A Laguerre-based Distributed Nonlinear Model Predictive Control Scheme for Dynamic Obstacle Avoidance on Multi-Rotor UAVs

1st Oscar J. Gonzalez V.
Centre for Autonomous and Cyber-Physical Systems
Cranfield University
Cranfield, UK
oscar.gonzalez-villarreal@cranfield.ac.uk

2nd Antonio Tsourdos
Centre for Autonomous and Cyber-Physical Systems
Cranfield University
Cranfield, UK
a.tsourdos@cranfield.ac.uk

***Abstract*—This paper presents a Distributed NMPC approach for the problem of Dynamic Obstacle Avoidance based on Laguerre Polynomials combined with potential fields. The proposed approach aims at tackling the computational burden by using a combination of different methods to achieve real-time performance. Moreover, the approach benefits from being able to represent the projected paths of each UAV using a significantly reduced amount of information, thus reducing the bandwidth required for a practical implementation. The paper presents a complete derivation of the proposed approach along with an algorithm for its implementation, and discusses aspects specifically related to Multi-Rotor UAV models which can help to reduce the computational burden even further. The performance of the proposed approach is demonstrated through a simulation of 50 UAVs sharing common airspace.**

***Index Terms*—Nonlinear Model Predictive Control (NMPC), Parametric Curves, Distributed Control Systems, Obstacle Avoidance, Potential Field, Unmanned Aerial Vehicles (UAVs)**

I. Introduction

Over the recent years, Unmanned Aerial Vehicles (UAVs), sometimes called drones, have become increasingly popular thanks to their wide range of potential applications such as aerial inspection, search and rescue, package deliveries, autonomous agriculture, and so on. However, although industry is looking forward to introduce them into society, many technical challenges still prevent their safe operation which makes them fundamentally unable to be completely trustworthy. One of these challenges is that of path planning and collision avoidance. Unlike conventional industrial robots which operate in a relatively controlled/known environment for which solutions can be found prior to them executing their tasks, UAVs must be able to operate safely in complex and rapidly evolving dynamic environments. This requires them to be able to make accurate and complex decisions in real time, adjusting their paths according to the current (and in some cases future) state of the environment. Additionally, they often must be able to execute some relatively complex task whilst ensuring this safety property at all times which pose an additional layer of difficulty.

Special thanks to the Trustworthy Autonomous Systems Node in Security

Although there exist a wide range of methods to tackle the obstacle avoidance problem, Nonlinear Model Predictive Control (NMPC) has recently gained a significant amount of interest, in part thanks to the recent advances in computational power. Its popularity comes mainly from its generic ability to tackle complex dynamical systems by solving Optimal Control Problems (OCPs) in real time. It's basic operation relies on finding a plan over a given prediction horizon at each sampling time, but only applying the first part of the sequence, thus giving the vehicle a natural "re-planning" ability [1]. Various works based on this type of approach have been proposed in recent years. Authors from [2] discussed how the use of NMPC can integrate both path planning and tracking in a systematic way, something which is often tackled separately. The work from [3] presented a trajectory planning algorithm under positioning accuracy constraints. In [4], a sensor-based task constrained MPC approach is proposed which uses information from a dual-depth camera on a robotic system for motion planning. Similarly, work from [5] proposed a path planning and obstacle avoidance approach for UAVs using information from a depth camera. To obtain real-time performance, work from [6] proposed an auto-generated NMPC algorithm for real-time obstacle avoidance of ground vehicles.

Most of the studies presented above have been done considering a single vehicle or "agent" interacting with a dynamic environment. However, the challenge is significantly increased when a multi-agent system is considered, in particular when considering how they interact and share information. Work from [7] presented a multi-robot distributed NMPC formation combining potential field methods with a modified A* algorithm to avoid singularities. In [8], a distributed MPC is used for formation of multi-agent systems with collision avoidance and agreement constraints. Most of these works rely on ad-hoc communication between the UAVs. However, an interesting alternative was discussed in [9]; a Legible MPC approach for autonomous driving where the control system is proposed to make its control actions "legible" by an external observer, which could serve as a "decentralised backup strategy" for UAVs in case of loss of communication.

This paper presents a Laguerre-based Distributed NMPC which tackles the computational burden by using input-parameterised optimal solutions combined with a potential field approach to tackle the dynamic obstacle avoidance problem. Although works from [1], [10] have presented similar use of parametric curves, the proposed approach differs from all the above in the sense that it is implemented in a distributed multi-uav fashion. Moreover, we present a multi-rotor UAV scenario where by using simplified dynamics, the computational burden can be reduced further, thus enhancing its potential for real-time applications. The efficacy of the proposed approach is presented with simulations of up to 50 UAVs sharing common airspace whilst satisfying the safe-distance constraints.

