Aerospace Science and Technology 105 (2020) 105965

Contents lists available at ScienceDirect

Aerospace Science and Technology

www.elsevier.com/locate/aescte

Towards a PDE-based large-scale decentralized solution for path planning of UAVs in shared airspace

Reza Radmanesh [a], Manish Kumar [b], Donald French [c], David Casbeer [d]

[a] NASA Jet Propulsion Laboratory, 4800 Oak Grove Dr, Pasadena, CA 91109, USA
[b] Department of Mechanical and Materials Engineering, University of Cincinnati, Cincinnati, OH 45221, USA
[c] Department of Mathematical Science, University of Cincinnati, Cincinnati, OH 45221, USA
[d] Control Science Center of Excellence, Air Force Research Laboratory, Wright-Patterson AFB, OH 45433, USA

ARTICLE INFO

Article history:
Received 13 November 2019
Received in revised form 16 April 2020
Accepted 2 May 2020
Available online 12 June 2020
Communicated by Hever Moncayo

Keywords:
Decentralized control
Trajectory planning
Partial Differential Equation (PDE) method
Unmanned Air Vehicles (UAVs)
Large scale optimization

ABSTRACT

Recently, there has been a tremendous increase of interest in utilizing Unmanned Aerial Vehicles (UAVs) for a number of civilian applications. With this increased interest, it is imperative that these UAVs are able to operate in shared airspace for enhanced efficiency. Multi-UAV systems are inherently safety-critical systems, which means that safety guarantees must be made to ensure no undesirable configurations, such as collisions, occur. This paper proposes a decentralized method based on a Partial Differential Equation (PDE) to generate collision-free 3D trajectories for multiple UAVs operating in a shared airspace. This method exploits the dynamical properties of multi-phase fluids flowing through a porous medium by modeling the porosity values as a function of the risk of collision. To highlight the feasibility for on-board implementation, we propose propose a machine learning technique for obtaining computationally efficient solutions of the PDE describing flow movements in porous medium. This method has been compared via a simulation study to two other path planning strategies, centralized and sequential planning, and the advantages of this method are presented. Furthermore, results from an experiment using three UAVs have been presented to demonstrate the applicability of the proposed method to real-world implementation.

1. Introduction

Unmanned Air Vehicles (UAVs) have traditionally been used in military operations for a number of years. Recently, UAVs have generated a lot of interest due to their potential application in civilian domains such as emergency management, law enforcement, precision agriculture, package delivery, and imaging/surveillance [1,2]. These applications could require the UAVs to fly over in urban areas. Therefore, the full potential of UAVs can only be realized if they are allowed to utilize the same airspace in a safe and reliable manner. Hence, before the use of UAVs becomes a reality in civilian domains, it is critical that the challenges emanating from integration of UAVs in the National Airspace System (NAS) are extremely critical are solved. One of the most important challenges is the ability for a UAV to not only plan its own path for fulfilling a mission but also to re-plan or adjust its trajectory in order to avoid collision with other aircraft.

Another factor beyond UAVs integrating into airspace is the dramatic increase in the number aircraft over the last 50 years. This increase in manned aircraft, along with incorporation of unmanned fleet in future, will pose severe challenges to the Air Traffic Control (ATC). Hence, the Radio Technical Commission for Aviation (RTCA) and also Federal Aviation Administration (FAA) have been charged with a responsibility to implement a seamless change from ATC to Air Traffic Management (ATM) by 2020 [3,4] that incorporates mechanisms to plan/re-plan the paths of UAVs to avoid collisions with other aircraft.

Various attempts have been done to address the path planning issue of vehicles in presence of other vehicles. One group of studies refers to a central computational center, which is computationally intractable as the number of agents grows [5,6]. The other approach is distributed/decentralized trajectory planning [7–10]. Decentralized planners break the optimization problem into smaller sub-problems, with the rationale that solving several small problems is faster and more scalable than solving one large centralized problem [11]. Several previous studies focus the decentralized path planning and task assignment for multiple UAVs, but most of them focus on planar motion with simplified vehicle dynamics. One of the primary issues with decentralized methods is enforcement of

E-mail addresses: Reza.Radmanesh@jpl.nasa.gov (R. Radmanesh), Manish.Kumar@uc.edu (M. Kumar), Donald.French@uc.edu (D. French), David.Casbeer@us.af.mil (D. Casbeer).

https://doi.org/10.1016/j.ast.2020.105965
1270-9638/

global constraints [12]. Kuwata et al. presented a method based on receding horizon mixed integer linear programming without considering the temporal cooperation [13]. Generating a guaranteed flyable and smooth path is the focus of the strategy presented by the McLain et al. that uses a coordination function to achieve cooperative timing among teams of agents by coordinating the path characteristics [14]. Decoupling the state-space with more complex control laws is presented in [15].

Considering the distributed/decentralized approach, in one strategy, Van Den Berg et al. [16,17] have used reciprocal velocity obstacles for real-time multi-agent navigation. Implementation of virtual potential fields while maintaining a specific formation is applied in [18] and [19]. These studies have provided interesting results but have not considered trajectory planning and collision avoidance simultaneously. In other studies, by defining the safety parameters using reachability analysis, optimal system trajectories have been provided [20,21]. The discretization of the state space along with Hamilton-Jacobian Partial Differential Equations (PDEs) presents an issue for scalability of the solver as the complexity of these methods increases exponentially as the number of agents grows [22]. One major issue with the aforementioned methods is that uncertainties from factors such as model errors and wind disturbances were not considered.

Partial Differential Equations (PDEs) are typically used to model system properties related to fluid mechanics and thermodynamics. They have also been proven useful in areas of modeling of mechanical and biological systems [23]. This paper utilizes the fluid mechanical property of flow in porous media. Partial resistance to fluid flow is among the principal characteristics of the porous media to be considered. For example, fluid flows from a source to a sink point because of the fact that the sink corresponds to lower potential energy. During this process, the fluid finds the path with the highest permeability zones or the path of least resistance. The same behavior can be found in heat transport where the material's thermal conductivity dictates the path that heat travels [24] through.

Several studies regarding the application of PDE equations and PDE modeling for control purposes have appeared in recent literature. For example, Barooha et al. has studied vehicular platooning by applying decentralized bidirectional control [25]. In this study, each vehicle is modeled as a double integrator and then a PDE approximation is derived for the discrete platoon dynamics. Then, the derived PDE model is used to explain the stability of the closed-loop system while adding more vehicles in platoon. Similarly, Frihauf and Krstic [26] introduced a stable deployment method of agents using a PDE-based approach. In their method, the agents' dynamics are modeled by *reaction-advection-diffusion* class of PDEs. By introducing a feedback law, the agents are allowed to deploy into a family of geometric curves and the stability of the method is then guaranteed by enabling a leader approach. In [27], Sarlette and Sepulchre proposed a discretized system of agents with local interactions modeled using PDEs. In their study, they found that for coordination of subsystems through local interactions, by assuming the symmetry in translation of subsystems value, distributed control systems can benefit from the discrete approximation of one-dimensional PDEs. Pimenta et al., in [28], designed feedback control laws for swarm of robots based on models from fluid dynamics. By applying the in-compressible fluid model, a solution for pattern generation task is generated. In their study, using smoothed-particle hydrodynamics (SPH) technique and Kernell equations, they aimed to devise a decentralized controller. Similarly, flocking of autonomous agents using a PDE based on Cucker-Smale model is proposed in [29]. Furthermore, based on the analysis of particle flow, which is a result of PDE Cucker-Smale technique, the stability of the technique is proven.

In other study [30], stochastic mapping and coverage strategy using robotic ensembles by PDE-based optimization is studied. In their study, by using a advection-diffusion-reaction PDE model, they task the agents to gather the data and identify the region of interest. Mapping task is then solved as a convex optimization problem using a spatially dependent coefficient in the PDE model. Furthermore for the coverage problem, their study suggests an optimal control problem formulation in which the generated PDE model is expressed as a bilinear control system with the agents' states as the control inputs. In another study, an advection-diffusion equation, which is originally based on Kolmogorov forward PDE equations, is studied for spatio-temporal evaluation of the swarm of agents [31]. Another application of PDE equations in multi-agent systems is introduced by Galstyan et al. [32]. In their paper, they have addressed swarm of chemotactic microscopic robotics based on the diffusion model, introduced by PDE equations and showed the result of their study in one-dimension solution. In [33], Prorok et al., represented the agents' spatial dynamics over time by PDE based Fokker-Planck diffusion model in a swarm of robots. Another study of swarm, using the same PDE based Fokker-Planck technique, inspired from the honeybee behavior in nature, is addressed in [34].

In this paper, based on our previous work [24], a novel cooperative approach for multi-UAV path planning problem is proposed which has been inspired by use of PDEs in fluid mechanics, thermodynamics, circuits, and heat transfer. PDEs are used to model the fluid flow throughout the domain through the derivation of the surface's porosity. The non-porous part in the fluid domain is considered to be analogous to dangerous regions in the UAV environment which UAVs need to avoid. In the previous paper [24], we provided a centralized approach to solving the PDE that uses the information about trajectories of all the UAVs. This paper advances the previous work by proposing a decentralized scheme for obtaining the PDE solution that allows multi-agent control for a predefined communication topology while guaranteeing the flight-safety and maintaining fleet cohesiveness. The proposed approach is decentralized, in that it does not need the complete trajectory information of all UAVs. Instead, this algorithm just needs only the initial and goal locations, and arrival times for all UAVs. This is motivated from realistic scenarios where this information is available initially but the UAVs are not able to share their trajectories during the mission. Furthermore, a sequential planning approach is proposed in this paper. Finally, the performance of the proposed method is demonstrated using numerical simulations and compared with other methods.

The rest of the paper is organized as follows. In Section 2, the problem is explained and formulated. Section 2.2, designing the team analogy and various scenarios are discussed, and in Section 5, the equations and implementation details of the fluid motions are discussed in details. Three control strategies, as discussed before, are provided in detail in Section 6, and a comparison of these three methods is provided in Section 7. Analysis of these methods using a realistic UAV scenarios are carried out in Sections 8 and 9, respectively. At the end of this paper, the experiments were provided in Section 10 to integrate the proposed method.

