g(0) = 0$.

Figure 7 depicts the block diagram of control system (24)-(25). This system can be seen as a cascade connection of two component parts of which the first is an integrator. By focusing in the system (24), we consider variable $\boldsymbol{\eta}$ as a virtual controller input. We suppose that a smooth feedback control law exists in the form $\eta = \varphi(\xi)$ with $\varphi(0) = 0$; such that the origin of

$$\dot{\boldsymbol{\xi}} \quad = \quad \boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\varphi}(\boldsymbol{\xi}) \tag{29}$$

is asymptotically stable.

We assume that we know a Lyapunov function $V_1(\xi)$ that satisfies the inequality

$$\frac{\partial \boldsymbol{V_1}}{\partial \boldsymbol{\xi}} \cdot [\boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\varphi}(\boldsymbol{\xi})] \quad \leq \quad -\boldsymbol{W}(\boldsymbol{\xi}) \tag{30}$$

where $W(\xi)$ is positive definite. By introducing the terms $g(\xi) \cdot \varphi(\xi)$ on the right-hand side of equation (24) we can rewrite the control system (24)-(25) in the following manner:

$$\dot{\boldsymbol{\xi}} \quad = \quad [\boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\varphi}(\boldsymbol{\xi})] + \boldsymbol{g}(\boldsymbol{\xi}) \cdot [\boldsymbol{\eta} - \boldsymbol{\varphi}(\boldsymbol{\xi})] \tag{31}$$

$$\dot{\boldsymbol{\eta}} \quad = \quad \boldsymbol{u} \tag{32}$$

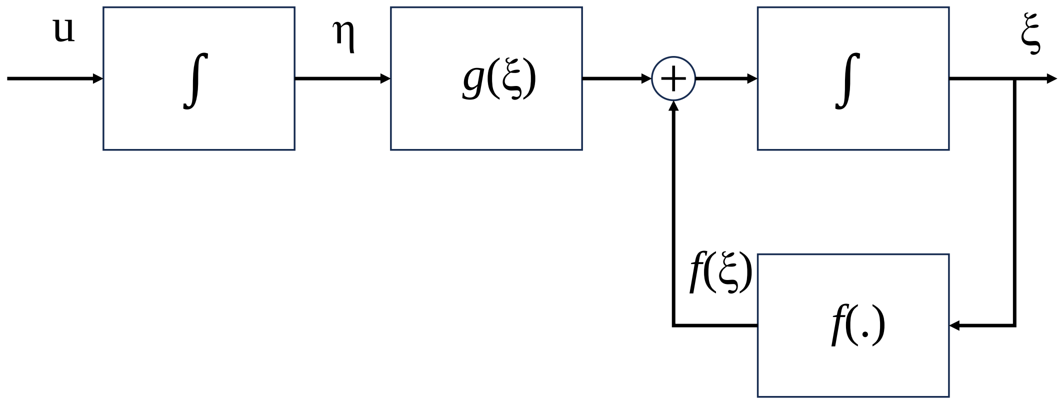

Figure 7: Backstepping control design for the system (24)-(25)

We now proceed to introduce the change of variables:

$$\boldsymbol{z} = \boldsymbol{\eta} - \boldsymbol{\varphi}(\boldsymbol{\xi}) \tag{33}$$

The output z can be viewed as the error between the state η and the virtual control $\varphi(\xi)$. The change in variables results in the following system:

$$\dot{\boldsymbol{\xi}} \quad = \quad [\boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\varphi}(\boldsymbol{\xi})] + \boldsymbol{g}(\boldsymbol{\xi}) \cdot [\boldsymbol{\eta} - \boldsymbol{\varphi}(\boldsymbol{\xi})] \tag{34}$$

$$\dot{\boldsymbol{z}} \quad = \quad \boldsymbol{u} - \dot{\boldsymbol{\varphi}}(\boldsymbol{\xi}) \tag{35}$$

Since f, g and φ are known, the derivative $\dot{\varphi}(\xi)$ can be calculated the following equation:

$$\dot{\boldsymbol{\varphi}}(\boldsymbol{\xi}) \quad = \quad \frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{\xi}} \cdot [\boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\eta}] \tag{36}$$

Now by taking $v = u - \dot{\varphi}(\xi)$ the transformed system (34)-(35) is reduced to the below cascade connection:

$$\dot{\boldsymbol{\xi}} \quad = \quad [\boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\varphi}(\boldsymbol{\xi})] + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{z} \tag{37}$$

$$\dot{\boldsymbol{z}} \quad = \quad \boldsymbol{v} \tag{38}$$

The new transformed system given by equations (37)-(38) has the same form as the starting control system (24)-(25), despite the fact that the first subsystem has an asymptotically stable origin when the input is zero.

Next, we consider as a candidate Lyapunov function the following positive definite function:

$$\boldsymbol{V_2}(\boldsymbol{\xi}, \boldsymbol{\eta}) \quad = \quad \boldsymbol{V_1}(\boldsymbol{\xi}) + \frac{\mathbf{1}}{\mathbf{2}} \cdot \boldsymbol{z}^{\mathbf{2}} \tag{39}$$

where $V_1(\xi)$ is a positive definite function.

By calculating the derivative of the candidate Lyapunov function $V_2(\xi, \eta)$ we obtain the following expression:

$$\begin{aligned} \dot{\boldsymbol{V}}_{\boldsymbol{2}}(\boldsymbol{\xi},\boldsymbol{\eta}) &= \frac{\partial \boldsymbol{V_1}}{\partial \boldsymbol{\xi}} \cdot [\boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\varphi}(\boldsymbol{\xi})] + \frac{\partial \boldsymbol{V_1}}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{z} + \boldsymbol{z} \cdot \boldsymbol{v} \\ &\leq -\boldsymbol{W}(\boldsymbol{\xi}) + \frac{\partial \boldsymbol{V_1}}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{z} + \boldsymbol{z} \cdot \boldsymbol{v} \end{aligned} \tag{40}$$

We choose the backstepping control v as

$$\boldsymbol{v} = -\frac{\partial \boldsymbol{V_1}}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{g}(\boldsymbol{\xi}) - \boldsymbol{\kappa} \cdot \boldsymbol{z}, \quad \boldsymbol{\kappa} > \boldsymbol{0} \tag{41}$$

By substituting (41) into (40), we obtain

$$\dot{\boldsymbol{V}}_{\boldsymbol{2}}(\boldsymbol{\xi},\boldsymbol{\eta}) \leq -\boldsymbol{W}(\boldsymbol{\xi}) - \boldsymbol{\kappa} \cdot \boldsymbol{z}^{\boldsymbol{2}} \tag{42}$$

The above result shows that the origin $(\xi = 0,\ z = 0)$ is asymptotically stable. Since $\varphi(0) = 0$, we can conclude that the origin $(\xi = 0,\ \eta = 0)$ is asymptotically stable for the original system (24)-(25).

Now by considering the equation: $v = u - \dot{\varphi}\ (\xi)$, and by substituting for $v,\ z$ and $\dot{\varphi}$, the following state feedback control law is obtained:

$$\boldsymbol{u} = \frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{\xi}} \cdot [\boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\eta}] - \frac{\partial \boldsymbol{V_1}}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{g}(\boldsymbol{\xi}) - \boldsymbol{\kappa} \cdot [\boldsymbol{\eta} - \boldsymbol{\varphi}(\boldsymbol{\xi})] \tag{43}$$

This concludes the proof of Theorem 1. ■

3.2.2 General system Backstepping case

Next, we move from the system (24)-(25) to the more general system of the form:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\eta} \tag{44}$$

$$\dot{\boldsymbol{\eta}} = \boldsymbol{f}_{\boldsymbol{\alpha}}(\boldsymbol{\xi},\boldsymbol{\eta}) + \boldsymbol{g}_{\boldsymbol{a}}(\boldsymbol{\xi},\boldsymbol{\eta}) \cdot \boldsymbol{u} \tag{45}$$

where f_α and g_a are smooth. If $g_a(\xi,\eta)$ ≠0, the input transformation

$$\boldsymbol{u} = \frac{\boldsymbol{1}}{\boldsymbol{g}_{\boldsymbol{a}}(\boldsymbol{\xi},\boldsymbol{\eta})} \cdot [\boldsymbol{u}_{\boldsymbol{a}} - \boldsymbol{f}_{\boldsymbol{\alpha}}(\boldsymbol{\xi},\boldsymbol{\eta})] \tag{46}$$

will yield (52)-(53) to the following control system:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\eta} \tag{47}$$

$$\dot{\boldsymbol{\eta}} = \boldsymbol{u_a} \tag{48}$$

Thus, if a stabilizable state feedback control law $\varphi(\xi)$ and a Lyapunov function $\mathrm{V}(\xi)$ exist such that the conditions of Theorem 1 are satisfied for (44) then the Theorem and (46) yield

$$\begin{aligned}\boldsymbol{u} = \frac{\mathbf{1}}{\boldsymbol{g_a}(\boldsymbol{\xi},\boldsymbol{\eta})} \cdot \{\frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{\xi}} \cdot [\boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi}) \cdot \boldsymbol{\eta}] - \frac{\partial \boldsymbol{V_1}}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{g}(\boldsymbol{\xi}) - \boldsymbol{\kappa} \cdot [\boldsymbol{\eta} - \boldsymbol{\varphi}(\boldsymbol{\xi})] \\ - \boldsymbol{f_\alpha}(\boldsymbol{\xi},\boldsymbol{\eta})\}\end{aligned} \tag{49}$$

for some $\kappa > 0$ and

$$\boldsymbol{V_2}(\boldsymbol{\xi},\boldsymbol{\eta}) \quad = \quad \boldsymbol{V_1}(\boldsymbol{\xi}) + \frac{\mathbf{1}}{\mathbf{2}} \cdot [\boldsymbol{\eta} - \boldsymbol{\varphi}(\boldsymbol{\xi})]^2 \tag{50}$$

as respectively a stabilizing state feedback control law and a candidate Lyapunov function for the control system (44)-(45).

By using the above backstepping methodology recursively we can stabilize systems that are in strict feedback form given as follows:

$$\begin{aligned}\dot{\boldsymbol{\xi}} &= \boldsymbol{f_0}(\boldsymbol{\xi}) + \boldsymbol{g_0}(\boldsymbol{\xi}) \cdot \boldsymbol{\eta_1} \\ \dot{\boldsymbol{\eta}}_\mathbf{1} &= \boldsymbol{f_1}(\boldsymbol{\xi},\boldsymbol{\eta_1}) + \boldsymbol{g_1}(\boldsymbol{\xi},\boldsymbol{\eta_1}) \cdot \boldsymbol{\eta_2} \\ \dot{\boldsymbol{\eta}}_\mathbf{2} &= \boldsymbol{f_2}(\boldsymbol{\xi},\boldsymbol{\eta_1},\boldsymbol{\eta_2}) + \boldsymbol{g_2}(\boldsymbol{\xi},\boldsymbol{\eta_1},\boldsymbol{\eta_2}) \cdot \boldsymbol{\eta_3} \\ &\vdots \quad \vdots \quad \vdots \\ \dot{\boldsymbol{\eta}}_{\boldsymbol{k-1}} &= \boldsymbol{f_{k-1}}(\boldsymbol{\xi},\boldsymbol{\eta_1},\dots,\boldsymbol{\eta_{k-1}}) + \boldsymbol{g_{k-1}}(\boldsymbol{\xi},\boldsymbol{\eta_1},\dots,\boldsymbol{\eta_{k-1}}) \cdot \boldsymbol{\eta_k} \\ \dot{\boldsymbol{\eta}}_{\boldsymbol{k}} &= \boldsymbol{f_k}(\boldsymbol{\xi},\boldsymbol{\eta_1},\dots,\boldsymbol{\eta_k}) + \boldsymbol{g_k}(\boldsymbol{\xi},\boldsymbol{\eta_1},\dots,\boldsymbol{\eta_k}) \cdot \boldsymbol{u}\end{aligned} \tag{51}$$

where $\xi \in \mathbb{R}^n$, η_1 to η_k are scalars, and f_0 to f_k vanish at the origin.

By approaching the control system (**51**) with the backstepping control design procedure, an overall stabilizing state feedback $u = \varphi_k(\xi,\eta_1,\dots,\eta_k)$ along with a Lyapunov function $V_k(\xi,\eta_1,\dots,\eta_k)$ can be obtained, thus stabilizing the equilibrium of the system.

3.3 Model Predictive Control

Model predictive control is an advanced control strategy widely applied in engineering and industry to optimize the performance of complex systems. MPC is a model-based control technique that takes into account the dynamic behavior of the system, predicting its future response based on a mathematical model. By making predictions about future system behavior, the MPC method can intelligently selects control actions to steer the controlled system toward desired outcomes, while satisfying constraints.

3.3.1 Introduction to Model Predictive Control

Predictive control has been a part of automatic control systems since the early 1980s [147], finding extensive applications in both process industries and academia. The key factor driving its widespread adoption is its unique capability to predict future outputs and optimize the current timeframe effectively.

The fundamental concept underlying MPC control involves predicting future outputs of a dynamic model by incorporating current measurements and system models. These predictions are generated by making appropriate adjustments to the input variables.

MPC controllers offer several significant advantages, including:

1. Handling MIMO systems: Is well-suited for controlling complex systems with multiple inputs and outputs, making it versatile in various industrial applications.
2. Dealing with constraints: Can effectively accommodate constraints on various system variables, ensuring safe and feasible operation within defined limits.
3. Applicability to different processes: MPC strategy is adaptable to both linear and nonlinear processes, making it an ideal choice for diverse control scenarios.
4. Ease of implementation: MPC controllers offer a user-friendly implementation, making them easily accessible to control engineers and personnel responsible for system operation.

MPC is a broad term that encompasses various control methodologies, all sharing a common feature: the utilization of a system model to determine optimal control actions, through an optimization process. The methodology employed by an MPC-type controller is depicted in a simplified manner for the case of a single-input single-output (SISO) system in Figure 8. This process can be broadly analyzed and broken down into the following steps:

1. At each time step k the predictions about the future plant outputs are calculated based on a model of the system for a predetermined horizon H_p. The prediction

horizon H_p is the number of the predicted future time steps and shows how far the controller predicts in to the future. The predicted outputs $\hat{y}(k+i)$ where $i = 1, \dots, H_p$ are based on the past input outputs of the system and the future control moves $u(k+i-1)$, $i = 1, \dots, H_c - 1$.

2. The number of control moves to until the future time step $H_c - 1$ is called control horizon and is symbolized by H_c. Inside the control horizon, the control moves change, while outside they remain constant. The control horizon moves u are estimated by optimizing a cost function J. The cost function that is often selected is a summation of quadratic errors in order to minimize the difference between the predicted future outputs and the desired given set points. The control horizon is chosen always smaller than the prediction horizon. Usually only the first couple of control moves have a significant effect on the predicted output behavior.
3. After the estimation of control moves $u(k+i-1)$ for $i = 1, \dots, H_c - 1$at time instant, only the first one $u(k)$ is implemented in the system and the remaining ones

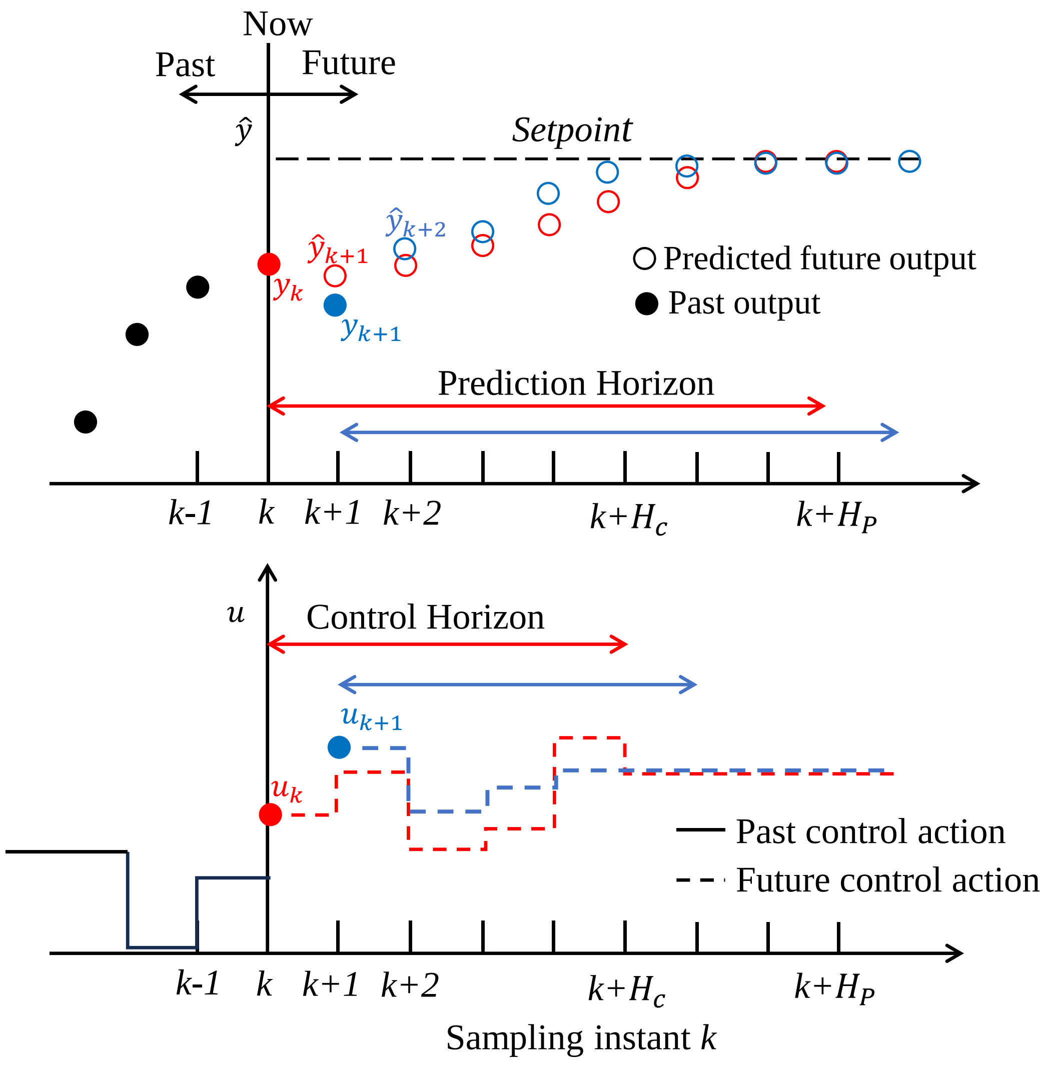

Figure 8: MPC methodology for a SISO system

are rejected. After applying $u(k)$, the real desired output $y(k)$ is calculated. Now the prediction and the control horizon shift forward by one time step and the MPC controller repeats the same cycle of calculations to compute the optimal u for the next time step $k+1$.

Because of the forward moving nature of the prediction horizon, the MPC method is also referred to as receding horizon control.

3.3.2 Nonlinear Tracking MPC

As already outlined in the introductory chapter 3, the concept behind the NMPC scheme is as follows: at each sampling instant, we optimize the predicted future behavior of the system over a finite time horizon $k = 1, \dots, N_p$ with $N_p \geq 1$. Then, we utilize the first element of the obtained optimal control sequence as a feedback control value for the subsequent sampling interval. In the section, a mathematical description of this basic idea is provided for the constant reference case x_{ref}.

Consider the following nonlinear discrete-time dynamical system:

$$\boldsymbol{x}(\boldsymbol{k}+\mathbf{1}) = \boldsymbol{f}(\boldsymbol{x}(\boldsymbol{k}), \boldsymbol{u}(\boldsymbol{k})) \tag{52}$$

where $u(k) \in U$ represents the input vector and $x(k) \in X$ the state vector. In order to find a feedback control law which stabilizes the system at x_{ref} is to have x_{ref} being equilibrium of the nominal closed system:

$$\boldsymbol{x}(\boldsymbol{k}+\mathbf{1}) = \boldsymbol{f}(\boldsymbol{x}(\boldsymbol{k}), \boldsymbol{\mu}(\boldsymbol{x}(\boldsymbol{k}))) \tag{53}$$

where $\mu(x(k))$ is the NMPC-feedback law $\mu: X \to U$.

A necessary condition for this is that there exists a control value $u^* \in U$ with

$$\boldsymbol{x_{ref}} = \boldsymbol{f}(\boldsymbol{x_{ref}}, \boldsymbol{u}^*) \tag{54}$$

The cost function $l(x,u),\ l: X \times U \to \mathbb{R}_0^+$ employed in our optimization should be designed to penalize the deviation of the distance of an arbitrary state $\boldsymbol{x}$ from the reference state x_{ref}.

Another important requirement regarding the cost function is that when we are in the equilibrium x_{ref} and use the control value u^* the cost should 0, in order to stay on the equilibrium, while outside the cost should be positive, i.e.

$$\boldsymbol{l\big(x_{ref}\,, u^*\big) = 0 \; and \; l(x,u) > 0 \; for \; all \; x \in X, \in U \; with \; x \neq x_{ref}} \tag{55}$$

As already mentioned in subsection 3.3.1, one of the main reasons for the success of NMPC and MPC controllers in general is its ability to take constraints into consideration. In this context, we take into account constraints on both the control input and the system state. Thus, we defined a nonempty state constraint set $\mathbb{X} \subseteq X$ and for each $x \in X$ a nonempty control constraint set is introduced as $\mathbb{U} \subseteq U$. The idea behind these sets is that we want the trajectories to be inside set $\mathbb{X}$ while the control inputs lie in $\mathbb{U}$.

Having a cost function that satisfies (55) and a prediction horizon $N_p \geq 2$, the basic NMPC problem can be formulated. In the optimal control problem (OCP) a set of control sequences is introduced as $\mathbb{U}^N(x_0) \subseteq U^N$ over which the optimization occurs, where x_0 depicts the initial point of the system (52).

$$\begin{gathered} \underbrace{\mathbf{min}}_{\boldsymbol{u}} \boldsymbol{J_N(x_0, u)} \\ \boldsymbol{s.t.\ u(k)} \in \mathbb{U}, \boldsymbol{x(k)} \in \mathbb{X} \\ \boldsymbol{x(0, x_0) = x_0} \\ \boldsymbol{x(k+1) = f(x(k), u(k))} \end{gathered} \tag{56}$$

where J_N is given by:

$$\boldsymbol{J_N(x,u) = \sum_{k=0}^{N_p-1} l\big(x(k), u(k)\big)} \tag{57}$$

In the above OCP it is assumed that an optimal control sequence $u^*(.)$ exists. However, it should be acknowledged that in general, the existence of $u^*(.)$ cannot be guaranteed. Nevertheless, reasonal continuity and compactness conditions can be imposed to rigorously establish the existence of such an optimal control sequence. For specific cases concerning a general infinite-dimensional state space, relevant theorems addressing this existence can be found in the works Keerthi and Gilbert [148], or Dolezal [149].

3.3.3 Stability of Nonlinear Tracking MPC

Traditionally, the stability analysis of the MPC control law, as presented in the preceding section, relies on Lyapunov-based tools and theorem. In what follows, a brief stability analysis for the NMPC scheme is presented with stabilizing terminal constraint.

Let us present some essential definitions in stability theory. First, the origin $(0,0) \in \mathbb{R}^n$ is termed *asymptotically stable* (AS) for the system $x(k+1) = f(x(k), u(k))$ if the origin is *locally stable* and there exists $\delta > 0$ such that if $\|x(0,0) - x_{eq}\| < \delta$, then $\lim\limits_{t\to\infty}\|x(0,0) - x_{eq}\| = 0$. Local stability implies that for every $\varepsilon \in \mathbb{R}_+$ there exists a $\delta \in \mathbb{R}_+$ such that $|x| > \delta$ implies $\{|\varphi(k;x)| < \varepsilon \mid \forall k \in \mathbb{I}_+\}$.

Next, the definitions of some useful function classes are given: A function is considered Kclass if it is continuous for all values in its domain, it takes the value of zero at the origin and it is strictly increasing. Moreover, a function is considered K_∞class if it possesses all the properties of a Kclass function, but it is not bounded. Lastly, a function $\beta(x,k)$ is considered a KL class function if $\forall k \in \mathbb{I}_+$, $\beta(\cdot,k)$ remains a $\mathcal{K}$-class function and $\{\beta(s,i) \to 0 \mid i \to \infty,\ \forall s \in \mathbb{R}_+\}$, while $\beta(s,\cdot)$ remains nonincreasing.

When applying terminal constraints in the MPC optimization problem, the feedback law is only defined for those initial states x_0, for which the optimization problem within the NMPC algorithm remains feasible. Feasibility in this context means that there exists a valid control sequence that leads to a trajectory starting from x_0 and satisfying both dynamics and terminal constraints. Such initial values are called feasible and the set of all the respective values create the feasible set. This feasible set is defined as:

$$\mathbb{X}_{\boldsymbol{N}} \coloneqq \{\boldsymbol{x} \in \mathbb{R} \mid \boldsymbol{there\ exists\ u}(.) \in \boldsymbol{U^N}\} \tag{58}$$

Theorem 2 Consider the NMPC Problem given by (**56**) and optimization horizon $N \in \mathbb{N}$. Let the following assumptions hold:

a. The point $x^* \in \mathbb{X}$ is an equilibrium for an admissible control value u^*, i.e., there exists a control value $u^* \in \mathbb{U}(x^*)$ with $f(x^*, u^*) = x^*$.
b. The running cost $l: X \times U \to \mathbb{R}_0^+$ satisfies $l(x^*, u^*) = 0$ for u^* from a.
c. Suitable functions $a_1, a_2, a_3 \in K_\infty$ exist such that $a_1(|x|) \le J_N(x) \le a_2(|x|)$ and $l(x,u) \ge a_3(||x||)$ hold for all $x \in \mathbb{X}$ and $u \in U$

Then the nominal NMPC closed-loop system given by (54) with NMPC-feedback law μ_N is asymptotically stable on $\mathbb{X}_N$.

The Proof of Theorem 2 is beyond the scope of this Thesis; a detailed proof can be found in [150].

4. Introduction to quadrotor dynamics and control

This chapter provides a fundamental overview of UAVs and quadrotors, spotlighting the attributes of their dynamics and control mechanisms. Specifically, in subsection 4.1 a detailed examination delves into quadrotor dynamics, revealing the intricate interaction of forces dictating their flight. Moving forward, subsection 4.3 expands the discourse to encompass the implementation to PID control techniques specifically designed for quadrotors. Concluding this chapter, the task of quadrotor trajectory tracking is addressed, offering insights into the complexities involved in attaining precise autonomous flight paths.

4.1 Introduction to UAVs and quadrotors

A UAV refers to an aircraft that operates without the presence of a human pilot. Initially developed for military applications, the use of UAVs has rapidly expanded into diverse sectors, including scientific research, surveillance, aerial photography, and even product deliveries. Within the realm of UAVs, a distinct subgroup known as rotary wing UAVs possesses unique attributes. These vehicles are capable oof vertical hovering, takeoff, and landing and exhibit exceptional maneuverability.

One prominent member of the rotary wing UAV family is the quadrotor, commonly referred to as a quadcopter. This UAV model employs four motors, each connected to a propeller, to generate the necessary thrust for airborne lift. Each rotor is driven by an independent DC motor, converting electrical energy into mechanical force. Two of the quadrotor's motors rotate clockwise, while the remaining two rotate counter-clockwise. This arrangement results in the opposite torques generated by each rotor being balanced out by their counterpart on the opposite side. This configuration of pairs rotating in opposite directions eliminates the need of a tail rotor which counterbalances the torque produced by the rotation of the main rotor in the conventional helicopter.

4.2 The quadrotor's structure and four basic movements

The quadrotor's motion is primarily governed by four electric motors, which inherently impose limitations on the number of controllable variables during flight missions. Consequently, the quadrotor can be categorized as an under-actuated system with 6 degrees of freedom (D.O.F.). This configuration complicates the quadrotor control task, since it has only four independent control inputs, which are less than the system's D.O.F. These specific input variables pertain to the quadcopter's essential movements, crucial for stabilizing its attitude and altitude.

To obtain to control over the quadrotor, adjustments in the angular velocities Ω_i, where $i = 1,2,3,4$ of the propellers are made. Each rotor generates both the thrust and torque around its center of rotation. The proper change of the propeller's velocity leads to smooth movement of the vehicle in space.

In hovering condition all the propellers rotate with uniform angular velocities to counteract the gravitational force. This state allows the quadrotor to sustain flight without external forces or torques including movement from its position. Figure 9 shows the quadrotor structure model in hovering condition, where all the propellers have the same speed $\Omega_H = \Omega_1 = \Omega_2 = \Omega_3 = \Omega_4$. In this position the quadrotor performs stationary flight and forces or torques move it from its current state.

In Figure 9 the main body of the quadrotor is illustrated in black, while the fixed body frame B is highlighted in red. The blue color represents the angular speeds of the propellers. Each rotor has a velocity variable, and to enhance clarity, two arrows are used per propeller: a curved arrow signifies rotation direction, and the straight arrow indicates the vertical thrust velocity vector.

In order to make the quadrotor fly, four variables should be chosen to be controlled. This limitation arises from the restriction of achieving only up to four degrees of freedom out of the six available. The selection of those four variables is chosen according to the quadrotor's fundamental maneuvers that enable adjustments in both heigh and attitude.

The four fundamental movements of the quadrotor are namely thrust or throttle U_1, roll U_2, pitch U_3, and yaw U_4. The thrust command U_1 is provided by uniformly increasing (or decreasing) the speeds of all propeller. This action generates a vertical force within the body-fixed frame, causing the quadrotor to ascend or descend.

The roll command U_2 is executed by adjusting the speed of the left propeller (increase or decrease) while simultaneously adjusting the speed of the right propeller in the opposite direction (decrease or increase). This action creates a torque around the e_x^b axis, causing the quadrotor to turn.

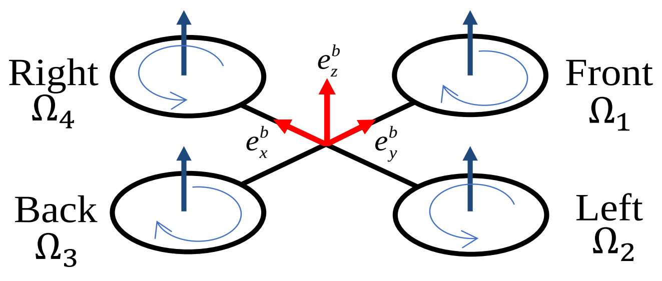

Figure 9: Quadrotor system in hovering condition

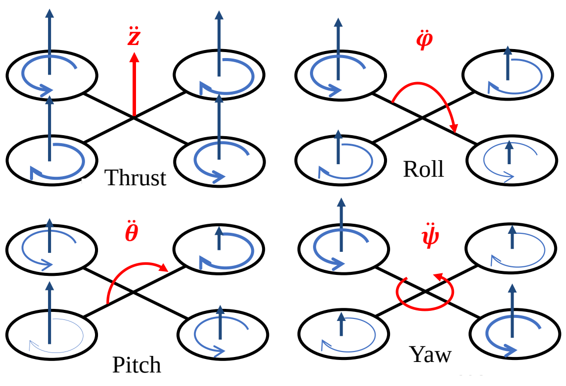

Figure 10: Four basic movements of the quadrotor

The pitch command U_3 is similar to the roll and is achieved by increasing (or decreasing) the speed of the back propeller while simultaneously adjusting the speed of the front propeller in the opposite direction. This action results in a torque along the e_y^b, which makes the quadrotor turn.

The yaw command U_4 involves adjusting the speeds of both front and rear propellers together, while simultaneously adjusting the speeds of the left right propellers in the opposite manner. This generates a torque around the e_z^b, resulting in the quadrotor's rotation. Yaw is enabled by the counter-clockwise rotation of the left-right propellers and the clockwise rotation of the front-back pair.

In Figure 10 the four basic movements of the quadrotor are depicted. The width of each arrow is proportional to the propeller's angular speed.

4.3 Quadrotor Mathematical Modelling

The mathematical model of the quadrotor describes the link between the movement and attitude with the external influences and input values. Knowing the 4 angular velocities of the propellers it is possible to predict the attitude and altitude of the quadcopter. The present model is based on the following assumptions:

- Quadrotor is a rigid body.
- Quadrotor has a symmetrical structure (the inertia matrix is diagonal).
- The center of mass and the body fixed frame origin coincide.
- The propellers are rigid.

To describe the motion of a 6 D.O.F. rigid body two reference frames are used:

- The earth inertial frame (E frame)
- The body fixed frame (B frame)

4.3.1 Newton-Euler Model

The quadrotor vehicle is a nonlinear system with under-actuation and strong coupling. Figure 11 depicts an X-configuration quadrotor, where the four rotors are divided into two pairs of (1,3) and (2,4) which rotate in opposite directions in order to compensate for the interaction of the reaction torques generated. The right cooperation between the rotors speed ensures the quadrotor's basic movements in aerial space as follows: vertical motion is achieved by increasing or decreasing the speed of all rotors; the differential speeds of rotors (1,3) and (2,4) contributes into the roll and pitch motions coupled with forward motions respectively, while yaw motion is performed through the difference of counter-torques generated by each propeller.

