Rigid Formation Control using Hovercrafts

A Spatial Model Predictive Control Approach

Avinash Siddaramappa

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Rigid Formation Control using Hovercrafts

A Spatial Model Predictive Control Approach

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Avinash Siddaramappa

November 23, 2015

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis is a part for the project, "Amphibian" conducted by Department of Maritime and Transportation Technology, TU Delft.

Abstract

The steady increase in volume of goods to be transported over land and water has necessitated the development of efficient and alternative methods of transportation. In current practices considerable amount of transportation delay is caused due to switching between different modes of transportation resulting in economic losses. Another difficulty in the transportation sector arises when handling extremely large objects, such as wind turbines and cargo containers. In order to overcome these difficulties, an alternative method of transportation has been proposed recently. The proposed alternative method consists of the development of a formation control framework using hovercrafts. Formation control is a widely researched topic due its potential benefits in reducing the system cost, structure flexibility and improving the efficiency of the overall system.

State-of-the-art methods for formation control are focused towards developing a controller to track a predefined trajectory, and the trajectory generation is an additional problem which is addressed off-line. A path tracking formation control framework is an area which has received less attention in the literature and can have possible improvements. Hence, this thesis work answers the research question of generating a feasible path for a set of initial and final conditions, and tracking the generated path.

Model Predictive Controller (MPC) is a promising framework, because of its constraint handling capabilities. Hence, it accounts for the major component of this thesis. A new framework for formation control is proposed which uses spatial-domain parametrization instead of the conventional time-domain parametrization. MPC being a computationally expensive task, the proposed framework reduces the number of decision variables and constraint evaluations which can help to achieve the desired Real Time (RT) performance.

The specific research question addressed in this thesis work is to develop a path tracking formation control framework while minimising the absolute and relative position error for each vehicle in the formation. A nonlinear dynamic model for the hovercraft is obtained using first principles and its structure is used to design an optimization based controller. An Optimal Control Problem (OCP) is formulated which is solved using direct collocation method. In this method a continuous problem is discretized into a number of collocation points, and the resulting problem is solved using state-of-the-art Nonlinear Program (NLP) solvers such

as, Sparse Nonlinear OPTimizer (SNOPT) and Nonlinear Programming Systems Optimization Laboratory (NPSOL). The performance of the proposed framework is illustrated using numerical simulations.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my sincere gratitude to my MSc thesis supervisor dr.ir. Tamás Keviczky, Associate Professor, Delft Center for Systems and Control for his continuous support and valuable guidance during the execution of the thesis.

I would also like to thank Per Rutquist, Senior developer, Tomlab optimization for his enthusiastic support in gaining deeper understanding of the software.

Finally, I thank all those who made this journey a pleasant experience.

Delft University of Technology
November 23, 2015

Avinash Siddaramappa

Dedicated to my Appa and Amma

Chapter 1

Introduction

Formation control is an important issue in coordinated control for a group of autonomous vehicles. In formation control, a group of autonomous vehicles are required to follow a predefined trajectory while maintaining a desired spatial pattern. Over conventional systems, moving in formation has many advantages such as, reduced system cost, higher efficiency and structure flexibility [1].

Formation control is widely applied in search and rescue missions, security patrol, transportation of large loads, etc. In automated highways, the capacity of the transportation network is greatly increased if vehicles move at a desired velocity, while maintaining a specific distance between them [2]. Formation control has found its importance even in military missions, where a group of autonomous vehicles are required to maintain different patterns to attain the best surveillance of an area and reduce the fuel consumption [3].

Transportation is one of those sectors where formation control has shown its competency. In present scenario of transporting large loads, a multiple wheeled heavy truck is used which is large enough to transport the load individually. However, this comes at a cost of increased fuel consumption. Using multiple smaller vehicles increases the efficiency and also has an advantage of flexibility based on the structure of the load [4], [5], [6] .

1-1 Aim and motivation

The steady increase in volume of goods that are being transported has necessitated the development of efficient and alternative methods of transportation. In the present transportation scenario, waterways account for a major part due to its inexpensiveness. However, in order to meet the requirement of the end customer, switching between water and land is inevitable. This switch causes a great transportation delay, resulting in economic losses. Researchers in the transportation sector are examining every possible way to reduce this delay in order to satisfy their basic working principle i.e. *Time is Money*.

The Department of Maritime and Transportation Technology, TU Delft has proposed an alternative method of transportation using hovercrafts. Being amphibious vehicle, hovercrafts

can reduce the delay caused due to switching between land and water transports. Apart from their amphibious nature, these vehicles have an advantage of being adaptable, cost effective, and environmentally friendly [7]. However, operating them is a challenging task as the vehicle always maintains a thin layer of air between itself and the surface of motion. While this diminishes the friction, the drawback is that lack of friction causes lateral slip. Apart from lateral slip, the vehicle also has restrictions on the braking capabilities. When many such vehicles are used for transportation of cargo, meeting the desired safety requirements becomes a challenge. This calls for development of an autonomous vehicle to make the system accident free and to make the best use of available capacity.

The next major difficulty is the transportation of large loads such as, huge cargo container or wind turbine blades. As these wind turbines are installed at remote locations(offshore beds or hilly areas), it is infeasible to use large and heavy vehicles for their transportation. In the present scenario, human-controlled multiple vehicles are driven simultaneously in formation to carry the load to its destination. As mentioned previously, when hovercrafts are used similarly, controlling them becomes a challenge. Hence, this necessitates the development of a formation control framework using autonomous hovercraft vehicles.

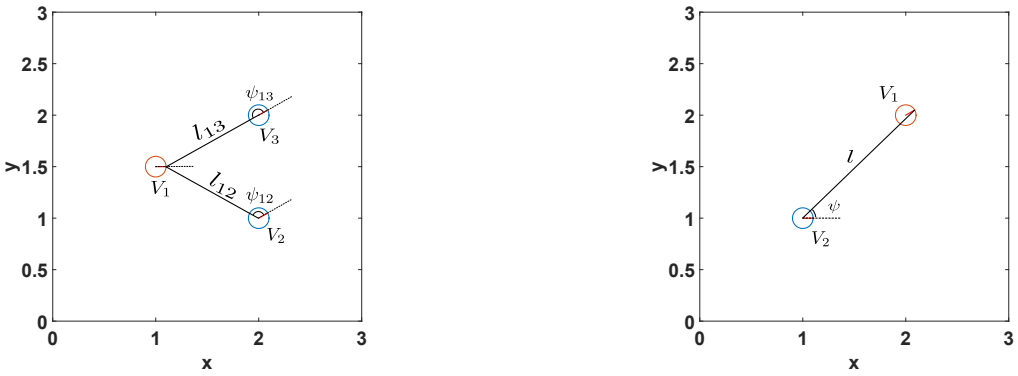
The research work in this thesis aims at providing a framework for formation control using hovercrafts which will be useful in transportation of large loads. Hovercrafts are nonlinear, underactuated and nonholomic systems making the control problem a challenging task. Moreover, formation control further restricts the motion of each vehicle making the problem complex and challenging.

1-2 State-of-the-art methods

Formation control using multiple vehicles is widely investigated in literature. The approaches proposed in the literature study can be roughly classified as *leader-follower*, *virtual-structure* and *potential field approach*. In this section, state-of-the-art methods that exist are reviewed. Based on the findings of the literature survey, a problem statement for this thesis has been formulated.

Leader-follower method

The leader-follower pattern is a widely recognized method in formation control [8]. As the name suggests, in this method a vehicle termed as *leader* tracks a trajectory while the rest of the vehicles in formation, termed as *followers* maintain a desired distance and relative angle with respect to their leader thus maintaining the desired formation. There are two types of controllers viz. $l-l$ and $l-\psi$. In $l-l$ controller the objective of follower V_1 is to maintain desired lengths l_{12} and l_{13} between its position and its two leaders V_2 and V_3. In case of $l-\psi$ the objective is to maintain a desired length and relative angle with respect to the leader as shown in Figure 1-1. Applications of the leader-follower approach using feedback linearization and optimization based controllers are found in [9] and [10] respectively.



(a) $l - l$ controller **(b)** $l - \psi$ controller

Figure 1-1: Leader-follower method

Virtual-structure method

In the leader-follower method, all agents may have a single leader or each vehicle may have its own unique leader. In the case that a leader fails, then its followers will no longer be able to maintain their positions, and in turn their followers, and so on. This chain dependency makes the system highly fault intolerant. This called for development of the virtual-structure method [11]. This method addresses the vulnerability by defining the leader in software. As the leader is not subjected to any physical failure, this increases the robustness of the overall system. It is often termed as the 'virtual-leader' method. Application of the virtual-structure method using geometric control in decentralised mode and optimization based control in centralised mode is seen in [12] and [13] respectively.

Potential field method

Potential field method transforms the whole horizontal plane of motion into an artificial potential function. In the resulted artificial potential function, the initial position of robots are depicted as *peaks* and destination as *valley*. Due to this each robot tend to get attracted towards the *valley*, while ensuring collision avoidance(as the position of each robot is a peak). Application of this method to stabilise the formation is demonstrated in [14]. In [15], the social potential field method is extended and evaluated in the presence of agent failure and imperfect sensory input.

1-3 Research objectives

From state-of-the-art methods presented in the previous section it is seen that, all existing methods aim at developing a controller to track a predefined trajectory, and the trajectory generation is an additional problem which is addressed off-line. A path tracking formation control framework is an area which has recieved less attention in literature and can have possible improvements. Hence, this thesis work answers the research question of generating a feasible path for a set of initial and final conditions, and tracking the generated path.

As seen in the previous section, both nonlinear, and optimization based controllers are capable of achieving the formation control. However, using nonlinear controllers may result in unrealistic control inputs causing constraint violations. Apart from constraint violation for path tracking application nonlinear controllers require an additional path-trajectory converter. In case of optimization methods, both path-trajectory converter and feedback control is solved in an unified manner. Another added advantage is the constraint handling capability of optimization based methods.

Both the centralised and the decentralised methods have their own advantages and disadvantages. In decentralised method it is simple to include additional agents in to the framework. However, the major drawback of using decentralised method is, it demands for complex processors on each of its agents. Whereas, in the case of centralised method it requires a single processor capable of providing solutions to all agents. The major drawback of centralised method is the computational requirements increase as the number of agents increase.

Based on the discussion above, this thesis has the following research goals.

- Investigate a formation control framework for path tracking applications.
- Design an optimization based controller in a centralised manner.
- Explore the use of state-of-the-art fast optimization solvers which will enable the implementation of Model Predictive Controller (MPC) for hovercrafts.
- Analyse the scalability of the proposed method compared to the conventional time-domain parametrization.

1-4 Organization of the Report

In this chapter the motivation for a rigid formation control using hovercraft is explained. State-of-the-art methods for formation control are briefly explained and the problem statement for this thesis is formulated. The remaining part of the report is organised as follows.

The modelling of a single hovercraft is explained in Chapter 2. The equations of motion for hovercrafts are obtained using the first principles, and the modelling assumptions are illustrated.

The framework for a path tracking formation control is explained in Chapter 3. The chapter starts with explaining the algorithm used to generate an optimal path for a given set of initial and final conditions. A new solution for path tracking application using spatial-domain is introduced and its need is motivated. Based on the introduced spatial domain a suitable Optimal Control Problem (OCP) is formulated and the choice of its tuning parameters are motivated. The new framework aims at reducing the number of decision variables in the resulting MPC. The chapter ends by comparing the outcome of the proposed framework with the existing methods.

After explaining the theoretical aspects of the controller, Chapter 4 shows the competency of the proposed method with the help of numerical simulation studies. Performance of different Nonlinear Program (NLP) fast solvers are shown and the significance of different sections in

the framework are explained. The chapter ends by exhibiting the results of different test cases for a typical formation control problem such as, starting from different initial conditions, and vehicle failure analysis.

Report finally concludes with Chapter 5 where the main research findings of the thesis are discussed, and recommendations for the improvement of the framework is proposed.

The path generating algorithm used to generate an optimal paths and the method used to solve the formulated OCP are explained in Appendices A and B, respectively.

Chapter 2

Modelling of Hovercraft

The first and foremost requirement in the design, analysis and simulation of a model based control strategy is to have a reliable and acceptable description or the model of the system. The system under consideration can be seen as a rigid body, actuated by different propellers and rudders. The complete dynamics of such a system is complicated and it would be difficult to model them accurately considering all the external forces and disturbances(for example, the aerodynamic forces acting on the hovercraft, the external disturbance due to wind, etc). Even if an accurate model of such a system is obtained, it is too complicated and huge in terms of the number of states and inputs for the controller design. Therefore, it is interesting to consider a simplified model which retains the most important dynamics of the system, and a model with minimum number of states and inputs.

In Section 2-1, each component present in a hovercraft is illustrated and its working principle is explained. As the states of the system are represented in to different frames the kinematic relations between different frames are derived to convert variables from reference to frame of interest. The equations of motion for a hovercraft is derived using Euler-Lagrange method by making some modelling assumptions which is explained in Section 2-2.

2-1 Working principle

The motivation to propose hovercrafts for logistics is its capability to travel both over land and water, thus reducing the transportation delay. It is the construction of the vehicle which makes it possible to be amphibious. In this section, the working of a hovercraft is explained, which will aid in understanding the mathematical model derived in further sections.

A hovercraft has two different propellers viz. lift and thrust. As the name suggest, lift propeller is used to lift the vehicle slightly off the ground, and maintain a thin layer of air between the vehicle and plane of motion. The thrust propeller is used to move the vehicle forward. The lift propeller blows air underneath the craft, which is contained by the skirt. A skirt is a long flexible material mounted on the perimeter and underneath the craft with holes at the centre. The skirt serves the purpose of maintaining the air cushion. When the

lift propeller blows air into the skirt, it inflates the skirt and excess air gushes out from holes present underneath the vehicle. This escaping air coming from the bottom holes creates a frictionless cushion of air.

The thrust propeller generates a force which propels the hovercraft forward. As it blows air towards the rear end of the vehicle, it creates a thrust in the opposite direction and the craft moves forward. This propeller is placed at the rear end of the vehicle.

The rudder is placed behind the thrust propeller and rotates along its vertical axis. When the rudder makes no angle with the longitudinal axis of the vehicle, the effective thrust is utilised in moving the vehicle forward. But when it makes a non zero angle with longitudinal axis, only a part of thrust is utilized in moving the hovercraft forward and rest hits the face of rudder and gets deflected, causing an equivalent torque in opposite direction. This torque gradually rotates the vehicle as it moves. Figure 2-1 shows a typical hovercraft model along with its components. The orientation, or the heading direction of the hovercraft is known as *yaw* and angular velocity is known as *yaw rate*. Due to negligible friction between vehicle and plane of motion, it slips laterally at the time of rotation. As this motion is not actuated, the system is *underactuated*. This is the major difference in dynamics of a wheeled vehicle and a hovercraft.

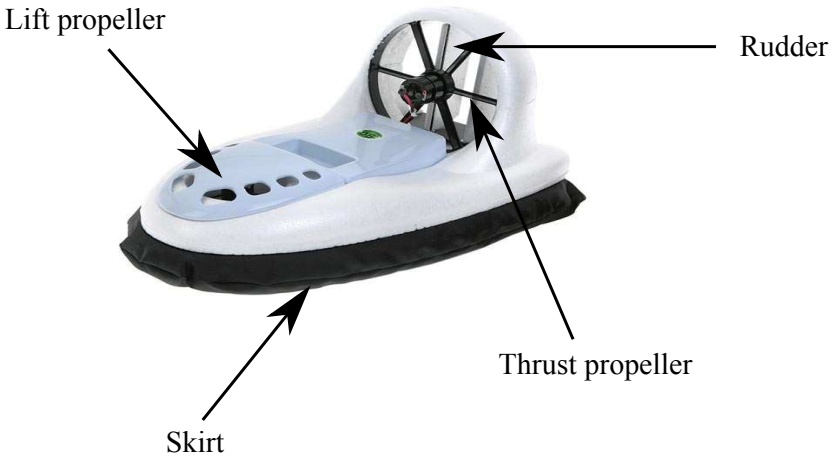


Figure 2-1: Hovercraft model

2-2 Hovercraft model

As a hovercraft cannot move vertically, its position is depicted in the global x-y coordinate. Apart from position, another important variable is the heading of vehicle i.e. *yaw* angle. Hence, it is modelled as a body with 3 Degrees of Freedom (DoF) with planar x-y position and *yaw* angle. In order to depict the vehicle's states, two different coordinate frames are defined viz. inertial $\{I\}$ and body-fixed $\{B\}$ frame. The inertial frame is a reference frame which remains unchanged, and all the measurements are made with respect to this particular coordinate frame. In this case earth-fixed reference frame is considered to be inertial [16]. The body-fixed frame is attached with the body and the frame proceeds as the vehicle moves. The origin of body-fixed frame lies on the Center of Gravity (CoG) of the body, which is

depicted as (x_G, y_G). Figure 2-2 shows two different coordinate frames associated with the hovercraft.

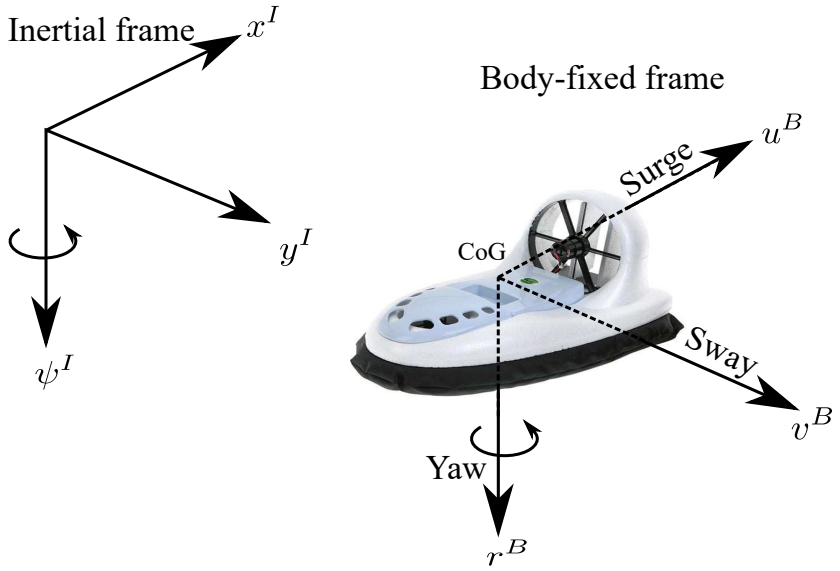


Figure 2-2: Body-fixed and Inertial reference frames

2-2-1 Kinematic relations

As mentioned in the previous section, two different frames of reference are used to depict the vehicle's states. This calls for a relation which can convert variables from its reference frame to the frame of interest. In order to do so, kinematics, which is a branch of mechanics that studies the motion of a body without considering the forces and torques acting on it is used. In this section, the kinematic relations pertaining to the hovercraft are presented which will be used to obtain the equations of motion.

Position and orientation of the hovercraft are represented by $\left(x^I, y^I, \psi^I\right)$ and are described relative to inertial reference frame. Linear and angular velocities are represented by $\left(u^B, v^B, r^B\right)$ that are relative to body-fixed reference frame. Based on this general notations, states of the system are described by the following vectors,

$$\eta^I = \left[x^I, y^I, \psi^I\right]^T$$
$$\mathcal{V}^B = \left[u^B, v^B, r^B\right]^T$$

where η^I is the position and orientation vector, and $\mathcal{V}^B$ is the velocity vector. Position and orientation of the vehicle in inertial reference frame is depicted using x^I, y^I, and ψ^I respectively. *Surge* and *sway* velocities are depicted by u^B and v^B respectively, and r^B depicts *yaw* rate represented in body-fixed reference frame.