2. Problem formulation

In this section we present the general form of problem statement. Consider N UAVs with dynamics described by:

$$\forall i \in \{1, ..., N\}: \quad \dot{x}_i = \mathfrak{f}_i(x_i, u_i, d_i); \qquad u_i \in \mathcal{U}_i, d_i \in \mathcal{D}_i \tag{1}$$

In equation (1), the states of each vehicle $x_i \in \mathbb{R}^{\mathbb{N}}$, control of UAV i $u_i \in \mathcal{U}_i$ and the effects of the environment (i.e. disturbances of the UAV network) $d_i \in \mathcal{D}_i$ are assumed to be bounded.

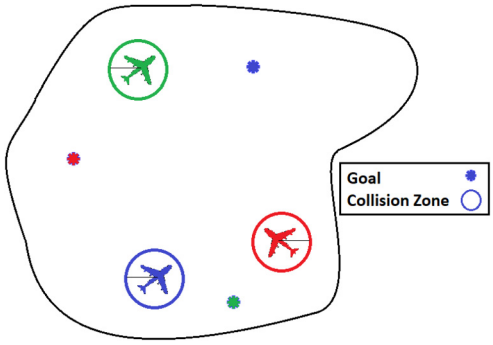

Fig. 1. Problem setup. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

We further assume that the flow field $\mathfrak{f}_i : \mathbb{R}^N \times \mathcal{U}_i \times \mathcal{D}_i \longrightarrow \mathbb{R}^N$ is uniformly continuous, bounded, and Lipschitz continuous in x_i for fixed u_i and d_i.

In order to implement the collision avoidance feature, we define a collision zone $\mathbb{D}_{ij}(t)$ associated with a vehicle i, $\forall j \in \{1, ..., N\}$, for all time of its flight. Collision zone is defined by equation (2).

$$\forall i \& j \in \{1, \cdots, N\} \quad \& \quad i \neq j$$
$$\mathbb{D}_{ij}(t) = \{x \in \mathbb{R}^n : ||x_i - x_j(t)|| \leq \mathbb{d}_c\} \tag{2}$$

In equation (2), $||.||$ represents the Euclidean distance. The collision zone ensures that, at any time, a vehicle needs to keep its minimum, predefined distance $\mathbb{d}_c$ from other vehicles. A schematic of the problem being addressed in this paper is shown in Fig. 1. The figure shows multiple UAVs each with its own collision zone and goal location. Each UAV has to travel from its current location to the goal location without infringing upon the collision zone of other UAVs.

This form of presentation of the problem would further assist the comparisons with the other provided methods such as Backward Reachability Sets (BRS) method in Numerical Scenarios Section.

2.1. Solution outline

In this section, we aim to present the overall flow of the proposed centralized and decentralized trajectory planning approach. We introduce a new term called 'Prediction Set (PS)' which is defined as all possible spatial positions that a UAV, with $a-priori$ known kinematic and dynamic constraints, can occupy in a certain amount of time. A detailed explanation and examples on how to compute and implement PSs is provided in Section 5. For calculating the PSs, we have utilized the principle of flow of multi-phase fluids in porous medium. In Section 5, after providing a brief discussion of the fluid dynamical properties used in this paper, we have presented a machine learning technique to improve the computational efficiency in obtaining PSs. The goal of Section 5 is to provide all the Stream Lines (**SL**) directing the vehicle from the initial point to the final position within a certain amount of time and while satisfying the vehicle dynamic/kinematic constraints. Based on these SLs, the decentralized controller, proposed in Section 6.2, finds the safest (collision-free) and fastest path for the vehicles.

2.2. Problem analysis

In this section, we assume that every member of the agent has knowledge of start and goal locations, and *arrival times of every member of the team*. The members do not share the exact trajectory information with others. Hence, generating a strategy for an agent for every possible actions taken by the other members of the team is the primary focus of this section. Considering the schematic fluid movement in porous medium [24], as presented in Fig. 2, the vehicle given its dynamical constraint, is only capable of occupying certain states during the next time steps to reach their *a-priori defined* goal locations within the *a-priori defined* time constraint. Backward Reachability Sets (BRS) have been defined in [22] and [20] as the set of states from which there exists a control such that, for all non-anticipative disturbances, the system is driven to the goal position within the specific time horizon while satisfying all constraints such as collision with obstacles. In this paper, due to the fact that each of the UAVs are oblivious about the decisions of the other agents and the UAV's trajectory planning are happening in real-time, a novel concept called Prediction Set (PS) is introduced.

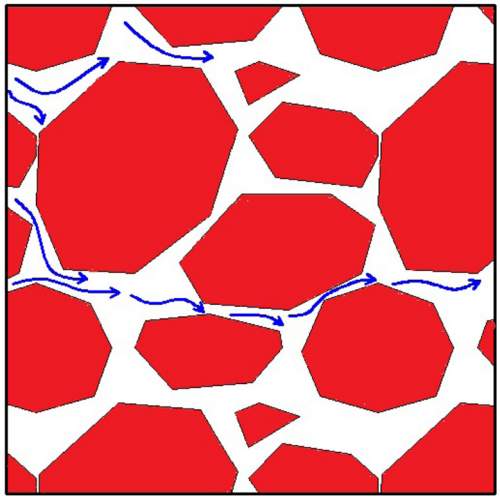

Fig. 2. Typical topographical map considered in of the problem.

3. Overview of PDE based path planning approach

Here, we provide the overview of the PDE based technique for UAV trajectory planning. This method is highly dependent on finite element technique for the fluid particles. Based on the fluid analogy, the fluid tends to flow from a higher potential to a lower potential in a field. For instance, consider a water tap. By opening the valve, the water runs from a higher potential (i.e., tap) to a lower potential (i.e., sink), in shortest time. Fig. 3 represents a schematic of streamlines.

To make the problem more clear, we will introduce the following key for the rest of the paper.

- The UAV is considered as a particle in a flow. The equations of fluid flow, directs the UAV from the initial point to the final point in an optimal fashion [5,35]. The equations governing the fluid motions and the solution method are provided in this section.
- Each stream line represents a path for the UAV. The post-processing, as explained in [5], is carried out to ensure the path can be followed by the UAV under its kino-dynamic constraints.
- Potential functions f, originated from the fluid analogy, is designed to represent the higher potentials at the starting point and lower potential at goal locations.

3.1. Fluid analogy

Continuity Equation of fluid mechanics is used as the basis function f for generating all of the possible path solutions in the

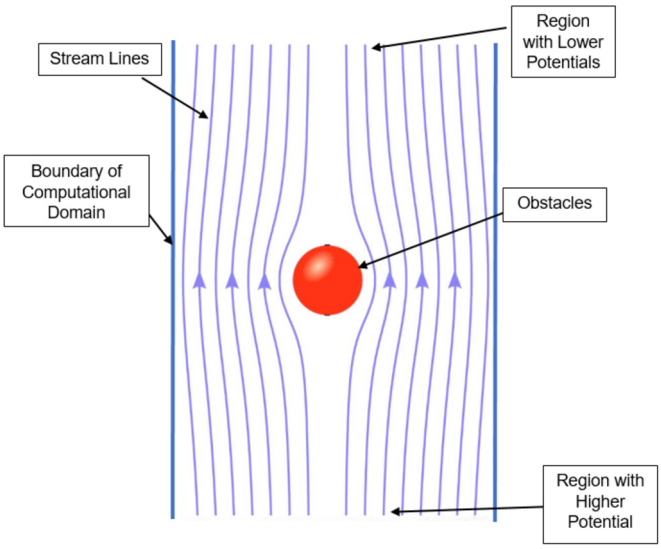

Fig. 3. Scheme of the flow, stream lines and boundary condition of computational domain.

porous domain $\mathbb{C}$. This is represented in equation (3), where ϕ is the porosity of the evaluated three-dimensional point, ρ is the density of the fluid, and u is the velocity of the fluid.

$$(\phi\rho)_t + \nabla.(\rho u) = f \quad \text{in} \quad \mathbb{C}. \tag{3}$$

Then by applying the *Darcy's Law* to equation (4), we describe the flow of the fluid through the porous domain. The steady state solution of equation (3) is applied to obtain:

$$-\sum_{e\in\{x,y,z\}} [\frac{\partial}{\partial e}\kappa\frac{\partial u}{\partial e}] = f; \tag{4}$$

In equation (4), κ represents the permeability of the medium $\mathbb{C}$. It should be noticed that due to finite element method for particles in fluid, the computational domain is always dealing with real numbers. The *Second Order Centered Difference* method is applied for each spatial point that interpolates the fluid particles properties using only the values of the cell before and after the currently evaluated cell to approximate each cell's porosity. To do that, equation (4) is discretized in x, y and z directions using finite difference method [36]. The substitution in (4) results in (5). The indices in equation (5) represent the grid-points. The variables h_x, h_y and h_z are the step sizes in terms of x, y and z directions respectively. They are specifically defined differently because the x, y, and z grid intervals are not equidistant. $O(h_e^2)$ is the error of numerical estimation and can be neglected [37] considering small grid size h_e.

$$\begin{aligned}
&-\frac{\kappa_{i+1/2,j,k}(u_{i+1,j,k}-u_{i,j,k})-\kappa_{i-1/2,j,k}(u_{i,j,k}-u_{i-1,j,k})}{h_x^2}\\
&-\frac{\kappa_{i,j+1/2,k}(u_{i,j+1,k}-u_{i,j,k})-\kappa_{i,j-1/2,k}(u_{i,j,k}-u_{i,j-1,k})}{h_y^2}\\
&-\frac{\kappa_{i,j,k+1/2}(u_{i,j,k+1}-u_{i,j,k})-\kappa_{i,j,k-1/2}(u_{i,j,k}-u_{i,j,k-1})}{h_z^2}\\
&= f_{i,j,k}
\end{aligned} \tag{5}$$

The variables (i, j, k) in equation (5) represent the cell labels of the tessellated area. Equation (5) is solved for the entire domain $\mathbb{C}$ while accounting for the boundary conditions.

The results of Equation (5) are multiple streamlines with varying fluid velocity u at every grid point. Since there is only one starting point and one final point in the domain, the number of solutions are ***reduced by applying the flux boundary conditional constraints*** which require that solutions start at time 't_0' at (x_0, y_0, z_0) and finish at time '$\mathbb{T}$' at (x_f, y_f, z_f). The process can be explained as below.

