National Technical University of Athens

School of Mechanical Engineering

Section of Mechanical Design and Automatic Control

Control Systems Laboratory

Development of Predictive Navigation Schemes for Aircraft-like Vehicles

SPYROS MANIATOPOULOS

Submitted in partial fulfillment of the requirements for the degree of

DIPLOMA IN MECHANICAL ENGINEERING

Diploma Thesis Advisor: Professor Kostas J. Kyriakopoulos

Athens, March 2012

[This page intentionally left blank.]

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Μηχανολόγων Μηχανικών

Τομέας Μηχανολογικών Κατασκευών και Αυτομάτου Ελέγχου

Εργαστήριο Αυτομάτου Ελέγχου

Ανάπτυξη Μεθοδολογιών Προβλεπτικής Πλοήγησης για Οχήματα τύπου Αεροπλάνου

ΣΠΥΡΟΣ ΜΑΝΙΑΤΟΠΟΥΛΟΣ

Κατατέθηκε για την εκπλήρωση των υποχρεώσεων για την απόκτηση του τίτλου του

ΔΙΠΛΩΜΑΤΟΥΧΟΥ ΜΗΧΑΝΟΛΟΓΟΥ ΜΗΧΑΝΙΚΟΥ

Υπεύθυνος Καθηγητής: Κωνσταντίνος Ι. Κυριακόπουλος

Αθήνα, Μάρτιος 2012

Development of Predictive Navigation Schemes for Aircraft-like Vehicles

Diploma Thesis

Contact info:

sp.maniato@gmail.com

smaniato@ieee.org

Submitted in Partial Fulfillment of the Requirements
for the Degree of Diploma in Mechanical Engineering
in the School of Mechanical Engineering at the
National Technical University of Athens, 2012

Athens, Greece

"The significant problems we face cannot be solved at the same level of thinking we were at when we created them."
- Albert Einstein

"We are what we repeatedly do.
Excellence, then, is not an act, but a habit."
- Stephen R. Covey

"If a man does not face adversity in life, he cannot develop a reliable and robust personality."
- Yamamoto Tsunetomo

Acknowledgments

First and foremost, I would like to thank my advisor, Professor Kostas J. Kyriakopoulos for our collaboration over the past one and a half years. He inspired and motivated me to excel. His guidance and deep knowledge of the field of Robotics and Controls were crucial to my success. I am also grateful for his advice during the PhD applications process. I am certain that his support and recommendation influenced the outcome the most.

In addition, I would like to thank the other two members of my Diploma thesis examining committee, for their help as course instructors and for inspiring me, along with Prof. Kyriakopoulos, to delve into the area of Dynamics, Controls, and Robotics in the first place. I am especially thankful to Prof. Papadopoulos for supporting my decision to continue my studies in the United States, via thoughtful advice and a letter of recommendation.

Furthermore, I want to express my gratitude to Ass. Professor Dimos Dimarogonas – KTH Royal Institute of Technology, Stockholm, Sweden – for his invitation to visit KTH in May 2011. His experience in the field of multi-agent systems had a massive impact on my work. Our collaboration resulted in the submission of a research paper [1] that will appear at *The 2012 American Control Conference*. I am also thankful to him for his advice, support, and recommendation regarding my PhD applications.

I would also like to thank my labmates at NTUA's Control Systems Lab. Namely, Dimitra Panagou, Giannis Roussos, Alina Eqtami, Minas Liarokapis, Pantelis Katsiaris, Panos Marantos, George Karras, Shahab Heshmati, Ioannis Filippidis, George Karavas, Dimitris Pylorof and Kostas Vasilakis. They were all both great colleagues and trusted friends.

Last but not least, I would like to thank my parents, Stamatia and Michail, my grandmother, Efstathia, and my brother, Marios, for their support and understanding.

To conclude, I would like to thank Stefan Schäckeler, whose LaTeX Ph.D. thesis template I used as a reference to typeset this work, as well as Donald Knuth, the creator of TeX.

Spyros Maniatopoulos
Athens, February 2012

Development of Predictive Navigation Schemes for Aircraft-like Vehicles

Spyros Maniatopoulos

School of Mechanical Engineering
National Technical University of Athens
Athens, Greece
2012

ABSTRACT

This thesis is concerned with the performance aspect of a two-dimensional navigation and collision avoidance problem. It considers both single and multiple vehicle settings. In particular, the vehicles (agents) considered in this work are aircraft-like in the sense of non-holonomic kinematics and bounded lower speed. Therefore agents are modeled as unicycles to resemble the kinematics of an aircraft flying at constant altitude. By performance we refer to task related criteria, like control effort, deviation from a nominal speed or path, arrival time etc.

The proposed scheme combines the Navigation Functions (NFs) methodology with the Model Predictive Control (MPC) framework. The motivation for this work was the fact that a robot navigating by tracking an artificial potential field's gradient direction cannot choose one trajectory over another. The specifics change depending on the setting (single or multi-agent navigation), but the main idea is to deviate from the direction of a NF's gradient. A NF-based (feedback) control law is used as reference and an added deviation term is calculated by solving a Finite Horizon Optimal Control Problem (FHOCP). The performance criteria are encoded in the FHOCP's cost functional. The lower speed bound is satisfied by an appropriate NF-based linear velocity control law. An important advantage of the proposed scheme is that the navigation properties (convergence, collision avoidance) of the NF-based control scheme are preserved.

Two settings are examined. In the single agent navigation setting, a single nonholonomic agent navigating in a workspace that contains static obstacles is considered. The artificial potential field used is an extension of the original NFs, a Dipolar NF. Dipolar NFs allow convergence to a desired orientation in addition to target destination. It should be noted that, in the single agent case, our contribution is only a minor extension of [2]. In the multi-agent setting, the case of N nonholonomic agents navigating in the same

workspace is examined. The multi-agent extension of the NFs methodology, a (dipolar) Decentralized Navigation Function (DNF), is used to generate the artificial potential. Furthermore two cases in the multi-agent setting are explored. Centralized predictive navigation and decentralized predictive navigation. The difference lies in the information available to each agent's predictive controller for the solution of the FHOCP. In the decentralized case, each agent does not take into account the decisions of others in the control law calculation (a form of uncertainty). Therefore we employ event-triggered (E-T) recalculations of the FHOCP. An event occurs when the discrepancy between the predicted an actual performance of the agent exceeds some limit.

To verify the efficacy of the proposed schemes, i.e., the improvement in performance with regard to (wrt) the (D)NF-based control laws, various navigation scenarios were simulated in MATLAB. Simulations also provide comparative results, demonstrating the difference in performance between each instance of the scheme: DNF-based control law (no deviation), Centralized predictive navigation, Decentralized predictive navigation (no E-T) and Decentralized E-T predictive navigation.

Keywords: Nonholonomic vehicle, Navigation Functions, Model Predictive Control, Predictive Navigation, Multi–agent system, Decentralized navigation, Event-triggered navigation, Air–Traffic Management

Ανάπτυξη Μεθοδολογιών Προβλεπτικής Πλοήγησης
για Οχήματα τύπου Αεροπλάνου

Σπύρος Μανιατόπουλος

Σχολή Μηχανολόγων Μηχανικών
Εθνικό Μετσόβιο Πολυτεχνείο
Αθήνα, Ελλάδα
2012

ΕΚΤΕΝΗΣ ΠΕΡΙΛΗΨΗ [1]

Σύνοψη

Αυτή η Διπλωματική εργασία ασχολείται με το ζήτημα της λειτουργικής απόδοσης ενός προβλήματος δυδιάστατης πλοήγησης και αποφυγής συγκρούσεων. Θα μελετήσουμε τόσο το πρόβλημα ενός οχήματος, όσο και αυτό των πολλών. Συγκεκριμένα, τα οχήματα (πράκτορες) είναι τύπου αεροπλάνου, υπό την έννοια ότι μοντελοποιούνται με μη-ολονομική κινηματική και κατώτατο όριο στην ταχύτητά τους. Με τον όρο απόδοση αναφερόμαστε σε κριτήρια σχετικά με το εκάστοτε σενάριο, όπως ελαχιστοποίηση της ενέργειας, απόκλιση από κάποια ονομαστική ταχύτητα ή τροχιά κ.α.

Το σχήμα ελέγχου που προτείνουμε συνδυάζει τη μέθοδο των Συναρτήσεων Πλοήγησης (ΣΠ) με αυτή του Προβλεπτικού Ελέγχου (Παραρτήματα A.1, A.2). Το κίνητρο για αυτή τη δουλειά είναι το γεγονός ότι ένα ρομπότ που κινείται ακολουθώντας την κλίση ενός τεχνητού δυναμικού πεδίου δεν μπορεί να επιλέξει την τροχιά του. Οπότε, η κεντρική ιδέα είναι το όχημα (πράκτορας) να αποκλίνει από την κατεύθυνση που ορίζει η κλίση της Συνάρτησης Πλοήγησης. Ένας νόμος ανάδρασης, βασισμένος στη ΣΠ, χρησιμοποιείται ως αναφορά, και σε αυτόν προστίθεται ένας όρος απόκλισης που προκύπτει από τη λύση ενός προβλήματος βελτίστου ελέγχου με πεπερασμένο ορίζοντα. Τα κριτήρια λειτουργικής απόδοσης κωδικοποιούνται σε ένα συναρτησιακό, ενώ η απαίτηση για φραγμένη κατώτερη ταχύτητα ικανοποιούνται από τον αρχικό νόμο ελέγχου για τη γραμμική ταχύτητα. Βασικό

[1] Λόγω της πληθώρας όρων της θεωρίας συστημάτων, αυτομάτου ελέγχου και ρομποτικής, που δυστυχώς δεν τυγχάνουν επιτυχημένης μετάφρασης στα Ελληνικά, αλλά εκ των πραγμάτων εμπεριέχονται στην παρούσα εργασία, το κυρίως μέρος αυτής είναι στην Αγγλική γλώσσα. Παρολαυτά, σε αυτή την ενότητα παρέχεται μια εκτεταμμένη περίληψη στα Ελληνικά.

πλεονέκτημα της μεθόδου είναι ότι, επιπλέον, διατηρούνται οι μαθηματικές εγγυήσεις που προσέφερε το αρχικό σχήμα ελέγχου, βασισμένο εξ΄ ολοκλήρου σε ΣΠ.

Στην περίπτωση του ενός οχήματος, το τεχνητό δυναμικό πεδίο είναι μια διπολική Συνάρτηση Πλοήγησης (Παράρτημα A.1.3), ενώ στο πολυπρακτορικό σύστημα χρησιμοποιούνται διπολικές Αποκεντρωμένες Συναρτήσεις Πλοήγησης (Παράρτημα A.1.4). Στο πολυπρακτορικό σενάριο, προτείνουμε και συγκεντρωμένο και αποκεντρωμένο σχήμα προβλεπτικής πλοήγησης. Η διαφορά έγκειται στην πληροφορία που έχει στη διάθεσή του ο πράκτορας όταν λύνει το πρόβλημα βελτίστου ελέγχου. Στην αποκεντρωμένη περίπτωση, ο κάθε πράκτορας δε λαμβάνει υπόψιν τις αποφάσεις των υπολοίπων (μιας μορφής αβεβαιότητα). Οπότε, προτείνουμε την εκτέλεση των υπολογισμών βάσει συμβάντων (Παράρτημα A.3), και όχι απλά ανά τακτά χρονικά διαστήματα. Ένα συμβάν λαμβάνει ξώρα όταν η διαφορά μεταξύ της προβλεπόμενης και της πραγματικής απόδοσης ενός πράκτορα υπερβαίνει κάποιο όριο.

Για να επαληθεύσουμε την αποτελεσματικότητα του προτεινόμενου σχήματος ελέγχου, δηλαδή τη βελτίωση στη λειτουργική απόδοση σε σξέση με τον αρχικό νόμο ελέγχου βασισμένο σε ΣΠ, προσομοιώσαμε ποικίλα σενάρια στο MATLAB. Από αυτά εξάγαμε συγκριτικά αποτελέσματα, τόσο για την περίπτωση του ενός πράκτορα, όσο και για την περίπτωση του πολυπρακτορικού συστήματος (αποκεντρωμένου και μη).

Ακολουθεί μια εκτενής περίληψη του κυρίου μέρους αυτής της εργασίας στα Ελληνικά.

Λέξεις Κλειδιά: Nonholonomic vehicles, Navigation Functions, Model Predictive Control (MPC), Predictive Navigation, Multi–agent system, Decentralized navigation, Event-triggered navigation, Air–Traffic Management

Βιβλιογραφική Ανασκόπηση

Μια κλάση μεθόθων που έξουν προταθεί για το πρόβλημα του προγραμματισμου πορείας είναι τα τεχνητά δυναμικά πεδία [3]. Συγκεκριμένα, οι Αποκεντρωμένες Συναρτήσεις Πλοήγησης (ΑΣΠ) [4] είναι μια πολυπρακτορική επέκταση των Συναρτήσεων Πλοήγησης [5]. Άλλες προσεγγίσεις για την πλοήγηση ή/και το συντονισμό πολυπρακτορικών συστημάτων χρησιμοποιούν Προβλεπτικό Έλεγχο (π.χ. [6, 7, 8, 9, 10]).

Οι δουλειές που σχετίζονται περισσότερο με την παρούσα εργασία είναι οι [7] και [2]. Στην [7], προτείνεται μια μέθοδος όπου ένα σχήμα Προβλεπτικού Ελέγχου παράγει ενδιάμεσους

στόχους για μια ΑΣΠ. Στην [2], οι συγγραφείς παρουσιάζουν ένα πλαίσιο για την πλοήγηση ενός ρομποτ χρησιμοποιώντας επίσης προβλέψεις. Το πρόβλημα του βελτίστου ελέγχου πεπερασμένου ορίζοντα αντιμετωπίζεται με τη χρήση Τυχαιοποιημένων Αλγορίθμων [11].

Μεθοδολογίες Προβλεπτικού Ελέγχου βασισμένων σε συμβάντα έχουν προταθεί στις εργασίες [12, 13, 14].

Πρόβλημα και Προσέγγιση

Μοντελοποίηση

Τα υπό μελέτη οχήματα (είτε ένα, είτε πολλαπλά) μοντελοποιούνται ως μονόκυκλα. Δηλαδή σαν κινηματικά αεροσκάφη που κινούνται σε σταθερό ύψος.

$$\dot{\mathbf{q}}_i = \begin{bmatrix} \dot{\mathbf{p}}_i \\ \dot{\phi}_i \end{bmatrix} = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\phi}_i \end{bmatrix} = \begin{bmatrix} v_i \cos\phi_i \\ v_i \sin\phi_i \\ \omega_i \end{bmatrix}, \tag{1}$$

όπου $\mathbf{p}_i$ η θέση και ϕ_i ο προσανατολισμός του οχήματος i. Το διάνυσμα ελέγχου είναι:

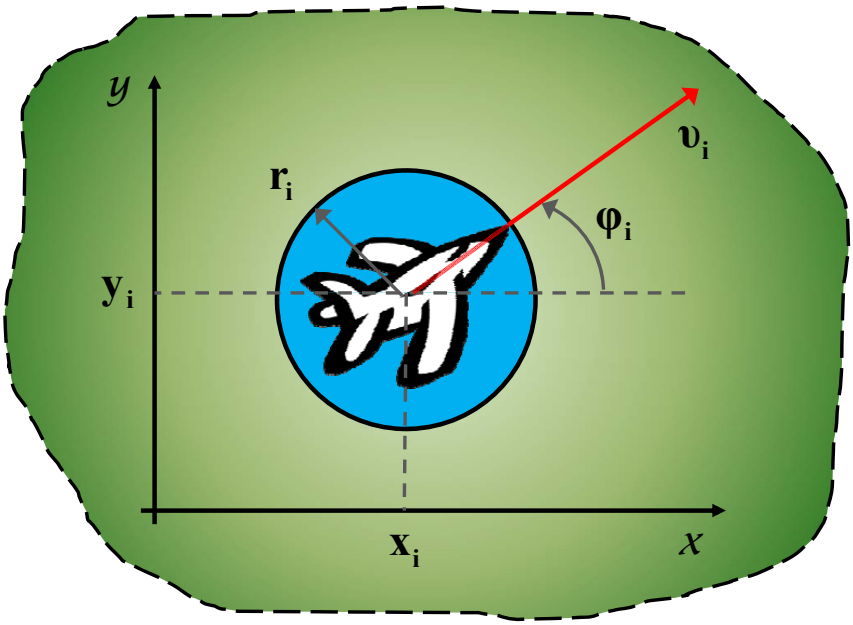
$$\mathbf{u}_i = [v_i \; \omega_i]^\top$$

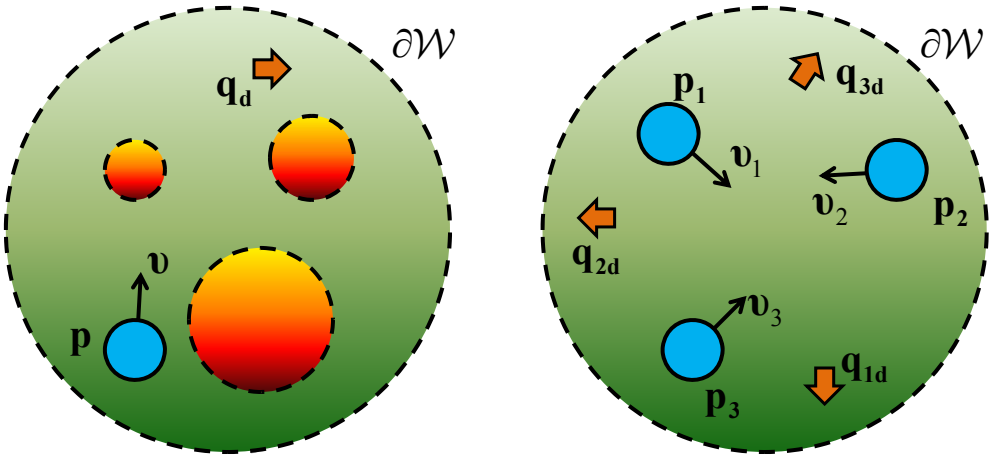
Τα οχήματα θεωρούνται μη-σημειακά. Συγκεκριμένα, κάθε ένα βρίσκεται στο εσωτερικό ενός δίσκου, δηλαδή της προστατευμένης περιοχής του, όπως φαίνεται στο σχήμα.

Η ονομαστική ταχύτητα πτήσης του κάθε οχήματος δίνεται από τη σχέση:

$$U_i = \begin{cases} U_{id}, & ||\mathbf{p}_i - \mathbf{p}_{id}|| > r_0 \\ \dfrac{||\mathbf{p}_i - \mathbf{p}_{id}||}{r_0} \cdot U_{id}, & ||\mathbf{p}_i - \mathbf{p}_{id}|| \leq r_0 \end{cases}, \tag{2}$$

όπου r_0 η ακτίνα ενός κύκλου γύρω από τον προορισμό $\mathbf{q}_{id}$ του εκάστοτε οχήματος.





Προσέγγιση (Προβλεπτική Πλοήγηση)

Θα χρησιμοποιήσουμε ως αναφορά τον παρακάτω νόμο για τον έλεγχο του ενός πρακτορα

$$v = -\text{sgn}(P) \cdot U, \tag{3α}$$

$$\omega = -k_\phi(\phi - \phi_{nh}) + \dot{\phi}_{nh}, \tag{3β}$$

όπου

$$\Phi(\mathbf{p}) = \frac{\gamma_d}{(\gamma_d^k + H_{nh} \cdot G \cdot \beta_0)^{1/k}}, \tag{4}$$

και τον παρακάτω για κάθε όχημα του πολυπρακτορικού συστηματος:

$$v_i = \begin{cases} -s_i \cdot U_i, & \dfrac{\partial \Phi_i}{\partial t} \leq U_i(|P_i| - \epsilon) \\ -s_i \cdot \dfrac{\dfrac{\partial \Phi_i}{\partial t} + \epsilon U_i}{|P_i|}, & \dfrac{\partial \Phi_i}{\partial t} > U_i(|P_i| - \epsilon) \end{cases} \tag{5α}$$

$$\omega_i = -k_\phi(\phi_i - \phi_{nhi}) + \dot{\phi}_{nhi}. \tag{5β}$$

όπου

$$\Phi_i(\mathbf{p}_i) = \frac{\gamma_{di} + f_i}{((\gamma_{di} + f_i)^k + H_{nhi} \cdot G_i \cdot \beta_{0i})^{1/k}}, \tag{6}$$

Τα κριτήρια λειτουργικής απόδοσης κωδικοποιούνται στο ολοκλήρωμα του συναρτησιακού

$$J(t, \mathbf{q}, \mathbf{u}, T) = \int_t^{t+T} \Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau))\mathrm{d}\tau + \Phi(\mathbf{q}(t+T)), \tag{7}$$

όπου T είναι ο χρονικός ορίζοντας της πρόβλεψης, ενώ t ο πραγματικός χρόνος.

Εισάγουμε μια απόκλιση θ από την κατεύθυνση που ορίζει η κλίση του πεδιου ϕ_{nh} και υπολογίζουμε τη βέλτιστη απόκλιση θ^* λύνοντας το παρακάτω πρόβλημα βελτίστου ελέγχου:

$$J(t, \mathbf{q}, \mathbf{u}, T) = \int_t^{t+T} \Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau))\mathrm{d}\tau + \Phi(t+T), \tag{8}$$

$$J^*(t, \mathbf{q}, T) = \min_{\theta[t,t+T)} J(t, \mathbf{q}, \mathbf{u}, T), \tag{9}$$

$$\theta^*[t, t+T) = \arg J^*(t, \mathbf{q}, T), \tag{10}$$

Οπότε τα νέα σχήματα προβλεπτικής πλοήγησης, για έναν και πολλαπλούς πράκτορες, δίνονται από τις παρακάτω σχέσεις:

- Ένα όχημα/πράκτορας:

$$v = -\mathrm{sgn}(P)U, \tag{11α}$$

$$\omega = -k_\phi(\phi - \phi_{nh} - \theta^*) + \dot{\phi}_{nh} + \dot{\theta}^*. \tag{11β}$$

- Πολυπρακτορικό σύστημα, $i \in \mathcal{N}$:

$$v_i = \begin{cases} -s_i \cdot U_i, & \frac{\partial \Phi_i}{\partial t} \leq U_i(|P_i| - \epsilon) \\ -s_i \cdot \dfrac{\dfrac{\partial \Phi_i}{\partial t} + \epsilon U_i}{|P_i|}, & \frac{\partial \Phi_i}{\partial t} > U_i(|P_i| - \epsilon) \end{cases} \tag{12α}$$

$$\omega_i = -k_\phi(\phi_i - \phi_{nhi} - \theta_i^*) + \dot{\phi}_{nhi} + \dot{\theta}_i^*. \tag{12β}$$

Το πρόβλημα βελτίστου ελέγχου πεπερασμένου ορίζοντα θα αντιμετωπιστεί με τη χρήση Τυχαιοποιημένων Αλγορίθμων. Συγκεκριμένα, μας ενδιαφέρει το αποτέλεσμα του παρακάτω λήμματος:

Λήμμα 1: Απαιτούμενος αριθμός δειγμάτων ([2],[11],[15]):

Ο αριθμός δειγμάτων N_s που εγγυάται ότι το $J^*(\cdot)$ είναι 'πιθανό κοντινό ελάχιστο' του $J(\cdot)$, σε επίπεδο α και εμπιστοσύνη $1-\delta$ είναι:

$$N_s \geq \frac{\ln(1/\delta)}{\ln(\frac{1}{1-\alpha})} \tag{13}$$

Σε κάθε στιγμή υπολογισμού t_k, κάθε/ο πράκτορας θα εκτελεί τον παρακάτω αλγόριθμο:

Αλγόριθμος: Βελτιστοποίηση Πεπερασμένου Ορίζοντα

Παράμετροι: α, δ, Θ, $\mathcal{P}(\theta)$, T, $J(t_k, \mathbf{q}, \mathbf{u}, T)$

1: Υπολόγισε τον επαρκή αριθμό δειγμάτων N_s από το Λήμμα 1.

2: Δημιούργησε N_s τυξαία δείγματα θ^m, $m = 1, \ldots, N_s$,
από ένα σύνολο $\Theta \subset (-\frac{\pi}{2}, +\frac{\pi}{2})$, σύμφωνα με την κατανομή $\mathcal{P}(\theta)$.

3: για $\mathrm{m} = 1 : \mathrm{N_s}$

4: Δημιούργησε μια υποψήφια απόκλιση $\theta^m[t_k, t_k + T)$ ως εξής:
$\theta^m[t_k, t_k + T) = (1 - \frac{\tau}{T})\theta^*(t_k) + (\frac{\tau}{T})\theta^m$, όπου $\tau \in [0, T)$.

5: Προσομοίωσε το σύστημα στο διάστημα $[t_k, t_k + T)$ χρησιμοποιώντας την $\theta^m[t_k, t_k + T)$.

6: Υπολόγισε το $J^m(t_k, \mathbf{q}, \mathbf{u}, T)$.

7: τερματισμός βρόγχου

8: Διάλεξε $\theta^*[t_k, t_k + T) = \arg\min_{\theta^m[t_k, t_k+T)} J^m(t_k, \mathbf{q}, \mathbf{u}, T)$.

Η συνολική απόκλιση για $t > 0$ δίνεται από την αλληλουχία

$$\theta^*(t) \triangleq \theta^*[0, t_1)|\ldots|\theta^*[t_k, t_{k+1})|\ldots|\theta^*[t_f, t_{k+1})|\ldots|\theta^*(t \geq t_f), \tag{14}$$

όπου t_f είναι η χρονική στιγμή που ένας πράκτορας εισήλθε στην περιοχή ακτίνας r_0 γύρω από τον προορισμό του.

Θεωρητικά Αποτελέσματα και Προσομοιώσεις

Μαθηματικές Εγγυήσεις

Τα παρακάτω μαθηματικά συμπεράσματα θεμελιώνουν την προτεινόμενη μέθοδο. Οι αποδείξεις τους υπάρξουν στο Παραρτημα A.5, στα Αγγλικά.

Λήμμα 2: Συνεχής και Φραγμένη Απόκλιση:

Η απόκλιση από την κλίση της ΣΠ, όπως υπολογίζεται από τον παραπάνω Αλγόριθμο, είναι μια συνεχής συνάρτηση του χρόνου (κλάσης C^0) και ικανοποιεί το όριο

$$|\theta^*(t)| < \frac{\pi}{2}$$

Θεώρημα 1: Σύνολο δειγματοληψίας:

Έστω ένα όχημα που κινείται με βάση το προτεινόμενο σχήμα ελέγχου, και ψ η γωνία μεταξύ της κλίσης της ΣΠ και του προσανατολισμού του οχήματος. Σε κάθε χρόνο εκτέλεσης t_k του Αλγορίθμου, το σύνολο $\Theta \subset (-\frac{\pi}{2}, +\frac{\pi}{2})$ θα δίνεται από τη σχέση

$$\Theta_k = (-\frac{\pi}{2} + |\psi_i(t_k) - \theta_i^*(t_k)|\,, +\frac{\pi}{2} - |\psi(t_k) - \theta^*(t_k)|).$$

Τότε εάν το $|\psi| = |\phi - \phi_{nh}|$ είναι αρχικά μικρότερο του $\frac{\pi}{2}$, θα παραμείνει για πάντα στο διάστημα $[0, \frac{\pi}{2})$.

Πόρισμα 1: Πεπερασμένη γραμμική ταχύτητα:

Εφ' όσων από το Θεώρημα 1 $|\phi - \phi_{nh}| \in [0, \frac{\pi}{2})$, η προβολή της κλίσης $\nabla\Phi$ στον άξονα του οχήματος, P, δεν είναι ποτέ μηδενική για $\mathbf{p} \neq \mathbf{p}_d$. Οπότε, η γραμμική ταχύτητα v στο νέο νόμο ελέγχου δεν πάει ποτέ στο άπειρο.

Πόρισμα 2: Κατεύθυνση κίνησης:

Το Θεώρημα 1 υπονοεί ότι αν ένα όχημα ξεκινήσει στον υποχώρο πίσω από τον προορισμό του, με το αρχικό διάνυσμα της κλίσης να τον οδηγει προς τα εμπρός ($P < 0$), τότε θα χρησιμοποιηθεί μόνο κίνηση προς τα εμπρός για πλοήγηση και αποφυγή συγκρούσεων. Αυτή η συνθήκη είναι απαραίτητη για εφαρμογή σε οχήματα τύπου αεροπλάνου.

Θεώρημα 2: Αποφυγή συγκρούσεων:

Ένα όχημα που κινείται με το προτεινόμενο σχήμα ελέγχου παραμένει πάντα ασφαλές, δηλαδή, δε συμβαίνουν ποτέ συγκρούσεις.

Θεώρημα 3: Σύγκλιση στον επιθυμητό προορισμό:

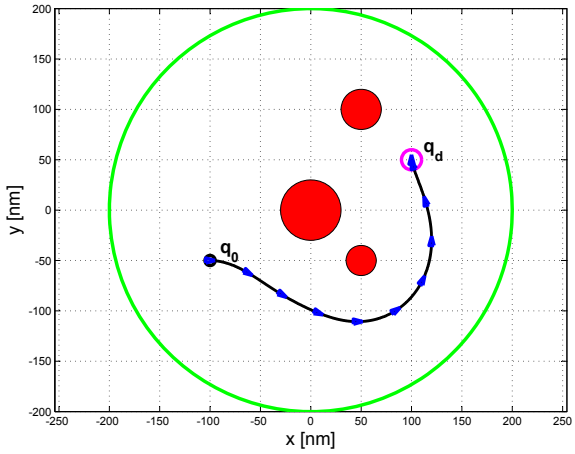
Το ένα όχημα, καθώς και το πολυπρακτορικό σύστημα, επιδέξονται τον ορισμό μιας συνεχούς συνάρτησης Luapunov , V. Επιπλέον, κάθε πράκτορας συγκλίνει στον προορισμό του $\mathbf{p}_d$ με τον επιθυμητό προσανατολισμό ϕ_d.

Αποτελέσματα Προσομοιώσεων

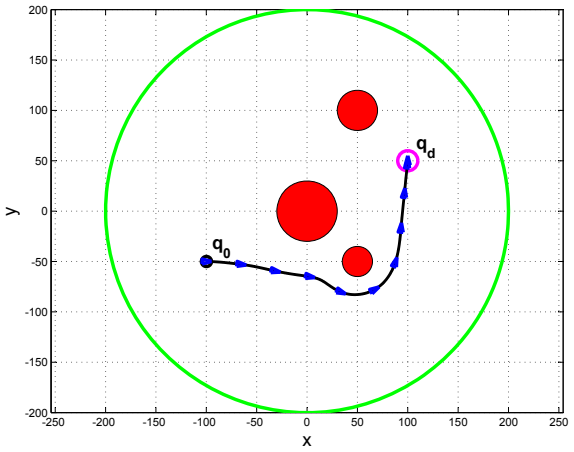
Για τις προσομοιώσεις χρησιμοποιήθηκαν στοιχεία από το αεροσκάφος Airbus A321 . Συγκεκριμένα, για πτήση σε υψόμετρο 33000 πόδια, η βέλτιστη ταχύτητα πτήσης είναι 454 κόμβοι [16]. Άρα θέτουμε $U = 454$ για κάθε όχημα. Η προστατευόμενη ζώνη κάθε οχήματος θα ειναι $r_i = 2.5$ ναυτικά μίλια για το πολυπρακτορικό σύστημα (οπότε $2 \cdot r_i = 5$), και $r = 5$

ναυτικά μίλια για το σενάριο με το ένα όχημα. Τέλος, $r_0 = 10$ ναυτικά μίλια σε όλα τα σενάρια.