The translational and rotational dynamics occur using the second Newton's law of linear and angular conservation. To describe the kinematics of the quadrotor, two coordinate systems are defined, namely the earth-fixed frame $E = \{x_E, y_E, z_E\}$, which is considered fixed with respect to the earth, and the fuselage, or body-fixed frame $B = \{x_B, y_B, z_B\}$, , linked to the rigid body of the quadrotor. The two frames are depicted in Figure 11. In this case, the x_B axis is in the quadrotor's normal flight direction, y_B is orthogonal to x_B and positive to starboard in the horizontal plane, whereas z_B is oriented in the ascendant sense and orthogonal to the plane created from the vectors x_B and y_B. The linear position of the quadrotor in frame E is denoted with the vector $\xi = [x, y, z]^T \in \mathbb{R}^3$, while the orientation of the quadrotor is described in frame B by Euler's angles roll φ, pitch θ, and yaw ψ, thus forming the vector $\eta = [\varphi, \theta, \psi]^T \in \mathbb{R}^3$. Furthermore, let $\Omega = [p, q, r]^T \in \mathbb{R}^3$ and $V = [u, v, w]^T \in \mathbb{R}^3$ denote the angular and linear velocity of the quadrotor

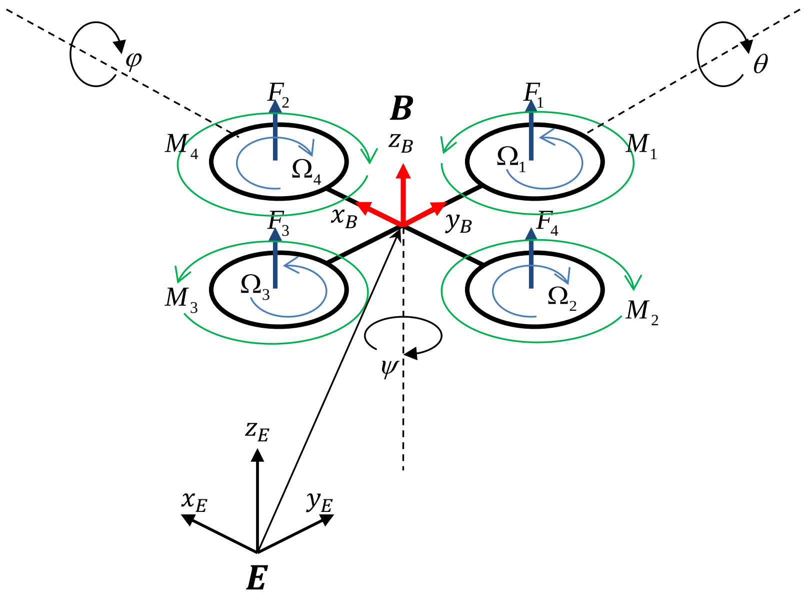

Figure 11: Schematic overview of the earth and body fixed frames of the quadrotor

vehicle in frame B, respectively. Thus, the translational and rotational kinematic model is given by:

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{R_t} \cdot \boldsymbol{V} \\ \dot{\boldsymbol{\eta}} &= \boldsymbol{R_r} \cdot \boldsymbol{\Omega} \end{aligned} \tag{59}$$

Where R_t and R_r are the transformation matrices between the two frames which are given by [49]:

$$\boldsymbol{R_t}(\boldsymbol{\eta}) = \begin{bmatrix} C_\theta \cdot C_\psi & S_\varphi \cdot S_\theta \cdot C_\psi - C_\varphi \cdot S_\psi & C_\varphi \cdot S_\theta \cdot C_\psi - S_\varphi \cdot S_\psi \\ C_\theta \cdot S_\psi & S_\varphi \cdot S_\theta \cdot S_\psi - C_\varphi \cdot C_\psi & C_\varphi \cdot S_\theta \cdot S_\psi - S_\varphi \cdot C_\psi \\ -S_\theta & S_\varphi \cdot C_\theta & C_\varphi \cdot C_\theta \end{bmatrix} \tag{60}$$

$$\boldsymbol{R_r}(\boldsymbol{\eta}) = \begin{bmatrix} 1 & S_\theta \cdot T_\theta & C_\theta \cdot T_\theta \\ 0 & C_\varphi & -S_\theta \\ 0 & S_\varphi / C_\theta & S_\varphi / C_\theta \end{bmatrix} \tag{61}$$

And $S_{(\cdot)}$, $C_{(\cdot)}$ and $T_{(\cdot)}$ are the abbreviations for $\sin(\cdot)$, $\cos(\cdot)$ and $\tan(\cdot)$ respectively.

Assuming null disturbances, the quadrotor's dynamics equation can be expressed in frame B by:

$$\begin{aligned} \boldsymbol{m} \cdot \dot{\boldsymbol{V}} &= -\boldsymbol{\Omega} \times (\boldsymbol{m} \cdot \boldsymbol{V}) - \boldsymbol{F_{aero}} - \boldsymbol{F_{grav}} + \boldsymbol{T} \\ \boldsymbol{I} \cdot \dot{\boldsymbol{\Omega}} &= -\boldsymbol{\Omega} \times (\boldsymbol{m} \cdot \boldsymbol{\Omega}) - \boldsymbol{M_{aero}} - \boldsymbol{M_{gyro}} + \boldsymbol{\tau} \end{aligned} \tag{62}$$

where $m \in \mathbb{R}$ and $I = diag(I_x, I_y, I_z) \in \mathbb{R}^{3x3}$ denote the mass and the inertia matrix of the quadrotor, $F_{grav} = m \cdot R_t^T \cdot G$ is the force due to gravity (where $G = [0 \quad 0 \quad g]^T$ is the gravity vector), $M_{gyro} = [-J_r \cdot \dot{p} \cdot \Omega_r \quad J_r \cdot \dot{q} \cdot \Omega_r \quad 0]^T$ is the gyroscopic moment formed by the rotations of the rotors around their axis (where J_r is the rotor inertia), Ω_r are the overall residual rotor speeds, and $F_{aero} = K_t \cdot V$ and $M_{aero} = K_r \cdot \Omega$ are the aerodynamical drag forces and moments, with $K_t = diag(K_x, K_y, K_z)$ and $K_r = diag(K_\varphi, K_\theta, K_\psi)$ [64], [151] denoting the aero dynamical drag coefficient matrices. The rotation of the quadrotor's propellers generates the forces responsible for its movement in space. Each rotor produces a lift force F_i and a reactive moment M_i. The force $T \in \mathbb{R}^3$ and torque $\tau \in \mathbb{R}^3$ generated by each rotor can be expressed as $F_i = b \cdot \Omega_i^2$, $M_i = d \cdot \Omega_i^2$ [18], [152], where b is the thrust coefficient, d is the drag coefficient and Ω_i is the angular velocity of rotor i, $i \in \{1,2,3,4\}$.

Proper arrangement of forces and moments leads to a total thrust $T \in \mathbb{R}^3$ and a control torque $\tau \in \mathbb{R}^3$ given by:

$$\boldsymbol{T} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{b} \cdot \sum_{i=1}^{4} \boldsymbol{\Omega_i^2} \end{bmatrix} \tag{63}$$

$$\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{l \cdot b \cdot (-\Omega_2^2 + \Omega_4^2)} \\ \boldsymbol{l \cdot b \cdot (-\Omega_3^2 + \Omega_1^2)} \\ \boldsymbol{d \cdot (\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2)} \end{bmatrix} \tag{64}$$

Using the Newton Euler formalism equation (62) can be expressed in the inertia frame E as:

$$\begin{aligned} \boldsymbol{m \cdot \ddot{x}} &= \boldsymbol{(cos\varphi \cdot sin\theta \cdot cos\psi + sin\varphi \cdot sin\psi) \cdot U_1 - K_x \cdot \dot{x}} \\ \boldsymbol{m \cdot \ddot{y}} &= \boldsymbol{(cos\varphi \cdot cos\theta \cdot sin\psi + sin\varphi \cdot cos\psi) \cdot U_1 - K_y \cdot \dot{y}} \\ \boldsymbol{m \cdot \ddot{z}} &= \boldsymbol{cos\varphi \cdot cos\theta \cdot U_1 - m \cdot g - K_z \cdot \dot{z}} \\ \boldsymbol{I_x \cdot \ddot{\varphi}} &= \boldsymbol{\dot{\theta} \cdot \psi \cdot (I_y - I_z) + l \cdot U_2 + J_r \cdot \dot{\theta} \cdot \Omega_r - K_\varphi \cdot \dot{\varphi}} \\ \boldsymbol{I_y \cdot \ddot{\theta}} &= \boldsymbol{\dot{\varphi} \cdot \psi \cdot (I_z - I_x) + l \cdot U_3 + J_r \cdot \dot{\varphi} \cdot \Omega_r - K_\theta \cdot \dot{\theta}} \\ \boldsymbol{I_z \cdot \ddot{\psi}} &= \boldsymbol{\dot{\varphi} \cdot \dot{\theta} \cdot (I_x - I_y) + U_4 - K_\psi \cdot \dot{\psi}} \end{aligned} \tag{65}$$

In order to control the flight mechanism of the quadrotor, a control input U is defined which consists of four control actions U_i, with i ∈ {1,2,3,4}. U_1 is the control action related to the total thrust and is responsible for the change of altitude, while U_2 , U_3 and U_4 are control actions related to the moments generated around the body axes x_B, y_B, z_B, respectively; more specifically, U_2 , U_3 and U_4 control the desired rotation for the roll angle φ pitch angle θ and yaw angle ψ respectively. The control input vector U is given by:

$$\begin{bmatrix} \boldsymbol{U_1} \\ \boldsymbol{U_2} \\ \boldsymbol{U_3} \\ \boldsymbol{U_4} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b \cdot ((\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)} \\ \boldsymbol{l \cdot b \cdot (-\Omega_2^2 + \Omega_4^2)} \\ \boldsymbol{l \cdot b \cdot (-\Omega_3^2 + \Omega_1^2)} \\ \boldsymbol{d \cdot (\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2)} \end{bmatrix} \tag{66}$$

4.3.2 Quadrotor State Space Modeling

The translational and rotational dynamics of the systems given by (**65**) can be defined by utilizing a state space representation. This new model of the system's dynamics is a set of first-order differential equations, and is given by the following equations:

$$\begin{aligned}
x_1 &= x\\
x_2 &= \dot{x_1}\\
x_3 &= y\\
x_4 &= \dot{x_3}\\
x_5 &= z\\
x_6 &= \dot{x_5}\\
x_7 &= \varphi\\
x_8 &= \dot{x_7}\\
x_9 &= \theta\\
x_{10} &= \dot{x_9}\\
x_{11} &= \psi\\
x_{12} &= \dot{x_{11}}
\end{aligned} \tag{67}$$

$$\begin{aligned}
\dot{x_1} &= x_2\\
\dot{x_2} &= (\cos x_7 \cdot \sin x_9 \cdot \cos x_{11} + \sin x_{11} \cdot \sin x_7) \cdot \frac{U_1}{m} - \frac{K_x}{m} \cdot x_2\\
\dot{x_3} &= x_4\\
\dot{x_4} &= (\sin x_7 \cdot \sin x_9 \cdot \cos x_{11} + \cos x_{11} \cdot \sin x_7) \cdot \frac{U_1}{m} - \frac{K_x}{m} \cdot x_4\\
\dot{x_5} &= x_6\\
\dot{x_6} &= \cos x_7 \cdot \cos x_9 \cdot \frac{U_1}{m} - \frac{g}{m} - \frac{K_z}{m} \cdot x_6\\
\dot{x_7} &= x_8\\
\dot{x_8} &= x_{10} \cdot x_{12} \cdot \frac{(I_y - I_z)}{I_x} + l \cdot \frac{U_2}{I_x} + \frac{J_r \cdot x_{10} \cdot \Omega_r}{I_x} - \frac{K_\varphi}{I_x} \cdot x_8\\
\dot{x_9} &= x_{10}\\
\dot{x_{10}} &= x_8 \cdot x_{12} \cdot \frac{(I_z - I_x)}{I_y} + l \cdot \frac{U_3}{I_y} + \frac{J_r \cdot x_8 \cdot \Omega_r}{I_y} - K_\theta \cdot x_{10}\\
\dot{x_{11}} &= x_{12}\\
\dot{x_{12}} &= x_8 \cdot x_{10} \cdot \frac{I_x - I_y}{I_z} + l \cdot \frac{U_4}{I_z} - K_\psi \cdot x_{12}
\end{aligned} \tag{68}$$

The system of twelve nonlinear first -order differential equations provided above, constitutes a mathematical representation of the quadrotor dynamics. To solve the above system, classical numerical methodologies are employed such as Runge-Kutta methods. The solution of this system at each time step yields the spatial position of the quadrotor.

4.4 PID setpoint quadrotor control

Originating in the early 1920s for ship steering automation, PID control systems gained prominence in manufacturing due to their versatile performance and tunability without specific system models. Their appeal lies in their simple architecture and suitability across processes. The acronym "PID" encompasses the Proportional, Integral, and Derivative actions, functioning as a three-part feedback mechanism, which computes an error and adjusts through these components to enhance control accuracy over time.

The quadrotor vehicle is characterized as an underactuated system, implying that it can attain a limited set of desired states within its six D.O.F., due to the constraint of having only four available control inputs. This constraint aligns with quadrotor's ability to govern only a handful of states, and this selection corresponds precisely to the four basic maneuvers outlined in section 4.2.

Hence, the four principal variables selected for control encompass the quadrotor's vertical height (z), pitch angle (θ), roll angle (φ), and yaw angle (ψ). These four parameters hold the key to establishing both altitude stability and height control for the quadrotor.

To manage the quadrotor's behavior and uphold in a specific position, it becomes necessary to determine the appropriate rotational speeds of the propellers. This procedure is recognized as inverse dynamics and, in practice, it is not always attainable and, in numerous instances, lacks a unique solution. Developing an inverse model for the quadrotor necessitates the introduction of simplifications to the system's dynamics.

The fundamental principles of the dynamics are succinctly encapsulated in equations (**62**) within subsection 4.3.1. This equation unveils the interconnection between the quadrotor's accelerations in relation to its basic movements.

Another system of equations that relates basic movements with the propellers' squared speed is described via (69).

$$\begin{aligned}
\boldsymbol{U_1} &= \boldsymbol{b} \cdot \left(+\boldsymbol{\Omega_2^2} + \boldsymbol{\Omega_4^2} + \boldsymbol{\Omega_1^2} + \boldsymbol{\Omega_3^2}\right) \\
\boldsymbol{U_2} &= \boldsymbol{l} \cdot \boldsymbol{b} \cdot \left(-\boldsymbol{\Omega_2^2} + \boldsymbol{\Omega_4^2}\right) \\
\boldsymbol{U_3} &= \boldsymbol{l} \cdot \boldsymbol{b} \cdot \left(-\boldsymbol{\Omega_1^2} + \boldsymbol{\Omega_3^2}\right) \\
\boldsymbol{U_4} &= \boldsymbol{d} \cdot \left(+\boldsymbol{\Omega_2^2} + \boldsymbol{\Omega_4^2} - \boldsymbol{\Omega_1^2} - \boldsymbol{\Omega_3^2}\right) \\
\boldsymbol{\Omega_r} &= \boldsymbol{\Omega_1} - \boldsymbol{\Omega_2} + \boldsymbol{\Omega_3} - \boldsymbol{\Omega_4}
\end{aligned} \tag{69}$$

The mathematical expression for the control action for the PID controller can be articulated within the temporal framework as follows:

$$\boldsymbol{u(t) = K_P \cdot e(t) + K_I \cdot \int_0^t e(\tau)d\tau + K_D \cdot \frac{de(t)}{dt}} \tag{70}$$

In order to facilitate the implementation of control algorithms, a simplification of the quadrotor dynamics is necessary to establish a more accessible inverse model. Equation (**62**) can be restructured based on three key considerations:

- **Simplified angular contributions**: Complicated angular terms, arising from cross-couplings, can be simplified due to minor hovering-induced changes.
- **Handling angular accelerations**: Directly referencing angular accelerations to Euler angle accelerations, bypassing complexities of frame transformations.
- **Control algorithm focus**: With four propellers and a focus on stabilizing attitude and height, equations for x and y positions are excluded.

After the above assumptions, equation (71) describes the quadrotor dynamics, which will be used for controlling the system.

$$\boldsymbol{\begin{aligned} \ddot{z} &= -g + \frac{1}{m} \cdot (\cos\theta \cdot \cos\varphi) \cdot U_1 \\ \ddot{\varphi} &= \frac{U_2}{I_x} \\ \ddot{\theta} &= \frac{U_3}{I_y} \\ \ddot{\psi} &= \frac{U_4}{I_z} \end{aligned}} \tag{71}$$

Where U_1, U_2, U_3, U_4 inside (71) are formulated according to (70).

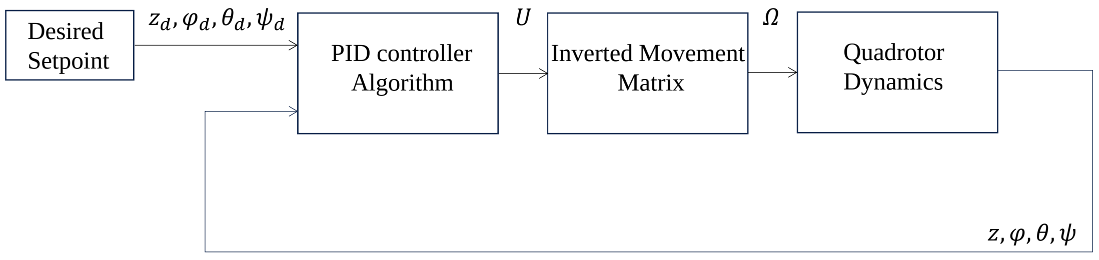

Figure 12: PID setpoint control block diagram

The above block describes the control loop of a quadrotor that performs the task of reaching a set of four desired values. The block control algorithm receives the desired values from the task and the measured values from the block quadrotor dynamics. The output of the control continues to the inverted movement matrix which relates the four basic movements with the rotational speeds of the propellers.

In Figure 12 the PID controller algorithm block constitutes the central aspect of the control system, processing the desired task and current states to generate signals for fundamental movement, mitigating position errors via PID techniques. In the next block namely inverted movement matrix the propeller speeds are computed by utilizing the basic movement signal values.

The PID quadrotor control, which centers on reaching desired setpoints, serves as the foundation for more intricate challenges such as trajectory tracking. While setpoint control focuses on stabilizing quadrotor movements around specific positions, trajectory tracking extends this concept by orchestrating a sequence of setpoints to guide the quadrotor along a predefined path. This transition from setpoint control to trajectory tracking adds a layer of complexity, demanding advanced control strategies to ensure precise and dynamic maneuvering through diverse trajectories.

4.5 The trajectory tracking problem and the PID approach

The quadrotor operates as an under-actuated system, controlling six degrees of freedom using only four motors. This necessitates control of a subset of four degrees of freedom. Notably, control of x and y coordinates rely on pitch and roll orientations, respectively, leading to two distinct subsets for control. Commands encompass three position coordinates and yaw orientation, while employing roll and pitch orientation controllers.

The set of first three PID controllers are responsible for the position control of the states x, y, z. This PID controller block regulates orientation and generates thrust controller U_1 from position references $(x_{ref}, y_{ref}, z_{ref})$.

After having the position commands U_1, U_x, U_y calculated, the rest of the quadrotor basic commands can be found via the attitude control. The objective of the attitude controller is to ensure that the attitude of the quadrotor described by its Euler's angles φ, θ, ψ tracks the desired trajectory values $\varphi_r, \theta_r, \psi_r$ asymptotically. In order to achieve the above goal, the control inputs U_2, U_3, U_4 should be estimated in such a way that Euler's angles will follow the desired trajectory attitude angles which are derived from the position controller. An overall control structure of the PID trajectory tracking scheme is given in Figure 13.

The trajectory tracking challenge for quadrotors using PID controllers can be hindered by the controllers' linear nature, struggling with complex nonlinear dynamics, external disturbances, and coupling between degrees of freedom. These limitations can lead to imprecise trajectory following and oscillations. Employing nonlinear control methods offers advantages by effectively handling intricate dynamics, uncertainties, and coupling in quadrotors. Techniques like model predictive control and backstepping control already mentioned in chapters 3.2 and 3.3 provide robustness, enabling the quadrotor to achieve accurate and stable trajectory tracking, especially in demanding scenarios.

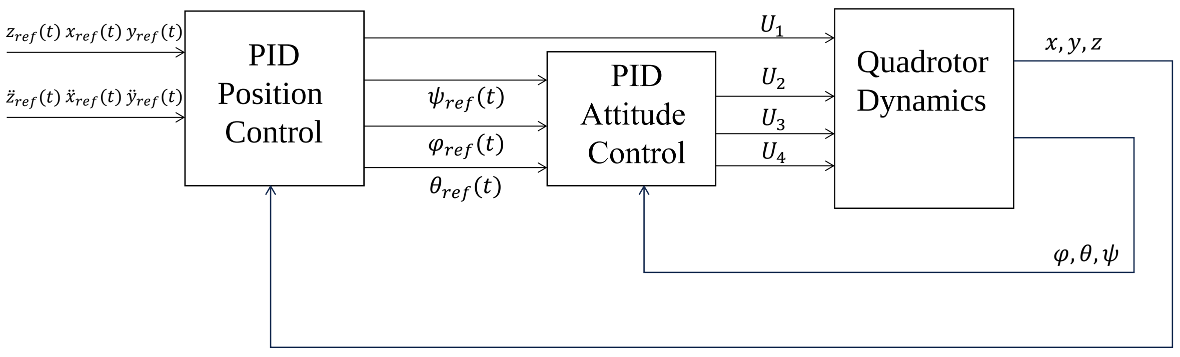

Figure 13: Overall PID structure for the trajectory tracking problem

5. Development and tuning of automatic control methods for quadrotor trajectory tracking

Within this chapter, two innovative approaches are explored that provide effective solutions to the challenging quadrotor trajectory tracking problem. Notably, both methods emphasize the utilization of nonlinear control methods to address this complex challenge. These developed methods encompass backstepping and MPC strategies, which were previously introduced in chapter 4. This chapter also entails comparative evaluations of these methods against alternative control techniques, highlighting the efficacy of each of the proposed approaches in advancing control design and tuning for quadrotor trajectory tracking.

5.1 A control strategy for quadrotor trajectory tracking based on backstepping control and radial basis function neural networks

The goal of this section is to tackle the quadrotor trajectory tracking challenge using a novel control scheme combining backstepping control and radial basis function NNs. Specifically, the proposed control approach is designed so as to guarantee Lyapunov stability of the closed-loop system based on a dynamic model of the quadrotor, and an RBF network which provides a data-driven approximation of unmodelled uncertainties of any type. The RBF network is trained using the FM algorithm, contributing to improved modeling precision and thus enhancing the proposed method to deliver improved tracking performance in the presence of unmodelled dynamics. The method is evaluated on two different simulated scenarios.

5.1.1 Quadrotor dynamics

A typical quadrotor vehicle can be seen in Figure 11. The quadrotor is considered to be a rigid body. In what follows we will consider the reference frames as stated in subsection 4.3.1,with $E = \{x_E, y_E, z_E\}$ denoting the inertial reference frame and frame $B = \{x_B, y_B, z_B\}$ representing the body fixed frame. The quadrotor state vector is defined as $X = [x, y, z, \varphi, \theta, \psi]^T$ with 6-D.O.F., where x, y, z denoting the position fo the quadrotor's center of mass and φ, θ, ψ being the Euler angles for roll, pitch and yaw. Each rotor revolves with angular speed Ω_i, with $i = 1, 2, 3, 4$.

5.1.1.1 The core reduced model dynamics

The construction of a nonlinear model without the effects of aerodynamical forces and moments for a quadrotor can be based on equation (**65**) as:

$$\begin{aligned}
\ddot{x} &= \frac{U_1}{m} \cdot (\cos\phi \cdot \sin\theta \cdot \cos\psi + \sin\phi \cdot \sin\psi) \\
\ddot{y} &= \frac{U_1}{m} \cdot (\cos\phi \cdot \sin\theta \cdot \sin\psi - \sin\phi \cdot \cos\psi) \\
\ddot{z} &= -g + \frac{U_1}{m} \cdot (\cos\phi \cdot \cos\theta) \\
\ddot{\phi} &= \dot{\theta} \cdot \dot{\psi} \cdot \frac{(I_y - I_z)}{I_x} + \frac{J_r}{I_x} \cdot \dot{\theta} \cdot \Omega_r + \frac{l}{I_x} \cdot U_2 \\
\ddot{\theta} &= \dot{\phi} \cdot \dot{\psi} \cdot \frac{(I_z - I_x)}{I_y} - \frac{J_r}{I_y} \cdot \dot{\phi} \cdot \Omega_r + \frac{l}{I_y} \cdot U_3 \\
\ddot{\psi} &= \dot{\theta} \cdot \dot{\phi} \cdot \frac{(I_x - I_y)}{I_z} + \frac{1}{I_z} \cdot U_4
\end{aligned} \tag{72}$$

where $U = [\,U_1\ U_2 U_3\ U_4]^T$ is given by (69) denotes the control input vector, m is the quadrotor mass, g is the acceleration due to the gravity on Earth, $I_x,\ I_y,\ I_z$ are the moments of inertia of the quadrotor rigid body, J_r is the moment of inertia of the propeller about its axis, l is the distance from the center of mass of the quadrotor to the axes of rotation of the propellers and $\Omega_r = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4$.

Given that the nonlinear dynamic model in (72) exclusively considers the forces and torques generated from the propellers, we label this model as the “the core reduced model” (CRM). The core reduced model takes the form of:

$$\ddot{X} = f(X, \dot{X}) = \alpha \cdot U \tag{73}$$

where

$$f = \begin{bmatrix} 0 \\ 0 \\ -g \\ \dot{\theta} \cdot \dot{\psi} \cdot \frac{(I_y - I_z)}{I_x} + \frac{J_r}{I_x} \cdot \dot{\theta} \cdot \Omega_r \\ \dot{\phi} \cdot \dot{\psi} \cdot \frac{(I_z - I_x)}{I_y} - \frac{J_r}{I_y} \cdot \dot{\phi} \cdot \Omega_r \\ \dot{\theta} \cdot \dot{\phi} \cdot \frac{(I_x - I_y)}{I_z} \end{bmatrix}, \tag{74}$$

$$\boldsymbol{a} = \begin{bmatrix} \frac{\cos\phi\cdot\sin\theta\cdot\cos\psi+\sin\phi\cdot\sin\psi}{m} & 0 & 0 & 0 \\ \frac{ccos\phi\cdot\sin\theta\cdot\sin\psi-\sin\phi\cdot\cos\psi}{m} & 0 & 0 & 0 \\ \frac{\cos\phi\cdot\cos\theta)}{m} & 0 & 0 & 0 \\ 0 & \frac{l}{I_x} & 0 & 0 \\ 0 & 0 & \frac{l}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix} \tag{75}$$

5.1.1.2 Full model dynamics

The CRM defined by equations (73)-(75) is a simplified model that does not consider various effects that act on the quadrotor like aerodynamic friction, wind gusts, ground effects, gyroscopic torques, etc. In scenarios where certain effects are known, it becomes possible to enhance model accuracy. For instance, equation (74) can be adjusted to encompass aerodynamic friction, resulting in a modified formulation:

$$\boldsymbol{f}_{full} = \begin{bmatrix} -\frac{K_x\cdot\dot{x}}{m} \\ -\frac{K_y\cdot\dot{y}}{m} \\ -\frac{K_z\cdot\dot{z}}{m} \\ \dot{\theta}\cdot\dot{\psi}\cdot\frac{(I_y-I_z)}{I_x}+\frac{J_r}{I_x}\cdot\dot{\theta}\cdot\Omega_r-\frac{K_\varphi\cdot\dot{\varphi}}{I_x} \\ \dot{\phi}\cdot\dot{\psi}\cdot\frac{(I_z-I_x)}{I_y}-\frac{J_r}{I_y}\cdot\dot{\phi}\cdot\Omega_r-\frac{K_\theta\cdot\dot{\theta}}{I_y} \\ \dot{\theta}\cdot\dot{\phi}\cdot\frac{(I_x-I_y)}{I_z}-\frac{K_\psi\cdot\dot{\psi}}{I_z} \end{bmatrix} \tag{76}$$

where $K_x, K_y, K_z, K_\varphi, K_\theta, K_\psi$ are the aerodynamic drag coefficients. In the context of this study, the more detailed quadrotor model arising by replacing equation (74) with equation (76) will be regarded as the complete quadrotor model. Conversely, when only the CRM is accessible, any unaccounted dynamics such as aerodynamic drag will manifest as uncertainties:

$$\ddot{\boldsymbol{X}} = \boldsymbol{f}(\boldsymbol{X},\dot{\boldsymbol{X}}) = \boldsymbol{\alpha}\cdot\boldsymbol{U}+\boldsymbol{h}(\boldsymbol{X},\dot{\boldsymbol{X}}) \tag{77}$$

where U, f, a are defined through equations (74), (69) and (76) respectively, and h represents the uncertainties defined as:

$$\boldsymbol{h}(\boldsymbol{X}, \dot{\boldsymbol{X}}) = [\,\boldsymbol{h_x}, \boldsymbol{h_y}, \boldsymbol{h_z}, \boldsymbol{h_\phi}, \boldsymbol{h_\theta}, \boldsymbol{h_\psi}]^{\boldsymbol{T}} \tag{78}$$

where h_x, h_y, h_z account for position uncertainties and h_ϕ, h_θ, h_ψ account for Euler angles uncertainties. The basic idea behind the proposed approach is that the uncertainties not taken into account by the CRM, can be learned by an RBF network, based on input – output data from the system.

5.1.2 Quadrotor Backstepping-RBF controller

The significance of equation (72) lies in its depiction of the quadrotor's position's dependence on the Euler angles. With a desired trajectory given by $[x_d, y_d, z_d, \psi_d]$, the desired angles φ_d and θ_d must be determined. The proposed scheme encompasses discrete controllers for position, attitude, altitude and heading. Specifically, the altitude, attitude, and heading controllers generate the input signals U_1, U_2, U_3 and U_4, while the position controller is designed to track the desired angles φ_d and θ_d by utilizing implicit control signals denoted as U_x and U_y. These controllers are collectively structured through the utilization of the backstepping method using the CRM (73)-(75), while concurrently accounting for uncertainties estimated by neural networks.

In this study, RBF networks are chosen for modeling uncertainties based on their ability to approximate intricate data relationships through radial basis activation functions as already pointed in subsection 2.2.1. These networks excel at capturing non-linear patterns, performing effectively in regression and classification tasks. Their flexibility and capacity to generalize make them well-suited applications for addressing uncertainties. The RBF networks employed in this context are trained using the symmetric fuzzy means algorithm, a method thoroughly detailed in subsection 2.2.2. This algorithm serves as the cornerstone for effectively training and estimating uncertainties in the RBF network's predictions, contributing to the overarching framework outlined in this study.

5.1.2.1 State Space Representation

The model in (77) can be represented in state space by using equation (**67**) with:

$$\begin{aligned} \boldsymbol{X} &= [\,\boldsymbol{x}\;\dot{\boldsymbol{x}}\;\boldsymbol{y}\;\dot{\boldsymbol{y}}\;\boldsymbol{z}\;\dot{\boldsymbol{z}}\;\boldsymbol{\phi}\;\dot{\boldsymbol{\phi}}\;\boldsymbol{\theta}\;\dot{\boldsymbol{\theta}}\;\boldsymbol{\psi}\;\dot{\boldsymbol{\psi}}\,]^{\boldsymbol{T}} \\ &= [\,\boldsymbol{x_1}\;\boldsymbol{x_2}\;\boldsymbol{x_3}\;\boldsymbol{x_4}\;\boldsymbol{x_5}\;\boldsymbol{x_6}\;\boldsymbol{x_7}\;\boldsymbol{x_8}\;\boldsymbol{x_9}\;\boldsymbol{x_{10}}\;\boldsymbol{x_{11}}\;\boldsymbol{x_{12}}\,]^{\boldsymbol{T}} \end{aligned} \tag{79}$$

We denote the desired trajectory with the vector given by the following equation:

$$\boldsymbol{X^d} = [\, \boldsymbol{x_1^d\ x_3^d\ x_5^d\ x_7^d\ x_9^d\ x_{11}^d}] \tag{80}$$

In order to track the trajectory successfully, the error signal $z_i = x_i^d - x_i$ must be zero.