In the domain $\mathbb{C}$, multiple streamlines or flow paths from the origin to the goal position are generated. Incompressible flow dynamics represented by equation (5) dictate that the streamlines follow the velocity fields and finish at the goal position. The streamlines are evaluated with the cost equation, (i.e. safely reaching to the goal position with the shortest distance traveling) and are converted to arrays of (x, y, z) points using the *Runge-Kutta Method (RKM)* algorithm [24]. The RKM method yields all the routes that travel from the initial point to the goal position without colliding to any other agents (i.e., solid part of computational domain $\mathbb{C}$) and violating the fluid constraints. Further studies on this method are provided clearly in [24].

The streamlines, generated from the technique explained, contain spatio-temporal information about the fluid particle. We will use this characteristic of solution in next section to generate the Prediction Sets for UAVs. It is obvious that only the streamlines satisfying the UAV kinematic constraints and dynamic constraints are used to generate all the paths.

4. Prediction sets (PS)

The set of states for the future time steps for the team to reach their target can be defined as:

Definition 1. Prediction Sets (PS) for total time of flight $\mathbb{T}$ corresponding to the multiple agents with the shared information of target and arrival time can be found using the solution for PDE (6) defined in the porous medium.

$$\begin{aligned}
&PS_{i,\mathbb{g}} : \{\exists \mathbf{SL}_{\mathbb{g}} : (\phi\rho)_t + \nabla(\rho u) = f \quad \text{in} \quad \mathbb{C}\},\\
&\textbf{s.t.} \quad \forall t \in [t_1, \mathbb{T}] \quad \& \quad \mathbf{SL}_{\mathbb{g}} \notin [g(x) \cup \jmath(x)];
\end{aligned} \tag{6}$$

where $PS_{i,\mathbb{g}}$ represents the prediction sets for UAV i for goal position $\mathbb{g} \in G$ and $\mathbf{SL} = \bigcup(x, t)$ indicates the streamlines, directing the UAV from the initial point to the UAV's respective goal location. In simpler words, each streamline, after the proposed post-processing by [24], is a potential path, containing information about position which is labeled with time. In equation (6), $\mathbb{T}$ is the total time considered for the opposing team, t_1 is the start time of prediction, $g(x)$ and $\jmath(x)$ represent the topographical map and the solid part of porous medium respectively. It may be noted that although $g(x)$ as the topographical map of environment is fixed and predefined, and the function $\jmath(x)$ is constantly changing. This function is changing to adapt the threats and collisions in the environment [24].

Numerical solution of equation (6) is presented in Section 5 and it leads to extracting all the streamlines from the initial position to the goal location [24]. By adding the time constraint from $[t_1, \mathbb{T}]$, all the acceptable solutions can found. □

Fig. 4 represents the prediction sets for the blue team for reaching their targets located at G_1 and G_2. This prediction set is then utilized for trajectory planning of the red team.

It is assumed that the vehicles share the target and time of arrival with the opponents. Otherwise, if only the arrival time is shared, a general form of Definition 1 can be implemented as defined in equation (7).

$$\forall \mathbb{g} \in G \quad : \quad PS_{i,total} = \bigcup_{g} PS_{i,\mathbb{g}} \tag{7}$$

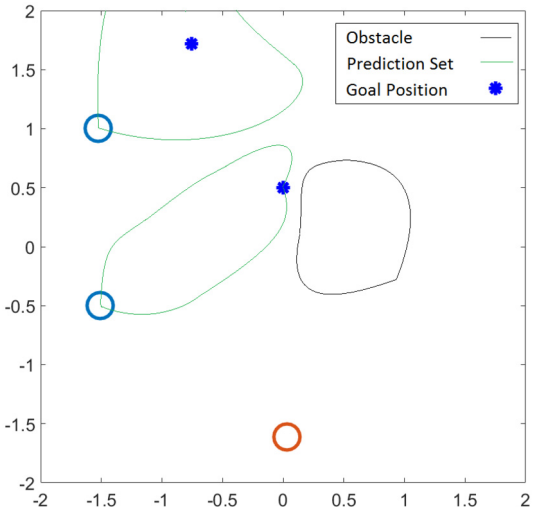

Fig. 4. Prediction set for team blue for reaching to their goal position within the "a-priori" defined time. In this scenario a region is affected by the position of obstacle.

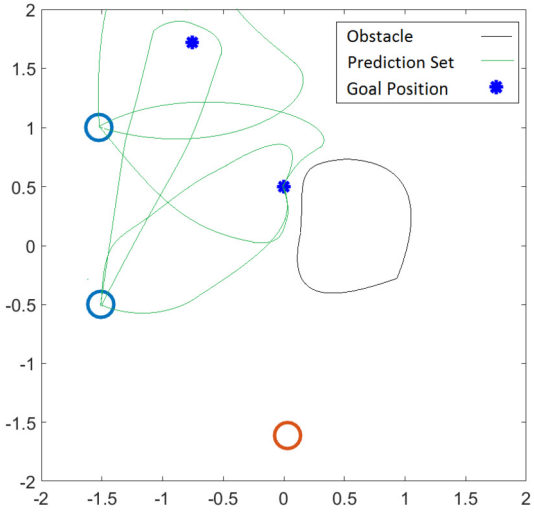

Fig. 5. The result of prediction set upon having only the total time of flight. The goal position for the opponent team (blue) has not been provided but the total time of flight is mentioned.

For example, consider the prediction set provided in Fig. 4. If the target position is not provided, then the prediction set would be represented by Fig. 5.

5. Governing equations for centralized approach

Here, we will assume the vehicles in the environment as the particles in multi-phase flow. The equations of multi-phase flow are often computationally extensive to solve. In this section, we aim to simplify the multi-phase flow equations and explain the governing equations. The results of these equations, is implemented in a centralized scheme, using multi-phase flow, and in a decentralized scheme, using a single-phase flow.

5.1. Governing equations for developing the centralized approach

In this section, governing equations for a multi-phase flow are briefly introduced. Further details on analytical representations of fluid flow in a porous medium can be found in [38] and [39]. Simulating the multi-phase flow with a single set of conservation equations for the whole flow field is considered in this paper. Moreover, surface tension is not considered and only sub-surface flow is considered in this paper. Since the material properties and the flow field are, in general, discontinuous across the interface, the differential form of the governing equations must be interpreted as a weak form [38]. To make it further simple, we will start with high fidelity equation for multi-phase flow (i.e. equation (8) and (10)) and by using simple assumptions, we will calculate less complex form of the equations for the purpose of machine learning technique.

As a preliminary introduction, it is necessary to discuss a PDE binary function $\mathbb{H}$, as shown in equation (8), which is equal to 1 if the particular fluid is there and 0 elsewhere. This $\mathbb{H}$ function is the tensor of fluid flow representing the presence of the flows in different phases (such as oil and water) in certain coordinates. This binary function is generated through a function δ as the delta Dirac function, which needs to be approximated during the numerical simulation. The approximation for this function is completely explained in equation (8) [38].

$$\forall(x, y, z); \quad \mathbb{H} = \int_{V(t)} \delta(x - x')\delta(y - y')\delta(z - z')dV. \tag{8}$$

In equation (8), the integral is over a volume V bounded by a contour S and the prime coordinates (such as x', y', z') represent the points in the front surface of the fluid (before interacting with the porous medium). For simplicity, in equation (8), it is assumed that the values are symmetric. Then we define:

$$\nabla\mathbb{H} = -\int_{S} \delta(x - x')\delta(y - y')\delta(z - z')n'dS \tag{9}$$

Considering the variation of density of fluid in a porous medium, we add another parameter to our functions as **H** which is dependent on the density of the flow in porous medium.

$$\rho(x, y, z, t) = \rho_k \mathbf{H}(x, y, z, t) + \rho_0(1 - \mathbf{H}(x, y, z, t)), \tag{10}$$

where:

$$\mathbf{H} = \omega.\mathbb{H}(x, y, z, t) \tag{11}$$

ω is a parameter determining the dependency of the function **H** on the pressure. In equation (10), ρ_k is the density where **H** is 1 and ρ_0 is the density where **H** is 0.

In this case, the Navier-Stokes equations are defined as:

$$\begin{aligned}\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla.\rho\mathbf{u}\mathbf{u} = &-\nabla P \\ &+ \rho f + \nabla \cdot \mu(\nabla\mathbf{u} + \nabla^T u) \\ &+ \int \sigma\kappa' n' \delta^{\beta}(x - x')ds';\end{aligned} \tag{12}$$

This equation is valid for the whole domain. **u**, in equation (12), represents the velocity domain, P is the pressure, and f is the body force. δ^{β} is a three dimensional δ function and $\beta = 3$ represents the dimensions. κ is the curvature for two-dimensional flow and n is a unit vector normal to the front. It may be noted that the "'" sign can be neglected by assuming the symmetry of the medium [40]. Mass conservation of the flow is introduced by equation (13).

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho\mathbf{u} = 0. \tag{13}$$

The assumption of in-compressible flow is not yet determined to be accurate since the drop in pressure is a common phenomenon in nature when the fluid passes through porous medium.

The aim of this paper is to solve *numerically* a fully-implicit discretized set of nonlinear equations represented as [39,41,42]:

$$g(x^{n+1}, x^n, u^{n+1}) = 0, \tag{14}$$

where g is the residual vector which tends to 0. In equation (14), x represents the states, and u is the control variable and n is the time step. In order to reduce the computational effort associated with the high-fidelity model solution, a reduced-order modeling approach is considered in this paper. This results in approximation errors but provides the advantage in terms of the computational gain. Hence, the Newton's method is considered as a solver to equation (14). The application of Newton's method requires the existence of first and second derivatives.