Velocities expressed in body-fixed frame are transformed to inertial frame as,

$$\dot{\eta}^I = J(\psi^I)\mathcal{V}^B$$
$$\begin{bmatrix} \dot{x}^I \\ \dot{y}^I \\ \dot{\psi}^I \end{bmatrix} = \begin{bmatrix} \cos(\psi^I) & -\sin(\psi^I) & 0 \\ \sin(\psi^I) & \cos(\psi^I) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u^B \\ v^B \\ r^B \end{bmatrix}$$

where $J(\psi^I)$ is the rotational matrix [17].

2-2-2 Equation of motion

Equations of motion for a hovercraft can be obtained using two widely known methods, viz. Newton-Euler, and Euler-Lagrange method. Both methods result in an equivalent set of equations and both have their merits. For simpler systems, Newton-Euler method is the preferred choice as it is easy and intuitive. However, as the complexity of the system increases, it becomes difficult to apply Newton-Euler method and this is where the Euler-Lagrange method has its advantages [18]. Based on this, Euler-Lagrange method is used to derive equations of motion for the hovercraft vehicle.

The Lagrangian approach involves three basic steps. First, suitable expressions for the vehicle's kinetic and potential energy are formulated. The Lagrangian L is then computed as,

$$L = T - V$$

where T and V are system's kinetic, and potential energy respectively. Finally, the following Lagrangian equation is applied.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\eta}^I}\right) - \frac{\partial L}{\partial \eta^I} = \tau \tag{2-1}$$

where τ represents the control input.

In case of hovercraft there is no potential energy, as it always travels on the same plane. Thus, the Lagrangian consists of only the kinetic energy, T which is written as,

$$L = T$$
$$L = \frac{1}{2}\mathcal{V}^{B^T} M \mathcal{V}^B \tag{2-2}$$

Where M is mass and inertia matrix with $M = \text{diag}(m,\ m,\ I_z)$. m represents mass of the vehicle and I_z is the moment of inertia around z axis.

The final step in deriving the equations of motion is to apply the Lagrangian equation, (2-1). However, deriving the equations of motion from the Lagrange's equation in terms of quasi-coordinates instead of the standard Lagrange's equation is simpler [19]. This is written as,

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \mathcal{V}^{B1}}\right) + \tilde{\mathcal{V}}^{B2}\frac{\partial L}{\partial \mathcal{V}^{B1}} = \tau_1 \tag{2-3a}$$
$$\frac{d}{dt}\left(\frac{\partial L}{\partial \mathcal{V}^{B2}}\right) + \tilde{\mathcal{V}}^{B2}\frac{\partial L}{\partial \mathcal{V}^{B2}} + \tilde{\mathcal{V}}^{B1}\frac{\partial L}{\partial \mathcal{V}^{B1}} = \tau_2 \tag{2-3b}$$

where $\mathcal{V}^{B1}$ is the linear velocity vector, $\left[u^B, v^B, 0\right]^T$ and $\mathcal{V}^{B2}$ is the angular velocity vector, $\left[0, 0, r^B\right]^T$. τ_1 and τ_2 represent thrust and torque inputs respectively. Lastly, $\tilde{\mathcal{V}}^{B1}$ and $\tilde{\mathcal{V}}^{B2}$ are matrices defined as,

$$\tilde{\mathcal{V}}^{B1} = \begin{bmatrix} 0 & 0 & v^B \\ 0 & 0 & -u^B \\ -v^B & u^B & 0 \end{bmatrix}, \tilde{\mathcal{V}}^{B2} = \begin{bmatrix} 0 & -r^B & 0 \\ r^B & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2-4}$$

Accordingly (2-2) is written as,

$$L = \frac{1}{2}\mathcal{V}^{B1T} M_1 \mathcal{V}^{B1} + \frac{1}{2}\mathcal{V}^{B2T} M_2 \mathcal{V}^{B2} \tag{2-5}$$

where M_1 and M_2 are the mass and inertia matrices respectively,

$$M_1 = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & 0 \end{bmatrix}, M_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{2-6}$$

On substituting (2-4), (2-5) and (2-6) in (2-3), the following equations are obtained.

$$\begin{aligned} m\dot{u}^B - mv^B r^B &= \tau_1 \\ m\dot{v}^B + mu^B r^B &= 0 \\ I_z \dot{r}^B &= \tau_2 \end{aligned}$$

The above equations depict the motion of vehicle due to control inputs τ_1 and τ_2, neglecting external noise such as wind gusts. However, as the vehicle moves, some of the force is dissipated due to air resistance and friction between surface of motion and skirt. These dissipation terms are incorporated in the final equations of motion. The static friction is modelled in terms of *surge*, *sway*, and moment (as the amount of static friction is much higher than air resistance). Hence, the final equations of motion for the hovercraft are given in (2-7).

$$\dot{u}^B = v^B r^B - \frac{d_u}{m} u^B + \frac{\tau_1}{m} \tag{2-7a}$$

$$\dot{v}^B = -u^B r^B - \frac{d_v}{m} v^B \tag{2-7b}$$

$$\dot{r}^B = -\frac{d_r}{I_z} r^B + \frac{\tau_2}{I_z} \tag{2-7c}$$

where d_u, d_v, and d_r are coefficients of static and rotational friction respectively [20].

2-2-3 Modelling assumptions

A hovercraft is an underactuated, nonholonomic and nonlinear system. It is not possible to consider all external disturbances while modelling the system. Even if an accurate model is obtained, it would be too complicated and unmanageable in terms of number of states and inputs for the system. Hence, the following assumptions are made to obtain a simple model, yet encapsulate all the major nonlinearities of the system,

- Constant lift is available throughout the run.
- Structure of the vehicle is rigid and symmetric.
- Origin of body-fixed frame coincides with CoG.
- All vehicles in the formation are similar.
- Air resistance and aerodynamic disturbances are negligible.

2-3 Summary

In this chapter, modelling aspects of the hovercraft are discussed. The steps taken to derive the equations of motion for the hovercraft is explained, and the derived model is based some modelling assumptions that are also illustrated. Next step is to design a controller extracting the model, the framework for formation control and the design procedure of the controller is explained in the next chapter.

Chapter 3

Formation control design

In the previous chapter, the equations of motion for hovercrafts are derived. Once a reliable model for the vehicle is obtained, the next step is to design a model based controller with path tracking capabilities. Before designing the controller a trivial question that needs to be answered is, 'How to reach the destination given a set of initial and final conditions?' An open solution is to track a straight line joining the points. However, as discussed in the previous chapter, the vehicle has nonholonomic constraints making the path generation a non-trivial problem. Apart from nonholonomic constraints, the vehicle also has restrictions in moving backwards, as the reverse movement of vehicle is not actuated. Hence, this calls for a feasible path generation algorithm to generate a path for a given set of initial and final conditions, and design a controller to track the resulting path. In the proposed framework, for a given set of initial and final conditions, a feasible path is generated off-line and the designed controller tracks this path. Figure 3-1 shows the block diagram of framework for formation control.

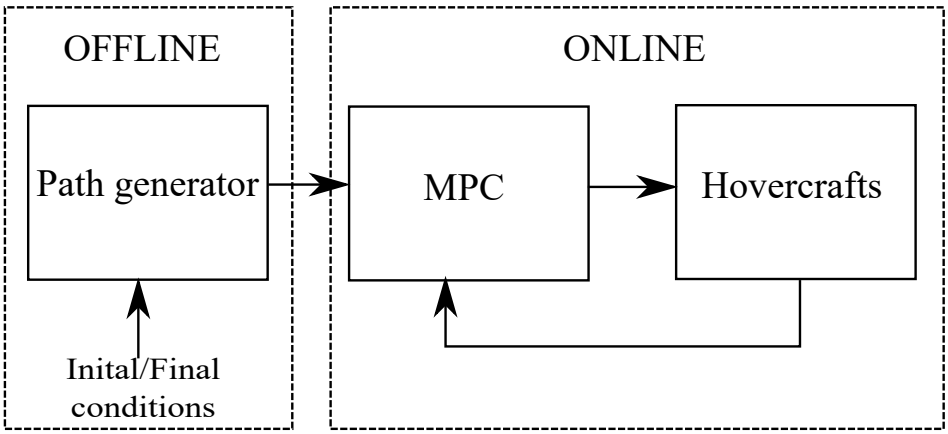


Figure 3-1: Formation control framework

In Section 3-1 the basic aim of the control design is explained. The framework is introduced in Section 3-2 by defining the reference formation in a suitable way. After defining the formation, feasible paths for different vehicles in the formation are generated in Section 3-3. Path

tracking controller requires a new coordinate frame to represent the vehicle's position, this is motivated and in Section 3-4, and the steps taken to reformulate the system dynamics to the new coordinate frame are explained. Based on the new coordinate frame, an Optimal Control Problem (OCP) is formulated in Section 3-5. The choice of different tuning parameters of the controller is motivated in Section 3-6 and a comparison of the new formation control framework with existing method is made. Finally a brief summary is provided in Section 3-7.

3-1 Objectives of the formation control framework

The formation control framework is aimed in fulfilment of the following objectives.

- Generate a dynamically feasible path for hovercrafts, for a given set of initial and final conditions.
- Design a Model Predictive Controller (MPC) to attain formation control for the Non-linear Time Invariant model of hovercraft vehicle that tracks the generated path.
- The primary objectives of formation control framework is to track the generated path while minimising the absolute and relative position error in formation. The framework should be capable of keeping the error within the bounds of 5 cm.
- Minimise the computational burden, by reducing the number of decision variables and constraints evaluation in MPC.
- Comparison of the proposed framework with the existing time-domain parametrized methods.

3-2 Description of the reference formation

Before generating an optimal path, it is helpful to define the formation in terms of vectors. Any formation in x-y plane consisting of n number of vehicles is represented by a vector joining the central vehicle and each member vehicle in formation. Let us call this central vehicle as V_c. This central vehicle is a virtual member in addition to member vehicles in formation. Hence, the overall number of vehicles are $n+1$. With n number of real vehicles represented as V_1, V_2, ..., V_n and an additional virtual vehicle, V_c.

The position of virtual vehicle V_c is obtained with respect to the positions of each member vehicle with the help of the following equations.

$$V_{cx} = \frac{1}{n}\left(V_{1x} + V_{2x} + \cdots + V_{nx}\right)$$
$$V_{cy} = \frac{1}{n}\left(V_{1y} + V_{2y} + \cdots + V_{ny}\right)$$

where V_{nx} and V_{ny} represent x and y coordinate of vehicle V_n, respectively.

Using the position of all vehicles, n number of vectors are constructed, each joining the member and virtual vehicle. These vectors are represented as $\vec{V}_{1c}$, $\vec{V}_{2c}$, ..., $\vec{V}_{nc}$. Figure 3-2

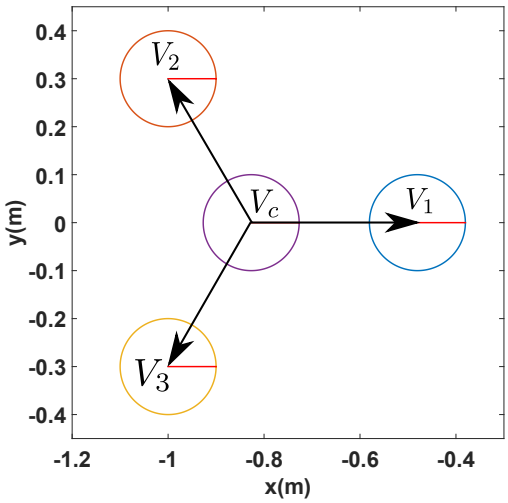


Figure 3-2: Depiction of formation as vectors

shows a triangular formation using three member vehicles and a virtual vehicle, where the red line represents vehicle's heading.

Based on the position of each member vehicle in formation relative to the virtual vehicle, these vehicles are categorised as, F, B, L and R, which is explained in Algorithm 1. This categorisation is used in further sections.

Algorithm 1 Vehicle position

Input: $\vec{V}_{kc}$

if $\angle\vec{V}_{kc} > 0$ AND $\angle\vec{V}_{kc} < \pi$ **then**
  $V_{LR,k} = L$
else
  if $\angle\vec{V}_{kc} > \pi$ AND $\angle\vec{V}_{kc} < 2\pi$ **then**
    $V_{LR,k} = R$
  else
    $V_{LR,k} = C$
  end if
end if
if $\angle\vec{V}_{kc} > 3\pi/2$ AND $\angle\vec{V}_{kc} < \pi/2$ **then**
  $V_{FB,k} = F$
else
  if $\angle\vec{V}_{kc} > \pi/2$ AND $\angle\vec{V}_{kc} < 3\pi/2$ **then**
    $V_{FB,k} = B$
  else
    $V_{FB,k} = C$
  end if
end if $\forall k \in \{1, 2, \ldots, n\}$

Output: $V_{LR,k}, V_{FB,k}$

3-3 Path generation

In order to design a path tracking formation control, the first step is to generate a feasible path for a given set of initial and final conditions. In this section, path generation algorithm suitable for a hovercraft is discussed. To generate a path between a given set of initial and final conditions, many techniques are available in literature such as, Balkcom-Mason, Reeds-Shepp and Dubin's curves [21]. As discussed earlier, a hovercraft not only has nonholonomic constraints, but also restrictions in reverse motion. Out of the above mentioned techniques, Balkcom-Mason curves are suitable for the systems with differentiable drives, such as cars. Hence, this technique is not suitable for hovercraft. The major drawback in using Reeds-shepp curves is that the algorithm assumes the vehicle to be capable of moving in reverse direction. Dubin's path algorithm not only includes all constraints on motion of the hovercraft, but also includes a constraint on minimum turning radius, thus, generating the shortest, and a feasible path. Hence, Dubin's path generating algorithm is chosen for path generation.

Dubin's path algorithm is used to generate the reference path for the virtual vehicle. The reference paths for member vehicles in formation are generated with respect to the virtual vehicle's path. This is explained in the following subsections.

3-3-1 Reference path for the virtual vehicle

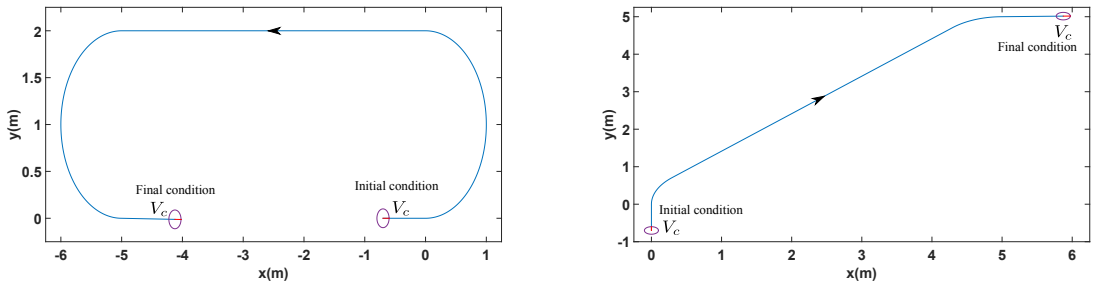
Dubin's path consists of three sections viz. *Curve*, *Straight*, and again a *Curve*. The curves are part of a circle with radius $r_{\min}$, and the *straight* section is a tangent line common to both the curves. One end of the first curve coincides with the given initial condition where as, one end of the second curve coincides with the given final condition. The other end of both curves are part of tangent line i.e. *Straight* part of the curve. Thus, a vehicle tracking the path takes a turn to follow the curve section, then travels in a straight line and again takes a turn to reach the required final condition.

Based on direction of turning, Dubin's path is subdivided into, LSL, RSR, LSR and RSL. With L, S and R depicting *Left curve*, *Straight* and *Right curve* sections of the path, respectively. As the path includes minimum number of curves and these curves have the radius of curvature equivalent to maximum turning capabilities of the vehicle, making it the shortest path for a given set of initial and final conditions. Procedure to generate these paths are illustrated in Appendix A.

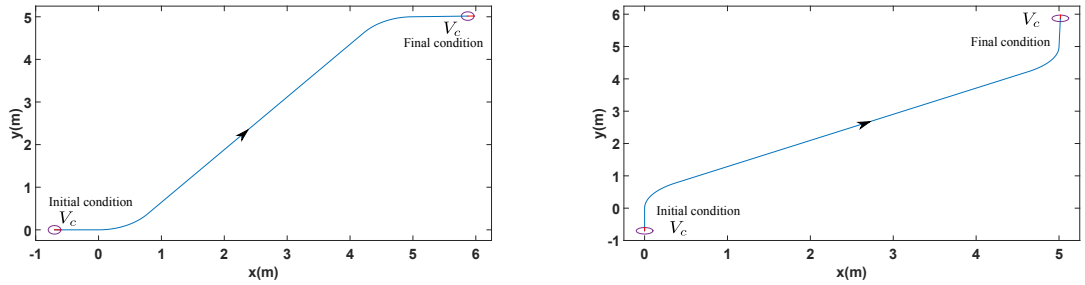
In order to make it suitable for formation control, Dubin's path is modified by appending straight lines at both ends. Motivation behind the modification is explained in Section 3-4-2. Modified Dubin's path is the reference for virtual vehicle and generating reference paths for member vehicles in formation is explained in next subsection. Figure 3-3 shows the modified Dubin's path and its variations.

3-3-2 Reference path for the member vehicles

Reference paths for the member vehicles in formation is generated by sweeping the virtual vehicle, V_c and the vectors $\vec{V}_{1c}$, $\vec{V}_{2c}$, ..., $\vec{V}_{nc}$ along the reference path generated in previous subsection. The path swept by these vectors are the reference paths for the member vehicles in the formation. These paths in x-y plane is given by equations in Algorithm 2.



(a) LSL **(b)** RSR



(c) LSR **(d)** RSL

Figure 3-3: Variations in modified Dubin's path

Algorithm 2 Reference path for member vehicles

Input: $V_{FB,k}$

if $V_{FB,k} = F$ OR $V_{FB,k} = C$ **then**

$V_{kx} = V_{cx} + |\vec{V_{kc}}| \cos\left(\theta + \angle\vec{V_{kc}}\right)$

$V_{ky} = V_{cy} + |\vec{V_{kc}}| \sin\left(\theta + \angle\vec{V_{kc}}\right)$

else

if $V_{FB,k} = B$ **then**

$V_{kx} = V_{cx} - |\vec{V_{kc}}| \cos\left(\theta + \left(\pi - \angle\vec{V_{kc}}\right)\right)$

$V_{ky} = V_{cy} - |\vec{V_{kc}}| \sin\left(\theta + \left(\pi - \angle\vec{V_{kc}}\right)\right)$

end if

end if $\forall k \in \{1, 2, \ldots, n\}$

Output: V_{kx}, V_{ky}

Where θ is the heading angle of V_c. Figure 3-4 shows the reference paths for vehicles in formation for a triangular formation.