The paper is organised as follows: Section II presents the details of the proposed approach in 3 subsections, namely: II-A Input-Parameterised NMPC, II-B Dynamic Obstacle Avoidance and II-C Distributed NMPC, and provides a general algorithm for its implementation in section II-D. Section III presents a simulation of the proposed applied to a simplified UAV model presented in III-A, and discusses its impact in the computational burden of the proposed algorithms. Finally, section IV emits conclusions and discusses future work.

II. Laguerre-based Distributed Nonlinear Model Predictive Control

In this section we will present the proposed approach by introducing 3 main sub-parts, namely: input-parameterised NMPC, dynamic obstacle avoidance and distributed NMPC optimisation. For reference, the approach will follow closely the notation and derivations presented in [11].

A. Input-Parameterised Nonlinear Model Predictive Control

Let us begin by introducing the basic objective function representing the input-parameterised NMPC defined by:

$$J = (X_r - \hat{X})^T Q (X_r - \hat{X}) + (U_r - \hat{U})^T R (U_r - \hat{U}) \quad \text{(1a)}$$

$$st. \quad \hat{x}_k = x_0 \quad \text{(1b)}$$

$$\hat{x}_{k+i} = f(\hat{x}_{k+i-1}, \hat{u}_{k+i-1}) \quad \forall i = [1, ..., N_p] \quad \text{(1c)}$$

$$\hat{U} = \mathbb{N}\hat{\eta} \quad \text{(1d)}$$

$$X_{min} \leq \hat{X} \leq X_{max} \quad \text{(1e)}$$

$$U_{min} \leq \hat{U} \leq U_{max} \quad \text{(1f)}$$

where $Q > 0 \in \mathbb{R}^{N_p n_x \times N_p n_x}$ and $R > 0 \in \mathbb{R}^{N_p n_u \times N_p n_u}$ are positive definite matrices for penalizing state and input errors, respectively, with Q typically selected as a block diagonal matrix ($Q = blkdiag([q_{k+1}, q_{k+2}, \ldots, q_{k+N_p}])$), q_{k+N_p} sometimes referred to as the terminal weight, and R typically selected as a diagonal matrix with a constant value ($R = r_u I^{N_p n_u \times N_p n_u}$); $X_r = [x_{r_{k+1}}^T, x_{r_{k+2}}^T, \cdots, x_{r_{k+N_p}}^T]^T \in \mathbb{R}^{N_p n_x}$, $\hat{X} = [\hat{x}_{k+1}^T, \hat{x}_{k+2}^T, \cdots, \hat{x}_{k+N_p}^T]^T \in \mathbb{R}^{N_p n_x}$, $U_r = [u_{r_k}^T, u_{r_{k+1}}^T, \cdots \cdots, u_{r_{k+N_p-1}}]^T \in \mathbb{R}^{N_p n_u}$, $\hat{U} = [\hat{u}_k^T, \hat{u}_{k+1}^T, \cdots, \hat{u}_{k+N_p-1}^T]^T \in \mathbb{R}^{N_p n_u}$ are future state references, states, input references and inputs column-vectors, respectively; (1b) is the initial condition; (1c) are the state dynamics; (1d) is the input-parameterisation equality with $\hat{\eta} \in \mathbb{R}^{N_\eta n_u}$ begin the parameterised decision vector, and $\mathbb{N} \in \mathbb{R}^{N_p n_u \times N_\eta n_u}$ begin the input-parameterisation matrix; and (1f) and (1e) are the inputs and state constraints, with $X_{max}/X_{min} \in \mathbb{R}^{N_p n_x}$ and $U_{max}/U_{min} \in \mathbb{R}^{N_p n_u}$.

Note that this basic objective function may be modified in later stages when the obstacle avoidance terms are included, as well as when the optimisation is set in a distributed fashion, or in case terminal conditions are required. Moreover, further details related to the input-parameterisation equation (1d) will be discussed later in this section.