The Newton method requires a Jacobian matrix at the kth iteration as:

$$\mathbb{J} \in \mathbb{R}^{2^{Ng} \times 2^{Ng}}; \quad \mathbb{J}^k = \frac{\partial g^{n+1,k}}{\partial x^{n+1,k}}. \tag{15}$$

In this paper, a piece-wise representation of equation (14) is considered which is explained in detail in reference [43]. The residual function in equation (14) can be represented as equation (16).

$$\begin{aligned} g^{n+1} \approx\ & g^{i+1} + \frac{\partial g^{i+1}}{\partial x'^{i+1}}(x^{n+1} - x'^{i+1}) \\ & + \frac{\partial g^{i+1}}{\partial x'^{i}}(x^{n} - x'^{i}) \\ & + \frac{\partial g^{i+1}}{\partial u'^{i+1}}(u^{n+1} - u'^{i+1}) + \Omega(\mathbb{x}^2) = 0. \end{aligned} \tag{16}$$

Instead of state at time step n in equation (16), the residual vector is approximated using the closest state at time step i, found from the learning process [43] and $\Omega(\mathbb{x}^2)$ is the second and higher order residual. It may be noted here that among the state vectors, the solution for pressure is not obtained as mentioned in Theorem 1 below.

Remark 1. Among different solutions available in literature [44], a popular attempt is to solve for the pressure and eliminate one of the other parameters such as velocity over the solution space. This enhances the results in terms of reduction in calculation time. This necessitates the discussion of assumption of "slightly compressible flow" and a "non-compressible flow".

It may be noticed that the nonlinear approach still adds unnecessary complexity to the solution. In Theorem 1, we aim to provide more assumptions, such as neglecting change in pressure, to reduce the complexity, and their effects on the optimality of the solution. In particular, we will provide sufficient proof that approximation of equation (16) will lead to an acceptable solution for equation (13).

Theorem 1. *The proposed numerical solution approach given by equation (16) for solving equation (13) for a specific state x and u converges to the exact solution without imposing any assumption on the gradient of pressure over the entire domain (i.e., specific point convergence method) [45].*

Proof. The linear approximation given by equation (16) involves second order tensors. In this step, since i is considered sufficiently close to n, the equation (16) can be written as equation (17):

$$\begin{aligned} g^{n+1} =\ & \mathbb{J}^{i+1}(x^{n+1} - x'^{i+1}) \\ & + \frac{\partial g^{i+1}}{\partial x'^{i+1}}(x^{n} - x'^{i}) \\ & + \frac{\partial g^{i+1}}{\partial u^{i+1}}(u^{n+1} - u'^{i+1}) + \Omega(\mathbb{x}^2) = 0. \end{aligned} \tag{17}$$

Upon having a precise learning set, the function Ω as the residual of approximation, is in order of x^2 and by assuming the sufficient learning data, the order of x^2 would be negligible [46]. In equation (17), the term by term analysis indicates that the residual vector is not dependent on pressure and hence, assumption of any variation of pressure (and hence on compressibility of flow) is not going to affect the solution. Hence, this completes the proof of this Theorem. □

Remark 2. Using the results of Theorem 1, we will understand that upon having a reliable, and very vast number of pre-solved solutions for the PDE equation, the solution using the approximation of equation (16) converges to the optimal solution for PDE equations of path planning.

Remark 3. It is important to note that the result of equation (13) is '**u**' which can be decoupled as a function of time, i.e.:

$$\mathbf{u} = \sqcap_x(t)\vec{i} + \sqcap_y(t)\vec{j} + \sqcap_z(t)\vec{k} \tag{18}$$

In equation (18), $\vec{i}$, $\vec{j}$ and $\vec{k}$ represents the three unit vectors, respectively and $\sqcap_x$, $\sqcap_y$ and $\sqcap_z$ are functions, defining the **u** in each direction. Hence, we can identify the velocity of the particle in a sample plane xy as a function of x, y and t. This fact represents that the stream lines generated from equation (18) depend on t values and are associated with the position [38].

5.2. Implementation details and workflow

The high-fidelity simulations of multi-phase flow in porous medium would require substantial computer memory. Hence, for the initial training runs, the Jacobian information from [39] is used to reduce computational time requirement. The computational requirements for constructing the basis and generating the required matrices are comparable to the time required to perform the training simulations.

Solution for the proposed equation (16) is implemented in C++ and then imported into Matlab using an interface for further processing. The training represents new solutions, designated as x^{n+1}, in the vicinity of already saved solution (obtained from training set as mentioned above) for the scenario, defined by x^{i+1}. Over here, it may be noted from equation (17), solutions x^i and x^n obtained in the previous step are also required in computation of x^{n+1}.

5.3. Machine learning technique

Here, to improve the speed of computation of multi-phase flow in medium, we will implement a high-dimensional regression technique. The aim is to estimate the error introduced by surrogate models of parameterized dynamical system. Using the result of Theorem 1, we reformulate the equation (17), and generate the following equation:

$$\mathbb{J}^{\mathbf{i+1}} = \left.\frac{\partial \mathbf{g}}{\partial \bar{\mathbf{x}}}\right|_{(\mathbf{x'^{i+1}}, \mathbf{u'^{i+1}})}$$

$$\mathbb{B}^{\mathbf{i+1}} = \left.\frac{\partial \mathbf{g}}{\partial \bar{\mathbf{x}}}\right|_{\mathbf{x'^{i}}}$$

$$\mathbb{C}^{\mathbf{i+1}} = \left.\frac{\partial \mathbf{g}}{\partial \bar{\mathbf{u}}}\right|_{(\mathbf{x'^{i+1}}, \mathbf{u'^{i+1}})}$$

Here, we use x' to denote the states, received from the training simulation state, u' denotes the training controls, and i denotes the time step associated with the training state. The criterion for determining the best set from the training configuration is to find the values, indicated by $(x'^{i+1}, x'^{i}, u'^{i})$ to be zero as much as possible.

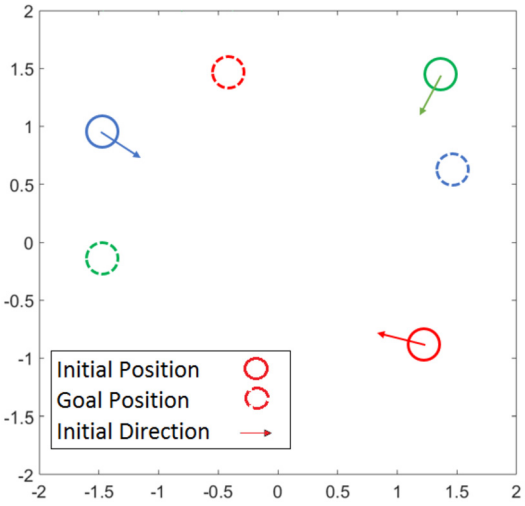

Fig. 6. The scheme of the problem setup. There are three agents with $a-priori$ known goal positions. Arrival time is also shared with the other agents.

Using the Proper Orthogonal Decomposition (POD) technique, we assume that a linear estimation exists in a low-dimensional affine subspace calculation as $x \approx \Phi \mathbb{z} + \bar{x}$ in which Φ is a POD basis and $\mathbb{z}$, is the reduced states. Furthermore, $\bar{x}$ indicates a reference state. Using the basis of POD technique results in lesser complex equations.

For solving the rest of equation, we implement the random forests (i.e., LASSO [47]) technique to map a large set of inexpensively computed error equations, and provide a better result. The method first performs feature-space partitioning via normal clustering, and subsequently constructs a 'local' regression model to predict the time-instantaneous error within each identified region of feature space. The verification of this method is provided in [43] with high-fidelity model. In this paper, we only implement lower complexity fluid dynamic model.

6. Control strategy

It is assumed that the vehicles presented here, have a simple kinematic model as:

$$\begin{aligned} x_i &= (\mathbf{x_i}, \mathbf{y_i}) \\ \mathbf{x_i} &= v_i \cos\theta_i \\ \mathbf{y_i} &= v_i \sin\theta_i \\ \dot{\theta}_i &= \omega_i \\ \underline{v} \leq v_i \leq \overline{v}, &|\omega_i| \leq \overline{\omega} \end{aligned} \tag{19}$$

where $x_i = (\mathbf{x_i}, \mathbf{y_i})$, θ_i, v_i represent the position, heading, and velocity of vehicle i. $\underline{v}$ is the lower bound of the velocity and $\overline{v}$ is the upper bound of the velocity. For test scenarios presented in this paper, the values for $\underline{v} = 0.5$, $\overline{v} = 1$ and $\overline{\omega} = 1$ have been chosen. It may be noted that for simplicity, the section uses simple two dimensional kinematic equations. The UAV model can be extended to three dimensions, as has been done in next sections. We present three different control strategies: i) Centralized Approach; ii) Decentralized Approach; and iii) Sequential Approach. The three approaches are tested against a scenario where there are three agents, and each of the agents knows $a-priori$ the goal position of the other agents. The total time of flight for each agent is also shared among all of the agents. The schematic of the problem is presented in Fig. 6.

```
Given the constraints for the UAVs;
Set different phases of fluids as number of UAVs;
Solve multi-phase flow for the computational domain (i.e. equation (10)), for any
random initial positions and goal locations;
Provide a learning set for centralized solutions;
t ⟵ 0;
iter ⟵ 0;
while Next position are not goal locations for the UAVs do
    while iter < maxiter do
        for The current states and constraints of the UAVs do
            Solve equation (16);
            Find the best u*(t) (best control decision for all the agents) through
            the learned scenarios;
            if Solution is acceptable then
                Save the solution;
                Set UAVs to follow based on the u*(t)
                Transmit the X* to all the other agents;
                t ⟵ t + 1
                else
                    Remove the unacceptable solution set;
                    Break;
                end
            end
        end
        iter ⟵ iter + 1;
    end
end
```

Algorithm 1: Centralized controller scheme using PDE method.

The initial states of the vehicles are indicated as:

$$x_1^0 = (-1.5, 1);$$
$$x_2^0 = (1.5, 1.5);$$
$$x_3^0 = (1.25, -1);$$

Each of the agents have a target located at:

$$c_1^{\mathbb{T}} = (1.5, 0.7);$$
$$c_2^{\mathbb{T}} = (-1.5, 0.8);$$
$$c_3^{\mathbb{T}} = (-0.5, 1.5);$$

For all the three control methods, centralized, decentralized and sequential, we present figures representing the configuration of agents for the above scenario at three different time instants: i) initial; ii) final and iii) intermediate when the agent configuration is most dense. The entire trajectories of all the agents are also plotted in all the figures. The simulations have been performed on a computer with the following specifications:

- Operating System: Windows 8 enterprise 64-bit
- Program language: MATLAB R2014a
- Processor: Intel(R) Core(TM)i7 -2500 CPU @ 3.30 GHz
- (2 CPUs)
- Memory: 8.00 GB RAM

In what follows, we explain the three control strategies in sections 6.1, 6.2 and 6.3.