3-4 Reformulation from temporal to spatial-domain

Designing a path tracking controller is a challenging task. This is because, the path is a set of points in x-y plane. Whereas, trajectory is the path parametrised in time based on system

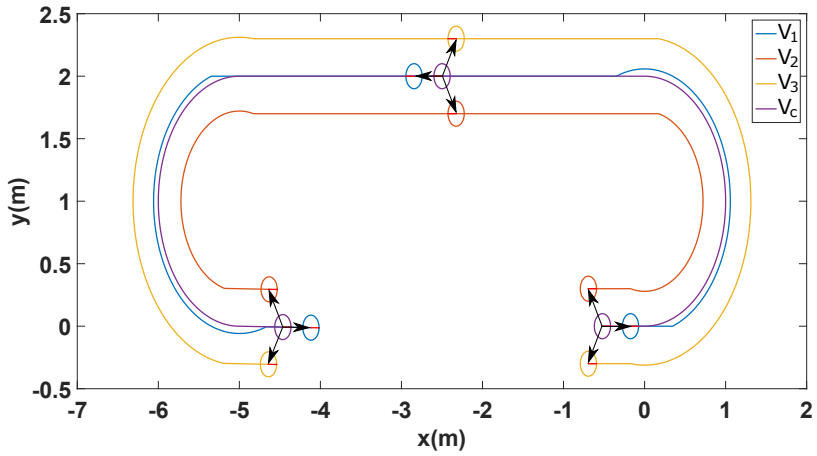


Figure 3-4: Reference paths for vehicles in formation

dynamics i.e. $(x(t), y(t))$. Hence, a resulting MPC for trajectory tracking application predicts the time dependent position of each vehicle till the prediction horizon and minimises the error between the predicted position and the reference trajectory. In the same way, a path tracking MPC should predict the position of each vehicle for a certain distance(prediction horizon) and minimise the error between the predicted position and the reference path. A controller should predict the position of each vehicle parametrized by the path curvature and its heading. In order to do so vehicles dynamics are reformulated from temporal to spatial/path domain [22]. More precisely, instead of using time as an independent variable, the progression of vehicle along the path is used as an independent variable. The major advantage of doing this is, the road information is then exactly available over the prediction horizon which is not the case in temporal domain. This reformulation is explained in the following subsections.

3-4-1 Modelling of vehicle position in spatial-domain

In the present scenario, the position of vehicle is represented in terms of its x and y coordinates. In order to represent the position in path domain, two new variable are introduced viz. s representing the progress of vehicle along the path and d^S representing the lateral deviation/offset from the path. Using these two new variables the position of vehicle is represented in the path coordinated frame $\left(s, d^S\right)$.

Switching from inertial to path coordinate frame is done by constructing a Frenet frame [23] as shown in Figure 3-5. At any position along the path s, a Frenet frame is constructed with tangent and normal vectors, $\hat{\vec{T}}(s)$ and $\hat{\vec{N}}(s)$ receptively. These vectors are given by the following equations.

$$\vec{T}(s) = \frac{\partial^2 f_{\text{path}}(s)}{\partial s}, \quad \hat{\vec{T}}(s) = \frac{\vec{T}(s)}{||\vec{T}(s)||_2}$$
$$\vec{N}(s) = \frac{\partial^2 f_{\text{path}}(s)}{\partial s^2}, \quad \hat{\vec{N}}(s) = \frac{\vec{N}(s)}{||\vec{N}(s)||_2}$$

Of these two vectors, $\vec{T}(s)$ is always tangential to the path at any given point s. This vector is used to find heading of the path at any point s. In contrast with $\vec{T}(s)$, at any given point s the normal vector $\vec{N}(s)$ always points lateral to the path. Hence, this vector is used in finding the offset between the path and any point in the plane.

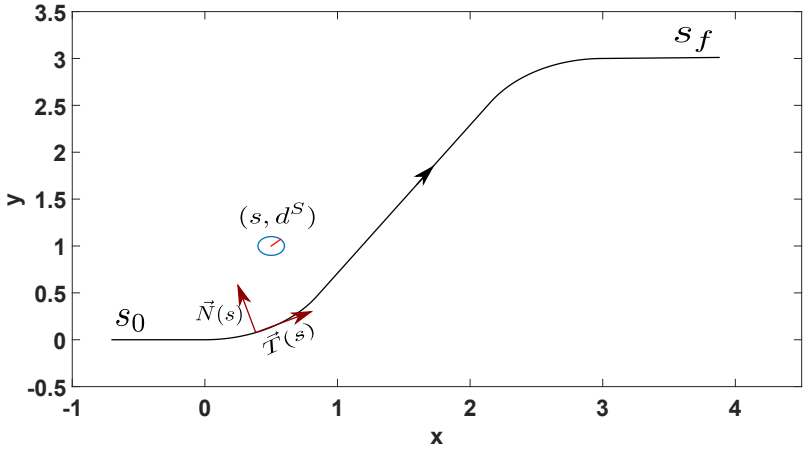


Figure 3-5: Reconstruction of hovercraft position using frenet frame

Heading of the path, $\psi_r(s)$ is defined as the angle between global x axis and the tangent vector, and curvature of the path $k(s)$ is defined as magnitude of normal vector as given in the following equations.

$$\begin{aligned} \psi_r(s) &= \arccos\left(\hat{\vec{T}}(s), \begin{bmatrix}1\\0\end{bmatrix}\right) \\ k(s) &= ||\vec{N}||_2 \end{aligned} \tag{3-1}$$

Let us consider a vehicle V_1 positioned at (x_1, y_1) as shown in Figure 3-6. Its position can be written in path coordinate frame by projecting a line l normal to the path. This line has its one extreme end on (x_1, y_1) and other on the path at point s. This line (with length d^S) is always the shortest line between the point (x_1, y_1), and the path. Hence, any point in x-y plane can be represented in path coordinate as $\left(s, d^S\right)$.

The position of vehicle can be transformed back to inertial frame using the following equation.

$$\begin{bmatrix} x^I \\ y^I \end{bmatrix} = f_{\text{path}}(s) + ||\vec{N}||_2 d^S$$

However, using the local path coordinate frame to represent vehicle's positions has some drawbacks. These drawbacks are,

- $\left(s, d^S\right)$ need not be unique for all (x, y)
- For some points in (x, y), the corresponding $\left(s, d^S\right)$ may not exist

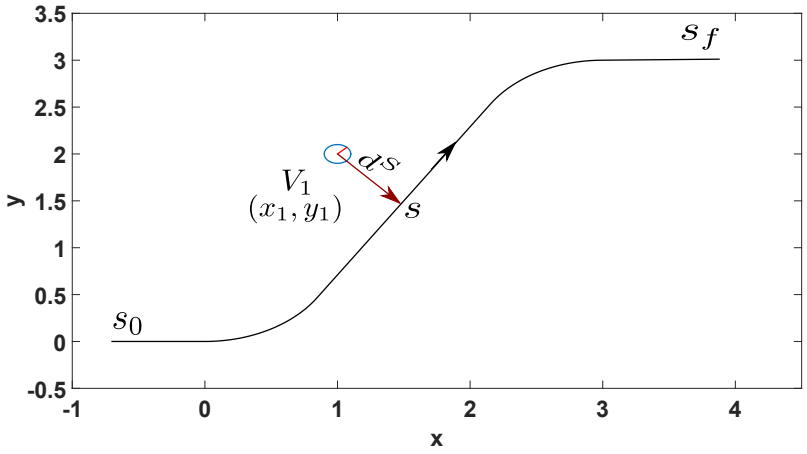
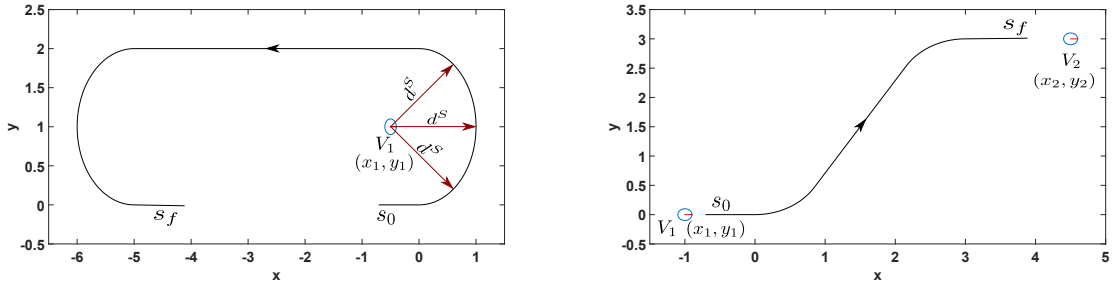


Figure 3-6: Vehicle position in spatial-domain

Figure 3-7 shows the drawbacks of using path coordinate to represent the vehicle's positions. The first drawback is shown in Figure 3-7a. Here, a vehicle V_1 is present at the centre of *curve* section of the path. This position is equidistant from all the points on the *curve*. Hence there is no unique representation in spatial domain for this particular position. The representation of the vehicle's position using path domain inherently depends on the path length. Hence, when a vehicle is present before the starting of path or after the end of path, then there is no valid representation for these particular positions in path domain. This drawback is shown in Figure 3-7b.



(a) Drawback 1 **(b)** Drawback 2

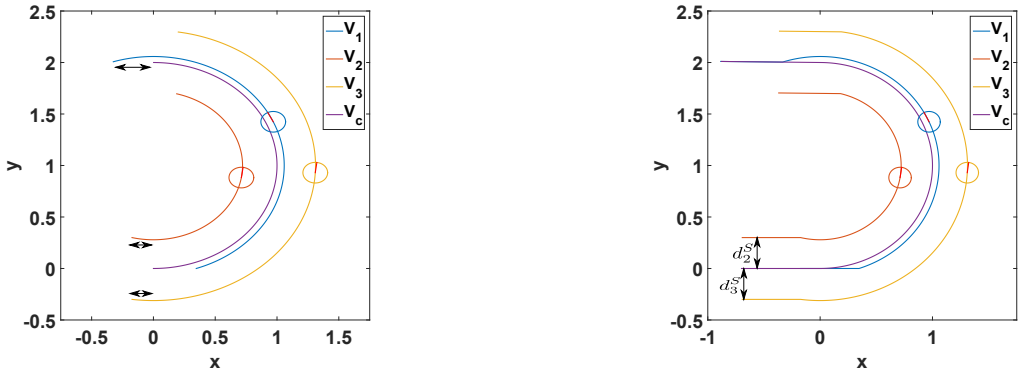
Figure 3-7: Drawbacks of using spatial coordinates

3-4-2 Reference paths in spatial-domain

Reference paths generated in Section 3-3 are now transformed to the spatial domain. In spatial domain, the reference path for the virtual vehicle V_c is simply the path length of Dubin's path (s) with zero offset, d^S. Whereas, for member vehicles in formation, it is the path length, s and an offset with respect to Dubin's path i.e. the shortest line between the point and Dubin's path.

As pointed out in the previous subsection, one of the drawback of using spatial domain is

that, for some points in (x, y) coordinates, the corresponding $\left(s, d^S\right)$ in spatial domain may not exist. Most importantly for the points located before the start of the path and after the end of the path. This drawback hinders in converting the paths to spatial domain. For a typical triangular formation shown in Figure 3-2, V_2 and V_3 are always behind V_c, and V_1 is always ahead of V_c. Hence, for sections of V_2 and V_3's paths that are before the starting of V_c path the resulting spatial coordinate will not exist. In the same way for the section of V_1's path that is after V_c's path there is no valid representation in spatial domain. This is shown in Figure 3-8a. In order to overcome this problem, the Dubin's path is modified by appending straight lines at both ends, for a length of distance of extreme vehicles in formation. This modification is further extended by distance d which will act as a runway to accelerate from zero to a common velocity (u^B_{ref}) and decelerate to zero at the end of run.



(a) Conventional paths **(b)** Modefied paths

Figure 3-8: Modification in Dubin's path

Using the modified Dubin's path, reference paths for each vehicle in formation is transformed into path domain according to the method specified previously. Figure 3-9 shows the variation in lateral offset for a triangular formation following typical LSL path. This path consist of a *left curve* at path length 0.72 m and *straight* section at 3.84 m and again a *left curve* at path length 6.85 m, and finally a *straight* section at 9.99 m. From Figure 3-9, the following important observations can be noted when the path is switching from *straight* to *curve*,

- For vehicle V_1, the offset d^S changes smoothly initially and has discontinuous point at the end of the switch.
- For vehicles V_2 and V_3, the offset d^S has the discontinuous point initially and changes smoothly.
- For a triangular, even though both the vehicles V_2 and V_3 are displaced by same distance, the lateral offset for V_3 is higher in the *curve* section compared to that of V_2

This is because of the position of vehicles in formation. As V_1 is always ahead of V_c, while shifting from *straight* to *curve* section the point of minimum distance varies smoothly according to *curve* section of Dubin's path and the discontinuous point is because of the non smooth reference path. Where as, in the case of V_2 and V_3 which are behind V_c the discontinuous point

is again because of non smooth reference path of V_2 and V_3 and gradually varies according to the *curve* section of their respective paths.

The difference in lateral offset for V_2 and V_3 in *curve* section is because of the reference path which turns towards *Left*. As the vehicle V_2 lies inside the *curve*, the point of minimum distance is nearer to Dubin's path compared to V_3, which lies outside the curve. This difference increases as the curvature of path increases.

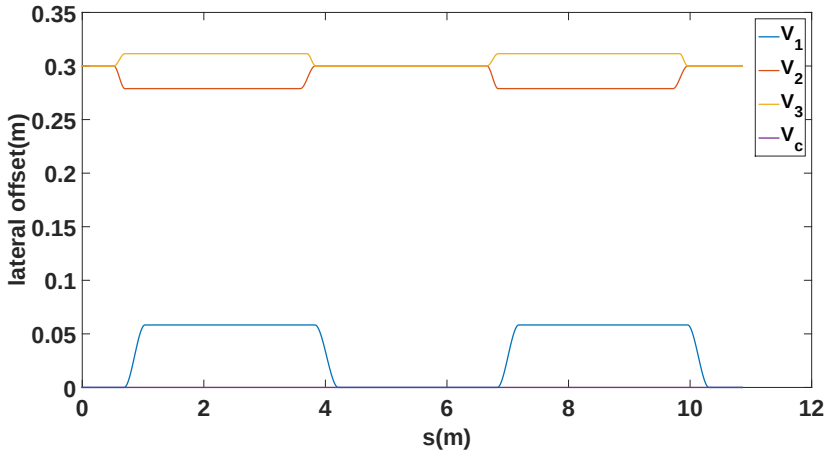


Figure 3-9: Lateral offset d^S for different vehicles in formation

3-4-3 Spatial reformulation of vehicle dynamics

After representing vehicle's position and the reference paths in local path coordinate, the next step is to reformulate the system dynamics from temporal to spatial/path domain. Figure 3-10 shows the vehicle's motion according to path coordinate. From Figure 3-10, the linear velocity of the vehicle projected along the path is given as,

$$\begin{aligned} u_s &= (\rho - d^S)\dot{\psi}_r \\ &= u^B \cos(e^S_\psi) - v^B \sin(e^S_\psi) \end{aligned} \tag{3-2}$$

where ρ is the radius of curvature of path, d^S is the perpendicular offset from the path, e^S_ψ is the error in heading and $\dot{\psi}_r$ is time derivative of heading of the path.

Using (3-2) the dynamic relations for progression of vehicle along the path s and perpendicular offset, d^S is written as,

$$\begin{aligned} \dot{s} &= \rho\dot{\psi}_r \\ &= \left(\frac{1}{1-\frac{d^S}{\rho}}\right) u^B \cos(e^S_\psi) - v^B \sin(e^S_\psi) \\ \dot{d}^S &= u^B \sin(e^S_\psi) + v^B \cos(e^S_\psi) \end{aligned} \tag{3-3}$$

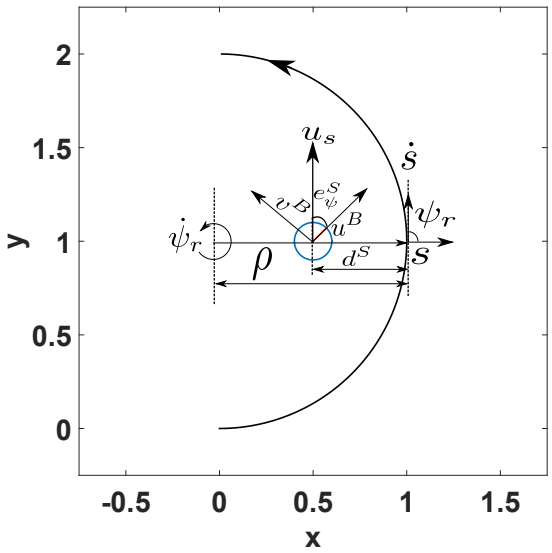


Figure 3-10: Spatial reformulation of vehicle dynamics

Equations of motion for hovercraft derived in Section 2-2-2 is written in vector form as,

$$\begin{aligned}\frac{d}{dt}\xi &= f(\xi,\mathcal{U})\\ \xi &= \left[x^I, y^I, \psi^I, u^B, v^B, r^B\right]^T\\ \mathcal{U} &= [\tau_1, \tau_2]^T\end{aligned}$$

The temporal model of the hovercraft can be reformulated to spatial-domain by the following equation.

$$\frac{d}{ds}\xi = \frac{d}{dt}\xi\frac{dt}{ds}$$

Substituting (3-3) in the above equation results in,

$$\begin{aligned}\frac{d}{ds}\xi &= \frac{f(\xi,\mathcal{U})}{u^B\cos(e_\psi^S) - v^B\sin(e_\psi^S)}\left(1 - d^S k\right)\\ \xi &= \left[u^B, v^B, r^B, d^S, e_\psi^S\right]^T\\ \mathcal{U} &= [\tau_1, \tau_2]^T\end{aligned}$$

where k is the curvature of path $\left(\frac{1}{\rho}\right)$. Hence, the equations of motion for hovercraft in

spatial-domain is written in (3-4),

$$\frac{du^B}{ds} = \frac{v^B r^B - B_1 u^B + \tau_1/m}{u^B \cos(e^S_\psi) - v^B \sin(e^S_\psi)} \left(1 - d^S k\right) \tag{3-4a}$$

$$\frac{dv^B}{ds} = \frac{-u^B r^B - B_2 v^B}{u^B \cos(e^S_\psi) - v^B \sin(e^S_\psi)} \left(1 - d^S k\right) \tag{3-4b}$$

$$\frac{dr^S}{ds} = \frac{-B_3 r^B + \tau_2/I_z}{u^B \cos(e^S_\psi) - v^B \sin(e^S_\psi)} \left(1 - d^S k\right) \tag{3-4c}$$

$$\frac{dd^S}{ds} = \frac{u^B \sin(e^S_\psi) + v^B \cos(e^S_\psi)}{u^B \cos(e^S_\psi) - v^B \sin(e^S_\psi)} \left(1 - d^S k\right) \tag{3-4d}$$

$$\frac{de^S_\psi}{ds} = \frac{r^B \left(1 - d^S k\right)}{u^B \cos(e^S_\psi) - v^B \sin(e^S_\psi)} - k \tag{3-4e}$$

3-5 Formulation of the OCP

The general task of driving a vehicle is natural to humans. After a certain period of learning, drivers are able to handle the vehicle without conscious effort. Some drivers tend to control the vehicle with aggressive manoeuvres, while some others tend to be more gentle in controlling the vehicle. In the case of autonomous vehicles, they do not have a learning behaviour. Every single run is completely new, without any memory.

Designing a controller for formation control is even more challenging task. This is because, even if a single vehicle fails to achieve the required performance, the whole formation gets distorted, deformation is highly undesirable in the case of formation control.

Formation control can be achieved by both nonlinear controllers and MPC. For a path tracking formation control, MPC has a major advantage over nonlinear controller. In case of nonlinear controller, a separate path-trajectory converter is to be designed and the resulting trajectory is tracked by the controller. Where as, in the case of MPC, the problem of trajectory generation as well as the feedback controller are addressed simultaneously by formulating an OCP. This OCP is solved in receding horizon fashion to achieve formation control. However, the major disadvantage of using MPC is that, as the number of vehicles increase, the number of decision variables increase, there by increasing the computational burden. This hinders in achieving the Real Time (RT) performance to the best possible accuracy. An attempt is made to formulate an OCP which reduces the number of decision variables in formation control, and guarantee RT performance. The steps followed in formulating a suitable OCP are explained in further subsections.