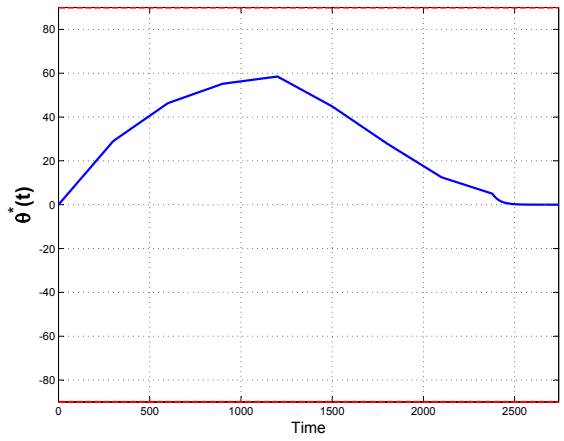
Προσομοίωση: Ένα όχημα

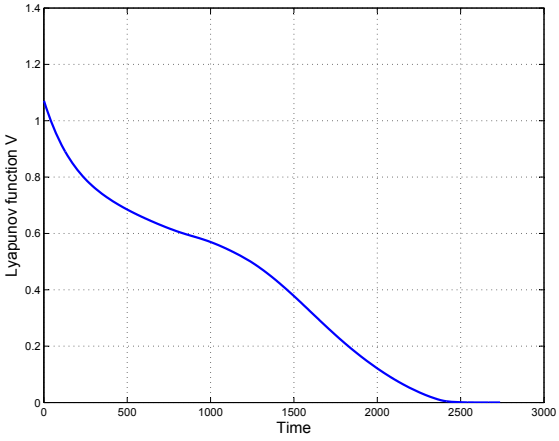


(α) Ελεγκτής με Συναρτήσεις Πλοήγησης μόνο.



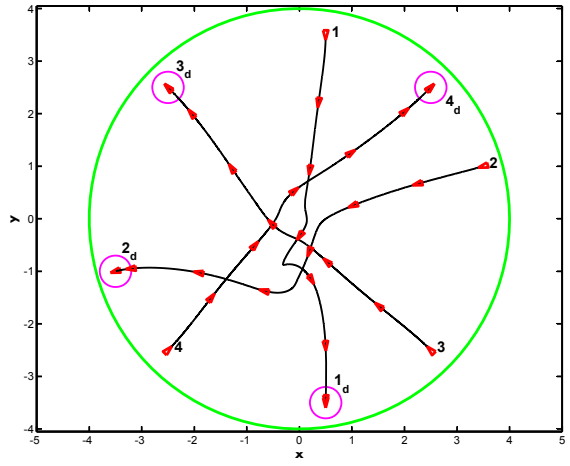
(β) Προβλεπτική πλοήγηση.



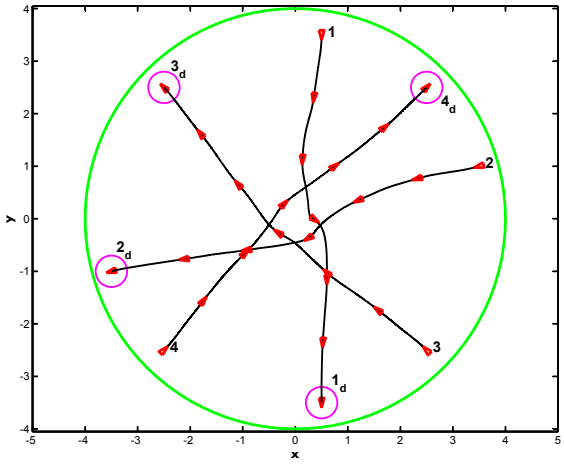


Παράμετροι σεναρίου: $T = 700,\ T_c = 300,\ N_s = 29,\ Q = \frac{1}{4 \cdot R_w^2},\ R = 500.$

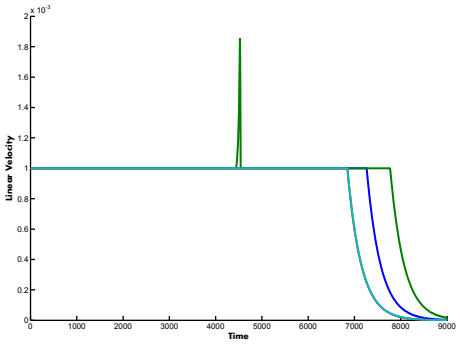
Προσομοίωση: Πολυπρακτορικό σύστημα



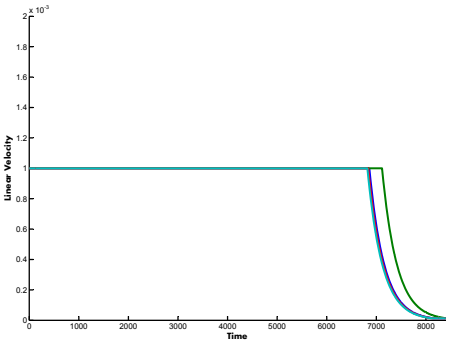
(ε) Ελεγκτής με Συναρτήσεις Πλοήγησης μόνο.



(φ) Προβλεπτική πλοήγηση.



(γ) Ελεγκτής με Συναρτήσεις Πλοήγησης.



(η) Προβλεπτική πλοήγηση.

Συγκριτικά Αποτελέσματα

Ποσοτικά αποτελέσματα που συγκρίνουν την αποτελεσματικότητα του προτεινόμενου σχήματος ελέγχου με τον αρχικό ελεγκτή παρέχονται στην Ενότητα 4.3 (Αγγλικά).

Contents

List of Figures

Chapter 1

Preface

1.1 Introduction

This work lies at the intersection of Robot Navigation, Multi-Agent Systems and Model-Predictive Control (MPC), as depicted in Fig. 1.1. The ultimate objective is the design of control schemes that can guarantee the convergence of one or more agents to their goal configurations (destination plus desired orientation) along collision-free trajectories, while taking into account task-specific performance criteria, e.g., minimum control effort.

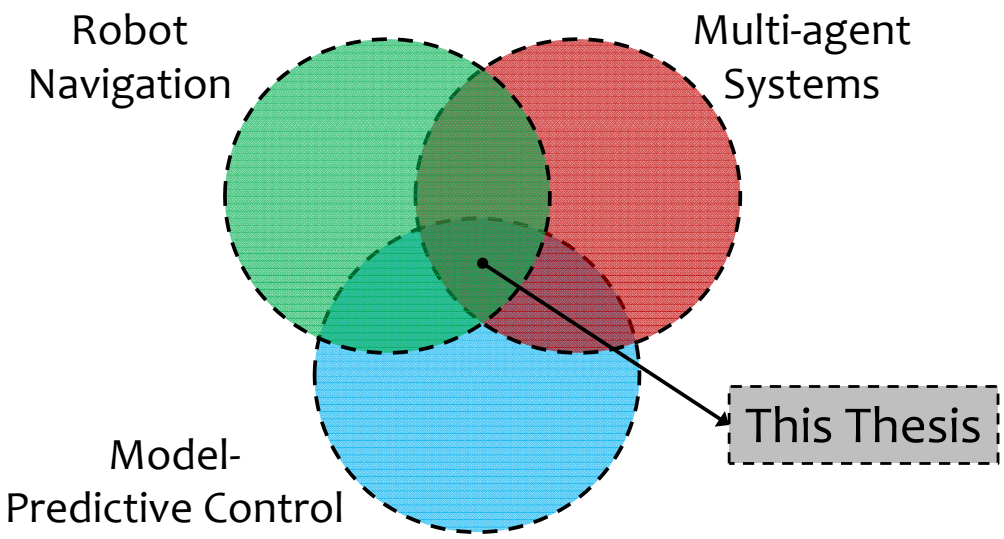


Figure 1.1: The approach proposed in this thesis employs tools from the topics of Robot Navigation, Multi-Agent Systems and Model-Predictive Control.

The motivation for this work came from the emerging topic of automated Air Traffic Control (ATC), which has attracted a lot of attention during the last years [19]. But, automated ATC is just a subset of multi-agent navigation problems, that also include

settings where multiple robotic vehicles (ground, underwater, aerial) operate in a shared workspace. However, automated ATC favors the design of control schemes that tackle the issue of performance, since aircrafts operate over time horizons long enough to allow for the solution of optimization problems on-line. MPC is one such scheme, as it generally involves computationallly costly optimization.

Robot navigation, in particular Path or Motion planning, is a fundamental problem in the field of robotics. In simple terms, it involves finding a safe path from the initial to the desired configuration. Safety means avoidance of collision with obstacles while using this path. We make a distintion with regard to (w.r.t.) the term *collision*[1]. In the single agent case, collision refers to static obstacles that lie in the agent's configuration space (state constraints). In particular, these constraints result from actual obstacles in the agent's workspace. One such obstacle is the boundary of the workspace. In the multi-agent case, collision refers both to the intersection of two or more agents' "protected zones" and to the intersection of an agent's protected zone with the workspace boundary. A multi-agent setting with static obstacles is not examined in this work.

By the term agent, we refer to a physical entity, e.g., a robotic agent, equipped with computing power, sensing capabilities, actuators and some form of control scheme. The kinematics and dynamics of an agent are of particular interest to Robot navigation as each level of complexity in modelling requires increasingly advanced control schemes. In this work, we focus are attention to a particular type of agent, a kinematic aircraft-like agent. As will be discussed in detail later, an aircraft-like agent is an oriented two dimensional disk of radius equal to the protected zone of the aircraft it represents. Its kinematics are those of the unicycle, i.e., it is underactuated. This means that the agent has three states, its position in a coordinate reference frame, and its orientation relative to this frame, but only two control inputs, linear velocity and angular velocity. This underactuation introduces a type of constraint on the agent's motion called *nonholonomic*. Nonholonomic constraints will be discussed in detail in a later section. Apart from an aircraft, the unicycle model is also an abstraction for vehicles with similar kinematics, like a differential drive robot or a car. However, for aircrafts, a lower bound on its speed is also required. Vehicle dynamics are not examined in this work. In addition, there is no uncertainty, stochastic or otherwise, due to external disturbances or model-system mismatch. However, it will become evident that in a decentralized setting, the lack of

[1]In ATC terminology, the terms *conflict* and *loss of separation* are used to describe what we refer to as collision in this work. However, we will adopt the more general term collision, which is used in the robot navigation literature. Likewise, collision avoidance substitutes the ATC term *conflict resolution*.

information w.r.t. the decisions of other agents, is a form of uncertainty w.r.t. each particular agent.

Finally, the term *performance* does not refer to an algorithm's computational complexity but rather to physical quantities which depend on the particular task. Examples of such *task-specific performance criteria* include, but are not limited to, minimum control effort criteria, such as minimum fuel consumption, minimum deviation from a nominal speed, e.g., an aircraft's cruising speed, or nominal path, and minimum arrival time. These criteria are stated as mathematical expressions and encoded in a functional (the performance measure). This *cost* functional is one of the building blocks of a Model-Predictive Control scheme. It should be noted that, in this work, hard constraints, such as collision avoidance and a lower bound on an aircraft's speed, are part of the navigation problem, not performance criteria to be optimized.

All of the terms introduced in the introductory passage are explicitly defined in Chapter 2, along with some additional terms and assumptions required in order to mathematically describe the problem statement.

1.2 Literature Review

A class of methods used to solve path planning problems is artificial potential fields [3]. In particular, Decentralized Navigation Functions (DNFs) [4] are a multi-agent extension of Rimon's and Koditschek's Navigation Functions (NFs) methodology [5]. Dipolar DNFs offer guaranteed collision avoidance and convergence to target destination with the desired orientation. Other approaches to the multi-agent navigation or coordination problem employ Model Predictive (MPC) or Receding Horizon Control (RHC) Examples include [6, 7, 8, 9, 10].

A rough illustration of the literature related to our work is provided in Fig. 1.2. Most relevant to this work are [7] and [2]. In [7], a distributed approach where MPC is used to generate way-points for DNFs is proposed. At each iteration, each agent solves an optimization problem and broadcasts its solution to others. The scheme is distributed since, at each recalculation time, each agent solves an optimization problem and broadcasts its solution to others. No theoretical guarantees were given on the convergence of the overall scheme. In [2], the authors present a framework for the navigation of a *single* robot that combines RHC and control Lyapunov functions. Control inputs are parametrized using a perturbation on the direction of a potential field's gradient. The Finite Horizon

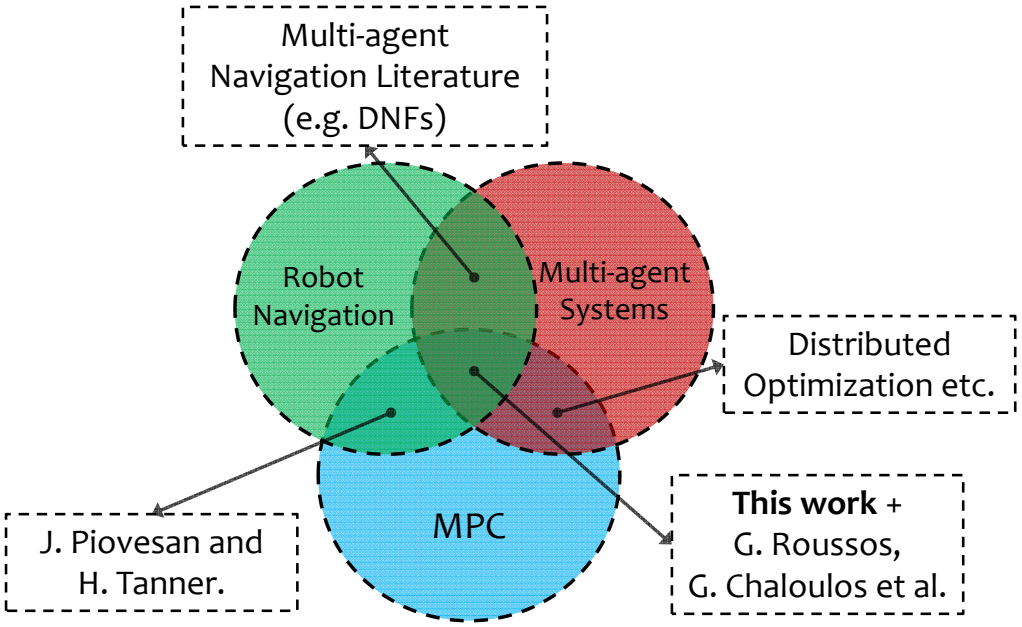


Figure 1.2: Rough illustration of the related literature.

Optimal Control Problem (FHOCP) is solved using randomized algorithms [11, 15] and Lyapunov-like stability conditions are given in [20].

Event-triggered (E-T) nonlinear MPC appears in [13, 14]. In addition, event-triggered strategies for model-predictive controllers have been proposed in [12]. In the above, events are used to trigger a recalculation of a FHOCP when the system's evolution differs from the predicted one, due to *external* disturbances.

1.3 Contributions

The contributions of this work can be summarized as follows:

- Extension of the work in [2] for a single *nonholonomic* vehicle using dipolar NFs.
- Extension of the above scheme to *multiple* vehicles using dipolar DNFs.
- Improvements in order to make the scheme applicable to ATC scenarios.
- *Decentralization* of the new scheme using an event-triggered strategy.
- We offer hard, *mathematical guarantees* on the properties of the proposed schemes.

However, the main contribution of this work is the Decentralized Event-Triggered Predictive Navigation scheme. Our results have already been published in [1]. Our paper [1] and this thesis extend ideas from [2] to a decentralized multi-agent navigation setting by employing Dipolar DNFs [4]. The aircraft-like vehicles considered here are modeled as unicycles. The fact that agents navigating by tracking an artificial potential field's gradient cannot choose one trajectory over another motivates us to introduce a deviation from the direction of the DNF's gradient. This deviation is calculated by solving a FHOCP, using the *randomized algorithms* –based control design in [11] and [15]. The sampling set is adjusted at each iteration such that the navigation properties of the original scheme are preserved. Performance requirements are encoded in the cost functional. A DNF-based controller guarantees that agents' velocity is lower bounded. The overall scheme is decentralized in the sense of [21], i.e., agents' predictive controllers do not communicate during navigation. The only additional information required, compared to the original scheme using just DNFs, is the broadcast of each agent's target configuration to other agents at the beginning of the collision avoidance manoeuvre. This information exchange is minimal compared to the broadcast of state and/or input trajectories. Decentralization introduces uncertainty wrt each agent. This motivates us to use event-triggered recalculations of the FHOCP. An execution rule based on the discrepancy between the predicted and actual performance of an agent generates the events. The inter-event times are lower bounded in terms of the maximum values of the involved variables. Each agent derives its own event times independently, in a decentralized and asynchronous manner.

1.4 Diploma Thesis Structure

The rest of this thesis is organized as follows. Chapter 2 casts the problem statement in a mathematical setting, presents the tools that will be employed in our approach, and also outlines our approach. In Chapter 3, the Predictive Navigation scheme for a single aircraft-like vehicle is presented along with simulation results. Chapter 4 includes the main contributions of this work. In §4.1, we present the Centralized Predictive Navigation scheme for multiple aircraft-like vehicles, whereas the Decentralized Event-Triggered scheme is presented in §4.2. Simulation results are provided in both sections, but §4.3 presents additional *comparative*[2] simulations. Chapter 5 discusses our results, summarizes our conclusions, and offers future research directions.

[2]The original DNF–based controller is compared to the proposed centralized and decentralized predictive navigation schemes in terms of a performance measure (§2.1.2).

Finally, two appendices are provided. Appendix A provides details on mathematical and control theoretic tools that are mentioned and/or employed throughout this work. The mathematical proofs of various lemmas, corollaries and theorems, in particular, are gathered in Appendix A.5. Appendix B elaborates on the MATLAB code developed for simulations of our proposed scheme.

Chapter 2

Problem Statement & Approach of Solution

2.1 Technical Problem Statement

Regarding the accuracy of the model used to describe the motion of aircraft-like agents, we are interested in the simplest abstraction. In particular, we want an agent model that can capture three basic features of an aircraft:

- The lateral degree of freedom (d.o.f.) is not actuated;
- A volume of airspace surrounding an aircraft (protected zone) should not be infringed upon by any another aircraft;
- Aircrafts cannot use zero-speed manoeuvres due to aerodynamic limitations such as stalling.

These criteria are taken into account in §2.1.1, where an adequate agent model is presented. Briefly, the first criterion suggests the use of a nonholonomic model [22], the second leads to modelling aircrafts as non-point agents and the third to lower bounding agents' speed.

Regarding the control objectives, we set three requirements:

- Convergence to the destination with the desired orientation;
- Safety, i.e., trajectories are collision-free;
- Improved performance wrt a performance measure.

The individual parts of our problem statement are introduced in the following sections.

2.1.1 Agent Modelling

Aircraft-like vehicles (agents) navigating in a 2-dimensional workspace $\mathcal{W} \subset \mathbb{R}^2$ are considered. In order to model the kinematics of an aircraft flying at constant altitude, each agent is described by the unicycle model:

$$\dot{\mathbf{q}}_i = \begin{bmatrix} \dot{\mathbf{p}}_i \\ \dot{\phi}_i \end{bmatrix} = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\phi}_i \end{bmatrix} = \begin{bmatrix} v_i \cos\phi_i \\ v_i \sin\phi_i \\ \omega_i \end{bmatrix}, \tag{2.1}$$

where $i \in \mathcal{N} = \{1, \ldots, N\}$ and $\mathbf{I}_i \triangleq [\cos\phi_i \ \ \sin\phi_i]^\top$ is a Jacobian matrix. The index i will be omitted in the single agent scenario, whereas in the multi-agent setting $N \in \mathbb{N}$ is the total number of agents. Also $\mathbf{p}_i = [x_i \ y_i]^\top$ is the position vector of agent i wrt an earth fixed frame and $\phi_i \in (-\pi, \pi]$ its heading angle, i.e., the angle between agent i's longitudinal axis and the earth-fixed x-axis, as depicted in Figure 2.1. The configuration of each agent is then $\mathbf{q}_i \in Q = \mathcal{W} \times (-\pi, \pi]$. The control vector consists of the linear and the angular velocities as follows:

$$\mathbf{u}_i = [v_i \ \omega_i]^\top$$

Each aircraft-like agent is subject to one nonholonomic constraint [22] due to underactuation in the lateral d.o.f. In Pfaffian form, this constraint is expressed as:

$$[-\sin\phi_i \ \ \cos\phi_i \ 0] \ \dot{\mathbf{q}}_i = 0 \tag{2.2}$$

Agents are considered non-point. Each one occupies a disk (its protected zone), of radius r_i, centered at $\mathbf{p}_i \in \mathcal{W}$ (see Figure 2.1). The agents' workspace is assumed to be a 2D disk of radius R_w, centered at the origin of the earth-fixed frame. Its boundary ∂W is considered an obstacle. Furthermore, a destination $\mathbf{p}_{id}$ and a desired orientation ϕ_{id}, at $\mathbf{p}_{id}$, is assigned to each agent. Therefore, the desired configuration of agent i is $\mathbf{q}_{id} = \left[\mathbf{p}_{id}^\top \ \phi_{id}\right]^\top$.

Definition 2.1 (Collision). *In the single agent case, the term* collision *will refer to the intersection of an agent's protected zone with the (static) obstacles that lie in $\mathcal{W}$ or with the workspace boundary ∂W. In the multi-agent case, the term* collision *will refer to the intersection of two or more agents' protected zones or an agent's protected zone with the workspace boundary ∂W.*

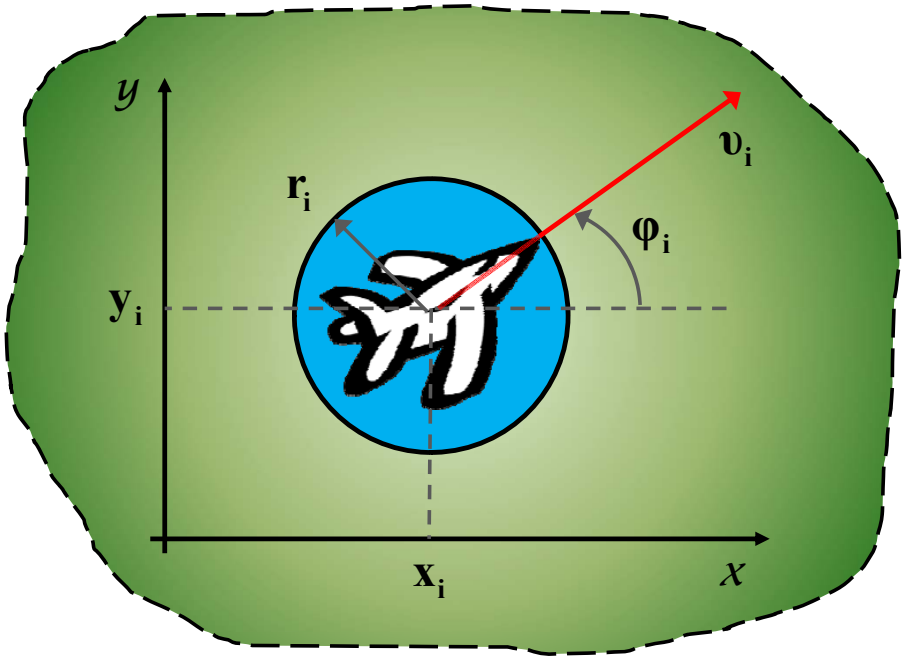


Figure 2.1: Aircraft-like agent i and its protected zone (blue) of radius r_i. The aircraft sketch is not in scale wrt the protected zone.

Figure 2.2 illustrates the two different scenarios; single agent and multi-agent.

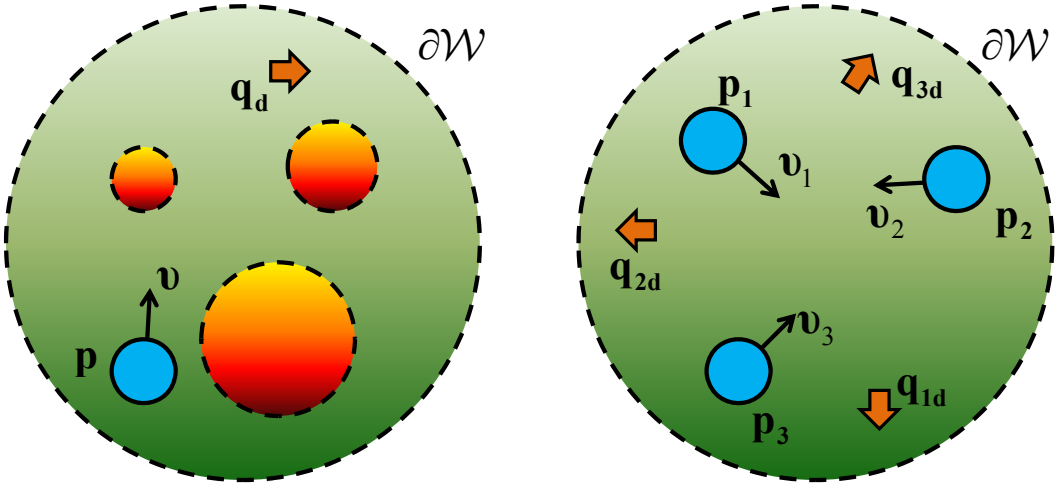


Figure 2.2: Single agent (left) and multi-agent (right) scenarios. Blue is for agents, red is for obstacles, and the orange arrows represent the goal configuration of each agent.

Finally, since we want to approximately model the motion of an aircraft, agents' speed should be lower bounded. This is in contrast to a car-like vehicle which is allowed to stop (zero speed) during a collision avoidance manoeuvre. Therefore, a nominal speed U_i is assigned to each aircraft-like agent, as follows:

$$U_i = \begin{cases} U_{id}, & ||\mathbf{p}_i - \mathbf{p}_{id}|| > r_0 \\ \dfrac{||\mathbf{p}_i - \mathbf{p}_{id}||}{r_0} \cdot U_{id}, & ||\mathbf{p}_i - \mathbf{p}_{id}|| \leq r_0 \end{cases}, \tag{2.3}$$

where r_0 is the radius of a circular region around each agent's destination and U_{id} is the desired absolute speed — constant or varying independently of the control scheme —, that corresponds to an aircraft's cruising speed. The switch in (2.3) is continuous; once inside the region $||\mathbf{p}_i - \mathbf{p}_{id}|| \leq r_0$, an agent's speed is linearly reduced as a function of the *distance* from the destination. A plot of equation (2.3) appears in Figure 2.3.

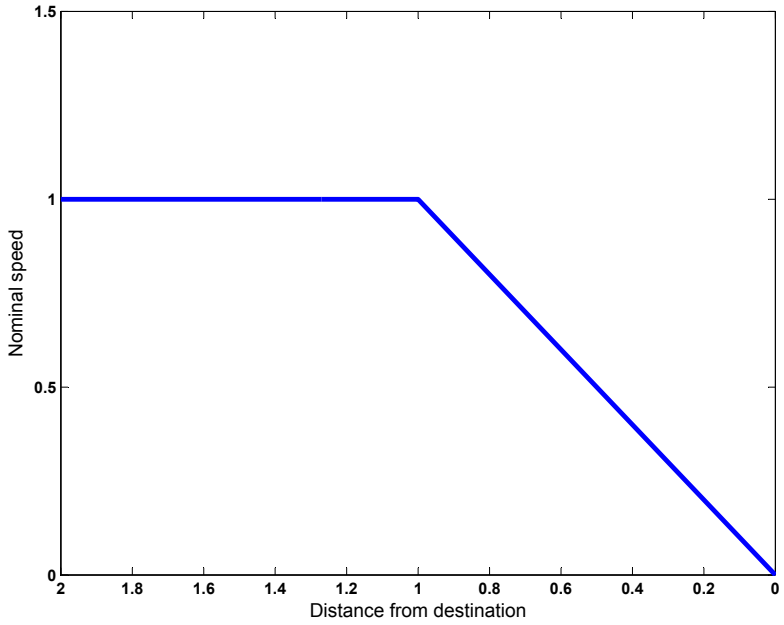


Figure 2.3: Nominal speed U_i as a function of the distance from the destination $||\mathbf{p}_i - \mathbf{p}_{id}||$, as defined in the continuous switch (2.3). In this example, $r_0 = 1$ and $U_i = 1$.

Model used in Simulations

In the numerical examples and simulation results that we will present in Chapters 3 and 4, we will use realistic figures. Specifically, we will assume that all aircraft are of type

`Airbus A321` (depicted in Fig. 2.4), flying at an altitude of 33000 ft, a typical cruising altitude for commercial flights [7]. The airspeed at this altitude must belong to the range $[336, 540]$ knots ($\frac{nm}{h}$), with a nominal value of 454 knots [16]. Thus, we set

$$U_{id} = 454 \text{ knots }, \forall i \in \mathcal{N}.$$

Figure 2.4: Picture of an Airbus A321.

Regarding the aircrafts' protected regions, we will use $r = 5$ nm (nautical miles) for the single agent case. This corresponds to a minimum distance of 5 nm between the vehicle and the boundary of an obstacle (undesired region of the workspace-airspace). In the multi-agent setting, we will use $r_i = 2.5$ nm, $\forall i \in \mathcal{N}$. Again, this corresponds to a minimum distance (separation) of $2r_i = 5$ nm between any two vehicles. Finally, the radius r_0 of the neighbourhood around each agent's destination $\mathbf{p}_{id}$ is set to $2.5r_i$, for all scenarios.

2.1.2 Performance Measure

We will now mathematically define the term *performance*. High-level requirements, such as minimum fuel consumption, can be represented by a performance measure of the following form:

$$\int_{t_0}^{t_f} \Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau)) d\tau, \tag{2.4}$$

where $[t_0, t_f]$ is the time interval over which the performance of the system is quantified, $\Lambda(\cdot, \cdot)$ is a convex, positive definite function of the state, $\mathbf{q}$, and the control input, $\mathbf{u}$, and

$\tau \in [t_0, t_f]$ is the integration variable. The exact structure of $\Lambda(\cdot, \cdot)$, which we will call the *running cost* function, may be straightforwardly dictated by the task requirements or the choice may be less obvious. In the latter case, it is up to the control designer to formulate the running cost function such that is corresponds to the actual task requirements. Some examples are provided below:

- A running cost function that quantifies a trajectory tracking specification would be of the form $\Lambda(\mathbf{q}(\tau)) = (\mathbf{q}(\tau) - \mathbf{r}(\tau))^\top \mathbf{Q}(\mathbf{q}(\tau) - \mathbf{r}(\tau))$, where $\mathbf{r}(t)$ is the desired trajectory and $\mathbf{Q}$ is a square, symmetric, weighting matrix of appropriate dimensions.
- A running cost function that quantifies a minimum control effort specification would be of the form $\Lambda(\mathbf{u}(\tau)) = \mathbf{u}(\tau)^\top \mathbf{R}\mathbf{u}(\tau)$, where, likewise, $\mathbf{R}$ is a square, symmetric, weighting matrix of appropriate dimensions.
- The individual performance requirements can be combined in one running cost function by summation, e.g.,

$$\Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau)) = (\mathbf{q}(\tau) - \mathbf{r}(\tau))^\top \mathbf{Q}(\mathbf{q}(\tau) - \mathbf{r}(\tau)) + \mathbf{u}(\tau)^\top \mathbf{R}\mathbf{u}(\tau).$$

The functions in the examples provided above are quadratic and zero-mean attractive cost functions. These functions penalize, i.e., assign higher values to, deviation from the zero valued inputs, $(\mathbf{q}(\tau) - \mathbf{r}(\tau)) = 0$ and $\mathbf{u}(\tau) = 0$ respectively. A visualization of such (quadratic) functions is given in Fig. 2.5 (right). It is not necessary for the cost functions to be quadratic. However, quadratic functions have some welcomed properties. If $\mathbf{Q}$ and $\mathbf{R}$ are chosen to be positive definite, as is the case in practice, then $\Lambda(\cdot, \cdot)$ is strictly convex and there exists a unique global minimum [23]. Other cost functions that may be used, depending on the task-specific performance requirements, include one-sided attractive cost function, point-wise repulsive cost functions (see Fig. 2.6) and linearly repulsive cost functions.