6.1. Centralized approach

For this control strategy, we will directly solve the equations of multi-phase flow (i.e. (10)) using equation (16), to get the SLs. Then by using the post-processing technique [24], we would find the shortest and safest distance for the agents. Algorithm 1, provides more details of centralized controller.

This control strategy considers a non-homogeneous multi-phase flow and solves the numerical equations using the approach pre-

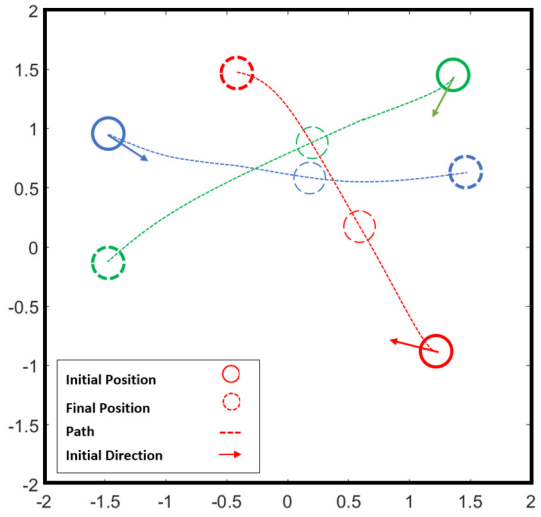

Fig. 7. The centralized approach for path planning. There are three agents with goal positions known $a-priori$ to all agents. Arrival time is also shared with the other agents.

Table 1
Comparison between MILP and proposed PDE method for different scenarios.

Number of Agents	PDE Normalized Cost	MILP TS $\times 10^{-1}$ (s)	PDE TS $\times 10^{-1}$ (s)
3	1.0001	6.1	4.1
5	1.0001	6.7	4.3
10	1.0005	7.4	6.7
100	1.002	42.5	40.2

sented in equation (16) to find the best possible path as an optimal solution. The path followed by each agent using this control scheme is shown in Fig. 7. In this case, all three vehicles appear to deviate slightly from a straight line trajectory in order to avoid collision with each other. Since the centralized controller is quite restrictive, the deviation is small that results from finding the most optimal solution. The time instant at which the configuration is most dense (at which agents are closest to each other) is $t = 10$ showing that the vehicles are close but still capable of avoiding each others' collision zone.

6.1.1. Comparison with optimal solution

It becomes imperative to compare the solution obtained from the proposed centralized approach with that of the globally optimal solution in terms of optimality and computational time. In this section, we compare the computational effort and optimality of the proposed method with respect to Mixed Integer Linear Programming (MILP) based method which provides globally optimal solution. Table 1 represents results from several scenarios with multiple agents to compare the optimality and computational efforts of the two methods. For each scenario, 25 trials with random initial positions have been averaged and reported in this table. It may be noted that the normalized cost (shown in the second column) is the ratio of cost obtained by the proposed method with that of cost obtained by the MILP method. Also, TS represents the time of solution required for the two methods. The table clearly shows that the proposed method provides computational advantage over the MILP method at very slight trade-off in optimality.

6.2. Decentralized approach

6.2.1. Decentralized policy

In this section, it is important to define the method for implementing the PSs in the algorithm. In this method, the time-varying

```
Given the constraints for the UAVs;
Solve single-phase flow for the computational domain (i.e. equation (10)), for any
random initial positions and goal locations;
Provide a learning set for the decentralized solutions;
t ⟵ 0;
iter ⟵ 0;
while Next position are not goal locations for the UAVs do
    for The current states and constraints of the UAVs do
        while iter < maxiter do
            for Any individual UAV i do
                Generate the set of obstacles G_i using equation (20);
                Add generated G_i(t) to J(t) as the solid form of the domain for
                avoiding;
                Solve equation (16) with the new environment;
                Find the best u*(t) (best control decision for all the agents) through
                the learned scenarios;
                if Solution is acceptable then
                    save the solution;
                    Set UAVs to follow based on the u*(t)
                    t ⟵ t + 1
                    else
                        Remove the unacceptable solution set;
                        Break;
                    end
                end
            end
            iter ⟵ iter + 1;
        end
    end
end
```

Algorithm 2: Decentralized controller scheme using PDE method.

obstacle induced for agent i caused by the PS of the agent j is denoted by $O_i^j(t)$. Once this $O_i^j(t)$ is computed, it is imported into algorithm as the total set of time varying obstacle set $\mathcal{G}_i^j(t)$ as shown in equation (20).

$$\forall i \& j \in N \quad \text{and} \quad i \neq j;$$
$$\mathcal{G}_i(t) = \bigcup_j O_i^j(t); \tag{20}$$

where:

$$O_i^j(t) = \left\{ \bigcup \forall (x(t), y(t), z(t)) \in PS_j(t) \right\} \tag{21}$$

In equation (20), N represents the total number of agents. The set of obstacles depends on current time step t. Then for each time step, it is necessary for agent i to avoid collision with the set of obstacles represented in equation (21). Algorithm 2, provides details of the proposed PDE decentralized control scheme for the agents.

6.2.2. Implementation

For solving using the decentralized approach, it is assumed that the vehicles are only aware of the arrival time $\mathbb{T}$, the initial position, and the goal position of each of the agents. In order to ensure collision avoidance, it is required to deviate much more from the straight line (as compared to the centralized solution) to reach the goal position. The solution obtained for this problem is represented by Fig. 8.

As it can be noticed in Fig. 8, the agents deviated from their path more as compared to the centralized solution. The dense configuration is represented again at $t = 10$, showing the vehicles maintain their separation. Fig. 9, represents the prediction set for each of the vehicles at time $t = 10$.

It is necessary to mention that in the Fig. 9, the separation is only guaranteed upon avoiding the prediction set of the other agents at all time.

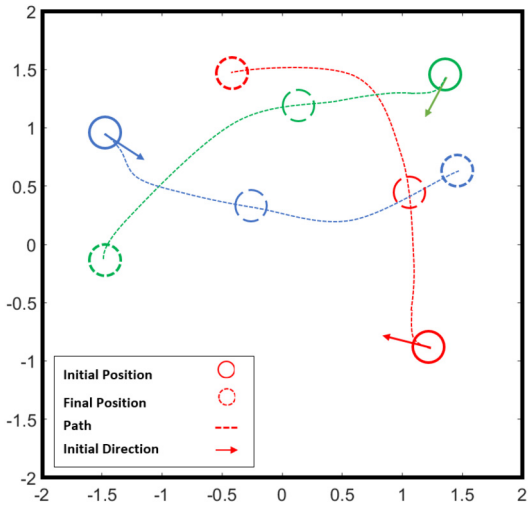

Fig. 8. Trajectories obtained using the decentralized approach. There are three agents with a-priori known goal positions. Arrival time is also shared with the other agents.

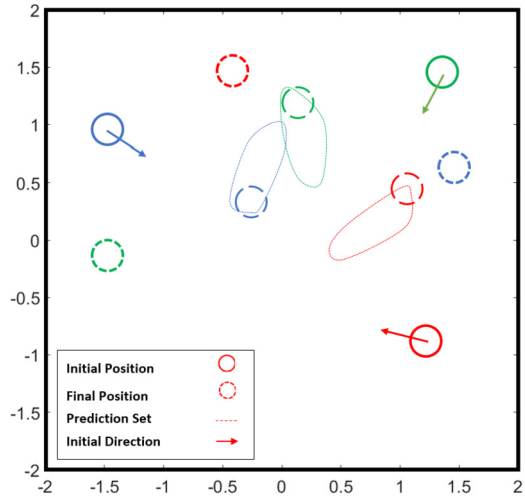

Fig. 9. Prediction Set (PS) for the scenario in Fig. 8 at time $t = 10$. As it can be seen each agent avoids the region which is predicted for the other agents.

6.3. Sequential planning

In order to make the N-vehicle path planning problem safe and tractable, a reasonable structure is imposed to the problem so that a strict priority ordering is assigned to each of the vehicles. Under this approach, during the trajectory planning for each of the vehicles for reaching to their respective targets, higher priority vehicle would neglect the presence of lower priority vehicle. This is totally different for the low priority vehicle as it needs to consider the presence of higher priority vehicles while planning its trajectory. Hence, for the descending priority of the agents, the prediction set of obstacles can be given by equation (22), for any higher priority vehicle i compared to the priority of j.

$$\forall j \in \{1, ..., i-1\};$$
$$O^i_j(t) = \bigcup_i \left[\exists (x, y, z) \in PS_i(t)\right]; \tag{22}$$

Considering equation (20), then the higher priority agent plans its trajectory using a single phase fluid flow in porous medium [24] and generates a new prediction sets of obstacles due to its path for lower priority agent. For defining the scenario here, it is

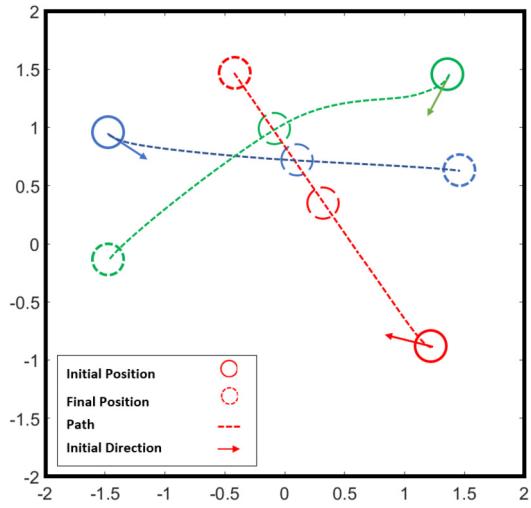

Fig. 10. Trajectory planning for three agents using the sequential approach. The red agent has the priority for the trajectory planning over the blue and the blue has priority over the green.

required to assign the priority first. The scenario is generated with the decreasing order of priority as red, blue and green agents respectively. Fig. 10, represents the results for this scenario.