3-5-1 Generic OCP formulation

The objective of the control problem is to track a path with specific velocity profiles by finding the control inputs for each vehicle in the formation. This is attained by formulating an OCP which deals with the problem of finding a control law for a given system such that a certain optimality criterion is achieved. A control problem includes a cost function to be minimised,

which is a function of state and control variables. In order to solve the problem, it should satisfy basic constraints. Typically these are the system dynamics and bounds on system state and control input.

A typical OCP in mathematical form is written as,

$$\begin{aligned} \min_{\mathcal{U}_i} \quad & \int_{s_0}^{s_f} \mathcal{L}\left(\xi_i(s), \mathcal{U}_i(s)\right) \\ \text{subject to} \quad & \frac{d\xi_i}{ds} = f(\xi_i, \mathcal{U}_i) \\ & \xi_i(0) = \xi_{i0} \\ & \underline{\mathcal{U}} \leq \mathcal{U}_i \leq \overline{\mathcal{U}} \\ & \underline{\xi} \leq \xi_i \leq \overline{\xi} \quad \forall i \in \{1, 2, \ldots, n+1\} \end{aligned}$$

where $\mathcal{L}\left(\xi_i(s), \mathcal{U}_i(s)\right)$ is the cost function to be minimised while satisfying system dynamics, initial conditions and bounds on states and control inputs. Notice that the cost function is integral over distance travelled by a vehicle along the path, but not over time. Hence, this involves the vehicle dynamics formulated along the path which is already done in the previous section. In an eventual MPC, the resulting OCP is solved at every iteration with the initial conditions of states represented as, ξ_{i0}.

The resulting OCP is solved using direct collocation method explained in Appendix B. In this method, the continuous problem is discretized to a finite dimensional problem, termed as collocation points n. Meaning, the OCP is solved only at these collocation points. For the states and control inputs between these collocation points, they are simply interpolated using either cubic interpolation or splines [24]. By doing so, computational burden in solving the problem is reduced by a greater margin. After discretizing the problem, resulting Nonlinear Program (NLP) is solved using fast solvers such as, Sparse Nonlinear OPTimizer (SNOPT), Nonlinear Programming Systems Optimization Laboratory (NPSOL) etc.

3-5-2 Cost function

The most important step in formulating an OCP, is to frame a suitable cost function for the problem. The cost function consists of two parts viz. reference tracking error and control action. In further subsections, the framing of a suitable cost function is explained.

Path tracking

For a path tracking control an obvious choice of cost function is to minimise the tracking error. In Section 3-3 reference paths for each vehicle in the formation is generated. These path are then transformed from global to path frame. In spatial-domain, the variable d^S represent the lateral offset from the modified Dubin's path, representing the tracking error. In order to track the reference path, each vehicle should track its d^S. Hence, the error in d^S is minimised to achieve the best tracking.

This can be written in mathematical form as,

$$\min_{\mathcal{U}_i} \int_{s_{0_1}, s_{0_2}, \ldots, s_{0_n}}^{s_{f_1}, s_{f_2}, \ldots, s_{f_n}} K_d \left(\left(d_1^S - d_{r,1}^S \right)^2 + \left(d_2^S - d_{r,2}^S \right)^2 + \cdots + \left(d_{n+1}^S - d_{n+1,r}^S \right)^2 \right)$$

where, d_i^S and $d_{r,i}^S$ represent the lateral offset(state) and the reference offset for vehicle $V_i \quad \forall i \in \{1, 2, \ldots, n+1\}$ respectively. K_d is the tuning parameter and $\mathcal{U}$ is the set of control inputs.

Velocity tracking

In formation control the crucial variable of interest is the surge velocity. It might sound obvious that each vehicle should travel at a same velocity in order to maintain formation. This is true only for *straight* paths, in case of *curve* paths this is no longer true. This is because the vehicle on outer side should travel a larger distance compared inner vehicles. If all vehicles travel at same velocity even in the *curve* section of path, then the innermost vehicle will always be ahead, and the outermost vehicle will always lag behind the rest of the vehicles in formation. In order to overcome this situation, different velocity profiles are generated for each vehicle in formation based on the path and their position in formation.

As described in Section 3-4-2, Dubin's path is modified by extending it on both ends by a *straight* section. This extended path acts as a section to accelerate and decelerate between zero and common reference velocity (u_{ref}^B). Meaning, initially each vehicle in formation accelerates from zero to a common reference velocity at constant acceleration. At the end of the path each vehicle decelerates from a common reference velocity to zero at constant deceleration. For a vehicle V_n the reference velocity during the *curve* section depends on its position in formation as well as the direction of turning of the *curve*. If a vehicle is positioned on the left side of the virtual vehicle(V_c), and the *curve* turns towards left then, this particular vehicle will lie on the inner side of the curve. For this combination of vehicle position and *curve* direction, the vehicle should slow down from the common reference velocity u_{ref}^B. If a particular vehicle V_n lie on the right side of the virtual vehicle(V_c) and the *curve* turn towards left, then the vehicle lie on outer side of the *curve* and should speed up from the common reference velocity. This is illustrated in Algorithm 3. Where, u_{ref}^B is the common reference velocity, R is the minimum turning radius of V_c's path and d_k^S is the lateral offset for vehicle V_k.

From the equations in Algorithm 3, it can be seen that, the velocity for each vehicle depends on their respective offset d^S. In Figure 3-9 it was shown that, each vehicle has a discontinuous point in offset, which in turn reflects on its velocity. This discontinuity in velocities is not desirable, as it may result in jerky behaviour in motion. Apart from jerky motion, it may also demand higher computational requirements. Hence, the velocities are calculated only at extreme points of switch between *straight* and *curve*. For velocities in between these extreme points, values are interpolated using cubic interpolation.

Figure 3-11 shows the velocity references for a triangular formation shown in Figure 3-2, where vehicle V_2 is on the left, and V_1 and V_3 are on the right side of V_c. Reference path is of form LSL, meaning it turns towards left both the times. It can be seen that initially all vehicles accelerate till u_{ref}^B which is the common reference velocity. Once u_{ref}^B is reached, V_c maintains

Algorithm 3 Reference velocity

Input: $curve$, $V_{LR,k}$

if $(curve = L \quad \text{AND} \quad V_{LR,k} = L) \quad OR \quad (curve = R \quad \text{AND} \quad V_{LR,k} = R)$ **then**

  $u^B_{r,k} = \left(R - d^S_k\right) u^B_{\text{ref}}$

else

  if $(curve = L \quad \text{AND} \quad V_{LR,k} = R) \quad OR \quad (curve = R \quad \text{AND} \quad V_{LR,k} = L)$ **then**

    $u^B_{r,k} = \left(R + d^S_k\right) u^B_{\text{ref}}$

  end if

end if $\quad \forall k \in \{1, 2, \ldots, n\}$

Output: $u^B_{r,k}$

the same velocity. In order to maintain the formation, vehicles V_1 and V_3 accelerate further. This is because, these vehicles are present on the outer side of the *curve*. In the same way, vehicle V_2 which is present of the inner side of the *curve* decelerates. Once all vehicles reach their respective velocities, they maintain the same velocity for the remaining *curve* section of path. Once the vehicles reach the *straight* section of path, their velocities become equal to u^B_{ref}. The same procedure is followed for the second *curve* and finally the velocities for all vehicle become u^B_{ref} and decelerate to zero.

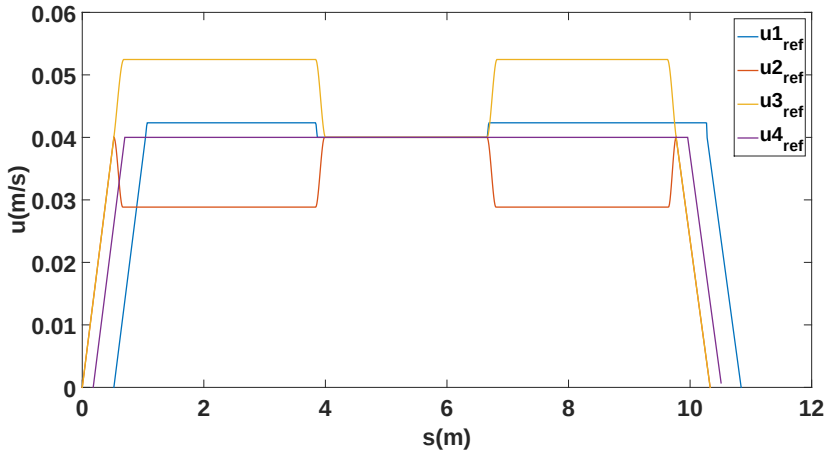


Figure 3-11: Reference velocities

Velocity tracking can be written in mathematical form as,

$$\min_{\mathcal{U}_i} \int_{s_{0_1}, s_{0_2}, \ldots, s_{0_n}}^{s_{f_1}, s_{f_2}, \ldots, s_{f_n}} K_u \left(\left(u^B_1 - u^B_{\text{rf},1}\right)^2 + \left(u^B_2 - u^B_{\text{rf},2}\right)^2 + \cdots + \left(u^B_{n+1} - u^B_{\text{rf},n+1}\right)^2 \right)$$

where, u^B_i and $u^B_{\text{rf},i}$ represent the surge velocity(state) and the reference velocity for vehicle $V_i \quad \forall i \in \{1, 2, \ldots, n+1\}$ respectively. K_u is the tuning parameter.

Control input

While designing a tracking MPC, care should be taken on the control input. Generally the resulting control inputs exhibits an inconsistent oscillations in order to provide the best accuracy. However, this oscillating behaviour may damage the actuators when applied on the practical set-up. In order to suppress these oscillations derivative of the control inputs are minimised which can be written in mathematical form as,

$$\min_{\mathcal{U}_i} \int_{s_{0_1},s_{0_2},\ldots,s_{0_n}}^{s_{f_1},s_{f_2},\ldots,s_{f_n}} K_{\mathcal{U}} \left(\frac{d\mathcal{U}_1}{ds}^2 + \frac{d\mathcal{U}_2}{ds}^2 + \cdots + \frac{d\mathcal{U}_{n+1}}{ds}^2 \right)$$

3-5-3 Constraints

In order to formulate a MPC problem, appropriate constraints are to be placed on the system variables. In this section constraints placed on the system are explained.

Input constraints

The input for each hovercraft is the forward thrust and the torque. The thrust is limited by zero in reverse direction, as the vehicle is not actuated backwards. In forward direction it is limited by $\overline{\tau_1}$.

The hovercraft should be capable of turning on both sides, the torque is limited by $\underline{\tau_2}$ and $\overline{\tau_2}$.

These input constraints are written as,

$$\begin{aligned} &\underline{\tau_{1i}} \leq \tau_{1i}(s) \leq \overline{\tau_{1i}} \\ &\underline{\tau_{2i}} \leq \tau_{2i}(s) \leq \overline{\tau_{2i}} \quad \forall i \in \{1, 2, \ldots, n+1\} \end{aligned}$$

where, $\tau_{1i}(s)$ and $\tau_{2i}(s)$ represent thrust and torques on vehicle V_i respectively.

The above input constraints can be written in compact form as,

$$\underline{\mathcal{U}_i} \leq \mathcal{U}_i(s) \leq \overline{\mathcal{U}_i} \quad \forall i \in \{1, 2, \ldots, n+1\}$$

State constraints

In the current formulation, there are no bounds on the states due to the following reasons,

- A hovercraft is capable of reaching any position in the x-y plane.
- The control input acts on the acceleration of the hovercraft. By adding bounds on input, bounds on acceleration are imposed. Hence, adding additional constraint on velocities is not desirable

In order to impose initial conditions on system states, equality constraints are added. These constraints can be written as,

$$\xi_i(0) = \xi_{i0} \quad \forall i \in \{1, 2, \ldots, n+1\}$$

3-5-4 Resulting OCP

All the prerequisites for the formation control using hovercraft are introduced in the previous sections. The resulting OCP is written as,

$$\begin{aligned}
\min_{\mathcal{U}_i} \quad & \int_{s_{1_0},s_{2_0},\ldots,s_{n_0}}^{s_{1_f},s_{2_f},\ldots,s_{n_f}} K_d \left(\left(d_1^S - d_{r,1}^S\right)^2 + \left(d_2^S - d_{r,2}^S\right)^2 + \cdots + \left(d_{n+1}^S - d_{r,n+1}^S\right)^2 \right) + \\
& K_u \left(\left(u_1^B - u_{\mathrm{rf},1}^B\right)^2 + \left(u_2^B - u_{\mathrm{rf},2}^B\right)^2 + \cdots + \left(u_{n+1}^B - u_{\mathrm{rf},n+1}^B\right)^2 \right) + \\
& K_{\mathcal{U}} \left(\frac{d\mathcal{U}_1}{ds}^2 + \frac{d\mathcal{U}_2}{ds}^2 + \cdots + \frac{d\mathcal{U}_{n+1}}{ds}^2 \right) \\
\text{subject to} \quad & \frac{\xi_i}{ds} = f\left(\xi_i, \mathcal{U}_i\right) \\
& \xi_i(0) = \xi_{i0} \\
& \underline{\mathcal{U}_i} \leq \mathcal{U}_i(s) \leq \overline{\mathcal{U}_i} \quad \forall i \in \{1, 2, \ldots, n+1\}
\end{aligned} \tag{3-5}$$

where, state vector ξ and control input $\mathcal{U}$ are defined as,

$$\begin{aligned}
\xi &= \left[u^B, v^B, r^B, d^S, e_\psi^S\right]^T \\
\mathcal{U} &= \left[\tau_1, \tau_2\right]^T
\end{aligned}$$

3-6 Formulation of MPC

The OCP framed in the previous section is solved for a certain horizon length with the present states as initial conditions and the resulting control inputs are supplied to each hovercraft, as each vehicles move the states are updated and the OCP solved with these updates states. Figure 3-12 shows the block diagram of the control scheme.

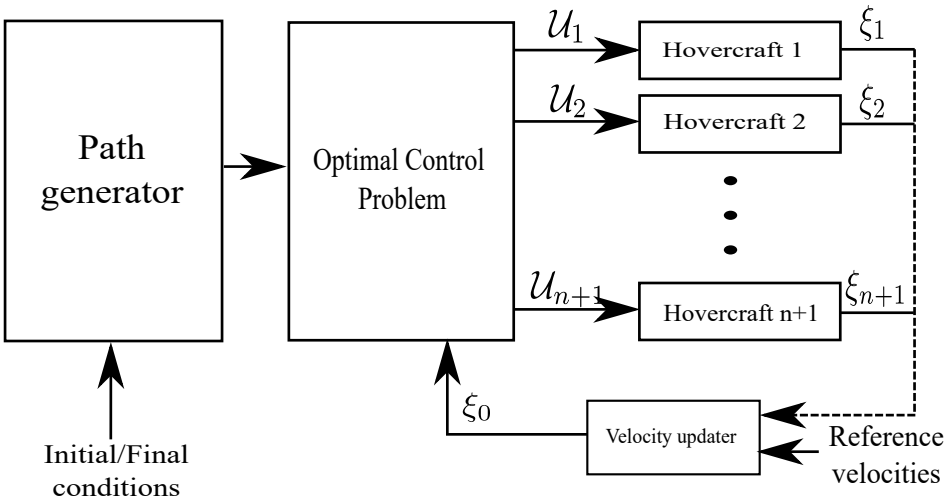


Figure 3-12: Block diagram for the control scheme

An interesting part in the block diagram is the velocity updater. This block updates the reference velocity for each vehicle, based on the error in the formation. Even though the controller tries to track the reference velocity profile, it always results in a small error. This error keeps on integrating, subsequently causes a major error in position. This position error deforms the formation which is highly undesirable. In order to avoid this, the reference velocity, u_r^B is updated at each iteration based on the relative position error.

Algorithm 4 Velocity updater

Input: $\left(s_i, d_i^S\right), \vec{V}_{kc}$

$\begin{bmatrix} x_i^I \\ y_i^I \end{bmatrix} = f_{\text{path}}(s_i) + ||\vec{N}||_2 d_i^S$

$D_{k_c} = \sqrt{\left(x_k^I - x_c^I\right)^2 + \left(y_k^I - y_c^I\right)^2}$ ▷ Distance between virtual and member vehicles

$E_{k_c} = |\vec{V_{kc}}| - D_{k_c}$ ▷ Error in positional states w.r.t reference formation

$E_m = \texttt{max}(\texttt{abs}(E_{kc}))$

if $E_m \geq t_1$ **then return** OFF

end if

if $E_m \geq t_2$ **then**

    $u_{\text{rf},i}^B = \left[1 - (K_{1_i} E_m)\right] u_{r,i}^B$

else

    $u_{\text{rf},k}^B = \left[1 + (K_{2_k} E_{k_c})\right] u_{r,k}^B$

    $u_{\text{rf},n+1}^B = u_{r,n+1}^B$

end if $\quad \forall i \in \{1, 2, \ldots, n+1\}, \forall k \in \{1, 2, \ldots, n\}$

Output: $u_{\text{rf},i}^B$

Algorithm 4 illustrates the procedure of the velocity updater. The position errors are calculated by converting the spatial-domain variables $\left(s_i, d_i^S\right)$ to global frame. Based on the relative position error, reference velocities for member vehicles in formation are updated using the equation specified in Algorithm 4, where K_{2_k} is a tuning parameter. During the run if one of the vehicle fails(can no longer be actuated), this results in deformation from the reference formation. In order to avoid this, an attempt is made by slowing down all the vehicles in formation including the virtual vehicle. This is done with the help of absolute maximum error. When this error crosses the threshold t_2, velocities for all vehicles is reduced based on the tuning parameter K_{1_i}. If the position error exceeds the threshold t_1, then whole system is turned OFF. Significance of the velocity updater is illustrated in Section 4-1-3.

The first and foremost decision that has to be made while designing a MPC is the selection of sampling frequency. The rule of thumb is to take at least 1/10th of the rise time. This rule hold true for Single Input Single Output (SISO) system. However, there is no such rule of thumb for a Multiple Input Multiple Output (MIMO) system. Ideally the sampling time should be as high as possible. In the case of MPC using a very high sampling rate is not possible, as the controller has to compute the control input with in each sampling interval and the computation time depends on many factors like, complexity of the system, type of solver used, amount of accuracy required and the initial guess. Based on the solver's speed and system dynamics sampling time of 0.2 s is chosen.

As said previously, the initial guess is the major factor in reducing the computation time

for each iteration. If the initial guess is far away from the solution, the computation time can be very high, which is not desirable to obtain RT performance. In order to achieve RT performance, the solution of previous iteration is provided as the initial guess for the next iteration. Significance of initial guess is illustrated in Section 4-1-4.

3-6-1 Tuning parameters

Every controller has some parameters that are to be tuned in order to achieve the desired performance. Wrong choice of parameters may even result in infeasible solutions. In this section tuning parameters for the OCP and the velocity updater are discussed.