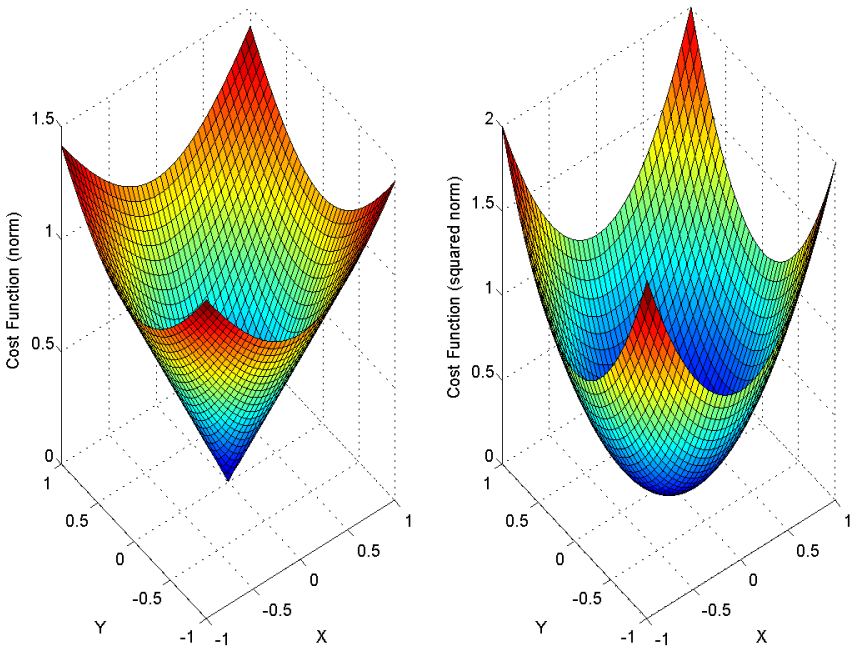


Figure 2.5: Zero-mean attractive cost functions; they penalize deviation from the zero valued inputs. Euclidean norm (left) and squared Euclidean norm (right).

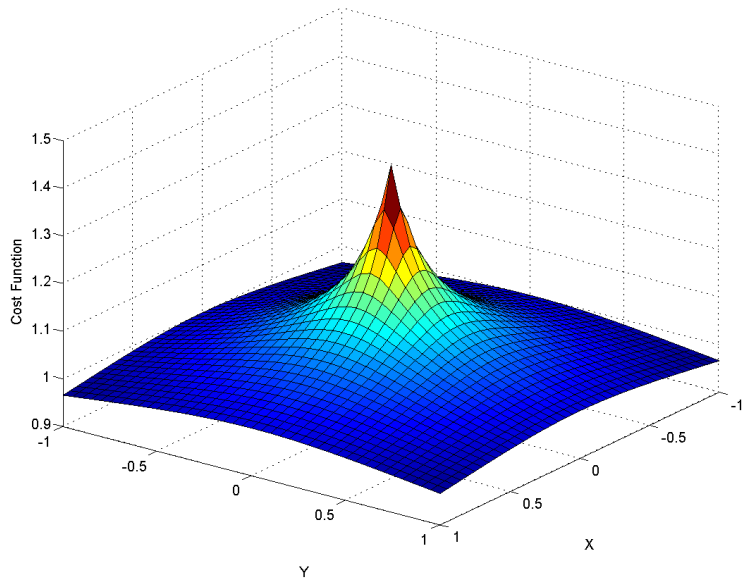


Figure 2.6: A point-wise repulsive cost function.

The integral in equation 2.4 is a special case of *functional*. The mathematical term "functional" refers to a map from a space of functions into its underlying scalar field. In our case, Eq. 2.4 assigns a positive scalar value to each pair of state and input trajectories $\mathbf{q}(\tau)$, $\mathbf{u}(\tau)$, where $\tau \in [t_0, t_f]$. In the remainder of this thesis, we will refer to Eq. 2.4 as the performance measure, or running cost, and to $\Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau))$ as the running cost function. The term "cost functional" will be introduced later.

2.1.3 Problem Statements

The three control objectives mentioned in §2.1 (convergence, safety, performance) are integrated into the following problem statements. We describe both the original formulations (convergence and safety only), and the ones including the performance requirement. In the multi-agent case, we also make a distinction between centralized and decentralized performance. In the former, there is one performance measure for the entire multi-agent system. The corresponding running cost function is $\Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau))$, where $\mathbf{q} = [\mathbf{q}_1^\top \ \mathbf{q}_2^\top \ \dots \mathbf{q}_N^\top]^\top$ and $\mathbf{u} = [\mathbf{u}_1^\top \ \mathbf{u}_2^\top \ \dots \mathbf{u}_N^\top]^\top$. In the case of decentralized performance, each agent tries to minimize its own performance measure $\int_{t_0}^{t_f} \Lambda_i(\mathbf{q}_i(\tau), \mathbf{u}_i(\tau))d\tau$.

Single agent

Problem 2.1 (**Single agent Navigation (original)**). *Design a control law for an agent, described by the kinematic model (2.1), that will steer the agent to its target destination $\mathbf{p}_d$ with the desired orientation ϕ_d, while avoiding collisions with static obstacles and the workspace boundary ∂W.*

Problem 2.2 (**Single agent Navigation (performance)**). *Design a control law for an agent, described by the kinematic model (2.1), such that the resulting state and input trajectories, $\mathbf{q}(\tau)$ and $\mathbf{u}(\tau)$ respectively, minimize a performance measure $\int_{t_0}^{t_f} \Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau))d\tau$ and are also a solution to Problem 2.1.*

Multi-agent

Problem 2.3 (**Multi-agent Navigation (original)**). *Design a control law for each agent $i \in \mathcal{N}$, described by the kinematic model (2.1), that will steer the agent to its target destination $\mathbf{p}_{id}$ with the desired orientation ϕ_{id}, while avoiding collisions, as defined in Definition 2.1.*

Problem 2.4 (**Multi-agent Navigation (centralized performance)**). *Design a control law for each agent $i \in \mathcal{N}$, described by the kinematic model (2.1), such that the resulting state and input trajectories of the multi-agent system,* $\mathbf{q}$ *and* $\mathbf{u}$ *respectively, minimize a performance measure* $\int_{t_0}^{t_f} \Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau))d\tau$*, and is also a solution to Problem 2.3.*

Problem 2.5 (**Multi-agent Navigation (decentralized performance)**). *Design a control law for each agent $i \in \mathcal{N}$, described by the kinematic model (2.1), such that the resulting state and input trajectories of each agent,* $\mathbf{q}_i$ *and* $\mathbf{u}_i$ *respectively, minimize each performance measure* $\int_{t_0}^{t_f} \Lambda_i(\mathbf{q}_i(\tau), \mathbf{u}_i(\tau))d\tau$*, and is also a solution to Problem 2.3.*

It will become apparent that, in practice, the minimization requirement in Problems 2.2, 2.4 and 2.5 will have to be relaxed. That is, the resulting solutions will still take into account the corresponding performance measure, but they will be suboptimal.

2.2 Approach of Solution

Briefly, the approach we take to tackling the problems in §2.1.3 employs three methodologies; Navigation Functions (NFs), Nonlinear Model Predictive Control (NMPC) and Randomized Algorithms. Specifically, NFs and NMPC are combined to derive novel control laws and Randomized Algorithms are used to tackle the online Finite Horizon Optimal Control Problem (FHOCP). Once again, in the single agent case, our contribution is only a minor extension of [2]; nonholonomic kinematics instead of single integrator. In the decentralized multi-agent setting, we also employ concepts from Event-Triggered (E-T) control. The reasons will become apparent as we present the decentralized methodology.

2.2.1 Navigation Functions–based Schemes

In this section we provide a brief overview of navigation schemes based on the Navigation Functions (NF) methodology. In particular, we start with the control law for a single, nonholonomic agent based on Dipolar Navigation Functions (see Appendix A.1.3). Then we present the control laws which this work will expand on. That is, control laws based on Decentralized dipolar Navigation Functions (see Appendix A.1.4) for the navigation of multiple nonholonomic agents. A introduction to the Navigation Functions methodology, from Khatib's artificial potential fields to Decentralized Navigation Functions (DNF), is provided in Appendix A.1.

Single nonholonomic agent

Given an agent, described by the kinematic model (2.1), a spherical workspace $\mathcal{W}$, static obstacles in $\mathcal{W}$, and a desired configuration $\mathbf{q}_d \in \mathcal{W} \times (-\pi, \pi]$ (see Fig. 2.2–left), one can define a dipolar Navigation Function, based on [24] and [25], as follows:

$$\Phi(\mathbf{p}) = \frac{\gamma_d}{(\gamma_d^k + H_{nh} \cdot G \cdot \beta_0)^{1/k}}, \tag{2.5}$$

where $\gamma_d(\mathbf{p}) = ||\mathbf{p} - \mathbf{p}_d||^2$ is the distance from the destination, and the obstacle function G is defined in Appendix A.1.2. Furthermore, the term β_{0i} refers to the workspace bounding obstacle, while H_{nh} corresponds to the artificial obstacle used to render the potential field dipolar (see Appendix A.1.3 for figures and mathematical details). Finally, $k > 0$ is a tuning parameter for this class of NFs.

Navigation for the nonholonomic agent is based on the projection of the dipolar NF's gradient $\nabla\Phi = [\Phi_x \ \Phi_y]^\top$ on the agent's heading direction (longitudinal axis):

$$P = [\cos\phi \ \ \sin\phi] \cdot \nabla\Phi. \tag{2.6}$$

The projection is depicted in Figure 2.7. The sign of P, $s = \text{sgn}(P)$, determines the direction of motion (forward/reverse). The sign function is defined as follows:

$$\text{sgn}(x) \triangleq \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0. \end{cases} \tag{2.7}$$

Nonholonomic system cannot be stabilized by continuous, time-invariant, state feedback control laws [26]. In [24], the authors propose a discontinuous feedback scheme. Control laws $\mathbf{u} = [v \ \omega]^\top$ are of the generic form:

$$v = -k_1 \text{sgn}(P) \cdot ||\nabla\Phi||\,, \tag{2.8a}$$

$$\omega = -k_2(\phi - \phi_{nh}) + \dot{\phi}_{nh}, \tag{2.8b}$$

where the *nonholonomic heading angle* ϕ_{nh}, and the projection of the current position vector with respect to the destination, on the direction of the desired orientation, d, are

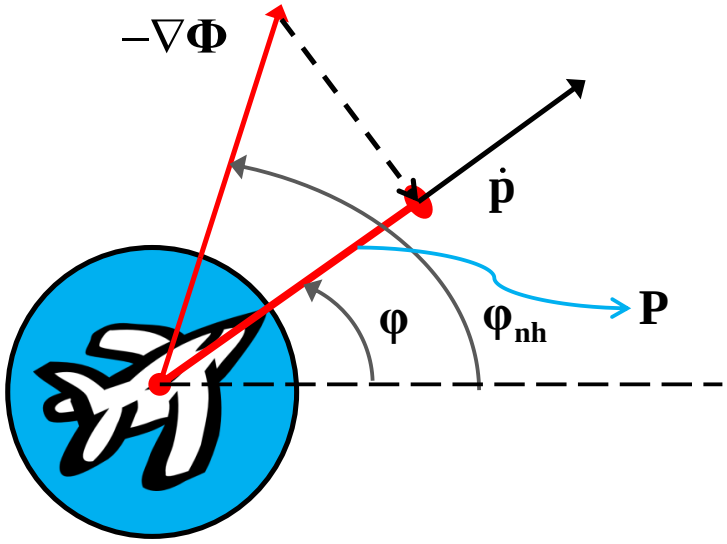


Figure 2.7: Projection of dipolar NF's gradient $\nabla\Phi$ on the agent's heading direction.

given as follows:

$$\phi_{nh} \triangleq \text{atan2}(\text{sgn}(d)\Phi_y, \text{sgn}(d)\Phi_x), \tag{2.9}$$

$$d = [\cos\phi_d \;\; \sin\phi_d] \cdot (\mathbf{p} - \mathbf{p}_d). \tag{2.10}$$

In practice, the time derivative $\dot{\phi}_{nh}$ is calculated numerically at each time step. The two-argument function atan2$(\cdot,\cdot)$ is defined as:

$$\text{atan2}(y, x) \triangleq \arg(x, y), \quad (x, y) \in \mathbb{C}.$$

Therefore, sgn(d) is equal to 1 in front of the target configuration, and -1 behind the target configuration. Finally, k_1 and k_2 are positive real gains. To ensure the continuity of ϕ_{nh} on the destination, where the potential field's gradient vanishes, $\nabla\Phi = 0$, we use the following approximation scheme [27]:

$$\hat{\phi}_{nh} \triangleq \begin{cases} \phi_{nh}, & \rho > \epsilon \\ \dfrac{\phi_{nh}\left(-2\rho^3 + 3\epsilon\rho^2\right) + \phi_d\left(-2\left(\epsilon - \rho\right)^3 + 3\epsilon\left(\epsilon - \rho\right)^2\right)}{\epsilon^3}, & \rho \leq \epsilon \end{cases} \tag{2.11}$$

where $\rho = \sqrt{\Phi_x^2 + \Phi_y^2}$ and ϵ is a small positive constant. Thus, $\hat{\phi}_{nh}$ is continuous when

$\rho = 0$, where $\mathbf{p} = \mathbf{p}_d$:

$$\hat{\phi}_{nh}(\mathbf{p}_d) = \lim_{\mathbf{p}\to\mathbf{p}_d} \hat{\phi}_{nh} = \lim_{\rho\to 0} \hat{\phi}_{nh} = \hat{\phi}_{nh}\Big|_{\rho=0} = \phi_d.$$

Control laws of the form (2.8) track the direction of a dipolar NF's gradient. Using the norm of $\nabla\Phi$ to scale the linear velocity v is not mandatory. In an ATC-like scenario, aircraft-like agents use their nominal speed U (see Eq. (2.3)), rather than $||\nabla\Phi||$. That is, $v = -\text{sgn}(P)U$. The interested reader is also referred to [28] for an analysis of control laws derived from gradient flows, including the discontinuous type, similar to (2.8a).

Multiple nonholonomic agents

Similarly, in a multi-agent setting (see Fig. 2.2–right), a dipolar Decentralized Navigation Function (DNF) is defined *for each* aircraft-like agent $i \in \mathcal{N}$, as it appears in [29]:

$$\Phi_i(\mathbf{p}_i, \mathbf{p}') = \frac{\gamma_{di} + f_i}{\left((\gamma_{di} + f_i)^k + H_{nhi} \cdot G_i \cdot \beta_{0i}\right)^{1/k}}, \tag{2.12}$$

where $\mathbf{p}'$ refers to the current position of agent i and the current positions of all other agents (since they appear in the "obstacle" function G_i):

$$\mathbf{p}' = [\mathbf{p}_1^\top \ \dots \ \mathbf{p}_{i-1}^\top \ \mathbf{p}_{i+1}^\top \ \dots \ \mathbf{p}_N^\top]^\top.$$

Note that for the construction of Φ_i, no knowledge of agents' $j \in \mathcal{N}$, $j \neq i$, desired configurations (or even destinations) $\mathbf{q}_{jq}$ is required by agent i. The dependence of Φ_i on $\mathbf{p}'$, and even $\mathbf{p}_i$, will be dropped for the sake of notational brevity. Additional details on the Multi-agent Navigation Functions methodology, including the construction of G_i and f_i, are provided in Appendix A.1.4, and in the following references: [18, 29, 30, 31].

Collision avoidance and convergence depend on a decreasing rate for potential Φ_i, whose time derivative can be expanded to two terms:

$$\dot{\Phi}_i = \frac{\partial \Phi_i}{\partial t} + \nabla_i \Phi_i \cdot \dot{\mathbf{p}}_i = \sum_{j\neq i} \nabla_j \Phi_i^\top \cdot \mathbf{I}_j v_j + P_i v_i. \tag{2.13}$$

The second term is due to the motion of agent i, while the partial derivative $\dfrac{\partial \Phi_i}{\partial t}$ sums the effect of all, but the i^{th}, agents' motion on Φ_i; $j \in \mathcal{N}$, $j \neq i$. We remind the reader

that $\mathbf{I}_i \triangleq [\cos\phi_i \ \ \sin\phi_i]^\top$, and that $\nabla\Phi_i = [\Phi_{ix} \ \Phi_{iy}]^\top$.

A DNF-based control law (similar to the one in [17]), that solves the multi-agent navigation and collision avoidance Problem 2.3, and is suitable for aircraft-like vehicles appears below. The linear and angular velocities, v_i and ω_i respectively, are given by:

$$v_i = \begin{cases} -s_i \cdot U_i, & \dfrac{\partial \Phi_i}{\partial t} \leq U_i(|P_i| - \epsilon) \\ -s_i \cdot \dfrac{\dfrac{\partial \Phi_i}{\partial t} + \epsilon U_i}{|P_i|}, & \dfrac{\partial \Phi_i}{\partial t} > U_i(|P_i| - \epsilon) \end{cases} \tag{2.14a}$$

$$\omega_i = -k_\phi(\phi_i - \phi_{nhi}) + \dot{\phi}_{nhi}, \tag{2.14b}$$

where the nominal speed U_i is given by 2.3, $s_i = \text{sgn}(P_i)$, k_ϕ is a positive real gain, and ϵ is a small positive constant. Similar to the single agent case, the *nonholonomic heading angle* is defined as:

$$\phi_{nhi} \triangleq \text{atan2}(\text{sgn}(d_i)\Phi_{iy}, \text{sgn}(d_i)\Phi_{ix}),$$
$$d_i = [\cos\phi_{id} \ \ \sin\phi_{id}] \cdot (\mathbf{p}_i - \mathbf{p}_{id}).$$

In the linear velocity control law (2.14a), the nominal speed U_i is applied as long as it can guarantee the decrease of Φ_i, while the angular velocity control law (2.14b) is responsible for tracking the direction of the dipolar DNF's gradient, $\nabla\Phi_i$, by reducing the tracking error, $(\phi_i - \phi_{nhi})$, between the agent's orientation and the nonholonomic heading angle. In Eq. (2.13), note that each agent is assumed to measure the position, orientation and linear velocity of all other agents, but has no knowledge of their desired configurations.

2.2.2 Model-Predictive Navigation for Nonholonomic Agents

The inability of control laws (2.14a)–(2.14b) to handle performance criteria motivates the use of Model-Predictive Control (MPC) methodologies. The term *Model-Predictive Navigation* in particular was introduced by Piovesan and Tanner in [2]. Their approach, along with the extensions developed in our work, is described below.

We introduce a performance measure (an extension of (2.4)) as follows:

$$J(t, \mathbf{q}, \mathbf{u}, T) = \int_t^{t+T} \Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau))\mathrm{d}\tau + M(\mathbf{q}(t+T)), \tag{2.15}$$

where $\Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau))$ is the running cost function (see §2.1.2) that encodes the desired performance criteria, and $M(\mathbf{q}(t+T))$ is called the *terminal cost* and will be discussed later. MPC is based on minimizing the functional (2.15), using iterative finite horizon optimization, at each sampling time $\mathbf{q}(t)$, resulting in a feedback[1] scheme. The finite horizon is quantified by the *prediction horizon* $T > 0$. Then, the minimizing control policy

$$\mathbf{u}^*[t, t+T) = \arg\min_{\mathbf{u}} J(t, \mathbf{q}, \mathbf{u}, T), \tag{2.16}$$

derived during the prediction phase $[t, t+T)$, is applied to the actual system over the time interval $[t, t+T_c)$, where $0 < T_c < T$ is the (typically fixed) *control horizon*. Thus, if t is the current sampling instant, the next one would be $t + T_c$. In our predictive navigation framework, a Navigation Function will be used as the terminal cost in (2.15), i.e., $M(\mathbf{q}(t+T)) \triangleq \Phi(\mathbf{q}(t+T))$. This is because the value of a NF at $t+T$ can serve as a heuristic or approximation of the cost-to-go from $t+T \to \infty$. It will become apparent that no special requirements need to apply for a NF to be used as a terminal cost. For example, we do not require the dipolar NF to be a *Control Lyapunov function* (which it is not).

The rest of this section will specifically refer to Model-Predictive Navigation, not MPC in general. A brief, general overview of Optimal and Model-Predictive Control is provided in Appendix A.2. Furthermore, in this section we will discuss the proposed approach in general, and the specifics of single agent and multi-agent navigation problems, with performance criteria, will be presented in detail in Chapters 3 and 4, respectively.

Agents using a feedback control law based solely on the negated gradient of a Navigation Function, cannot choose one trajectory over another while navigating. Their trajectory only depends on initial–final conditions and NF's parameters which are set at priori and, in addition, are not intuitive for tuning. To this end, we introduce a deviation $\theta(t)$ from the direction of the dipolar NF's (DNF's in the multi-agent case) gradient, ϕ_{nh}. This deviation is translated into a control input, for use during the prediction phase, as follows:

$$\bar{\mathbf{u}}(\tau) \triangleq \begin{bmatrix} 0 \\ \bar{\omega} \end{bmatrix} = \begin{bmatrix} 0 \\ k_\phi \theta + \dot{\theta} \end{bmatrix}, \quad \tau \in [t, t+T) \tag{2.17}$$

where k_ϕ is once again a positive gain. To simplify the notation, an existing NF–based feedback control law will be denoted as "$-k(\mathbf{q}, \nabla\Phi)$". The control input $\bar{\mathbf{u}}$ is incorporated

[1] Each optimization is performed in an open-loop fashion, but the resulting scheme is a feedback control scheme since the system state is sampled, and the control policy/input is recalculated, at each time instant. However, it is not a *closed-loop* feedback control scheme.

into the original framework as follows:

$$\mathbf{u} = \mathbf{u}(\mathbf{q}, \nabla\Phi, \theta) \triangleq -k(\mathbf{q}, \nabla\Phi) + \bar{\mathbf{u}}(\theta). \tag{2.18}$$

We are now ready to establish the generic Model-Predictive Navigation framework for nonholonomic vehicles. Consider the following Finite Horizon Optimal Control Problem (FHOCP):

$$J(t, \mathbf{q}, \mathbf{u}, T) = \int_t^{t+T} \Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau))\mathrm{d}\tau + \Phi(t+T), \tag{2.19a}$$

$$J^*(t, \mathbf{q}, T) = \min_{\theta[t,t+T)} J(t, \mathbf{q}, \mathbf{u}, T), \tag{2.19b}$$

$$\theta^*[t, t+T) = \arg J^*(t, \mathbf{q}, T), \tag{2.19c}$$

where the term $\Phi(t+T)$ is used to simplify the notation of $\Phi(\mathbf{q}(t+T))$, since this differs depending on the scenario (single or multi-agent). At each sampling instant, $\{\dots, t_k, t_{k+1}, t_{k+2}, \dots\} = \{\dots, t_k, t_k + T_c, t_k + 2T_c, \dots\}$, the FHOCP (2.19) is solved for $\theta^*[t, t+T)$ and the resulting control input

$$\mathbf{u}^* = -k(\mathbf{q}, \nabla\Phi) + \bar{\mathbf{u}}(\theta^*)$$

is applied over the control horizon $[t_k, t_k + T_c)$. By letting "$-k(\mathbf{q}, \nabla\Phi)$" be the control laws for v and ω presented in the previous section (§2.2.1), we get for each case:

- Single agent setting

$$v = -\mathrm{sgn}(P)U, \tag{2.20a}$$

$$\omega = -k_\phi(\phi - \phi_{nh} - \theta^*) + \dot{\phi}_{nh} + \dot{\theta}^*. \tag{2.20b}$$

- Multi-agent setting, $i \in \mathcal{N}$

$$v_i = \begin{cases} -s_i \cdot U_i, & \dfrac{\partial \Phi_i}{\partial t} \leq U_i(|P_i| - \epsilon) \\ -s_i \cdot \dfrac{\dfrac{\partial \Phi_i}{\partial t} + \epsilon U_i}{|P_i|}, & \dfrac{\partial \Phi_i}{\partial t} > U_i(|P_i| - \epsilon) \end{cases} \tag{2.21a}$$

$$\omega_i = -k_\phi(\phi_i - \phi_{nhi} - \theta_i^*) + \dot{\phi}_{nhi} + \dot{\theta}_i^*. \tag{2.21b}$$

All parameters involved have been defined in the previous section. The specifics of the

calculation of θ^*, $\boldsymbol{\theta}^*$, and θ_i^*, in the single, centralized multi-agent, and decentralized multi-agent settings will be described in the corresponding chapters. Figure 2.8 illustrates the deviation θ from the direction of the dipolar NF's gradient.

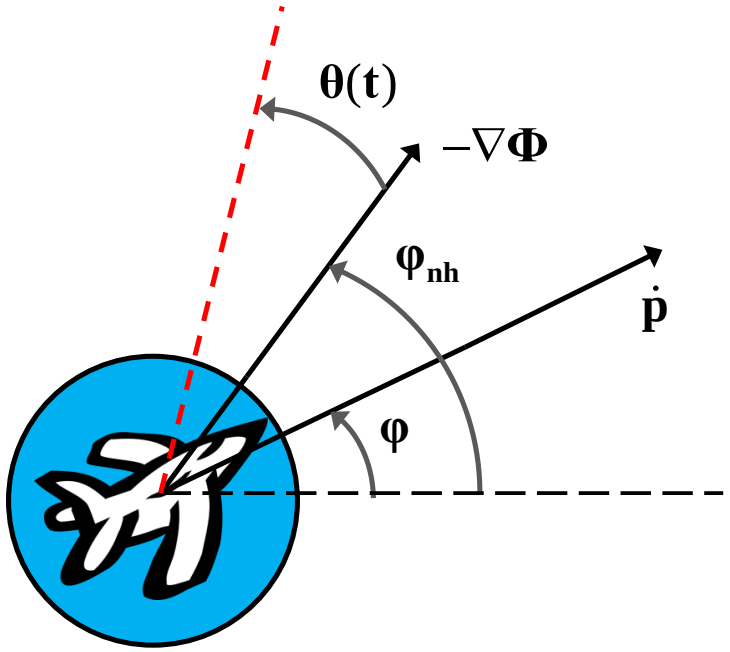


Figure 2.8: Nonholonomic heading angle ϕ_{nh} and deviation θ.

The following section presents the theoretic background of the optimization tool that will be employed to tackle the FHOCP (2.19), i.e., *Randomized Algorithms*. Their application to our Predictive Navigation setting with be presented for each setting individually.

Randomized Algorithms

The optimization implied in (2.19) is computationally complex and requires the use of specialized optimization algorithms. However by relaxing the need for optimality, a class of randomized algorithms used in distribution-free statistical learning methods can be used to solve the FHOCP. The basic idea is to generate a sufficient number of candidate solutions by sampling a set (according to some probability distribution $\mathcal{P}$), simulate the system's dynamics using each candidate, and select the one that performs best [2].

Using the notation is [2], we state the following definition and lemma from [11] and [15]. Let $\sigma \in \Sigma$ be a decision vector and $R : \Sigma \to \mathbb{R}$ a cost functional. One is interested in

the decision parameter $\sigma^\star$ that yields the best performance $R^\star = \inf \sigma \in \Sigma R(\sigma)$. Take N_s independent and identically distributed (i.i.d.) random samples σ_i, $i = 1 \dots N_s$ from Σ according to a probability distribution $P(\sigma)$. Then $R^* = \min_i R(\sigma_i)$ can approximate $R^\star$ [15].

Definition 2.2 (**Probable near minimum** [11]). *Given $R(\sigma)$, $\delta \in (0,1)$, and $\alpha \in (0,1)$, a number $R^* \in \mathbb{R}$ is said to be a probable near minimum of $R(\sigma)$ to level α and confidence $1-\delta$, if there exists a set $\tilde{\Sigma} \subseteq \Sigma$ measuring $\mathbb{P}\{\tilde{\Sigma}\} \leq \alpha$, such that*

$$\mathbb{P}\{\inf_{\Sigma} R(\sigma) \leq R^* \leq \inf_{\Sigma \setminus \tilde{\Sigma}} R(\sigma)\} \geq 1 - \delta$$

.

Lemma 2.1 (**Number of samples** [2],[11],[15]). *The number of samples N_s that guarantees R^* is a "probable near minimum" of $R(\sigma)$ to level α and confidence $1-\delta$ satisfies:*

$$N_s \geq \frac{\ln(1/\delta)}{\ln(\frac{1}{1-\alpha})} \tag{2.22}$$

[This page intentionally left blank.]

Chapter 3

Single-agent Predictive Navigation

In this Chapter, we present a brief extension of the Model–Predictive Navigation scheme in [2] to the case of nonholonomic (aircraft-like) agents. The analysis will be rather superficial, since the main contribution of this work involves the multi-agent setting. Therefore, additional discussion and mathematical details will be provided in Chapter 4.

3.1 Predictive Navigation for a Nonholonomic Agent

Consider an aircraft-like agent, described by the kinematic model (2.1), navigating in a workspace (airspace) $\mathcal{W}$. Undesired regions of the workspace will be modelled as obstacles (see Figure 3.1). The performance criteria are encoded in a performance measure (cost functional) as stated in Eq. (3.1).

$$J(t, \mathbf{q}, \mathbf{u}, T) = \int_t^{t+T} \Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau)) \mathrm{d}\tau + \Phi(\mathbf{p}(t+T)), \tag{3.1}$$

where the running cost $\Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau))$ is a function of the agent's state and controls, that encodes the performance criteria, and a dipolar Navigation Function (see Appendix A.1.3) Φ is used as the terminal cost (see §2.2.2).

We are now ready to restate Problem 2.2, i.e., the predictive navigation problem for a nonholonomic agent:

Problem 3.1 (Centralized Predictive Navigation). *Given a dipolar DNF (2.12), a running cost function $\Lambda(\mathbf{q}, \mathbf{u})$, and a prediction horizon T, derive, for each of the prediction intervals $[t_k, t_k + T)$, the control strategies $\mathbf{u}^*[t_k, t_k + T)$ that minimize (3.1) for each interval, in such a way that their concatenation, applied over $t \in [0, \infty)$, is also a solution to the original navigation Problem 2.1.*

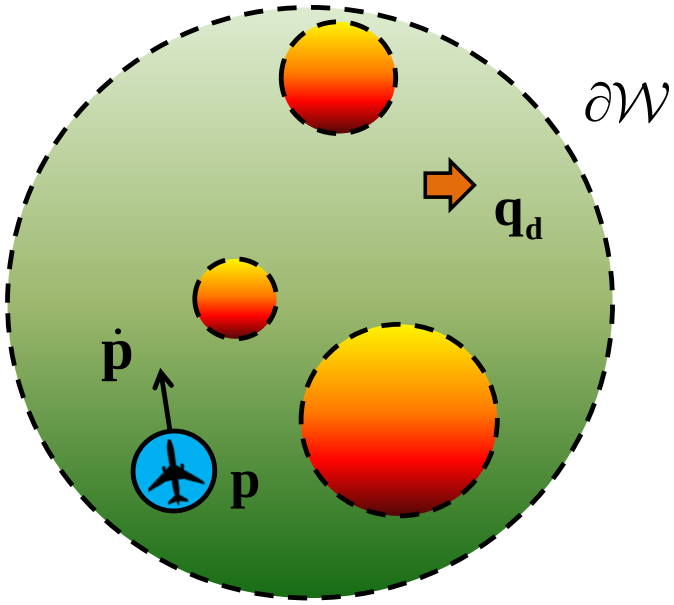


Figure 3.1: An aircraft-like vehicle (blue disk) navigating a workspace with obstacles, i.e. undesired regions (red disks). Its goal configuration $\mathbf{q}_d$ is depicted with an orange arrow.

