A SIMULINK APPROACH TO MODELING HETEROGENEOUS DRONE FORMATIONS FOR AGENT-BASED SIMULATION

Gennaro Raspaolo[1], Giuliano D'Alterio[1], Immacolata Notaro[1] & Egidio D'Amato[2]

[1]Department of Engineering, University of Campania Luigi Vanvitelli, 81031, Aversa (CE), Italy
[2]Department of Science and Technology, University of Naples Parthenope, 80143, Napoli, Italy

Abstract

In the dynamic world of autonomous systems, the demand for accurate simulation platforms has grown exponentially, enabling rigorous testing of complex algorithms and strategies in a risk-free virtual environment before real-world deployment. An effective simulator, crucial to researchers, should possess accuracy, reliability, flexibility, an intuitive interface, support for various simulation modes, realistic environment representation, data tracking, parallelization, and thorough documentation. In this paper, the integration of these features in a simulink-based simulator is approached.

Keywords: UAV, Quadcopter, Simulation, Matlab, Simulink

1. Introduction

The use of small Unmanned Aerial Systems (sUAS) is expanding across diverse fields, introducing both advancements and challenges. Notably, autonomous drone swarms have emerged as a significant technological development [1]. Current sUAS are commonly equipped with collision detection and avoidance systems to enhance safety and longevity [2, 3]. These systems are integral in the creation of drone swarm platforms, addressing challenges such as extensive area scanning and three-dimensional space coverage. However, given the novelty of drone swarm research, in the ever-evolving landscape of autonomous systems, the demand for sophisticated simulation platforms has become always more important. In this context, simulations are invaluable tools, allowing researchers and engineers to rigorously test and validate complex algorithms, strategies, and formations in a risk-free virtual environment before real-world deployment. For instance, the presence of any kind of fault is difficult to emulate on real prototype, and a digital twin can become useful for testing fault detection algorithms [4, 5].
An effective simulator must include several requirements to ensure accuracy, reliability, and flexibility:

- Accuracy and reliability: the simulator must be precise and validated to ensure that results faithfully reflect real-world system behaviors.
- Flexibility and modularity: the ability to easily modify the simulator to adapt to different scenarios or incorporate new agents, rules, or environments without rebuilding the entire system is crucial.
- Intuitive user interface: An intuitive user interface simplifies simulation design, result visualization, and adjustment of simulation parameters.
- Support for several simulation modes: the simulator should support various simulation modes, such as real-time simulation, batch simulation, and accelerated simulation.
- Realistic environment representation: it should represent the environment in which the simulation takes place, including details such as weather conditions, terrain, obstacles, and other environmental variables.

- Data tracking and analysis: a good simulator should allow easy tracking and analysis of simulation data to extract valuable insights and assess system performance.
- Support for parallelization: the capability to run simulations in parallel can significantly accelerate execution times, allowing efficient simulation of complex scenarios.
- Adequate documentation: clear and comprehensive documentation is essential to ensure that users can understand and correctly utilize all features of the simulator.
- Possibility of validation and verification: The simulator should allow validation and verification of obtained results, providing users with confidence in the accuracy and reliability of simulations.

Integrating these characteristics into a simulator will contribute to creating a powerful and versatile tool for executing accurate and informative simulations in a variety of contexts.
The focus of the proposed simulator is on a heterogeneous autonomous system comprised of N_r agents, distributed as follows:

- N_a fixed-wing UAVs: These Unmanned Aerial Vehicles boast extended endurance and payload capacity, strategically employed to ensure heightened temporal resolution, albeit with a trade-off in spatial coverage.
- N_h Rotorcraft Unmanned Aerial Vehicles (RUAV) with VTOL capabilities: Specifically designed for surveillance and monitoring, these vehicles offer superior spatial resolution, thanks to their Vertical Take-Off and Landing capabilities.
- N_g Unmanned Ground Vehicles or Unmanned Surface Vessels (UGV-USV): These ground or water-based robotic entities serve as mobile base stations, equipped with substantial payload capacities and autonomy.