Weights on state and control input

OCP presented in (3-5) has some weights on path tracking, reference velocity tracking, and control. These weights are represented by K_d, K_u, and $K_{\mathcal{U}}$. In formation control, all the member vehicles should track the reference path with the highest possible accuracy in order to maintain the desired formation. Path tracking depends on two parameters, viz. path length, s and lateral offset, d^S. The path length is governed by the velocity tracking and the lateral offset is governed by the path tracking cost. At every iteration, reference velocity is updated with the help of the velocity updater. Hence, a higher weight is provided to the lateral offset compared to velocity tracking. Aggressive control inputs may result in jerky behaviour in motion, sometimes it may even damage the actuators. Hence, the control inputs should have considerable amount of weights. However, if the weights on control inputs are very high the vehicles may move very slowly or may not even move. Hence, the weights on control inputs are lower than that of velocity tracking. Final weights on state and control input are,

$$K_d > K_u > K_{\mathcal{U}}$$

Weights on velocity updater

Algorithm 4 illustrates the procedure of the velocity updater. At every iteration, based on relative position error, E_{k_c} reference velocity for each member vehicle in formation is updated by,

$$u^B_{\mathrm{rf},k} = \left[1 + (K_{2_k} E_{k_c})\right] u^B_{r,k}$$

In above equation the tuning parameter, K_{2_k} for each member vehicle is chosen based on their respective position in formation. If a specific vehicle V_n is ahead of the virtual vehicle, then the tuning parameter for this specific vehicle should be greater than or equal to zero. This is because, if a vehicle V_n is travelling at lower velocity, the relative position error E_{k_c} is positive. In this case, the velocity updater should update the reference velocity for this vehicle by a higher value. In the same way, if the specific vehicle is travelling at a higher

velocity then the positional error E_{k_c} is negative. In this case, the velocity updater should update the reference velocity by a lower value. This is possible only when K_{2_k} is greater than or equal to zero. Otherwise, the velocity updater may generate negative reference velocity, resulting in an infeasible solution.

In the same way, if a specific member vehicle in formation is behind the virtual vehicle, then the tuning parameter for this specific vehicle should be less than or equal to zero. Conditions for the tuning parameter K_{2_k} summarised by following equation.

$$K_{2_k} \begin{cases} \geq 0, & \text{if member vehicle is ahead of virtual vehicle} \\ \leq 0, & \text{if member vehicle is behind virtual vehicle} \end{cases}$$

If a member vehicle in the formation fails and cannot actuate for the remaining run, then the relative position error for this specific vehicle keeps on increasing. When the error cross the threshold t_2, then the remaining vehicles in formation should start decelerating and finally come to rest. In this case velocity updater updates reference velocity not only for each member vehicle but also for the virtual vehicle by following equation,

$$u^B_{\mathrm{rf},i} = \left[1 - (K_{1_i} E_m)\right] u^B_{r,i} \tag{3-6}$$

In the above equation the value of tuning parameter K_{1_i} is based on the threshold t_2. If a specific vehicle, V_n fails, irrespective of its position, E_m is always positive. In order to reduce the velocity of all vehicles in formation K_{1_i} should be chosen such that $(K_{1_i} E_m)$ is always between zero and 1. If the value is greater than 1, then it results in an infeasible solution due to negative velocity. Where as, if the value is negative, then the vehicle starts accelerating instead of decelerating. A major difficulty in choosing the value of K_{1_i} is that if $E_m >> t_2$, there are still chances of infeasibility. Hence, K_{1_i} is chosen such that, $K_{1_i} << \frac{1}{t_2}$. But choosing a very low value for K_{1_i} may decelerate the vehicles slowly, still incapable of avoiding deformation. An alternative to this is by updating the velocity by following equation instead of (3-6).

$$u^B_{\mathrm{rf},i} = \left[1 - (K_{1_i} t_2)\right] u^B_{r,i}$$

such that,

$$0 < K_{1_i} < \frac{1}{t_2}$$

Prediction horizon

Prediction horizon is an important tuning parameter in MPC. In case of the direct collocation method, the continuous problem is discretized into n number of collocation points. The solver evaluates the cost function and the problem constraints are satisfied only for these number of points. Hence, even if the prediction length is increased, there is no significant change in terms of computation time, but there may be differences in accuracy. However, by increasing the number of collocation points, computation time is increased by a larger margin. This is because, by increasing the number of collocation points, the number of decision variables are increased, there by increasing the computation time of the solver to solve the OCP. In this work, a prediction length of 0.5 m and 5 collocation points are chosen.

3-6-2 Comparison

The MPC has its advantages of constraint handling and specially for path tracking application it can predict the optimal trajectory and achieve feedback control in an unified manner. However, MPC demands a fast solver and powerful processor to solve the problem fast enough to achieve RT performance. In order to gain the required performance with the available solvers several attempts are made to simplify the problem and reduce the number of variables. As discussed earlier, a hovercraft is a nonlinear, nonholonomic and an underactuated system. Hence, any further simplifications in the model may deviate it from the actual response. Hence, in the proposed method an attempt is made to reduce the number of decision variables and the number of constraint evaluations which may result in RT performance. In this section a comparative study with existing time-domain parametrized method is made and its scalability is assessed.

The conventional time-domain parametrized method for formation control problem using hovercrafts can be written as,

$$\begin{aligned} \min_{\mathcal{U}_i} \quad & \int_{t_0}^{t_f} \mathcal{L}\left(\xi_i(t), \mathcal{U}_i(t)\right) \\ \text{subject to} \quad & \frac{d\xi_i}{dt} = f(\xi_i, \mathcal{U}_i) \\ & \xi_i(0) = \xi_{i0} \\ & \underline{\mathcal{U}} \leq \mathcal{U}_i(t) \leq \overline{\mathcal{U}} \\ & \underline{\xi} \leq \xi_i(t) \leq \overline{\xi} \end{aligned}$$

where the state vector ξ is defined as,

$$\xi_i = \left[x_i^I, y_i^I, \psi_i^I, u_i^B, v_i^B, r_i^B, l_i^I, \psi_i^I\right]^T, \quad \forall i \in \{1, 2, \ldots, n\}$$

Notice that the conventional time-domain parametrized method for formation control consists of 8 states. With $\left(x_i^I, y_i^I, \psi_i^I\right)$ depicting the position and orientation of each vehicle in formation, $\left(u_i^B, v_i^B, r_i^B\right)$ depicting *surge* and *sway* velocities, and *yaw* rate respectively. The additional states, l_i^I and ψ_i^I depict the position and orientation of each member vehicle in formation with respect to the virtual vehicle. Whereas, using spatial-domain for formation control results in reducing the number of states by 3 $\left(\xi = \left[u^B, v^B, r^B, d^S, e_\psi^S\right]^T\right)$. This reduction in number of states becomes significant as the number of vehicles in formation increase. As each state should satisfy its dynamic equation and bounds, by reducing the number of states, their corresponding constraint evaluations are also reduced.

For each state in a system, an OCP results in n number of decision variables(n is the number of collocation points). Every state has its dynamic equations and bounds on its value. Therefore by reducing the number of states in the system, the resulting number of decision variables are reduced and even their corresponding constraints(dynamic equation, bounds and initial conditions).

Table 3-1 shows the comparison in number of decision variables and constraint evaluations for a formation with m number of vehicles and an OCP solved with n number of collocation points.

Figure 3-13 shows the comparison in number of constraints evaluation with the proposed spatial-domain parametrization with the conventional time-domain parametrization. It is clearly seen that as the number of vehicles increase the the reduction in number of constraints evaluation become significant.

Table 3-1: Comparison in number of decision variables

Parameters	Existing method	Proposed method
States	8	5
Decision variables	$8mn$	$5mn$
Bounds	$16mn$	$10mn$
Dynamic equations	$8mn$	$5mn$
Initial conditions	$8m$	$5m$

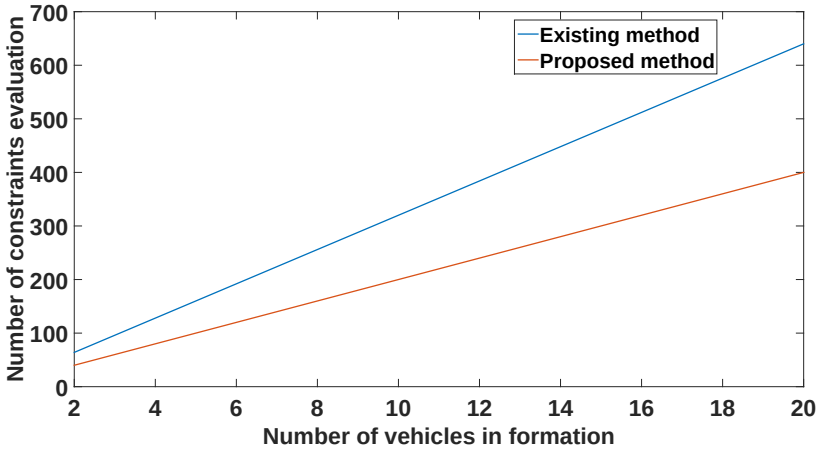


Figure 3-13: Scalability

3-7 Summary

In this chapter the proposed framework for formation control in spatial-domain is discussed. The proposed framework is intended for path tracking applications. Hence, for a given set of initial and final conditions a feasible path is generated using Dubin's path algorithm. A path tracking MPC should predict the position of the vehicle for a certain length(prediction horizon). In order to predict the vehicle's position according to path, a new path coordinate frame is introduced to represent the position of each vehicle in the formation. The vehicle dynamics derived in time-domain are then reformulated to spatial-domain. Finally, a suitable OCP is formulated in spatial-domain to track the path and maintain the formation throughout the run. The comparative study shows that the new framework has reduced the number of decision variables in the MPC which can guarantee the desired RT performance.

After introducing all the prerequisites of the proposed framework in the next chapter results of various numerical simulation are presented. These results will help in analysing the

performance of the controller. The chapter also includes the numerical simulation results of different test cases for formation control such as, vehicle failure, starting from different initial conditions etc.

Chapter 4

Simulation Results

In previous chapters all the ingredients for formation control framework are described. In this chapter the simulation results of Model Predictive Controller (MPC) is presented and the performance of the framework is analyzed. In all simulations the time taken by solver to compute solutions is tracked, this will aid in assessing Real Time (RT) performance. The main goal of this chapter is to show the results of the framework and implementation of changes to improve accuracy of results.

The tuning parameters of both Optimal Control Problem (OCP) and velocity updater are kept constant for all the simulations. Values of these tuning parameters are shown in Table 4-2. For all the simulations a common reference path is chosen which has a switch from *straight* to *curve* at path lengths 0.72 m and 6.85 m, and has switch from *curve* to *straight* at 3.84 m and 9.99 m with respect to the virtual vehicle V_c. Details of the computer used to conduct these simulations are listed in Table 4-1.

Table 4-1: Computer details

Component	Details
Processor	Intel(R) Xeon(R) CPU E5-1620 v3 @ 3.5GHz
Memory(RAM)	32 GB
Number of processor cores	4

Table 4-2: Values of tuning parameters

Parameters	Symbol	Value
Cost weight: d^S	K_d	1
Cost weight: u^B	K_u	1/12
Cost weight: $\mathcal{U}$	$K_{\mathcal{U}}$	1/100
Velocity updater V_1	K_{2_1}	10
Velocity updater V_2	K_{2_2}	-10
Velocity updater V_3	K_{2_3}	-10
Vehicle failure V_1	K_{1_1}	2.8
Vehicle failure V_2	K_{1_2}	2.9
Vehicle failure V_3	K_{1_3}	3.1
Number of collocation points	N	5
Prediction horizon	s_f - s_0	0.5
Threshold	t_1	0.4
Threshold	t_2	0.1

In Section 4-1 significance of different blocks in the framework are discussed followed by analysing the performance of the framework and suggestions for improving the performance. After analysing the performance of the framework, the results of numerical simulations for different test cases are discussed in Section 4-2.

4-1 Precision and accuracy

The designed controller provides results with certain accuracy which can be improved by making changes to the problem. This can be done by choosing a different solver, providing a better initial guess for the optimization problem etc. In this section, different methods to improve the accuracy of results are discussed. Accuracy of the results are analysed with the help of two basic parameters viz. computation time for each iteration and the position error in formation.

4-1-1 Influence of approximating the offset

The reference path for each member vehicle in formation consist of a discontinuous point while switching between *straight* and *curve*. This is shown in Figure 3-4. This discontinuity is reflected in the lateral offset d^S, as shown in Figure 3-9. This discontinuity in path may result in jerky behaviour in motion. It may also lead to higher computation time for the iterations where the specific discontinuous point is part of prediction horizon. In order to avoid this, the lateral offset is approximated by interpolating with extreme points. By doing so the position error in formation may increase but there will be considerable amount reduction in computation time.

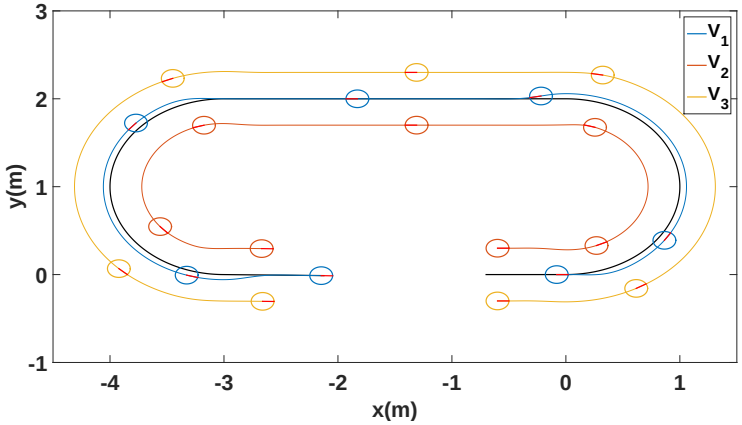
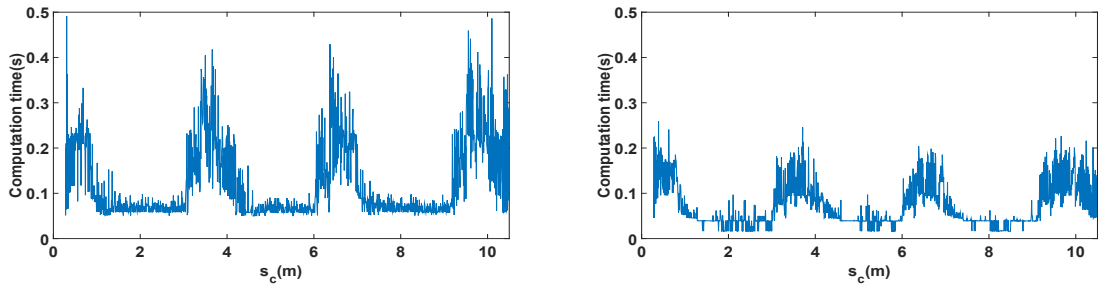


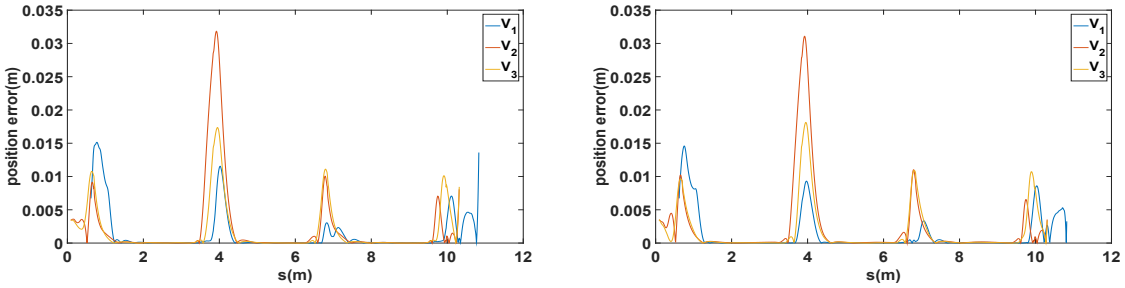
Figure 4-1: x-y plot

In Figure 4-2 it can be clearly seen that, approximating the lateral offset has reduced the computation time greatly. Reduction in computation time helps in achieving the desired RT performance. In Table 4-3, it can be seen that approximating the offset has accounted for 89% decrease in maximum computation time. Apart from reduction in maximum time taken for a single iteration, the overall mean time taken for solving the problem has also decreased by 66%. Due to the smoothed lateral offset Figure 4-4 shows that the controller is capable of tracking the reference more accurately. The method has also improved the position error by a negligible margin. As the method has improved the computation without deteriorating the position error, this approximation is used in all further simulations.



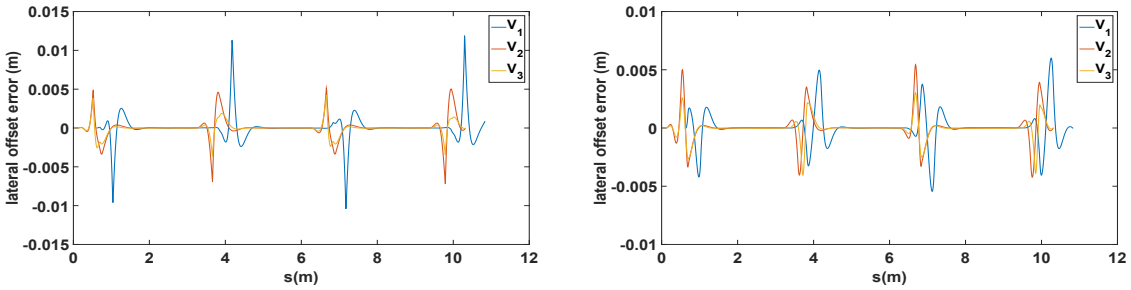
(a) Without approximating the offset **(b)** With approximating the offset

Figure 4-2: Comparison of computation time



(a) Without approximating the offset **(b)** With approximating the offset

Figure 4-3: Comparison of position error



(a) Without approximating the offset **(b)** With approximating the offset

Figure 4-4: Comparison of error in lateral offset

Table 4-3: Comparison of results with approximated d^S

Parameters	Actual offset	Approximated offset
Max computation time (s)	0.4901	0.2589
Mean computation time (s)	0.1246	0.0748
Max pos error (m)	0.0315	0.0311
Mean pos error (m)	0.0020	0.0021

4-1-2 Influence of solver

In literature large number of fast solvers are available to solve the resulted Nonlinear Program (NLP). Each have their own advantages and disadvantages based on the application. In this section, the performance of two different solvers are analysed. The first one is Sparse Nonlinear OPTimizer (SNOPT) [25], and the second one is Nonlinear Programming Systems Optimization Laboratory (NPSOL) [26], [27].

In order to compare the performance of each solver, the function value for each iteration is also taken in to account apart from computational time and position error.