As shown in Fig. 10, at first, the red agent as a higher priority agent, plans its trajectory in a straight line without deviation from its own optimal path. Then the blue agent generates its path to reach to final position, considering only the red agent's information for generating set of $O^i_j(t)$, using equation (22). Since there was no collision between the blue and red agents, a straight line path from blue agent's initial to final position emerges as the optimal solution.

In Fig. 10, the green agent has been considered as the third priority, which means that for equation (22), the prediction sets of red and blue agents need to be taken into account. As a result, the green agent generates its trajectory with big deviation from the straight line to avoid the high priority agents.

7. Comparison of the proposed methods

This section briefly discusses the relative advantages and limitations of the proposed methods. Essentially, each method offers a trade-off between the computational effort and the optimality of the path.

7.1. Centralized approach

Given a time of flight for the vehicles, the centralized approach solves computationally expensive multi-phase fluid equation in porous medium to find the most optimal solution possible. On the other hand, the advantage of this method is that the optimal solution results in maintenance of the required separation between the vehicles.

7.2. Decentralized approach

This method assumes only the knowledge of total time of flight, and does not know exact trajectories of the other agents. The computations for this approach are fairly inexpensive comparing to the centralized solution as each of the vehicles is individually finding its best solution while avoiding the prediction sets of other vehicles, and hence the safety or collision avoidance is guaranteed. But the optimality is sacrificed in order to obtain paths that would ensure that an agent would avoid all possible paths of other agents given by their prediction sets.

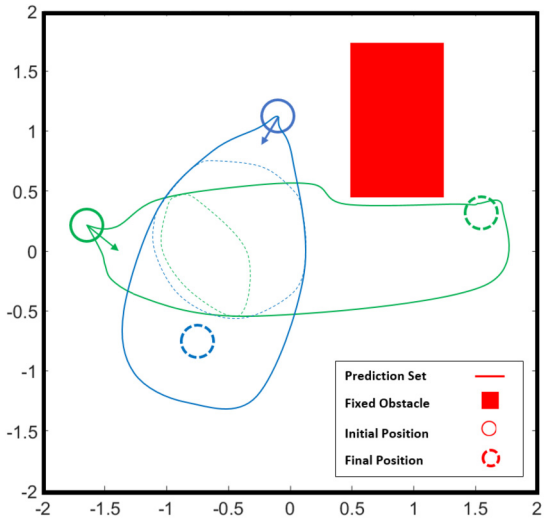

Fig. 11. The failed scenario where the $PS^{green}_{green}(t)$ for the green agent at time $t = 5$ is in the $PS^{blue}_{green}(t)$. The total time of travel is considered 15 seconds.

7.3. Sequential planning

In sequential planning, like decentralized approach, we assume the knowledge over the total time of flight. As expected, the solution is not optimal compared to the centralized controller but as the number of agents rises, it is more time consuming than the decentralized controller due to sequential hierarchy. The safety or collision avoidance is still guaranteed in the planning.

8. Analysis of decentralized solution convergence

In this section, we want to present the conditions for convergence of the solution for the decentralized approach. In this approach, generating the prediction sets over time t, $O^j_i(t)$, is the most critical aspect for convergence analysis. As mentioned in equation (7), set of PS_{total}, is a set of all trajectories, which for any agent, leads to the final position in a given time t. Let us define PS^i_i as the set of PSs, generated from the streamlines of agent i, which leads agent i to its goal position. The intuitive convergence criterion is that there should be at least one streamline in PS^i_i for the agent i, i.e.:

$$PS^i_i \neq \varnothing; \tag{23}$$

After relaxing this criterion, the set of obstacles caused by the movements of other agents for agent i, given by $O^j_i(t)$, requires to be satisfied. If the set of $O^j_i(t)$ at time step t, covers the whole $PS^i_i(t)$ at the same time step, then the decentralized solution would not converge. For understanding these criteria, consider the scenario, represented in Fig. 11.

In this scenario, as shown in Fig. 11, the initial positions and target positions of the agents are:

$$x^0_{green} = (-1.75, 0.25); \quad c^{\mathbb{T}}_{green} = (1.5, 0.3);$$
$$x^0_{blue} = (-0.2, 1.1); \quad c^{\mathbb{T}}_{blue} = (-0.75, -0.75);$$

The time of travel for the agents is 15 seconds. At $t = 5$ seconds, the prediction sets of two vehicles match together, which causes the solution not to converge to the final position. This example can be generalized by the following Theorem.

Theorem 2. *The convergence of the decentralized solution, provided in Section 6.2, is ensured if the following conditions are satisfied:*

Table 2
Comparison between centralized HJ technique presented in [48] and proposed centralized PDE method using PSs for different scenarios.

Number of Agents	Centralized HJ TS $\times 10^{-1}$ (s)	Centralized PDE TS $\times 10^{-1}$ (s)
3	7.3	4.1
5	15.36	4.3
10	24.39	6.7

$$PS^i_i \neq \varnothing; \tag{24}$$
$$PS^i_i(t) \nsubseteq \mathcal{G}_i(t); \tag{25}$$

Proof. The equation (24) guarantees the existing of at least one streamline which leads the agent from the initial point to final position. Equation (25) intuitively represents the positions that the agent i has at time t that have never been occupied by the prediction sets generated from other agents. □

8.1. Comparison with BRS method

In this section, we analyze the solution provided in this paper with Hamilton-Jacobian (HJ) variational inequality [22,20]. This method computes the BRSs of $\mathcal{V}(\mathbf{t})$. In simple words, the BRS is the set of states from which there exists a control such that, for all non-anticipative disturbances, the system is driven into the target set $\mathcal{T}(t)$ in the time horizon $[t, \mathbb{T}]$ without first entering the obstacle set $\mathcal{G}(\mathbf{t})$.

The solution for the BRS method can be explained as follows. We can find the states of all vehicles $\S = (x_1, x_2, ..., x_N)$, and generate the target set $\mathcal{T}(t)$ as:

$$\mathcal{T}(t) = \{\S : l(t, \S) \leq 0\} \tag{26}$$

$\mathcal{T}$ corresponds to all vehicles reaching to their targets and the obstacle set $\mathcal{G}(\mathbf{t})$ to correspond the combination of all risky zones $\mathcal{A}_{\mathbf{ij}}$. Then equation (27) is the HJ required to be solved for the optimal control problem.

$$\max\{\min\{D_t\mathbb{V}(t, \S) + \mathcal{H}(t, \S, D_\S\mathbb{V}), l(t, \S) - \mathbb{V}(t, \S)\} \\ -g(t, \S) - \mathbb{V}(t, \S)\} = 0, \tag{27}$$
$$\mathbb{V}(\mathbb{T}, \S) = \max\{l(\mathbb{T}, \S) - g(\mathbb{T}, \S)\} \tag{28}$$
$$\mathcal{H}(t, \S, \mathcal{P}) = \min\max_{\mathcal{P}} f(t, \S, u, d) \tag{29}$$

In equation (27), $\mathbb{V}$ is the viscosity [48]. Furthermore, $g(t, \S)$ is the implicit surface function. In equation (29), f is the governing control function. Combining all the values of $\mathbf{V} = \{\S : \mathbb{V} \leq 0\}$, would result in all the reachability sets, and the joint optimal control obtained is the solution of the equation (30).

$$u^*(t, \S) = arg\min\max_{u} \mathcal{H}(t, \S, D_\S\mathbb{V}) \tag{30}$$

Implementation of this algorithm for UAV trajectory planning is completely explained in [22]. The comparison between the centralized HJ and the proposed centralized PDE based method is provided in Table 2.

As it is obvious, the dimensionality of the joint state $\S$ affects the computational time requirement as it is provided in Table 2.

8.2. Effect of information update

In this section, we aim to study the scalability property of the proposed algorithm. The method is based on the calculation of PSs which are calculated at the beginning of the scenario based on

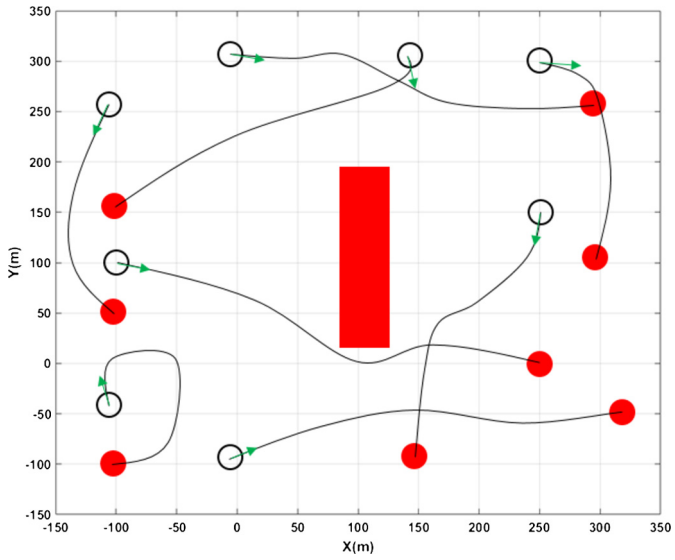

Fig. 12. Paths generated for the 8 agents within 27 seconds.

Table 3
A comparison between the effect of update rate on solution time, computational time and normalized cost.

Update rate	Mission time (seconds)	Computational time (seconds)	Normalized cost
1	20	70.2	1.04
2	22	52.1	1.09
3	27	46.9	1.19
4	29	45.0	1.33

initial states and goal positions. This leads to PSs to include all potential streamlines originating from initial states to final goals resulting into infeasible solutions. In practice, an update of PSs at intermediate stages could refine these PSs to discard those streamlines that have not been taken by the UAVs resulting into relevant and smaller PSs. To demonstrate this, we run a scenario for 8 agents as represented in Fig. 12. In this scenario, the maximum velocity of agent is $\overline{v} = 22m/s$ and the minimum velocity is $\underline{v} = 18$. The maximum turning rate is set to $\overline{\omega} = \pi/4$.