5.1.2.2 Controller Design

From (72) we have that $\ddot{x}$ and $\ddot{y}$ are given as:

$$\begin{aligned} \ddot{x} &= \frac{U_1}{m} \cdot (\cos\phi \cdot \sin\theta \cdot \cos\psi + \sin\phi \cdot \sin\psi) \\ \ddot{y} &= \frac{U_1}{m} \cdot (\cos\phi \cdot \sin\theta \cdot \sin\psi - \sin\phi \cdot \cos\psi) \end{aligned} \tag{81}$$

From the above equations we denote as U_x and U_y the following equations:

$$\begin{aligned} U_x &= \cos\phi \cdot \sin\theta \cdot \cos\psi + \sin\phi \cdot \sin\psi \\ U_y &= \cos\phi \cdot \sin\theta \cdot \sin\psi - \sin\phi \cdot \cos\psi \end{aligned} \tag{82}$$

The angles φ_d and θ_d can be determined by (81) as:

$$\begin{aligned} \varphi_d &= \arcsin\left(U_x \cdot \sin\psi_d - U_y \cdot \cos\psi_d\right) \\ \theta_d &= \arcsin\left(\frac{U_x \cdot \cos\psi_d + U_y \cdot \sin\psi_d}{\cos\varphi_d}\right) \end{aligned} \tag{83}$$

Combining equations (77), (78), (81) and (82) we obtain:

$$\begin{aligned} \dot{x}_2 = \ddot{x} &= \frac{U_1 \cdot U_x}{m} + \frac{h_x}{m} \\ \dot{x}_4 = \ddot{y} &= \frac{U_1 \cdot U_y}{m} + \frac{h_y}{m} \end{aligned} \tag{84}$$

We proceed by building the backstepping control law for the following nonlinear second order system:

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= (\cos x_7 \cdot \sin x_9 \cdot \cos x_{11} + \sin x_{11} \cdot \sin x_7) \cdot \frac{U_1}{m} - \frac{h_x}{m} \end{aligned} \tag{85}$$

Step 1: To proceed we define the reference error function for state x_1:

$$\boldsymbol{z_1 = x_1^d - x_1} \tag{86}$$

The time derivative of z_1 is given by:

$$\boldsymbol{\dot{z_1} = \dot{x_1^d} - \dot{x_1} = \dot{x_1^d} - x_2} \tag{87}$$

Considering the Lyapunov function for z_1:

$$\boldsymbol{V_1\big(z_1(t)\big) = \frac{1}{2} \cdot z_1^2(t)} \tag{88}$$

The derivative of the Lyapunov function V_1 is:

$$\boldsymbol{\dot{V_1}\big(z_1(t)\big) = z_1 \cdot \dot{z_1} = z_1 \cdot (\dot{x_1}^d - x_2)} \tag{89}$$

Step 2: In what follows we consider a change in variables by introducing the virtual control input v_1:

$$\boldsymbol{z_2 = v_1 - x_2} \tag{90}$$

The derivative of the above function variable z_2 is:

$$\boldsymbol{\dot{z_2} = \dot{v_1} - \dot{x_2}} \tag{91}$$

By utilizing (91), (89) can be written as:

$$\boldsymbol{\dot{V_1}\big(z_1(t)\big) = z_1 \cdot (\dot{x_1}^d + z_2 - v_1)} \tag{92}$$

The stabilization of z_1 can be obtained by designing the first virtual control input v_1 such that:

$$\boldsymbol{v_1 = x_1^d + k_1 \cdot z_1,\ \ k_1} \in \mathbb{R}^+ \tag{93}$$

We proceed with defining the augmented V_2 as:

$$\boldsymbol{V_2\big(z_1(t), z_2(t)\big) = V_1\big(z_1(t)\big) + \frac{1}{2} \cdot z_2(t)^2} \tag{94}$$

With the time derivative gibe by:

$$\boldsymbol{\dot{V_2}\big(z_1(t), z_2(t)\big) = -k_1 \cdot\ z_1^2 + z_2 \cdot \left(\frac{U_x \cdot U_1}{m} + \frac{h_x}{m} - \ddot{x}_1^d - k_1 \cdot \dot{z}_1 - z_1\right)} \tag{95}$$

The control law U_x is designed as:

$$\boldsymbol{U_x = m \cdot \frac{-\frac{h_x}{m} + \ddot{x}_1^d + z_1 + k_1 \cdot \dot{z}_1 - k_2 \cdot z_2}{U_1}} \quad , \boldsymbol{k_1,\ k_2} \in \mathbb{R}^+ \tag{96}$$

By choosing the control law given by (**96**) in equation (**95**) one obtains $\dot{V}_2(z_1(t), z_2(t)) \leq 0$. Consequently, the control system (**85**) by choosing the Lyapunov function (**94**) and the feedback control law as (**96**) is asymptotically stable according to Theorem 1, given in section 3.2.1 of chapter 3.

Applying the same technique for states x_3, x_5, x_7, x_9 and x_{11}, we derive all the control laws, summarized below:

$$\begin{aligned}
\boldsymbol{U_1} &= \boldsymbol{\frac{m}{\cos x_7 \cdot \cos x_9} \cdot \left(g + \ddot{x}_5^d + z_5 + k_5 \cdot \dot{z}_5 - k_6 \cdot z_6 - \frac{h_z}{m}\right)} \\
\boldsymbol{U_2} &= \boldsymbol{\frac{I_x}{l} \cdot \left(-f_4 + \ddot{x}_7^d + z_7 + k_7 \cdot \dot{z}_7 - k_8 \cdot z_8 - \frac{h_\phi}{I_x}\right)} \\
\boldsymbol{U_3} &= \boldsymbol{\frac{I_y}{l} \cdot \left(-f_5 + \ddot{x}_9^d + z_9 + k_9 \cdot \dot{z}_9 - k_{10} \cdot z_{10} - \frac{h_\psi}{I_y}\right)} \\
\boldsymbol{U_4} &= \boldsymbol{I_z \cdot \left(-f_6 + \ddot{x}_{11}^d + z_{11} + k_{11} \cdot \dot{z}_{11} - k_{12} \cdot z_{12} - \frac{h_\psi}{I_z}\right)} \\
\boldsymbol{U_x} &= \boldsymbol{\frac{m}{U_1} \cdot \left(\ddot{x}_1^d + z_1 + k_1 \cdot \dot{z}_1 - k_2 \cdot z_2 - \frac{h_x}{m}\right)} \\
\boldsymbol{U_y} &= \boldsymbol{\frac{m}{U_1} \cdot \left(\ddot{x}_3^d + z_3 + k_3 \cdot \dot{z}_3 - k_4 \cdot z_4 - \frac{h_y}{m}\right)}
\end{aligned} \tag{97}$$

where $h_x, h_y, h_z, h_\phi, h_\theta, h_\psi$ are approximated by six RBF, trained with the FM algorithm as described above. The proposed control framework is summarized in Figure 14.

5.1.3 Trajectory Tracking simulation set up

This section delves into the generation of flying simulation data, a vital task for creating a dynamic environment. Additionally, it covers the critical RBF network training stage, where the network is adeptly trained to model system uncertainties using the FM algorithm. Furthermore, performance evaluation across two diverse trajectory scenarios showcases the method’s versatility and effectiveness.

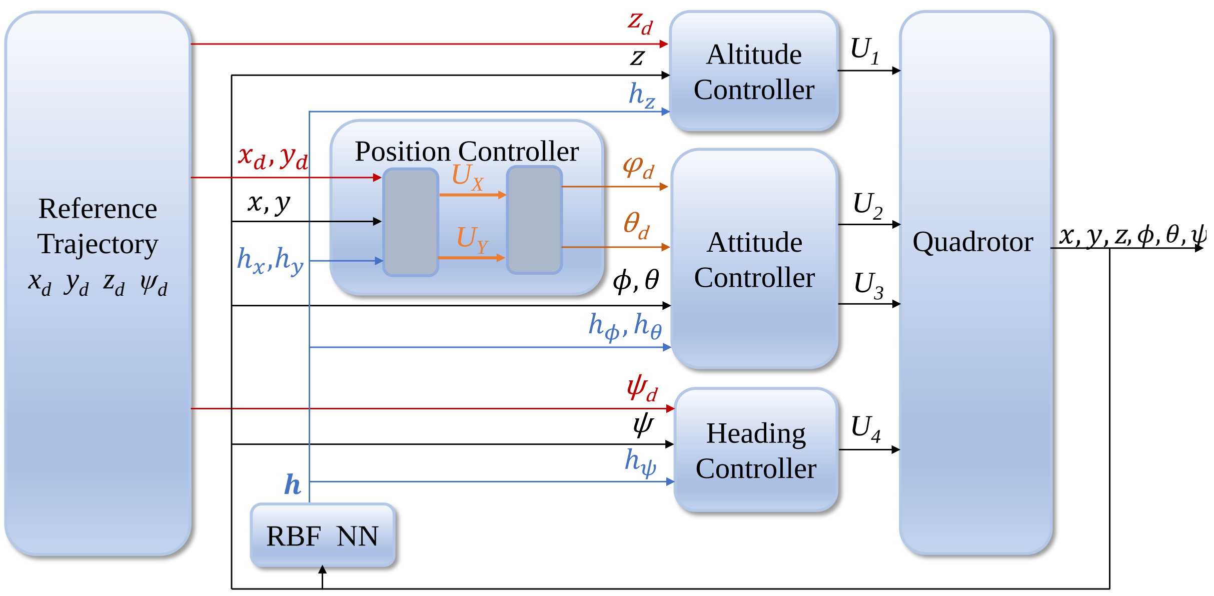

Figure 14: RBF-Backstepping control scheme

5.1.3.1 Data Generation

To train the neural networks, a number of data have been generated through flying simulations based on the full model, which in this case plays the role of a real quadcopter, exhibiting dynamics unmodelled by the CRM. Values for the quadrotor physical parameters can be found in [153] while the aerodynamic drag coefficients k_i are depicted in Table 7. The designed scheme was fully programmed in MATLAB environment and the full plant dynamics were solved by using the Runge-Kutta (4,5) formula. The full model was excited using random inputs and data were collected for all state variables $[X\ \dot{X}]^T$ and accelerations $\ddot{X}$ with a sampling rate of 250 Hz. The uncertainties $h(X, \dot{X})$ to be given as targets to the neural networks were calculated by utilizing equation (77) in the following form:

$$\boldsymbol{h}(\boldsymbol{X}, \dot{\boldsymbol{X}}) = \ddot{\boldsymbol{X}} - \boldsymbol{f}(\boldsymbol{X}, \dot{\boldsymbol{X}}) - \boldsymbol{a} \cdot \boldsymbol{U} \tag{98}$$

In total, a number of 20000 data samples were collected.

Table 7: Values for Aerodynamic drag coefficients

Aerodynamical coefficient	k_x	k_y	k_z	k_φ	k_θ	k_ψ
Value	-0.096	-0.222	-0.092	-0.06	-0.06	-0.02

5.1.3.2 RBF Network Training

The generated data were divided into three separate subsets: 50% for training, 25% for validation, and 25% for testing. The training subset was used for determining the model parameters, the validation set for selecting the most appropriate model, and the testing for providing an independent estimation of the model accuracy.

In this work, six RBF neural networks were utilized to approximate the uncertainties $h(X,\dot{X})$. Each RBF network employes two input variables, each corresponding to distinct states for the quadrotor. For example, the RBF modelling h_x, uses as inputs the state variables x and $\dot{x}$.

Model selection for the FM algorithm is controlled by the number of fuzzy subspaces s. The optimal value of s for each one of the six trained networks was found by performing an exhaustive search procedure in the range [4 - 50]. The model exhibiting the best modelling performance on the validation subset was used for each individual uncertainty. The mean absolute error (MAE) was used as metric for evaluating the model performance of each RBF network:

$$\boldsymbol{MAE_i = \frac{1}{N} \cdot \sum_{k=1}^{K} |y_i(k) - \hat{y}_i(k)|} \tag{99}$$

where $y_i(k)$ and $\hat{y}_i(k)$ represent the real measurements and model predictions for RBF network i, respectively.

5.1.3.3 Controller Implementation

To evaluate the performance of the proposed RBF-backstepping controller, two distinct reference spatial trajectories scenarios were employed.

The first simulation was performed on a spiral trajectory, which exhibits simple geometrical characteristics. This trajectory is represented by the following equation:

$$\begin{array}{ccc} \boldsymbol{x_{ref}(t)} & = & \boldsymbol{0.5 \cdot cos(0.5 \cdot t)} \\ \boldsymbol{y_{ref}(t)} & = & \boldsymbol{0.5 \cdot sin(0.5 \cdot t)} \\ \boldsymbol{z_{ref}(t)} & = & \boldsymbol{1 + \frac{t}{10}} \end{array} \tag{100}$$

The second trajectory, named “complex trajectory”, presents more complicated geometrical characteristics and is given by:

$$\begin{aligned} \boldsymbol{x_{ref}(t)} &= \boldsymbol{cos(0.15 \cdot t) - cos(0.15 \cdot t)^3} \\ \boldsymbol{y_{ref}(t)} &= \boldsymbol{sin(0.15 \cdot t) - sin(0.15 \cdot t)^3} \\ \boldsymbol{z_{ref}(t)} &= \boldsymbol{0.3 \cdot t} \end{aligned} \tag{101}$$

For both trajectories, the controller sampling time was set equal to 0.1s. The performance of the RBF – backstepping method was evaluated against a standard backstepping controller, using only the CRM (73)-(75), without taking into account the RBF uncertainty approximation. The parameters for both controllers were tuned using particle swarm optimization.

5.1.4 Results and Discussion

Table 8 depicts the results for the RBF network training procedure, including the MAE on the testing set, the selected number of fuzzy subspaces and the resulting number of RBF centers for each of the 6 networks. It can be seen that the FM algorithm manages to select the RBF centers in such a way, so as to produce a low MAE value for all the uncertainty models. The results of the two controllers are depicted in Table 9 and Table 10, which include the sum of absolute errors (SAE) per dimension and the sum of tracking errors (STE) for both schemes, in the case of the spiral and the complex trajectory, respectively. The sum of absolute errors is given by:

$$\boldsymbol{SAE_i} = \sum_{\boldsymbol{t=1}}^{\boldsymbol{TS}} \left| \boldsymbol{x_i(t) - x_i^d(t)} \right| \tag{102}$$

where $t = 1, \dots, TS$ represents the running time of the flight simulation, and $x_i(k)$, $x_i^d(k)$ represent the real state measurement and the desired state for the ith dimension, with $i = x, y, z.$

The sum of tracking errors, corresponds to the sum of Euclidean distances between the actual position of the quadrotor and the reference trajectory and can be used to assess the overall performance of the proposed controller; STE is calculated as follows:

$$\boldsymbol{STE} = \sum_{\boldsymbol{t=1}}^{\boldsymbol{TS}} \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{3}} \left(\boldsymbol{q_i(t) - p_i(t)}\right)^{\boldsymbol{2}}} \tag{103}$$

where $q(t) = [x\ y\ z]$ and $p(t) = [x^d\ y^d\ z^d]$ are vectors representing the actual quadrotor Cartesian coordinates and the reference trajectory coordinates at time t, respectively.

Visual representations for the performance of the two controllers can be seen in Figures 15-18 which depict the tracking errors in the three spatial dimensions, and the 3D trajectories for both controllers, together with the reference, in the case of the spiral and the complex trajectory, respectively. Taking into account the tracking errors shown in Table 9 regarding the spiral trajectory, the RBF – backstepping control scheme outperforms its rival in all spatial dimensions, while it manages to produce an improvement of approximately 24% in terms of STE.

A similar result is obtained for the complex trajectory where once more the proposed control method manages to outperform the standard backstepping approach, as it can be seen by examining the metrics of Table 10. The proposed approach still produces lower control tracking errors in terms of SAE in all three dimensions, while at the same time it achieves a significant improvement of 23.3% when taking into account the STE.

This result is due to the fact that the proposed scheme, enhanced by the efficient performance of the RBF models, manages to approximate adequately the effect of aerodynamic friction, which the standard backstepping controller fails to capture as it is solely based on the CRM. The difference between the two controllers is more obvious in the 3D graphs at the beginning of each trajectory, where the proposed method manages to approach the reference more efficiently, but even later, the superiority of the RBF-backstepping controller is still visible in graphs depicting the error per dimension. Notice that the difference is more significant in the y dimension, which is to be expected, as the value used for the drag coefficient k_y happens to be higher than k_x and k_z.

It should be noted that in this simulation, the difference in terms of unmodelled dynamics between the two controllers is just due to aerodynamic friction, which is taken into account by the RBF networks, but not by the CRM. In a real-world situation though, the difference is expected to be significantly higher, due to the increased presence of unmodelled dynamics and uncertainties which are not accounted by the CRM, but can be identified by the RBF network (such as ground effects, gyroscopic torques, etc.)

Table 8: RBF Modeling results on the testing subset

RBF model	$\boldsymbol{h_x}$	$\boldsymbol{h_y}$	$\boldsymbol{h_z}$	$\boldsymbol{h_\varphi}$	$\boldsymbol{h_\theta}$	$\boldsymbol{h_\psi}$
# of fuzzy subspaces	29	31	27	27	19	21
# of RBF centers	77	86	32	67	47	81
MAE	1.86 E-05	2.47 E-05	1.04E-04	1.10E-06	2.06E-06	1.8E-04

Table 9: Controller tracking errors for the spiral trajectory

Controller	**SAE x(m)**	**SAE y(m)**	**SAE z(m)**	**STE (m)**
RBF - Backstepping	4.456	3.718	3.902	7.503
Backstepping	4.539	6.375	4.896	9.877

Table 10: Controller tracking errors for the complex trajectory

Controller	**SAE x(m)**	**SAE y(m)**	**SAE z(m)**	**STE (m)**
RBF - Backstepping	3.962	2.837	0.1810	5.090
Backstepping	4.567	4.452	0.2933	6.638

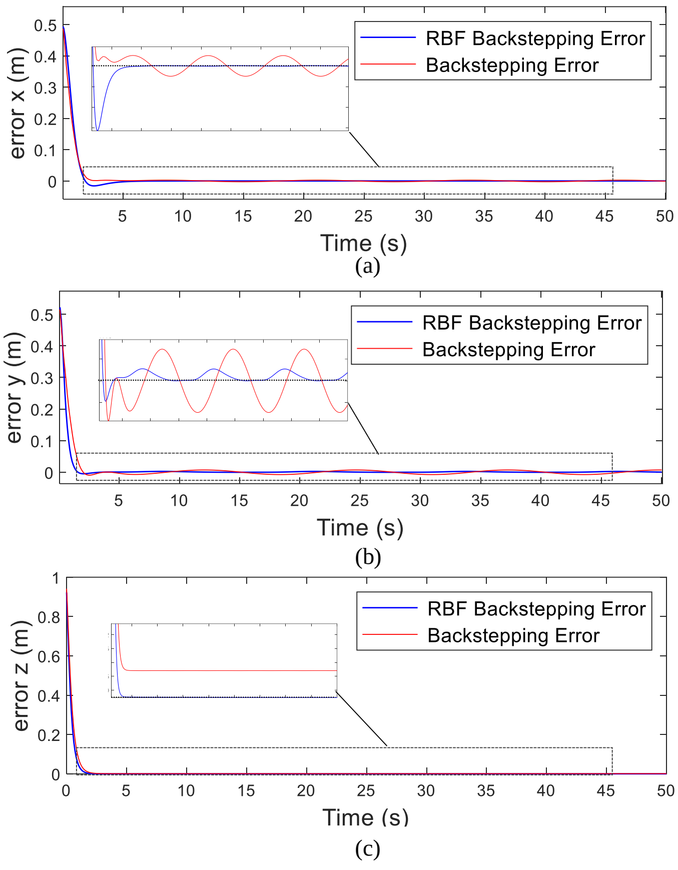

Figure 15: Spiral trajectory error for (a) *x*, (b) *y*, (c) *z* dimensions

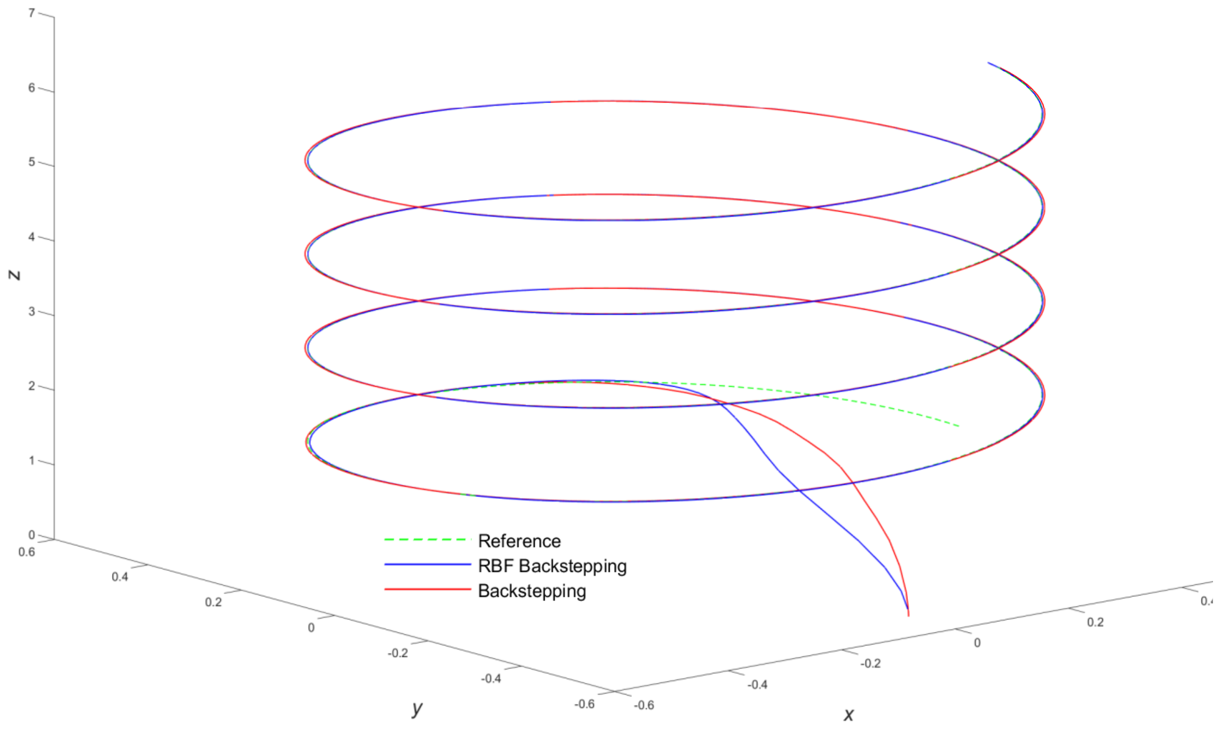

Figure 16: 3D Spiral trajectory simulation results

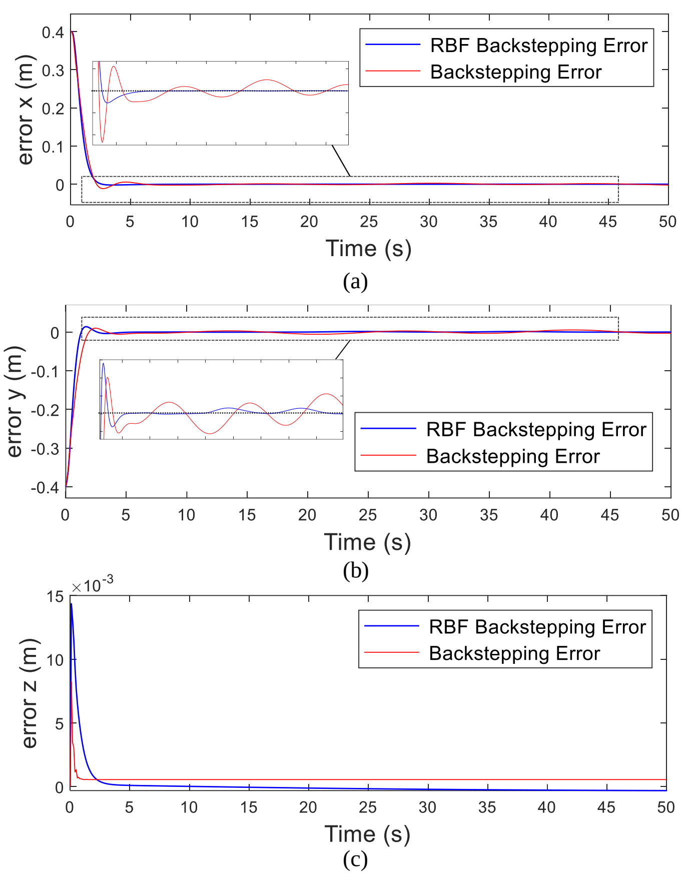

Figure 17: Complex trajectory error for (a) *x*, (b) *y*, (c) *z* dimensions

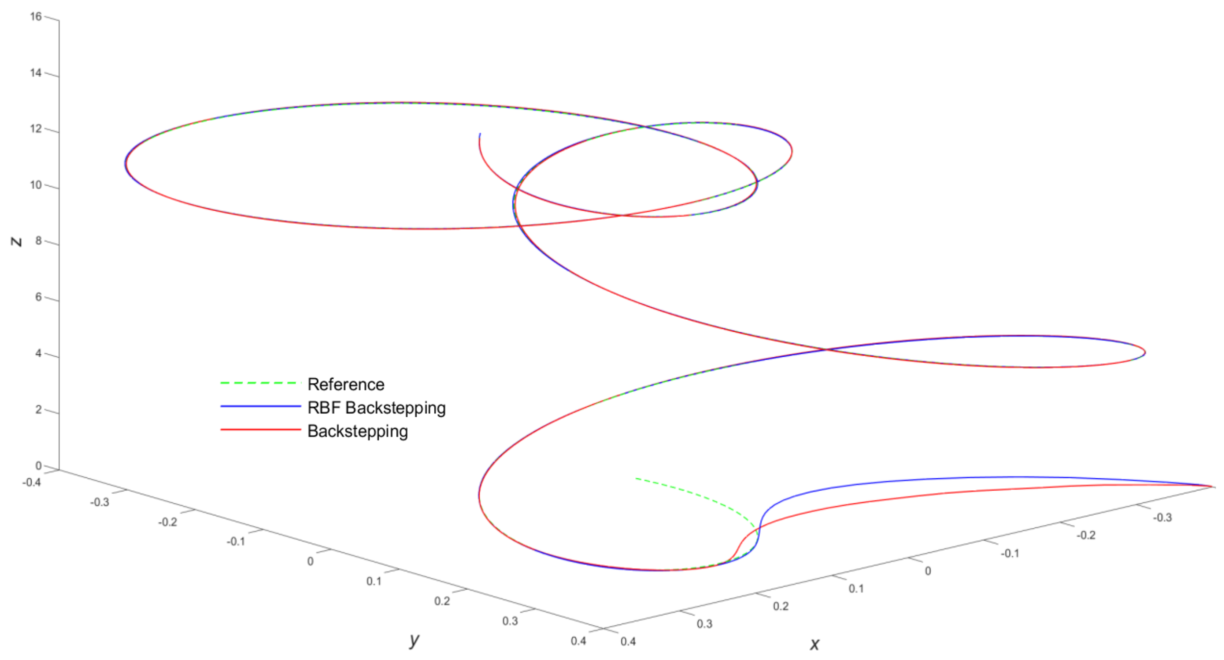

Figure 18: 3D Complex trajectory simulation results

In summary, the RBF-backstepping controller has showcased its trajectory tracking capabilities and its capacity to take into account aerodynamic friction effects. Nevertheless, delving deeper into the challenges of quadrotor control, the demand for an efficient multi-controller tuning approach becomes more pronounced, given the common use of multiple controllers in trajectory tracking schemes. With this in mind, the subsequent section will delve into the imperative need of devising such a method within the quadrotor trajectory tracking control framework.

5.2 A new cooperative PSO optimization approach for tuning an MPC-based quadrotor trajectory tracking scheme

In this section, the objective is to develop a novel method for tuning a quadrotor trajectory-tracking MPC framework utilizing cooperative particle swarm optimization. The control framework includes two subsystems: an MPC controller for path following and a PID scheme for attitude stabilization. These subsystems necessitate numerous tuning parameters, optimized by a CPSO scheme. Multiple swarms optimize distinct solution vector components-MPC and PID parameters-collaboratively enhancing path tracking in the integrated control framework. The approach is assessed through diverse trajectory simulation experiments.

5.2.1 Quadrotor mathematical model

The quadrotor modeling presented in this section builds upon the analysis conducted in section 4.3.1, notably relying on Newton-Euler formulation for modeling dynamics. The quadrotor's dynamics equation assuming null disturbances in the body frame is given by equation (**62**), while by using the Newton-Euler formalism (**65**) expressed in the inertial frame is given by the following equation:

$$\begin{aligned}
m \cdot \ddot{x} &= (\cos\varphi \cdot \sin\theta \cdot \cos\psi + \sin\varphi \cdot \sin\psi) \cdot U_1 - K_x \cdot \dot{x} \\
m \cdot \ddot{y} &= (\cos\varphi \cdot \cos\theta \cdot \sin\psi + \sin\varphi \cdot \cos\psi) \cdot U_1 - K_y \cdot \dot{y} \\
m \cdot \ddot{z} &= \cos\varphi \cdot \cos\theta \cdot U_1 - m \cdot g - K_z \cdot \dot{z} \\
I_x \cdot \ddot{\varphi} &= \dot{\theta} \cdot \psi \cdot (I_y - I_z) + l \cdot U_2 + J_r \cdot \dot{\theta} \cdot \Omega_r - K_\varphi \cdot \dot{\varphi} \\
I_y \cdot \ddot{\theta} &= \dot{\varphi} \cdot \psi \cdot (I_z - I_x) + l \cdot U_3 + J_r \cdot \dot{\varphi} \cdot \Omega_r - K_\theta \cdot \dot{\theta} \\
I_z \cdot \ddot{\psi} &= \dot{\varphi} \cdot \dot{\theta} \cdot (I_x - I_y) + U_4 - K_\psi \cdot \dot{\psi}
\end{aligned} \tag{104}$$

The flight control of the quadrotor is achieved through the control input U, whih is defined by four control actions U_i with, $i \in \{1,2,3,4\}$. The control input is given by equation (66):

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b \cdot ((\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ l \cdot b \cdot (-\Omega_2^2 + \Omega_4^2) \\ l \cdot b \cdot (-\Omega_3^2 + \Omega_1^2) \\ d \cdot (\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{bmatrix} \tag{105}$$

5.2.2 Controller design

5.2.2.1 Error MPC position control

In this section a control framework that deals with the trajectory tracking problem of an autonomous discrete-time nonlinear quadrotor system is given. To be more specific, a linear error-based MPC strategy is constructed in order to control the quadrotor position. The error model derived is divided into two different position controllers. The first one controls the altitude of the quadrotor via the input U_1 while the second one is responsible for the control in the x-y plane through the inputs u_x, u_y, which are derived as follows

The position system (104) can be rewritten in state space form as $\dot{\bar{\xi}}(t) = f\left(\bar{\xi}(t), u_{\xi}(t)\right)$, where $\bar{\xi}(t) = [\, x(t)\; u(t)\; y(t)\; v(t)\; z(t)\; w(t)\,]$ stands for the augmented system state space vetor, and $u(t),\ v(t),\ w(t)$ are the linear velocities of the quadrotor's mass center:

$$\dot{\bar{\xi}}(t) = \begin{bmatrix} u(t) \\ u_x(t) \cdot \dfrac{U_1(t)}{m} \\ v(t) \\ u_y(t) \cdot \dfrac{U_1(t)}{m} \\ w(t) \\ -g + \cos(\theta(t)) \cdot \cos(\varphi(t)) \cdot \dfrac{U_1(t)}{m}) \end{bmatrix} \tag{106}$$

where

$$\begin{aligned} u_x(t) &= \cos(\psi(t)) \cdot \sin(\theta(t)) \cdot cos(\varphi(t)) + sin(\psi(t)) \cdot sin(\varphi(t)) \\ u_y(t) &= sin(\psi(t)) \cdot sin(\theta(t)) \cdot cos(\varphi(t)) - cos(\psi(t)) \cdot sin\,(\varphi(t))) \end{aligned} \tag{107}$$

The reference trajectory is provided off-line and a virtual reference vehicle with the same dynamics as the quadrotor is proposed along with the real one on the same track. Assuming that there is no external disturbance to the virtual reference quadrotor, the dynamics can be written in the form:

$$\dot{\bar{\xi}}_{ref}(t) = f\left(\bar{\xi}_{ref}(t), u_{\xi ref}(t)\right) \tag{108}$$

where $\bar{\xi}_{ref} = (t) = [\, x_{ref}(t)\; u_{ref}(t)\; y_{ref}(t)\; v_{ref}(t)\; z_{ref}(t)\; w_{ref}(t)\,]$ and $u_{\xi ref}(t) = [\, u_{xref}\; u_{yref}\; U_{1ref}]$ are the reference state vector and the reference input control, respectively.

Subtracting the virtual system given by (108) from the augmented position system (106) and using the forward Euler method for altitude z, the error model is obtained as a discrete time linear equation:

$$\boldsymbol{\tilde{x}_{ref}(k+1) = A \cdot \tilde{x}_{\xi}(k) + B(k) \cdot \tilde{u}_{\xi}(k)} \tag{109}$$

where $\tilde{x}_{\xi}(k) = \bar{\xi}(k) - \bar{\xi}_{ref}(k)$ is the position error vector and $\tilde{u}_{\xi}(k) = u(k) - u_{\xi ref}(k)$ is the control input error vector. The error model (109) is divided into two discrete state-space subsystems, namely the altitude and attitude one.

The height position error model for the altitude subsystem takes the following form:

$$\boldsymbol{\tilde{x}_{z}(k+1) = A_z \cdot \tilde{x}_{z}(k) + B_z(k) \cdot \tilde{u}_{z}(k)} \tag{110}$$

where matrices A_z and $B_z(k)$ are given as follows:

$$\boldsymbol{A_z = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, B_z(k) = \begin{bmatrix} 0 \\ \frac{\Delta t}{m} \cdot \cos(\theta(k)) \cdot cost(\varphi(\kappa)) \end{bmatrix}} \tag{111}$$

with Δt being the sampling time.

Successful path tracking for a desired trajectory is possible by finding the suitable control inputs. In order to achieve this, an unconstrained MPC problem is designed for solving the tracking error problem by utilizing the formulation given of (56) in subsection 3.3.2. The OCP problem takes the following form:

$$\begin{gathered} \boldsymbol{\underbrace{\min}_{\bar{u}_z} J_z(\bar{x}_0, \bar{u}_z)} \\ \boldsymbol{s.t.\;\; \bar{x}_z(0, \bar{x}_0) = \bar{x}_0} \\ \boldsymbol{\tilde{x}_{z}(k+1) = A_z \cdot \tilde{x}_{z}(k) + B_z(k) \cdot \tilde{u}_{z}(k)} \end{gathered} \tag{112}$$

with J_z is the given by:

$$\boldsymbol{J_z(\bar{x}_0, \bar{u}_z) = \bar{x}_z^T \cdot Q_z \cdot \bar{x}_z + \bar{u}_z^T \cdot R_z \cdot \bar{u}_z + \bar{x}_z^T(k + N_p/k) \cdot G_z \cdot \bar{x}_z(k + N_p/k)} \tag{113}$$

where $\bar{x}_z = \hat{\tilde{x}}_z - \hat{\tilde{x}}_{zref}$, $\bar{u}_z = \hat{\tilde{u}}_z - \hat{\tilde{u}}_{zref}$ and $\bar{x}_z(k + N_p /k) = \tilde{x}_z(k + N_p /k) - \tilde{x}_{zref}(k + N_p/k)$. Matrices Q_z, R_z and G_z are positive definite and penalize divergences from the references state, input and terminal state, respectively. The predictions of the plant output $\tilde{x}_z(k + j /k)$ $\tilde{x}_z(k + N_p /k)$ are computed using a linear time-varying state-space model of the vehicle using (110) and (111), giving:

$$\boldsymbol{\tilde{x}_z(k+1) = P_z(k/k) \cdot x_z(k/k) + H_z(k/k) \cdot \tilde{u}_z(k)} \tag{114}$$

where $\tilde{u}(k/k) = U_1(k) - u_{zref}(k)$ and $\hat{\tilde{x}}_z \in \mathbb{R}^{N_p}$ is the height state-space vector. The reference height and input state error vectors are:

$$\boldsymbol{\hat{\tilde{x}}_{zref}} = \begin{bmatrix} \boldsymbol{\tilde{x}_{zref}(k+1/k) - \tilde{x}_{zref}(k/k)} \\ \vdots \\ \boldsymbol{\tilde{x}_{zref}(k+N_p/k) - \tilde{x}_{zref}(k/k)} \end{bmatrix}, \ \boldsymbol{\hat{\tilde{u}}_{zref}} = \begin{bmatrix} \boldsymbol{\tilde{u}_{zref}(k/k) - \tilde{u}_{zref}(k-1/k)} \\ \vdots \\ \boldsymbol{\tilde{u}_{zref}(k+N_c-1/k) - \tilde{u}_{zref}(k-1/k)} \end{bmatrix} \tag{115}$$

where N_p is the prediction horizon, showing how far the controller predicts to the future, and N_c is the control horizon, which constitutes the number of consecutive moves to be manipulated by the controller for minimizing the cost function J_z. The control moves $\boldsymbol{\hat{\tilde{u}}_z}$ at time k are evaluated by solving the minimization problem (**112**); however, only the first control move is implemented on the system, while the remaining ones are rejected. After applying the control move $\hat{\tilde{u}}_{zref}(k/k)$, the system moves to a new position at time $k + 1$ and the minimization problem is solved again for the new current state, yielding a new control action.