By taking a single-shooting approach, simulating the system over a "nominal/guessed" input vector $\bar{U} \in \mathbb{R}^{N_p n_u}$ to obtain a "nominal" state trajectory $\bar{X} \in \mathbb{R}^{N_p n_x}$ and linearising over the resulting trajectories using first-order Taylor Series expansion, the following models can be derived:

$$\hat{U} = \mathbb{N}\eta = \bar{U} + \delta\hat{U} \quad \text{(2a)}$$

$$\hat{X} = \bar{X} + H\delta\hat{U} = \bar{X} + H(\mathbb{N}\eta - \bar{U}) \quad \text{(2b)}$$

where $H \in \mathbb{R}^{N_p n_x \times N_p n_u}$ is defined as:

$$H = \begin{bmatrix} h_{1,1} & 0 & \cdots & 0 \\ h_{2,1} & h_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_{N_p,1} & h_{N_p,2} & \ldots & h_{N_p,N_p} \end{bmatrix} \quad \text{(3)}$$

with the inner terms defined by the recursive expression:

$$h_{k,j} = \begin{cases} B_{k-1} & k = j \\ A_{k-1} h_{k-1,j} & k > j \end{cases} \quad \text{(4)}$$

and:

$$A_k = \frac{\partial f(\hat{x}_k, \hat{u}_k)}{\partial \delta \hat{x}_k} \quad B_k = \frac{\partial f(\hat{x}_k, \hat{u}_k)}{\partial \delta \hat{u}_k} \quad \text{(5)}$$

By substituting equations (2a) and (2b) in cost function (1), rearranging in terms of the decision variable ($\hat{\eta}$), and disregarding constant terms, the following standard Quadratic Program (QP) can be obtained:

$$J = \hat{\eta}^T E \hat{\eta} + 2 f^T \hat{\eta} \quad st. \quad M\hat{\eta} \leq \gamma \quad \text{(6a)}$$

$$E = \mathbb{N}^T (H^T Q H + R) \mathbb{N} \quad \text{(6b)}$$

$$f = -\mathbb{N}^T \left[H^T Q (X_r - \bar{X} - H\bar{U}) - R(\bar{U} - U_r) \right] \quad \text{(6c)}$$

$$M = \begin{bmatrix} H\mathbb{N} \\ -H\mathbb{N} \\ \mathbb{N} \\ -\mathbb{N} \end{bmatrix} \quad \gamma = \begin{bmatrix} X_{max} - \bar{X} + H\bar{U} \\ -(X_{min} - \bar{X} + H\bar{U}) \\ U_{max} \\ -U_{min} \end{bmatrix} \quad \text{(6d)}$$

where $E \in \mathbb{R}^{N_\eta n_u \times N_\eta n_u}$ is the Hessian; $f \in \mathbb{R}^{N_\eta n_u}$ is the linear term; and M and γ are the constraints matrix and vector, respectively.

An important term to consider here is the matrix multiplication $H_\mathbb{N} = H\mathbb{N}$ which can be used to simplify some of the expressions above, as well as to reduce some of the computational burden and memory usage as discussed in [11]. This results in an expression of the form:

$$H_\mathbb{N} = \begin{bmatrix} (h_\mathbb{N})_1^T & (h_\mathbb{N})_2^T & \cdots & (h_\mathbb{N})_{N_p}^T \end{bmatrix}^T \quad \text{(7)}$$

with:

$$(h_{\mathbb{N}})_k = \begin{cases} B_{k-1}\mathbb{N}_{k-1} & k=1 \\ A_{k-1}(h_{\mathbb{N}})_{k-1} + B_{k-1}\mathbb{N}_{k-1} & k>1 \end{cases} \quad (8)$$

where $\mathbb{N}_{k-1}$ is the input-parameterisation matrix related to the k^{th} time-step.

By following a Sequential Quadratic Programming (SQP) approach and repeatedly solving the aforementioned QP to obtain a "new nominal/guess", the solution will eventually lead to the local optimal around the starting solution. Further details regarding starting solution will be discussed later in this section. Moreover, it should be noted that although the proposed approach in this paper relies on a single-shooting optimisation, an equivalent multiple-shooting formulation can be obtained as derived in [11].

Once the solution for the parameterised vector ($\hat{\eta}$) is found, the original decision variable can be recovered using the original equality (1d). Only the first input is sent to the system and the process is repeated in the following time-step which is the well known receding horizon strategy. The resulting optimum can then be shifted to "hot-start" the initial guess at the next sampling time as performed in the Real-Time Iteration (RTI) Scheme [12]. Finally, to achieve real-time performance, only a single SQP iteration is typically computed [12].