With the single calculation of PS at the beginning of the scenario, no feasible solutions are found. We then calculate the updated PSs based on agents' current states during the mission and study the effect of this update in convergence of the solutions. This scenario requires all the agents to reach to their respective goal locations withing the time bound of $[0, \mathbb{T}]$ where $\mathbb{T}$ is the cap time or maximum allowable time to complete the mission. To study the effect of update interval, we carry out the following task. For each update interval, we determine the mission time which represents the time taken to complete the mission. We also record normalized cost which refers to the cost (sum of total distance traveled by all agents) obtained using the proposed PDE solver divided by the cost obtained by the centralized BRS method which was found by updating the information at each time step. The cost obtained by the BRS method is calculated to be 5781.2.

For the above scenario, Table 3 presents the mission time, the time of computation, and the normalized cost for the different update intervals. The update interval considered in Fig. 12, is every 3 seconds. The cap of time $\mathbb{T}$ is considered 30 seconds.

It is intuitive and can be seen in Table 3 that the computational time decreases with increase in update interval. On the other hand, the mission time increases with the update rate. This is because, the agents are required to calculate the safest path for till the next update and it will cause them to go on a longer path if the update happens at larger interval. It is worth mentioning for any update interval greater than 4, the solution can not be obtained within maximum allowed time of 30 sec.

9. UAV scenario

In this section, we extend the application of the decentralized solution to a multi-Unmanned Aerial Vehicle (UAV) system in a three-dimensional environment with fixed obstacles have been studied. Each of the UAVs, follows the Dubins path dynamics given by equation (20). A scenario is generated with five UAVs. Initial conditions are shown in Fig. 13.

In this scenario, three UAVs, labeled as UAV1, UAV2 and UAV3 started from $(0, 0, 200)$, $(1800, 2200, 200)$ and $(2050, 400, 200)$, respectively. The goal location for these three vehicles are $(2200, 2250, 200)$, $(200, 0, 300)$ and $(0, 2200, 200)$, respectively. The positions are specified in meters. The maximum and minimum velocity for UAVs are set to be 15 and 5, meter per seconds. The total time of flight is specified as 50 seconds. As it can be seen in Fig. 13, UAVs were able to avoid obstacles in three dimensions while avoiding the obstacles. It is noteworthy that the advantage of using this method is that near optimal solutions in a computationally inexpensive manner are obtained.

10. Experimental results

In this paper, we experimentally validated our results via a simple task assignment problem. In the experiment, the objective was for UAVs to come to a formation while avoiding collision with each other. Subsequently, one of the UAVs was tasked to act as a faulty UAV and was tasked to land. In order to implement this, we assume to cut the network and sent a message to others that landing for this faulty agent would happen in certain seconds. The other UAVs were required to maintain separation from the faulty agent and stay in the formation. The separation is guaranteed as the UAVs predict the PSs of faulty agent at each time step and avoid locating themselves into those certain positions. The only communication that happened between each of the agents were at the beginning of the mission when they inform the others about their respective goal locations and times of arrival.

The UAV platforms used were Ascending Technologies, GmbH which are equipped with an IMU (accelerometer, gyroscope, magnetometer) and pressure sensor. Localization is carried out using a motion capture system and the data is recorded in a log file. The onboard computation unit is a 1.6 GHz Atom processor with 1 GB of RAM. Communication with the quadcopters for monitoring experiment progress is via 802.11n or 802.11s networking. All algorithm development is in C++ using ROS and MAVlink protocol was used to establish the communication of ground station with quadcopters.

The data delivered from the motion capture system was filtered to reduce any noise in the data. The scenario involved 3 quadcopters, initially located in $(2, 6, 2)$, $(12, 1, 1.5)$ and $(14, 7, 2)$, reporting their arrival time as 12 seconds and their respective goal locations as $(5.5, 3, 1.5)$, $(7, 3.5, 1.5)$ and $(6.2, 4.1, 1.5)$. Computations were all done on-board. Fig. 14, represents the initial planning of the experiment for the entire 12 second of mission time.

The second part of the experiment was designed to validate the fast on-board computation of safe paths using the proposed method. One of the agents (faulty agent) was cut off from the network and then based on the safety protocol, defined $a - priori$, each of the quadcopters became aware of the faulty agent and the landing mission of the faulty agents in 7 seconds from the current location. The other agents were set to maintain 2 meter separation in Z direction and maintain the formation. Fig. 15 shows the positions of UAVs while they are maintaining their separation from

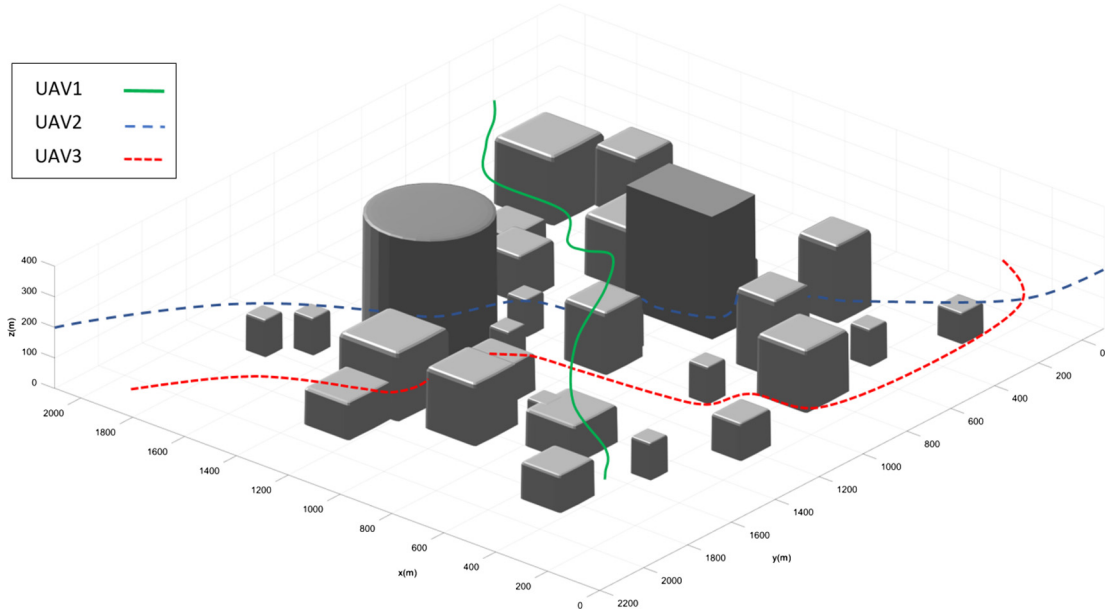

Fig. 13. Three UAVs flying in an urban environment containing obstacles. The time of flight is considered to be 50 seconds. The decentralized flight controller is assigned where the UAVs are only aware of the total time of flight $a-priori$.

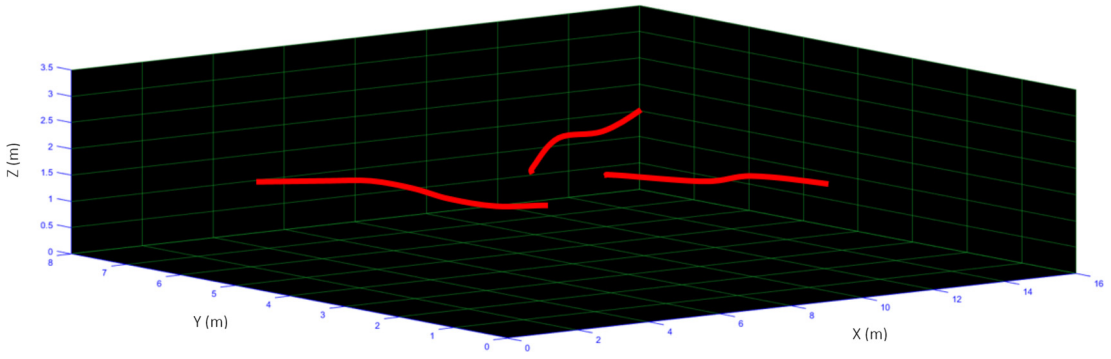

Fig. 14. Paths generated for the three UAVs for 12 seconds.

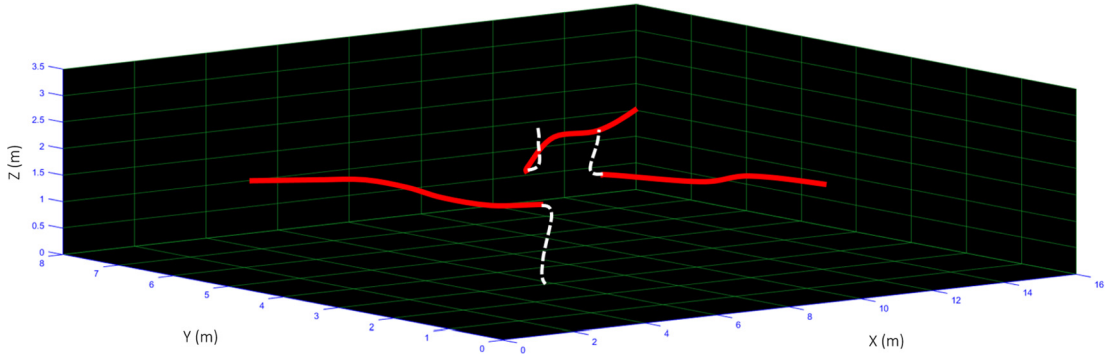

Fig. 15. Three UAVs generated safe paths (shown in dashed lines) after the fault was detected in one of the agents which landed.

the faulty UAV. The paths generated when the fault is detected is shown by dashed lines. Fig. 16 shows after the faulty UAV landed and safe paths were generated.

11. Conclusion

The formulation and solution of multi-UAV trajectory planning problem in the framework of Partial Differential Equations (PDEs) are presented in this paper. The framework implements the multi-UAV trajectory planning method using the PDEs that represent multi-phase fluid flow in porous media. First, the solution for the PDE is obtained using a proposed centralized approach. A comparison of this method with the MILP approach which provides globally optimal solution confirms the optimality of the proposed approach while providing an advantage in computational efficiency. Then, by defining the prediction sets, a decentralized approach for finding the collision-free and optimal path is proposed. The prediction sets are obtained using the information on time of flight for each UAV. Finally, a sequential approach is presented that carries out sequential trajectory planning based on priority of UAVs.