Thus, the control input applied to the quadrotor is:

$$\boldsymbol{U_1(k) = \tilde{u}(k/k) + u_{zref}(k)} \tag{116}$$

In a similar way the x-y motion position error model takes the following form:

$$\boldsymbol{\hat{\tilde{x}}_{xy}(k+1) = A_{xy} \cdot \tilde{x}_{xy}(k) + B_{xy}(k) \cdot \hat{\tilde{u}}_{xy}(k)} \tag{117}$$

where matrices A_{xy} and $B_{xy}(k)$ are given as followed:

$$\boldsymbol{A_{xy}} = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{\Delta t} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{\Delta t} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}, \boldsymbol{B_{xy}(k)} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{\frac{\Delta t}{m} \cdot U_1(k)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\frac{\Delta t}{m} \cdot U_1(k)} \end{bmatrix} \tag{118}$$

Following the same procedure as above, the OCP problem for the problem of minimizing the tracking error in the x-y direction takes the following form:

$$\begin{gathered}\underbrace{\min}_{\bar{u}_{xy}} J_{xy}(\bar{x}_0,\bar{u}_{xy})\\ s.t.\ \ \bar{x}_{xy}(0,\bar{x}_0)=\bar{x}_0\\ \hat{\tilde{x}}_{xy}(k+1)=A_{xy}\cdot\tilde{x}_{xy}(k)+B_{xy}(k)\cdot\hat{\tilde{u}}_{xy}(k)\end{gathered} \tag{119}$$

Where the cost function J_{xy} is given by:

$$J_{xy}(\bar{x}_0,\bar{u}_{xy})=\bar{x}_{xy}^T\cdot Q_{xy}\cdot\bar{x}_{xy}+\bar{u}_{xy}^T\cdot R_{xy}\cdot\bar{u}_{xy}+\bar{x}_{xy}^T(k+N_p/k)\cdot G_{xy}\cdot\bar{x}_{xy}(k+N_p/k) \tag{120}$$

where Q_{xy}, R_{xy} and G_{xy} are positive definite matrices. The rest of the matrices are computed in similar manner to the height predictive control scheme.

By minimizing the cost function J_{xy}, the control input $\bar{u}_{xy}$ is obtained, where $\bar{u}_{xy}(k/k)=\left[\bar{u}_x(k/k)\ \ \bar{u}_y(k/k)\right]^T$ and:

$$u_{xy}(k)=\bar{u}_{xy}(k/k)+u_{xyref}(k) \tag{121}$$

The error reference state-space vector $\hat{\tilde{x}}_{xyref}\in\mathbb{R}^{N_p}$ and the reference input $\hat{\tilde{u}}_{xyref}$ are obtained in the same way as in the height control mechanism.

Rewriting equation (107) for time instance k, we obtain:

$$\begin{aligned}u_x(t)&=\ \cos(\psi(t))\cdot\sin(\theta(t))\cdot cos(\varphi(t))+sin(\psi(t))\cdot sin(\varphi(t))\\ u_y(t)&=\ sin(\psi(t))\cdot sin(\theta(t))\cdot cos(\varphi(t))-cos(\psi(t))\cdot sin\,(\varphi(t)))\end{aligned} \tag{122}$$

By setting $\psi_{ref}=0$ and using (121), (122), the refences values for the roll angle φ_{ref} and pitch angle θ_{ref} are calculated; these are given as set point to the attitude control loop which will be described in the next subsection.

5.2.2.2 PID attitude control

The role of the position controller lies in generating the desired values for roll and pitch angles. These values are determined based on the current reference position of the trajectory. However, for the quadrotor system to effectively track the desired trajectory, it becomes essential to compute the control inputs U_1, U_2, U_3 and U_4. This intricate task is entrusted to the inner attitude controller, responsible for regulating the orientation angles φ and θ. These angles are steered using reference inputs supplied by the outer position controller. In this subsection, the design of the inner attitude loop adopts standard PID controllers, as discussed in section 4.4.

The rotational dynamic model given in section 4.3.1 by equation (**62**) can be expressed as:

$$\boldsymbol{I} \cdot \dot{\boldsymbol{\Omega}} = \boldsymbol{\tau} \tag{123}$$

where (123) can be rewritten using the notation of equations (104) and (105) as:

$$\begin{aligned} \boldsymbol{I_x} \cdot \ddot{\boldsymbol{\varphi}} &= \boldsymbol{U_2} \\ \boldsymbol{I_y} \cdot \ddot{\boldsymbol{\theta}} &= \boldsymbol{U_3} \\ \boldsymbol{I_z} \cdot \ddot{\boldsymbol{\psi}} &= \boldsymbol{U_4} \end{aligned} \tag{124}$$

The objective of the attitude controller is to ensure that Euler's angles φ, θ, ψ track the desired trajectory values $\varphi_{ref}, \theta_{ref}, \psi_{ref}$ asymptotically, by applying the suitable control signals U_2, U_3, U_4 to the quadrotor. Three independent PID controllers are utilized to control the angular accelerations of the quadrotor, one for each Euler angle.

By applying forward Euler method on continuous PID controllers, each controller can be expressed in velocity form as follows

$$\begin{aligned} \boldsymbol{u_j(k)} = {} & \left(\boldsymbol{K_{Pj}} + \boldsymbol{K_{Dj}} \cdot \boldsymbol{N_j}\right) \cdot \boldsymbol{e_j(k)} \\ & + \left(\boldsymbol{N_j} \cdot \boldsymbol{\Delta t} \cdot \boldsymbol{K_{Pj}} + \boldsymbol{K_{Ij}} \cdot \boldsymbol{\Delta t} - \mathbf{2} \cdot \boldsymbol{K_{Dj}} \cdot \boldsymbol{N_j} - \mathbf{2} \cdot \boldsymbol{K_{Pj}}\right) \cdot \boldsymbol{e_j(k-1)} \\ & + \left(\boldsymbol{K_{Pj}} - \boldsymbol{K_{Pj}} \cdot \boldsymbol{N_j} \cdot \boldsymbol{\Delta t} - \boldsymbol{K_{Ij}} \cdot \boldsymbol{\Delta t} + \boldsymbol{K_{Ij}} \cdot \boldsymbol{N_j} \cdot \boldsymbol{\Delta t^2} \cdot \boldsymbol{K_{Dj}} \cdot \boldsymbol{N_j}\right) \\ & \cdot \boldsymbol{e_j(k-2)} - \left(\boldsymbol{N_j} \cdot \boldsymbol{\Delta t} - \mathbf{2}\right) \cdot \boldsymbol{u_j(k-1)} - \left(\mathbf{1} - \boldsymbol{N_j} \cdot \boldsymbol{\Delta t}\right) \\ & \cdot \boldsymbol{u_j(k-2)} \end{aligned} \tag{125}$$

The index j with $j = 1,2,3,4$ describes the Euler's angles φ, θ, ψ, while r_j describes the reference angles $\varphi_{ref}, \theta_{ref}, \psi_{ref}$, of the jth channel. The control signal is denoted by $u_j(k)$ and the error signal by $e_j(k)$ at each discrete time instant k. The error signal is the difference between the actual value $\eta_j(k)$ and referencer_j. K_{Pj}, K_{Ij}, K_{Dj} are the controller proportional, integral and derivative gains and N_j is the low pass filter coefficient of the derivative term of the PID controller.

By using equations (123), (124), the control inputs U_2, U_3, U_4 can be estimated through the following equations:

$$\begin{aligned} \boldsymbol{U_2} &= \boldsymbol{I_x \cdot u_\varphi} \\ \boldsymbol{U_3} &= \boldsymbol{I_y \cdot u_\theta} \\ \boldsymbol{U_4} &= \boldsymbol{I_z \cdot u_\psi} \end{aligned} \tag{126}$$

An important addition to the proper design of the control strategy takes into consideration the saturation bounds for each one of the control signals. With the primary objective of limiting and controlling the quadrotor's thrust force, many saturation-constraint based methodologies have been developed [154]. The enhancement in quadrotor control design arising from the introduction of saturated limits for control signals can be elucidated as follows. Firstly, both the total thrusts U_1 and control torque $\boldsymbol{\tau}$ are confined within certain bounds due to physical constraints imposed by the electrical motors. Secondly, the control inputs need to operate within a certain range that facilitates smooth and uninterrupted trajectory tracking. Lastly, enforcing boundaries in input signals mitigates flight scenarios that could lead to the quadcopter losing stability. Based on these considerations, eight input saturation limits are enforced for the four control inputs U_1, U_2, U_3, U_4 encompassing both upper and lower bounds. This strategic positioning of bounds on control inputs ensures the quadrotor’s position and attitude control operates within defined limits, as shown in Figure 19.

5.2.3 Control scheme tuning using a CPSO-based optimizer

The standard PSO algorithm could be applied for tuning the proposed quadrotor control scheme, but in this case, the elements of each particle should contain all the controller parameters to be tuned, which should be optimized concurrently; Figure 20 shows the corresponding particle structure in this case.

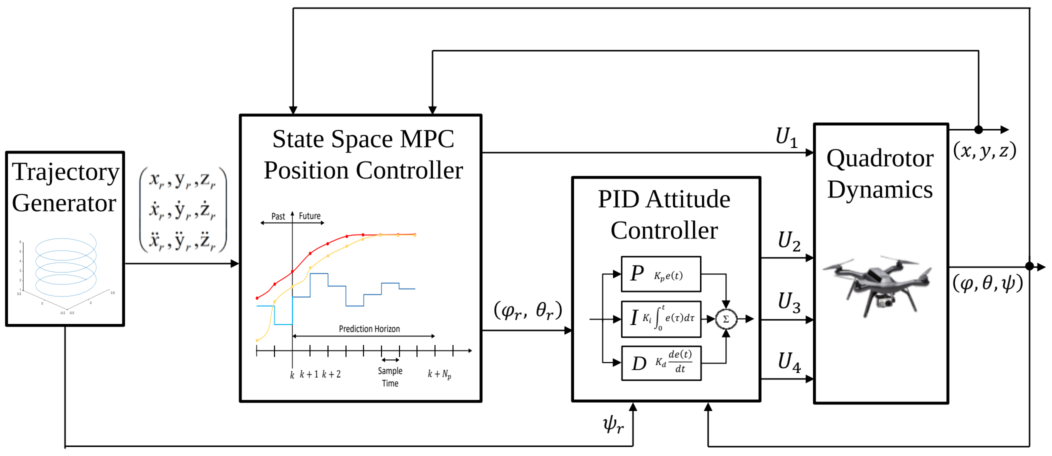

Figure 19: Overall quadrotor control structure

$$\underbrace{Q_{z_1}\ Q_{z_2}\ R_z\ \ G_{z_1}\ G_{z_2}\ Q_{xy_1}\ Q_{xy_2}\ R_{xy}\ G_{xy_1}\ G_{xy_2}\ N_c\ N_p}_{\text{MPC control parameters}}\ \underbrace{K_{P_\theta}\ K_{I_\theta}\ K_{D_\theta}\ N_\theta\ K_{P_\psi}\ K_{I_\psi}\ K_{D_\psi}\ N_\psi\ K_{P_\varphi}\ K_{I_\varphi}\ K_{D_\varphi}\ N_\varphi\ U_{1_B}\ U_{2_B}\ U_{3_B}\ U_{4_B}}_{\text{PID control parameters}}$$

Figure 20: Overall particle structure in the case of standard PSO

However, there is a high number of parameters involved, pertaining to the two control subsystems. From an optimization perspective, the increased dimensionality of the search space makes it difficult to discover even a good suboptimal solution. On the other hand, a distinction can be made between two different parts in each particle, where each part controls a different mechanism of the integrated control scheme. To be more specific, the first part contains the parameters from the position control strategy of the MPC controller, while the second one the parameters from the attitude control through the implementation of PID controllers. The existence of two different groups of parameters within the particle, leads to the idea of using two cooperative swarms. As previously stated in subsection 2.4, the CPSO framework is employed in this context to fine tune the quadrotor’s control parameters. This advancement permits the simultaneous optimization of both MPC and PID parameters. Throughout the optimization procedure, the two distinct swarms independently evolve while maintaining effective information exchange. This integrated approach serves to proficiently optimize the quadrotor’s control parameters, further enhancing its performance. The particles for the MPC swarm which contain the position control tuning parameters are denoted as P_1x_i, while the particles of the PID swarm containing the attitude control tuning parameters are denoted as P_2x_i.

The following equations display the parameters controlled by each one of the swarms P_1 and P_2:

$$\boldsymbol{P_1x_i = [Q_{z1}\ Q_{z1}\ R_z\ G_{z1}\ G_{z2}\ Q_{xy1}\ Q_{xy2}\ R_{xy}\ G_{xy1}\ G_{xy2}\ N_c\ N_p\ ]} \tag{127}$$

$$\boldsymbol{P_2x_i = [K_{P\theta}\ K_{I\theta}\ K_{D\theta}\ N_\theta\ K_{P\varphi}\ K_{I\varphi}\ K_{D\varphi}\ N_\varphi\ K_{P\psi}\ K_{I\psi}\ K_{D\psi}\ N_\psi\ U_{1B}\ U_{2B}U_{3B}U_{4B}]} \tag{128}$$

For each swarm the position and velocity vector are updated according to the following equations:

$$\boldsymbol{P_kv_{i,j}(t+1) = w \cdot P_kv_{i,j}(t) + c_1 \cdot r_{1,i} \cdot [P_ky_{i,j}(t) - P_kx_{i,j}(t)] + c_2 \cdot r_{2,i} \cdot [P_k\hat{y}_j(t) - P_kx_{i,j}(t)]} \tag{129}$$

$$\boldsymbol{P_k x_{i,j}(t+1) = P_k x_{i,j}(t) + P_k v_{i,j}(t+1)} \quad (130)$$

where the ith particle in dimension j is represented as $P_k x_{i,j}(t)$ for the swarm k, the speed for particle i in dimension j is represented as $P_k v_{i,j}(t)$, the personal best position of the ith particle in dimension j is denoted as $P_k y_{i,j}(t)$, and $P_k \hat{y}_j(t)$ represents the best particle position among all particles for the jth dimension.

The velocity of each particle $P_k v_{i,j}(t)$ is controlled by a clamping constant $P_k v_{max}$ that regulates the maximum velocity update to a defined range of [-$P_k v_{max}$, $P_k v_{max}$]. The addition of the clamping constant is used to control the exploration-exploitation trade-off, by affecting the particles' ability to explore a small, or big part of the search space.

As each swarm is connected to a specific part of the solution vector of the optimization procedure, the right cooperation between the agents is essential in order to calculate the fitness function. The implementation of this task is feasible by using a context vector. This vector provides a suitable context in which the individuals from each swarm can be evaluated. To form the context vector, the global best particle $\hat{y}$ from one swarm, is combined with each particle $x_i, i = 1, \dots, s$ from the other swarm. Therefore, in order to calculate the fitness functions for swarm P_k, all the respective particles should be concatenated with the global best particle of the complementary swarm $P_{(2k)^{2-k}}\hat{y}$. This concatenation leads to the full set of parameters for both controllers, which is used to simulate the quadrotor flight. The fitness function is calculated as the sum of the Euclidean distances (SED) between the reference trajectory and the actual quadrotor trajectory during the simulation flight, based on the following equation:

$$\boldsymbol{f = \sum_{t_s=1}^{t_{end}} \sqrt{\sum_{i=1}^{3} \big(q_i(t_s) - p_i(t_s)\big)^2}} \quad (131)$$

where $t_s = 1, \dots, t_{end}$ is the running time of the flight simulation, $q(t_s) = [x_{ref}, y_{ref}, z_{ref}]$ is the vector containing the Cartesian coordinates of the reference altitude and attitude of the quadcopter at time t_s, $p(t_s) = [x, y, z]$ is the vector containing the Cartesian coordinates of the actual altitude and attitude achieved by the quadrotor at time t_s, and t_{end} is the end time of the simulation.

The personal best position $P_k y_i(t)$ for each particle i of each swarm k depicts the best result found for this particle up to iteration t:

$$\boldsymbol{P_k y_i(t+1)} = \begin{cases} \boldsymbol{P_k y_i(t)}, & \boldsymbol{if\ f(P_k x_i(t+1), P_{(2k)^{2-k}}\hat{y}(t)) \geq f(P_k y_i(t), P_{(2k)^{2-k}}\hat{y}(t-1))} \\ \boldsymbol{P_k x_i(t+1)}, & \boldsymbol{if\ (P_k x_i(t+1), P_{(2k)^{2-k}}\hat{y}(t)) < f(P_k y_i(t), P_{(2k)^{2-k}}\hat{y}(t-1))} \end{cases} \tag{132}$$

The global best position of each particle f each particle i for each swarm is updated using:

$$\boldsymbol{P_k\hat{y}_i(t)} = \mathbf{arg}\min_{\boldsymbol{P_k y_i(t)}} \boldsymbol{f}\left(\boldsymbol{P_k y_i(t), P_{(2k)^{2-k}}\hat{y}(t-1)}\right), \quad \boldsymbol{1 \leq i \leq s} \tag{133}$$

Figure 21 depicts a schematic overview of the MPC and PID swarms working together towards optimizing the tuning parameters.

The described algorithm has two important advantages with respect to the optimization procedure when used for tuning the controller parameters. The first one is associated with the fitness function evaluation after each separate group of controller parameters has been updated. This leads to a finer-grained assignment avoiding the classic two steps forward - one step back problem often encountered in standard PSO. The second advantage is related to the increased combinations of different individuals from different swarms, which boost the solution diversity.

5.2.4 Experimental Set-up

This section aims to demonstrate the effectiveness of the proposed tuning methodology through simulated experiments. The proposed control scheme was fully programmed in MATLAB®

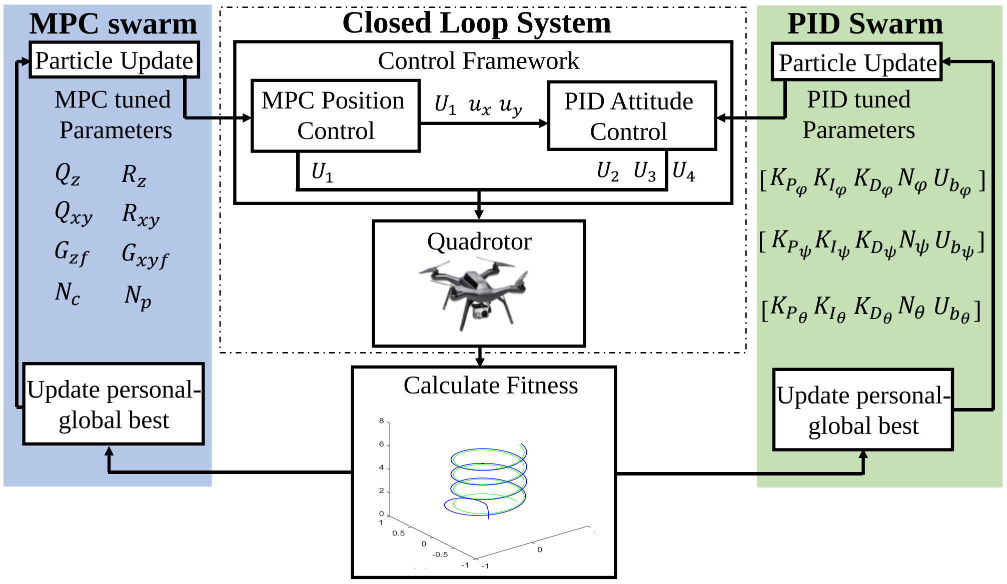

Figure 21: Overview of the two cooperative swarms, working towards optimizing the tuning

environment and the simulations were performed by solving the system of ODEs (104) and (105) at each time instant to calculate the quadrotor's position and attitude.

In order to perform the simulations, the values shown in Table 11 were assigned to the quadrotor parameters.

The CPSO algorithm designed was evaluated through the tuning of the quadrotor control framework across two scenarios, utilizing distinct reference spatial trajectories. These trajectories exhibit varying geometrical characteristics within the 3-D space, necessitating diverse tuning strategies for optimal trajectory tracking. A robustness assessment was additionally conducted by tuning the controller for one trajectory and subsequently applying it to an unfamiliar trajectory. This test aimed to assess the optimizer's capacity to provide robust quadrotor control parameters across different scenarios.

Table 11: Quadrotor parameters

Symbol	**Description**	**Value**	**Unit**
l	Arm length	0.24	m
m	Mass of quadrotor	1	kg
I_x	Body moment of inertia around *x* axis	8 10^{-3}	N m s^2
I_y	Body moment of inertia around *y* axis	8 10^{-3}	N m s^2
I_z	Body moment of inertia around *z* axis	14.2 10^{-3}	N m s^2
J_r	Rotational moment of inertia	1.08 10^{-6}	N m s^2
b	Thrust coefficient	54.2 10^{-6}	N s^2
d	Drag coefficient	1.1 10^{-6}	N m s^2
K_t	Aero dynamical damping translational matrix	diag(0.048,0.11,0.046)	N m s^2
K_r	Aero dynamical damping rotational matrix	diag(0.03,0.03,0.01)	N m s^2
g	Acceleration due to gravity	9.81	m s^2

Table 12: Operational parameters for the proposed cooperative algorithm

Parameter	**Symbol**	**Value**
Population	P	25
Inertia coefficient	w	0.5
Nostalgia coefficient	c_1	1.65
Envy coefficient	c_2	1.65
Velocity clamping coefficient	$P_k v_{max}$	0.2
Maximum number of generations	Gen	400
Convergence constant	ε	0.2

To facilitate comparison, two distinct metaheuristic search methods were employed to fine-tune the MPC-PID framework in addition to the cooperative PSO algorithm. The initial approach involves a classical PSO algorithm, elaborated upon 2.3. The second method incorporates a genetic algorithm that employs binary encoding, scattered crossover, and Gaussian mutation techniques. Furthermore, the performance of the MPC-PID framework was also compared to a standard PID-based control scheme for trajectory tracking [155]. This scheme also utilizes two control loop parts, namely the inner loop control (ILC) and the outer loop control (OLC), where ILC is responsible for the Euler angle control that regulates the attitude of the vehicle, while OLC regulates the position in the x-y-z plane; however, in this case both control subsystems are based on PID control, while tuning is performed manually.

The operational parameters used by COOP-PSO on all experiments, are given in Table 12. Parameter selection for all comparison approaches was based on suggestions in literature, along with trial and error.

5.2.5 Results and Discussion

The first trajectory chosen to evaluate the proposed cooperative optimizer namely the spiral trajectory, displays simple geometrical characteristics and constitutes a standard benchmark to demonstrate a quadrotors' ability to fly automatically.

$$\begin{aligned} x_{ref}(t) &= 0.5\cdot cos(0.5\cdot t) \\ y_{ref}(t) &= 0.5\cdot sin(0.5\cdot t) \\ z_{ref}(t) &= 1+\frac{t}{10} \end{aligned} \tag{134}$$

The second trajectory (135), is termed “composite” due to its composition from three distinct sub-trajectories: a spiral [156], a helical cone [157] and a linear dependent [82] trajectory. The amalgamation of trajectories exhibiting diverse characteristics renders the optimization process a challenging endeavor for the optimizer.

$$x_{ref}(t) = \begin{cases} 0.8\cdot \cos(0.6\cdot t) & t\in[0,22] \\ 6.52-0.26\cdot t & t\in(22,38] \\ -6+(6.2-0.1\cdot t)\cdot cost(0.5\cdot t) & t\in(38,65] \\ -6.166 & t\in(65,80] \end{cases}$$

$$y_{ref}(t) = \begin{cases} 0.8\cdot \sin(0.6\cdot t), & t\in[0,22] \\ 0.367\cdot t-7.62 & t\in(22,38] \\ -6+(6.2-0.1\cdot t)\cdot sint(0.5\cdot t) & t\in(38,65] \\ 5.73 & t\in(65,80] \end{cases} \tag{135}$$

$$z_{ref}(t) = \begin{cases} 0.1\cdot t+2, & t\in[0,22] \\ 0.22\cdot t-0.68 & t\in(22,38] \\ 12.5-0.125\cdot t & t\in(38,65] \\ 8.7-0.066\cdot t & t\in(65,80] \end{cases}$$

The simulated results also contain a robustness test for the proposed cooperative optimization tuning technique. In this case, the tuning parameters generated for the spiral trajectory simulation runs, were used to guide the quadrotor so as to follow a new trajectory (136), namely the complex helical [94]. The geometry of this trajectory exhibits an elliptic trail with complex characteristics.

$$\begin{aligned} x_{ref}(t) &= cos(0.5\cdot t)-cos^3(0.4\cdot t) \\ y_{ref}(t) &= sin(0.2\cdot t)-sin^3(0.4\cdot t) \\ z_{ref}(t) &= 0.3\cdot t \end{aligned} \tag{136}$$

Due to the stochastic nature of all metaheuristic optimizers, which produce a different result for each run, 20 runs were performed for each trajectory for the cases of COOP-PSO, GA and PSO. To validate the results, a *t*-test between the proposed cooperative strategy and each of the comparison methods was implemented. In the cases of GA and PSO the null hypothesis defines that the results from COOP-PSO and the rivaling method are generated from normal distributions with equal means, while in the case of PID that the results from COOP-PSO are generated from a normal distribution with a mean equal to the fitness generated by the PID approach.

Based on above, Tables 13-15 present the results for all methods, including the mean fitness values and standard deviations, the best fitness values, and *p*-values resulting from the *t*-tests for the spiral, composite and complex helical trajectories, respectively. The response for the position variables x, y, z and the attitude angles φ, θ, ψ for the case of the spiral trajectory is shown in Figure 22, while the flight of the quadrotor in the 3-D space is visually depicted in Figure 23. Figures 24-25 and Figures 26-27 present the corresponding results for the cases of the composite trajectory and the complex helical trajectory where the robustness case was tested, respectively. For each one of the three different trajectories, a randomly selected simulation out of the 20 runs is depicted in the figures.

Table 13: Tuning performance metrics for the spiral trajectory

	Fitness average	Fitness standard deviation	Best Fitness	*p*-value
COOP-PSO	36.59	3.111	32.17	-
GA	39.80	2.564	34.16	0.001
PSO	38.47	2.052	35.06	0.029
PID	-	-	40.86	6.67E-06

Table 14: Tuning performance metrics for the composite trajectory

	Fitness average	Fitness standard deviation	Best Fitness	*p*-value
COOP-PSO	89.47	4.075	82.49	-
GA	97.12	5.076	84.86	5.99E-06
PSO	94.97	6.082	83.94	0.002
PID	-	-	94.47	2.72E-05

Table 15: Robustness metrics for the complex helical trajectory

	Fitness average	Fitness standard deviation	Best Fitness	*p*-value
COOP-PSO	106.81	4.777	98.75	-
GA	112.25	5.960	101.89	0.003
PSO	110.51	3.356	100.19	0.007
PID	-	-	10E+05	0

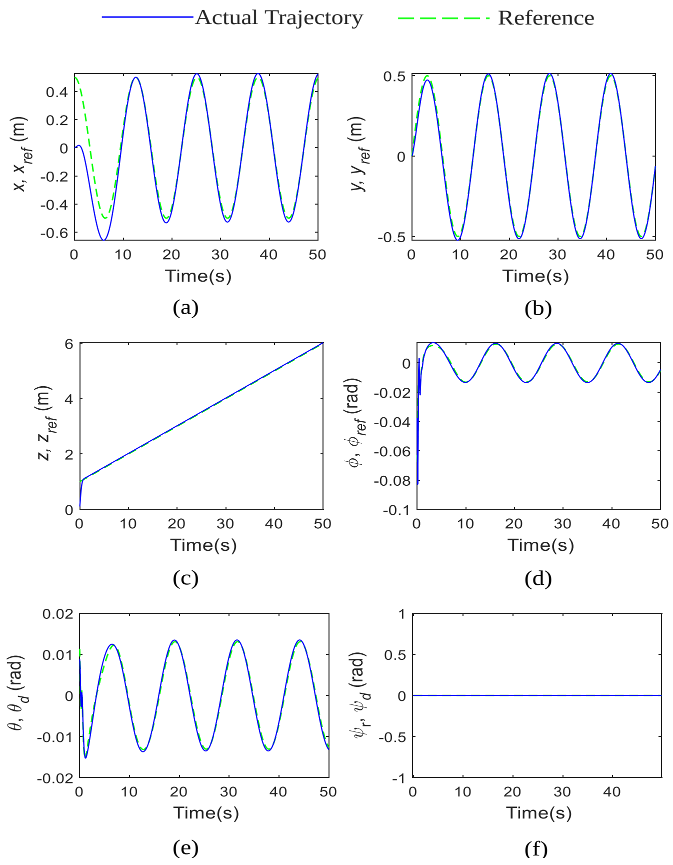

Figure 22: Spiral trajectory simulation results for (a) position *x*, (b) position *y*, (c) position *z*, (d) roll angle *φ*, (e) pitch angle *θ*, (f) yaw angle *ψ*

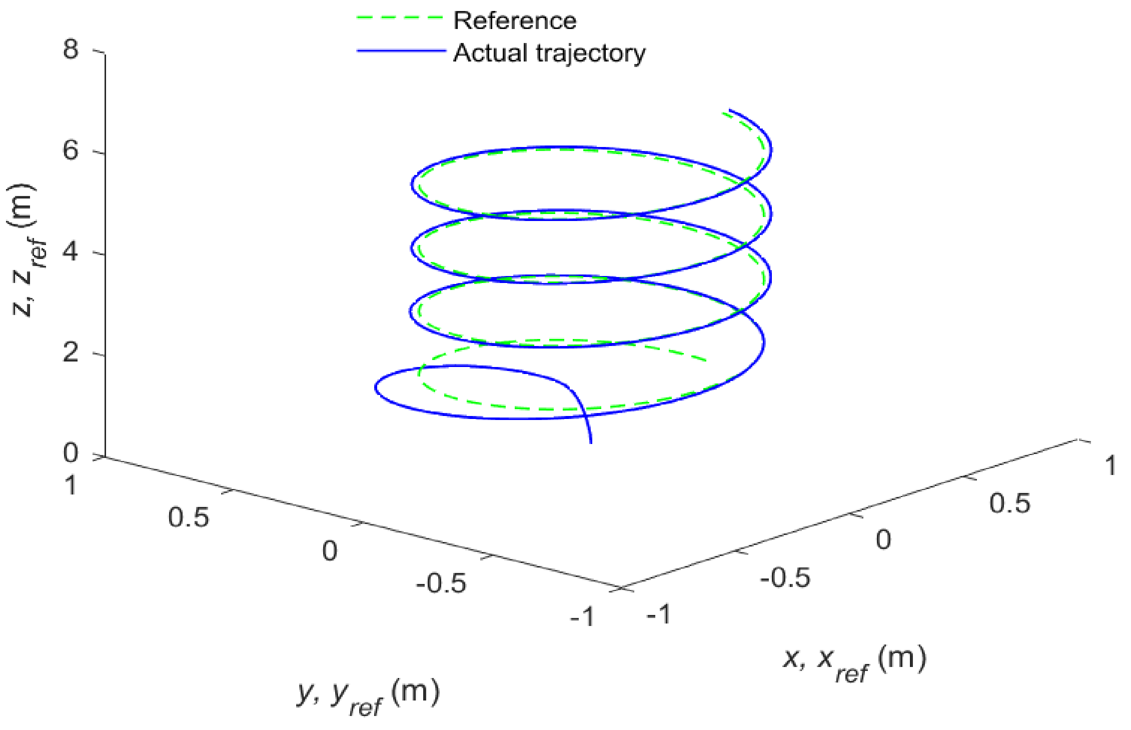

Figure 23: 3-D simulation results for the spiral trajectory

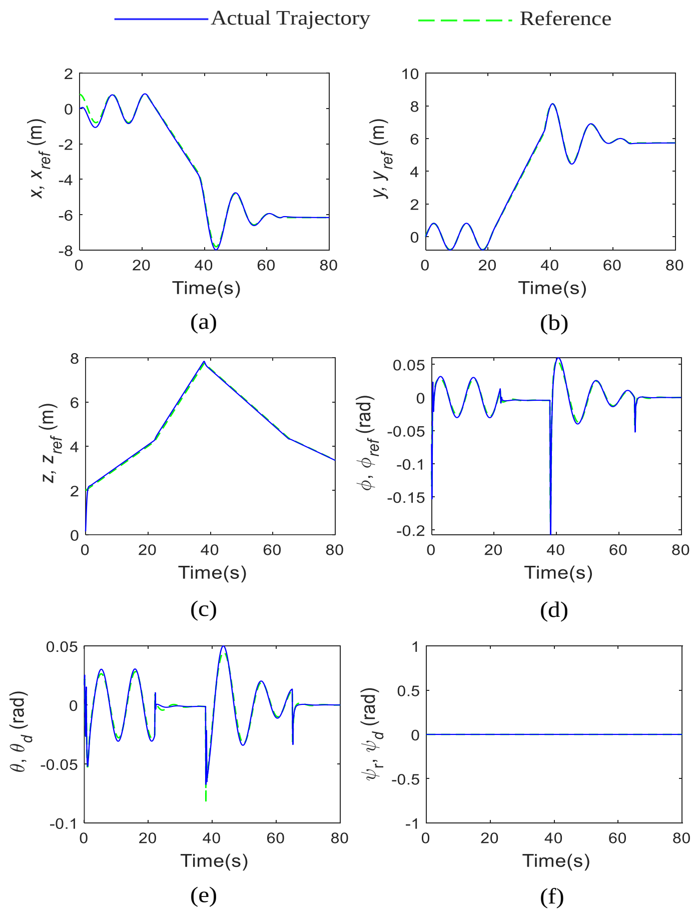

Figure 24: Composite trajectory simulation results for (a) position *x*, (b) position *y*, (c) position *z*, (d) roll angle φ, (e) pitch angle θ, (f) yaw angle ψ