1) Laguerre Polynomials: In this work, the well-known Laguerre Polyonimals were used as the input-parameterisation basis function which have been studied in various works such as [11], [13], [14]. It is important to note that although the current work focuses on their particular implementation, any type of input-parameterisation such as Kautz Polynomials [15], Chebyshev Polynomial [11], B-Spline curves [10] and Move-Blocking [11] may be used. An important benefit of them are their simple recursive nature which results in them having recursive feasibility properties [14]. Moreover, their decaying properties may have an important part when defining obstacle avoidance trajectories given they can essentially be used to represent a momentary deviation which will quickly dissipate, thus giving a predictable behaviour after the obstacle has been avoided or in case of loss of communications between the UAVs. Indeed, work from [15] discussed how they can be combined with optimal feedback control to limit their effects only for constraint handling. Lastly, they have been shown to give competitive performance in [11] when appropriately tuned whilst ensuring smooth control actions.

Considering linear independence between the parameterisation of each input, ie. each input of the system having its own polynomial, as opposed to other type of input-parameterisations such as control allocation [16], the Laguerre Polynomials of each input at each time step are given by:

$$\begin{bmatrix} L(1) \\ L(2) \\ \vdots \\ L(N_L) \end{bmatrix}_{k+1} = \begin{bmatrix} a_L & 0 & \cdots & 0 \\ \beta & a_L & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ (-a_L)^{N_L-2}\beta & \ddots & \ddots & a_L \end{bmatrix} \begin{bmatrix} L(1) \\ L(2) \\ \vdots \\ L(N_L) \end{bmatrix}_k \quad (9)$$

where a_L is the decay-rate, typically selected based on the desired settling time, $\beta = (1 - a_L^2)$ and N_L is the number of Laguerre coefficients.

By taking $L_0 = \sqrt{\beta}\begin{bmatrix} 1 & -a_L & \cdots & (-1)^{N_L-1}a_L^{N_L-1} \end{bmatrix}^T$ as an initial condition and iterating system (9) forward N_p times, the Laguerre Polynomials of each input can then be combined appropriately to give an input-parameterisation matrix ($\mathbb{N}$) of the form:

$$\mathbb{N} = \left[\begin{array}{ccc} (L_0^1)^T & 0 & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & 0 & (L_0^{n_u})^T \\ \hline \vdots & \vdots & \vdots \\ \hline (L_{N_p-1}^1)^T & 0 & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & 0 & (L_{N_p-1}^{n_u})^T \end{array}\right] = \left[\begin{array}{c} \mathbb{N}_0 \\ \hline \vdots \\ \hline \mathbb{N}_{N_p-1} \end{array}\right] \quad (10)$$

where the notation (L_k^i) indicates the input-parameterisation of the i^{th} input, at the k_{th} time step. Note that a different decay-rate a_L may be selected for different inputs if desired.

We will discuss later how this type of input-parameterisation can have an impact for communications when implementing the proposed approach in a distributed fashion.

B. Dynamic Obstacle Avoidance

One of the benefits of predictive controllers is their inherent ability to anticipate future events to take appropriate actions which often result in smoother control actions when compared to non-predictive schemes. Although there exist a wide range of approaches to tackle this problem, two of the main methods that have been used for it are potential fields such as [7], and distance constraints such as [8], each one having their own advantages and disadvantages. As an example, the solution from a potential field approach will often be sub-optimal when compared to distance constraints given they have an overall effect on the cost function even when the obstacle might not even violate the safe-distance constraint. On the other hand, solving a constrained problem might increase the computational burden significantly.

As we are looking to propose a real-time capable approach, this paper uses a potential field approach. Although one might choose a squared potential field term as in [7], this work uses a linear potential term described by:

$$\hat{P}_k^i = \frac{k_{pot}}{(\hat{d}_k)_i - d_{min}} \quad (11)$$

where P_k^i is the potential field of the i_{th} obstacle at time k, k_{pot} is a tuning weight to adjust the potential field's effect on the overall cost, $(\hat{d}_k)_i = \sqrt{(\hat{x}_k - \hat{x}_{obs_k^i})^2 + (\hat{y}_k - \hat{y}_{obs_k^i})^2 + (\hat{z}_k - \hat{z}_{obs_k^i})^2}$ is the distance between the UAV and the i^{th} obstacle at time-step k, and d_{min} is the minimum safe-distance that must be maintained to the obstacle, typically selected as the summation of the safe-radius of both UAVS, eg. $d_{min} = r_{UAV_1} + r_{UAV_2}$.

Note that this term can include any type of obstacles as long as they have a predictable trajectory over the prediction horizon. As an example, one might include the trajectory of an incoming projectile following a simple parabolic trajectory.