Our work focuses on the development of a simulator to be used in the Simulink environment, a state-of-the-art tool for system-level modeling and simulation. Simulink provides a versatile and intuitive platform, with an intuitive user interface as well as data analysis and parallelization support, useful to model the dynamics of heterogeneous drone formations.
The proposed approach is based on the principles of agent-based simulation, where each agent operates autonomously, governed by internal rules and responsive to dynamic interactions with the environment and other agents. This methodology allows us to make decentralized formation structure and capture emergent behaviors resulting from the collective actions of individual agents.
An agent is defined as an autonomous software entity positioned within environment, capable of actions and coordination with other agents to achieve predefined goals [6]. Consequently, Multi-Agent System (MAS) technology has evolved as a framework for embedding autonomous behavior and decision-making into computer systems [7, 8]. An Agent-Based Simulation (ABS) model comprises interacting intelligent entities mirroring real-world relationships within an artificial environment [9].
Through a visual and modular interface the proposed simulator integrates several agent models and environmental factors, incorporating the complexities of real-world scenarios and providing a high-fidelity representation of the interactions between heterogeneous drone agents, mission objectives, and environmental conditions.
In its first version, the proposed simulator implements not only the dynamics of the agents but also their autopilots to ensure that each vehicles can follow an assigned reference state, in order to make it ready to use for example in path planning problems [10].
The objective is to make a tool that is able to facilitate the design of an autonomous formation, capable of fulfilling mission objectives while navigating through a complex web of constraints. These constraints span a spectrum that includes vehicle characteristics, temporal constraints, spatial limitations, task sequencing, obstacles, as well as weather and terrain considerations.

2. Dynamic model

In order to define the equations of motion the reference frames must be defined. The most common choice in aeronautics, regarding the fixed reference frame, is the NED reference (North-East-Down), depicted in figure 1, whose X_E-axis points to the local North, the Y_E-axis towards East and the Z_E-axis completes the right-handed frame by pointing downwards, towards the centre of the Earth. Finally, the center O of the reference is on the surface of the Earth. Strictly speaking, this reference frame is not inertial due to the Earth rotation but, since this rotation is slow compared to the vehicle motion, the inertial approximation is reasonable. As local reference frame, we can consider a body frame with the X_B-axis aligned with the longitudinal reference line, the Z_B-axis, normal to X_B to build the symmetry plane of the aircraft and the Y_B-axis to form a right-handed frame. NED and body reference frames are shown in Figure 1.

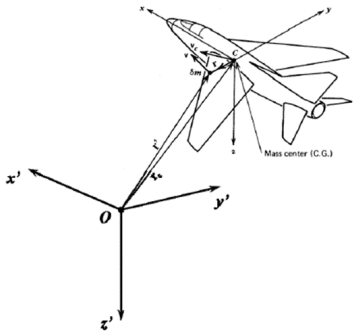


Figure 1 – NED reference frame.

In the body reference frame, Euler equations can be derived as follows:

$$\begin{aligned} F &= m(\dot{v}_c + \tilde{\omega}_B \times v_C) \\ M &= I_B \dot{\omega}_B + \tilde{\omega}_B \times I_B \omega_B \end{aligned} \tag{1}$$

where $\boldsymbol{F}$ represents the external forces acting on the vehicle and linked to the motion of the centre of mass, $\boldsymbol{M}$ is the external moment acting on the centre of mass and calculated as the derivative of the angular momentum $\boldsymbol{h}$, $v_c = [U,V,W]^T$ is the center of mass velocity vector, $\boldsymbol{I}$ is the matrix of inertia of the vehicle and $\tilde{\omega}_B = [P,Q,R]^T$ represents its angular velocity.
To know the pose of the aircraft in the NED reference frame, the following equations are needed:

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix}_E = \boldsymbol{R}_{EB} \begin{Bmatrix} U \\ V \\ W \end{Bmatrix}_B \tag{2}$$

$$\begin{Bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} = T \begin{Bmatrix} P \\ Q \\ R \end{Bmatrix} \tag{3}$$

Equation 2 can be integrated to give the trajectory coordinates $x(t)$, $y(t)$, $z(t)$, where R_{EB} represents the rotation matrix between the NED and the body reference frame, where the rotation angles along the body axis are called Euler angles, in particular yaw ψ (Z-axis), pitch θ (Y-axis), roll ϕ (X-axis).

$$\boldsymbol{R}_{EB} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ -\cos\theta\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\cos\theta \\ \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix} \tag{4}$$

Equation 3 can be integrated to find the attitude of the aircraft in terms of Euler angles $\phi(t)$, $\theta(t)$, $\psi(t)$, with $\boldsymbol{T}$ a matrix to bind the angular velocity vector in body frame with the derivative of the Euler angles.

$$\boldsymbol{T} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \tag{5}$$

2.1 Forces and moments

To complete the equations of motion, three types of forces and moments are considered: gravitational, propulsion and aerodynamics. The gravitational forces can be expressed in the body frame considering a generic orientation of the vehicle and using matrix 4, as shown in 6.