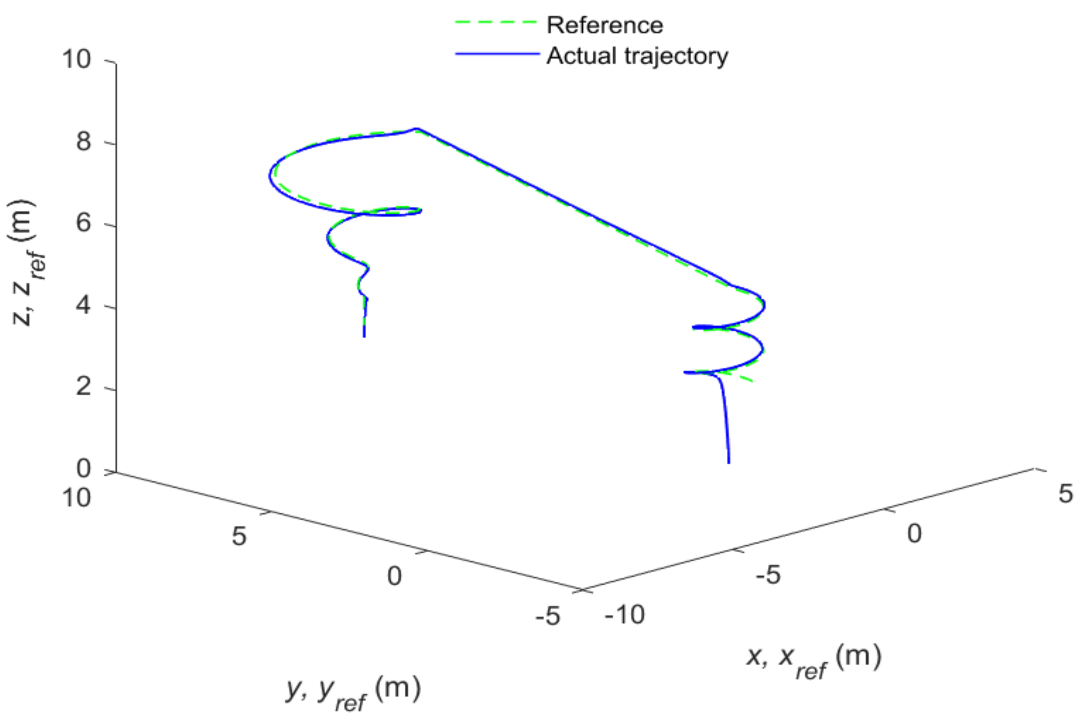

Figure 25: 3-D simulation results for the composite trajectory

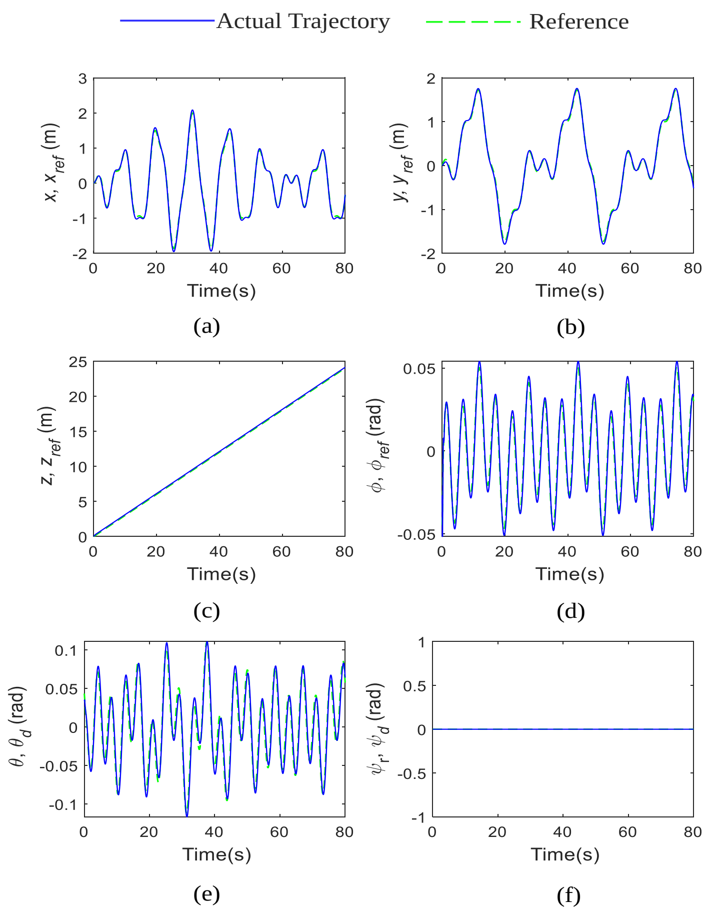

Figure 26: Complex helical simulation results for (a) position *x*, (b) position *y*, (c) position *z*, (d) roll angle *φ*, (e) pitch angle *θ*, (f) yaw angle *ψ*

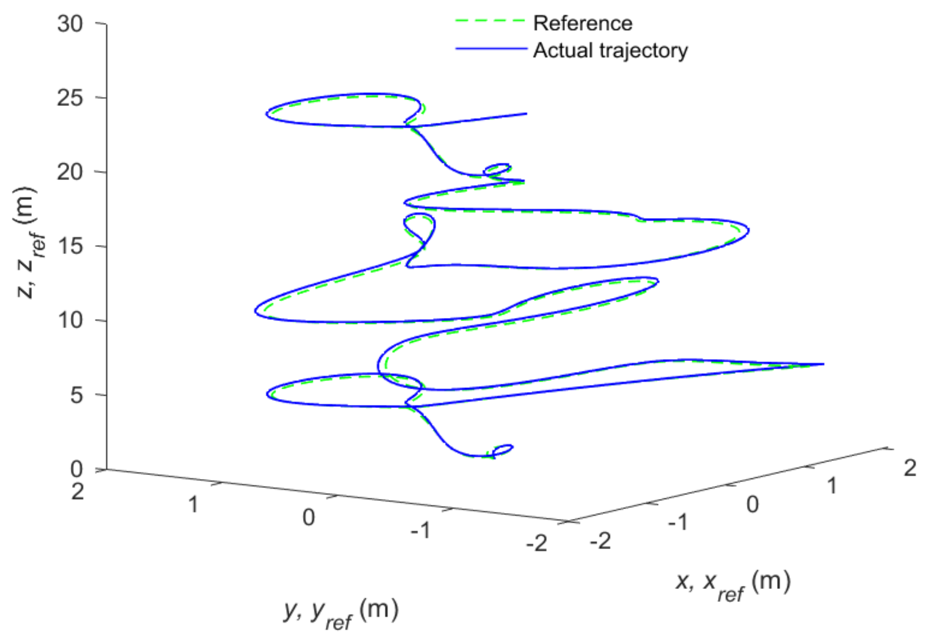

Figure 27: 3-D simulation results for the complex helical trajectory

As it can be seen by the figures, the attitude and position control schemes satisfy their respective goals and the overall control framework manages to track accurately the reference trajectory in all cases. The efficiency of the proposed scheme is highlighted by the scores in Tables 13-15 where the cooperative PSO-tuned controller results to superior performance metrics over its rivals in all the tested trajectories.

To be more specific, as far as the spiral trajectory is concerned, the proposed approach outperforms all the methods used for comparison, producing the best run, which results to the minimum offset error from the reference trajectory. More importantly, it also outperforms its rivals in terms of the average result produced from the 20 runs, with a statistical significance higher than 97% according the produced *p*-values.

The composite trajectory forms a more demanding test compared to the spiral one, as it essentially comprises three different sub-trajectories, and thus provides a strong indication of the ability of the tuning method to cope with reference signals that exhibit different geometrical characteristics. In this respect, the COOP-PSO algorithm produces superior results compared to the remaining methods, considering either the best produced run, or the average from the 20 runs. Regarding the later, the superiority of the proposed approach is confirmed with a statistical significance of 99%.

In the last tested trajectory, namely the complex helical one, the objective was to evaluate the robustness of the different tuning methods; this can be assessed by asking the controllers to follow a trajectory different than the one used for tuning their parameters. To this end, the control parameters that were generated by tuning on the spiral trajectory, were then applied to a situation where the quadrotor was asked to follow the complex helical trajectory. The results confirm the superiority of the proposed cooperative technique in delivering robust design parameters, both in terms of the best run and the average of 20 runs. The *p*-values produced from the *t*-test show that this conclusion is statistically significant with a confidence level of 99% or higher. From a practical point of view, this is an important result, as in a real-world situation it is not expected that the quadcopter will be tuned beforehand for all possible trajectories that it may be asked to track – most probably at some point during its operation the quadcopter will need to perform moves that are not part of the trajectories used for tuning it.

It should be noted that, besides producing controllers with better tracking abilities and lower offset error, the proposed approach also manages to produce consistent results, as indicated by the standard deviation values from the average of the 20 runs, which are kept within a reasonable range throughout all the scenarios. This outcome comes in contrast with the metrics provided by the PSO and GA algorithms, which exhibit variations in standard deviation values, depending heavily on the geometry of the respective trajectory. This is a strong indication that the proposed approach yields consistent results, even for spatial trajectories with different geometrical attributes.

Among the rest of the competing methods, PSO seems to produce slightly better results compared to standard GA on average, albeit noticeably worse than the proposed approach. The standard PID control scheme on the other hand produces a tracking error which is considerably higher than the rest of the methods for all three tested cases; especially in the third case evaluating the robustness of the methods, the tracking error of the PID scheme is orders of magnitude higher, essentially failing to follow the reference trajectory. This can be explained by the superiority of the MPC scheme employed by the rest of the methods for position control, over the standard PID technique.

The success of the proposed scheme can be attributed to the cooperation between the two different swarms which exchange information while evolving simultaneously. Some insight into this cooperation can be obtained through Figure 28, which depicts the respective fitness value per generation for the MPC and PID swarms, in the spiral and composite trajectories. It can be seen that the lead in terms of lower fitness value between the two swarms may change many times, as each swarm helps the other to evolve through their mutual cooperation.

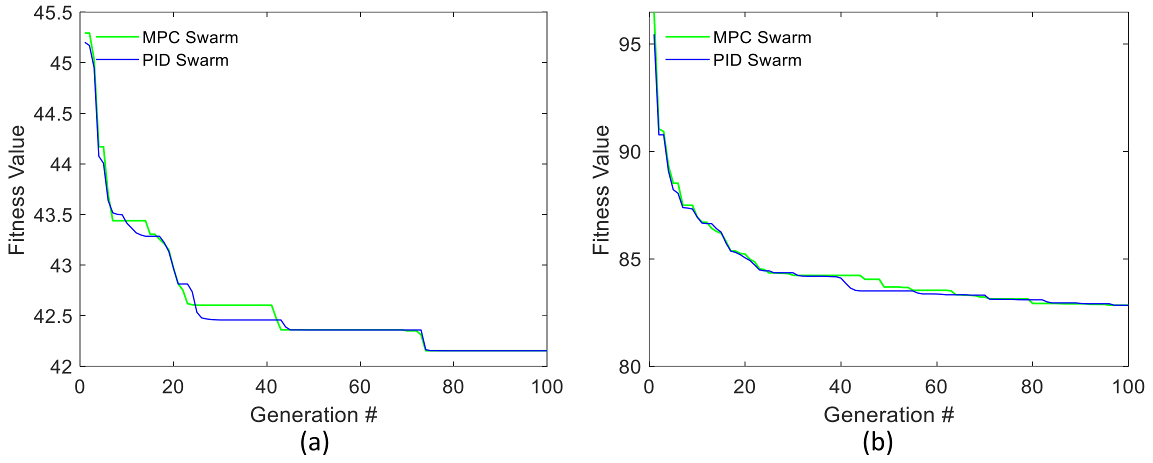

Figure 28: Change of best fitness value per generation for the MPC and PID swarms in the case of (a) spiral trajectory, (b) composite trajectory

5.3 Qualitative comparison of developed trajectory tracking control methods

Within this section, a qualitative examination unfolds, highlighting the distinctions between two control methodologies developed for quadrotor trajectory tracking, as discussed in sections 5.1 and 5.2, i.e., the RBF-backstepping controller and the MPC-PID control scheme. Although a quantitative comparative analysis poses challenges given the inherent complexities and unique traits of each approach, the objective is to clarify their individual advantages, limitations, and suitability for this application.

Starting with the RBF-backstepping controller, it's crucial to highlight its inherent qualities. The controller is known for its asymptotic stability, designed in alignment with Lyapunov stability principles, ensuring the system's gradual convergence towards the desired state. An additional advantage lies in its ability to effectively address model uncertainties or disturbances through the integration of RBF models, significantly improving tracking performance, especially in scenarios where model accuracy is a concern. Furthermore, the step-by-step design process, exemplified in Section 5.1.2.2, offers a structured and methodical approach for implementation.

On the other hand, the MPC-PID controller stands out for its ability to establish a control strategy through the resolution of an online optimal control problem, as elaborated in Section 5.2.2.1. Importantly, this approach doesn't require an in-depth comprehension of the dynamics of the specific systems, which has the advantageous effect of simplifying the construction of the control framework and streamlining the implementation process. Furthermore, the careful selection of the CPSO algorithm becomes pivotal in the context of quadrotor control, given that the MPC-PID control method involves multiple controllers and, consequently, entails a significant number of tuning parameters.

Despite the merits, both controllers exhibit limitations. The MPC-PID controller relies on a linear model within the OCP, which may not accurately capture the inherent nonlinearities of the quadrotor model. This drawback can impact control performance, especially in situations where nonlinear effects are significant. Additionally, the process of formulating and solving an online problem at each discrete time step can be computationally intensive, which can pose a significant drawback when implementing this approach on real quadrotors.

On the contrary, the RBF-backstepping controller, although relying on a classic PSO technique for controller tuning, derives its enhanced performance primarily from the incorporation of RBF models within the backstepping control framework. This integration significantly enhances tracking accuracy and robustness. However, it’s worth noting that this controller’s effectiveness is intricately linked to the unique dynamics of the quadrotor it was designed for, which restricts its applicability to various UAV platforms. Additionally, it’s imperative to emphasize that the backstepping control law is primarily tailored for application in strict feedback systems, which represents a notable limitation.

In summary, the choice between the RBF-backstepping and MPC-PID controllers should be made based on specific application requirements and considerations. The RBF-backstepping controller stands out for its ability to provide asymptotic stability and manage model uncertainties through a clear control law expression. On the other hand, the MPC-PID controller offers a control law derived from solving an OCP, and can handle constraints. It's worth emphasizing that the CPSO framework is particularly valuable for efficiently fine-tuning a significant number of control parameters, especially when dealing with control schemes that involve multiple controllers, as is the case with the MPC-PID controller. Hence, the decision should be based on whether priorities lie in achieving stability, utilizing explicit control laws, managing constraints, or optimizing tuning for a high number of control parameters within the specific application. A meticulous assessment of the system's nonlinear characteristics and the operational environment becomes essential in the process of choosing the most appropriate controller for quadrotor trajectory tracking assignments.

6. Development of adaptive nonlinear MPC control scheme using online RBF networks

Within this chapter, we delve into the challenge of devising a control methodology for time-varying systems, incorporating the use of adaptive online models. To address this challenge, specific focus will be on the utilization of nonlinear control techniques, leveraging their adaptability and effectiveness in managing the intricacies of time-varying systems. Specifically, the proposed control scheme, will be applied towards simulation-based applications, with instances including the control of a NARX system and a CSTR reactor, dynamically adjusting its model's parameters in response to system changes, thus ensuring robust and effective control evolving dynamics.

6.1 Introduction to online adaptive models and their vital role in controlling time-varying systems

In the context of time-varying systems, the challenge of control is particularly challenging. These kinds of systems demonstrate dynamic behaviors that evolve over time in reaction to varying external factors or operational conditions. While static systems find sufficiency in fixed control strategies, time-varying systems necessitate dynamic solutions, and at the heart of this requirement lies the pivotal role of online adaptive models.

Online adaptive models, the foundation of effective control in time-varying systems, offer dynamic and responsive structures. These models have the unique ability to learn and adapt in real-time, which renders them highly suitable for situations characterized by frequently shifting and unpredictable dynamics. Online adaptive models excel at adjusting control parameters as new data becomes available, ensuring the control system is able to complete its regulation task throughout the system's continually changing conditions.

To highlight the importance of online adaptive models, one needs to evaluate real-world scenarios within the domain of time-varying systems. Regarding autonomous robotic systems, adaptive algorithms enable robots to navigate shifting terrains or respond to unforeseen damage, ultimately guaranteeing unwavering performance [158], [159]. In the domain of financial markets, adaptive trading algorithms adjust strategies to go align accordingly with the perpetually changing market conditions, maximizing returns [160]. Moreover, in the aerospace field, online models enable aircrafts to maintain stable flight amidst wind turbulence conditions [161].

The necessity for online adaptive models occurs from the pivotal requirement to efficiently control time-varying systems. These models inherently possess the capacity to adjust and acquire knowledge as dynamics evolve, proving significant in achieving the required performance and stability tasks in time-varying systems. Within this context, the adaptive symmetric fuzzy means algorithm emerges [162] as a powerful tool.

6.2 The adaptive symmetric fuzzy means algorithm

The adaptive symmetric fuzzy means (ASFM) algorithm relies on the fuzzy partitioning technique of the input space, as detailed in subsection 2.2.2 of Chapter 2. It is utilized to identify when, and which specific centers should be added to, or removed from the network's hidden layer. To this end the algorithm features a dual-level adaptation: the adjustment of connection weights between the hidden layer and the output layer, and the altercation of the hidden layer's structure.

Upon the initialization of the algorithm, the domain of each input variable $i,\ i = 1, \dots, N$ requires to be partitioned into c_i symmetrical triangular fuzzy sets. The parameter L shows the number of nodes in the hidden layer and is initialized as zero. Furthermore, the operational parameters of the algorithm necessitate definition, including: the number N_d, representing consecutive times steps that a center is not assigned to an input example data before it is removed from the hidden layer of the network; the size N_s defining the moving time window used for retaining input-output examples; and the forgetting factor λ utilized in the Recursive Least Squares (RLS) method.

Upon receiving the first input example data $[\boldsymbol{x}(\mathbf{1}), y(1)]$ the algorithm initializes the parameter L to 1 and determines the fuzzy subspace $\boldsymbol{A^l} = [A^l_{1,j_1}, A^l_{2,j_2}, \dots, A^l_{N,j_N}]$ that is nearer to the input example, according to the minimum distance criterion:

$$\boldsymbol{j_i} = \underset{\mathbf{1 \le j \le c_i}}{\mathbf{arg\,max}} \left[\boldsymbol{\mu_{A_{i,j}}}\left(\boldsymbol{x_i}(\mathbf{1})\right)\right], \boldsymbol{i} = \mathbf{1}, \dots, \boldsymbol{N} \tag{137}$$

When the first hidden node of the first fuzzy subspace is created, the algorithm commences a dynamic center location matrix $\boldsymbol{C} \in \mathbb{R}^{L \times N}$, which is utilized to store the center location at each time instance, and the Activation History Vector (AHV) $\boldsymbol{h}$. The center location matrix contains at each time instance the centers of the hidden layer nodes and its dimension is $L \times N$. The size of AHV is equal to the number of hidden nodes and contains the last time instant that an input example was assigned to each fuzzy subspace.

After the generation of the first hidden node, the dimension $\boldsymbol{C}$ is $1 \times N$ and $\boldsymbol{h}$ contains only one element. To be more specific, the center of the initial hidden node is assigned to the single row of matrix of $\boldsymbol{C}$:

$$\boldsymbol{C^1} \leftarrow \boldsymbol{a^1} \tag{138}$$

and the value of the first element of $\boldsymbol{h}$ is initialized to1:

$$\boldsymbol{h}^{\mathbf{1}}(\mathbf{1}) = \mathbf{1} \tag{139}$$

Afterwards, the algorithm calculates the vectors $\boldsymbol{z}$, $\boldsymbol{w}$ and matrix $\mathbf{P}$, which are calculated at each time instant. The vector $\boldsymbol{z} \in \mathbb{R}^{L\times 1}$ includes the responses of the hidden layer nodes, while the vector $\boldsymbol{w} \in \mathbb{R}^{L\times 1}$ contains the connecting weights between the hidden and output layer of the RBF network. Lastly matrix $\mathbf{P} \in \mathbb{R}^{L\times L}$ is the inverse of the covariance matrix, used in the RLS algorithm.

Naturally, when the first data input example becomes available, all the aforementioned vectors and matrices consist of only one element, and their calculation proceeds as follows:

$$\boldsymbol{z}(\mathbf{1}) = \boldsymbol{g}(\|\boldsymbol{x}(\mathbf{1}) - \boldsymbol{C}^{\mathbf{1}}\|_{\mathbf{2}}) \tag{140}$$

$$\boldsymbol{w}(\mathbf{1}) = \frac{\boldsymbol{y}(\mathbf{1})}{\boldsymbol{z}(\mathbf{1})} \tag{141}$$

$$\mathbf{P}(\mathbf{1}) = \frac{\mathbf{1}}{\mathbf{z}(\mathbf{1})^{\mathbf{2}}} \tag{142}$$

where $g(\cdot)$ is the radial basis activation function. This completes the necessary calculations upon the arrival of the first input data. The rest of the steps are followed at each discrete time instant $k > 1$.

To determine whether a new center should be added at time step k, the algorithm computes the distances between $\boldsymbol{x}(k)$ and the chosen subspaces and finds the subspace l_0 that exhibits the minimum distance rd^{l_0} from the input data example $\boldsymbol{x}(k)$.

$$\boldsymbol{rd}^{\boldsymbol{l}_0}(\boldsymbol{x}(\boldsymbol{k})) = \min_{\mathbf{1}\le \boldsymbol{l}\le \boldsymbol{L}} \boldsymbol{rd}^{\boldsymbol{l}}[\boldsymbol{x}(\boldsymbol{k})] \tag{143}$$

$$\boldsymbol{l}_{\mathbf{0}} = \mathbf{arg}\min_{\mathbf{1}\le \boldsymbol{l}\le \boldsymbol{L}} [\boldsymbol{rd}^{\boldsymbol{l}}(\boldsymbol{x}(\boldsymbol{k}))] \tag{144}$$

The algorithm only adds a new center when the distance rd^{l_0} is greater than one, otherwise it does not add a new hidden node to the network structure and proceeds with updating the AHV vector.

When a new node is added the algorithm augments the value of L by 1 and determines the new fuzzy subspace $A^L = \left[A^L_{1,j_1}, A^L_{2,j_2}, \dots, A^L_{N,j_N}\right]$ that is closer to the new data input $x(k)$, using the minimum distance criterion:

$$\boldsymbol{j_i = \arg\max_{1 \le j \le c_i} \left[\mu_{A_{i,j}}\big(x_i(k)\big)\right], i = 1, \dots, N} \tag{145}$$

The center of the created fuzzy subspace is the assigned to the newly introduced hidden layer node and the center location matrix is augmented to encompass the new hidden node center:

$$\boldsymbol{C^L \leftarrow a^L} \tag{146}$$

The AHV vector is also augmented by adding one more element, where the time step k is assigned, meaning that the kth data point represents the most recent input example allocated to the Lth fuzzy subspace.

$$\boldsymbol{h^L = k} \tag{147}$$

When the algorithm has decided not to add a new node to the hidden layer, the AHV is updated as follows:

$$\boldsymbol{h^{l_0} = k} \tag{148}$$

That is, the input data k is the last one assigned to the l_0 fuzzy subspace, where l_0 is given by equation (**144**)

Additionally, the algorithm at each time instance checks if a hidden node should be deleted by selecting the element of the AHV vector with the minimum value. This corresponds to the fuzzy subspace with longest time horizon in the past for which no input examples have been assigned:

$$\boldsymbol{l_r = \arg\max_{1 \le l \le L} \left[h^l(k)\right],} \tag{149}$$

However, when the inequality $\boldsymbol{h}^{l_r}(k) \le k - N_d$ holds, then no inputs examples have been assigned to the subspace l_r for N_d sequential time instances. In this case the hidden node l_r is considered redundant and it is deleted. This is possible by deleting the corresponding row of the center location matrix $\boldsymbol{C}^{l_r}$ and element of the AHV $\boldsymbol{h}^{l_r}$.

When a new hidden node is added or deleted to the existing network structure, the connection weights $\boldsymbol{w}$ must be recalculated. In order to achieve this, the moving time window of past input-output data is loaded and the responses of the current hidden layer structure $R(k) \in \mathbb{R}^{N_s \times L}$ are calculated:

$$\boldsymbol{R}(\boldsymbol{k}) = \begin{bmatrix} \boldsymbol{g}(\|\boldsymbol{x}(\boldsymbol{k}-\boldsymbol{N_s}) - \boldsymbol{C^1}\|_2 & \cdots & \boldsymbol{g}(\|\boldsymbol{x}(\boldsymbol{k}-\boldsymbol{N_s}) - \boldsymbol{C^L}\|_2 \\ \vdots & \ddots & \vdots \\ \boldsymbol{g}(\|\boldsymbol{x}(\boldsymbol{k}) - \boldsymbol{C^1}\|_2 & \cdots & \boldsymbol{g}(\|\boldsymbol{x}(\boldsymbol{k}) - \boldsymbol{C^L}\|_2 \end{bmatrix} \tag{150}$$

The new synaptic weights are then calculated via the standard least squares method:

$$\boldsymbol{w}(\boldsymbol{k}) = (\boldsymbol{R}^{\mathrm{T}}(\boldsymbol{k}) \cdot \boldsymbol{R}(\boldsymbol{k}))^{-1} \cdot \boldsymbol{R}^{\mathrm{T}}(\boldsymbol{k}) \cdot \boldsymbol{Y}(\boldsymbol{k}) \tag{151}$$

where N_s is the size of the chosen time window and $Y(k) \in \mathbb{R}^{N_s \times 1}$ is the vector containing the true process outputs over the time window N_s

The matrix $\mathrm{P}(k)$ is also calculated so that the RLS adaptation can be continued in the next time step:

$$\mathbf{P}(\boldsymbol{k}) = (\boldsymbol{R}^{\mathrm{T}}(\boldsymbol{k}) \cdot \boldsymbol{R}(\boldsymbol{k}))^{-1} \tag{152}$$

When no structural modification in the hidden layer is made, the number and the locations of the hidden nodes remain unaltered. In this case the connection weights are updated using the RLS algorithm with exponential forgetting, which is described via the following set of equations [163]:

$$\boldsymbol{w}(\boldsymbol{k}) = \boldsymbol{w}(\boldsymbol{k}-\mathbf{1}) + \boldsymbol{q}(\boldsymbol{k}) \cdot (\boldsymbol{y}(\boldsymbol{k}) - \boldsymbol{z}^{\boldsymbol{T}}(\boldsymbol{k}) \cdot \boldsymbol{w}(\boldsymbol{k}-\mathbf{1})) \tag{153}$$

$$\boldsymbol{q}(\boldsymbol{k}) = \mathbf{P}(\boldsymbol{k}-\mathbf{1}) \cdot \boldsymbol{z}(\boldsymbol{k}) \cdot \left(\boldsymbol{\lambda} + \boldsymbol{z}^{\boldsymbol{T}}(\boldsymbol{k}) \cdot \mathbf{P}(\boldsymbol{k}-\mathbf{1}) \cdot \boldsymbol{z}(\boldsymbol{k})\right)^{-1} \tag{154}$$

$$\mathbf{P}(\boldsymbol{k}) = \left(\boldsymbol{I} - \boldsymbol{q}(\boldsymbol{k}) \cdot \boldsymbol{z}^{\boldsymbol{T}}(\boldsymbol{k})\right) \cdot \frac{\mathbf{P}(\boldsymbol{k}-\mathbf{1})}{\boldsymbol{\lambda}} \tag{155}$$

With $\boldsymbol{y}(\boldsymbol{k})$ being the system output at time step k, and λ being the forgetting factor, meaning that at each time instant, the current data point is given unite wight and the input-output example that is n times old, is weighted by λ^n.

Remark: It is essential to point out that the adaptive SFM training algorithm has been shown to offer very high modelling accuracy, due to its ability to adjust the network's structure, combined with relatively low computational times. This property makes it an ideal candidate method for real time modeling.

The validity of these observations is substantiated through the algorithm's applications in modeling nonlinear dynamical systems, particularly in the context of control regulation tasks. In the next subsection a nonlinear MPC scheme is developed with guaranteed stability that utilizes the adaptive SFM model training.

6.3 A new nonlinear MPC control framework with online adjustable RBF neural networks

The goal of this section is to formulate a novel nonlinear MPC (NMPC) strategy, leveraging adaptive RBF neural networks for nonlinear system modeling. In this context, the system's dynamics are meticulously represented by RBF neural networks and fine-tuned online through the adaptive SFM algorithm presented in section 6.2. This ASFM-RBF-NMPC approach is based on Lyapunov stability theory, ensuring the closed-loop control system's asymptotic stability. Two distinct nonlinear systems are employed to demonstrate its efficacy in various regulation tasks, namely a NARX system and a time-varying CSTR reactor.

6.3.1 Problem formulation and preliminaries

Consider the quadratic cost function J for the constrained finite time OCP problem as:

$$\begin{gathered}\underbrace{\min}_{\boldsymbol{u}} \boldsymbol{J_N}(\boldsymbol{x_0}, \boldsymbol{u}) \\ \boldsymbol{s.t.}\ \boldsymbol{\Delta u(k)_{min}} \leq \boldsymbol{\Delta u(k)} \leq \boldsymbol{\Delta u(k)_{max}} \\ \boldsymbol{u_{min}} \leq \boldsymbol{u(k)} \leq \boldsymbol{u_{max}} \\ \boldsymbol{x(0, x_0) = x_0} \\ \boldsymbol{x(k+1) = f(x(k), u(k))}\end{gathered} \tag{156}$$

with J_N given by:

$$\boldsymbol{J_N(k)} = \sum_{i=1}^{H_p} [\boldsymbol{r(k+i)} - \boldsymbol{\hat{y}(k+i)}]^{\mathrm{T}} \cdot \boldsymbol{Q_i} \cdot [\boldsymbol{r(k+1)} - \boldsymbol{\hat{y}(k+1)}] + \sum_{j=1}^{H_u} \boldsymbol{\Delta u(k+j-1)}^{\mathrm{T}} \cdot \boldsymbol{R_j} \cdot \boldsymbol{\Delta u(k+j-1)}^{\mathrm{T}} \tag{157}$$

Where $f(\ \cdot)$ depicts the nonlinear process, $r(k)$ is the reference signal at a time k, $\hat{y}(k+i)$ is the future prediction of the process output, $u(k)$ is the control signal at time k, and $\Delta u(k+j-1)^{\mathrm{T}}$ is the incremental control move between two consecutive control signals, H_p is the prediction horizon, N_u is the control horizon (with $H_u \leq H_p$), and Q, R are weighting matrices that penalize divergence from reference state and input moves, respectively. MPC is an iterative optimization procedure in which the minimization of the performance cost function J takes places at each time

k. The solution of the above optimal control problem is an optimal control sequence from which only the first control signal is implemented into the real system.

The nonlinear dynamical system is presented by the following nonlinear difference equation:

$$\boldsymbol{y(k+1) = f\big(y(k), \dots, y\big(k - n_y + 1\big), u(k), \dots, u(k - n_u + 1)\big)} \tag{158}$$

where $y(k)$ and $u(k)$ denote the output and input of the controlled process at time k, n_y and n_u are the numbers of maximum lags in the outputs and inputs respectively.

6.3.2 Online Based Neural Network MPC

A critical issue when forming the MPC problem is the selection of an appropriate model that can sufficiently predict the output of the real system. Especially in the case of nonlinear systems the selection of a proper predictor equation that can successfully model the nonlinear system dynamics is essential in order to guarantee good performance for the given optimization problem. The online nonlinear MPC problem is tackled by utilizing a nonlinear RBF neural network model as a predictor which can adjust its structure and parameters in real time according to the ASFM algorithm mentioned in subsection 6.2.