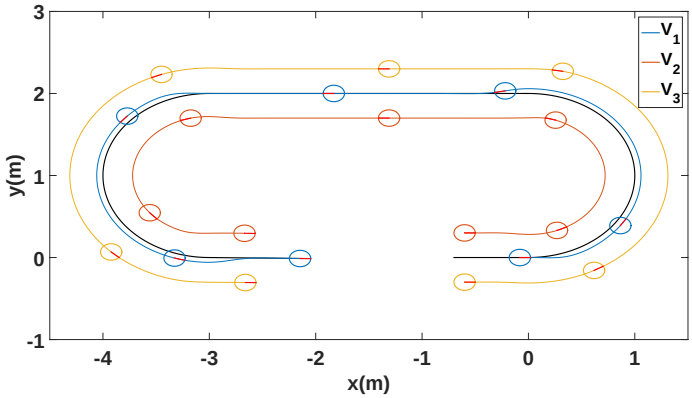
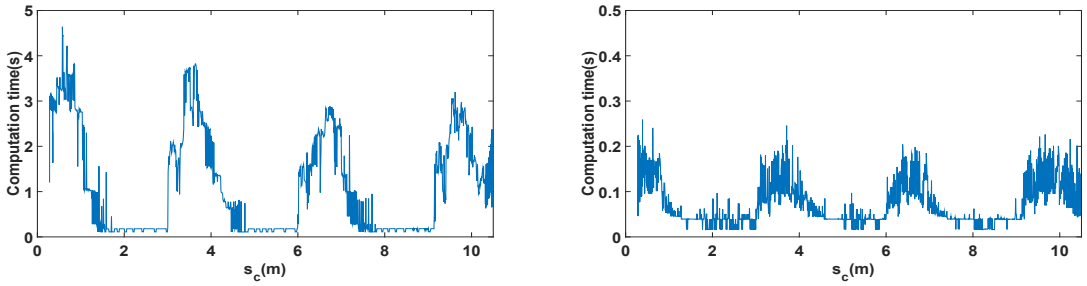
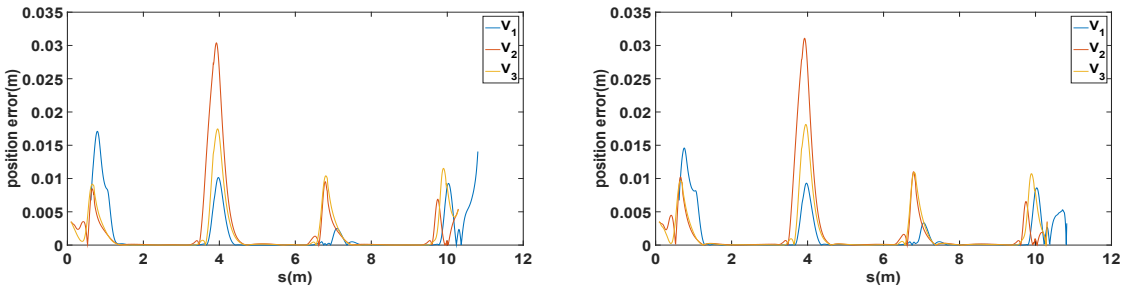


Figure 4-5: x-y plot



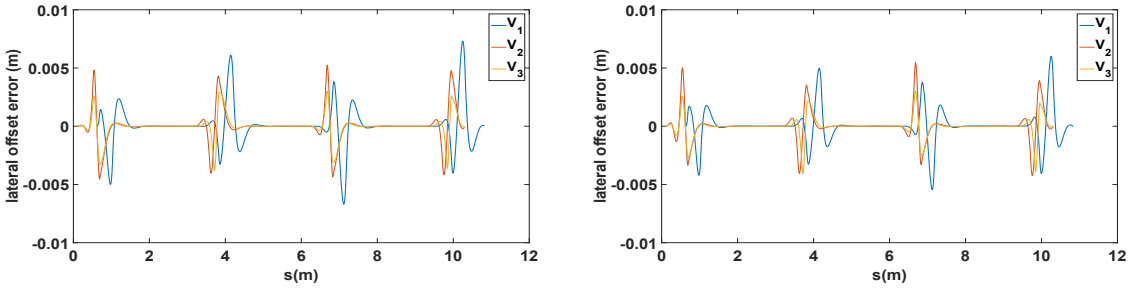
(a) NPSOL **(b)** SNOPT

Figure 4-6: Comparison of computation time



(a) NPSOL **(b)** SNOPT

Figure 4-7: Comparison of positional error



(a) NPSOL **(b)** SNOPT

Figure 4-8: Comparison of error in lateral offset

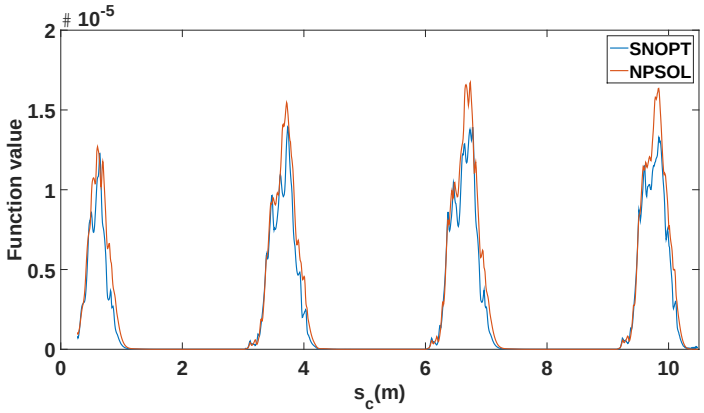


Figure 4-9: Function value

The above Figures clearly indicate that SNOPT has the better performance with respect to all the criteria. The function value for each iteration shown in Figure 4-9 indicates that both the solvers result in same solutions for most part of the path. But the time taken by NPSOL is much higher than that of SNOPT making it highly undesirable for this application. This is due to sparse nature of the problem, SNOPT being a sparse solver it is capable of delivering the better results compared to NPSOL which is a preferred solver for dense problems. As SNOPT has delivered better performance both in terms of position error as well as computation time. This solver is used in all further simulations.

4-1-3 Influence of velocity updater

In Section 3-6 it is highlighted that due to small error in velocity tracking the position error is integrated. In order to rectify this integration effect, a velocity updater block is introduced which updates the reference velocity based on the relative position error for each vehicle in the formation. In this section significance of this velocity updater block is illustrated.

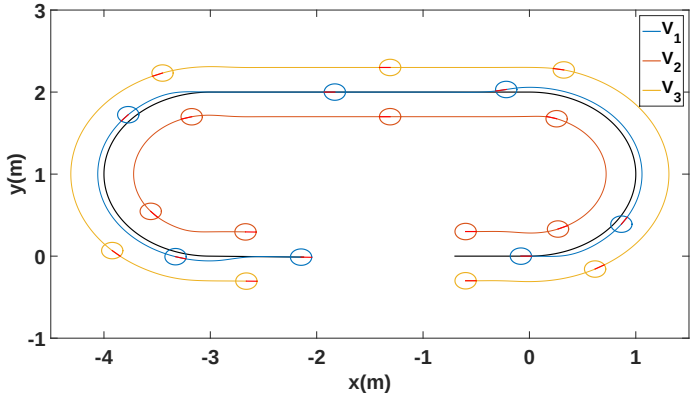
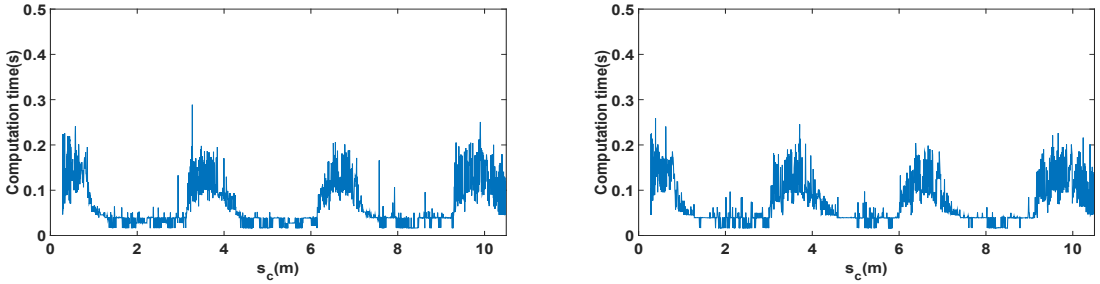
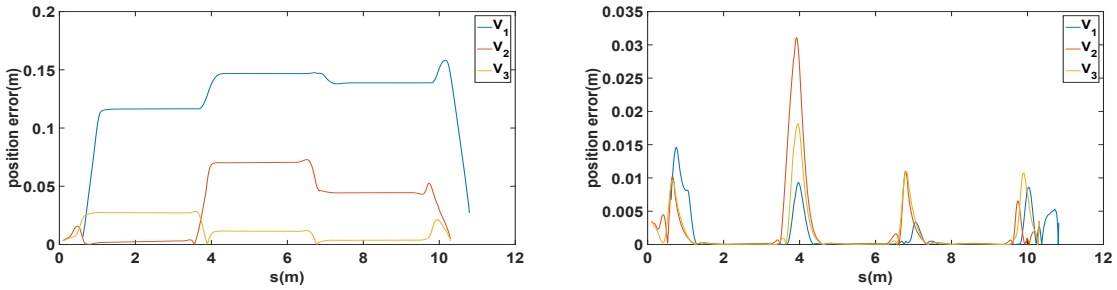


Figure 4-10: x-y plot



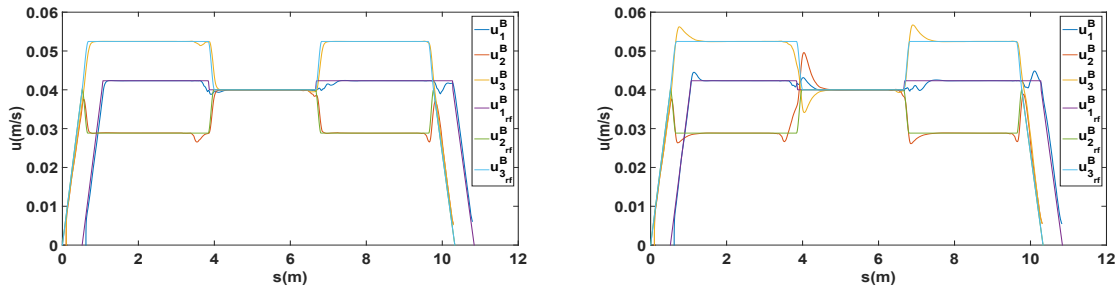
(a) Without the velocity updater **(b)** With the velocity updater

Figure 4-11: Comparison of computation time



(a) Without the velocity updater **(b)** With the velocity updater

Figure 4-12: Comparison of position error



(a) Without the velocity updater **(b)** With the velocity updater

Figure 4-13: Comparison of surge velocity

In Figure 4-13 it is shown that the controller is capable of tracking the reference velocity. However, the small error in tracking the velocity has resulted in integration of relative position error which is shown in Figure 4-12a. In order to suppress this error, the surge velocity is updated based on the relative position error after each iteration. By doing so, the velocity may deviate from its original reference by a very large margin, but it keeps the relative position error within the bounds, which is one of the basic objectives of the controller. Thus maintaining the desired formation.

4-1-4 Influence of initial guess

In solving an optimization problem apart from the choice of a suitable solver, the next most important choice to be made is the initial guess for the variables in the problem. This guess reflects the quality of the solution provided and the time taken to solve the problem. If the initial guess is far from the actual solution, it may take longer time to compute the results which is not desirable for RT applications. Hence, providing a suitable initial guess is crucial in obtaining best results.

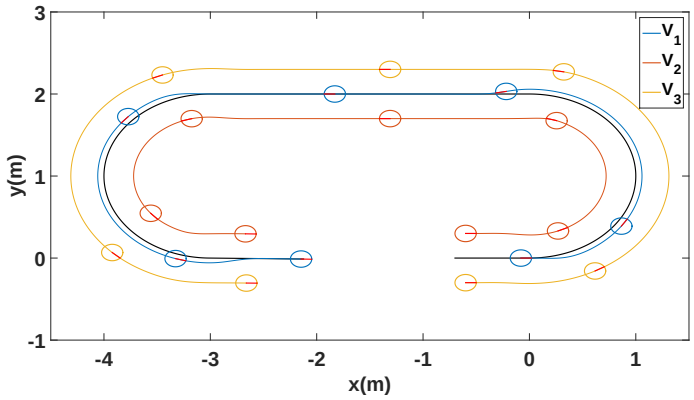
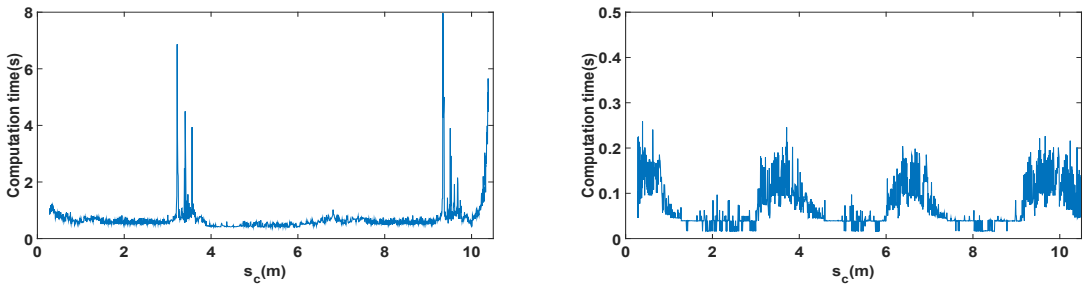
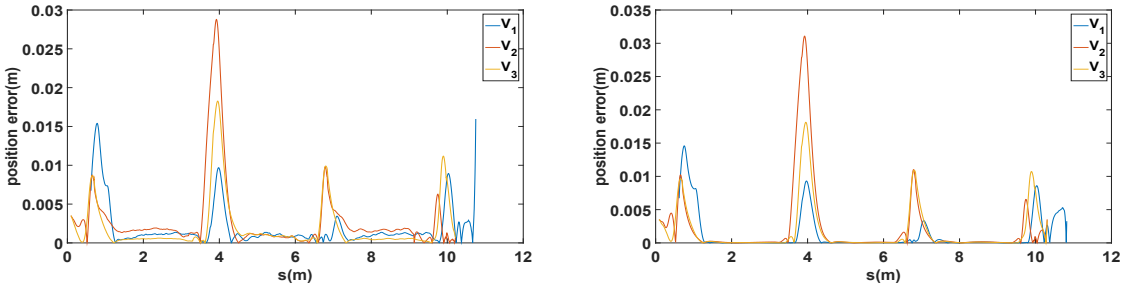


Figure 4-14: x-y plot



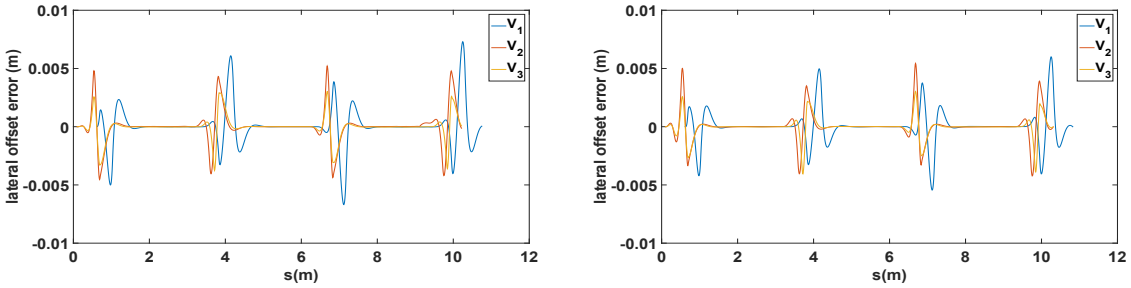
(a) Common initial guess **(b)** Updated initial guess

Figure 4-15: Comparison of computation time



(a) Common initial guess **(b)** Updated initial guess

Figure 4-16: Comparison of position error



(a) Common initial guess **(b)** Updated initial guess

Figure 4-17: Comparison of error in lateral offset

For one set of simulations a common initial guess is provided for all iterations, and for another set of simulations the solution of previous iteration is provided as an initial guess for the next iteration. The significance of providing the solution of previous iteration as an initial guess is clearly shown in Figure 4-15. By providing a common initial guess the solver is capable of finding an accurate solution, but the time taken to solve the problem is very high. Thus, failing to deliver RT performance.

To study the different test cases in the next section, the lateral offset for all the member vehicles in formation is approximated using cubic interpolation, and the reference velocity is updated at each iteration based on the relative position error. The problem is solved using

SNOPT solver and the solution of previous iteration is provided as an initial guess for the next iteration.

4-2 Study of different test cases

In the previous section various aspects of improving the accuracy of results are discussed. By incorporating all the suggestions, the formation control framework is assessed based on different test cases. In this section, the results of these test cases are discussed.

4-2-1 Disturbance analysis

When vehicles are moving in an environment they are always susceptible to external disturbances. The designed controller should be capable of rejecting these external disturbances and drive the vehicles along the desired path. In this section the performance of the controller for an external disturbance is discussed.

In simulation study an impulse disturbance is applied on all the vehicles in different directions. For vehicle V_1 an impulse at path length 4.3 m act on it simulating a stopping behaviour and at path length 8.2 m an impulse is applied from right side simulating a lateral push. For vehicle V_2 an impulse at path length 2.05 m is applied from left side simulating a lateral push from opposite direction. For vehicle V_3 an impulse is applied at path length 6.3 m from behind in the direction of motion simulating a forward push.

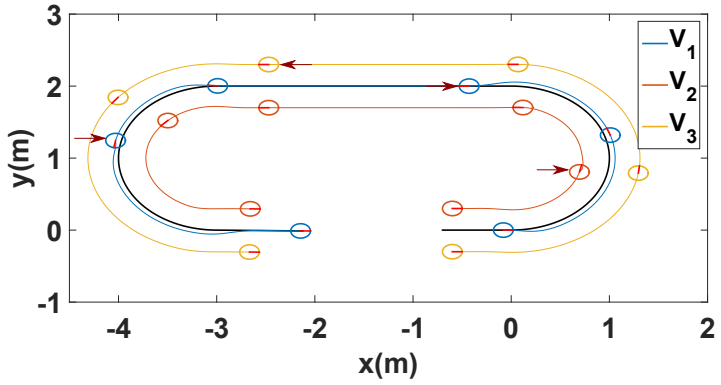
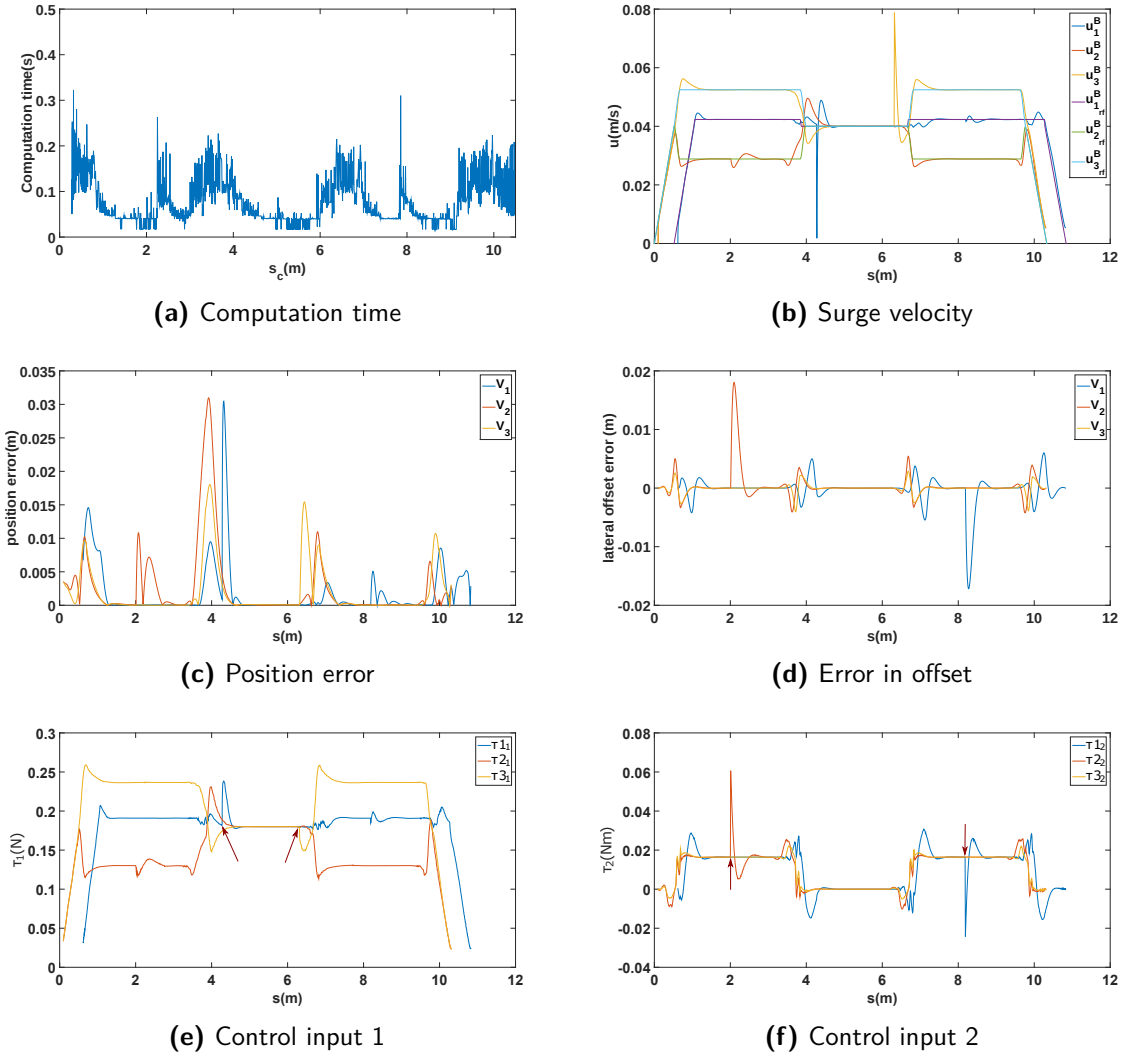


Figure 4-18: x-y plot



(a) Computation time

(b) Surge velocity

(c) Position error

(d) Error in offset

(e) Control input 1

(f) Control input 2

Figure 4-19: Test results

The control inputs in Figure 4-19e and Figure 4-19f show that the controller applies the counter input for the disturbances and maintain the reference formation while tracking the path. For vehicle V_2 at path length 4.3 m when a disturbance is applied opposite to the direction of motion, the controller applies the counter input by increasing the thrust for this particular vehicle. Thus accelerating the vehicle catch up with rest of the vehicles and maintain the formation. In the same way at path length 6.3 m the vehicle V_3 undergoes disturbance pushing the vehicle forward. As shown in Figure 4-19e the controller decelerates this particular vehicle in order to maintain the formation. Vehicles V_2 and V_1 are pushed laterally at path lengths 2.05 m and 8.2 m respectively driving them off their desired paths as shown in Figure 4-19d. In order bring them back on their paths the controller applied counter torques as shown in Figure 4-19f.