The optimal control problem to be solved for each prediction interval $[t_k, t_k + T)$ is:

$$J(t_k, \mathbf{q}, \mathbf{u}, T) = \int_{t_k}^{t_k+T} \Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau))\mathrm{d}\tau + \Phi(\mathbf{p}(t_k + T)), \tag{3.2a}$$

$$J^*(t_k, \mathbf{q}, T) = \min_{\theta^*[t_k, t_k+T)} J(t_k, \mathbf{q}, \mathbf{u}, T), \tag{3.2b}$$

$$\theta^*[t_k, t_k + T) = \arg J^*(t_k, \mathbf{q}, T), \tag{3.2c}$$

where θ^* is the optimum deviation from the direction of the dipolar NF's negated gradient. Then the resulting deviation $\theta^*[t_k, t_k + T)$ is applied to the agent, according to (2.20), over the control phase $[t_k, t_{k+1}) = [t_k, t_k + Tc)$, and the FHOCP (3.2) is solved again for the new system state $\mathbf{q}(t_{k+1})$.

Since deviations $\theta^*[t_k, t_k + T)$ will be calculated in a finite horizon manner, denote as $\theta^*(t)$ the *concatenation* of deviations over $t \in [0, \infty)$ as follows:

$$\theta^*(t) \triangleq \theta^*[0, t_1)|\theta^*[t_1, t_2)|\ldots|\theta^*[t_k, t_{k+1})\ldots, \tag{3.3}$$

obtained iteratively by solving the FHOCP (3.2) at each recalculation time t_k, and applying the resulting deviation $\theta^*[t_k, t_k + T)$ only over the time interval $[t_k, t_{k+1})$, where

$t_{k+1} = t_k + T_c$. We set $t_0 = 0$ and $\theta^*(0) \triangleq 0$.

Randomized Finite Horizon Optimization (Single–agent)

The optimization in the FHOCP (3.2) will be tackled using Randomized Algorithms. We restate Lemma 2.1 in the current setting:

Lemma 3.1 (**Number of samples (Single–agent)**). *The number of samples N_s that guarantees $J^*(t_k, \mathbf{q}, T)$ is a "probable near minimum" of $J(t_k, \mathbf{q}, \mathbf{u}, T)$, to level α and confidence $1 - \delta$, satisfies:*

$$N_s \geq \frac{\ln(1/\delta)}{\ln(\frac{1}{1-\alpha})}$$

At each recalculation time t_k, the centralized authority calculates a near optimum deviation vector $\theta^*[t_k, t_k + T)$ by executing Algorithm 1.

Algorithm 1: Single–agent Finite Horizon Optimization

`Parameters`: α, δ, Θ, $\mathcal{P}(\theta)$, T, $J(t_k, \mathbf{q}, \mathbf{u}, T)$

1: `Calculate` a sufficient number of samples N_s using Lemma 3.1.

2: `Generate` N_s i.i.d. random samples θ^m, $m = 1, \ldots, N_s$,
from a set $\Theta \subset (-\frac{\pi}{2}, +\frac{\pi}{2})$, according to the probability distribution $\mathcal{P}(\theta)$.

3: `for` m = 1 : N_s

4: `Generate` a candidate deviation $\theta^m[t_k, t_k + T)$ as:
$\theta^m[t_k, t_k + T) = (1 - \frac{\tau}{T})\theta^*(t_k) + (\frac{\tau}{T})\theta^m$, where $\tau \in [0, T)$ is a dummy time variable.

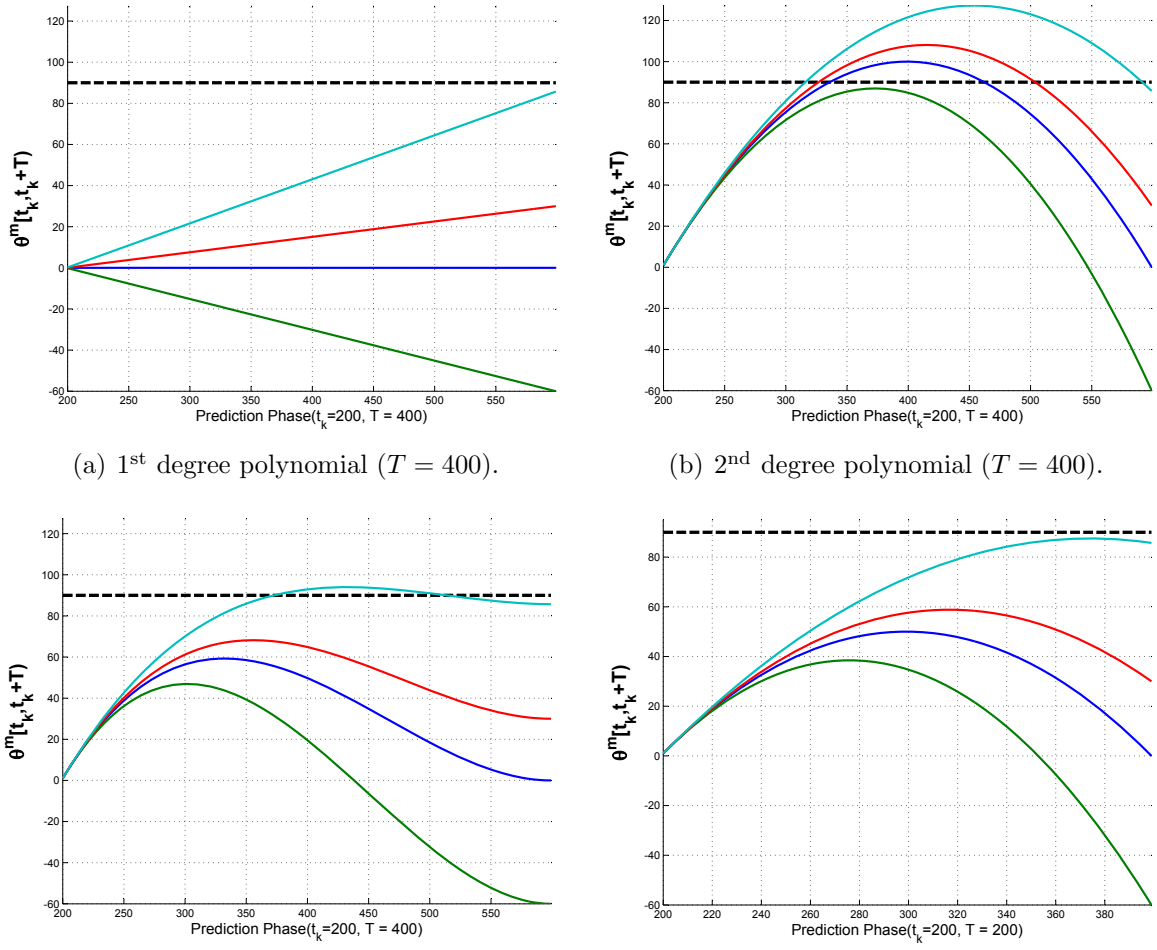
5: `Simulate` the agent's dynamics over $[t_k, t_k + T)$ using $\theta^m[t_k, t_k + T)$.

6: `Calculate` $J^m(t_k, \mathbf{q}, \mathbf{u}, T)$.

7: `end loop`

8: `Pick` $\theta^*[t_k, t_k + T) = \arg\min_{\theta^m[t_k, t_k+T)} J^m(t_k, \mathbf{q}, \mathbf{u}, T)$.

Each candidate deviation (Algorithm 1, Step 4) is a line segment (1st degree polynomial) connecting the point $(t_k, \theta^*(t_k))$ with one of the N_s sampled points $(t_k + T, \theta^m(t_k + T))$ of Step 2. Higher degree polynomials would result in "smoother" deviations but the condition $|\theta^*(t)| < \frac{\pi}{2}$ (proven in Lemma 3.2) would not hold for *any* value of the involved parameters (see Figure 3.2).



(a) 1st degree polynomial ($T = 400$).

(b) 2nd degree polynomial ($T = 400$).

(c) 3rd degree polynomial ($T = 400$).

(d) 2nd degree polynomial ($T = 200$).

Figure 3.2: Generation of candidate deviations $\theta^m[t_k, t_k + T)$ — Step 4, Algorithm 1 — using polynomials of increasing degree. Note that, despite the fact that samples $\theta^m \in \Theta \subset (-\frac{\pi}{2}, +\frac{\pi}{2})$, we *cannot* guarantee that $\theta^m[t_k, t_k + T) \in \Theta \subset (-\frac{\pi}{2}, +\frac{\pi}{2})$ for polynomials higher than 1st degree. This is influenced by the following parameters: $\theta(t_k), \dot{\theta}(t_k), T, \theta^m$.

Finally denote by $t_f = \inf\{t : ||\mathbf{p} - \mathbf{p}_d|| \leq r_0\}$ the time instant when the agent enters a neighbourhood r_0 of its destination $\mathbf{p}_d$. For $t \geq t_f$, θ will be given as a function of the distance $S = ||\mathbf{p} - \mathbf{p}_d||$:

$$\theta(t \geq t_f) = \theta(S) = S^2 \cdot \frac{\theta^*(t_f)}{{r_0}^2}.$$

We could say that the predictive controller is "turned-off" for $t \geq t_f$. This deviation term has the following mathematical properties:

i. $\theta(S = r_0) = \theta^*(t_f)$,

ii. $\theta(S = 0) = 0$,

iii. $\dot{\theta}(S = 0) = \dfrac{d\theta}{dS} \cdot \dfrac{dS}{d\mathbf{p}} \cdot \dot{\mathbf{p}}|_{S=0} = 0$.

Therefore $\theta^*(t)$ is restated as in (3.4)

$$\theta^*(t) \triangleq \theta^*[0, t_1)| \ldots |\theta^*[t_k, t_{k+1})| \ldots |\theta^*[t_f, t_{k+1})| \ldots |\theta^*(t \geq t_f). \tag{3.4}$$

Lemma 3.2 (Continuity and Boundedness of deviations). *The deviation (3.4), where $\theta^*[t_k, t_k + T)$ is calculated by Algorithm 1 and applied over the control phase $[t_k, t_{k+1})$, is a continuous function of time (class C^0) that satisfies the bound:*

$$|\theta^*(t)| < \frac{\pi}{2}$$

Proof. The proof of Lemma 3.2 can be found in Appendix A.5, Proof A.1.

Sampling Set Θ

The non-holonomic nature of the unicycle and the control law (2.20b) make it impossible to accurately track a desired heading angle. A tracking error, $(\phi - \phi_{nh} - \theta^*)$, will be present at all times. This means that the result of Lemma 3.2, i.e., $|\theta^*(t)| < \frac{\pi}{2}$, does not guarantee that ϕ also satisfies $|\phi - \phi_{nh}| < \frac{\pi}{2}$. Satisfaction of this condition is necessary to preserve the navigation properties of the original DNF–based control law (2.14b). At each recalculation time t_k, the sampling set $\Theta \subset (-\frac{\pi}{2}, +\frac{\pi}{2})$ has to be adjusted accordingly in order to take into account the tracking error.

Theorem 3.1 (Sampling Set adjustment (Single agent)). *Consider an agent described by Eq. (2.1) under the control law (2.21b) and denote by ψ the angle between the*

field's gradient and the agent's longitudinal axis. At each recalculation time t_k, let the sampling set $\Theta \subset (-\frac{\pi}{2}, +\frac{\pi}{2})$ in Algorithm 1 be

$$\Theta_k = (-\frac{\pi}{2} + |\psi_i(t_k) - \theta_i^*(t_k)|\,, +\frac{\pi}{2} - |\psi(t_k) - \theta^*(t_k)|).$$

Then if $|\psi| = |\phi - \phi_{nh}|$ is initially smaller than $\frac{\pi}{2}$, it will always remain in $[0, \frac{\pi}{2})$.

Proof. The proof of Theorem 3.1 can be found in Appendix A.5, Proof A.2.

3.2 Analysis of Navigation Properties

We will now demonstrate how the predictive navigation scheme preserves the navigation properties of the original, DNF–based controller (2.20). This will be accomplished through a series of corollaries and theorems.

Corollary 3.1 (Finite linear velocity). *Since, by Theorem 3.1, $|\phi - \phi_{nh}| \in [0, \frac{\pi}{2})$, the projection of the field's gradient $\nabla\Phi$ on an agent's longitudinal axis, P, is never zero for $\mathbf{p} \neq \mathbf{p}_d$. Therefore the linear velocity v in (2.20a) does not go to infinity.*

Proof. The proof of Corollary 3.1 can be found in Appendix A.5, Proof A.4.

Corollary 3.2 (Direction of motion). *Theorem 3.1 implies that if an agent starts in the subspace behind its target ($d < 0$ in §2.2.1), with the initial negated gradient vector driving it forward ($P < 0$), only forward motion will be used for navigation and collision avoidance. This is a necessary condition for application to aircraft-like vehicles.*

Proof. The proof of Corollary 3.2 can be found in Appendix A.5, Proof A.5. The requirements of Corollary 3.2 are mild and represent reasonable physical conditions.

Theorem 3.2 (Collision Avoidance). *An agent described by (2.1), navigating under the control laws (2.20a)–(2.20b) remains always* safe, *i.e., no collisions occur at any time.*

When the result of Theorem 3.2 is combined with that of Theorem 3.1 and Corollary 3.2, we get that the agent will avoid collisions without ever resorting to a change in the direction of motion (from forward to reverse).

Proof. The proof of Theorem 3.2 can be found in Appendix A.5, Proof A.6.

Theorem 3.3 (Convergence of the multi–agent system). *An agent described by (2.1), navigating under the control laws (2.20a)–(2.20b), admits a continuous Lyapunov function, V. In addition, the agent converges to its target destination $\mathbf{p}_d$ with the desired orientation ϕ_d.*

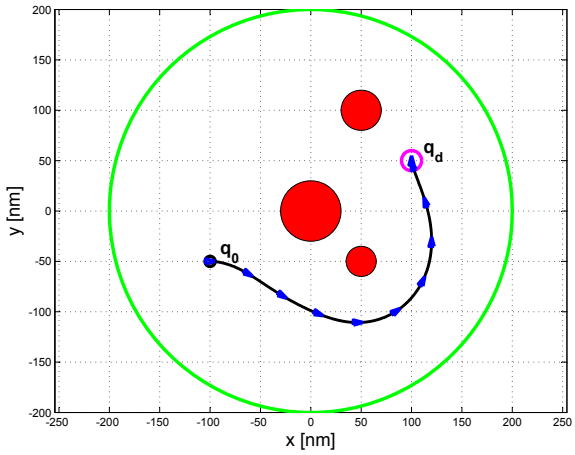
Proof. The proof of Theorem 3.3 can be found in Appendix A.5, Proof A.8.

3.3 Simulation Results (Single agent)

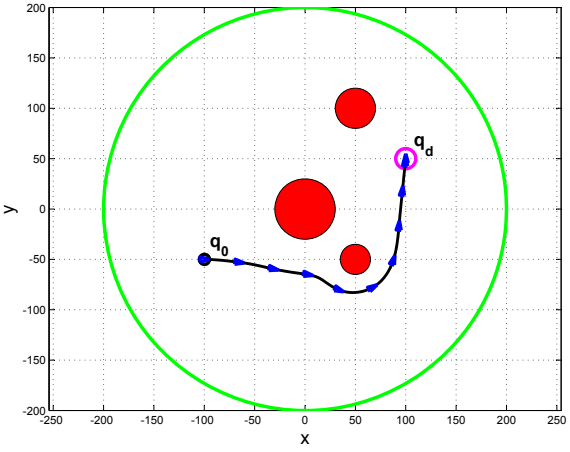
The performance of the proposed scheme is verified via MATLAB simulations. The agent parameters used in the simulation below are given in §2.1.1. In addition to those, $k_\phi = 3 \cdot 10^{-3}$, and $k = 7$. The number of samples is calculated by Lemma 3.1. The cost functional used is as follows:

$$J(t_k, \mathbf{q}, \mathbf{u}, T) = \int_{t_k}^{t_k+T} [(\mathbf{p} - \mathbf{p}_d)^\top Q(\mathbf{p} - \mathbf{p}_d) + R \cdot \omega^2] \mathrm{d}\tau + \Phi(\mathbf{p}(t_k + T)). \tag{3.5}$$

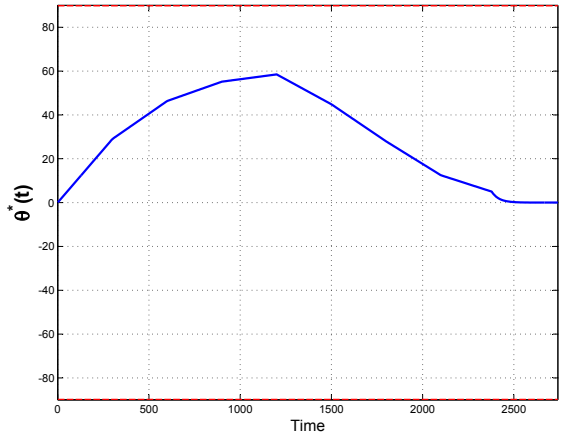
The linear velocity v does not appear in the cost functional as it is regulated independently by Equations (2.20a)–(2.3). We used the *uniform* probability distribution for $\mathcal{P}(\theta)$ in Algorithm 1. Any parameters not provided, will appear in the caption of the respective figure.



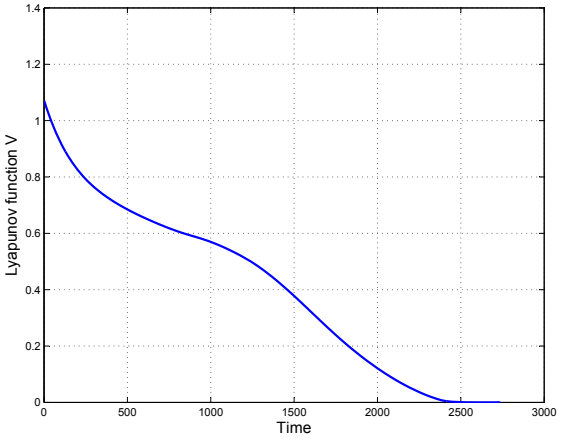
(a) NF–based Controller.



(b) Predictive Navigation scheme.

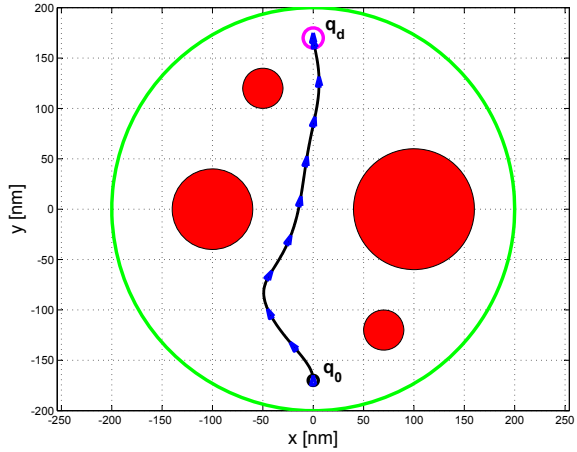


(c) Optimum deviation $\theta^*(t)$.

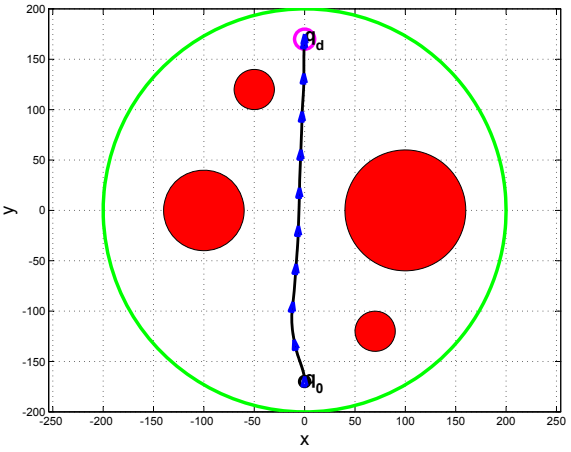


(d) Lyapunov function V (see Proof A.8).

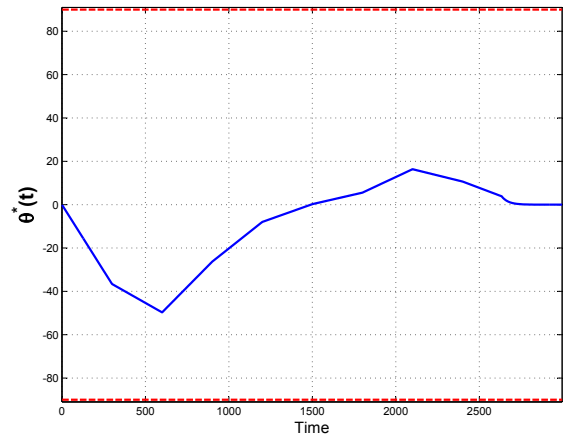
Figure 3.3: Scenario 1 – Parameters: $T = 700$ (sec), $T_c = 300$ (sec), $N_s = 29$.



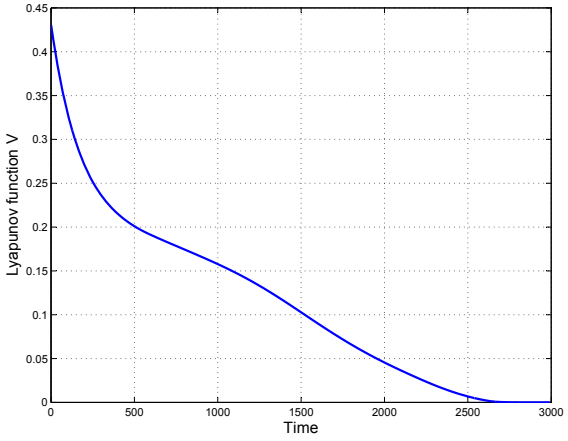
(a) NF–based Controller.



(b) Predictive Navigation scheme.



(c) Optimum deviation $\theta^*(t)$.



(d) Lyapunov function V (see Proof A.8).

Figure 3.4: Scenario 2 – The NF-based controller forces the agent to make unnecessary turns. Using the predictive navigation scheme results in an improved trajectory. Parameters: $T = 500,\ T_c = 300,\ N_s = 29,\ Q = \frac{1}{4 \cdot R_w^2},\ R = 0.$

[This page intentionally left blank.]

Chapter 4

Multi-agent Predictive Navigation

Throughout this chapter, we will use the notation initially introduced in Chapter 2. The multi-agent navigation setting involves N aircraft-like agents navigating in a common workspace, as already illustrated in Figure 2.2(right). Each agent $i \in \mathcal{N} = \{1, 2, \ldots, N\}$ is described by the kinematic model (2.1) and the additional properties described in §2.1.1. In addition, a dipolar Decentralized Navigation Function (DNF) Φ_i (2.12) is defined for each agent. Issues regarding the performance criteria differ between the centralized and decentralized navigation settings. These will be described separately in §4.1 and §4.2 respectively. Finally, in §4.3, we present comparative simulation results.

4.1 Centralized Predictive Navigation

4.1.1 Centralized Predictive Control Scheme

This section provides the details and analysis of the approach presented in §2.2.2 that are specific to a centralized multi-agent setting. Specifically, in this setting, there exists a centralized authority that is responsible for deriving the deviation of all agents, i.e., the decision (deviation) vector

$$\boldsymbol{\theta}^* = [\theta_1^*, \theta_2^*, \ldots, \theta_N^*]^\top.$$

Since the centralized authority has access to the state $\mathbf{q}_i$, control inputs $\mathbf{u}_i$ and deviation θ_i^* of all agents, we will use the following notation for the multi-agent system. The system's configuration, position and control vectors respectively are as follows:

$$\begin{aligned}
\mathbf{q} &\triangleq [\mathbf{q}_1^\top \ \mathbf{q}_2^\top \ \ldots \mathbf{q}_N^\top]^\top \in Q^N, \\
\mathbf{p} &\triangleq [\mathbf{p}_1^\top \ \mathbf{p}_2^\top \ \ldots \mathbf{p}_N^\top]^\top \in \mathbb{R}^{2N}, \\
\mathbf{u} &\triangleq [\mathbf{u}_1^\top \ \mathbf{u}_2^\top \ \ldots \mathbf{u}_N^\top]^\top \in \mathbb{R}^{2N}.
\end{aligned}$$

In the centralized setting, performance is encoded in a single performance measure which is a function of the multi-agent system as a whole. That is,

$$J(t, \mathbf{q}, \mathbf{u}, T) = \int_{t}^{t+T} \Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau)) \mathrm{d}\tau + \sum_{i=1}^{N} \Phi_i(\mathbf{p}(t+T)), \tag{4.1}$$

where the running cost $\Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau))$ is a function of the multi-agent system's state and controls, that encodes the performance criteria, and the terminal cost is the sum of all agents' DNFs. The centralized predictive navigation scheme is depicted in Figure 4.1.

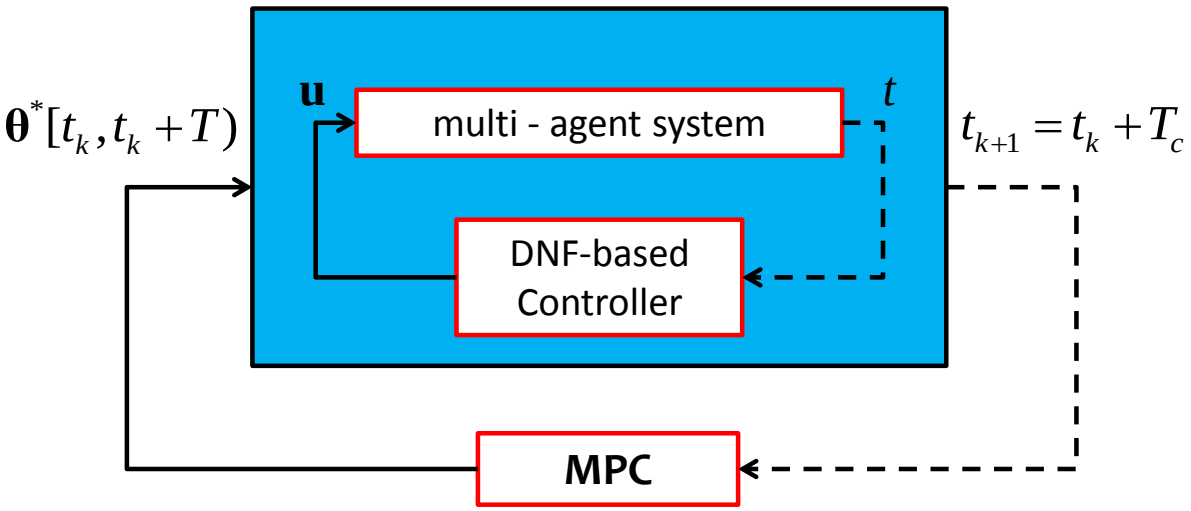


Figure 4.1: The centralized MPC scheme updates the deviation vector $\boldsymbol{\theta}^*$ at each recalculation time t_k, and feeds it to the DNF–based controllers.

The DNF–based controllers, Eq. (2.21), provide inputs at each time instant t, whereas the centralized authority (MPC scheme) updates the deviation vector $\boldsymbol{\theta}^*$ at each recalculation time instant $t_{k+1} = t_k + T_c$, where T_c is the control horizon (see §2.2.2), and $k \in \mathbb{N}$.

We are now ready to restate Problem 2.4, i.e., the centralized multi-agent predictive navigation problem:

Problem 4.1 (Centralized Predictive Navigation). *For the multi-agent system, given a dipolar DNF (2.12), a running cost function $\Lambda(\mathbf{q}, \mathbf{u})$, and a prediction horizon T, derive, for each of the prediction intervals $[t_k, t_k + T)$, the control strategies $\mathbf{u}^*[t_k, t_k + T)$ that minimize (4.1) for each interval, in such a way that their concatenation, applied over $t \in [0, \infty)$, is also a solution to the multi-agent navigation Problem 2.3.*

The optimal control problem to be solved for each prediction interval $[t_k, t_k + T)$ is:

$$J(t_k, \mathbf{q}, \mathbf{u}, T) = \int_{t_k}^{t_k+T} \Lambda(\mathbf{q}(\tau), \mathbf{u}(\tau))\mathrm{d}\tau + \sum_{i=1}^{N} \Phi_i(\mathbf{p}(t_k + T)), \tag{4.2a}$$

$$J^*(t_k, \mathbf{q}, T) = \min_{\boldsymbol{\theta}^*[t_k, t_k+T)} J(t_k, \mathbf{q}, \mathbf{u}, T), \tag{4.2b}$$

$$\boldsymbol{\theta}^*[t_k, t_k + T) = \arg J^*(t_k, \mathbf{q}, T), \tag{4.2c}$$

where $\boldsymbol{\theta}^* = [\theta_1^*, \theta_2^*, \ldots, \theta_N^*]^\top$, as already mentioned. Then the resulting deviation vector $\boldsymbol{\theta}^*[t_k, t_k + T)$ is applied to the multi-agent system, according to (2.21), over the control phase $[t_k, t_{k+1}) = [t_k, t_k + Tc)$, and the FHOCP (4.2) is solved again for the new system state $\mathbf{q}(t_{k+1})$. Note that the recalculation instants, $\{\ldots, t_k, t_{k+1,\ldots}\}$, are common for all agents, i.e., the recalculations are *synchronous* in the centralized setting[1].

Since deviations $\boldsymbol{\theta}^*$ will be calculated in a finite horizon manner, denote as $\boldsymbol{\theta}^*(t)$ the *concatenation* of deviations over $t \in [0, \infty)$ as follows:

$$\boldsymbol{\theta}^*(t) \triangleq \boldsymbol{\theta}^*[0, t_1)|\boldsymbol{\theta}^*[t_1, t_2)|\ldots|\boldsymbol{\theta}^*[t_k, t_{k+1})\ldots, \tag{4.3}$$

obtained iteratively by solving the FHOCP (4.2) at each recalculation time t_k, and applying the resulting deviation $\boldsymbol{\theta}^*[t_k, t_k + T)$ only over the time interval $[t_k, t_{k+1})$. We set $t_0 = 0$ and $\boldsymbol{\theta}^*(0) \triangleq 0$.

Randomized Finite Horizon Optimization (Centralized)

In the centralized setting, we tackle the FHOCP (4.2) similarly to the single agent case. Rather than solving for the deviation $\theta^*[t_k, t_k + T)$, we solve for the deviation vector $\boldsymbol{\theta}^*[t_k, t_k + T) = [\theta_{1k}^*[t_k, t_k + T), \theta_{2k}^*[t_k, t_k + T), \ldots, \theta_{Nk}^*[t_k, t_k + T)]^\top$. To simplify the notation, if necessary, $\boldsymbol{\theta}_k^*$ will be used instead of $\boldsymbol{\theta}^*[t_k, t_k + T)$ to refer to the deviation vector over the time interval $[t_k, t_k + T)$, and $\boldsymbol{\theta}_k^*(t')$ will be used to denote the *value* of the deviation at a specific time instant $t' \in [t_k, t_k + T)$.