The OCP problem formulated by equations (156) and (157) can be rewritten in matrix form by using the adjusting RBF neural network (4) to the following nonlinear NMPC OCP:

$$\boldsymbol{\hat{J}(k) = [r(k) - \hat{y}(k)]^{\mathrm{T}} \cdot M \cdot [r(k) - \hat{y}(k)] + \Delta u(k)^{\mathrm{T}} \cdot R \cdot \Delta u(k)} \tag{159}$$

$$\begin{aligned} \boldsymbol{\Delta u_{min} \leq \Delta u(k) \leq \Delta u_{max}} \quad & \boldsymbol{\forall\, j \in [0, \dots, H_u - 1]} \\ \boldsymbol{\Delta u(k + H_u + j) = 0,} \quad & \boldsymbol{\forall\, j \geq 0} \\ \boldsymbol{r\big(k + H_p + j\big) - \hat{y}\big(k + H_p + j\big) = 0, \forall\, j \geq 0} & \end{aligned} \tag{160}$$

$\boldsymbol{r}(k)$ is the reference vector given at time instance k, $\hat{y}(k)$ is the prediction future output vector given at time instance k, $\Delta u(k)$ is the difference in control actions vector, and $\boldsymbol{M} \in \mathbb{R}^{H_p \times H_p}$, $\boldsymbol{R} \in \mathbb{R}^{H_u \times H_u}$ are diagonal square matrices related with penalty reference divergence and economy in sequential control actions. The above quantities are depicted by the following relations:

$$\boldsymbol{r(k) = \big[r(k+1), r(k+2), \cdots, r\big(k + H_p\big)\big]^{\mathrm{T}}} \tag{161}$$

$$\boldsymbol{\hat{y}(k)} = \left[\boldsymbol{\hat{y}(k+1)}, \boldsymbol{\hat{y}(k+2)}, \cdots, \boldsymbol{\hat{y}\left(k+H_p\right)}\right]^{\mathrm{T}} \tag{162}$$

$$\boldsymbol{\Delta u(k)} = [\boldsymbol{\Delta u(k)}, \boldsymbol{\Delta u(k+1)}, \cdots, \boldsymbol{\Delta u(k+H_u-1)}]^{\mathrm{T}} \tag{163}$$

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{\mu(1)} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{\mu(H_p)} \end{bmatrix}, \ \boldsymbol{R} = \begin{bmatrix} \boldsymbol{\rho(1)} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{\rho(H_u)} \end{bmatrix} \tag{164}$$

with $\mu(j)$ and $\rho(i)$ where $j = 1, \dots, H_p$ and $i = 1, \dots, H_u$, representing weighting factors given by:

$$\boldsymbol{\mu(1)} \le \boldsymbol{\mu(2)} \le \cdots \le \boldsymbol{\mu(H_p)} \tag{165}$$

$$\boldsymbol{\rho(1)} \le \boldsymbol{\rho(2)} \le \cdots \le \boldsymbol{\rho(H_u)} \tag{166}$$

At his stage, it’s crucial to highlight that employing an RBF network for the system model results in an output prediction which is nonlinear in control inputs. As a result, an optimizer should be considered that takes into account the complexity of the constrained on-line NMPC problem, while the convergence for the optimization problem is also a critical aspect to be considered.

To solve the optimization problem given by equations (159)-(160) the sequential quadratic programming (SQP) iterative method is utilized. The basic idea behind the SQP optimizer is that a sequence of optimization subproblems is solved, each of which optimizes a quadratic model of the objective with a linearization of the respective constraints. An important trait of the algorithm is that it can consider constraints on the control input and output variables of the process. This feature will be exploited in next section in order to prove the convergence of the closed loop control system.

6.3.3 Stability Analysis

In this subsection, convergence analysis for the recursive feasibility and stability of the proposed NMPC controller is given for the nominal stability case:

Theorem 3: Suppose that the optimization problem (159)-(160) is feasible at time $k = 0$. Then the proposed problem is recursively feasible for the set-point tracking case.

Proof: Consider the optimal control sequence at a recalculating time instance $t = k$ as:

$$\boldsymbol{u}^*(\boldsymbol{k}) = \left[\boldsymbol{u}^*(\boldsymbol{k}), \boldsymbol{u}^*(\boldsymbol{k}+\mathbf{1}), \cdots, \boldsymbol{u}^*\left(\boldsymbol{k}+\boldsymbol{H_p}-\mathbf{1}\right)\right]^{\mathrm{T}} \tag{167}$$

where the respective optimal state sequence is given by:

$$\boldsymbol{y}^*(\boldsymbol{k}) = \left[\boldsymbol{y}^*(\boldsymbol{k}), \boldsymbol{y}^*(\boldsymbol{k}+\mathbf{1}), \cdots, \boldsymbol{y}^*\left(\boldsymbol{k}+\boldsymbol{H_p}\right)\right]^{\mathrm{T}} \tag{168}$$

However, at time instance $k + H_p$ the predicted state which starts at time k will be inside the terminal set which in our case is formed by the zero terminal constraint region. Due to the condition imposed by the third equation of (160) for all $y(k)$ inside the zero terminal constraint region and for all t the existence of an admissible input sequence input $u_{ZTC}(\cdot)$ can be guaranteed.

Now, consider a new control input sequence at time instance $k + 1$ which be constructed based on the current optimal sequence:

$$\widetilde{\boldsymbol{u}}(\boldsymbol{k}+\mathbf{1}) = \begin{cases} \boldsymbol{u}^*(\boldsymbol{k}+\boldsymbol{i}), & \boldsymbol{i} \in \mathbb{I}_{[\mathbf{1},\boldsymbol{H_p}-\mathbf{1}]} \\ \boldsymbol{u_{ZTC}}(\boldsymbol{k}+\boldsymbol{i}), & \boldsymbol{i} = \boldsymbol{H_p} \end{cases} \tag{169}$$

The input $\widetilde{\boldsymbol{u}}(k+1)$ is admissible and due to the third equation of (160) the predicted state trajectory $\tilde{y}(k+1+H_p)$ at the time instance $k+1+H_p$ will be inside the terminal region. Thus, the optimal control problem has a feasible solution at time instance $t = k + 1$, which means that the proposed MPC algorithm is recursive feasible.

This concludes the proof of Theorem 3.■

In what follows, the asymptotic convergence analysis of proposed MPC algorithm will be given. The asymptotic stability is investigated for the closed loop control system by utilizing the decreasing monotonicity concept of the MPC cost function. The asymptotical convergence of the online AFM-RBF-NMPC is discussed in Theorem 4:

Theorem 4: Consider the nonlinear constrained optimal control problem given by (159)-(160). If the weighting factors are constructed via (165) and (166) then the closed loop control system of the proposed AFM-RBF-NMPC ensures asymptotic convergence for the no model-plant mismatch case.

The MPC cost functional at time instance k has the following form:

$$\hat{\boldsymbol{J}}(\boldsymbol{k}) = \boldsymbol{e}(\boldsymbol{k})^{\mathrm{T}} \cdot \boldsymbol{M} \cdot \boldsymbol{e}(\boldsymbol{k}) + \boldsymbol{\Delta u}(\boldsymbol{k})^{\mathrm{T}} \cdot \boldsymbol{R} \cdot \boldsymbol{\Delta u}(\boldsymbol{k}) \tag{170}$$

$$= \sum_{j=1}^{H_p} \boldsymbol{\mu}(\boldsymbol{j}) \cdot \boldsymbol{e}^{2}(\boldsymbol{k}+\boldsymbol{j}) + \sum_{j=1}^{H_u} \boldsymbol{\rho}(\boldsymbol{j}) \cdot \boldsymbol{\Delta u}^{2}(\boldsymbol{k}+\boldsymbol{j}-\mathbf{1})$$

The optimal control input sequence at time k found by the optimization procedure can be defined as $\boldsymbol{u}^{*}(k) = [u(k), u(k+1), \dots, u(k+H_u-1)]^{T}$. Now the suboptimal control postulated at time $k+1$ can be defined as $\boldsymbol{u}_{S}(k+1) = [u(k+1), \dots, u(k+H_u-1), u(k+H_u-1)]^{T}$. The control sequence $\boldsymbol{u}_{S}(k+1)$ is formed based on the control derived at time k. Therefore, for the suboptimal control $\boldsymbol{u}_{S}(k+1)$, the cost function can be defined as:

$$\hat{\boldsymbol{J}}_{s}(\boldsymbol{k}+\mathbf{1}) = \sum_{j=2}^{H_p+1} \boldsymbol{\mu}(\boldsymbol{j}) \cdot \boldsymbol{e}^{2}(\boldsymbol{k}+\boldsymbol{j}) + \sum_{j=2}^{H_u} \boldsymbol{\rho}(\boldsymbol{j}) \cdot \boldsymbol{\Delta u}^{2}(\boldsymbol{k}+\boldsymbol{j}-\mathbf{1}) \tag{171}$$

By taking the difference between the cost functions $\hat{J}_{S}(k+1)$ and $\hat{J}(k)$ we have:

$$\begin{aligned} &\hat{\boldsymbol{J}}_{s}(\boldsymbol{k}+\mathbf{1}) - \hat{\boldsymbol{J}}(\boldsymbol{k}) \\ &\quad = \boldsymbol{\mu}(\boldsymbol{H_p}) \cdot \boldsymbol{e}^{2}(\boldsymbol{k}+\boldsymbol{H_p}+\mathbf{1}) - \boldsymbol{\mu}(\mathbf{1}) \cdot \boldsymbol{e}^{2}(\boldsymbol{k}+\mathbf{1}) - \boldsymbol{\rho}(\mathbf{1}) \cdot \boldsymbol{\Delta u}^{2}(\boldsymbol{k}) \\ &\quad + \sum_{j=2}^{H_p-1} \left(\boldsymbol{\mu}(\boldsymbol{j}-\mathbf{1}) - \boldsymbol{\mu}(\boldsymbol{j})\right) \cdot \boldsymbol{e}^{2}(\boldsymbol{k}+\boldsymbol{j}) \\ &\quad + \sum_{j=2}^{H_u-1} \left(\boldsymbol{\rho}(\boldsymbol{j}-\mathbf{1}) - \boldsymbol{\rho}(\boldsymbol{j})\right) \cdot \boldsymbol{\Delta u}^{2}(\boldsymbol{k}+\boldsymbol{j}-\mathbf{1}) \end{aligned} \tag{172}$$

Taking into account the terminal constraint equality equation (160) and the relations (165) and (166) we can get:

$$\begin{aligned} &\hat{\boldsymbol{J}}_{s}(\boldsymbol{k}+\mathbf{1}) - \hat{\boldsymbol{J}}(\boldsymbol{k}) \\ &\quad = -\boldsymbol{\mu}(\mathbf{1}) \cdot \boldsymbol{e}^{2}(\boldsymbol{k}+\mathbf{1}) - \boldsymbol{\rho}(\mathbf{1}) \cdot \boldsymbol{\Delta u}^{2}(\boldsymbol{k}) \\ &\quad + \sum_{j=2}^{H_u-1} \left(\boldsymbol{\rho}(\boldsymbol{j}-\mathbf{1}) - \boldsymbol{\rho}(\boldsymbol{j})\right) \cdot \boldsymbol{\Delta u}^{2}(\boldsymbol{k}+\boldsymbol{j}-\mathbf{1}) \\ &\quad + \sum_{j=2}^{H_p-1} \left(\boldsymbol{\mu}(\boldsymbol{j}-\mathbf{1}) - \boldsymbol{\mu}(\boldsymbol{j})\right) \cdot \boldsymbol{e}^{2}(\boldsymbol{k}+\boldsymbol{j}) \leq \mathbf{0} \end{aligned} \tag{173}$$

Furthermore, by considering the optimal control sequence at time instance $\boldsymbol{k+1}$ denoted by $\boldsymbol{u}^*(\boldsymbol{k+1})$ for the optimization problem (159)-(160), then one can state that $\boldsymbol{J}^*(\boldsymbol{k+1}) \leq \hat{\boldsymbol{J}}_{\boldsymbol{s}}(\boldsymbol{k+1})$. Then one can conclude:

$$\hat{\boldsymbol{J}}^*(\boldsymbol{k+1}) - \hat{\boldsymbol{J}}(\boldsymbol{k}) \leq \hat{\boldsymbol{J}}_{\boldsymbol{s}}(\boldsymbol{k+1}) - \hat{\boldsymbol{J}}(\boldsymbol{k}) \leq \boldsymbol{0} \tag{174}$$

Hence the cost function is monotonically decreasing with respect to time and based on the modified Barbalat's lemma [141].

$$\lim_{\boldsymbol{t}\to\infty} \|\boldsymbol{e}(\boldsymbol{t})\| = \boldsymbol{0} \tag{175}$$

the ASFM-RBF-NMPC closed loop control system is asymptotically stable.

This completes the proof of Theorem 4. ■

6.3.4 Simulation results and discussion

To validate the proposed NMPC control technique's effectiveness, two distinct systems were utilized. The first system encompassed a nonlinear benchmark problem described by a NARX single input-single output (SISO) discrete time system, serving as a basis for testing the tracking ability of the NMPC controller. The second system involved a time-varying nonlinear continuous steered tank reactor. To validate the proposed control method, an RBF-NMPC controller was also employed, where RBF networks were pre-trained offline using the fuzzy means algorithm. Throughout all subsequent simulation, the SQP solver is employed for solving the NMPC problem.

6.3.4.1 Application 1: Control of a NARX discrete time SISO system

The system under the certain application is described by the following discrete input-output model:

$$\boldsymbol{y}(\boldsymbol{k}) = \boldsymbol{0.72} \cdot \boldsymbol{y}(\boldsymbol{k-1}) + \boldsymbol{0.025} \cdot \boldsymbol{y}(\boldsymbol{k-2}) \cdot \boldsymbol{u}(\boldsymbol{k-1}) + \boldsymbol{0.01} \cdot \boldsymbol{u}^{\boldsymbol{2}}(\boldsymbol{k-2}) + \boldsymbol{0.2} \cdot \boldsymbol{u}(\boldsymbol{k-3}) \tag{176}$$

The objective is to make the system output $y(k)$ track a reference trajectory utilizing the proposed NMPC controller and the time varying reference input $r(k)$ given by the following equation:

$$\boldsymbol{r}(\boldsymbol{k}) = \sin\left(\boldsymbol{k} \cdot \frac{\boldsymbol{\pi}}{\boldsymbol{100}}\right), \quad \boldsymbol{0} < \boldsymbol{k} \leq \boldsymbol{400} \tag{177}$$

The ASFM method was utilized online in order to identify a suitable RBF configuration for modeling the aforementioned system. The input to the RBF model consisted of three previous values of the input u and two previous values of the output y:

$$\boldsymbol{x}(\boldsymbol{k}) = [\boldsymbol{y}(\boldsymbol{k}-\mathbf{1})\ \boldsymbol{y}(\boldsymbol{k}-\mathbf{2})\ \boldsymbol{u}(\boldsymbol{k}-\mathbf{1})\ \boldsymbol{u}(\boldsymbol{k}-\mathbf{2})\ \boldsymbol{u}(\boldsymbol{k}-\mathbf{3})] \tag{178}$$

The operational parameters used by the NMPC controller and the ASFM training algorithm are given in Table 16 and Table 17 respectively. Furthermore, Table 18 displays the Mean Absolute Error (MAE) results for both control methods implemented in terms of control tracking error. Figure 29 illustrates the RBF network prediction responses for both the adaptive SFM and the

Table 16: Operational parameters for the ASFM RBFNN in application 1

Parameter	Description	Value
N_d	Number of time steps that a center is not assigned to an input example, before it is removed from the hidden layer	1000
N_s	Size of the moving time window used for storing past input-output examples	200
λ	Forgetting factor for the RLS method	1
# fuzzy partitions	Number of fuzzy subspaces in the entire input space	14

Table 17: Operational parameters for the NMPC controller in application 1

Parameter	Description	Value
N_P	Prediction horizon	10
N_C	Control horizon	8
$\mu(j), j = 1, \dots N_p$	State variables penalty weighting factors	1.65
$\rho(j), j = 1, \dots N_c$	Control input weighting factors	0.7
Δu_min	Minimum value of difference in control actions	-0.1
Δu_max	Maximum value of difference in control actions	+0.1

Table 18: MAE metrics for NARX system control

Control Method	**MAE**
NMPC-ASFM-RBFNN	0.0042
NMPC-RBFNN	0.0464

offline SFM algorithm. Additionally, Figure 30 displays the closed-loop response of the NARX system, while Figure 31 provides the input profile for the proposed ASFM-NMPC controller. Figure 32 and Figure 33 track the evolution of the Root Mean Square Error (RMSE) of the output. of the output variable y and the hidden layer structure over time, respectively, for both the ASFM and offline SFM methods.

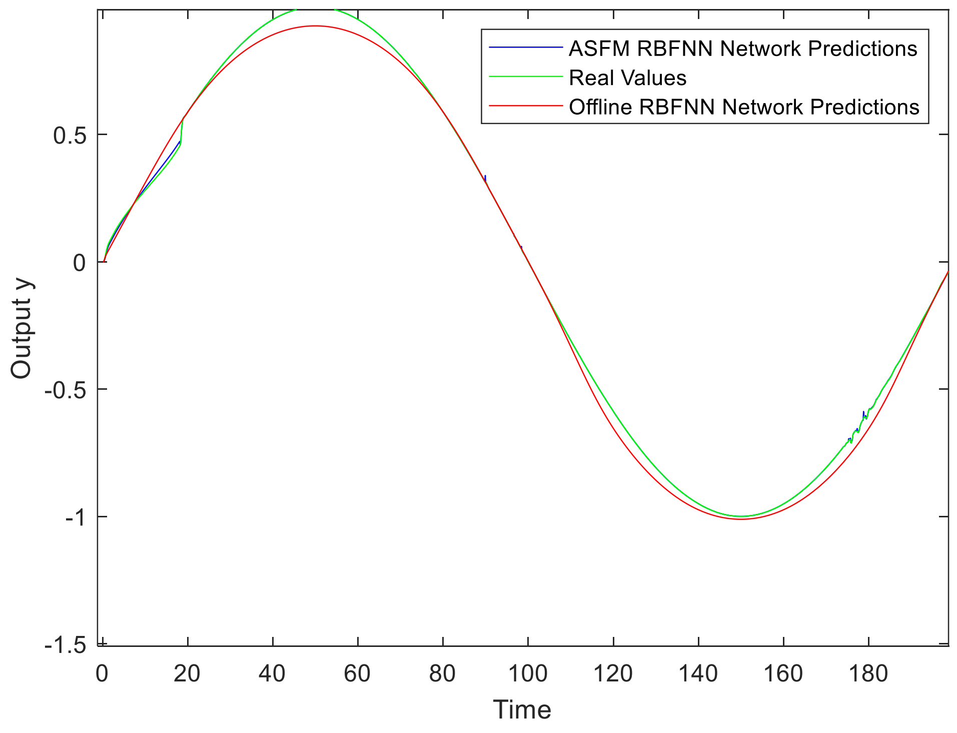

Figure 29: RBF network predictions using different training methodologies for the NARX system

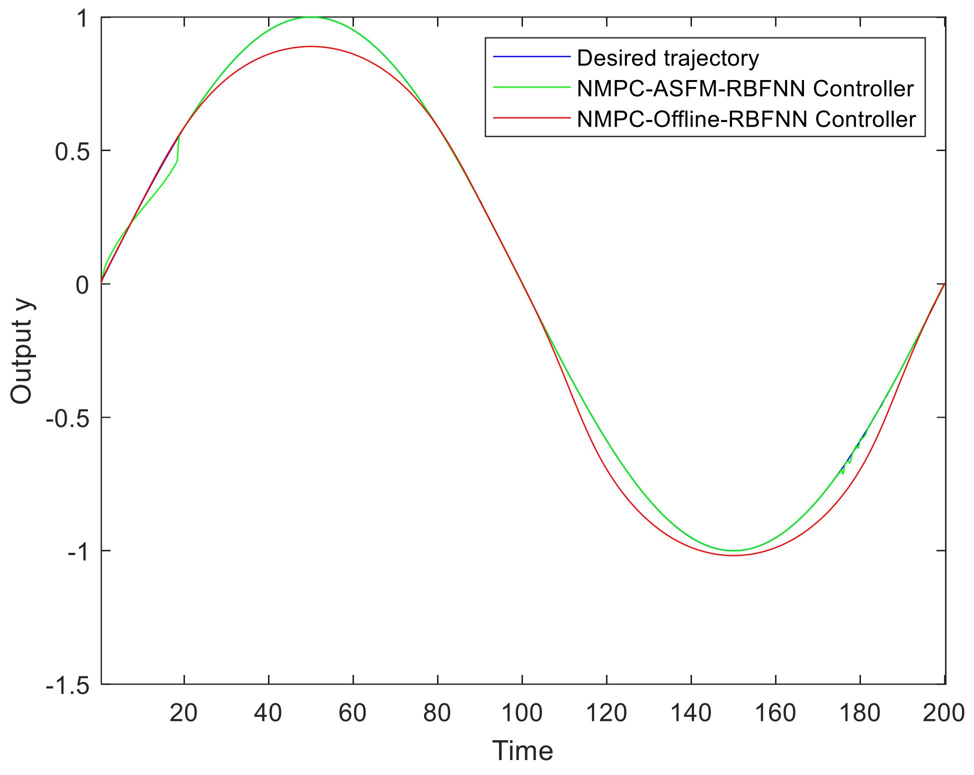

Figure 30: Closed-loop response using the closed loop NMPC controller for NARX system

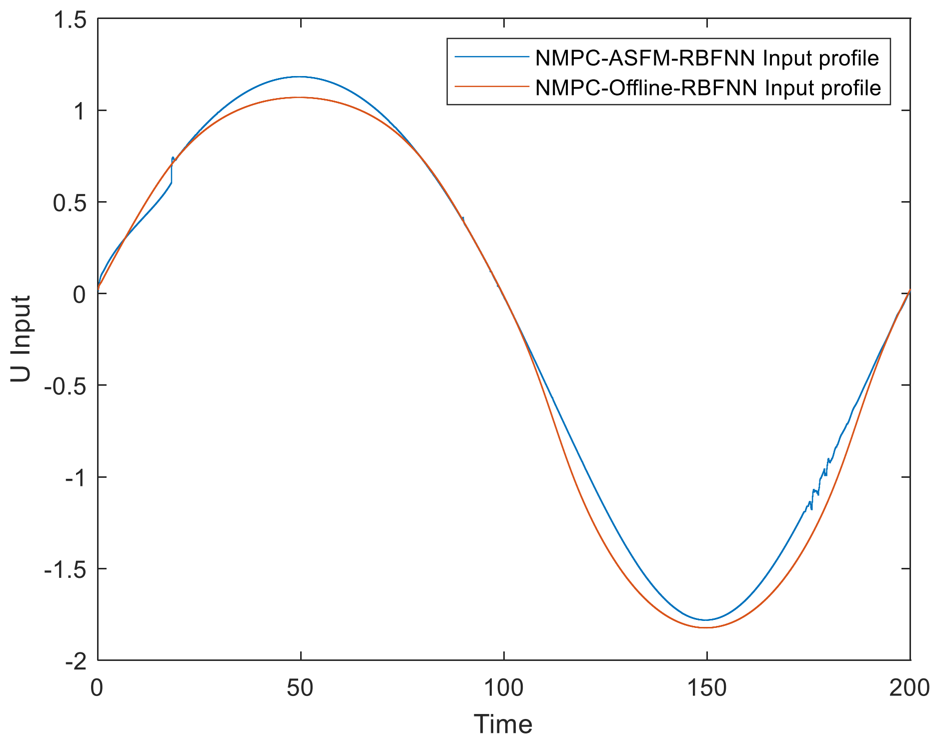

Figure 31: ASFM-NMPC controller-input profile for NARX system

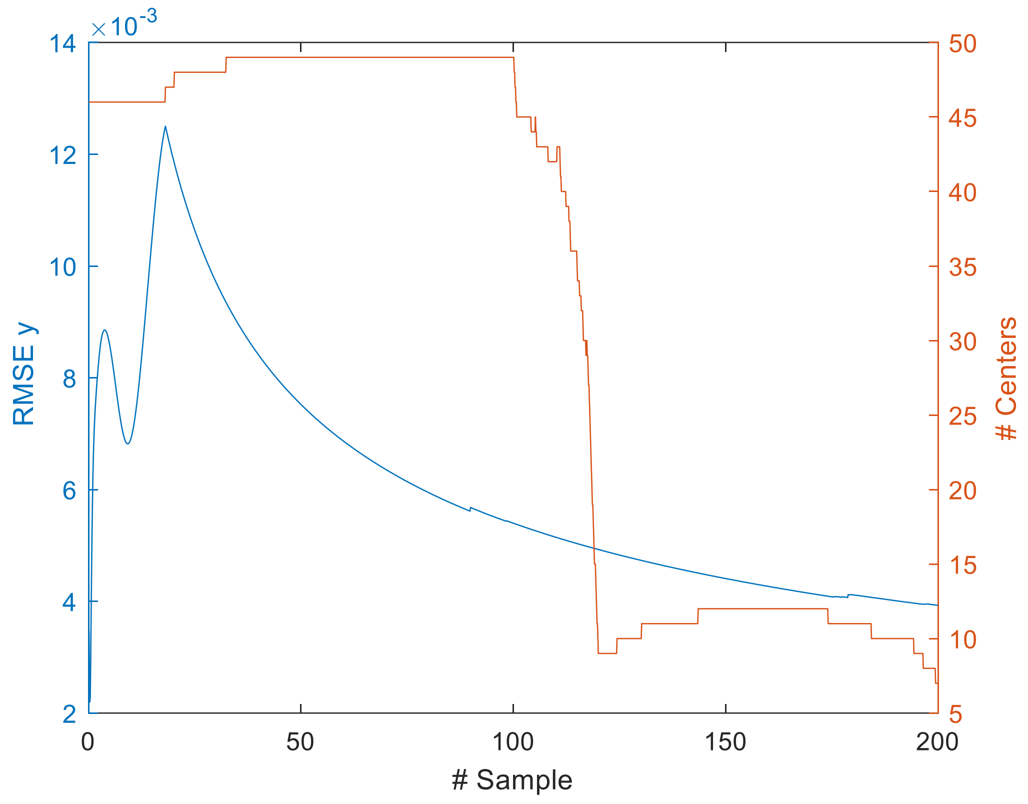

Figure 32: ASFM RBFNN RMSE of output y and hidden layers structure evolution over sample iterations for NARX system

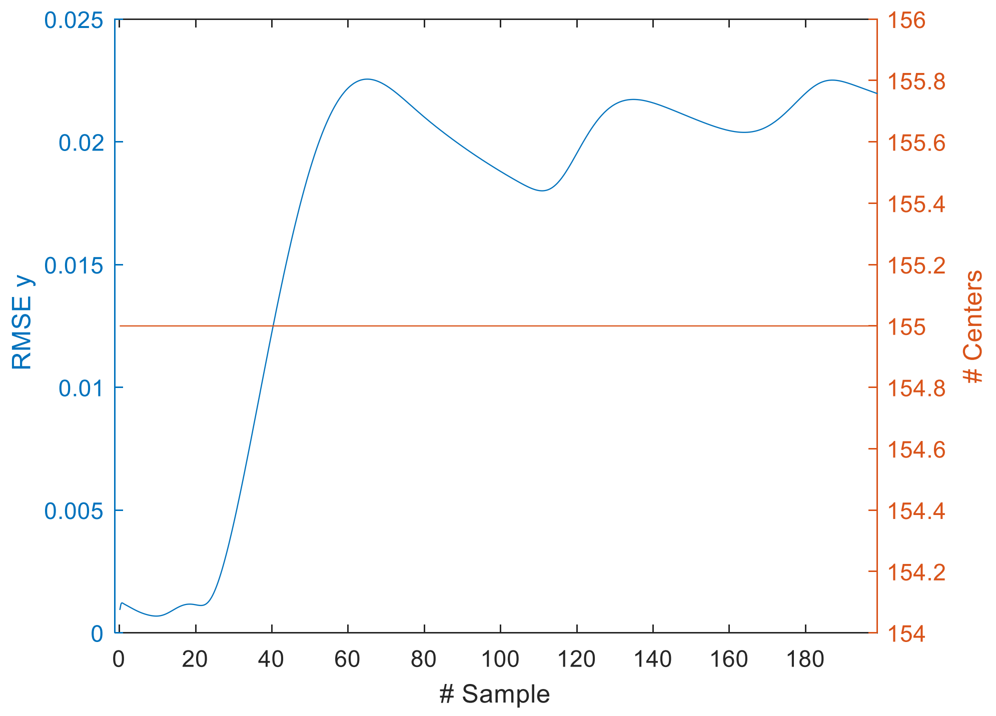

Figure 33: RBFNN RMSE of output y and hidden layers structure evolution over sample iterations in the case of NARX system

The figures clearly illustrate that the system's output y accurately follows the desired trajectory throughout the simulation. The effectiveness of our proposed scheme is particularly evident in Table 18, where the NMPC-ASFM controller outperforms its competitor, achieving a remarkable improvement of 90.9%. This success can be attributed to the enhanced performance of the adaptive RBF-NN incorporated within our scheme, as evidenced in Figure 29. In contrast, the offline RBFNN predictions fail to capture the system's behavior when relying solely on offline data from the NARX system.

Figure 30 demonstrates that the proposed ASFM-NMPC approach adeptly follows the desired trajectory. This in contrast to the simple RBFNN-NMPC controller, which struggles to efficiently track the provided reference. This difficulty arises from its incapacity to incorporate new data from adapting the network's parameters. Additionally, it's important to highlight that the proposed scheme consistently maintains a control input profile where the differences in control action inputs remain within the prescribed limits, as specified in Table 17.

Figure 32 provides insight into the development of the RMSE for the output variable y and the hidden layer structure as each new data sample is introduced.

The initial rise in RMSE for output y can be attributed to the offline training phase of the $f_{y\,RBF}$ model. During this phase, the generated input-output data are formed by employing random input values within the range of [-1, 1], in accordance with the discrete output equation (176). However, as more sample data are incorporated into the ASFM and new centers are added, RMSE consistently decreases. This stands in contrast to Figure 32 where the RBF model is exclusively trained offline. In Figure 32 the RMSE for the model's output continually rises and then oscillates around a certain value, unable to decrease even when new data samples area available. This disparity can be attributed to the offline-SFM's limitation in adapting to incoming data from the system and modifying the network's structure, a capability presents in its ASFM counterpart.

6.3.4.2 Application 2: Control of a multiple steady state CSTR time-varying reactor in changing operating regions

The objective of this application is to control a CSTR reactor utilizing an adaptive RBF network model, as the system's behavior undergoes dynamic shifts over time. The CSTR described by a set of nonlinear ODEs which can be found in [164] and is given by the following equations:

$$\boldsymbol{\frac{dc_A}{dt} = \frac{F}{V} \cdot \left(c_{A,in} - c_A\right) - 2 \cdot k_0 \cdot \exp\left(-\frac{E}{R \cdot T}\right) \cdot c_A^2} \tag{179}$$

$$\boldsymbol{\frac{dT}{dt} = \frac{F}{V} \cdot (T_{in} - T) + 2 \cdot \frac{(-\Delta H)_R}{\rho \cdot C_p} \cdot k_0 \cdot \exp\left(-\frac{E}{R \cdot T}\right) \cdot c_A^2 - \frac{U\Lambda}{V \cdot \rho \cdot c_p} \cdot \left(T - T_j\right)} \tag{180}$$

where k_0 is the reaction frequency factor, E depicts the reaction activation energy, R is the gas constant, T is the temperature and c_A is the conversion of reactant A to product B. The kinetic parameter values of equations are given in Table 19.

Under specific configurations of the CSTR operational parameters, the process demonstrates the presence of multiple stable steady states, both upper and lower, alongside an unstable intermediate state. While it is relatively straightforward to control the CSTR when operating around each individual stable steady state point, controlling the system through its entire operational range, encompassing the unstable steady state, presents a challenging endeavor.

Consequently, the objective of the subsequent application involves the implementation of an NMPC configuration to govern the output concentration c_A, employing the temperature of the coolant T_j as the manipulated variable. The specific focus of this application was to evaluate the effectiveness of the proposed method in dynamically modeling the reactor, with particular emphasis on scenarios encompassing variations in the operating range. The CSTR was simulated by solving a system of ODEs.

Due to the presence of multiple steady states, it was not feasible to approximate the system dynamics across the entire operating range with model of this type:

$$\boldsymbol{c_A(k) = f_{RBF}(\, T_j(k-1), T_j(k-2), \dots, T_j(k-i))} \tag{181}$$

i.e., a model using as inputs only past values of T_j. This comes up from the fact that for the same sequence of inputs, multiple potential values of the concentration $c_A(k)$ exist, contigent on the proximity of the CSTR to a particular steady state at that time instant. To address this issue, an

ARX (AutoRegressive with eXogenous inputs) model was chosen. This model correlates the current concentration $c_A(k)$ not only with the previous coolant temperature $T_j(k-1)$ but also with preceding values of two state variables: concentration $c_A(k-1)$ and the temperature inside the reactor $T(k-1)$:

$$\boldsymbol{c_A(k) = f_{C_A RBF}(\,T_j(k-1), c_A(k-1), T(k-1))} \tag{182}$$

The inclusion of such a model, using the previous values of two state variables as inputs, adds complexity to the task of calculating predictions over the prediction horizon. At each time step, the model (182) is utilized for predicting the next N_p time steps in the optimization problem (159)-(160). However, to forecast beyond the first future time instance, input values for both state variables during the preceding time steps are needed. Hence, an additional model is vital for predicting the dynamic evolution of the second sate variable:

$$\boldsymbol{T(k) = f_{T\,RBF}(\,T_j(k-1), c_A(k-1), T(k-1))} \tag{183}$$

To simulate scenarios involving changes in the operating region, an initial offline training phase was conducted. During this phase, a random dataset, equivalent in size to the moving time window, was generated by changing the coolant temperature T_j within the range $[164\ 350]$ every 0.5 seconds, in order to provide the initial training data for the RBF network models. In this range, the generated data exclusively pertained to the lower equilibrium region, with all concentrations falling below 0.2. The sampling time chosen for the control process was set as well to 0.5 seconds. The operational parameters used for training both RBF networks used in this study are shown in Table 20.