Having defined this, the overall cost (1) can then be modified to include this term as:

$$J_{total} = J + J_{pot} = J + \min \sum_{i=1}^{N_{obs}} \left(\sum_{k=1}^{N_p} P_k^i \right) \tag{12}$$

To apply this, we can formulate an output linearised prediction model of the potential fields of each obstacle using the nominal trajectory ($\bar{X}$) obtained with the nominal input ($\bar{U}$) as:

$$\hat{Y}_i = \bar{Y}_i + (H_{\mathbb{N}})_i \hat{\eta} \tag{13}$$

where $(H_{\mathbb{N}})_i$ is defined as:

$$(H_{\mathbb{N}})_i = \begin{bmatrix} C_{1,i} & 0 & \cdots & 0 \\ 0 & C_{2,i} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & C_{N_p,i} \end{bmatrix} H_{\mathbb{N}} \tag{14}$$

with $C_{k,i} = \frac{\partial P_k}{\partial \hat{x}_k}$ being the output matrix.

Because the overall term is only of first order nature, the effect of each obstacle at each time-step can be included by adding the columns of $(H_{\mathbb{N}})_i^T$ for all obstacles $i = [1, N_{obs}]$ to the linear term (f), resulting in:

$$f_{total} = f + \sum_{i=1}^{N_{obs}} \left[(H_{\mathbb{N}})_i^T \mathbb{1}^{N_p} \right] \tag{15}$$

with $\mathbb{1}^{N_p}$ being a vector of ones of size N_p.

This will have an impact on reducing the overall computational burden of the proposed approach which will be discussed later in the simulation section III.

An important thing to notice here is that because the NMPC relies on continously "guessing" and optimising around the the nominal trajectories, there exist the possibility that a "new" obstacle with a imminent collision path is introduced in the prediction horizon and enters the "dangerous-zone" depicted in yellow in figure 1.

This can ultimately cause a number problems, especially when linearising around it. As an example, if the obstacle comes close to the limit d_{min}, the solution might become numerically unstable, or make the linearised model consider a very agressive maneouver is required. On the other hand, if the obstacle happens to pass the limit, ie. predicting a collision, the minimisation of the potential field term would push the solution to go to the limit itself, rather than pushing it away from it. Finally, although the exact expression for $C_{k,i}$ matrix will depend on the arrange of the state vector x_k and the respective 3D (x, y, z) axis, the linarisation expression for a given axis at a given time-step, eg (w_k) will be given by:

$$(C_{k,i})_{w_k} = \frac{-k_{pot}(w_k - w_{obs_k^i})}{(d_k)_i[(d_k)_i - d_{min}]^2} = \frac{-k_{pot}(w_k - w_{obs_k^i})}{(d_k)_i(\alpha_k)_i^2} \tag{16}$$

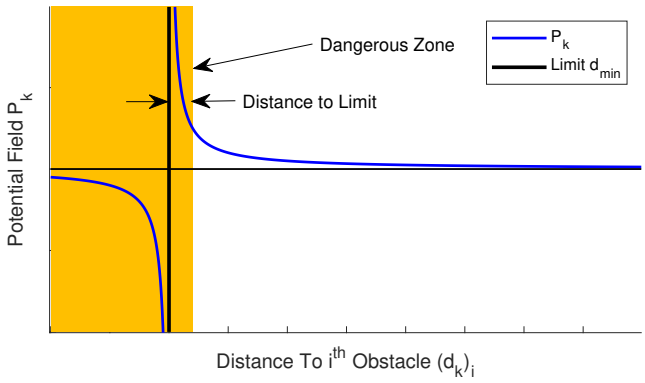

Fig. 1. Potential Field Graph depicting the "dangerous-zone" in yellow.

From this we can see that both the distance $(d_k)_i$, and the distance to the limit $(\alpha_k)_i = (d_k)_i - d_{min}$ are required to remain non-zero positive.

To tackle the aforementioned problems, we propose to impose a limit on these expressions by:

$$(d_k)_i^* = \begin{cases} (d_k)_i & (d_k)_i \geq \epsilon_{min} \\ \epsilon_{min} & (d_k)_i < \epsilon_{min} \end{cases} \tag{17}$$

$$(\alpha_k)_i^* = \begin{cases} (\alpha_k)_i & (\alpha_k)_i \geq \epsilon_{min} \\ \epsilon_{min} & (\alpha_k)_i < \epsilon_{min} \end{cases} \tag{18}$$

Thus ensuring that the obstacle is always "pushed" away, even if entering the "dangerous" zone.