We restate Lemma 2.1 in the current setting:

Lemma 4.1 (Number of samples (Centralized)). *The number of samples N_s that guarantees $J^*(t_k, \mathbf{q}, T)$ is a "probable near minimum" of $J(t_k, \mathbf{q}, \mathbf{u}, T)$, to level α and*

[1] This is in contrast to the *Decentralized Event-Triggered* scheme in §4.2.

confidence $1-\delta$*, satisfies:*

$$N_s \geq \frac{\ln(1/\delta)}{\ln(\frac{1}{1-\alpha})}$$

At each recalculation time t_k, the centralized authority calculates a near optimum deviation vector $\boldsymbol{\theta}^*[t_k, t_k+T)$ by executing Algorithm 2.

Algorithm 2: Centralized Finite Horizon Optimization

`Parameters`: α, δ, Θ, $\mathcal{P}(\boldsymbol{\theta})$, T, $J(t_k, \mathbf{q}, \mathbf{u}, T)$

1: `Calculate` a sufficient number of samples N_s using Lemma 4.1.

2: `Generate` N_s i.i.d. random samples $\boldsymbol{\theta}^m$, $m = 1, \ldots, N_s$,
from a set $\Theta \subset (-\frac{\pi}{2}, +\frac{\pi}{2})^{\mathrm{N}}$, according to the probability distribution $\mathcal{P}(\boldsymbol{\theta})$.

3: `for` m = 1 : $\mathrm{N_s}$

4: `Generate` a candidate deviation vector $\boldsymbol{\theta}^m[t_k, t_k+T)$ as:
$\boldsymbol{\theta}^m[t_k, t_k+T) = (1-\frac{\tau}{T})\boldsymbol{\theta}^*(t_k) + (\frac{\tau}{T})\boldsymbol{\theta}^m$, where $\tau \in [0, T)$ is a dummy time variable.

5: `Simulate` the multi-agent system's dynamics over $[t_k, t_k+T)$ using $\boldsymbol{\theta}^m[t_k, t_k+T)$.

6: `Calculate` $J^m(t_k, \mathbf{q}, \mathbf{u}, T)$.

7: `end loop`

8: `Pick` $\boldsymbol{\theta}^*[t_k, t_k+T) = \arg\min_{\boldsymbol{\theta}^m[t_k, t_k+T)} J^m(t_k, \mathbf{q}, \mathbf{u}, T)$.

The i^{th} row of the candidate deviation vector $\boldsymbol{\theta}^m[t_k, t_k+T)$ (Step 4, Algorithm 2) is a line segment (1st degree polynomial) connecting the point $(t_k, \theta_i^*(t_k))$ with the i^{th} element of one of the N_s samples $(t_k+T, \theta_i^m(t_k+T))$ of Step 2, $i \in \mathcal{N}$. Higher degree polynomials would result in "smoother" deviations but the condition $|\theta_i^*(t)| < \frac{\pi}{2}$ (proven in Lemma [# Chapter 3]) would not hold for *any* value of the involved parameters, (see Figure 3.2).

Finally denote by $t_f^i = \inf\{t : ||\mathbf{p}_i - \mathbf{p}_{id}|| \leq r_0\}$ the time instant when agent i enters a neighbourhood r_0 of its destination $\mathbf{p}_{id}$. For $t \geq t_f^i$, θ_i will be given as a function of the

distance $S_i = ||\mathbf{p}_i - \mathbf{p}_{id}||$:

$$\theta_i(t \geq t_f^i) = \theta_i(S_i) = {S_i}^2 \cdot \frac{\theta_i^*(t_f^i)}{{r_0}^2}.$$

We could say that the centralized authority no longer calculates a deviation for this agent for $t \geq t_f^i$. This deviation term has the following mathematical properties:

i. $\theta_i(S_i = r_0) = \theta_i^*(t_f^i)$,

ii. $\theta_i(S_i = 0) = 0$,

iii. $\dot{\theta}_i(S_i = 0) = \dfrac{d\theta_i}{dS_i} \cdot \dfrac{dS_i}{d\mathbf{p}_i} \cdot \dot{\mathbf{p}}_i|_{S_i=0} = 0.$

Therefore $\boldsymbol{\theta}^*(t)$ is restated as in (4.4)

$$\boldsymbol{\theta}^*(t) \triangleq \boldsymbol{\theta}^*[0, t_1)| \ldots |\boldsymbol{\theta}^*[t_k, t_{k+1})| \ldots |\boldsymbol{\theta}^*[t_f^i, t_{k+1})| \ldots |\boldsymbol{\theta}^*(t \geq t_f), \tag{4.4}$$

where t_f^i is the first time an agent (say, agent i) entered the neighbourhood of its destination, and t_f the time instant the last agent did so.

Lemma 4.2 (Continuity and Boundedness of deviations). *Each row of the deviation vector (4.4), where each $\boldsymbol{\theta}^*[t_k, t_k + T)$ is calculated via Algorithm 2 and applied over the control phases $[t_k, t_{k+1})$, $0 < t_{k+1} - t_k < T$, is a continuous function of time (class C^0), that satisfies the bound*

$$|\theta_i^*(t)| < \frac{\pi}{2}, \quad \forall i \in \mathcal{N}.$$

Proof. The proof of Lemma 4.2 can be found in Appendix A.5, Proof A.1.

Sampling Set Θ

The non-holonomic nature of the unicycle and the control law (2.21b) make it impossible to accurately track a desired heading angle. A tracking error, $(\phi_i - \phi_{nhi} - \theta_i^*)$, will be present at all times. This means that the result of Lemma 4.2, i.e., $|\theta_i^*(t)| < \frac{\pi}{2}$, does not guarantee that ϕ_i also satisfies $|\phi_i - \phi_{nhi}| < \frac{\pi}{2}$. Satisfaction of this condition is necessary to preserve the navigation properties of the original DNF–based control law (2.14b). At

each recalculation time t_k, the sampling set $\Theta = \Theta^1 \times \Theta^2 \times \cdots \times \Theta^N \subset (-\frac{\pi}{2}, +\frac{\pi}{2})^N$ has to be adjusted accordingly in order to take into account the tracking error.

Theorem 4.1 (Sampling Set adjustment (Centralized)). *Consider the multi-agent system, whose agents $i \in \mathcal{N}$ are described by Eq. (2.1) and navigate under the control law (2.21b), and denote by ψ_i the angle between the field's gradient and the agent's longitudinal axis. At each recalculation time t_k, let the sampling set Θ in Algorithm 2 be*

$$\Theta_k = \Theta_k^1 \times \Theta_k^2 \times \cdots \times \Theta_k^i \times \cdots \times \Theta_k^N,$$

where

$$\Theta_k^i = (-\frac{\pi}{2} + |\psi_i(t_k) - \theta_i^*(t_k)|\,, +\frac{\pi}{2} - |\psi_i(t_k) - \theta_i^*(t_k)|).$$

Then if $|\psi_i| = |\phi_i - \phi_{nhi}|$ is initially less than $\frac{\pi}{2}$, it will always remain in $[0, \frac{\pi}{2})$, $\forall i \in \mathcal{N}$.

Proof. The proof of Theorem 4.1 is almost identical to that of the decentralized case (Theorem 4.4), which can be found in Appendix A.5, Proof A.2. This proof discusses only the sampling of one agent, i.e., Θ_k^i, a subset of Θ_k. The extension is straightforward.

4.1.2 Analysis of Navigation Properties

We will now demonstrate how the multi-agent predictive navigation scheme preserves the navigation properties of the original, DNF–based controller. This will be accomplished through a series of corollaries and theorems. The main objective of this section is to mathematically verify the applicability of the proposed Centralized Predictive Navigation scheme to aircraft-like vehicles.

Corollary 4.1 (Finite linear velocity). *Since, by Theorem 4.1, $|\phi_i - \phi_{nhi}| \in [0, \frac{\pi}{2})$, $\forall i \in \mathcal{N}$, the projection of the field's gradient $\nabla_i \Phi_i$ on each agent's longitudinal axis, P_i, is never zero for $\mathbf{p}_i \neq \mathbf{p}_{id}$. Therefore the linear velocity v_i in (2.21a) does not go to infinity.*

Proof. The proof of Corollary 4.1 is almost identical to that of the decentralized case (Theorem 4.3), which can be found in Appendix A.5, Proof A.4.

Corollary 4.2 (Direction of motion). *Theorem 4.1 implies that if an agent starts in the subspace behind its target ($d_i < 0$ in §2.2.1), with the initial negated gradient vector driving it forward ($P_i < 0$), only forward motion will be used for navigation and collision avoidance. This is a necessary condition for application to aircraft-like vehicles.*

Proof. The proof of Corollary 4.2 is almost identical to that of the decentralized case (Theorem 4.4), which can be found in Appendix A.5, Proof A.5. The requirements of Corollary 4.2 are mild and represent reasonable physical conditions.

Theorem 4.2 (Collision Avoidance). *A team of agents described by (2.1), navigating under the control law (2.21a) for linear velocities remains always* safe, *i.e., no collisions occur at any time.*

Proof. The proof of Theorem 4.2 is identical to that of the decentralized case (Theorem 4.5) and that of the original DNF–based controller. It can be found in Appendix A.5, Proof A.7.

Theorem 4.3 (Convergence of the multi–agent system). *The multi-agent system described by (2.1), navigating under the control laws (2.21a)–(2.21b), where the deviations (4.4) are computed in a centralized manner, admits a continuous Lyapunov function, V. In addition, each agent i converges to its target destination $\mathbf{p}_{id}$ with the desired orientation ϕ_{id}.*

Proof. The proof of Theorem 4.3 is similar to that of the decentralized case (Theorem 4.6). A proof sketch is provided in Appendix A.5, Proof A.9, and the rest can be inferred from Proof A.10.

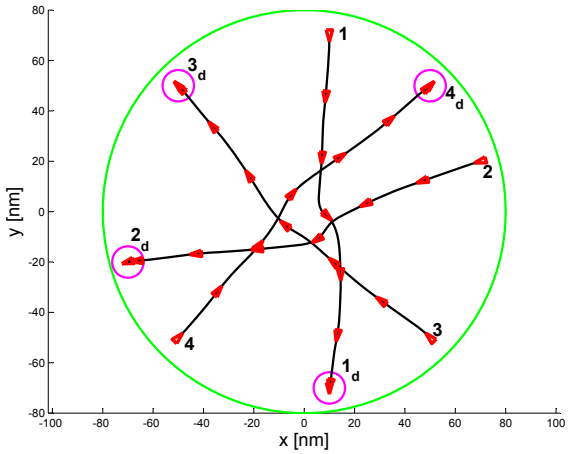
4.1.3 Simulation Results (Centralized)

The performance of the centralized scheme is verified via MATLAB simulations. The agents' parameters used in the simulation below are given in §2.1.1. In addition to those,

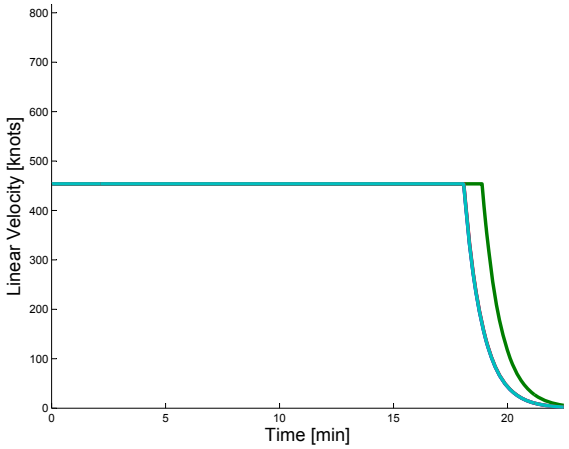
$k_\phi = 5 \cdot 10^{-4}$, and $k = 10$. The number of samples is calculated by Lemma 4.1. The cost functional used is as follows:

$$J_i(t, \mathbf{q_i}, \mathbf{u_i}, T) = \int_t^{t+T} [Q\,||\mathbf{p} - \mathbf{p}_d||^2 + R_1(\mathbf{v} - U_i)^2]d\tau + \sum_{i=1}^{N} \Phi_i(\mathbf{p}(t_k + T), \tag{4.5}$$

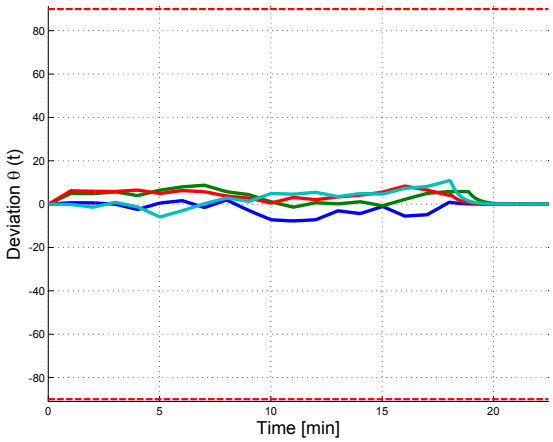
We used the *uniform* probability distribution for $\mathcal{P}(\theta)$ in Algorithm 2. Any parameters not provided, will appear in the caption of the respective figure.



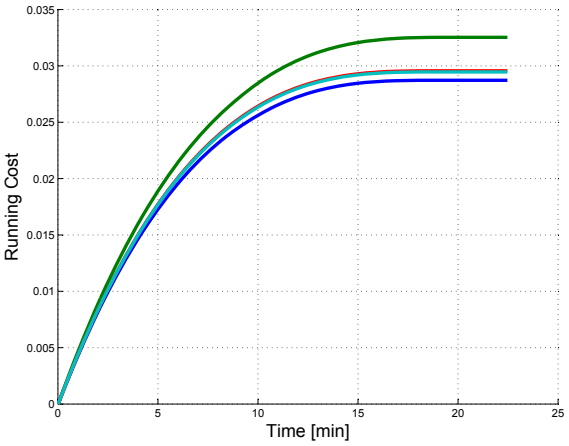
(a) Paths resulting from a run of the centralized predictive navigation scheme.



(b) Linear velocities were maintained at 454 knots.



(c) Optimum deviations $\theta_i^*(t)$.



(d) Evolution of running cost in Eq. (4.5).

Figure 4.2: Parameters: $T = 600$ (10 min), $T_c = 60$ (1 min), $N_s = 88$.

4.2 Decentralized Predictive Navigation

4.2.1 Decentralized Predictive Control Scheme

This section provides the details and analysis of the approach presented in §2.2.2, that are specific to a decentralized multi-agent setting. Specifically, in this setting, each agent is equipped with its own predictive controller, and is responsible for calculating its deviation θ_i^* from the direction of a dipolar DNF, Φ_i, in a decentralized manner. However, since an agent now solves a Finite Horizon Optimal Control Problem with limited information, discrepancies appear between its predicted state and actual state, over each time interval $[t, t+T_c)$. Subsequently, there are also performance discrepancies, in terms of the performance measure (cost functional). A simplified overview of the decentralized predictive navigation scheme is given in Fig. 4.3.

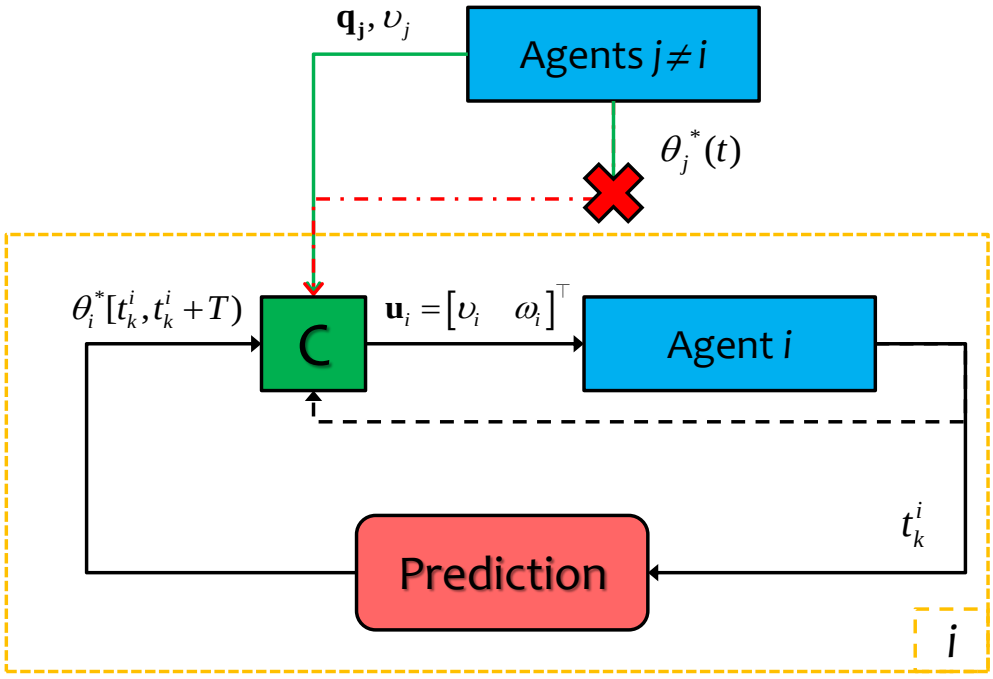


Figure 4.3: The decentralized MPC scheme on each agent updates the deviation θ_i^* at each recalculation time t_k, and feeds it to the DNF–based controller C (2.21). Note that agent i does not have knowledge other agents' decisions, i.e., deviations $\theta_j^*(t)$. The derivation of recalculation time instants $\{\dots, t_k^i, t_{k+1}^i, \dots\}$ is not depicted in this figure.

As we shall see, recalculations of the FHOCP, i.e., executions of Algorithm 3, are *asynchronous* in general. They are derived implicitly by an *execution rule* (event generator).

One additional piece of information that we will require in the decentralized approach is knowledge of all other agents' goal configurations by each agent i. This is accomplished by a broadcasting of each agent's goal configuration $\mathbf{q}_{id}$ to others, at the beginning of the collision avoidance manoeuvre, i.e., at $t = t_0 = 0$. However, this is not mandatory in order to guarantee safety and convergence. It is just a simplification to facilitate the prediction phase, since the design of an estimator is beyond the scope of this work.

Similarly to the single agent and centralized multi-agent settings, we state the cost functional, and the new, decentralized predictive navigation problem, Problem 4.2. For each agent $i \in \mathcal{N}$, the cost functional is given as follows:

$$J_i(t, \mathbf{q}_i, \mathbf{u}_i, T) = \int_t^{t+T} \Lambda_i(\mathbf{q}_i(\tau), \mathbf{u}_i(\tau))d\tau + \Phi_i(t+T), \tag{4.6}$$

where the running cost $\Lambda_i(\mathbf{q}_i, \mathbf{u}_i)$ encodes the performance criteria of agent i only. The terminal cost is the value of a dipolar DNF, Φ_i, at time $t + T$.

Problem 4.2 (Decentralized Predictive Navigation). *For each agent $i \in \mathcal{N}$, given a DNF (2.12), a running cost function $\Lambda_i(\mathbf{q}_i, \mathbf{u}_i)$, and a prediction horizon T, derive, for each of the prediction intervals $[t_k^i, t_k^i + T)$, the control strategies $\mathbf{u}_i^*[t_k^i, t_k^i + T)$ that minimize (4.6) for each interval, in such a way that their concatenation, applied over $t \in [0, \infty)$, is also a solution to the multi-agent navigation Problem 2.3.*

The FHOCP to be solved by each agent i, over a prediction interval $[t_k^i, t_k^i + T)$ is:

$$J_i(t_k^i, \mathbf{q_i}, \mathbf{u_i}, T) = \int_{t_k^i}^{t_k^i+T} \Lambda_i(\mathbf{q}_i(\tau), \mathbf{u}_i(\tau))d\tau + \Phi_i(t_k^i + T), \tag{4.7a}$$

$$J_i^*(t_k^i, \mathbf{q_i}, T) = \min_{\theta_i[t_k^i, t_k^i+T)} J_i(t_k^i, \mathbf{q_i}, \mathbf{u_i}, T), \tag{4.7b}$$

$$\theta_i^*[t_k^i, t_k^i + T) = \arg J_i^*(t_k^i, \mathbf{q_i}, T). \tag{4.7c}$$

The resulting deviation $\theta_i^*[t_k^i, t_k^i + T)$ is applied to the multi-agent system, according to (2.21), over the control phase $[t_k^i, t_{k+1}^i)$, and the FHOCP (4.2) is solved again for the new system state $\mathbf{q}(t_{k+1})$. As mentioned already, in general, $t_{k+1}^i \neq t_k^i + T_c$ in the decentralized event-triggered scheme. However, we have that $t_0^1 = t_0^2 = \cdots = t_0^N = t_0 \triangleq 0$.

Deviations θ_i^* will be calculated in a finite horizon manner. Thus, denote by $\theta_i^*(t)$ the *concatenation* of agent i's deviations over $t \in [0, \infty)$ as follows:

$$\theta_i^*(t) \triangleq \theta_i^*[0, t_1^i) | \theta_i^*[t_1^i, t_2^i) | \ldots | \theta_i^*[t_k^i, t_{k+1}^i) \ldots, \tag{4.8}$$

obtained iteratively by solving the FHOCP (4.7) at each recalculation time t_k^i, and applying the resulting deviation $\theta_i^*[t_k^i, t_k^i + T)$ over the control phase, i.e., only over the time interval $[t_k^i, t_{k+1}^i)$. We set $\theta_i^*(0) \triangleq 0$, $\forall i \in \mathcal{N}$.

Randomized Finite Horizon Optimization (Decentralized)

We restate Lemma 2.1 in the current setting:

Lemma 4.3 (**Number of samples (Decentralized)**). *The number of samples N_s that guarantees $J_i^*(t_k^i, \mathbf{q}, T)$ is a "probable near minimum" of $J_i(t_k^i, \mathbf{q}, \mathbf{u}, T)$, to level α and confidence $1 - \delta$, satisfies:*

$$N_s \geq \frac{\ln(1/\delta)}{\ln(\frac{1}{1-\alpha})}$$

At each recalculation time t_k^i, agent i calculates a near optimum deviation $\theta_i^*[t_k^i, t_k^i + T)$ by executing Algorithm 3.

Each candidate deviation (Algorithm 3, Step 4) is a line segment (1st degree polynomial) connecting the point $(t_k^i, \theta_i^*(t_k^i))$ with one of the N_s sampled points $(t_k^i + T, \theta_i^m(t_k^i + T))$ of Step 2. Higher degree polynomials would result in "smoother" deviations but the condition $|\theta_i^*(t)| < \frac{\pi}{2}$ (proven in Lemma 4.4) would not hold for *any* value of the involved parameters (see Figure 3.2).

Finally denote by $t_f^i = \inf\{t : ||\mathbf{p}_i - \mathbf{p}_{id}|| \leq r_0\}$ the time instant when agent i enters a neighbourhood r_0 of its destination $\mathbf{p}_{id}$. For $t \geq t_f^i$, θ_i will be given as a function of the distance $S_i = ||\mathbf{p}_i - \mathbf{p}_{id}||$: $\theta_i(t \geq t_f^i) = \theta_i(S_i) = {S_i}^2 \cdot \theta_i^*(t_f^i)/{r_0}^2$. We could say that the Predictive Controller is "turned-off". This deviation term has the following properties:

i. $\theta_i(S_i = r_0) = \theta_i^*(t_f^i)$,

ii. $\theta_i(S_i = 0) = 0$,

iii. $\dot{\theta}_i(S_i = 0) = d\theta_i/dS_i \cdot dS_i/d\mathbf{p}_i \cdot \dot{\mathbf{p}}_i|_{S_i=0} = 0$.

Therefore $\theta_i^*(t)$ is restated as in (4.9) and depicted in Figure 4.4:

$$\theta_i^*(t) \triangleq \theta_i^*[0, t_1^i) | \ldots | \theta_i^*[t_k^i, t_{k+1}^i) | \ldots | \theta_i(t \geq t_f^i). \tag{4.9}$$

Algorithm 3: Decentralized Finite Horizon Optimization

`Parameters`: α, δ, Θ, $\mathcal{P}(\theta)$, T, $J(t_k^i, \mathbf{q}, \mathbf{u}, T)$

1: `Calculate` a sufficient number of samples N_s using Lemma 4.3.

2: `Generate` N_s i.i.d. random samples θ_i^m, $m = 1, \ldots, N_s$,
from a set $\Theta \subset (-\frac{\pi}{2}, +\frac{\pi}{2})^{\mathrm{N}}$, according to the probability distribution $\mathcal{P}(\theta)$.

3: `for` $\mathrm{m} = 1 : \mathrm{N_s}$

4: `Generate` a candidate deviation vector $\theta_i^m[t_k^i, t_k^i + T)$ as:
$\theta_i^m[t_k^i, t_k^i + T) = (1 - \frac{\tau}{T})\theta_i^*(t_k^i) + (\frac{\tau}{T})\theta_i^m$, where $\tau \in [0, T)$ is a dummy time variable.

5: `Simulate` the multi-agent system's dynamics over $[t_k^i, t_k^i + T)$ using $\theta_i^m[t_k^i, t_k^i + T)$.

6: `Calculate` $J^m(t_k^i, \mathbf{q}, \mathbf{u}, T)$.

7: `end loop`

8: `Pick` $\theta_i^*[t_k^i, t_k^i + T) = \arg\min_{\theta^m[t_k^i, t_k^i + T)} J^m(t_k^i, \mathbf{q}, \mathbf{u}, T)$.

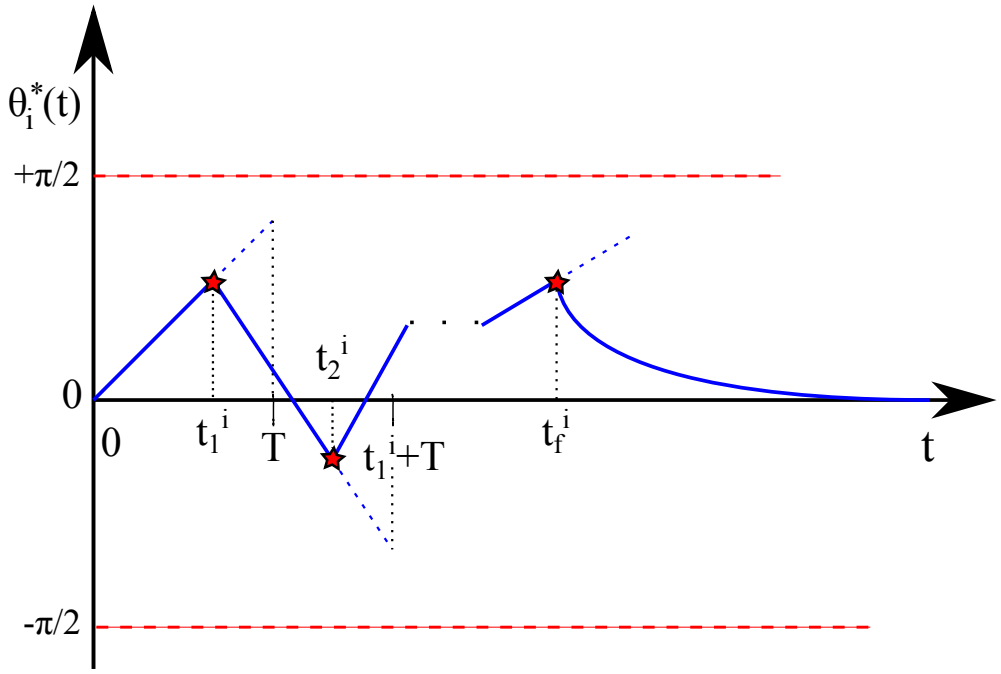


Figure 4.4: Concatenation of deviations as a function of time, Eq. (4.9), for agent i. Stars denote the time instants a new deviation is calculated (either t_k^i or t_f^i).

Lemma 4.4 (Continuity and Boundedness of deviations). *The deviation (4.9), where each $\theta_i^*[t_k^i, t_k^i + T)$ is calculated by Algorithm 3 and applied over the control phases $[t_k^i, t_{k+1}^i)$, $0 < t_{k+1}^i - t_k^i < T$, is a continuous function of time (class C^0) that satisfies the bound*

$$|\theta_i^*(t)| < \frac{\pi}{2}, \ \forall i \in \mathcal{N}.$$

Proof. The proof of Lemma 4.4 can be found in Appendix A.5, Proof A.1.

Sampling Set Θ

The non-holonomic nature of the unicycle and the control law (2.21b) make it impossible to accurately track a desired heading angle. A tracking error, $(\phi_i - \phi_{nhi} - \theta_i^*)$, will be present at all times. This means that the result of Lemma 4.2, i.e., $|\theta_i^*(t)| < \frac{\pi}{2}$, does not guarantee that ϕ_i also satisfies $|\phi_i - \phi_{nhi}| < \frac{\pi}{2}$. Satisfaction of this condition is necessary to preserve the navigation properties of the original DNF–based control law (2.14b). At each recalculation time t_k^i, the sampling set $\Theta \subset (-\frac{\pi}{2}, +\frac{\pi}{2})$ has to be adjusted accordingly in order to take into account the tracking error.

Theorem 4.4 (Sampling Set adjustment (Decentralized)). *Consider an agent described by Eq. (2.1) under the control law (2.21b) and denote by ψ_i the angle between the field's gradient and the agent's longitudinal axis. At each recalculation time t_k^i, let the sampling set Θ in Algorithm 3 be*

$$\Theta_k^i = (-\frac{\pi}{2} + \left|\psi_i(t_k^i) - \theta_i^*(t_k^i)\right|, +\frac{\pi}{2} - \left|\psi_i(t_k^i) - \theta_i^*(t_k^i)\right|).$$

Then if $|\psi_i| = |\phi_i - \phi_{nhi}|$ is initially smaller than $\frac{\pi}{2}$, it will always remain in $[0, \frac{\pi}{2})$.

Proof. The proof of Theorem 4.4 can be found in Appendix A.5, Proof A.2.

Nominal multi–agent System

In the centralized setting (central authority with perfect information) and in the absence of uncertainty (external disturbances, model-system mismatch etc.), the solution of Al-

gorithm 2 for each horizon $[t_k, t_k + T)$ would also be (a probable near) optimum for the actual evolution of agent i (over the control phase). In a decentralized setting however, limited information is a source of uncertainty wrt each agent's predictions. That is, each agent i does not have the information necessary to execute Step 5 of Algorithm 3. First let us state the assumptions wrt information available:

I. Each agent can measure the configuration (position & orientation, $\mathbf{q}_i$) of all other agents at each time instant.

II. Each agent can measure the speed (linear velocity v_i) of all other agents at each time instant. However, it has no knowledge of their nominal speed U_i.

III. Agents broadcast their target configuration $\mathbf{q}_{id}$ to all other agents at $t = t_0^i = 0$.

IV. Agents' predictive controllers do not exchange any information regarding their decisions (future state or input trajectories and deviation trajectory).

Assumptions I, II are present in similar settings, i.e., when DNFs are used for the navigation of nonholonomic agents [4]. The information exchange in Assumption III is minimal compared to the broadcasting of state/input trajectories. Assumption IV is what differentiates our decentralized scheme from a distributed [32] or cooperative approach [9].