The models trained with the ASFM methodology were incorporated into the NMPC configuration described in section 6.3.2. The CSTR is initialized at a concentration c_A equal to 0.1 which corresponds to the lower steady state point and then a step to the value of 0.4 occurs, corresponding to the unstable state-state point.

In Table 21, are shown the operational parameters chosen for NMPC controllers used in the CSTR system control. Table 22 contains the results for both controllers, including MAE for the closed-loop response of the CSTR system. Figures 34-35 provide visual representation of the responses of the network for the state variables c_A and T, comparing the adaptive SFM and the offline SFM algorithm. Figure 36 presents the performance of the NMPC controllers and Figure 37, provides a view of the input profiles for both controller systems. Finally, Figures 38-39 illustrate the evolution of the RMSE for the concentration c_A model and the hidden layer structure over time for the ASFM and offline SFM methods, respectively.

Table 19: CSTR parameter values in application 2

CSTR Process parameters	Description	Values
F	Flow rate	20 1/s
V	Volume	100 l
$U\Lambda$	Rate of change of thermal energy	20000 $\mathrm{J}/s \cdot K$
ρ	Density	1000 g/l
c_p	Heat capacity of reactive mixture	4.2 $\mathrm{J}/g \cdot K$
$(-\Delta H)_R$	Negative change in enthalpy	596619 J/mol
k_0	Reaction frequency factor	6.85E+11 l/s·mol
E	Activation energy	76543.704 J/mol
T_{in}	Inlet Temperature	275 K
$c_{A,in}$	Inlet Concentration of species A	1 mol/l

Table 20: Operational parameters for the ASFM RBFNN in application 2

Parameter	Description	Value
N_d	Number of time steps that a center is not assigned to an input example, before it is removed from the hidden layer	1000
N_s	Size of the moving time window used for storing past input-output examples	50
λ	Forgetting factor for the RLS method	0.91
# fuzzy partitions	Number of fuzzy subspaces in the entire input space	20

Table 21: Operational parameters for the NMPC controller in applications 2

Parameter	Description	Value
N_P	Prediction horizon	14
N_C	Control horizon	3
$\mu(j), j = 1, \dots N_p$	State variables penalty weighting factors	5.5
$\rho(j), j = 1, \dots N_c$	Control input weighting factors	0.1
Δu_min	Minimum value of difference in control actions	-10
Δu_max	Maximum value of difference in control actions	+10

Table 22: MAE metrics for CSTR system control

Control Method	**MAE**
NMPC-ASFM-RBFNN	0.0286
NMPC-RBFNN	0.2759

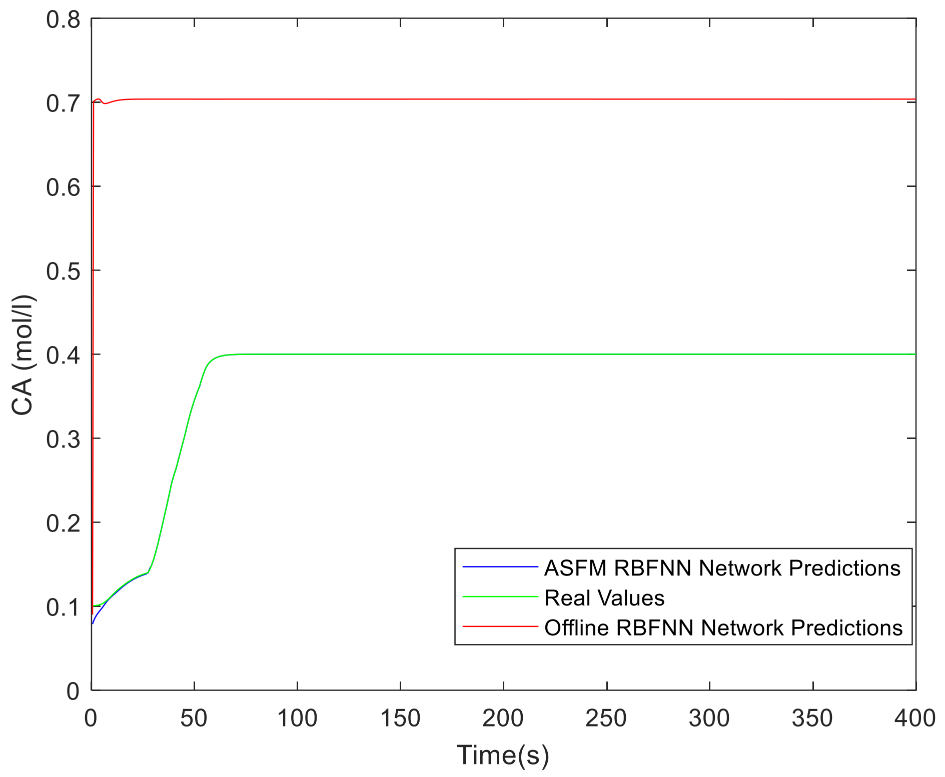

Figure 34: RBF network predictions for concentration c_A using different training methodologies for the CSTR system

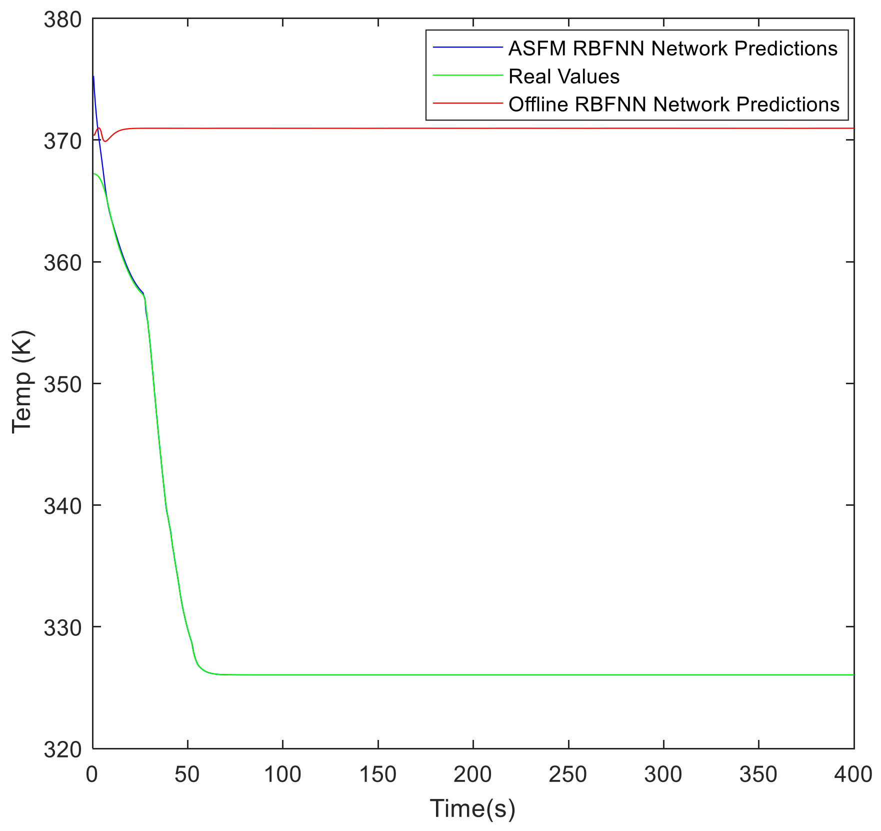

Figure 35: RBF network predictions for temperature T using different training methodologies for the CSTR system

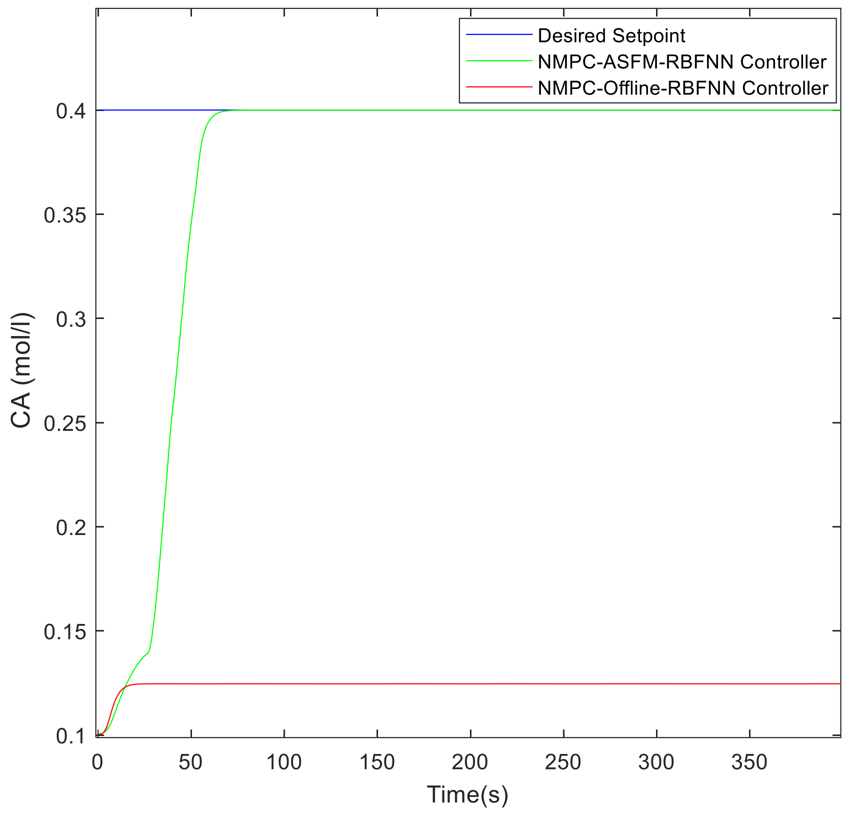

Figure 36: Closed-loop response using NMPC controller for CSTR system

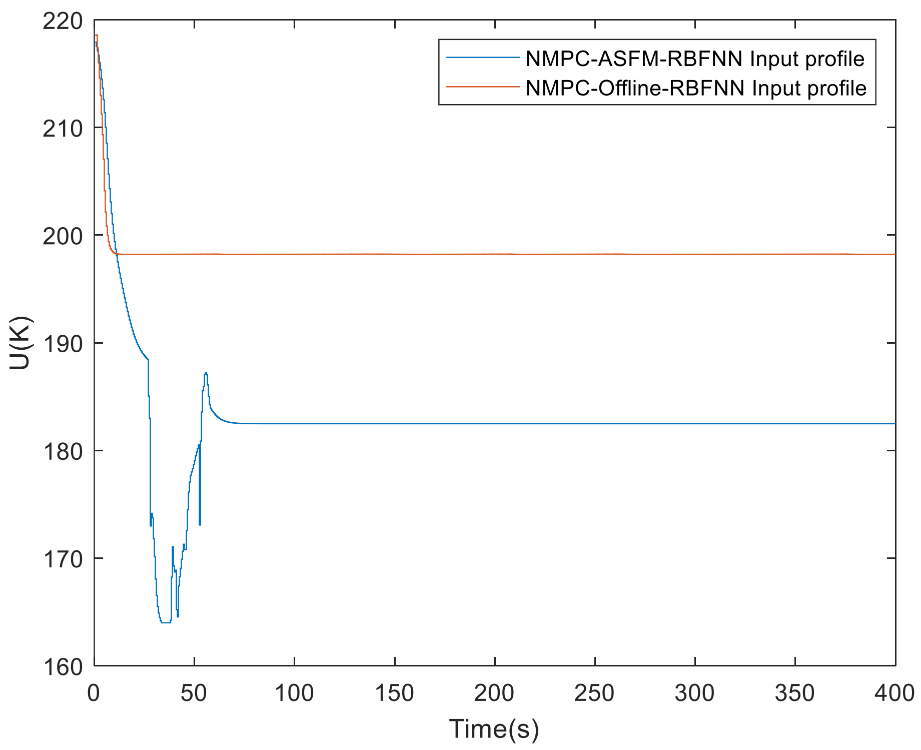

Figure 37: ASFM-NMPC controller-input profile for CSTR system

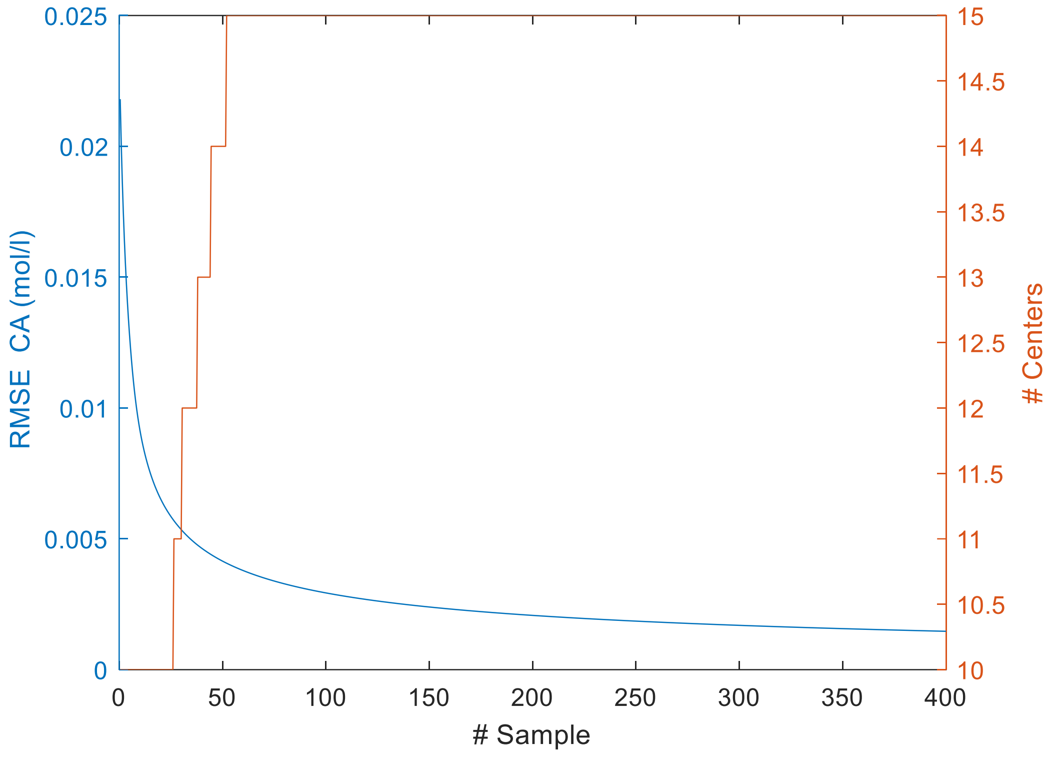

Figure 38: RMSE of output c_A and hidden layers structure evolution over sample iterations for the ASMF training algorithm

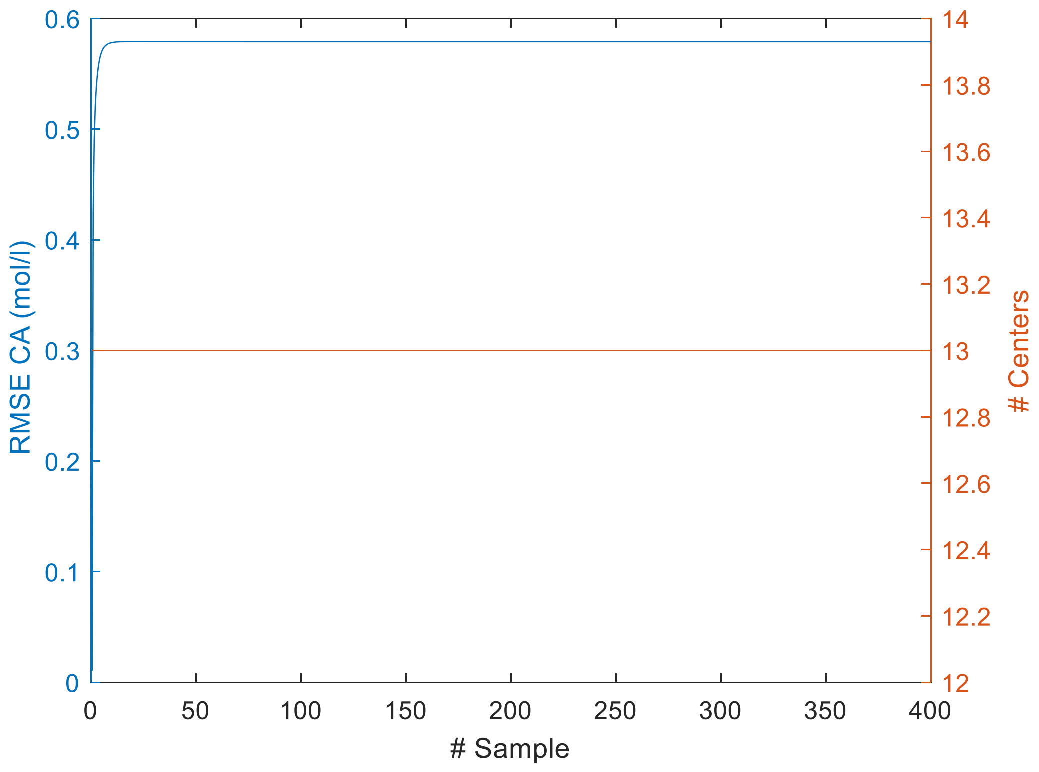

Figure 39: RMSE of output c_A and hidden layers structure evolution over sample iterations for the simple SFM training algorithm

Throughout Figure 34 and Figure 35 it can be clearly seen that the ASFM online training algorithm excels in producing network predictions that can accurately track real value of the system, even during changes in the system's operating region for both state variables, C_A and T. This stands in stark contrast to the simple offline SFM RBF prediction responses, which fails to track the real values. This disparity is due to the Offline SFM-RBF model's inability to adapt to the system's changing dynamics caused by shifts in the operating region.

Table 22 presents the results for both controllers, highlighting that the proposed NMPC-ASFM controller outperforms its rival with an improvement of 89.6%. This result is evident in Figure 36, which depicts the closed-loop response of both NMPC controllers for tracking the desired unstable setpoint $C_{A\,ref}$ equal to 0.4. The offline controller fails to track the desired setpoint, whereas the proposed controller accurately follows the desired trajectory by adding new centers that can describe the system's new operating region. It's worth noting that the minor "bump" in the initial seconds (0-40s) of the simulation occurs because no new centers were added to the pretrained offline models of $\boldsymbol{f_{C_A RBF}}$ and $\boldsymbol{f_{T\,RBF}}$. However, as time progresses and new real data inputs become available, new centers are added, improving the proposed controller's tracking ability.

Furthermore, Figure 37 demonstrates the NMPC-ASFM controller's ability to produce input control actions that consider the system's changing operating region, effectively steering the CSTR c_A, state toward the desired setpoint.

Figure 38 shows the RMSE development for c_A, and the hidden layer structure as each new data sample is introduced. Due to the transition to a different operating region, the number of centers increases, as new centers are needed to describe the new inputs. The slight initial increase in RMSE for c_A, as depicted in Figure 34 is due to the initial offline training phase of the $\boldsymbol{f_{C_A RBF}}$ and $\boldsymbol{f_{T\,RBF}}$ models, which do not include data near the new setpoint. Nevertheless, over time, as additional actual system output values are included as input samples in the adaptive RBF models, the RMSE steadily decreases. This is attributed to the ASFM's capability to adapt to incoming data from the system and adjust the network's structure accordingly. This contrasts with Figure 39, where the number of centers remains constant, and the RMSE of the trained offline-RBFNN increases until a certain point without decreasing. This occurs because the model's predictions do not align with the real values of the system, as shown in Figure 34 and Figure 35.

7. Conclusions

The goal of this dissertation was to harness computational intelligence techniques for the development and tuning of automatic control methods, with a primary emphasis on their application in unmanned aerial vehicles, to enhance and optimize control strategies for nonlinear systems.

The computational techniques employed in this thesis were instrumental in enhancing the effectiveness of nonlinear control methods. Specifically, two prominent nonlinear control methodologies were used in this thesis: Backstepping control and model predictive control. Backstepping control demonstrated its value as an approach well-suited for managing strict feedback systems, benefiting from its inherent stability properties. In contrast MPC stand out as a versatile and widely-adopted control technique, proficient in handling MIMO systems, handling constraints effectively, ensuring ease of implementation, and showcasing extensive applicability across diverse processes and systems.

The quadrotor is chosen as the central vehicle for control due to its complex and inherently nonlinear dynamics, making an ideal platform for developing and enhancing advanced control methods. This choice establishes the quadrotor as the primary nonlinear system for implementing the previously mentioned control techniques. Within this context, the complexities of nonlinear quadrotor dynamics are explored, showcasing the challenges associated with controlling it. The quadrotor's dynamics are modeled using the Newton-Euler model, encapsulating both translational and rotational dynamics within a state-space representation. In response to the challenge presented by the quadrotor trajectory tracking problem, two specialized automatic control methods were developed leveraging both the backstepping and MPC frameworks, incorporating computational techniques.

The first strategy developed to tackle the challenge of quadrotor trajectory tracking involved the creation of a new nonlinear control scheme. This scheme was designed by incorporating both backstepping control and RBF networks to effectively accommodate uncertainties within the model. The developed technique relies on a stabilizing backstepping controller in the Lyapunov sense for the closed loop system, and an RBF network which provides a data-driven approximation of unmodelled uncertainties of any type. The RBF network is trained using the SFM algorithm, enhancing modeling accuracy and enabling superior tracking performance, with the added advantages of a straightforward structure and rapid training speed. Crucially, the RBF-backstepping framework can be readily applied to real-word quadcopters, as RBF networks can be easily trained using actual data collected from the quadrotor's state and accelerations.

In contrast to the backstepping approach, MPC offers several advantages, including its ability to effectively handling constraints and improved adaptability to various nonlinear systems. The conjunction of MPC with PID controllers can lead to improved system performance and robustness. However, this integration introduces multiple tuning parameters, creating a need for an efficient optimization technique. This challenge is effectively addressed through the development of a CPSO framework, which plays a pivotal role in finely adjusting control parameters and ultimately enhancing the system's performance. More specifically, a CPSO framework was designed for optimizing the tuning of an MPC-PID-based quadrotor trajectory tracking scheme. The control framework developed consists of two subsystems: an MPC controller for path following and a PID scheme for attitude stabilization. The subsystem's tuning parameters are optimized respectively via an CPSO. Two different swarms are used for the MPC and PID tuning parameters, respectively; though each swarm controls a different set of parameters, they ultimately work together towards bestowing improved path tracking abilities to the integrated control framework. The proposed scheme excels in producing fine-tuning controllers as demonstrated throughout two different trajectory tracking scenarios. Additionally, its robustness is evident in successfully evaluating the quadrotor's tracking performance on a third, previously unseen trajectory, demonstrating its adaptability beyond the tuning trajectory.

In addition to the CPSO optimization framework for the quadrotor trajectory tracking, two CPSO schemes were specifically devised to tackle complex issues related to reactive power management in smart distribution grids and identifying model parameters in WWTPs. Specifically, the growing prevalence of PV installations within smart grids and their influence on reactive power control were addressed by developing a specialized CPSO optimization and control framework tailored to meet the unique requirements of DNs. Through comprehensive testing across various scenarios, this framework showcased its remarkable adaptability and its ability to yield favorable outcomes. Furthermore, in the context of tackling the parameter identification issue within WWTPs, a new CPSO-based identification framework has been introduced. This framework excels at accurately deducing critical parameters within a comprehensive first principles model, by means of solving a nonlinear optimization problem integrated in a system identification approach. The efficacy of this approach is confirmed through statistical testing, demonstrating its superiority when compared to other methodologies.

Up to this point, the MPC optimization problem that was developed relied on utilizing linearized predictive models. However, limitations arose, primarily related to its inability to capture the complex and nonlinear behavior of the system. These limitations prompted an exploration of alternative modeling approaches, leading to the consideration of RBF network models, which offer a more flexible and accurate representation of the system dynamics. More specifically, a new control method was designed that uses RBF network models within a regulatory MPC control framework. This nonlinear controller effectively employed the MPC methodology, ensuring guaranteed stability by deploying adaptive RBF networks as process models. The closed-loop system is asymptotically stable under the assumption of no model-plant mismatch. This hypothesis is facilitated by the online nature of the NMPC algorithm, wherein network predictions adapt to the current system input-output. Real-time updates of the model's network and structure were achieved through the use of the adaptive SFM algorithm, resulting in an efficient regulatory scheme.

The proposed online controller was assessed in two distinct scenarios: a SISO NARX system and a dynamic CSTR reactor, with a direct comparison to an offline NMPC controller. In both cases, the proposed controller exhibited superior performance, producing better tracking results comparted to its offline counterpart. In the first case, it demonstrated exceptional trajectory tracking with minimal modeling error, outperforming the offline controller. In the CSTR setpoint tracking scenario, the adaptive online NMPC controller demonstrated the ability to track the unstable reference point, in stark contrast to the compared controller, which failed entirely to achieve the desired tracking, highlighting the exceptional effectiveness and capability of the proposed method.

In term of future directions, the research will focus on enhancing the adaptive online NMPC controller by replacing the FM-LS-RLS adaptation strategy with a novel online adaptation scheme based solely on FM-RLS [165]. The FM-RLS algorithm will enable continuous, gradual training without the necessity of performing standard least squares whenever the network structure is altered. This innovative method is expected to provide several benefits, such as lowering the computational load, thus accelerating the overall online network training process, and reducing memory needs for storing the online process data. Furthermore, the resulting scheme will be evaluated on more complex and demanding control tasks, e.g., satellite control. Additionally, within the RBF-backstepping quadrotor trajectory control scheme, there are plans to address additional challenges, such as incorporating time-varying external uncertainties and implementing online adapting RBF networks within the control framework.

References

[1] P. Singhala, D. N. Shah, and B. Patel, “Temperature Control using Fuzzy Logic,” *Int. J. Instrum. Control Syst.*, vol. 4, no. 1, pp. 1–10, 2014, doi: 10.5121/ijics.2014.4101.

[2] H. Ying, L. Sheppard, and D. Tucker, “Expert-system-based fuzzy control of arterial pressure by drug infusion.,” *Med. Prog. Technol.*, vol. 13, no. 4, pp. 203–215, 1988.

[3] H. J. Vishnukumar, B. Butting, C. Müller, and E. Sax, “Machine learning and deep neural network — Artificial intelligence core for lab and real-world test and validation for ADAS and autonomous vehicles: AI for efficient and quality test and validation,” in *2017 Intelligent Systems Conference (IntelliSys)*, 2017, pp. 714–721. doi: 10.1109/IntelliSys.2017.8324372.

[4] Y. Tian, K. Pei, S. Jana, and B. Ray, “DeepTest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars,” in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 303–314. doi: 10.1145/3180155.3180220.

[5] M. J. D. Powell, “Radial Basis Functions for Multivariable Interpolation: A Review,” in *Algorithms for Approximation*, USA: Clarendon Press, 1987, pp. 143–167.

[6] D. Karamichailidou, A. Alexandridis, G. Anagnostopoulos, G. Syriopoulos, and O. Sekkas, “Modeling biogas production from anaerobic wastewater treatment plants using radial basis function networks and differential evolution,” *Comput. Chem. Eng.*, vol. 157, p. 107629, Dec. 2021, doi: 10.1016/j.compchemeng.2021.107629.

[7] D. Karamichailidou, V. Kaloutsa, and A. Alexandridis, “Wind turbine power curve modeling using radial basis function neural networks and tabu search,” *Renew. Energy*, vol. 163, pp. 2137–2152, Jan. 2021, doi: 10.1016/j.renene.2020.10.020.

[8] D. Karamichailidou, S. Koletsios, and A. Alexandridis, “An RBF online learning scheme for non-stationary environments based on fuzzy means and Givens rotations,” *Neurocomputing*, vol. 501, pp. 370–386, 2022, doi: https://doi.org/10.1016/j.neucom.2022.06.016.

[9] A. Alexandridis, H. Sarimveis, and K. Ninos, “Advances in Engineering Software A Radial Basis Function network training algorithm using a non-symmetric partition of the input space – Application to a Model Predictive Control configuration,” *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 830–837, 2011, doi: 10.1016/j.advengsoft.2011.05.026.

[10] A. Alexandridis, M. Stogiannos, N. Papaioannou, E. Zois, and H. Sarimveis, “An inverse neural controller based on the applicability domain of RBF network models,” *Sensors (Switzerland)*, vol. 18, no. 1, 2018, doi: 10.3390/s18010315.

[11] J. Moody and C. J. Darken, “Fast Learning in Networks of Locally-Tuned Processing Units,” *Neural Comput.*, vol. 1, no. 2, pp. 281–294, 1989, doi: 10.1162/neco.1989.1.2.281.

[12] A. Leonardis and H. Bischof, “An efficient MDL-based construction of RBF networks.,” *Neural networks Off. J. Int. Neural Netw. Soc.*, vol. 11, no. 5, pp. 963–973, Jul. 1998, doi: 10.1016/s0893-6080(98)00051-3.

[13] T. R. Holcomb and M. Morari, “Local Training for Radial Basis Function Networks:

Towards Solving the Hidden Unit Problem," *1991 Am. Control Conf.*, pp. 2331–2336, 1991, [Online]. Available: https://api.semanticscholar.org/CorpusID:13026432

[14] H. Sarimveis, A. Alexandridis, G. Tsekouras, and G. Bafas, "A fast and efficient algorithm for training radial basis function neural networks based on a fuzzy partition of the input space," *Ind. Eng. Chem. Res.*, vol. 41, no. 4, pp. 751–759, Feb. 2002, doi: 10.1021/ie010263h.

[15] M. Papadimitrakis and A. Alexandridis, "Active vehicle suspension control using road preview model predictive control and radial basis function networks," *Appl. Soft Comput.*, vol. 120, p. 108646, 2022, doi: https://doi.org/10.1016/j.asoc.2022.108646.

[16] K. Ninos, C. Giannakakis, I. Kompogiannis, I. Stavrakas, and A. Alexandridis, "Nonlinear Control of a DC-Motor Based on Radial Basis Function Neural Networks," no. 1, pp. 611–615, 2011.

[17] A. Alexandridis, E. Chondrodima, and H. Sarimveis, "Radial basis function network training using a nonsymmetric partition of the input space and particle swarm optimization," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 24, no. 2, pp. 219–230, 2013, doi: 10.1109/TNNLS.2012.2227794.

[18] S. Bouabdallah and R. Siegwart, "Full control of a quadrotor," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 153–158. doi: 10.1109/IROS.2007.4399042.

[19] M. Van, M. Mavrovouniotis, and S. S. Ge, "An Adaptive Backstepping Nonsingular Fast Terminal Sliding Mode Control for Robust Fault Tolerant Control of Robot Manipulators," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 49, no. 7, pp. 1448–1458, 2019, doi: 10.1109/TSMC.2017.2782246.

[20] C. Wen, J. Zhou, Z. Liu, and H. Su, "Robust Adaptive Control of Uncertain Nonlinear Systems in the Presence of Input Saturation and External Disturbance," *IEEE Trans. Automat. Contr.*, vol. 56, no. 7, pp. 1672–1678, 2011, doi: 10.1109/TAC.2011.2122730.

[21] H. Wang *et al.*, "Data-driven model predictive control of transcritical CO2 systems for cabin thermal management in cooling mode," *Appl. Therm. Eng.*, vol. 235, p. 121337, 2023, doi: https://doi.org/10.1016/j.applthermaleng.2023.121337.

[22] P. Stoffel, P. Henkel, M. Rätz, A. Kümpel, and D. Müller, "Safe operation of online learning data driven model predictive control of building energy systems," *Energy AI*, vol. 14, p. 100296, 2023, doi: https://doi.org/10.1016/j.egyai.2023.100296.

[23] X. Luo *et al.*, "Machine learning-based surrogate model assisting stochastic model predictive control of urban drainage systems," *J. Environ. Manage.*, vol. 346, p. 118974, 2023, doi: https://doi.org/10.1016/j.jenvman.2023.118974.

[24] S. Prabhu, S. Rangarajan, and M. Kothare, "Data-driven discovery of sparse dynamical model of cardiovascular system for model predictive control," *Comput. Biol. Med.*, vol. 166, p. 107513, 2023, doi: https://doi.org/10.1016/j.compbiomed.2023.107513.

[25] D. Q. Mayne, J. B. Rawlings, C. V Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000, doi: https://doi.org/10.1016/S0005-1098(99)00214-9.

[26] D. Q. Mayne, “Model predictive control: Recent developments and future promise,” *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014, doi: https://doi.org/10.1016/j.automatica.2014.10.128.

[27] M. G. Forbes, R. S. Patwardhan, H. Hamadah, and R. B. Gopaluni, “Model Predictive Control in Industry: Challenges and Opportunities,” *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 531–538, 2015, doi: https://doi.org/10.1016/j.ifacol.2015.09.022.

[28] S. J. Qin and T. A. Badgwell, “A survey of industrial model predictive control technology,” *Control Eng. Pract.*, vol. 11, no. 7, pp. 733–764, 2003, doi: https://doi.org/10.1016/S0967-0661(02)00186-7.

[29] M. Papadimitrakis, M. Stogiannos, H. Sarimveis, and A. Alexandridis, “Multi-Ship Control and Collision Avoidance Using MPC and RBF-Based Trajectory Predictions,” *Sensors*, vol. 21, no. 21, 2021, doi: 10.3390/s21216959.

[30] M. Stogiannos, A. Alexandridis, and H. Sarimveis, “Model predictive control for systems with fast dynamics using inverse neural models.,” *ISA Trans.*, vol. 72, pp. 161–177, Jan. 2018, doi: 10.1016/j.isatra.2017.09.016.

[31] G. A. Garcia, S. S. Keshmiri, and T. Stastny, “Robust and Adaptive Nonlinear Model Predictive Controller for Unsteady and Highly Nonlinear Unmanned Aircraft,” *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 4, pp. 1620–1627, 2015, doi: 10.1109/TCST.2014.2377711.

[32] J. H. Lee, “Model predictive control: Review of the three decades of development,” *Int. J. Control. Autom. Syst.*, vol. 9, no. 3, pp. 415–424, 2011, doi: 10.1007/s12555-011-0300-6.