This approach can also be seen as an equivalent to using soft-constraints when implementing the alternative constraints-based approach. Indeed, it is important to remember that the feasibility of the problem itself depends not only on the manoeuvering capabilities of the UAV being controlled, but also on the nature of the trajectories of the obstacles themselves, eg. their overall speeds.

Lastly, the user might use different safe distances d_{min} and weights k_{pot} for different obstacles and even at different time-steps k, eg. to capture their uncertainty and provide a more robust solution. Additionally, the user may consider a maximum distance d_{max} as an "activation" function, similar to [7] to select the distance at which the potential field of a given obstacle starts to take an effect.

C. Distributed Model Predictive Control

Although having a centralised MPC scheme to control multiple UAV vehicles may result in a relatively simple control structure to implement, it has a number disadvantages. As an example, by having a single node controlling the other nodes over a network, the overall Mission becomes vulnerable to having a single point of attack that would cause total failure. Instead, by having a distributed framework, the system has an inherent modularity which naturally gives it more flexibility and robustness against networked cyber-security attacks, as well as makes the design of the local controllers much simpler. It is for this reason that all the obstacles in (11) are considered

an "external" entity rather than a decision variable. Indeed, it is this last point which makes both types of approaches to differ where in a centralised approach, the agreement between each UAV would be done "simultaneously" at the time of the optimisation, in the sense that the "master" controller would dictate UAV_1 and UAV_2 to move in X and Y manner as appropriate at the exact same time.

In contrast, the distributed MPC approach relies on communication of information, including their predicted paths to achieve an agreement between the UAVs, which naturally can lead to having delays when sharing the latest projected paths. An alternative to this would be to have a completely decentralised approach with no communication where by observing other UAVs movement in real-time, one could "infer" their projected paths as recently discussed in [9] which introduced the concept of "legibility" in the design of MPC control systems. At this point it should be noted that regardless of whether it is a decentralised or distributed approach, the decisions being made by one UAV at a given time will typically consider the other UAVs will not change their current projected paths which will often lead to an overall sub-optimal solution when compared to the centralised approach. At the end, the implementation of distributed MPC it is often a matter of communication setup and assumptions.

In this paper, we assume that the information between the UAVs will be shared with a synchronous unit time-step delay, in the sense that each UAV will be aware of what the other UAV planned 1 time-step ago. The approach might be able to compensate multiple/time-varying time-steps delays for each UAV separately and in an asynchronous manner with very little adjustments. Moreover, the time-step delay itself could be used to incorporate robust/uncertainty considerations, eg. by adjusting the safe-distance to the obstacle according to the "age" of the information that was provided, ie. how long ago was the path provided. Nonetheless, both of these aspects were outside the scope of this paper. The specific details of the implementation will be discussed further in the final algorithm provided in section II-D.

On the other hand, a key benefit of using the proposed Laguerre-based input-parameterisation in this paper is the amount of information to be transmitted over the communication channels to the other UAVs when compared to other approaches such as the standard NMPC, in particular when considering them in a distributed optimisation fashion. To give an example of this, assuming the projected path of a given UAV is defined by a state vector composed of 3 positions ($x_k = [x, y, z]^T$) and has velocity/control vector ($u_k = [v_x, v_y, v_z]^T$) with a prediction horizon of $N_p = 100$, it results in its path being defined by 1212 bytes (assuming a float variable is used). In contrast, having $N_L = 3$ Laguerre polynomials per input results in its path being defined by 48 bytes, thus reducing nearly 25 times the required bandwidth.

Another important feature is that the projected path obtained by the Laguerre polynomials will always decay to zero as seen in figure 4.1 of [11], thus resulting in a predictable path in case of loss of communication from a given UAV. Although this might be directly implemented in the velocity vector which would result in the vehicle eventually becoming completely "static", in this paper we apply the proposed approach on the force/acceleration vector, which would result in the vehicle eventually having a constant velocity vector. We will describe this further in section III-A.

D. Algorithms and Methodology Summary

To give an overall summary of the proposed approach, we provide algorithm 1 which describes the required steps written in "pseudo-code" to be implemented by the local controllers of each UAV. To achieve this, we assume the algorithm receives some basic information described in "Data", such as: current state vector, nominal control vector obtained in the previous time step, as well as the state and parameterised control vectors of other UAVs obtained in the previous time steps. Other parameters related to tuning and optimisation setup such as weights, constraints and relevant sizes are also included.