Definition 4.1 (Nominal multi–agent System). *We define as* nominal *multi-agent system, wrt agent $i \in \mathcal{N}$, the one whose agents $j \in \mathcal{N}$ are described by (2.1) and navigate under the control laws (2.21a)–(2.21b) with:*

$$\theta_j[t_k^i, t_k^i + T) = \begin{cases} \theta_i^m[t_k^i, t_k^i + T), & j = i \\ 0, & j \neq i \end{cases}, \tag{4.10a}$$

$$\hat{U}_{jd} = \begin{cases} U_{id}, & j = i \\ v_j(t_k^i), & j \neq i \end{cases}, \quad i, j \in \mathcal{N}. \tag{4.10b}$$

Deviations $\theta_i^m[t_k^i, t_k^i + T)$ correspond to the samples generated by Algorithm 3. Regarding the prediction phase of agent i, it starts at the initial conditions: $\hat{\mathbf{q}}_j(t_k^i) = \mathbf{q}_j(t_k^i)$ and $\hat{v}_j(t_k^i) = v_j(t_k^i)$, $\forall j \in \mathcal{N}$, i.e., the predicted trajectories at $\tau = t_k^i$ are equal to the measurements made by agent i at $t = t_k^i$. The rest of the predicted trajectories are derived by calculating the direction of $-\nabla_j \Phi_j$ and applying control laws (2.21a)–(2.21b), as stated above, over $[t_k^i, t_k^i + T)$, $\forall j \in \mathcal{N}$. Therefore uncertainty enters the system since,

at each t_k^i, agent i solves the FHOCP (4.7) by simulating the dynamics of the *nominal* multi-agent system (Def. 4.1), i.e., without taking into account the future decisions of other agents (deviation from $-\nabla_j \Phi_j$, $j \neq i$) and their actual desired absolute speed, U_{jd}. In other words, agent i's prediction is based on the nominal system's assumptions (4.10), not the actual evolution of the multi-agent system.

4.2.2 Event-Triggered Executions

We introduce event-triggered execution of the predictive navigation scheme (independently on each agent), in order to tackle the deterioration in performance, caused by the uncertainty described in the previous subsection. For each agent i, denote by

$$\hat{C}_i(\tau) = \int_{t_k^i}^{\tau} \Lambda_i(\hat{\mathbf{q}}_i(\tau), \hat{\mathbf{u}}_i^*(\tau))dt, \qquad \tau \in [t_k^i, t_k^i + T), \tag{4.11a}$$

$$C_i(\tau) = \int_{t_k^i}^{\tau} \Lambda_i(\mathbf{q}_i(\tau), \mathbf{u}_i^*(\tau))dt, \qquad \tau \in [t_k^i, t_k^i + T) \tag{4.11b}$$

the predicted and real running costs respectively, where $\hat{\mathbf{q}}_i, \hat{\mathbf{u}}_i^*, \mathbf{q}_i, \mathbf{u}_i^*$ correspond to the predicted and real state and input trajectories of agent i while applying the optimum (wrt the nominal system) deviation $\theta_i^*[t_k^i, t_k^i + T) = \arg J_i^*(t_k^i, \mathbf{q}_i, T)$. Due to the uncertainty caused by decentralization, there will be a discrepancy between the predicted and actual running costs. An execution rule will be introduced to derive the recalculation times (events times) $\{t_1^i, t_2^i, \dots, t_k^i, \dots\}$. Its goal will be to maintain the discrepancy below some bound. Consider the inequality: $C_i(\tau) \geq \hat{C}_i(\tau) + c_\epsilon$, where $c_\epsilon > 0$ is the cost discrepancy bound.

Then an appropriate execution rule for the event times is:

$$t_{k+1}^i = \begin{cases} \tau_c, & \tau_c := \inf\{\tau : C_i(\tau) \geq \hat{C}_i(\tau) + c_\epsilon\} \\ t_k^i + T_c, & \text{if } C_i(t_k^i + T_c) < \hat{C}_i(t_k^i + T_c) + c_\epsilon \end{cases} \tag{4.12}$$

where $T_c < T$ is the (maximum) control horizon. Execution rule (4.12) requires the storing of the predicted cost $\hat{C}_i(\tau)$, corresponding to $\theta_i^*[t_k^i, t_k^i + T)$, and monitoring of the above inequality over the time interval $[t_k^i, t_k^i + T)$.

Let the running cost function $\Lambda_i(\cdot)$ be of the general form:

$$\Lambda_i(\cdot) = (\mathbf{q}_i - \mathbf{q}_{id})^\top \boldsymbol{Q}(\mathbf{q}_i - \mathbf{q}_{id}) + R_1(|v_i| - U_i)^2 + R_2\omega_i^2, \tag{4.13}$$

where $\boldsymbol{Q}$ is a matrix of constant coefficients and R_1, R_2 are constant parameters, used for weighting purposes.

Lemma 4.5. *For $\Lambda_i(\cdot)$ of the form (4.13) and a bound c_ϵ, the recalculation times $\{t_1^i, \dots, t_k^i, \dots\}$, implicitly defined by the execution rule (4.12) satisfy*

$$0 < \frac{c_\epsilon}{\Lambda_{Zi,k}^{max}} \leq t_{k+1}^i - t_k^i < T,$$

where

$$\Lambda_{Zi,k}^{max} \geq \Lambda_i(\cdot) - \Lambda_i(\hat{\cdot}),\ \tau \in [t_k^i, t_k^i + T),\ \forall k \in \mathbb{N} \text{ and } i \in \mathcal{N}.$$

Proof. The proof of Lemma 4.5 can be found in Appendix A.5, Proof A.3.

4.2.3 Analysis of Navigation Properties

We will now demonstrate how the multi-agent predictive navigation scheme preserves the navigation properties of the original, DNF–based controller. This will be accomplished through a series of corollaries and theorems. The main objective of this section is to mathematically verify the applicability of the proposed Decentralized Event-triggered Predictive Navigation scheme to aircraft-like vehicles.

Corollary 4.3 (Finite linear velocity). *Since, by Theorem 4.4, $|\phi_i - \phi_{nhi}| \in [0, \frac{\pi}{2})$, $\forall i \in \mathcal{N}$, the projection of the field's gradient $\nabla_i \Phi_i$ on an agent's longitudinal axis, P_i, is never zero for $\mathbf{p}_i \neq \mathbf{p}_{id}$. Therefore the linear velocity v_i in (2.21a) does not go to infinity.*

Proof. The proof of Corollary 4.3 can be found in Appendix A.5, Proof A.4.

Corollary 4.4 (Direction of motion). *Theorem 4.4 implies that if an agent starts in the subspace behind its target ($d_i < 0$ in §2.2.1), with the initial negated gradient vector driving it forward ($P_i < 0$), only forward motion will be used for navigation and collision avoidance. This is a necessary condition for application to aircraft-like vehicles.*

Proof. The proof of Corollary 4.4 can be found in Appendix A.5, Proof A.5. The requirements of Corollary 4.4 are mild and represent reasonable physical conditions.

Theorem 4.5 (Collision Avoidance). *A team of agents described by (2.1), navigating under the control law (2.21a) for linear velocities remains always* safe, *i.e., no collisions occur at any time.*

Proof. The proof of Theorem 4.5 is identical to that of the original DNF–based controller. It can be found in Appendix A.5, Proof A.7.

Theorem 4.6 (Convergence of the multi–agent system). *The multi-agent system described by (2.1), under the control laws (2.21a)–(2.21b), where the deviations (4.9) are computed in a decentralized event-triggered manner, admits a continuous Lyapunov function, V. In addition, each agent i converges to its target destination $\mathbf{p}_{id}$ with the desired orientation ϕ_{id}.*

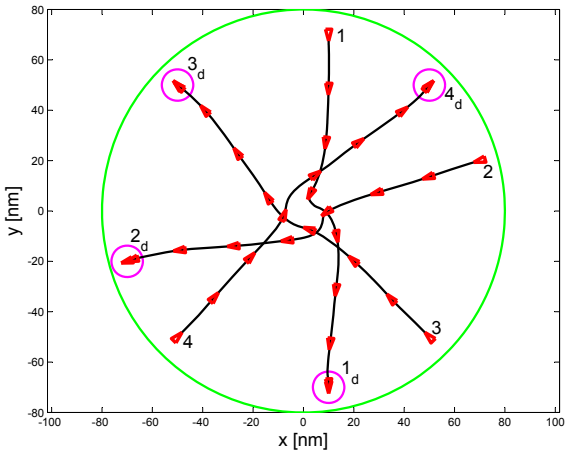
Proof. The proof of Theorem 4.6 can be found in Appendix A.5, Proof A.10.

4.2.4 Simulation Results (Decentralized Event-Triggered)

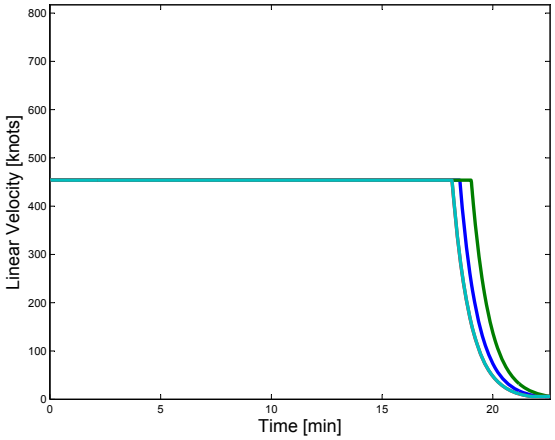
The performance of the centralized scheme is verified via MATLAB simulations. The agents' parameters used in the simulation below are given in §2.1.1. In addition to those, $k_\phi = 5 \cdot 10^{-4}$, and $k = 10$. The number of samples is calculated by Lemma 4.1. The cost functional used is as follows:

$$J_i(t, \mathbf{q_i}, \mathbf{u_i}, T) = \int_t^{t+T} [Q \, ||\mathbf{p}_i - \mathbf{p}_{id}||^2 + R_1(v_i - U_i)^2] d\tau + \Phi_i(t_k + T), \tag{4.14}$$

We used the *uniform* probability distribution for $\mathcal{P}(\theta)$ in Algorithm 2. Any parameters not provided, will appear in the caption of the respective figure. The incremental cost in (4.14) penalizes time spent away from the destination as well as any deviation from the nominal speed, U_i. The performance discrepancy bound is set at each iteration to $c_\epsilon = \frac{1}{T_c} \cdot \hat{C}_i(t_k^i + T_c)$.



(a) Paths resulting from a run of the decentralized E-T predictive navigation scheme.



(b) Evolution of running cost in Eq. (4.5).

Figure 4.5: Parameters: $T = 600$ (10 min), $T_c = 60$ (1 min), $N_s = 22$.

4.3 Comparative Simulation Results

In order to demonstrate the improvement in performance achieved by the proposed scheme, we present (normalized) comparative results for the following multi-agent navigation scenario involving $N = 4$ agents. The agents's initial and target configurations are depicted in Fig. 4.6 by the first and last triangle of each trajectory. The desired absolute speed U_{id} is set to $1 \cdot 10^{-3}$ for all agents. Finally, $R_w = 4$, $r_0 = 0.3$ (purple circles), $r_i = 0.05$, $\forall i \in \mathcal{N}$, $k_\phi = 5 \cdot 10^{-4}$ and $k = 10$.

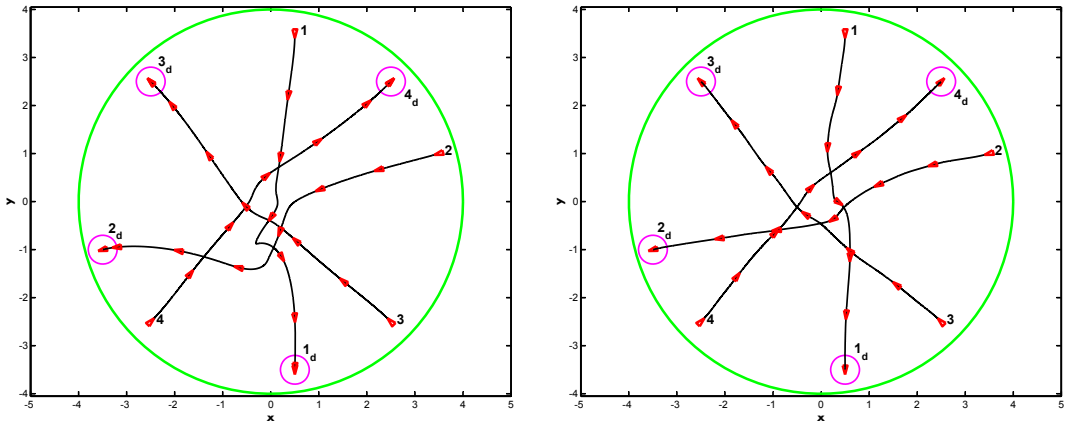
For the Predictive Navigation scheme (corresponding to Fig. 4.6(b)) we use for all agents the same cost functional:

$$J_i(t, \mathbf{q_i}, \mathbf{u_i}, T) = \int_t^{t+T} [Q\,||\mathbf{p}_i - \mathbf{p}_{id}||^2 + R_1(|v_i| - U_i)^2]d\tau + \Phi_i(t+T), \tag{4.15}$$

where $T = 1200$, $Q = \frac{10^{-4}}{4 \cdot R_w^2}$ and $R_1 = 1 \cdot 10^5$. Finally, $\alpha = 0.10$, $\delta = 0.10$ (resulting in $N_s = 22$) and $\mathcal{P}$ is the uniform distribution. The incremental cost in (4.15) penalizes time spent away from the destination as well as any deviation from the nominal speed, U_i. The performance discrepancy bound is set at each iteration to $c_\epsilon = \frac{1}{T_c} \cdot \hat{C}_i(t_k^i + T_c)$ and $T_c = 400$.

The convergence and collision avoidance properties of the proposed scheme are verified by Figures 4.6(b) and 4.8 respectively. The improvement in the linear velocities used by

each agent is depicted in Fig. 4.7. Finally, Fig. 4.9 shows the deviation trajectories $\theta_i^*(t)$ of each agent and Fig. 4.10 illustrates the effect of Theorem 4.4.



(a) Original DNF-based control scheme, (2.14a)-(2.14b).

(b) Decentralized Event-triggered Predictive Navigation, (2.21a) - (2.21b).

Figure 4.6: A multi-agent navigation scenario involving 4 agents. The trajectories resulting from the application of the original and the proposed schemes are depicted in 4.6(a) and 4.6(b) respectively. The red triangles, representing agent position and orientation, are not to scale.

In order to quantitatively demonstrate the efficacy of the proposed scheme, the original DNF-based approach (2.14a)-(2.14b) is compared with three instances of our scheme. These are:

i. Centralized Predictive Navigation (PN),

ii. Decentralized Predictive Navigation,

iii. Decentralized Event-triggered (E-T) Pred. Navigation.

The navigation scenario is the same as above. In each case, the overall performance, in terms of the incremental cost of (4.15), is calculated along each agent's trajectory and the mean values for 10 runs are presented in Table 4.1 for comparison.

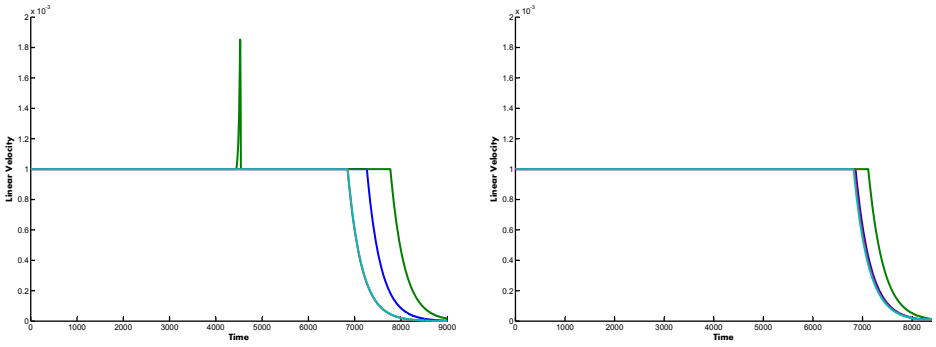


Figure 4.7: Linear velocities used during navigation. Left: DNF-based control law. The spike around $t = 4500$ corresponds to agent 2. Right: Proposed scheme. In accordance with the cost functional (4.15), the linear velocity was maintained equal to the nominal speed, U_i. Note that agents also converge to their destinations faster using the proposed scheme.

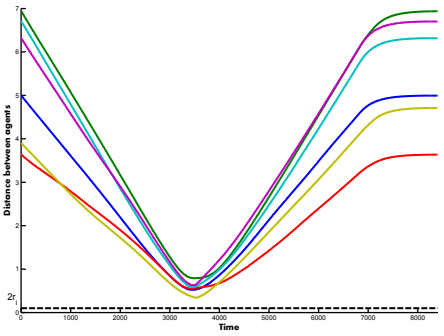


Figure 4.8: Distance between each pair of agents during the above navigation scenario. The dashed line represents the minimum safety distance, $2 \cdot r_i$.

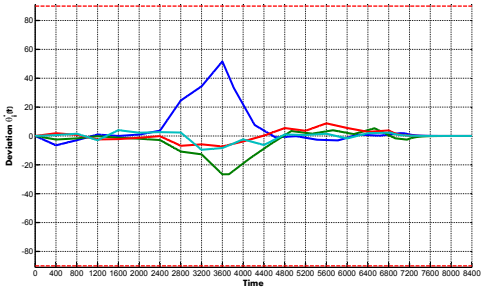


Figure 4.9: Deviation from the direction of the DNF's gradient, $\theta_i^*(t)$. Only agents 1 (blue) and 2 (green) recalculated due to performance discrepancies.

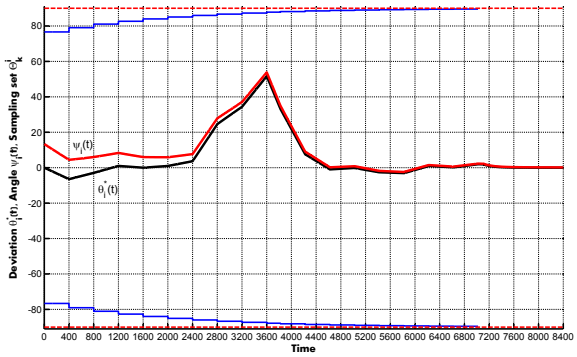


Figure 4.10: Deviation (black), angle between the field's gradient and the agent's longitudinal axis (red) and sampling set adjustments (blue) for agent $i = 1$. As the tracking error decreases, the sampling set tends to $(-\frac{\pi}{2}, +\frac{\pi}{2})$.

Table 4.1: Comparative results for 10 runs per instance of scheme

	Total Running Cost			
Agent	DNF law	Centrl. PN	Decntrl. PN	Decntrl. E-T PN
#1	0.1869	0.1829	0.1859	0.1830
#2	0.3175	0.2045	0.2372	0.2047
#3	0.1860	0.1860	0.1864	0.1863
#4	0.1862	0.1857	0.1859	0.1857
Sum	0.8767	0.7591	0.7954	0.7597

[This page intentionally left blank.]

Chapter 5
Conclusions and Future Work

5.1 Discussion

We proposed three schemes, for the navigation or a single agent, a centralized multi-agent, and a decentralized multi-agent systems, that consider the issue of operating performance.

Our main contribution, the Decentralized Event-triggered Predictive Navigation scheme, employs MPC to calculate deviations from the direction of a dipolar DNF's gradient. Randomized algorithms are used to find a probable near minimizer of the cost functional. The sampling set involved is adjusted at each iteration to take into account the agents' nonholonomic nature. Event-triggered recalculation of the deviation trajectories is used to tackle the discrepancy between the expected and actual performance, caused by the limited exchange of information between agents. The approach focuses on aircraft-like agents but can be reformulated to address scenarios involving other multiple robotic vehicles (wheeled, underwater). It is proven that the proposed scheme preserves the navigation properties of the original DNF-based approach. Simulations are used to demonstrate the performance improvement wrt previous methods and verify the collision avoidance and convergence properties of the scheme.

Major issues that demand our attention are the computational cost of predicting the evolution of the multi-agent system (Algorithm 3, Step 5), and the scalability of our scheme. Possbile solutions are proposed in the following section.

5.2 Future Work

Future research directions are towards the extension of the proposed scheme to a 3D navigation setting while avoiding an explosion in computational cost. To this end, further decentralization could be employed. Specifically, one could use a limited sensing/communication range for each aircraft-like vehicle of the multi-agent system. This

will reduce the computational burden of predicting the motion of N agents over the prediction horizon. It will also render the overall scheme scalable, since additional agents will not necessarily increase the number of agents in one's sensing range, and thus optimization time (execution time of Algorithm 3, Step 5).

In addition, it is possible to remove the requirement of broadcasting each agent's goal configuration information to other agents. We propose that a "motion estimator" can be designed. This module will be used to facilitate the prediction process in Algorithm 3 (see §4.2.1) in two ways. First of all, it will lift the aforementioned requirement, effectively removing the need for communication between agents (they will still need to sense/measure the configuration and linear velocity though). Second of all, it could reduce the computational cost of simulating the multi-agent system's dynamics over the prediction horizon. This could be achieved by using an *abstraction* of the multi-agent system for prediction purposes.

Furthermore, finding a problem-specific, but computationally efficient, method to tackle the finite-horizon optimization (4.7) is also of interest. However, there is another randomized optimization that could be used; the variant of Simulated Annealing in [33], which offers finite-time guarantees. It remains to be seen if it can out-perform the Randomized Algorithms [11, 15] employed in our work.

Finally, we would like to consider more complex (dynamic) aircraft models, as well as the effects of external, bounded or stochastic, disturbances (wind) on the discrepancy between prediction and actual performance. Our scheme will probably need to be "robustified" if it is to account for such disturbances.

Bibliography

[1] S. Maniatopoulos, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Decentralized event-triggered predictive navigation for aircraft-like vehicles," *The 2012 American Control Conference*, June 2012, (to appear). vi, 5

[2] J. L. Piovesan and H. G. Tanner, "Randomized model predictive control for robot navigation," in *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ser. ICRA'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 1817–1822. vii, x, xi, xiv, 3, 4, 5, 15, 19, 22, 23, 25

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. Journal of Robotics Research*, vol. 5, pp. 90–98, April 1986. x, 3, 64

[4] S. G. Loizou, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Decentralized feedback stabilization of multiple nonholonomic agents," in *in 2004 IEEE International Conference on Robotics and Automation*, 2003, pp. 3012–3017. x, 3, 5, 48

[5] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–508, 1992. x, 3, 67

[6] W. B. Dunbar and R. M. Murray, "Distributed receding-horizon control with application to multi-vehicle formation stabilization," *Automatica*, vol. 42, pp. 549–558, 2006. x, 3

[7] G. Roussos, G. Chaloulos, K. Kyriakopoulos, and J. Lygeros, "Control of multiple non-holonomic air vehicles under wind uncertainty using model predictive control and decentralized navigation functions," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, dec. 2008, pp. 1225 –1230. x, 3, 11

[8] Y. Kuwata, A. Richards, T. Schouwenaars, and J. How, "Decentralized robust receding horizon control for multi-vehicle guidance," in *American Control Conference, 2006*, june 2006, p. 6 pp. x, 3

[9] E. Franco, T. Parisini, and M. M. Polycarpou, "Design and stability analysis of cooperative receding-horizon control of linear discrete-time agents," *International Journal of Robust and Nonlinear Control*, vol. 17, no. 10-11, pp. 982–1001, 2007. x, 3, 48

[10] T. Keviczky, F. Borrelli, K. Fregene, D. Godbole, and G. J. Balas, "Decentralized Receding Horizon Control and Coordination of Autonomous Vehicle Formations," *Control Systems Technology, IEEE Transactions on*, vol. 16, no. 1, pp. 19–33, 2008. x, 3

[11] M. Vidyasagar, "Randomized algorithms for robust controller synthesis using statistical learning theory," *Automatica*, vol. 37, pp. 1515–1528, 2001. xi, xiv, 4, 5, 22, 23, 58

[12] A. Eqtami, D. Dimarogonas, and K. Kyriakopoulos, "Event-triggered strategies for decentralized model predictive controllers," *IFAC World Congress*, 2011. xi, 4, 75

[13] ——, "Novel event-triggered strategies for model predictive controllers," *50th IEEE Conf. Decision and Control & Eur. Control Conf.*, 2011. xi, 4, 75, 76

[14] P. Varutti, T. Faulwasser, B. Kern, M. Kogel, and R. Findeisen, "Event-based reduced-attention predictive control for nonlinear uncertain systems," in *Proc. IEEE Int Computer-Aided Control System Design Symposium*, 2010, pp. 1085–1090. xi, 4, 75

[15] G. C. R. Tempo and F. Dabbene, *Randomized Algorithms for Analysis and Control of Uncertain Systems*, ser. Communications and Control Engineering. Springer, 2003. xiv, 4, 5, 22, 23, 58

[16] (2004) User manual for the base of aircraft data (bada). Eurocontrol Experimental Centre. [Online]. Available: http://www.eurocontrol.fr/projects/bada/ xvi, 11

[17] G. Roussos and K. J. Kyriakopoulos, "Decentralised navigation and collision avoidance for aircraft in 3D space," *2010 American Control Conference, Baltimore, USA*, 2010. xxii, 19, 67

[18] S. G. Loizou and K. J. Kyriakopoulos, "Closed loop navigation for multiple holonomic vehicles," Control Systems Lab, National Technical University of Athens, Tech. Rep. 1, 2002. xxii, 18, 68, 69, 70

[19] I. R. K. K. E. S. A. L.-V. G. Chaloulos, J. Lygeros and P. Casek, "Comparative study of conflict resolution methods," iFLY Project, Deliverable D5.1, Tech. Rep., June 2009. 1

[20] H. Tanner and J. Piovesan, "Randomized receding horizon navigation," *Automatic Control, IEEE Transactions on*, vol. 55, no. 11, pp. 2640 –2644, nov. 2010. 4

[21] R. Scattolini, "Architectures for distributed and hierarchical model predictive control - a review," *Journal of Process Control*, vol. 19, pp. 723–731, 2009. 5

[22] A. M. Bloch, *Nonholonomic Mechanics and Control*, S. S. Antman, J. E. Marsden, L. Sirovich, and S. Wiggins, Eds. Springer, 2003, vol. 24. 7, 8

[23] D. Bertsekas, *Nonlinear programming*, ser. Optimization and Neural Computation Series. Athena Scientific, 1995. 12

[24] H. Tanner, S. Loizou, and K. Kyriakopoulos, "Nonholonomic stabilization with collision avoidance for mobile robots," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 3, 2001, pp. 1220 –1225 vol.3. 16

[25] S. G. Loizou and K. J. Kyriakopoulos, "Closed loop navigation for multiple nonholonomic vehicles," Control Systems Lab, National Technical University of Athens, Tech. Rep. 2. 16, 68

[26] R. W. Brockett, "Asymptotic stability and feedback stabilization," in *Differential Geometric Control Theory*. R.W. Brockett, R.S. Millman and H.J. Sussman, Eds., Boston, Birkhauser, 1983, pp. 181–191. 16

[27] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE Trasnactions on Robotics and Automation*, vol. 17, pp. 947–951, 2001. 17

[28] J. Cortes, "Achieving coordination tasks in finite time via nonsmooth gradient flows," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, dec. 2005, pp. 6376 – 6381. 18

[29] G. P. Roussos, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Distributed 3D navigation and collision avoidance for multiple nonholonomic agents," *European Control Conference 2009*, pp. 1830–1835, 2009. 18, 71

[30] M. M. Zavlanos and K. J. Kyriakopoulos, "Decentralized motion control of multiple mobile holonomic agents," in 11^{th} *Mediterranean Conference on Control and Automation.*, 2003. 18, 69

[31] K. J. K. S. G. Loizou, D. V. Dimarogonas and M. M. Zavlanos, "A feedback stabilization and collision avoidance scheme for multiple independent non-point agents," *Automatica*, vol. 42, no. 2, pp. 229–243, 2006. 18, 69, 70, 81

[32] G. Chaloulos, P. Hokayem, and J. Lygeros, "Distributed Hierarchical MPC for Conflict Resolution in Air Traffic Control," in *American Control Conference*, Baltimore, MD, USA, June 2010. 48

[33] A. Lecchini-Visintini, J. Lygeros, and J. M. Maciejowski, "Simulated annealing: Rigorous finite-time guarantees for optimization on continuous domains." in *NIPS*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. MIT Press, 2007. 58

[34] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in Applied Mathematics*, vol. 11, pp. 412–442, 1990. 64, 66, 67

[35] H. G. Tanner, S. G. Loizou, and K. J. Kyriakopoulos, "Nonholonomic navigation and control of cooperating mobile manipulators," *IEEE Transactions on Robotics and Automation*, vol. 19, pp. 53–64, 2002. 67

[36] G. P. Roussos, D. V. Dimarogonas, and K. J. Kyriakopoulos, "3D navigation and collision avoidance for a non-holonomic vehicle," *2008 American Control Conference, Seattle, Washington, USA*, pp. 3512–3517, 2008. 67, 68

[37] D. Kirk, *Optimal control theory: an introduction*, ser. Prentice-Hall networks series. Prentice-Hall, 1970. 71, 72, 73

[38] S. Jung, "Nonlinear model predictive control systems: Stability, robustness and real-time implementation," Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, New York, May 2002. 73

[39] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1680–1685, 2007. 74, 75

[40] H. Khalil, *Nonlinear systems.* Prentice Hall, 2002. 74

[41] D. Liberzon, *Switching in Systems and Control.* Birkhauser, 2003. 75, 86

[42] F. H. Clarke, Y. S. Ledyaev, R. J. Stern, and P. R. Wolenski, *Nonsmooth analysis and control theory*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1998. 77, 84

[43] F. Clarke, *Optimization and nonsmooth analysis*, ser. Classics in applied mathematics. SIAM, 1990. 77, 78

[44] ——, "Discontinuous feedback and nonlinear systems," 8^{th} *IFAC Symposium on Nonlinear Control Systems*, pp. 1–29, 2010. 77, 78

[45] D. Shevitz and B. Paden, "Lyapunov stability theory of nonsmooth systems," *IEEE Transactions on Automatic Control*, vol. 39, no. 9, pp. 1910–1914, 1994. 77, 78, 79, 84

[46] J. Cortes, "Discontinuous dynamical systems," *IEEE Control Systems Magazine*, vol. 28, no. 3, pp. 36–73, 2008. 77

[47] A. Filippov, *Differential equation with discontinuous right-hand sides*. Kluwer Academic Publishers, 1998. 77, 84

Appendix A

Mathematical Tools and Proofs

A.1 Note on Navigation Functions

A.1.1 Artificial Potential Fields

The idea of tackling the robot navigation (motion planning) problem using artificial potential fields came from Oussama Khatib [3]. The proposed potential function consisted of two terms, an attractive and a repulsive potential:

$$U(q) = U_{att}(q) + U_{rep}(q). \tag{A.1}$$

The attractive potential does not depend on the obstacles while the repulsive potential does not depend on the desired configuration (destination).

The robot (agent) is driven by an artificial force, which is based on the vector field generated by the gradient of the artificial potential:

$$\vec{F}(q) = -\nabla_q U(q) \tag{A.2}$$

However, this approach is prone to local minima, i.e., $\exists q \neq q_d$ s.t. $\vec{F}(q) = 0$. These local minima can "trap" the agent and prevent successful convergence to the destination (goal configuration).

A.1.2 Rimon-Koditschek Navigation Functions

Navigation Functions (NFs) are a class of scalar valued analytic maps on analytic manifolds with boundary, in particular *sphere worlds* [34]. This class is invariant under composition with analytic diffeomorphisms and the existence of a smooth NF is guaranteed

on any smooth manifold. The gradient vector field of a NF, if integrated, produces curves to the destination point that never leave the free space.