[33] R. Findeisen, L. Imsland, F. Allgower, and B. A. Foss, “State and Output Feedback Nonlinear Model Predictive Control: An Overview,” *Eur. J. Control*, vol. 9, no. 2, pp. 190–206, 2003, doi: https://doi.org/10.3166/ejc.9.190-206.

[34] S. S. Keerthi and E. G. Gilbert, “Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations,” *J. Optim. Theory Appl.*, vol. 57, no. 2, pp. 265–293, 1988, doi: 10.1007/BF00938540.

[35] S. V Raković and M. Lazar, “Minkowski terminal cost functions for MPC,” *Automatica*, vol. 48, no. 10, pp. 2721–2725, 2012, doi: https://doi.org/10.1016/j.automatica.2012.06.075.

[36] P. O. M. Scokaert, D. Q. Mayne, and J. B. Rawlings, “Suboptimal model predictive control (feasibility implies stability),” *IEEE Trans. Automat. Contr.*, vol. 44, no. 3, pp. 648–654, 1999, doi: 10.1109/9.751369.

[37] A. Bemporad, “A Predictive Controller with Artificial Lyapunov Function for Linear Systems with Input/State Constraints,” *Automatica*, vol. 34, no. 10, pp. 1255–1260, 1998, doi: https://doi.org/10.1016/S0005-1098(98)00066-1.

[38] Robert Mahony, Vijay Kumar, and Peter Corke, “Modeling, Estimation, and Control of Quadrotor,” *Ieee*, no. SEPTEMBER, p. 32, 2012.

[39] C. S. Sharp, O. Shakernia, and S. S. Sastry, “A vision system for landing an unmanned aerial vehicle,” in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and*

Automation (Cat. No.01CH37164), 2001, vol. 2, pp. 1720–1727 vol.2. doi: 10.1109/ROBOT.2001.932859.

[40] E. Camci, D. R. Kripalani, L. Ma, E. Kayacan, and M. A. Khanesar, "An aerial robot for rice farm quality inspection with type-2 fuzzy neural networks tuned by particle swarm optimization-sliding mode control hybrid algorithm," *Swarm Evol. Comput.*, vol. 41, pp. 1–8, 2018, doi: https://doi.org/10.1016/j.swevo.2017.10.003.

[41] F. Fraundorfer *et al.*, "Vision-based autonomous mapping and exploration using a quadrotor MAV," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4557–4564. doi: 10.1109/IROS.2012.6385934.

[42] T. Tomic *et al.*, "Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue," *IEEE Robot. Autom. Mag.*, vol. 19, no. 3, pp. 46–56, 2012, doi: 10.1109/MRA.2012.2206473.

[43] K. A. Ghamry and Y. Zhang, "Cooperative control of multiple UAVs for forest fire monitoring and detection," in *2016 12th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, 2016, pp. 1–6. doi: 10.1109/MESA.2016.7587184.

[44] E. Capello, A. Scola, G. Guglieri, and F. Quagliotti, "Mini Quadrotor UAV: Design and Experiment," *J. Aerosp. Eng.*, vol. 25, no. 4, pp. 559–573, 2012, doi: 10.1061/(asce)as.1943-5525.0000171.

[45] S. Bouabdallah, "Design and Control of Quadrotors With Application To Autonomous Flying," *École Polytech. Fédérale Lausanne, À La Fac. Des Sci. Tech. L'Ingénieur*, vol. 3727, no. 3727, p. 61, 2007, [Online]. Available: http://biblion.epfl.ch/EPFL/theses/2007/3727/EPFL_TH3727.pdf

[46] M. Önder Efe, "Sliding mode control for unmanned aerial vehicles research," *Stud. Syst. Decis. Control*, vol. 24, pp. 239–255, 2015, doi: 10.1007/978-3-319-18290-2_12.

[47] Y. Alaiwi and A. Mutlu, "Modelling, Simulation and Implementation of Autonomous Unmanned Quadrotor," *Int. J. Sci. Tech. Innov. Ind. "Machines. Technol. Mater.*, vol. 12, no. 8, pp. 320–325, 2018.

[48] J. Zhang, D. Gu, Z. Ren, and B. Wen, "Robust trajectory tracking controller for quadrotor helicopter based on a novel composite control scheme," Elsevier Masson SAS, 2019. doi: 10.1016/j.ast.2018.12.013.

[49] X. Li, H. Zhang, W. Fan, C. Wang, and P. Ma, "Finite-time control for quadrotor based on composite barrier Lyapunov function with system state constraints and actuator faults," *Aerosp. Sci. Technol.*, vol. 119, p. 107063, 2021, doi: 10.1016/j.ast.2021.107063.

[50] B. T. M. Leong, S. M. Low, and M. P. L. Ooi, "Low-cost microcontroller-based hover control design of a quadcopter," *Procedia Eng.*, vol. 41, no. Iris, pp. 458–464, 2012, doi: 10.1016/j.proeng.2012.07.198.

[51] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, "A prototype of an autonomous controller for a quadrotor UAV," *2007 Eur. Control Conf. ECC 2007*, pp. 4001–4008, 2007, doi: 10.23919/ECC.2007.7068316.

[52] L. D. Minh and C. Ha, “Modeling and control of quadrotor MAV using vision-based measurement,” in *International Forum on Strategic Technology 2010*, 2010, pp. 70–75. doi: 10.1109/IFOST.2010.5668079.

[53] S. Bouabdallah, A. Noth, and R. Siegwart, “PID vs LQ control techniques applied to an indoor micro quadrotor,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 2004, vol. 3, pp. 2451–2456 vol.3. doi: 10.1109/IROS.2004.1389776.

[54] H. Wang, Z. Li, H. Xiong, and X. Nian, “Robust H∞ attitude tracking control of a quadrotor UAV on SO(3) via variation-based linearization and interval matrix approach,” *ISA Trans.*, vol. 87, no. 3, pp. 10–16, 2019, doi: 10.1016/j.isatra.2018.11.015.

[55] G. V Raffo, M. Ortega, and F. Rubio, “Backstepping/nonlinear H∞ control for path tracking of a quadrotor unmanned aerial vehicle,” in *Proceedings of the American Control Conference*, Jul. 2008, pp. 3356–3361. doi: 10.1109/ACC.2008.4587010.

[56] G. V Raffo, M. G. Ortega, and F. R. Rubio, “An Integral Predictive/Nonlinear H∞ Control Structure for a Quadrotor Helicopter,” *Automatica*, vol. 46, no. 1, pp. 29–39, Jan. 2010, doi: 10.1016/j.automatica.2009.10.018.

[57] A. Das, K. Subbarao, and F. Lewis, “Dynamic inversion of quadrotor with zero-dynamics stabilization,” in *2008 IEEE International Conference on Control Applications*, 2008, pp. 1189–1194. doi: 10.1109/CCA.2008.4629582.

[58] H. Voos, “Nonlinear control of a quadrotor micro-UAV using feedback-linearization,” in *2009 IEEE International Conference on Mechatronics*, 2009, pp. 1–6. doi: 10.1109/ICMECH.2009.4957154.

[59] Y.-C. Choi and H.-S. Ahn, “Nonlinear Control of Quadrotor for Point Tracking: Actual Implementation and Experimental Tests,” *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 3, pp. 1179–1192, 2015, doi: 10.1109/TMECH.2014.2329945.

[60] H. Sira-Ramirez, M. Zribi, and S. Ahmad, “Dynamical sliding mode control approach for vertical flight regulation in helicopters,” *IEE Proc. Control Theory Appl.*, vol. 141, no. 1, pp. 19–24, 1994, doi: 10.1049/ip-cta:19949624.

[61] H. Wang, Y. Xuefei, Y. Tian, G. Zheng, and N. Christov, “Model-Free Based Terminal SMC of Quadrotor Attitude and Position,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, Oct. 2016, doi: 10.1109/TAES.2016.150303.

[62] H. Razmi and S. Afshinfar, “Neural network-based adaptive sliding mode control design for position and attitude control of a quadrotor UAV,” *Aerosp. Sci. Technol.*, vol. 91, pp. 12–27, 2019, doi: https://doi.org/10.1016/j.ast.2019.04.055.

[63] R. Miranda-Colorado, L. T. Aguilar, and J. E. Herrero-Brito, “Reduction of power consumption on quadrotor vehicles via trajectory design and a controller-gains tuning stage,” *Aerosp. Sci. Technol.*, vol. 78, pp. 280–296, Jul. 2018, doi: 10.1016/J.AST.2018.04.027.

[64] P. Tang, F. Zhang, J. Ye, and D. Lin, “An integral TSMC-based adaptive fault-tolerant control for quadrotor with external disturbances and parametric uncertainties,” *Aerosp. Sci. Technol.*, vol. 109, p. 106415, 2021, doi: https://doi.org/10.1016/j.ast.2020.106415.

[65] H. Liu, W. Zhao, S. Hong, F. Lewis, and Y. Yu, "Robust Backstepping-Based Trajectory Tracking Control for Quadrotors with Time Delays," *IET Control Theory Appl.*, vol. 13, pp. 1945 – 1954, Aug. 2019, doi: 10.1049/iet-cta.2018.6043.

[66] N. Koksal, H. An, and B. Fidan, "Backstepping-based adaptive control of a quadrotor UAV with guaranteed tracking performance," *ISA Trans.*, vol. 105, pp. 98–110, 2020, doi: 10.1016/j.isatra.2020.06.006.

[67] W. Liu, X. Cheng, and J. Zhang, "Command filter-based adaptive fuzzy integral backstepping control for quadrotor UAV with input saturation," *J. Franklin Inst.*, vol. 360, no. 1, pp. 484–507, 2023, doi: 10.1016/j.jfranklin.2022.10.042.

[68] N. P. Nguyen and S. K. Hong, "Active fault-tolerant control of a quadcopter against time-varying actuator faults and saturations using sliding mode backstepping approach," *Appl. Sci.*, vol. 9, no. 19, 2019, doi: 10.3390/app9194010.

[69] X. Heng, D. Cabecinhas, R. Cunha, C. Silvestre, and X. Qingsong, "A trajectory tracking LQR controller for a quadrotor: Design and experimental evaluation," *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2016-Janua, pp. 1–7, 2016, doi: 10.1109/TENCON.2015.7372729.

[70] C. Nicol, C. J. B. Macnab, and A. Ramirez-Serrano, "Robust neural network control of a quadrotor helicopter," *Can. Conf. Electr. Comput. Eng.*, no. 1, pp. 1233–1237, 2008, doi: 10.1109/CCECE.2008.4564736.

[71] T. Madani and A. Benallegue, *Adaptive Control via Backstepping Technique and Neural Networks of a Quadrotor Helicopter*, vol. 41, no. 2. IFAC, 2008. doi: 10.3182/20080706-5-kr-1001.01098.

[72] M. Wang, B. Chen, and C. Lin, "Fixed-TimeBackstepping Control of Quadrotor Trajectory Tracking Based On Neural Network," *IEEE Access*, vol. 8, pp. 177092–177099, Jan. 2020, doi: 10.1109/ACCESS.2020.3027052.

[73] X. Yu, Z. Lv, Y. Wu, and X. M. Sun, "Neural Network Modeling and Backstepping Control for Quadrotor," in *Proceedings 2018 Chinese Automation Congress, CAC 2018*, Jan. 2019, pp. 3649–3654. doi: 10.1109/CAC.2018.8623432.

[74] K. Liu, R. Wang, X. Wang, and X. Wang, "Anti-saturation adaptive finite-time neural network based fault-tolerant tracking control for a quadrotor UAV with external disturbances," *Aerosp. Sci. Technol.*, vol. 115, p. 106790, 2021, doi: 10.1016/j.ast.2021.106790.

[75] X. Lin, Y. Wang, and Y. Liu, "Neural-network-based robust terminal sliding-mode control of quadrotor," *Asian J. Control*, vol. 24, no. 1, pp. 427–438, 2022, doi: 10.1002/asjc.2478.

[76] A. Alexandridis, H. Sarimveis, and K. Ninos, "A Radial Basis Function Network Training Algorithm Using a Non-Symmetric Partition of the Input Space - Application to a Model Predictive Control Configuration," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 830–837, Oct. 2011, doi: 10.1016/j.advengsoft.2011.05.026.

[77] D. T. Tran, M. N. Nguyen, and K. K. Ahn, "RBF neural network based backstepping control for an electrohydraulic elastic manipulator," *Appl. Sci.*, vol. 9, no. 11, 2019, doi: 10.3390/app9112237.

[78] H. Hu, Y. Song, P. Fan, C. Diao, and N. Cai, "A Backstepping Controller with the RBF Neural Network for Folding-Boom Aerial Work Platform," *Complexity*, vol. 2022, 2022, doi: 10.1155/2022/4289111.

[79] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Constrained optimal attitude control of a quadrotor helicopter subject to wind-gusts: Experimental studies," in *Proceedings of the 2010 American Control Conference*, 2010, pp. 4451–4455. doi: 10.1109/ACC.2010.5531005.

[80] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Model predictive quadrotor control: Attitude, altitude and position experimental studies," *IET Control Theory Appl.*, vol. 6, no. 12, pp. 1812–1827, 2012, doi: 10.1049/iet-cta.2011.0348.

[81] G. V Raffo, M. G. Ortega, and F. R. Rubio, "MPC with Nonlinear $\mathscr{H}\infty$ Control for Path Tracking of a Quad-Rotor Helicopter," *IFAC Proc. Vol.*, vol. 41, no. 2, pp. 8564–8569, 2008, doi: https://doi.org/10.3182/20080706-5-KR-1001.01448.

[82] K. Chang, D. Ma, T. Li, and Y. Xia, "Active Disturbance Rejection and Predictive Control Strategy for a Quadrotor Helicopter," *IET Control Theory Appl.*, vol. 10, Jul. 2016, doi: 10.1049/iet-cta.2016.0125.

[83] J. H. Holland, "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence." The MIT Press, Apr. 29, 1992. doi: 10.7551/mitpress/1090.001.0001.

[84] M. Fleischer, "Simulated annealing: past, present, and future," in *Winter Simulation Conference Proceedings, 1995.*, 1995, pp. 155–161. doi: 10.1109/WSC.1995.478718.

[85] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, 2006, doi: 10.1109/MCI.2006.329691.

[86] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, vol. 4, pp. 1942–1948 vol.4. doi: 10.1109/ICNN.1995.488968.

[87] M. J. Mahmoodabadi, "State-varying optimal decoupled sliding mode control for the Lorenz chaotic nonlinear problem based on HEPSO and MLS," *Int. J. Model. Simul.*, vol. 41, no. 6, pp. 405–414, 2021, doi: 10.1080/02286203.2020.1772013.

[88] J. Tao and G. Sun, "Application of deep learning based multi-fidelity surrogate model to robust aerodynamic design optimization," *Aerosp. Sci. Technol.*, vol. 92, pp. 722–737, 2019, doi: https://doi.org/10.1016/j.ast.2019.07.002.

[89] H. Boubertakh, S. Bencharef, and S. Labiod, "PSO-based PID control design for the stabilization of a quadrotor," in *3rd International Conference on Systems and Control*, 2013, pp. 514–517. doi: 10.1109/ICoSC.2013.6750908.

[90] Y. Wang, Y. Chenxie, J. Tan, C. Wang, Y. Wang, and Y. Zhang, "Fuzzy radial basis function neural network PID control system for a quadrotor UAV based on particle swarm optimization," in *2015 IEEE International Conference on Information and Automation*, 2015, pp. 2580–2585. doi: 10.1109/ICInfA.2015.7279720.

[91] F. Yacef, O. Bouhali, M. Hamerlain, and A. Rezoug, "PSO optimization of Integral

Backstepping Controller for Quadrotor attitude stabilization," in *3rd International Conference on Systems and Control*, 2013, pp. 462–466. doi: 10.1109/ICoSC.2013.6750900.

[92] X. Luo *et al.*, "Optimal path planning for UAV based inspection system of large-scale photovoltaic farm," in *2017 Chinese Automation Congress (CAC)*, 2017, pp. 4495–4500. doi: 10.1109/CAC.2017.8243572.

[93] C. Peng *et al.*, "ADRC trajectory tracking control based on PSO algorithm for a quad-rotor," in *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, 2013, pp. 800–805. doi: 10.1109/ICIEA.2013.6566476.

[94] J.-J. Wang and G.-Y. Liu, "Saturated control design of a quadrotor with heterogeneous comprehensive learning particle swarm optimization," *Swarm Evol. Comput.*, vol. 46, pp. 84–96, 2019, doi: https://doi.org/10.1016/j.swevo.2019.02.008.

[95] M. J. Mahmoodabadi and N. Rezaee Babak, "Robust fuzzy linear quadratic regulator control optimized by multi-objective high exploration particle swarm optimization for a 4 degree-of-freedom quadrotor," *Aerosp. Sci. Technol.*, vol. 97, p. 105598, 2020, doi: https://doi.org/10.1016/j.ast.2019.105598.

[96] A. Alexandridis, E. Chondrodima, and H. Sarimveis, "Cooperative learning for radial basis function networks using particle swarm optimization," *Appl. Soft Comput. J.*, vol. 49, pp. 485–497, 2016, doi: 10.1016/j.asoc.2016.08.032.

[97] S. Ono, H. Maeda, K. Sakimoto, and S. Nakayama, "User-system cooperative evolutionary computation for both quantitative and qualitative objective optimization in image processing filter design," *Appl. Soft Comput.*, vol. 15, pp. 203–218, 2014, doi: https://doi.org/10.1016/j.asoc.2013.10.019.

[98] F. B. de Oliveira, R. Enayatifar, H. J. Sadaei, F. G. Guimarães, and J.-Y. Potvin, "A cooperative coevolutionary algorithm for the Multi-Depot Vehicle Routing Problem," *Expert Syst. Appl.*, vol. 43, pp. 117–130, 2016, doi: https://doi.org/10.1016/j.eswa.2015.08.030.

[99] C. R. Cutler and B. L. Ramaker, "Dynamic matrix control A computer control algorithm," *IEEE Trans. Automat. Contr.*, vol. 17, p. 72, 1979, [Online]. Available: https://api.semanticscholar.org/CorpusID:122480259

[100] Y. M. Ren *et al.*, "A tutorial review of neural network modeling approaches for model predictive control," *Comput. Chem. Eng.*, vol. 165, no. July, 2022, doi: 10.1016/j.compchemeng.2022.107956.

[101] N. Lanzetti, Y. Z. Lian, A. Cortinovis, L. Dominguez, M. Mercangöz, and C. Jones, "Recurrent Neural Network based MPC for Process Industries," in *2019 18th European Control Conference (ECC)*, 2019, pp. 1005–1010. doi: 10.23919/ECC.2019.8795809.

[102] L. Ma, X. Liu, X. Kong, and K. Y. Lee, "Iterative Learning Model Predictive Control Based on Iterative Data-Driven Modeling," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 8, pp. 3377–3390, 2021, doi: 10.1109/TNNLS.2020.3016295.

[103] E. Kang, H. Qiao, Z. Chen, and J. Gao, "Tracking of Uncertain Robotic Manipulators Using Event-Triggered Model Predictive Control With Learning Terminal Cost," *IEEE Trans.*

Autom. Sci. Eng., vol. 19, no. 4, pp. 2801–2815, 2022, doi: 10.1109/TASE.2022.3152166.

[104] M. Shouche, H. Genceli, P. Vuthandam, and M. Nikolaou, "Simultaneous constrained model predictive control and identification of DARX processes," *Automatica*, vol. 34, no. 12, pp. 1521–1530, 1998, doi: 10.1016/S0005-1098(98)80005-8.

[105] H. Genceli and M. Nikolaou, "A New Approach to Model Predictive Control and Identification," *IFAC Proc. Vol.*, vol. 29, no. 1, pp. 4557–4562, 1996, doi: https://doi.org/10.1016/S1474-6670(17)58400-1.

[106] P. Vuthandam and M. Nikolaou, *A computationally efficient formulation of simultaneous constrained model predictive control and identification*, vol. 3. 1997. doi: 10.1109/ACC.1997.610859.

[107] R. Hedjar, "Adaptive neural network model predictive control," *Int. J. Innov. Comput. Inf. Control*, vol. 9, no. 3, pp. 1245–1257, 2013.

[108] Y. C. Lin, D.-D. Chen, M.-S. Chen, X.-M. Chen, and J. Li, "A precise BP neural network-based online model predictive control strategy for die forging hydraulic press machine," *Neural Comput. Appl.*, vol. 29, no. 9, pp. 585–596, 2018, doi: 10.1007/s00521-016-2556-5.

[109] G. Carughi, G. Ducard, and C. Onder, "Online Neural-Network Learning and Model Predictive Control Applied to a Tilt-Rotor Unmanned Aerial Vehicle," in *2022 IEEE 17th International Conference on Control & Automation (ICCA)*, 2022, pp. 31–37. doi: 10.1109/ICCA54724.2022.9831923.

[110] A. Alexandridis and H. Sarimveis, *Control of processes with multiple steady states using MPC and RBF neural networks*, vol. 29, no. 2002. Elsevier B.V., 2011. doi: 10.1016/B978-0-444-53711-9.50140-1.

[111] H. G. Han, X. L. Wu, and J. F. Qiao, "Real-time model predictive control using a self-organizing neural network," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 24, no. 9, pp. 1425–1436, 2013, doi: 10.1109/TNNLS.2013.2261574.

[112] H. G. Han, L. Zhang, Y. Hou, and J. F. Qiao, "Nonlinear Model Predictive Control Based on a Self-Organizing Recurrent Neural Network," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 27, no. 2, pp. 402–415, 2016, doi: 10.1109/TNNLS.2015.2465174.

[113] Andries_P._Engelbrecht, "Andries_P._Engelbrecht_Computational_IntelligencBookZZ.org.pdf." p. 630, 2007.

[114] A. P. Alexandridis, C. I. Siettos, H. K. Sarimveis, A. G. Boudouvis, and G. V Bafas, "Modelling of nonlinear process dynamics using Kohonen's neural networks, fuzzy systems and Chebyshev series," in *European Symposium on Computer Aided Process Engineering - 11*, vol. 9, R. Gani and S. B. B. T.-C. A. C. E. Jørgensen, Eds. Elsevier, 2001, pp. 69–74. doi: https://doi.org/10.1016/S1570-7946(01)80007-9.

[115] M. Stogiannos, A. Alexandridis, and H. Sarimveis, "An enhanced decentralized artificial immune-based strategy formulation algorithm for swarms of autonomous vehicles," *Appl. Soft Comput.*, vol. 89, p. 106135, 2020, doi: https://doi.org/10.1016/j.asoc.2020.106135.

[116] J. Kennedy, "Swarm Intelligence BT - Handbook of Nature-Inspired and Innovative

Computing: Integrating Classical Models with Emerging Technologies," A. Y. Zomaya, Ed. Boston, MA: Springer US, 2006, pp. 187–219. doi: 10.1007/0-387-27705-6_6.

[117] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. USA: Prentice Hall PTR, 1998.

[118] A. Engelbrecht, "Particle swarm optimization: Velocity initialization," in *2012 IEEE Congress on Evolutionary Computation*, 2012, pp. 1–8. doi: 10.1109/CEC.2012.6256112.

[119] S. Y. Lim, M. Montakhab, and H. Nouri, "A constriction factor based particle swarm optimization for economic dispatch," *ESM 2009 - 2009 Eur. Simul. Model. Conf. Model. Simul. 2009*, no. Gaing, pp. 305–311, 2009.

[120] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham, "Inertia Weight strategies in Particle Swarm Optimization," in *2011 Third World Congress on Nature and Biologically Inspired Computing*, 2011, pp. 633–640. doi: 10.1109/NaBIC.2011.6089659.

[121] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to participle swam optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, 2004, doi: 10.1109/TEVC.2004.826069.

[122] C.-K. Goh and K. C. Tan, "A Competitive-Cooperative Coevolutionary Paradigm for Dynamic Multiobjective Optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 1, pp. 103–127, 2009, doi: 10.1109/TEVC.2008.920671.

[123] K. C. Tan, Y. H. Chew, T. H. Lee, and Y. J. Yang, "A cooperative coevolutionary algorithm for multiobjective optimization," in *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483)*, 2003, vol. 1, pp. 390–395 vol.1. doi: 10.1109/ICSMC.2003.1243847.

[124] F. Bergh and A. Engelbrecht, "Cooperative Learning in Neural Networks using Particle Swarm Optimizers," *South African Comput. J.*, vol. 26, pp. 84–90, Jan. 2000.

[125] M. Papadimitrakis, "Development of optimization and data-driven model predictive control methods using computational intelligence techniques : Design and applications with emphasis on the economic operation of engineering systems," PhD Thesis, 2023. doi: http://dx.doi.org/10.26265/polynoe-5347.

[126] M. Papadimitrakis, A. Kapnopoulos, S. Tsavartzidis, and A. Alexandridis, "A cooperative PSO algorithm for Volt-VAR optimization in smart distribution grids," *Electr. Power Syst. Res.*, vol. 212, p. 108618, 2022, doi: https://doi.org/10.1016/j.epsr.2022.108618.

[127] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, no. 2, p. 26113, Feb. 2004, doi: 10.1103/PhysRevE.69.026113.

[128] K. P. Schneider *et al.*, "Analytic Considerations and Design Basis for the IEEE Distribution Test Feeders," *IEEE Trans. Power Syst.*, vol. 33, no. 3, pp. 3181–3188, 2018, doi: 10.1109/TPWRS.2017.2760011.

[129] H. Zhang, B. Liu, and H. Wu, "Smart Grid Cyber-Physical Attack and Defense: A Review,"

IEEE Access, vol. 9, pp. 29641–29659, 2021, doi: 10.1109/ACCESS.2021.3058628.

[130] M. Erik, H. Pedersen, and M. E. H. Pedersen, “Good parameters for particle swarm optimization,” *Tech. Rep. HL1001, Hvass Lab.*, vol. HL1001, pp. 1–12, 2010.

[131] A. Rezaee Jordehi and J. Jasni, “Parameter selection in particle swarm optimisation: A survey,” *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 25, no. 4. Taylor & Francis, pp. 527–542, 2013. doi: 10.1080/0952813X.2013.782348.

[132] E. H. Houssein, A. G. Gad, K. Hussain, and P. N. Suganthan, “Major Advances in Particle Swarm Optimization: Theory, Analysis, and Application,” *Swarm Evol. Comput.*, vol. 63, no. February, p. 100868, 2021, doi: 10.1016/j.swevo.2021.100868.

[133] M. Papadimitrakis, N. Giamarelos, M. Stogiannos, E. N. Zois, N. A. Livanos, and A. Alexandridis, “Metaheuristic search in smart grid : A review with emphasis on planning , scheduling and power flow optimization applications,” *Renew. Sustain. Energy Rev.*, vol. 145, no. April, p. 111072, 2021, doi: 10.1016/j.rser.2021.111072.

[134] B. Zhao, Z. Xu, C. Xu, C. Wang, and F. Lin, “Network Partition-Based Zonal Voltage Control for Distribution Networks with Distributed PV Systems,” *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 4087–4098, 2018, doi: 10.1109/TSG.2017.2648779.

[135] D. Karamichailidou, A. Alexandridis, G. Anagnostopoulos, G. Syriopoulos, and O. Sekkas, “Modeling biogas production from anaerobic wastewater treatment plants using radial basis function networks and differential evolution,” *Comput. Chem. Eng.*, vol. 157, p. 107629, 2022, doi: https://doi.org/10.1016/j.compchemeng.2021.107629.

[136] A. Kapnopoulos and A. Alexandridis, “A cooperative particle swarm optimization approach for tuning an MPC-based quadrotor trajectory tracking scheme,” *Aerosp. Sci. Technol.*, vol. 127, p. 107725, 2022, doi: https://doi.org/10.1016/j.ast.2022.107725.

[137] I. Kalogeropoulos, A. Alexandridis, and H. Sarimveis, “Economic Oriented Dynamic Matrix Control of Wastewater Treatment Plants,” *J. Process Control*, vol. 118, pp. 202–217, 2022, doi: https://doi.org/10.1016/j.jprocont.2022.08.006.

[138] J. Alex *et al.*, “Benchmark Simulation Model no. 1 (BSM1),” *Rep. by IWA Taskgr. Benchmarking Control Strateg. WWTPs*, Jan. 2008.

[139] E. Mezura-Montes and C. Coello, “Constraint-handling in nature-inspired numerical optimization: Past, present and future,” *Swarm Evol. Comput.*, vol. 1, pp. 173–194, Dec. 2011, doi: 10.1016/j.swevo.2011.10.001.

[140] M. Driels, *Linear Control Systems Engineering*, 1st ed. McGraw-Hill Higher Education, 1995.

[141] H. Khalil, *Nonlinear Systems*, Vol. 2 no. New Jersey,: Prentice-Hall, 2003.

[142] A. Alexandridis and H. Sarimveis, “Nonlinear adaptive model predictive control based on self-correcting neural network models,” *AIChE J.*, vol. 51, pp. 2495–2506, Sep. 2005, doi: 10.1002/aic.10505.

[143] Y. Wang, L. Xie, and C. E. de Souza, “Robust control of a class of uncertain nonlinear systems,” *Syst. Control Lett.*, vol. 19, no. 2, pp. 139–149, 1992, doi:

https://doi.org/10.1016/0167-6911(92)90097-C.

[144] W. Li and M. Krstic, “Stochastic Nonlinear Prescribed-Time Stabilization and Inverse Optimality,” *IEEE Trans. Automat. Contr.*, vol. 67, no. 3, pp. 1179–1193, 2022, doi: 10.1109/TAC.2021.3061646.

[145] R. A. FREEMAN, M. KRSTIĆ, and P. V KOKOTOVIĆ, “Robustness of Adaptive Nonlinear Control to Bounded Uncertainties,” *Automatica*, vol. 34, no. 10, pp. 1227–1230, 1998, doi: https://doi.org/10.1016/S0005-1098(98)00070-3.

[146] M. Krstic, P. V Kokotovic, and I. Kanellakopoulos, *Nonlinear and Adaptive Control Design*, 1st ed. USA: John Wiley \& Sons, Inc., 1995.

[147] E. F. Camacho and C. Bordons, *Model Predictive Control - Second edition*. 2007.

[148] S. Keerthi and E. Gilbert, “An existence theorem for discrete-time infinite-horizon optimal control problems,” *IEEE Trans. Automat. Contr.*, vol. 30, no. 9, pp. 907–909, 1985, doi: 10.1109/TAC.1985.1104084.

[149] J. Dolezal, “Existence of Optimal Solutions in General Discrete Systems.,” *Kybernetika*, vol. 11, no. 4, pp. 301–312, 1975.

[150] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*. 2017. doi: 10.1007/978-3-319-46024-6.

[151] P. Tang, D. Lin, D. Zheng, S. Fan, and J. Ye, “Observer based finite-time fault tolerant quadrotor attitude control with actuator faults,” *Aerosp. Sci. Technol.*, vol. 104, p. 105968, 2020, doi: https://doi.org/10.1016/j.ast.2020.105968.

[152] P. Castillo, R. Lozano, and A. Dzul, “Stabilization of a mini rotorcraft with four rotors,” *IEEE Control Syst. Mag.*, vol. 25, no. 6, pp. 45–55, Dec. 2005, doi: 10.1109/MCS.2005.1550152.

[153] T. Bresciani, “Modelling, Identification and Control of a Quadrotor Helicopter,” *Master’s Thesis*, no. October, pp. 1–184, 2008.

[154] M. A. Johnson *et al.*, *PID control: New identification and design methods*. 2005. doi: 10.1007/1-84628-148-2.

[155] D. Mellinger, “Trajectory Generation and Control for Quadrotors,” Jan. 2012.

[156] M. Islam, M. Okasha, and M. Idres, “Dynamics and control of quadcopter using linear model predictive control approach,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 270, p. 12007, Dec. 2017, doi: 10.1088/1757-899X/270/1/012007.

[157] Z. He and L. Zhao, “Internal model control /backstepping sliding model control for quadrotor trajectory tracking,” in *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2017, pp. 1254–1258. doi: 10.1109/ITNEC.2017.8284977.

[158] Q. Shen, B. Jiang, and P. Shi, *Studies in Systems, Decision and Control 91 Fault Diagnosis and Fault-Tolerant Control Based on Adaptive Control Approach*, vol. 91. 2017. [Online]. Available: http://www.springer.com/series/13304

[159] F. Zhang, Y. Yu, Q. Wang, X. Zeng, and H. Niu, "A terrain-adaptive robot prototype designed for bumpy-surface exploration," *Mech. Mach. Theory*, vol. 141, pp. 213–225, 2019, doi: https://doi.org/10.1016/j.mechmachtheory.2019.07.008.

[160] M. A. H. Dempster and V. Leemans, "An automated FX trading system using adaptive reinforcement learning," *Expert Syst. Appl.*, vol. 30, no. 3, pp. 543–552, 2006, doi: https://doi.org/10.1016/j.eswa.2005.10.012.

[161] M. O'Connell, G. Shi, X. Shi, and S.-J. Chung, "Meta-Learning-Based Robust Adaptive Flight Control Under Uncertain Wind Conditions," in *International Conference on Robotics and Automation (ICRA),* Mar. 2020.

[162] A. Alexandridis, H. Sarimveis, and G. Bafas, "A new algorithm for online structure and parameter adaptation of RBF networks," *Neural Networks,* vol. 16, no. 7, pp. 1003–1017, 2003, doi: 10.1016/S0893-6080(03)00052-2.

[163] B. Åström, Karl Johan Wittenmark, *Adaptive control 1ed Astrom*. 1989.

[164] N. Kazantzis and C. Kravaris, "Synthesis of state feedback regulators for nonlinear processes," *Chem. Eng. Sci.*, vol. 55, no. 17, pp. 3437–3449, 2000, doi: 10.1016/S0009-2509(00)00006-3.

[165] D. Emmanouil, "Development of online training methods for radial basis function neural networks," Master's Thesis, 2023. doi: http://dx.doi.org/10.26265/polynoe-4696.