The algorithm is divided in 4 main sections, namely: state/data alignment (lines 2-4), prediction and linearisation (lines 5-8), optimisation (lines 9-11) and communication (line 12). A key part of the algorithm is the "shifting" performed in line 4, which is typically done by duplicating the last input as described in [11], although other types of shifting might be used, eg. by imposing the continuity of the Laguerre polynomials beyond the prediction horizon. Indeed, it is this part which could be modified to handle multiple time-step delays. Lastly, the algorithm assumes that the input-parameterisation ($\mathbb{N}$) of all UAVs is the same, and that the potential field terms k_{pot} and d_{min} remain constant. Nonetheless, they can be adjusted to fit a given scenario, eg. if vehicles of different sizes were used and/or different parameterisations were required.

Algorithm 1: Laguerre-based Local NMPC Controller for Dynamic Obstacle Avoidance

Data: $x_k, \bar{U}_{k-1}, x^i_{obs_{k-1}}, \hat{\eta}^i_{obs_{k-1}}, n_x, n_u, n_\eta, N_p, Q, R$
$X_{min}, X_{max}, U_{min}, U_{max}, a_L, N_L, \mathbb{N}, k_{pot}, d_{min}, \epsilon$

1 **begin**

/* State/Data alignment */

2 Decompress $\bar{U}^i_{obs_{k-1}} = \mathbb{N}\hat{\eta}_{obs_{k-1}} \quad \forall i = [1, N_{obs}]$

3 Predict $\hat{x}^i_{obs_k}$ using $\bar{u}^i_{obs_{k-1}} \quad \forall i = [1, N_{obs}]$

4 Shift $\bar{U}_{k-1}$ and $\bar{U}^i_{k-1} \quad \forall i = [1, N_{obs}]$ forward

/* Prediction and Linearisation */

5 Predict $\bar{X}$ using $\bar{U}_k$ and $x_0 = x_k$

6 Predict $\bar{X}^i_{obs}$ using $\bar{U}^i_{obs_k}$ and $\hat{x}^i_{obs_k} \forall i = [1, N_{obs}]$

7 Calculate $H_\mathbb{N}$ and $(H_\mathbb{N})_i$

8 Calculate E, f_{total}, M, γ

/* Optimisation */

9 Solve $\hat{\eta}_k = quadprog(E, f_{total}, M, \gamma)$

10 Decompress $\bar{U}_k = \mathbb{N}\hat{\eta}_k$

11 Select and apply first input $u_k = \bar{U}_k(1)$

/* Communication */

12 Communicate $\hat{x}_k$ and $\hat{\eta}_k$ to other UAVs

13 **end**

Result: $u_k, \bar{U}_k, \hat{\eta}_k$

III. Simulations

A. Multi-Rotor UAVs Positioning Model

Although one may apply the proposed approach directly in a fully nonlinear system, positioning models for multi-rotor UAVs can often be simplified to a mass-damper given by:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} -a_x\dot{x} + b_x F_x \\ -a_y\dot{y} + b_y F_y \\ -a_z\dot{z} + b_z F_z \end{bmatrix} \tag{19}$$

This is because of their omni-directional movement capabilities, thus allowing the NMPC to be used as a "high-level" planner which can be coupled with inner controllers, eg. by using control allocation techniques [16] or nonlinear control laws such as back-stepping and Nonlinear Dynamic Inversion (NDI) [11]. Moreover, the use of a linear model allows further reductions in the computational burden. For example, matrices E, M would be constant, thus allowing the use of special cases on QP solvers which can take advantage of this condition.

B. Simulation Setup and Results

To evaluate the performance of the proposed approach, a $T = 50\ (s)$ simulation with 50 UAVs sharing common airspace was performed. In this case each UAV had a random reference position assigned approximately every $3 \pm 1\ (s)$ in a cylindrical space with a 15 meter radius, and a 10 meter altitude. The new reference was assigned such that the UAV would pass close to the center of the cylindrical airspace to maximise the amount of possible collisions between the UAVs.

Assuming the state $\underline{\mathrm{x}}_k = [x, v_x, y, v_y, z, v_z]_k^T$ and the control vector $\underline{\mathrm{u}}_k = [F_x, F_y, F_z]_k^T$, system (19) was discretised using Zero-Order Hold with a sampling time of $T_s = 0.02\ (s)$ and coefficients $a_x = a_y = a_z = 0$ and $b_x = b_y = b_z = 1$, ie. a unit-mass in space. Moreover, the parameters used for the approach were selected as $Q = [1, 0.1, 1, 0.1, 1, 0.1]$, $R = [1, 1, 1]$, $N_p = 100$, $a_L = 0.7$, $d_{min} = 1$, $k_{pot} = 10$, $\epsilon = 0.5$. No state or input constraints were imposed. Moreover, the reference was injected at the end of the prediction horizon to give a smooth transition as discussed in [12].