4-2-2 Starting from a different initial condition

Apart from rejecting disturbances, the formation control framework should also be capable of driving vehicles into desired formation when they are started from a different initial conditions. In this section the performance of the framework for a case where the desired formation is triangular, with the vehicles starting in a straight line is presented.

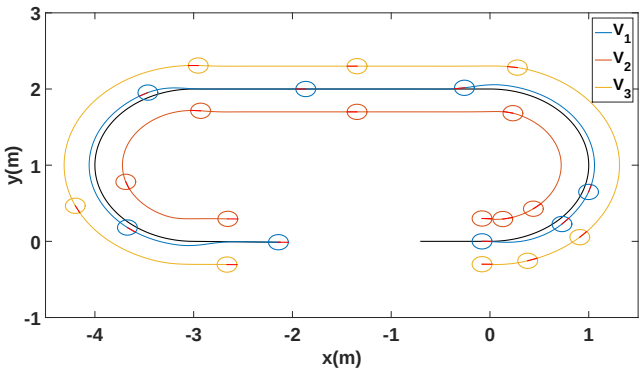
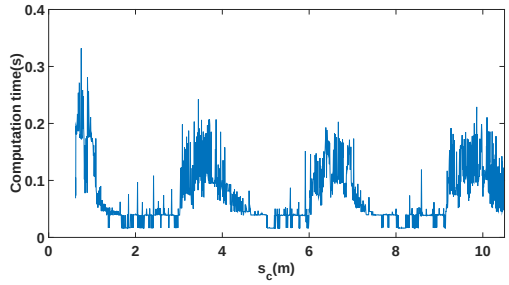
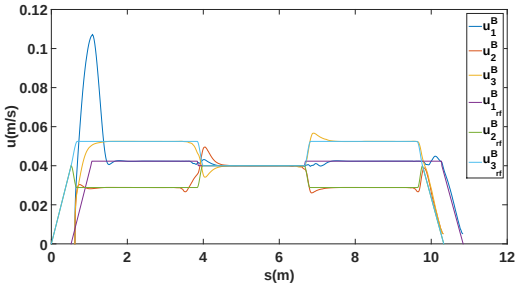


Figure 4-20: x-y plot

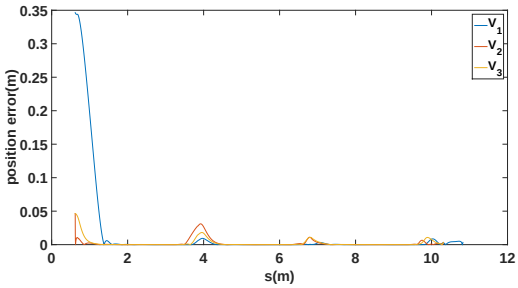
All the vehicles are started in straight line at path length 0.6196 m. Initially the vehicle V_1 has a relative position error of 0.3464 m with respect to virtual vehicle, and both the vehicles V_2 and V_3 have an error of 0.0464 m as shown in Figure 4-21c. Based on these position errors the velocity updater drives V_1 faster and V_2, V_3 slower to reduce the error and thus driving them into formation. The framework requires 0.6314 m to drive the vehicles into formation and reduce the relative position error to the desired bound of 5 cm. This can be further reduced by tuning the parameters, but doing so may saturate the control inputs.



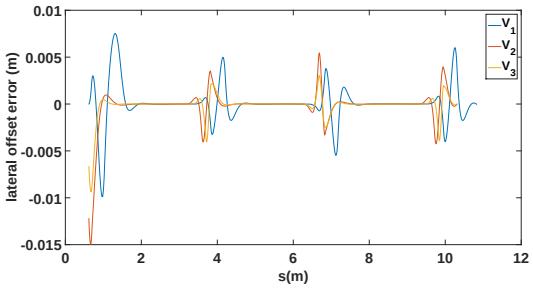
(a) Computation time



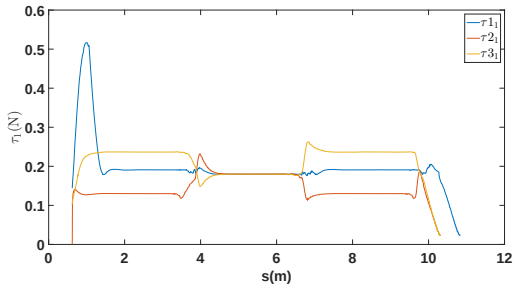
(b) Surge velocity



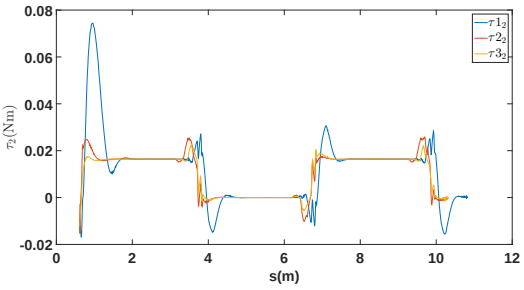
(c) Positional error



(d) Error in offset



(e) Control input 1



(f) Control input 2

Figure 4-21: Test results

4-2-3 Vehicle failure analysis

If an environment consists of multiple vehicles, there can be a case where one of the vehicle has failed and can no more actuate. If any vehicle in the formation fails then the other vehicles should halt in an attempt to maintain the rigid formation.

In this section, the performance of the framework for vehicle failure case is discussed. This is simulated by stopping the vehicle V_2 at path length 5.55 m and the results of the controller are recorded.

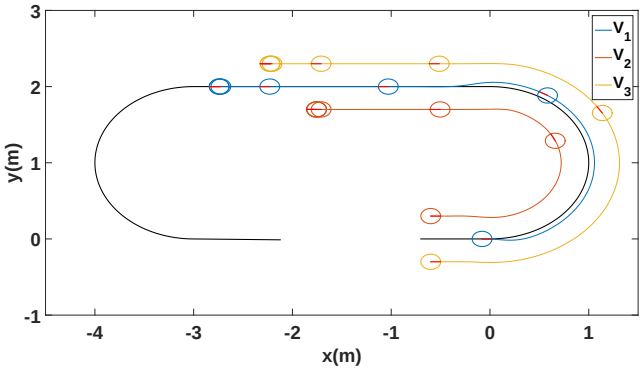
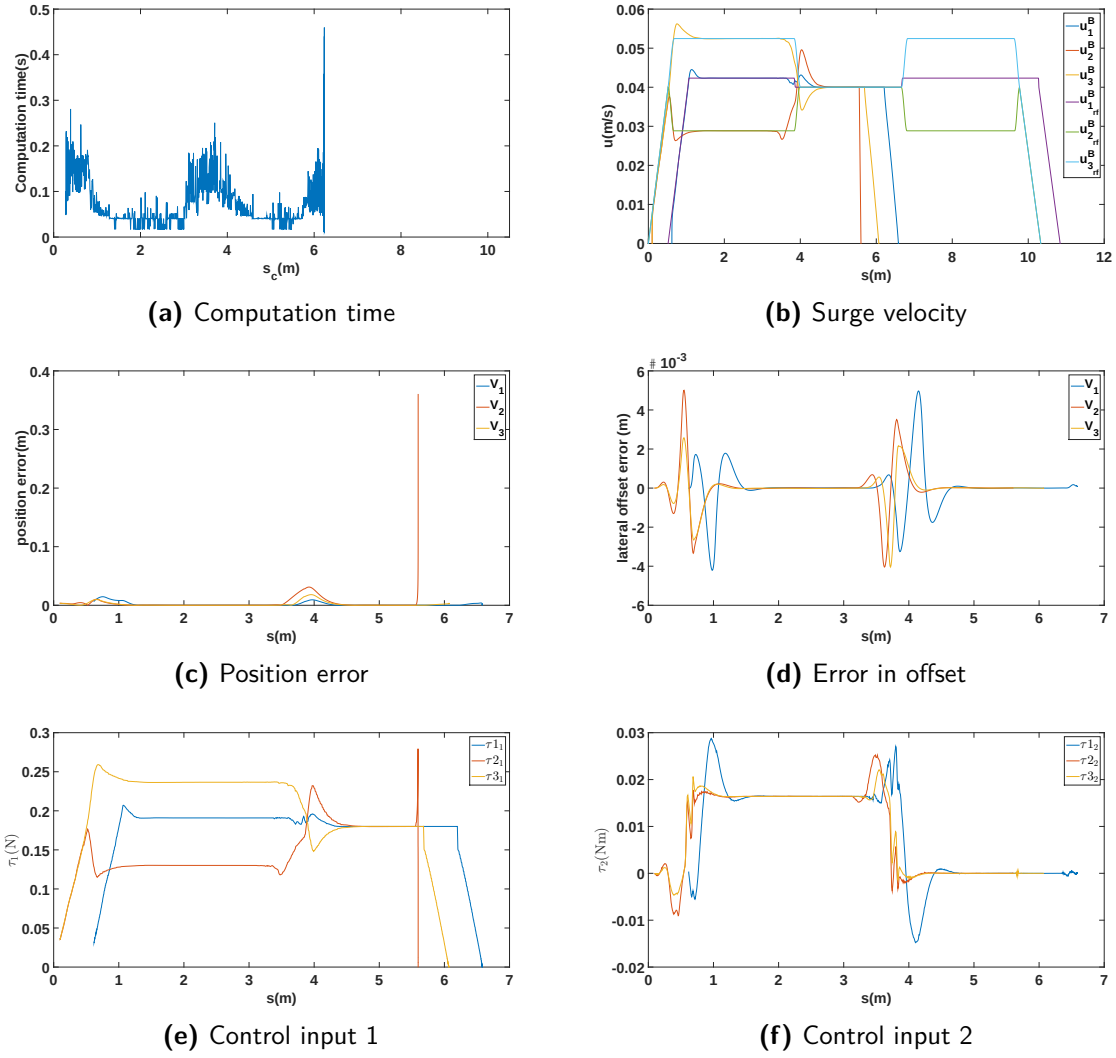


Figure 4-22: x-y plot

At 5.55 m the vehicle V_2 stops, simulating its failure. As a result, the controller tries to accelerate the vehicle V_2 in an attempt to catch up with rest of the vehicles and maintain the formation. As the vehicle cannot move whatsoever, the relative position error increases. When the error crosses the threshold t_2 all vehicles start to decelerate including the virtual vehicle. Even though the controller stops all vehicles the position error for the vehicle has exceeded the desired error bounds.



(a) Computation time

(b) Surge velocity

(c) Position error

(d) Error in offset

(e) Control input 1

(f) Control input 2

Figure 4-23: Test results

4-3 Summary

In this chapter the performance of the proposed framework for the formation control is analysed. With the help of numerical simulations it is seen that the proposed framework using spatial-domain is capable achieving the desired objectives. However, with the available solver the framework still cannot guarantee RT performance, as it takes more time than the sampling interval to solve the problem for some instances. This can improved either by choosing a different solver or by further approximating the lateral offset. The next chapter outlines the research findings of the thesis work.

Chapter 5

Conclusions and Future work

The main goal of this thesis was to develop a formation control framework for path tracking application. The motivation to pursue the research work in this direction was discussed in Chapter 1 supported by the conducted literature survey on the state-of-the-art methods. In this chapter a summary of the overall work is provided, followed by recommendations for the future work.

5-1 Summary

Hovercrafts are nonlinear, underactuated and nonholonomic systems, making the path tracking control design a challenging task. In order to design a model based controller, a simplified mathematical model of the system is derived. This model is derived using Euler-Lagrange method, based on the assumption that a constant lift is available throughout the run. After obtaining a model which can capture all the dominant dynamics of the system, a feasible path for a given set of initial and final conditions is generated using Dubin's algorithm. The generated path acts as a reference path for the virtual vehicle (an additional vehicle in software other than member vehicles in formation). For all the member vehicles in formation, reference paths are generated based on the reference path of virtual vehicle. A path tracking controller should generate an equivalent trajectory by predicting the vehicle's position parametrized by the path and its heading. In order to do so, a new coordinate frame is introduced to represent the vehicle's position based on path (spatial/path coordinate frame). Finally, the system dynamics are reformulated to this domain from the conventional time-domain. Model Predictive Controller (MPC) being a computationally expensive task, the comparison study in Section 3-6-2 clearly shows that the proposed method reduces the number of states for each vehicle in formation when compared to the existing time-domain parametrized method. By reducing the number of states, the number of decision variables and their corresponding constraints are also reduced. This reduction in number of constraint evaluations enables the solver to compute the solutions well within the sampling time and achieve Real Time (RT) performance.

Various techniques are discussed to improve the performance of the proposed framework. One of these techniques is to approximate the lateral offset. By approximating the lateral offset for certain intervals the computation time has reduced. However, it is seen that the solver is still not capable of solving the problem within the sampling time for all iterations.

The study of different test cases in Section 4-2 shows that the controller is capable of maintaining the formation when an unknown disturbance acts on the system. It is also shown that the controller can drive the vehicles to formation when starting from a different initial conditions. However, the plot of computation time shows that the solver takes more time to compute the solutions, hindering the desired RT performance. In the case of vehicle failure, the controller is capable of stopping the other vehicles in the formation, however the position error exceeds the bounds of 5 cm.

The mathematical model is derived based on a basic assumption that a constant lift is available throughout the run. However, this may not be possible while implementing the controller on the practical set-up. In a battery powered vehicle, the current delivered to the motor reduces as the battery discharges. Thus, decreasing the overall thrust developed. As the lift reduces, the friction between the skirt and plane of motion increases, which changes the dynamics of the system.

The major drawback of using path coordinates to represent the vehicle's position is that if a vehicle is positioned either before the starting point or after the end point, the path coordinate frame has no valid representation for this set of positions. Hence, if an external disturbance acts on the system near these extreme points, it may drive the vehicle beyond the defined reference path.

5-2 Future recommendations

Based on the limitations discussed in the previous section, the following recommendations can be incorporated in the future work to achieve desired performance.

- **Approximation of d^S** The path for each vehicle in the formation is discontinuous when switching between *straight* and *curve* segments. These discontinuous points result in higher computational time for the solver. Hence, the path can be smoothed by approximating the lateral offset further.
- **Solver** The performance of the controller depends on the solver used. Different solvers can be used in future work that may guarantee RT performance for all test cases.
- **Constant lift** The assumption of constant lift can be still incorporated in the practical set-up by designing an additional feedback controller for the lift motor.
- **Decentralised controller** The centralised controller demands extensive communication which may not be feasible with higher number of vehicles in the practical set-up. Hence, the framework can be extended to a decentralised controller.

Appendix A

Dubin's path algorithm

A-1 Dubin's path

The Dubin's car is introduced by Laster Dubin in 1957. Car has only three controls *turn left maximum*, *turn right maximum* and *go straight*. These control are indicated by L, R and S. L and R correspond to turning the vehicle by minimum turning radius, it is achieved by maximum steering angle.

All paths are categorized by 6 combinations of controls, they are: LSL, RSR, LSR, RSL, LRL and RLR. These paths can be subdivided into two major class, CSC and CCC. Where C correspond to *curve*.

As Dubin's car travels, the curve traced out along the run is the path. Dubin's path is the shortest path traced by the car between any two end points. It is assumed that car travel at a constant velocity. Equations of Dubin's car is written as,

$$\begin{aligned}\dot{x} &= \cos(\phi)\\ \dot{y} &= \sin(\phi)\\ \dot{\phi} &= u\end{aligned}$$

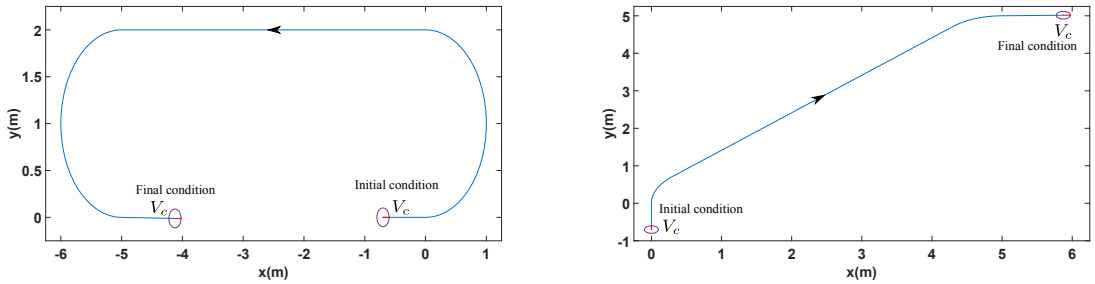
where, input $u = \{-1, 0, 1\}$. An optimization problem is framed as,

$$\begin{aligned}\min_{u} \quad & \int_{t=0}^{t_f} \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2 dt}\\ \text{subject to} \quad & \dot{\xi} = f(\xi, u, t)\\ & \xi(0) = \xi_0\\ & \xi(f) = \xi_f\end{aligned}$$

The resulting problem is a Mixed Integer Programming problem. There are other geometric methods to solve the problem which is discussed in following section.

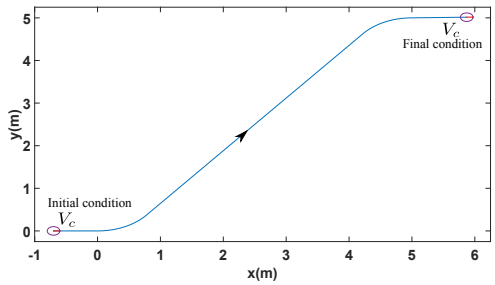
A-1-1 Curve-Straight-Curve (CSC)

As described earlier CSC consist of path with a turn followed by straight line and followed by another turn. It includes four out of six basic paths, $Left - Straight - Left$(LSL), $Right - Straight - Right$(RSR), $Left - Straight - Right$(LSR) and $Right - Straight - Left$ (RSL).