Briefly, a sphere world is a compact connected subset of Euclidean n-space E^n whose boundary is formed from the disjoint union of a finite number, $M+1$, of $(n-1)$-spheres. Thus, the *workspace* (largest sphere) is defined as

$$\mathcal{W} \triangleq \{q \in E^n : ||q||^2 \leq \rho_0^2\} \tag{A.3}$$

and the remaining M spheres bound the obstacles

$$\mathcal{O}_j \triangleq \{q \in E^n : ||q - q_j||^2 < \rho_j^2\},\ j = 1 \dots M \tag{A.4}$$

Note that for simplicity, the workspace is centered at the origin of the coordinate system. The *free space* (configuration space) is the result of removing all obstacles from $\mathcal{W}$:

$$\mathcal{F} \triangleq \mathcal{W} - \bigcup_{j=1}^{M} \mathcal{O}_j \tag{A.5}$$

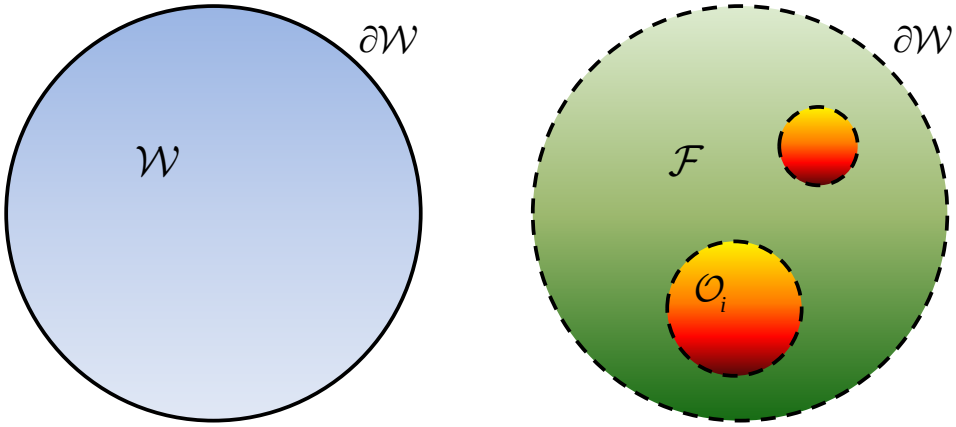


Figure A.1: Workspace $\mathcal{W}$ (blue), free space $\mathcal{F}$ (green) and obstacles (red) in E^2.

This sphere world (see Fig. A.1) is just a "model space"; a valid diffeomorphism to any manifold preserves the navigation properties of a NF, which are stated in the following.

Definition A.1 (Navigation Function [34]). *Let $\mathcal{F} \subset E^n$ be a compact connected analytic manifold with boundary. A map $\varphi : \mathcal{F} \to [0, 1]$, is a* navigation function *if it is:*

1. *Analytic on $\mathcal{F}$ (locally convergent power series exists);*
2. *Polar on $\mathcal{F}$, with unique minimum at $q_d \in \mathring{\mathcal{F}}$ (interior of $\mathcal{F}$);*
3. *Morse on $\mathcal{F}$ (all critical points are non-degenerate);*
4. *Admissible on $\mathcal{F}$ (uniformly maximal on $\partial\mathcal{F}$).*

Intuitively, φ takes the value of "1" at the boundary of $\mathcal{F}$ and the value of "0" only at the desired configuration q_d. The navigation function proposed in [34] is a composition of three functions, used to obtain all of the desired properties (Definition A.1):

$$\varphi \triangleq \sigma_d \circ \sigma \circ \hat{\varphi} = \frac{\gamma_d}{(\gamma_d^k + \beta)^{1/k}}$$

where

$$\hat{\varphi} = \frac{\gamma}{\beta} = \frac{\gamma_d^k}{\beta}, \quad \sigma(x) = \frac{x}{1+x}, \quad \sigma_d(x) = (x)^{1/k}, \ k \in \mathbb{N}$$

The target and obstacle functions, γ_d and β respectively, are defined as follows:

$$\gamma_d(q) \triangleq ||q - q_d||^2 \geq 0, \quad \beta(q) \triangleq \prod_{j=0}^{M} \beta_j(q) \geq 0, \ \forall q \in \mathcal{F}$$

The control law

$$\dot{q}(t) = -\nabla_q \varphi(q(t)),$$

where φ is a navigation function, applied to a holonomic agent, whose state is q, solves the motion planning problem for this agent. This method can be applied to any spherical agent moving in a workspace with obstacles, whose configuration space connected components are sphere worlds. In the case of a *non-point* agent, the Minkowski sum of agent with obstacles leads to the configuration space.

Finally, it should be noted that, even though a sufficiently large k guarantees that the destination is the only minimum (no local minima), the manifestation of *saddle points* (unstable equilibria) is unavoidable. However, the set of initial conditions that will result

in the agent getting trapped by a saddle constitute a set of Lebesgue measure zero. Therefore, in real applications, finite computation arithmetic renders it practically impossible for an agent to remain in a measure zero set.

A.1.3 Dipolar Navigation Functions

Conventional Navigation Functions are not suitable for the control of a non-holonomic vehicle, as they do not take into account the kinematic constraints that apply on such a vehicle. Use of the original Navigation Function as introduced by Rimon and Koditschek in [34], [5] with a feedback law for the control of a nonholonomic vehicle can lead to undesired behavior, like having the vehicle rotate in place. In order to overcome this difficulty *Dipolar Navigation Functions* have been developed [35], that offer a significant advantage: the integral lines of the resulting potential field are all tangent to the desired orientation at the origin, eliminating the need for in-place rotation at the origin, as the vehicle is driven there with the desired orientation. This is achieved by using the plane whose normal vector is parallel to the desired orientation, and includes the origin, as an additional artificial obstacle (see Fig. A.2).

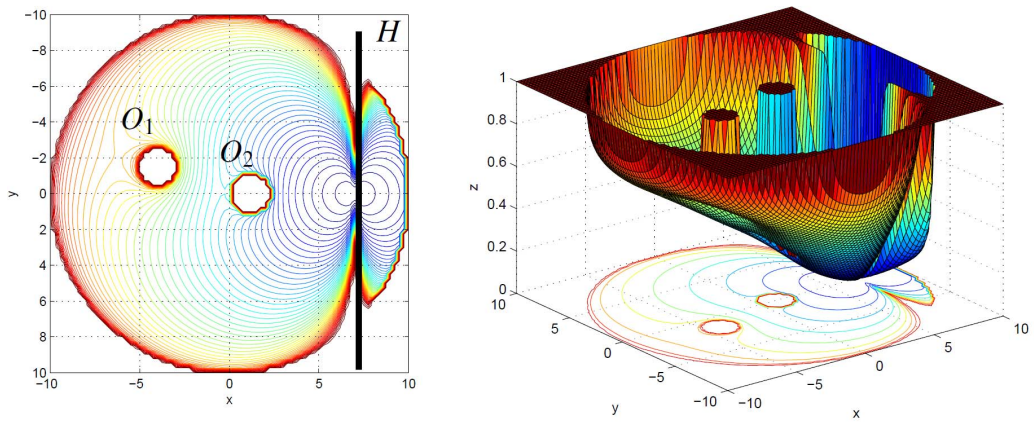


Figure A.2: [17] A Dipolar Navigation Function in a 2D workspace with two obstacles. Left: Level sets and artificial obstacle H. Right: Potential field over the workspace.

We briefly elaborate on dipolar Navigation Functions, as they appeared in [36]:

$$\Phi(p) = \frac{\gamma_d}{(\gamma_d^k + H_{nh} \cdot G \cdot \beta_0)^{1/k}}, \tag{A.6}$$

where $\gamma_d(p) = ||p - p_d||^2$ is the distance from the destination (only position, not orienta-

tion) and the obstacle function is as in §A.1.2:

$$G = \prod_{i=1}^{M} g_i \tag{A.7a}$$

$$g_i = ||p - p_i||^2 - (r + r_i)^2, \; i = 1 \dots M \tag{A.7b}$$

with r, p, r_i being the radius of the vehicle, and the position and radius of obstacle i, respectively, and M is the number of obstacles. As the workspace is considered spherical with radius r_{world}, the workspace bounding obstacle is[1] $\beta_0 = (r_{world} - r)^2 - ||p||^2$.

The factor H_{nh} is what makes the potential field dipolar. As explained before it is responsible for the repulsive potential created by the artificial obstacle used to align the trajectories at the origin with the desired orientation ϕ_d (either roll-pitch-yaw or just heading angle):

$$H_{nh} = \epsilon_{nh} + p_{nh} \tag{A.8a}$$

$$p_{nh} = \left|\left| J_d^{\top} \cdot (p - p_d) \right|\right|^2 \tag{A.8b}$$

$$J_d = J(\phi_d) \tag{A.8c}$$

where ϵ_{nh} is a small positive constant and $J(\cdot)$ is the transformation matrix between body-fixed and earth-fixed velocities[2]. Finally, k is again a positive tuning parameter.

Navigation Function (A.6) provides almost global convergence to the agent's destination, along with guaranteed collision avoidance [25]. The potential of such a Navigation Function in a 2D workspace with two obstacles O_1, O_2 is shown in Figure A.2. The target is with orientation $\phi_d = 0$ and the corresponding nonholonomic obstacle H is the line $x = 7$.

A.1.4 Multi-agent Navigation Functions

The Centralized Approach

Rimon-Koditschek Navigation Functions are not suitable for the navigation of multiple robots (agents), without proper modifications. Multi-Robot Navigation Functions (MRNFs) were introduced, in a centralized setting, in [18]. A central authority was as-

[1] Corrected wrt [36].

[2] See this thesis, §2.1.1, for the 2D case and [36] for the detailed, 3D case.

sumed to have knowledge of each agent's goal configuration (destination). The resulting potential function has the same form as in §A.1.2:

$$\varphi \triangleq \sigma_d \circ \sigma \circ \hat{\varphi} = \frac{\gamma_d}{(\gamma_d^k + G)^{1/k}},$$

but the obstacle function, G, is calculated differently. In particular, besides the distance between agents, it also encodes the possible collision schemes, termed *relations*, (see Fig. A.3 for an illustration). The set of relations between the members of a set can be defined as the set of all possible collision schemes between the members. A binary relation is a relation between two robots. Any relation can be expressed as a set of binary relations. A "relation tree" is the set of robots-obstacles that form a linked team. The number of binary relations in a relation is called the relation *level* (see Figures A.3 and A.4).

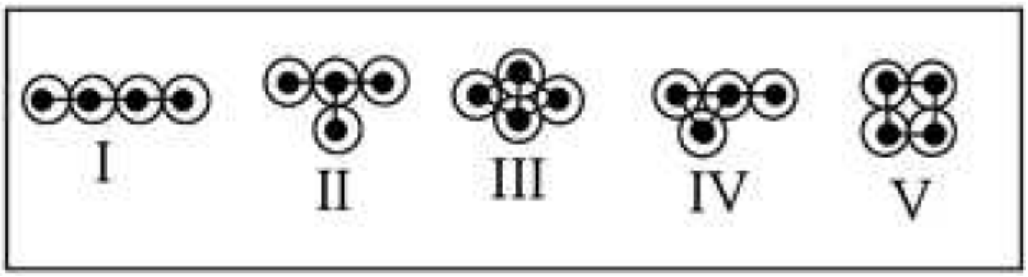


Figure A.3: [18] I, II are level-3; IV, V are level-4 and III is a level-5 relation.

In the definition of G (see [18] for details) three functions are used. Namely, the Robot Proximity Function measures the distance between two robots, the Relation Proximity Function provides a measure of the distance between the robots involved *in a relation* and the Relation Verification Function is zero if a relation holds, while no other relation from the same level holds.

The four properties of a NF, §A.1, are proven to hold for the new formulation.

Decentralized Navigation Functions

The centralized multiple robot navigation function [18] was extended to a decentralized navigation setting in [30] and [31]. In contrast to the centralized case, each agent plans its actions without knowing the destinations of the other agents. Asymptotic stability is guaranteed by the existence of a global Lyapunov function for the whole system, which is actually the sum of the separate NFs.

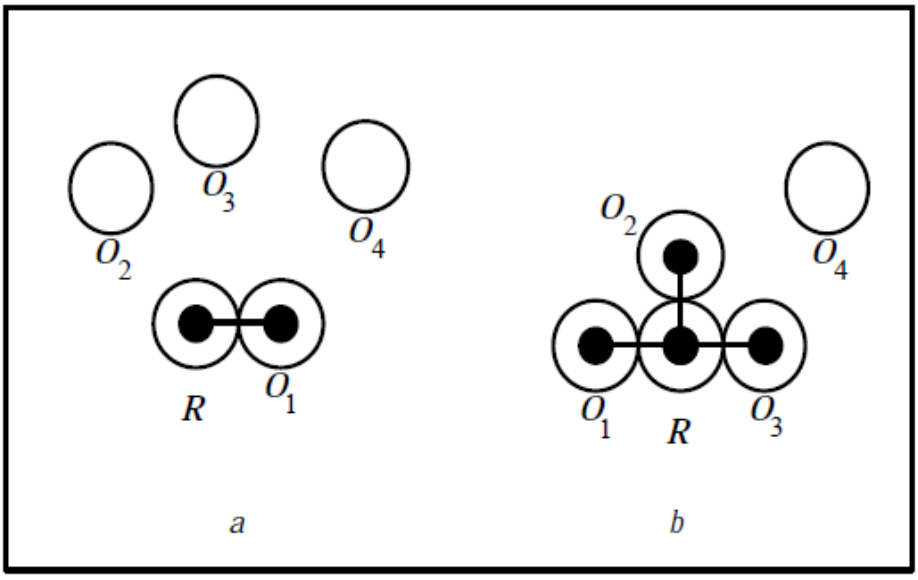


Figure A.4: [18] Part (a) represents a level-1, and part (b) a level-3 relation wrt agent R.

A Decentralized Navigation Function (DNF) is defined for each agent i:

$$\varphi_i(q) = \frac{\gamma_{di} + f_i}{\left((\gamma_{di} + f_i)^k + G_i\right)^{1/k}}, \; i \in \mathcal{N} \tag{A.9}$$

where $\mathcal{N} = \{1 \dots N\}$, where N is the total number of agents, and $q = [q_1 \; q_2 \dots q_N]^\top$. The key difference of the decentralized method with respect to the centralized is that the control law of each agent ignores the destinations of the others. The function f_i is added to the goal function γ_{di} so that the cost function φ_i attains positive values in proximity situations even when agent i has already reached its destination. It was proven in [31] that, the destination point is a non-degenerate local minimum of φ_i. The construction of G_i and f_i is explained in great detail also in [31].

Figure A.5 shows a contour plot of a DNF of an agent in an environment of 3 (other) agents denoted by Ai, [31]. The destination (goal) is also depicted.

The potential function (A.9) contains a *time-varying* element which corresponds to the movement in time of all the other agents apart from i. The time derivative of $\varphi_i(q)$ is given by:

$$\dot{\varphi}_i = \frac{\partial \varphi_i}{\partial q_i}\dot{q}_i + \frac{\partial \varphi_i}{\partial t} = \frac{\partial \varphi_i}{\partial q_i}\dot{q}_i + \sum_{j \neq i} \frac{\partial \varphi_i}{\partial q_j}\dot{q}_j$$

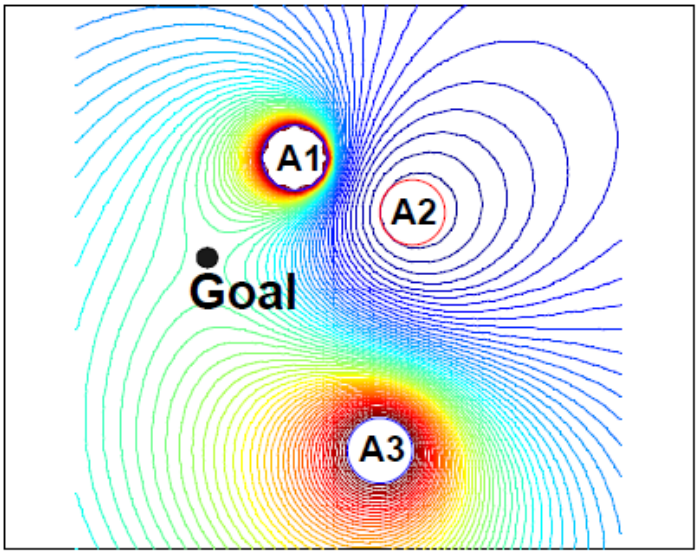


Figure A.5: Contour graph of a DNF as an agent moves amongst other agents.

The term $\frac{\partial \varphi_i}{\partial t}$ does not appear in the case of only static obstacles, as in the single agent case, §A.1.2. In the decentralized multi-agent case, it cannot be neglected in the stability analysis of the system.

Finally, DNFs have been combined with Dipolar NFs in [29] to tackle the distributed navigation and collision avoidance problem for nonholonomic, aircraft-like vehicles. In this case, the resulting potential is the one that is used throughout this thesis:

$$\Phi_i(q) = \frac{\gamma_{di} + f_i}{\left((\gamma_{di} + f_i)^k + H_{nhi} \cdot G_i \cdot \beta_{0i}\right)^{1/k}}, \; i \in \mathcal{N} \tag{A.10}$$

A.2 Note on Optimal and Model-Predictive Control

A.2.1 Optimal Control

Optimal Control theory is aimed at solving the following problem:

Problem A.1 (**Optimal Control Problem** [37]). *Find an admissible control* $u^* \in \mathcal{U}$ *which causes the system*

$$\dot{x}(t) = f(x(t), u(t), t), \; x(t_0) = x_0$$

to follow an admissible trajectory $x^* \in \mathcal{X}$ *that* minimizes *the performance measure*

$$J := h(x(t_f), t_f) + \int_{t_0}^{t_f} \Lambda(x(t), u(t), t)dt,$$

where t_f *is the final time,* $x(t_f)$ *is the final state and* Λ *is a given function (the running cost). Then,* u^* *is called an* optimal *control and* x^* *an* optimal *trajectory.*

If a relationship of the form:

$$u^*(t) = g(x(t), t)$$

can be found for the optimal control at time t*, then the function* $g(x(t), t)$ *is called the* optimal control law*, or the* optimal policy*.*

In a particular problem, either g or h may be missing. Furthermore, the Optimal Control Problem (OCP) is subject to one of the following boundary conditions [37]:

- Problems with Fixed Final Time:
  i. Final state specified;
  ii. Final state free;
  iii. Final state lying on a surface;
- Problem with Free Final Time:
  i. Final state fixed;
  ii. Final state free;
  iii. Final state lies on a moving point;
  iv. Final state lying on a surface;
  v. Final state lying on a moving surface;

The performance measure $\Lambda(x(t), u(t), t)$ quantitatively evaluates the performance of the system and is selected by the control designer. In certain cases the problem statement may clearly indicate what to select for a performance measure, whereas in other problems the selection is a subjective matter. Optimal Control Problems include, but are not limited to, minimum-time problems, terminal control problems, minimum-control-effort problems, tracking problems and regulation problems.

A.2.2 Model-Predictive Control

The optimal feedback control system could be derived by solving the Hamilton-Jacobi-Bellman (HJB) partial differential equation (PDE), for a given cost functional. For the linear system with quadratic cost, the HJB equation recuces to an ordinary differential equation, i.e., the Riccati equation, [37]. However, this PDE is generally hard and usually impossible to solve for the *nonlinear* system, in that the nonlinearity results in non-convexity on the control system [38]. In addition, most control systems have hard constraints on the states and control inputs. These include, but are not limited to, safe operation limits (state) and actuator saturation constraints (input). These remarks motivate the development and application of Model-Predictive Control (MPC) schemes.

MPC is a feedback control scheme that generates the control inputs based on iterative, open-loop optimization, over a finite horizon. For each iteration, the measured state acts as an initial condition. In particular, at a calculation time t, the current system state $x(t)$ is sampled and a cost minimizing control law is computed for a relatively short time horizon $[t, t+T)$. That is, one aims at minimizing a cost functional (performance measure) of the form

$$J(t, x, u, T) = \int_t^{t+T} \Lambda(x(\tau), u(\tau))d\tau + M(x(t+T)), \tag{A.11}$$

which consists of an incremental cost (also called running cost) and a terminal cost. The functional $J(\cdot)$ quantifies the cost of flowing along a system trajectory $x[t, t+T)$, with $x(t)$ the initial condition, under the control law $u[t, t+T)$. In the above, t denotes (current) time, T is the (fixed) prediction horizon and $\Lambda(x, u)$ is a positive definite function of x and u (running cost function). The function $V(\cdot)$ is an approximation of the infinite horizon cost-to-go from $t+T \to \infty$. A practical way of incorporating state and input constraints is through exterior penalty function ([38]) which, in general, implies a relaxation of the hard constraints. Stability and robustness require additional constraints and/or tightening of the existing constraints.

Definition A.2 (Nonlinear Model-Predictive Control scheme). *Consider the nonlinear dynamical system $\dot{x} = f(x, u)$, where $x \in \mathbb{R}^n$ is the state of the system and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input and f satisfies the standard local Lipschitz continuity conditions. Then, given a performance measure (A.11), the Finite Horizon Optimal*

Control Problem (FHOCP) at time t is to determine the optimal control input:

$$u^*[t, t+T) \triangleq \arg\min_u J(t, x, u, T),$$
$$s.t. \quad \dot{x} = f(x, u),\ x \in \mathcal{X} \subset \mathbb{R}^n,\ u \in \mathcal{U},$$

where $\mathcal{X}$ is the admissible region of the state space. This control law is then applied over the interval $[t, t+T_c)$, the control phase, where $0 < T_c < T$ is the (typically fixed) control horizon[3] *and the FHOCP is solved again for the updated system state $x(t+T_c)$.*

This process is then repeated (iterated) until the control objectives have been met.

A.3 Note on Event-Triggered Control

A.3.1 Event-Triggered Real-Time Scheduling

Event-triggered control was introduced by Tabuada [39] as a means of scheduling stabilizing control tasks on embedded processors, in real-time. This approach is based on the notion of Input-to-State Stability (ISS) [40]. A short overview of event-triggered control is presented in the following.

Consider a generic nonlinear system

$$\dot{x} = f(x, u) \tag{A.12}$$

and a feedback controller $u = k(x)$. Typically, sensor measurements and actuator updates are performed in a time-triggered fashion. That is, the system's state is sampled at time instants $t_0, t_1, t_2, t_3 \ldots$, the control input is computed as $u(t_i) = k(x(t_i))$ and the actuator values are updated. The term "time-triggered" simply means that $t_{i+1} - t_i = \tau_s$, where $\tau_s > 0$ is the sampling or update period. In [39], the case of event-triggered executions is considered. The sequence $t_0, t_1, t_2, t_3 \ldots$ of time instants is neither periodic nor pre-specified, but rather implicitly defined by an execution (triggering) rule, which is a function of the system's state. A measurement error e is introduced:

$$t \in [t_i, t_{i+1}) \Rightarrow e(t) = x(t_i) - x(t) \tag{A.13}$$

[3] In an *instantaneous* MPC scheme, T_c is equal to one timestep of the discrete implementation.

The feedback controller has been designed as to render the closed-loop system

$$\dot{x} = f(x, k(x+e)) \tag{A.14}$$

ISS with respect to the measurement error e, (A.13).

Definition A.3 (ISS Lyapunov function [39]). *A smooth function $V : \mathbb{R}^n \to \mathbb{R}_0^+$ is said to be an ISS Lyapunov function for the closed loop system (A.14), if there exist class $\mathcal{K}_\infty$ functions $\underline{\alpha}, \overline{\alpha}, \alpha$ and γ satisfying:*

$$\underline{\alpha}(|x|) \leq V(x) \leq \overline{\alpha}(|x|) \tag{A.15a}$$

$$\frac{\partial V}{\partial x} f(x, k(x+e)) \leq -\alpha(|x|) + \gamma(|e|) \tag{A.15b}$$

Then, by executing the control task when

$$\gamma(|e|) = \sigma\alpha(|x|), \ 0 < \sigma < 1 \tag{A.16}$$

there exists an ISS Lyapunov function for (A.12) and it is guaranteed to be decreasing. Note that, at execution time t_i, we have $e(t_i) = x(t_i) - x(t_i) = 0$ and $\gamma(|e(t_i)|) = 0$. The execution rule (A.16) guarantees global asymptotic stability by construction. In [39], it is proven that this rule does not result in an accumulation point (Zeno behavior), [41].

A.3.2 Event-Triggered Model-Predictive Control

Execution of a Nonlinear MPC (NMPC) scheme in an event-triggered fashion has been explored, amongst others, in [14], [13] for a single plant and in [12] in a decentralized setting. We shall use [13] as a reference point to state the basic principles behind event-triggered NMPC strategies.

Consider a nonlinear continuous time system with additive perturbation $w(t)$ (may represent modelling errors, external disturbances, other forms of uncertainty):

$$\dot{x}(t) = f(x(t), u(t)) + w(t), \tag{A.17}$$

where the additive term is bounded, $w(t) \in \mathcal{W} \subset \mathbb{R}^n$.

The *nominal* system is defined as (A.17) without the additive perturbation term:

$$\dot{x}(t) = f(x(t), u(t)). \tag{A.18}$$

The plant is controlled by a nonlinear Model-Predictive Controller (NMPC), as in §A.2.2. In [13], the FHOCP also includes a terminal state constraint, $x(t_i + T) \in \mathcal{E}_f \subset \mathbb{R}^n$, and a tightened constraint set $\mathcal{X}_{t-t_i}$, instead of $\mathcal{X}$, for robustness purposes, where t_i is a state measurement time (which is also a FHOCP recalculation time).

The goal of an event-triggered strategy is to enlarge, as much as possible, the inter-calculation period, $t_{i+1} - t_i$, for the system (A.17). The enlargement of the inter-calculation period results in the overall reduction of control updates which is desirable in numerous occasions, e.g., energy consumption reasons. By denoting as $\hat{x}(t_i+\tau, u(\cdot), x(t_i))$ the *predicted* state of the nominal system (A.18) at time $t_i + \tau$, based on a measurement of the actual state at time t_i, while applying a control trajectory $u(\cdot; x(t_i))$, an error e is defined, as in the previous section (A.13). However, in the event-triggered NMPC scheme, this "prediction" error represents the discrepancy (mismatch) between the actual state trajectory of the system and the one predicted during the solution of the FHOCP. This discrepancy is due to the perturbation $w(t)$. The prediction error is defined as:

$$e(t_i + \tau) = ||x(t_i + \tau) - \hat{x}(t_i + \tau, u(\cdot), x(t_i))||, \quad \tau \geq 0$$

Note that $\hat{x}(t_i, u(\cdot), x(t_i)) \equiv x(t_i)$. Given some assumptions, the prediction error defined above can be shown to be upper bounded by a function of time and system parameters.

Instead of measuring the system's state at the next time instant and solving the FHOCP for this new measurement, a triggering rule dictates the recalculation time t_{i+1}. In order to guarantee convergence, the optimal cost functional $J^*(\cdot)$ is used as a Lyapunov function. The need to have $J^*(\cdot)$ decreasing, provides with the sufficient conditions for triggering (see [13] for details).

A.4 Elements from Nonsmooth Analysis and Discontinuous Feedback

We review some elements from nonsmooth analysis, discontinuous feedback and Lyapunov theory for nonsmooth dynamical systems that have been used in the stability (conver-

gence) analysis of the proposed schemes. The interested reader can refer to [42], [43], [44], [45] and [46] for additional details on the topic of nonsmooth analysis from a control theoretic perspective.

We consider the vector differential equation with discontinuous right-hand side:

$$\dot{x} = f(x), \tag{A.19}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is measurable and essentially locally bounded. In such case, the classical concept of "solution" of a differential equation is inappropriate. We will adopt Filippov's solution concept in this work. Other alternatives include *Caratheodory* solutions and *Sample-and-hold* solutions (see [46] for a detailed overview).

Definition A.4 (Filippov Solution [47]). *In the case when n is finite, the vector function $x(\cdot)$ is called a solution of (A.19) in $[t_0, t_1]$ if it is absolutely continuous on $[t_0, t_1]$ and there exists $N_f \subset \mathbb{R}^n$, $\mu(N_f) = 0$, such that for all $N \subset \mathbb{R}^n$, $\mu(N) = 0$ and for almost all $t \in [t_0, t_1]$ we have:*

$$\dot{x} \in K[f](x) \equiv \overline{co}\{\lim_{x_i \to x} f(x_i) | x_i \notin N_f \cup N\} \tag{A.20}$$

Here, $\overline{co}$ denotes the closed "convex hull" and the set $K[f](x)$ is called the Filippov set. Another way to state Filippov solutions is via differential inclusions (multi-valued differential equations), as presented in [44]. The above definition of a solution, along with the assumption that f is measurable, guarantees the uniqueness of solutions of (A.19).

In the following, we shall need the definition of *Lipschitz* continuity.

Definition A.5 (Lipschitz Property [44]). *A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be Lipschitz on a set $S \subset \mathbb{R}^n$ if there exists $K \geq 0$ such that, for all $x, y \in S$,*

$$|f(y) - f(x) \leq K|y - x|.$$

We say that f is *locally* Lipschitz on S if each point $z \in S$ admits a radius $r > 0$ and a constant K (both depending on z) such that the Lipschitz condition holds for all $x, y \in B(, r)$. This is equivalent to requiring that the Lipschitz condition hold on any bounded subset S' of S (for some K depending on S') [44]. The Lipschitz property is closed under many operations, such as sums, lower or upper envelopes, compositions, etc. It is a fundamental result in analysis that a function which is Lipschitz on an open set in $\mathbb{R}^n$ is differentiable almost everywhere in the set (Rademacher's Theorem).

Lyapunov stability theorems have been extended to nonsmooth dynamical systems in [45]. There, the authors use the concept of *generalized gradient* which, for the case of finite-dimensional spaces is given by the following definition:

Definition A.6 (Generalized Gradient [43]). *Let $V : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz function. The generalized gradient $\partial_C V(x)$ of V at x is given by:*

$$\partial_C V(x) = \overline{co}\{\lim_{x_i \to x} \nabla V(x_i) | x_i \notin \Omega_V\}, \tag{A.21}$$

where Ω_V is the set of points in $\mathbb{R}^n$ where V fails to be differentiable.

See [44] for some properties in the calculus of generalized gradients.

Stability theorems for nonsmooth systems require the Lyapunov function to be *regular*. Regularity is based on the concept of *generalized derivative* which was defined by Francis Clarke as follows:

Definition A.7 (Generalized directional Derivative [43]). *Let f be Lipschitz near x and v be a vector in $\mathbb{R}^n$. The generalized directional derivative of f at x in the direction v is defined as:*

$$f^0(x; v) = \lim_{y \to x} \sup_{t \downarrow 0} \frac{f(y + tv) - f(y)}{t}, \tag{A.22}$$

which should not be confused with the *Dini derivate*.

Regularity of a function can now be defined as follows:

Definition A.8 (Regular function [43]). *The function $f : \mathbb{R}^n \to \mathbb{R}$ is called regular if*

i. $\forall v$, the usual one-sided directional derivative $f'(x; v)$ exists and

ii. $\forall v$, $f'(x; v) = f^0(x; v)$.

The following chain rule provides a calculus for the time derivative of the Lyapunov function in the nonsmooth case:

Theorem A.1 ([45]). *Let x be a Filippov solution to $\dot{x} = f(x)$ on an interval containing t and $V : \mathbb{R}^n \to \mathbb{R}$ be a Lipschitz and regular function. Then $V(x(t))$ is absolutely continuous, $\frac{d}{dt}V(x(t))$ exists almost everywhere (a.e.) and*

$$\frac{d}{dt}V(x(t)) \in^{a.e.} \dot{\tilde{V}}(x) := \bigcap_{\xi \in \partial V(x(t))} \xi^\top K[f](x(t))$$

We shall use the following, nonsmooth, version of LaSalle's invariance principle to prove the convergence of the prescribed system:

Theorem A.2 ([45]). *Let Ω be a compact set such that every Filippov solution to the autonomous system $\dot{x} = f(x)$, $x(0) = x(t_0)$ starting in Ω is unique and reamins in Ω for all $t \geq 0$. Let $V : \Omega \to \mathbb{R}$ be a time independent regular function, s.t. $v \leq 0$ $\forall v \in \dot{\tilde{V}}(x)$ (if $\dot{\tilde{V}}(x)$ is the empty set then this is trivially satisfied). Define $S = \{x \in \Omega | 0 \in \dot{\tilde{V}}(x)\}$. Then every trajectory in Ω converges to the largest invariant set, M, in the closure of S.*

A.5 Mathematical Proofs

Proofs of *all* lemmas, corollaries and theorems presented in this work have been collected here for easy reference. Since the three settings (single agent, centralized and decentralized multi-agent) have many similarities, many proofs are only presented once, for the sake of brevity. In this case, any noteworthy differences in the derivation of the proof are highlighted beneath the respective lemma, corollary or theorem.