Fig. 16. Faulty quadcopter has landed while the other two quadcopters maintained separation while calculating the PSs for the faulty agent. Safety protocol dictates that the faulty agents require to land in 3 seconds. The other two UAVs would calculate equation (21).

In all of methods, all vehicles are guaranteed to successfully arrive at their goal locations without any collisions. Out of the three methods, the decentralized method provides a good trade-off between computational requirement and optimality. This method was shown to be applied in a simulated 3D environment using a realistic scenario comprising of 3 UAVs and buildings as stationary obstacles. Finally, the proposed method was validated via an experiment using three UAVs where decentralized computations were carried out onboard the UAVs.

Declaration of competing interest

The authors hereby claim that this project does not have any conflict of interests.

References

[1] P. Tokekar, J. Vander Hook, D. Mulla, V. Isler, Sensor planning for a symbiotic UAV and UGV system for precision agriculture, IEEE Trans. Robot. 32 (6) (2016) 1498–1511.

[2] S. Hayat, E. Yanmaz, T.X. Brown, C. Bettstetter, Multi-objective UAV path planning for search and rescue, in: Robotics and Automation (ICRA), 2017 IEEE International Conference on, IEEE, 2017, pp. 5569–5574.

[3] J. Rios, D. Mulfinger, J. Homola, P. Venkatesan, NASA UAS traffic management national campaign: operations across six UAS test sites, in: Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th, IEEE, 2016, pp. 1–6.

[4] T.S. Perry, In search of the future of air traffic control, IEEE Spectr. 34 (8) (1997) 18–35.

[5] M. Radmanesh, M. Kumar, Flight formation of UAVs in presence of moving obstacles using fast-dynamic mixed integer linear programming, Aerosp. Sci. Technol. 50 (2016) 149–160.

[6] P. Chandler, S. Rasmussen, M. Pachter, UAV cooperative path planning, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, 2000, pp. 1255–1265.

[7] D. Panagou, M. Turpin, V. Kumar, Decentralized goal assignment and trajectory generation in multi-robot networks: a multiple Lyapunov functions approach, in: Robotics and Automation (ICRA), 2014 IEEE International Conference on, IEEE, 2014, pp. 6757–6762.

[8] H. Min, F. Sun, F. Niu, Decentralized UAV formation tracking flight control using gyroscopic force, in: Computational Intelligence for Measurement Systems and Applications, 2009. CIMSA'09. IEEE International Conference on, IEEE, 2009, pp. 91–96.

[9] R.W. Beard, T.W. McLain, D.B. Nelson, D. Kingston, D. Johanson, Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs, Proc. IEEE 94 (7) (2006) 1306–1324.

[10] D. Panagou, D.M. Stipanović, P.G. Voulgaris, Distributed coordination control for multi-robot networks using Lyapunov-like barrier functions, IEEE Trans. Autom. Control 61 (3) (2016) 617–632.

[11] X. Gu, Y. Zhang, J. Chen, L. Shen, Real-time decentralized cooperative robust trajectory planning for multiple UCAVs air-to-ground target attack, Proc. Inst. Mech. Eng., Part G, J. Aerosp. Eng. 229 (4) (2015) 581–600.

[12] M. Innocenti, L. Pollini, A. Bracci, Cooperative path planning and task assignment for unmanned air vehicles, Proc. Inst. Mech. Eng., Part G, J. Aerosp. Eng. 224 (2) (2010) 121–131.

[13] Y. Kuwata, J.P. How, Cooperative distributed robust trajectory optimization using receding horizon MILP, IEEE Trans. Control Syst. Technol. 19 (2) (2011) 423–431.

[14] T.W. McLain, R.W. Beard, Coordination variables, coordination functions, and cooperative-timing missions, J. Guid. Control Dyn. 28 (1) (2005) 150–161.

[15] I. Kaminer, O. Yakimenko, V. Dobrokhodov, A. Pascoal, N. Hovakimyan, C. Cao, A. Young, V. Patel, Coordinated path following for time-critical missions of multiple UAVs via l1 adaptive output feedback controllers, in: AIAA Guidance, Navigation and Control Conference and Exhibit, 2007, p. 6409.

[16] J. Van den Berg, M. Lin, D. Manocha, Reciprocal velocity obstacles for real-time multi-agent navigation, in: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, IEEE, 2008, pp. 1928–1935.

[17] J. Van Den Berg, S. Guy, M. Lin, D. Manocha, Reciprocal n-body collision avoidance, Robot. Res. (2011) 3–19.

[18] R. Olfati-Saber, R.M. Murray, Distributed cooperative control of multiple vehicle formations using structural potential functions, IFAC Proc. Vol. 35 (1) (2002) 495–500.

[19] Y.-L. Chuang, Y.R. Huang, M.R. D'Orsogna, A.L. Bertozzi, Multi-vehicle flocking: scalability of cooperative control algorithms using pairwise potentials, in: Robotics and Automation, 2007 IEEE International Conference on, IEEE, 2007, pp. 2292–2299.

[20] J.F. Fisac, M. Chen, C.J. Tomlin, S.S. Sastry, Reach-avoid problems with time-varying dynamics, targets and constraints, in: Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, ACM, 2015, pp. 11–20.

[21] A.M. Bayen, I.M. Mitchell, M.M. Oishi, C.J. Tomlin, Aircraft autolander safety analysis through optimal control-based reach set computation, J. Guid. Control Dyn. 30 (1) (2007) 68.

[22] S. Bansal, M. Chen, J.F. Fisac, C.J. Tomlin, Safe sequential path planning of multi-vehicle systems under presence of disturbances and imperfect information, in: American Control Conference, 2017.

[23] A. Quarteroni, Numerical Models for Differential Problems, vol. 2, Springer Science & Business Media, 2010.

[24] M. Radmanesh, P.H. Guentert, M. Kumar, K. Cohen, Analytical PDE based trajectory planning for unmanned air vehicles in dynamic hostile environments, in: American Control Conference (ACC), 2017, IEEE, 2017, pp. 4248–4253.

[25] P. Barooah, P.G. Mehta, J.P. Hespanha, Mistuning-based control design to improve closed-loop stability margin of vehicular platoons, IEEE Trans. Autom. Control 54 (9) (2009) 2100–2113.

[26] P. Frihauf, M. Krstic, Leader-enabled deployment onto planar curves: a PDE-based approach, IEEE Trans. Autom. Control 56 (8) (2011) 1791–1806.

[27] A. Sarlette, R. Sepulchre, A pde viewpoint on basic properties of coordination algorithms with symmetries, in: Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on, IEEE, 2009, pp. 5139–5144.

[28] L.C. Pimenta, G.A. Pereira, N. Michael, R.C. Mesquita, M.M. Bosque, L. Chaimowicz, V. Kumar, Swarm coordination based on smoothed particle hydrodynamics technique, IEEE Trans. Robot. 29 (2) (2013) 383–399.

[29] B. Piccoli, F. Rossi, E. Trélat, Control to flocking of the kinetic Cucker–Smale model, SIAM J. Math. Anal. 47 (6) (2015) 4685–4719.

[30] K. Elamvazhuthi, H. Kuiper, S. Berman, PDE-based optimization for stochastic mapping and coverage strategies using robotic ensembles, Automatica 95 (2018) 356–367.

[31] K. Elamvazhuthi, H. Kuiper, M. Kawski, S. Berman, Bilinear controllability of a class of advection-diffusion-reaction systems, IEEE Trans. Autom. Control (2018).

[32] A. Galstyan, T. Hogg, K. Lerman, Modeling and mathematical analysis of swarms of microscopic robots, in: Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005, IEEE, 2005, pp. 201–208.

[33] A. Prorok, N. Correll, A. Martinoli, Multi-level spatial modeling for stochastic distributed robotic systems, Int. J. Robot. Res. 30 (5) (2011) 574–589.

[34] N. Correll, H. Hamann, Probabilistic modeling of swarming systems, in: Springer Handbook of Computational Intelligence, Springer, 2015, pp. 1423–1432.

[35] S. Engebretsen, A PDE based approach to path finding in three dimensions: Solving a path finding problem for an unmanned aerial vehicle, Master's thesis, Institutt for matematiske fag, 2014.

[36] A.R. Mitchell, D.F. Griffiths, The Finite Difference Method in Partial Differential Equations, John Wiley, 1980.

[37] J.H. Ferziger, M. Peric, Computational Methods for Fluid Dynamics, Springer Science & Business Media, 2012.

[38] S. Trehan, L.J. Durlofsky, Trajectory piecewise quadratic reduced-order model for subsurface flow, with application to PDE-constrained optimization, J. Comput. Phys. 326 (2016) 446–473.

[39] M. Cardoso, L.J. Durlofsky, Linearized reduced-order models for subsurface flow simulation, J. Comput. Phys. 229 (3) (2010) 681–700.

[40] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, Y.-J. Jan, A front-tracking method for the computations of multiphase flow, J. Comput. Phys. 169 (2) (2001) 708–759.

[41] J. He, J. Sætrom, L.J. Durlofsky, Enhanced linearized reduced-order models for subsurface flow simulation, J. Comput. Phys. 230 (23) (2011) 8313–8341.

[42] J. He, L.J. Durlofsky, et al., Reduced-order modeling for compositional simulation by use of trajectory piecewise linearization, SPE J. 19 (05) (2014) 858–872.
[43] S. Trehan, K. Carlberg, L.J. Durlofsky, Error estimation for surrogate models of dynamical systems using machine learning, arXiv preprint arXiv:1701.03240, 2017.
[44] E.F. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction, Springer Science & Business Media, 2013.
[45] G. Liu, K. Dai, T.T. Nguyen, A smoothed finite element method for mechanics problems, Comput. Mech. 39 (6) (2007) 859–877.
[46] D. Talay, Numerical Solution of Stochastic Differential Equations, 1994.
[47] Z. Xu, R. Jin, H. Yang, I. King, M.R. Lyu, Simple and efficient multiple kernel learning by group LASSO, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), Citeseer, 2010, pp. 1175–1182.
[48] S. Bansal, M. Chen, S. Herbert, C.J. Tomlin, Hamilton-Jacobi reachability: a brief overview and recent advances, in: Decision and Control (CDC), 2017 IEEE 56th Annual Conference on, IEEE, 2017, pp. 2242–2253.