Considering the UAVs are navigating a dynamic environment, it is difficult to demonstrate the performance in a static graph given 2 UAVs can occupy the same space at different times. The result can be better appreciated in the animation given in (https://youtu.be/LhTsyzlaFfw). Figure 2 shows the minimum distances maintained between all 50 UAVs, clearly showing the distance constraint is satisfied.

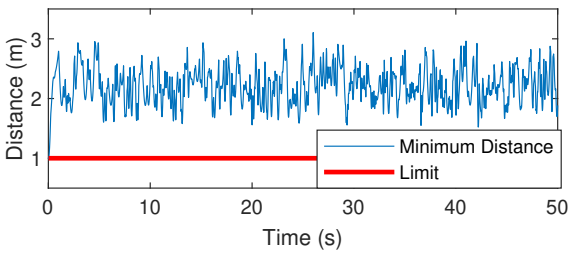

Fig. 2. Minimum distance maintained between all 50 UAVs.

IV. Conclusions and Future Work

This paper proposed a Distributed NMPC approach for Dynamic Obstacle Avoidance on Multi-rotor UAVs based on the so-called Laguerre Polynomials combined with Potential Field terms to reduce the computational burden, as well as to reduce the amount of information required to be transmitted, thus giving it the potential to be used for real-time applications. A complete derivation of the overall method is presented along with an algorithm for its implementation. The efficiency of the proposed approach is presented with a simulation of 50 UAVs.

Future work will focus on a physical implementation whilst including robust/uncertainty considerations to tackle multi-step time-varying delays on the communication channels.

Acknowledgment

This work was funded by UKRI Trustworthy Autonomous Systems Node in Security (Grant Number EP/V026763/1).

References

[1] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A Review of Motion Planning for Highway Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1826–1848, 2020.

[2] C. Huang, B. Li, and M. Kishida, "Model predictive approach to integrated path planning and tracking for autonomous vehicles," *IEEE Intelligent Transportation Systems Conference*, pp. 1448–1453, 2019.

[3] H. Zhou, H. L. Xiong, Y. Liu, N. D. Tan, and L. Chen, "Trajectory planning algorithm of UAV based on system positioning accuracy constraints," *Electronics (Switzerland)*, vol. 9, no. 2, 2020.

[4] M. Cefalo, E. Magrini, and G. Oriolo, "Sensor-Based Task-Constrained Motion Planning using Model Predictive Control," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 220–225, 2018.

[5] M. Iacono and A. Sgorbissa, "Path following and obstacle avoidance for an autonomous UAV using a depth camera," *Robotics and Autonomous Systems*, vol. 106, pp. 38–46, 2018.

[6] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," *European Control Conference*, pp. 4136–4141, 2013.

[7] T. P. Nascimento, A. G. Conceiçaõ, and A. P. Moreira, "Multi-Robot nonlinear model predictive formation control: The obstacle avoidance problem," *Robotica*, vol. 34, no. 3, pp. 549–567, 2016.

[8] L. Dai, Q. Cao, Y. Xia, and Y. Gao, "Distributed MPC for formation of multi-agent systems with collision avoidance and obstacle avoidance," *Journal of the Franklin Institute*, vol. 354, no. 4, pp. 2068–2085, 2017.

[9] T. Brüdigam, K. Ahmic, M. Leibold, and D. Wollherr, "Legible Model Predictive Control for Autonomous Driving on Highways," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 215–221, 2018.

[10] G. Garimella, M. Sheckells, and M. Kobilarov, "Robust obstacle avoidance for aerial platforms using adaptive model predictive control," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5876–5882, 2017.

[11] O. J. G. Villarreal, *Efficient Real-Time Solutions for Nonlinear Model Predictive Control with Applications*. PhD thesis, University of Sheffield, August 2021.

[12] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.

[13] L. Wang, *Model Predictive Control System Design and Implementation Using Matlab*. Springer, 2009.

[14] J. A. Rossiter, L. Wang, and G. Valencia-Palomo, "Efficient algorithms for trading of feasibility and performance in predictive control," *International Journal of Control*, vol. 83, no. 4, pp. 789–797, 2010.

[15] B. Khan, *Efficient Parameterise solutions of predictive control*. Ph.d. thesis, The University of Sheffield, 2013.

[16] T. A. Johansen and T. I. Fossen, "Control allocation—a survey," *Automatica*, vol. 49, no. 5, pp. 1087–1103, 2013.