Proof A.1 (Proof of Lemmas 3.2, 4.2, and 4.4 | Continuity and Boundedness). The continuity part is a direct result of Algorithms 1,2,3, respectively (Step 4) and the definition of $\theta_i(t \geq t_f^i)$. Consider a recalculation time t_k^i. Then

$$\theta_i^*(t_k^{i-}) = \theta^*_{i,[t_{k-1}^i, t_{k-1}^i+T)}(t_k^i) \text{ and } \theta_i^*(t_k^{i+}) = \theta_{i,[t_k^i, t_k^i+T)}(t_k^i) = \theta_i^*(t_k^{i-}),$$

because of Algorithms 1,2,3 respectively, (Step 4) for $\tau = 0$ (dummy variable). Now consider the time instant at which agent i's predictive controller is "turned-off", t_f^i. Then $\theta_i(t_f^{i+}) = \theta_i(S_i = r_0) = \theta_i^*(t_f^{i-})$ by construction. Thus, $\theta_i^*(t)$ is a continuous function of time.

We have $-\frac{\pi}{2} < \theta_i^*[0, t_1^i) | \theta_i^*[t_1^i, t_2^i) | \dots | \theta_i^*[t_k^i, t_{k+1}^i) | \cdots < \frac{\pi}{2}$ since $\theta_i^*(0) \triangleq 0$, each $\theta_i^*(t_k^i + T) \in \Theta \subset (-\frac{\pi}{2}, +\frac{\pi}{2})$ and each pair $\theta_i^*(t_k^i)$, $\theta_i^*(t_{k+1}^i)$ is connected by a line segment, $\forall k \in \mathbb{N}$. Finally $\left|\theta_i(t \geq t_f^i)\right| < \left|\theta_i^*(t_f^{i-})\right| < \dfrac{\pi}{2}$ since $\theta_i(t_f^i) = \theta_i^*(t_f^{i-})$, $\theta_i(S_i = 0) = 0$ and the only

critical point of $\theta_i(t \geq t_f^i)$ in $S_i \in [0, r_0]$ is at $S_i = 0$ by construction:

$$\dot{\theta}_i(S_i = 0) = \frac{d\theta_i}{dS_i} \cdot \frac{dS_i}{d\mathbf{p}_i} \cdot \dot{\mathbf{p}}_i|_{S_i=0} = 0.$$

Thus, deviations also satisfy the boundedness condition $|\theta_i^*(t)| < \frac{\pi}{2}$. ■

Proof A.2 (Proof of Theorems 3.1, 4.1, and 4.4 | Sampling Set).

At a recalculation time t_k (or t_k^i in the decentralized case), consider the dynamics of the term $[\psi_i - \theta_i^*]$ (which is continuous because of Lemmas 3.2, 4.2, 4.4 respectively) over a time interval spanned by $\tau \in [0, T)$:

$$\begin{aligned}\frac{d[\psi_i - \theta_i^*]}{dt} &= \dot{\psi}_i - \dot{\theta}_i^* = \dot{\phi}_i - \dot{\phi}_{nhi} - \dot{\theta}_i^* = \omega_i - \dot{\phi}_{nhi} - \dot{\theta}_i^* = \\ &= -k_\phi(\phi_i - \phi_{nhi} - \theta_i^*) + \dot{\phi}_{nhi} + \dot{\theta}_i^* - \dot{\phi}_{nhi} - \dot{\theta}_i^* = -k_\phi(\psi_i - \theta_i^*) \Rightarrow \\ &\Rightarrow \dot{\psi}_i = -k_\phi(\psi_i - \theta_i^*) + \dot{\theta}_i^* \Rightarrow \psi_i(\tau) = [\psi_i(0) - \theta_i^*(0)]e^{-k_\phi \tau} + \theta_i^*(\tau).\end{aligned}$$

Therefore: $|\psi_i(\tau)| \leq |\psi_i(0) - \theta_i^*(0)|\, e^{-k_\phi \tau} + |\theta_i^*(\tau)| \leq |\psi_i(0) - \theta_i^*(0)| + |\theta_i^*(\tau)|$. From Algorithms 1,2,3, note that $\theta_i^*(T) \in \Theta_0^i \Rightarrow |\theta_i^*(\tau)| \in [0, \frac{\pi}{2} - |\psi_i(0) - \theta_i^*(0)|)$. Thus: $|\psi_i(\tau)| < |\psi_i(0) - \theta_i^*(0)| + \frac{\pi}{2} - |\psi_i(0) - \theta_i^*(0)| \Rightarrow |\psi_i(\tau)| < \frac{\pi}{2}$, $\tau \in [0, T)$. This results in $|\psi_i(t_1^i)| < \frac{\pi}{2}$ since $t_1^i \in [0, T)$ and therefore $|\psi_i(t)| < \frac{\pi}{2}$, $\forall t > 0$, by also employing the following result of Lemmas 3.2, 4.2, 4.4 respectively: $\left|\theta_i^*(t_f^i)\right| < \frac{\pi}{2}$. ■

Proof A.3 (Proof of Lemma 4.5 | Lower bound on recalculation times).

Denote as $Z(\tau)$ the cost discrepancy:

$$Z_i(\tau) \triangleq C_i(\tau) - \hat{C}_i(\tau) = \int_{t_k^i}^{\tau} [\Lambda_i(\cdot) - \Lambda_i(\hat{\cdot})]dt \triangleq \int_{t_k^i}^{\tau} \Lambda_{Zi,k}(\tau),$$

where $\Lambda_{Zi,k}(\tau) \triangleq \Lambda_i(\mathbf{q}_i(\tau), \mathbf{u}_i^*(\tau)) - \Lambda_i(\hat{\mathbf{q}}_i(\tau), \hat{\mathbf{u}}_i^*(\tau))$.

The following conditions hold:

- $\Lambda_{Zi,k}$ is continuous in $\mathbf{q}_i$, $\mathbf{u}_i$,
- $\mathbf{q}_i$ is continuous in time,
- $|v_i|$ is continuous at all switching instants and
- ω_i is continuous in $[t_k^i, t_k^i + T)$.

Therefore $Z_i(\tau)$ is differentiable in $[t_k^i, t_k^i + T)$, with: $\dot{Z}_i(\tau) = \Lambda_{Zi,k}(\tau)$. Also note that $Z_i(\tau = t_k^i) = 0$, i.e. the discrepancy is zero at the beginning of the control phase. In addition, let $v_{max} = \max_\tau\{|v_i(\tau)|\}$. Thus $|v_i|$ is upper bounded by v_{max}. It is proven in Corollary 4.3 that $|v_i| \nrightarrow \infty \Rightarrow v_{max} \nrightarrow \infty$. The angular velocity ω_i is bounded, $||\mathbf{p}_i - \mathbf{p}_{id}|| < 2R_w$ and $\phi_i \in (-\pi, \pi]$. Thus $||\mathbf{q}_i - \mathbf{q}_{id}||$ is also bounded. Therefore

$$\exists \Lambda_{Zi,k}^{max} \geq \Lambda_{Zi,k}(\tau), \text{ s.t. } \dot{Z}_i^{max} = \Lambda_{Zi,k}^{max}.$$

Then the inter-event times satisfy $0 < c_\epsilon / \Lambda_{Zi,k}^{max} \leq t_{k+1}^i - t_k^i$. Finally, the $t_{k+1}^i - t_k^i < T$ side is trivially satisfied by the execution rule (4.12), since $T_c < T$ by default. ■

Proof A.4 (Proof of Corollaries 3.1, 4.1, and 4.3 | Finite linear velocity).

The projection of the field's gradient $\nabla_i \Phi_i$ on each agent's longitudinal axis, P_i can be written as $|P_i| = \left|\nabla_i \Phi_i^\top \cdot \mathbf{I}_i\right| = ||\nabla_i \Phi_i|| \cdot |\cos(\phi_i - \phi_{nhi})|$. For $||\nabla_i \Phi_i|| = 0$ to hold, agent i must have arrived at its destination $\mathbf{p}_i = \mathbf{p}_{id}$ [31]. For $|\cos(\phi_i - \phi_{nhi})| = 0$ to hold, the agent's longitudinal axis must be normal to the field's gradient $\nabla_i \Phi_i$, i.e., $|\phi_i - \phi_{nhi}| = \frac{\pi}{2}$, which contradicts $|\phi_i - \phi_{nhi}| \in [0, \frac{\pi}{2})$. Therefore $|P_i| \neq 0 \Rightarrow |v_i| \nrightarrow \infty$ in (2.21a). ■

Proof A.5 (Proof of Corollaries 3.2, 4.2, and 4.4 | Direction of motion).

The condition $d_i < 0$ substituted in the definition of ϕ_{nhi} results in:

$$\phi_{nhi} = atan2(-\Phi_{iy}, -\Phi_{ix}),$$

meaning that ϕ_{nhi} defines exactly the angle of $-\nabla_i \Phi_i$.

Initially, $P_i < 0$ therefore the agent's heading, ϕ_i, is initially in the direction of $-\nabla_i \Phi_i$, i.e., $\phi_i \in (\phi_{nhi} - \frac{\pi}{2}, \phi_{nhi} + \frac{\pi}{2})$. From Corollaries 3.1, 4.1, and 4.3, we have $P_i \nrightarrow 0$. Thus P_i remains negative and $-s_i = -sgn(P_i) > 0$ always holds. As a result the linear velocity in (2.21a) is always positive (forward motion). ■

The requirements of Corollaries 3.2, 4.2, and 4.4 are mild and represent reasonable physical conditions.

Proof A.6 (Proof of Theorem 3.2 | Single agent Collision Avoidance).

First of all, since the agent is considered spherical (disk), collisions can occur only by translation. Second of all, by definition, a Navigation Function is uniformly maximum on the boundary of obstacles, i.e., internal obstacles and the workspace boundary $\partial\mathcal{W}$. As a result, on the boundary of obstacles the negated gradient of a NF points away from

them. Assuming that at $t = 0$, the agent is safe, i.e., not involved in a collision, then $\Phi|_{t=0} < 1$. We will show that $\dot{\Phi} \leq 0$, $\forall t \geq 0$, which in turns implies that $\Phi(\mathbf{p}(t)) < 1$, $\forall t \geq 0$.

$$\dot{\Phi} = \nabla\Phi^{\top} \cdot \dot{\mathbf{p}} = \nabla\Phi^{\top} \cdot \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} v = -P\text{sgn}(P)U = -|P|\,U \leq 0$$

■

Proof A.7 (Proof of Theorems 4.2, and 4.5 | Collision Avoidance).

Since the agents are considered spherical (disks), collisions can occur only by translation. Thus, to ensure collision avoidance, it suffices to show that each agent i uses its linear velocity v_i (2.21a) to stay away from its neighbours. By definition, a Navigation Function is uniformly maximum on the boundary of obstacles, i.e., other agents and ∂W in the multi-agent setting. As a result, on the boundary of other agents the negated gradient of a DNF points away from them. It can be shown by (2.21a) that, for each agent i, the inner product $\nabla_i\Phi_i^{\top} \cdot \dot{\mathbf{p}}_i$ is non–positive. Specifically, we derive from (2.21a):

$$\nabla_i\Phi_i^{\top} \cdot \dot{\mathbf{p}}_i = \nabla_i\Phi_i^{\top} \cdot \begin{bmatrix} \cos\phi_i \\ \sin\phi_i \end{bmatrix} v_i \leq -P_i s_i U_i = -|P_i|\,U_i \leq 0 \tag{A.23}$$

Let us assume that a group of agents, which initially are sufficiently far apart from each other so that $\Phi_i|_{t=0} < 1$, $\forall i \in \mathcal{N}$, cause a collision. Since each Φ_i is continuous and differentiable *in space*, this would mean that at least one colliding agent i moved towards the direction of $\nabla_i\Phi_i$ causing the DNF Φ_i to attain its maximum value of 1. As shown in Eq. (A.23), this cannot be true and therefore no collisions can occur between agents under the control law (2.21a). ■

Proof A.8 (Proof of Theorem 3.3 | Single agent Convergence).

Consider the finite, strictly increasing sequence of recalculation time instants:

$$\pi := \{t_0, t_1, \ldots, t_k, t_{k+1}, \ldots, t_f\},\ k \in \mathbb{N},$$

where t_k denotes the time instant the agent executed Algorithm 1 (recalculation time), $t_{k+1} = t_k + T_c$, and $t_0 \triangleq 0$. The last element in π, t_f, denotes the time instant at which the agent entered the neighbourhood r_0 of its destination.

On each time interval $[t_k, t_{k+1})$, $t_k \neq t_f$, we employ the following Lyapunov function candidate:

$$V_k = \Phi + \frac{1}{2}(\phi - \phi_{nh} - \theta^*[t_k, t_{k+1}))^2. \tag{A.24}$$

We consider the extended system $\dot{\mathbf{x}}_\mathbf{s} = f(\mathbf{x}_\mathbf{s})$:

$$\mathbf{x}_\mathbf{s} = \begin{bmatrix} \mathbf{p}^\top & \phi & \phi_{nh} & \theta^* \end{bmatrix}^\top, \tag{A.25}$$

$$f(\mathbf{x}_\mathbf{s}) = \begin{bmatrix} v \cdot \mathbf{I}^\top & \omega & \dot{\phi}_{nh} & \dot{\theta}^* \end{bmatrix}^\top. \tag{A.26}$$

We can now follow Proof A.10, modulo the summations over the N agents, to show that V is strictly decreasing and that the agent converges to the singleton set:

$$\{\mathbf{q} : (\mathbf{p} = \mathbf{p}_d) \wedge (\phi = \phi_d)\},$$

i.e., the agent converges to its target destination with the desired orientation. ■

Proof A.9 (Proof of Theorem 4.3 | Convergence of the multi–agent system).

Consider the finite, strictly increasing sequence of recalculation time instants:

$$\pi := \{t_0, t_1, \ldots, t_k, t_{k+1}, \ldots, t_f\},\ k \in \mathbb{N},$$

where t_k denotes the time instant the centralized authority executed Algorithm 3 (recalculation time), $t_{k+1} = t_k + T_c$, and $t_0 \triangleq 0$. The last element in π, t_f, denotes the time instant at which the *last* agent entered the neighbourhood r_0 of its destination.

On each time interval $[t_k, t_{k+1})$, $t_k \neq t_f$, we employ the following Lyapunov function candidate:

$$V_k = \sum_{i=1}^{N} V_{ik},\ V_{ik} = \Phi_i + \frac{1}{2}(\phi_i - \phi_{nhi} - \theta_i^*[t_k, t_{k+1}))^2. \tag{A.27}$$

We consider the extended multi-agent system $\dot{\mathbf{x}} = f(\mathbf{x})$:

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}_1^\top & \ldots & \mathbf{p}_N^\top & \phi_1 & \ldots & \phi_N & \phi_{nh1} & \ldots & \phi_{nhN} & \theta_1^* & \ldots & \theta_N^* \end{bmatrix}^\top, \tag{A.28}$$

$$f(\mathbf{x}) = \begin{bmatrix} v_1 \cdot \mathbf{I}_1^\top & \ldots & v_N \cdot \mathbf{I}_N^\top & \omega_1 & \ldots & \omega_N & \dot{\phi}_{nh1} & \ldots & \dot{\phi}_{nhN} & \dot{\theta}_1^* & \ldots & \dot{\theta}_N^* \end{bmatrix}^\top. \tag{A.29}$$

We can now follow Proof A.10 to show that V is strictly decreasing and that the multi-system converges to the singleton set:

$$\{\mathbf{q} : (\mathbf{p}_i = \mathbf{p}_{id},\ \forall i \in \mathcal{N}) \wedge (\phi_i = \phi_{id},\ \forall i \in \mathcal{N})\},$$

i.e., all agents converge to their target destinations with the desired orientation. ■

Proof A.10 (Proof of Theorem 4.6 | Convergence of the multi–agent system).

Consider the finite, strictly increasing (because of Lemma 4.5) sequence of recalculation time instants:

$$\pi := \{t_0, t_1, \ldots, t_k, \ldots, t_f\},\ k \in \mathbb{N},$$

where t_k denotes the time instant *at least one* agent triggered an execution of Algorithm 3 (recalculation time) or entered the neighbourhood r_0 of its destination and $t_0 \triangleq 0$. The last element in π, t_f, denotes the time instant at which the *last* agent entered the neighbourhood r_0 of its destination. The sequence of recalculation times of an agent $i \in \mathcal{N}$ is a subset of π, $\{0, t_1^i, ..., t_k^i, ..., t_f^i\} \subseteq \pi$.

On each time interval $[t_k, t_{k+1})$, $t_k \neq t_f$, we employ the following Lyapunov function candidate:

$$V_k = \sum_{i=1}^{N} V_{ik},\ V_{ik} = \Phi_i + \frac{1}{2}(\phi_i - \phi_{nhi} - \theta_i^*[t_k, t_{k+1}))^2. \tag{A.30}$$

We consider the extended multi-agent system $\dot{\mathbf{x}} = f(\mathbf{x})$:

$$\mathbf{x} = \left[\, \mathbf{p}_1^\top \ \ldots\ \mathbf{p}_N^\top\ \phi_1\ \ldots\ \phi_N\ \phi_{nh1}\ \ldots\ \phi_{nhN}\ \theta_1^*\ \ldots\ \theta_N^* \,\right]^\top, \tag{A.31}$$

$$f(\mathbf{x}) = \left[\, v_1 \cdot \mathbf{I}_1^\top\ \ldots\ v_N \cdot \mathbf{I}_N^\top\ \omega_1\ \ldots\ \omega_N\ \dot{\phi}_{nh1}\ \ldots\ \dot{\phi}_{nhN}\ \dot{\theta}_1^*\ \ldots\ \dot{\theta}_N^* \,\right]^\top. \tag{A.32}$$

In order to apply the chain rule in [45] , we employ the Filippov set [47] $K[f(\mathbf{x})]$ and the generalized derivative [42] of $V_k(\mathbf{x})$:

$$K[f] = \begin{bmatrix} K[v_1]\mathbf{I}_1 \\ \vdots \\ K[v_N]\mathbf{I}_N \\ \omega_1 \\ \vdots \\ \omega_N \\ \dot{\phi}_{nh1} \\ \vdots \\ \dot{\phi}_{nhN} \\ \dot{\theta}_1^* \\ \vdots \\ \dot{\theta}_N^* \end{bmatrix}, \qquad \partial V_k = \begin{bmatrix} \sum_i \nabla_1 \Phi_i \\ \vdots \\ \sum_i \nabla_N \Phi_i \\ (\phi_1 - \phi_{nh1} - \theta_1^*) \\ \vdots \\ (\phi_N - \phi_{nhN} - \theta_N^*) \\ -(\phi_1 - \phi_{nh1} - \theta_1^*) \\ \vdots \\ -(\phi_N - \phi_{nhN} - \theta_N^*) \\ -(\phi_1 - \phi_{nh1} - \theta_1^*) \\ \vdots \\ -(\phi_N - \phi_{nhN} - \theta_N^*) \end{bmatrix}$$

We calculate the generalized time derivative of $V_k(\mathbf{x})$:

$$\begin{aligned}
\dot{\tilde{V}}_k &= \bigcap_{\xi \in \partial V} \xi^\top K[f] = \\
&= \sum_i^N \sum_j^N K[v_i] \nabla_i \Phi_j^\top \mathbf{I}_i + \sum_i^N \omega_i \cdot (\phi_i - \phi_{nhi} - \theta_i^*) \\
&\quad - \sum_i^N \dot{\phi}_{nhi} \cdot (\phi_i - \phi_{nhi} - \theta_i^*) - \sum_i^N \dot{\theta}_i^* \cdot (\phi_i - \phi_{nhi} - \theta_i^*) = \\
&= \sum_i K[v_i] P_i + \sum_i \sum_{j \neq i} K[v_j] \nabla_j \Phi_i^\top \mathbf{I}_j + \sum_i (\phi_i - \phi_{nhi} - \theta_i^*) \cdot \left(\omega_i - \dot{\phi}_{nhi} - \dot{\theta}_i^* \right) = \\
&= \sum_i K[v_i] P_i + \sum_i \sum_{j \neq i} K[v_j] \nabla_j \Phi_i^\top \mathbf{I}_j - \sum_i k_\phi \left(\phi_i - \phi_{nhi} - \theta_i^* \right)^2,
\end{aligned}$$

We discriminate between the following two sets of agents:

$$\begin{aligned}
\mathcal{N}_1 &\triangleq \left\{ i \in \mathcal{N} \,\middle|\, \frac{\partial \Phi_i}{\partial t} \leq U_i (|P_i| - \epsilon) \right\} \\
\mathcal{N}_2 &\triangleq \left\{ i \in \mathcal{N} \,\middle|\, \frac{\partial \Phi_i}{\partial t} > U_i (|P_i| - \epsilon) \right\},
\end{aligned}$$

with $\mathcal{N}_1 \bigcup \mathcal{N}_2 = \mathcal{N}$. By the control law (2.21a) we deduce:

$$K[v_i] = \begin{cases} -K[s_i] \cdot U_i, & i \in \mathcal{N}_1 \\ -K[s_i] \cdot \dfrac{\frac{\partial \Phi_i}{\partial t} + \epsilon U_i}{|P_i|}, & i \in \mathcal{N}_2 \end{cases}$$

We can now proceed with $\dot{\tilde{V}}_k$:

$$\begin{aligned}
\dot{\tilde{V}}_k &= \sum_{\mathcal{N}_1} \left\{ -K[s_i] P_i U_i + \frac{\partial \Phi_i}{\partial t} \right\} = \\
&= \sum_{\mathcal{N}_2} \left\{ -K[s_i] P_i \frac{U_i \epsilon + \frac{\partial \Phi_i}{\partial t}}{|P_i|} + \frac{\partial \Phi_i}{\partial t} \right\} - \sum_{\mathcal{N}} k_\phi \cdot (\phi_i - \phi_{nhi} - \theta_i^*)^2 = \\
&= \sum_{\mathcal{N}_1} \left\{ -|P_i| U_i + \frac{\partial \Phi_i}{\partial t} \right\} + \sum_{\mathcal{N}_2} \left\{ -|P_i| \frac{U_i \epsilon + \frac{\partial \Phi_i}{\partial t}}{|P_i|} + \frac{\partial \Phi_i}{\partial t} \right\} - \sum_{\mathcal{N}} k_\phi \cdot (\phi_i - \phi_{nhi} - \theta_i^*)^2 = \\
&= -\sum_{\mathcal{N}_1} \left\{ |P_i| U_i - \frac{\partial \Phi_i}{\partial t} \right\} - \sum_{\mathcal{N}_2} U_i \epsilon - \sum_{\mathcal{N}} k_\phi \left(\phi_i - \phi_{nhi} - \theta_i^* \right)^2 < 0
\end{aligned}$$

We deduced that $0 \notin \dot{\tilde{V}}_k$ because in $\mathcal{N}_1$, $|P_i|\, U_i - \frac{\partial \Phi_i}{\partial t} \geq U_i \epsilon$, and for both $\mathcal{N}_1, \mathcal{N}_2$, we have that $\sum_{\mathcal{N}} U_i > 0$ because $\mathbf{p}_i \neq \mathbf{p}_{id}$ holds for at least one agent since $t < t_f$.

Now consider a recalculation time $t_k \in \pi$. For the subset of agents $j \in \mathcal{N}$ that triggered at $t = t_k = t_k^j$, we have

$$\theta_j^*[t_{k-1}^j, t_k^j) = \theta_j^*[t_k^j, t_{k+1}^j) \Rightarrow \theta_j^*[t_{k-1}^j, t_k) = \theta_j^*[t_k, t_{k+1}^j),$$

The same holds if $t_k = t_f^j$, as proven in Lemma 4.4.

Using this result in V_k we get: $V_{k-1}(t_k) = V_k(t_k)$, i.e., the multiple Lyapunov functions [41] V_k are equal at $t = t_k$. It is easy to show that the same also holds for $t = t_f$. Finally, let $V_f = \sum_{i=1}^{N} \Phi_i + \frac{1}{2}(\phi_i - \phi_{nhi} - \theta_i(t \geq t_f))^2$. Using the same analysis, we deduce that $0 \in \dot{\tilde{V}}_f$. This is possible since for $t > t_f$, all agents have entered the neighbourhoods r_0 of $\mathbf{p}_{id}$, making $U_i = 0$ possible $\forall i \in \mathcal{N}$. Therefore the concatenation $V = V_0|V_1| \ldots |V_k| \ldots |V_f$ is a continuous, strictly decreasing Lyapunov function for the multi-agent system [41, p. 53].

Since each V_{ik}, and consequently V, is regular and the level sets of V are compact, we apply the nonsmooth version of LaSalle's invariance principle. Thus the multi-agent system converges to the largest invariant subset $S : S \triangleq \left\{ \left[\mathbf{p}^\top, \phi\right]^\top \mid 0 \in \dot{\tilde{V}} \right\}$. For $\dot{\tilde{V}} = 0$ to hold, we get:

$$\begin{aligned} S = \{\mathbf{q} :& (|P_i|\, U_i - \frac{\partial \Phi_i}{\partial t} = 0,\ \forall i \in \mathcal{N}_1) \wedge \\ & \wedge (U_i \epsilon = 0,\ \forall i \in \mathcal{N}_2) \wedge (\phi_i - \phi_{nhi} - \theta_i^* = 0,\ \forall i \in \mathcal{N})\}. \end{aligned}$$

Since $|P_i|\, U_i - \frac{\partial \Phi_i}{\partial t} \geq U_i \epsilon \geq 0$ (condition of $\mathcal{N}_1$), the equality $U_i = 0$ must hold inside S, requiring $\mathbf{p}_i = \mathbf{p}_{id}$ so that[4] $\phi_{nhi} = \phi_{id}$ and $\theta_i^* = 0$ (by construction), $\forall i \in \mathcal{N}$. Thus $\phi_i = \phi_{id}$ $\forall i \in \mathcal{N}$. Therefore S reduces to the singleton:

$$\{\mathbf{q} : (\mathbf{p}_i = \mathbf{p}_{id},\ \forall i \in \mathcal{N}) \wedge (\phi_i = \phi_{id},\ \forall i \in \mathcal{N})\},$$

i.e., all agents converge to their target destinations with the desired orientation. ■

The tools from Discontinuous Feedback and Nonsmooth Analysis used in the derivation of this proof are provided in Appendix A.4.

[4] This is a property of dipolar DNFs; see §A.1.3.

Appendix B
Software Structure

This Appendix elaborates on the implementation of the proposed predictive navigation schemes in MATLAB simulations. For the sake of brevity, only the simulation code for the Decentralized Event-Triggered Predictive Navigation scheme will be presented. The single agent and centralized multi-agent cases are much simpler and can be derived by simplifying the decentralized case.

Firstly, we provide some of the MATLAB variables and structures used in our simulations. Secondly, we list all MATLAB scripts and functions employed. Finally, we provide the structure of `Main.m`, which is the main routine of our code.

B.1 Definition of MATLAB Variables

We list the most important variables and structures used in our MATLAB simulations.

`ragent`, `Rworld` (variables): r_i, R_w.

`NFdata` (struct): Contains all parameters required for the definition of a Navigation Function (k, λ, h etc). For example, `NFdata.k` is the NF parameter k.

`alpha`, `delta`, `Tpred`, `Tctrl`, `Ns` (variables): $\alpha, \delta, T, T_c, N_s$.

`Unom`, `kf`, `ro` (variables): U_{id}, k_ϕ, r_0.

`agents` (struct): Contains all agents' state vectors and other variables. For example, `agents(iagent).p` is agent i's position vector $\mathbf{p}_i$ and `agents(iagent).fi` is agent i's orientation ϕ_i. In addition, `agents().history(,)` stores all agents' state history $\mathbf{q}(t)$.

`agents().Vq`, `dVdq`, `dfdt`: Φ_i, $\nabla_i \Phi_i$ and $\dfrac{\partial \Phi}{\partial t}$.

`th-optimum()`, `th-dot()`, `fnh`, `fnh-dot` (variables): $\theta_i^*[t_k + T)$, $\dot{\theta}_i$, ϕ_{nh} and $\dot{\phi}_{nh}$.

`u`, `omega` (variables): Control input vector $[u\ \omega]^\top$.

`mpc` (struct): Contains all agents' *predicted* state and inputs trajectories over $[t_k, t_k + T)$.

`Q`, `Ru`, `Rw`, `Li` (variables): Q, R_{11}, R_{22}, and running cost $\Lambda(\cdot)$.

`Jfunct`, `Mterm` (variables): Cost functional $J(\cdot)$ and terminal cost $M(t+T) = \Phi_i(t+T)$.

`trigger()` (variable): Monitors the triggering condition in (4.12).

`check` (variable): Checks for convergence of the multi-agent system.

B.2 MATLAB Routines and Functions

B.2.1 List of Routines and Functions

`Main.m` (script)
The main MATLAB routine. This is the m–file that should be executed (Run). It initializes most variables and structs, calls all other scripts (m-files) below, and also contains the main prediction–control loop.

`Data.m` (script)
Contains all constant parameters, i.e., those used by `NF.m`, `Prediction.m`, and other scripts and functions.

`Agents.m` (script)
Agents' initial and goal configurations are set in this scripts.

`Prediction.m` (script)
This scripts executes Algorithm 3. It also calls `PredNavigation.m` to simulate the dynamics of the multi-agent system over $[t_k, t_k + T)$.

`PredNavigation.m` (script)
Performs Step 5 of Algorithm 3.

`CtrlNavigation.m` (script)
Applies the actual control inputs to each agent.

`NF.m` (function)
Calculates the dipolar DNF's value Φ_i and its gradient $\nabla_i \Phi_i$.

`Lfunct.m` (function)
Calculates the running cost.

`thita1.m` (function)
Performs Step 4 of Algorithm 3, i.e., generates candidate deviations (1^{st} degree polynomials) over $[t_k, t_k + T)$.

`Plots.m` (script)
Uses the values stored in `agents.history` and other variables to generate all figures.

`Kyklos.m`, `crtriangle.m` (functions)
Called by `Plots.m` to create the circles and the red arrows (triangles) respectively.

B.2.2 Structure of Main.m

Without getting into unnecessary details, the structure of `Main.m` is summarized in the following algorithm.

Main.m: Decentralized Event-Triggered Predictive Navigation

1: Call `Data.m` and `Agents.m`

2: `Initialize` (preallocate) all structures and vectors

3: `while(check)` % Start main loop

4: `for i = 1:N` % For each agent $i \in \mathcal{N}$

5: Check triggering condition

6: Call `Prediction.m` % Execute Algorithm 3 over $[t_k^i, t_k^i + T)$

7: Call `CtrlNavigation.m` % Apply control inputs over $[t_k^i, t_{k+1}^i)$

8: `end for loop`

9: `end while loop` % if `check`$= 0$, exit main loop

10: Call `Plots.m`