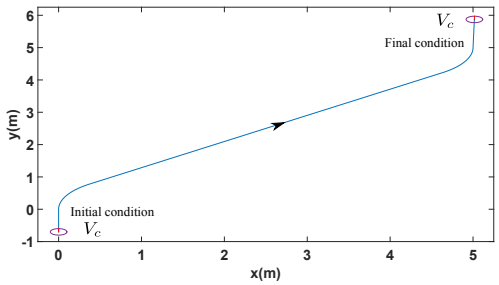


(a) LSL

(b) RSR



(c) LSR

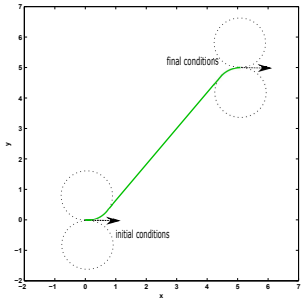


(d) RSL

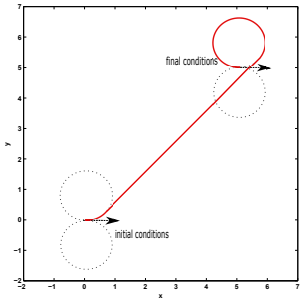
Figure A-1: Variations in CSC paths

As seen in Figure A-1 every CSC path consist of a line and two arcs, which are part of a circle. Hence, construction of a CSC path start with sketching of these circles and joining them with a common tangent, which corresponds to S part of CSC. To start with, two circles are drawn on either side of heading of initial and final position. These circles are of radius $r_{\min}$ and have initial and final position of hovercraft as tangent component. Next step is to draw a common tangent between two pairs of circles. For every pair of initial and final position there are eight possible tangents, of which only four are valid, as other four do not satisfy the heading condition for hovercraft. Figure A-2 shows the different tangents for each pair of circle.

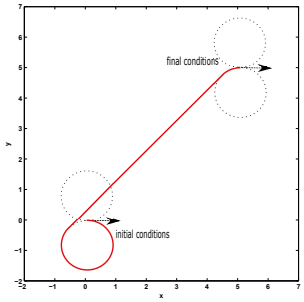
For any pair of initial and final positions there are four different CSC path drawn, out of these four paths the one minimum length is an optimal CSC, Dubin's path. Hence, the solution boils down to drawing of a common tangent for a pair of circles and choosing the resulting path with minimum length.



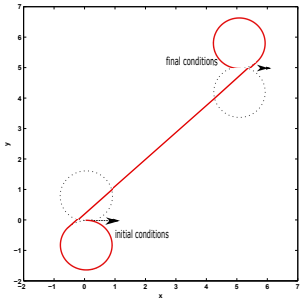
(a) LSL



(b) RSR



(c) LSR

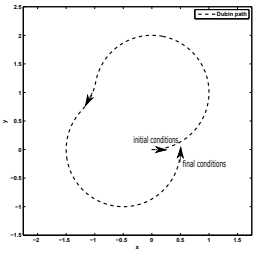


(d) RSL

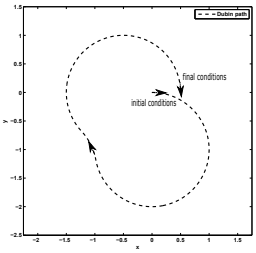
Figure A-2: Variations in CSC paths

A-1-2 Curve-Curve-Curve (CCC)

In case of initial and final positions being very close to each other and resulting circles overlap on each other, then it calls for a CCC path. It consists of two variations, $Left-Right-Left$ (LRL) and $Right-Left-Right$(RLR). Only difference between CCC and CSC is the straight line part is replaced by a curve. Figure A-3 shows different CCC paths, where it can be seen that the second curve has common point with the circles at initial and final points. This boils down to correctly placing the second circle with a common point with rest two.



(a) LRL



(b) RLR

Figure A-3: Variations in CCC path

Appendix B

Direct collocation method

In this chapter the direct collocation and method which is used to solve an Optimal Control Problem (OCP) is discussed. The direct collocation method reduces the continuous time problem into Nonlinear Program, i.e. Discretization of a problem. Once the problem becomes finites dimensional problem, it can be solved using various Nonlinear Program (NLP) technique like Sequential Quadratic Program (SQP) and Interior Point (IP) method. Hence the steps taken in solving a Optimal control Problem can be pointed out as follows [28],

- Transforming continues problem to finite dimensional problem
- Solve the resulting NLP using prescribed techniques
- Repeat the steps till required accuracy is reached

B-1 Direct methods of solving OCP

Two main methods to transcribe continues to finite dimensional problem are, Full collocation and direct shooting. In direct shooting discretization leads to solving boundary value problem. In this method guess for initial condition is made and states are predicted by propagating the differential equations and evaluate the error in boundary conditions, finally use NLP to reduce the error to zero. Shooting technique is widely used, as the transcription results in smaller number of variables. However, it faces a major disadvantage as any small change in initial condition may result in very large variations in final condition. This drawback has motivated to explore the capabilities of the direct collocation technique.

In direct collocation technique, the independent variable is discretized in to n intervals,

$$s = s_1 \leq s_2 \leq \cdots \leq s_N = s_f$$

Values of control input, $\mathcal{U}$ and states, ξ are found at these instances,

$$\mathcal{Y} = (\xi(s), \mathcal{U}(s))$$

Once the states and control inputs are known at discretized intervals, their values in between these intervals are predicted using interpolating polynomials, like Piecewise, Lagrange, Cubic, B-splines interpolating polynomials.

Once the appropriate parametrization is chosen, a NLP is formulated to solve for coefficients of the polynomials. The resulting NLP is solved using methods like SQP and IP. In the next section Full collocation method to solve OCP is explained.

B-1-1 Full Collocation

The first step in collocation method is to discretize continues variable into N points. Legendre-Gauss, Chebyshev-Gaussian-Labatto points describe how these points should be displaced among each other. The common feature is to transform variable s, $[s_0]$ to s_τ, $[-1, 1]$.

$$s_\tau = \left(\frac{s_f - s_0}{2}\right) s - 1$$

This results in transformation of problem to,

$$\begin{aligned}
\min \quad & \int_{-1}^{1} K_d d^2 + K_u \left(u - u_{ref}\right)^2 + K_v v^2 \\
\text{subject to} \quad & \frac{d\xi}{ds_\tau} = f(\xi, \mathcal{U}) \\
& \xi(-1) = \xi_{-1} && \text{(B-1)} \\
& \underline{u} \leq u(s_\tau) \leq \bar{u} \\
& \underline{d} \leq d(s_\tau) \leq \bar{d} \\
& \underline{\tau_1} \leq \tau_1(s_\tau) \leq \tau_1 \\
& \underline{\tau_2} \leq \tau_2(s_\tau) \leq \tau_2 && \text{(B-2)} \\
& && \text{(B-3)}
\end{aligned}$$

B-1-2 Legendre-Gauss-Lobatto points

As discussed in previous section, collocation method discretize the continuous problem to n points. These are the points at which system dynamics are evaluated, hence it requires a function to place n points optimally over an interval of $[-1, 1]$. These points are the roots of Legendre polynomial,

$$L_n(s) = \frac{1}{2^n n!} \frac{d^n}{ds^n} \left[\left(s^2 - 1\right)^n\right] \tag{B-4}$$

where, n is the number of points. Table B-1 shows first six Legendre polynomials and their roots.

Table B-1: Legendre quadrature

N	Legendre polynomial	Legendre-Gauss-labatto points
2	$\frac{1}{2}\left(3s^2-1\right)$	$\{-0.5774, 0.5774\}$
3	$\frac{1}{2}\left(3s^3-3s\right)$	$\{-0.7746, 0, 0.7746\}$
4	$\frac{1}{8}\left(35s^4-30s^2+3\right)$	$\{-0.8611, -0.3400, 0.3400, 0.8611\}$
5	$\frac{1}{8}\left(63s^5-70s^3+15s\right)$	$\{-0.9062, -0.5385, 0, 0.5385, 0.9062\}$
6	$\frac{1}{16}\left(231s^6-315s^4+105s^2-5\right)$	$\{-0.9325, -0.6612, -0.2386, 0.2386, 0.6612, 0.9325\}$

B-1-3 Chebyshev-Gaussian-Labatto Points (CGL)

CGL also called as Chebyshev nodes are used to distribute the collocation points. CGL are in the interval $(-1, 1)$ and are symmetrical about the origin. For any given natural number N Chebyshev nodes are computed by [29],

$$s_\tau = \cos\left(\frac{2k-1}{2n}\pi\right), k = 1, \ldots, n$$

First few CGL are illustrated in Table B-2

Table B-2: Chebyshev nodes

N	CGL
2	$\{-0.7071, 0.7071\}$
3	$\{-0.8600, 0, 0.8600\}$
4	$\{-0.9239, -0.3827, 0.3827, 0.9239\}$
5	$\{-0.9511, -0.5878, 0, 0.5878, 0.9511\}$
6	$\{-0.9659, -0.7071, -0.2588, 0.2588, 0.7071, 0.9659\}$

B-1-4 Lagrange Interpolating Polynomial

Next step in solving an OCP using collocation technique is to approximate the states, ξ and control input, $\mathcal{U}$ at other points based on their values found at collocation points. This approximation is done using many methods like, Lagrange Interpolation, Cubic polynomials, B-Splines curves.

Lagrange interpolating polynomial is a n^{th} degree polynomial $L(s)$ that passes through n points $\left((s_1, \xi_1), (s_2, \xi_2), \ldots, (s_n, \xi_n)\right)$ and is given by,

$$L_n(s) = \sum_{s_\tau=0}^{n} L_{s_\tau}\phi_\tau(s) \tag{B-5}$$

where,

$$\phi_\tau(s) = \prod_{k=0, k\neq\tau}^{n} \frac{s - s_k}{s_\tau - s_k}$$

When written explicitly,

$$\phi_\tau(s) = \frac{(s-s_1)(s-s_2)\dots(s-s_n)}{(s_0-s_1)(s_0-s_2)\dots(s_0-s_n)}L_{s_1} + \frac{(s-s_0)(s-s_2)\dots(s-s_n)}{(s_1-s_0)(s_1-s_2)\dots(s_1-s_n)}L_{s_2} + \\ \dots + \frac{(s-s_0)(s-s_1)\dots(s-s_{n-1})}{(s_n-s_0)(s_n-s_1)\dots(s_n-s_{n-1})}L_{s_n}$$

Lagrange interpolating polynomial is also written in terms of Legendre polynomial (B-4),

$$\phi_\tau(s) = \frac{1}{n(n+1)P_n(s)}\frac{(s^2-1)\dot{P_n}(s)}{s-s_\tau}, \tau = 0, 1, \dots, n \tag{B-6}$$

From (B-6) it is readily verified that,

$$\phi_\tau(s_j) = \begin{cases} 1 & \text{if } \tau = j \\ 0 & \text{if } \tau \neq j \end{cases} \tag{B-7}$$

and therefore by (B-7)

$$L_n(s_\tau) = L(s_\tau), \tau = 0, 1, \dots, n$$

Next requirement to solve the problem is to find the derivative of $L_n(s)$ in terms of n collocation points. Differentiating (B-5),

$$\dot{L_n}(s) = \sum_{\tau=0}^{n} DL(s_\tau)$$

where, D is given by,

$$D = \begin{cases} \frac{L_n(s_m)}{L_n(s_l)}\frac{1}{s_m-s_l} & m \neq l \\ -\frac{n(n+1)}{4} & m = l = 0 \\ \frac{n(n+1)}{4} & m = l = n \\ 0 & \text{otherwise} \end{cases}$$

B-1-5 OCP to NLP

The final step in the method is to transform the problem into equivalent NLP. This is explained in this section.

Using, (B-5) states, ξ and control input, $\mathcal{U}$ are approximated,

$$\xi_n(s) = \sum_{\tau=0}^{n} a_\tau \phi_\tau(s)$$

$$(U)_n(s) = \sum_{\tau=0}^{n} b_\tau \phi_\tau(s)$$

where, vectors a_τ and b_τ are decision variables Using approximation for states and control input, OCP (B-2) reduces to ,

$$\begin{aligned} \min \quad & \int_{-1}^{1} K_d d^2 + K_u \left(u - u_{ref}\right)^2 + K_v v^2 \\ \text{subject to} \quad & \dot{\xi_n}(s_\tau) = f(\xi_n, \mathcal{U}_n) \\ & \xi(s_0) = \xi_{s_0} \qquad \text{(B-8)} \\ & \underline{u} \leq u(s_\tau) \leq \bar{u} \\ & \underline{d} \leq d(s_\tau) \leq \bar{d} \\ & \underline{\tau_1} \leq \tau_1(s_\tau) \leq \tau_1 \\ & \underline{\tau_2} \leq \tau_2(s_\tau) \leq \tau_2 \qquad \text{(B-9)} \\ & \qquad \text{(B-10)} \end{aligned}$$

Next main step is to approximate the integral formula in cost function, this is done using Gauss-Labatto integration formula by which,

$$\int_{-1}^{1} K_d d^2 + K_u \left(u - u_{ref}\right)^2 + K_v v^2 = \sum_{\tau=0}^{n} K_d d_n(s_\tau)^2 + K_u \left(u_n(s_\tau) - u_{ref}\right)^2 + K_v v_n(s_\tau)^2 w_\tau$$

where, the weights w_τ are given by,

$$w_\tau = \frac{2}{n(n+1)} \frac{1}{L_n(s_\tau)^2}, \quad \tau = 0, 1, \ldots, n$$

Using above discretization and approximation OCP is approximated to following NLP,

$$\begin{aligned} \min_{a_k, b_k} \quad & \sum_{\tau=0}^{n} K_d d_n(s_\tau)^2 + K_u \left(u_n(s_\tau) - u_{ref}\right)^2 + K_v v_n(s_\tau)^2 w_\tau \\ \text{subject to} \quad & f(a_k, b_k) - D = 0 \\ & \xi(s_0) = \xi_{s_0} \qquad \text{(B-11)} \\ & \underline{u} \leq u(s_\tau) \leq \bar{u} \\ & \underline{d} \leq d(s_\tau) \leq \bar{d} \\ & \underline{\tau_1} \leq \tau_1(s_\tau) \leq \tau_1 \\ & \underline{\tau_2} \leq \tau_2(s_\tau) \leq \tau_2 \quad \tau = 0, 1, \ldots, n \qquad \text{(B-12)} \end{aligned}$$

Bibliography

[1] Y. Chen and Z. Wang, "Formation control: a review and a new consideration," in *Intelligent Robots and Systems*, pp. 3181–3186, IEEE, 2005.

[2] J. G. Bender, "An overview of systems studies of automated highway systems," *IEEE Transactions on vehicular technology*, vol. 40, no. 1, 1991.

[3] M. Martin, P. Klupar, S. Kilberg, and J. Winter, "Techsat 21 and revolutionizing space missions using microsatellites," 2001.

[4] A. Yufka, O. Parlaktuna, and M. Ozkan, "Formation-based cooperative transportation by a group of non-holonomic mobile robots," in *Systems Man and Cybernetics (SMC)*, pp. 3300–3307, IEEE, 2010.

[5] H. Bai and J. T. Wen, "Cooperative load transport: a formation-control perspective," *Robotics*, vol. 26, no. 4, pp. 742–750, 2010.

[6] D. Zhaohui, W. Min, and C. Xin, "Multi-robot cooperative transportation using formation control," in *Control Conference*, pp. 346–350, IEEE, 2008.

[7] K. Aslan, "Amphibian - transport over land and water," 2015.

[8] V. Kumar, J. Desai, and J. Ostrowski, "Control of formations for multiple robots," in *IEEE International Conference on Robotics and Automation*, (Leuven, Belgium), May 1998.

[9] A. Das, R. Fierro, V. Kumar, J. Ostrowski, J. Spletzer, and C. J. Taylor, "Vision based formation control of multiple robots," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 813–825, October 2002.

[10] J. Chen, D. Sun, J. Yang, and H. Chen, "A leader-follower formation control of multiple nonholonomic mobile robots incorporating receding-horizon scheme," *The International Journal of Robotics Research*, 2009.

[11] M. A. Lewis and K.-H. Tan, "High precision formation control of mobile robots using virtual structures," *Autonomous Robots*, vol. 4, no. 4, pp. 387–403, 1997.

[12] D. Maithripala, J. Berg, D. Maithripala, and S. Jayasuriya, "A geometric virtual structure approach to decentralized formation control," in *American Control Conference (ACC)*, pp. 5736–5741, IEEE, 2014.

[13] D. Gu and H. Hu, "A model predictive controller for robots to follow a virtual leader," *Robotica*, vol. 27, no. 06, pp. 905–913, 2009.

[14] R. Olfati-Saber and R. M. Murray, "Distributed cooperative control of multiple vehicle formations using structural potential functions," in *IFAC World Congress*, pp. 346–352, Citeseer, 2002.

[15] L. Barnes, M. Fields, and K. Valavanis, "Unmanned ground vehicle swarm formation control using potential fields," in *Control & Automation, 2007. MED'07. Mediterranean Conference on*, pp. 1–8, IEEE, 2007.

[16] T. I. Fossen, *Guidance and control of ocean vehicles*. 1994.

[17] C. A. Woolsey, "Review of marine control systems: Guidance, navigation, and control of ships, rigs and underwater vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 3, pp. 574–575, 2005.

[18] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[19] L. Meirovitch, *Dynamics and control of structures*. John Wiley & Sons, 1990.

[20] A. P. Aguiar, L. Cremean, and J. P. Hespanha, "Position tracking for a nonlinear underactuated hovercraft: Controller design and experimental results," in *Decision and Control. Proceedings.*, vol. 4, pp. 3858–3863, IEEE, 2003.

[21] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006. Available at http://planning.cs.uiuc.edu/.

[22] Y. Gao, A. Gray, J. V. Frasch, T. Lin, E. Tseng, J. K. Hedrick, and F. Borrelli, "Spatial predictive control for agile semi-autonomous ground vehicles," in *Proceedings of the 11th International Symposium on Advanced Vehicle Control*, 2012.

[23] M. P. Do Carmo and M. P. Do Carmo, *Differential geometry of curves and surfaces*, vol. 2. Prentice-hall Englewood Cliffs, 1976.

[24] C. De Boor, "A practical guide to splines," *Mathematics of Computation*, 1978.

[25] K. Holmström, A. O. Göran, and M. M. Edvall, "User's guide for tomlab/snopt1," 2008.

[26] K. Holmström and A. O. Göran, "User's guide for tomlab v3. 2.11," 2002.

[27] P. E. Rutquist and M. M. Edvall, "Propt-matlab optimal control software," *Tomlab Optimization Inc*, vol. 260, 2010.

[28] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*, vol. 19. Siam, 2010.

[29] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. No. 55, Courier Corporation, 1964.

Glossary

List of Acronyms

CoG	Center of Gravity
DoF	Degrees of Freedom
OCP	Optimal Control Problem
RT	Real Time
NLP	Nonlinear Program
SQP	Sequential Quadratic Program
IP	Interior Point
CGL	Chebyshev-Gaussian-Labatto Points
CCC	Curve-Curve-Curve
CSC	Curve-Straight-Curve
LSL	Left-Straight-Left
RSR	Right-Straight-Right
LSR	Left-Straight-Right
RSL	Right-Straight-Left
LRL	Left-Right-Left
RLR	Right-Left-Right
SNOPT	Sparse Nonlinear OPTimizer
NPSOL	Nonlinear Programming Systems Optimization Laboratory
MPC	Model Predictive Controller

SISO Single Input Single Output

MIMO Multiple Input Multiple Output