$$\begin{bmatrix} X_g \\ Y_g \\ Z_g \end{bmatrix}_B = T_{BI} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}_I \tag{6}$$

The gravitational forces are applied in the center of gravity of the vehicle so they do not contribute to the external moments. Propulsion and aerodynamics will be different for UAV and RUAV.
For the UAV model, a preliminary function to compute the propulsion force considers only the thrust applied along the X axis and centered in the center of gravity, thus not contributing to the moments. The aerodynamic forces (X,Y,Z) and moments (L,M,N) acting on the UAV model are shown in 7, expressed in terms of non-dimensional coefficients $C_x, C_y, C_z, C_l, C_m, C_n$, where ρ is the density, $V_C = |\boldsymbol{v}_c|$ the cruise speed, S the wing surface, b the wing span, c the chord.

$$\begin{aligned} X &= \frac{1}{2}\rho V_C^2 S C_x & L &= \frac{1}{2}\rho V_C^2 S b C_l \\ Y &= \frac{1}{2}\rho V_C^2 S C_y & M &= \frac{1}{2}\rho V_C^2 S c C_m \\ Z &= \frac{1}{2}\rho V_C^2 S C_z & N &= \frac{1}{2}\rho V_C^2 S b C_n \end{aligned} \tag{7}$$

However, usually a wind reference frame is considered to express aerodynamics forces and moments acting on an aircraft. In particular, the net aerodynamic force can be decomposed in drag (X_W wind axis component) and lift (Z_W wind axis component). The orientation of the wind reference frame with respect to the body reference frame is given by two angles, namely the angle of attack α and the side-slip angle β. These two angles can be computed given the components u,v,w which represent the decomposition of the velocity vector along the x,y,z body axis. The relationship between α and β with u,v,w is shown in equation 8, in which appears also the True AirSpeed (TAS).

$$\begin{aligned} \alpha &= \arctan\frac{w}{u} \\ \beta &= \arcsin\frac{v}{TAS} \\ TAS &= \sqrt{u^2+v^2+w^2} \end{aligned} \tag{8}$$

Ideally, the non-dimensional coefficients can be determined numerically through aerodynamic calculations and/or experimentally through wind tunnel tests and tabulated based on the system state.
To simplify the simulator, the aerodynamic forces ($\boldsymbol{F}$) and moments ($\boldsymbol{M}$) acting on an aircraft can be expressed as functions of the state variables and control surface deflections. For small perturbations around a trim condition, these functions can be linearized using a Taylor series expansion.
The general form of the Taylor series expansion for the aerodynamic force or moment (which can represent any aerodynamic force or moment) around a trim state is given by:

$$G = G_0 + \frac{\partial G}{\partial U}\Delta U + \frac{\partial G}{\partial V}\Delta V + \frac{\partial G}{\partial W}\Delta W + \cdots \tag{9}$$

where:

- G_0 is the value of the force or moment at the trim condition.
- $\frac{\partial G}{\partial U}$, $\frac{\partial G}{\partial V}$, and $\frac{\partial G}{\partial W}$ are the aerodynamic derivatives.

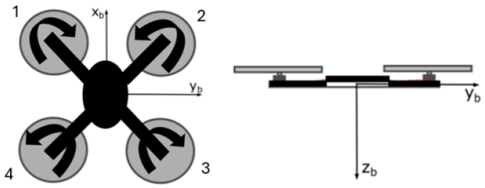


Figure 2 – Schematic of the RUAV convention considered.

- ΔU, ΔV, and ΔW are small perturbations of the state from the trim condition.

The aerodynamic derivatives represent the sensitivity of the forces and moments to changes in the state variables and control inputs.
On the other hand, for the RUAV model the aerodynamic model is neglected, while the propulsion force is computed considering a symmetric rotorcraft, with $2N$ motors which contribute to the external forces, along the Z axis, and to the external moments as shown in eq.10, where l is the distance between the rotors and the centre of gravity.

$$\begin{aligned} F_z &= -\sum_{i}^{2N} T_i \\ M_x &= l \cdot \sum_{i}^{N} T_i^{x,right} - l \cdot \sum_{i}^{N} T_i^{x,leftt} \\ M_y &= l \cdot \sum_{i}^{N} T_i^{y,right} - l \cdot \sum_{i}^{N} T_i^{y,left} \\ M_z &= l \cdot \sum_{i}^{N} T_i^{z,right} - l \cdot \sum_{i}^{N} T_i^{z,left} \end{aligned} \tag{10}$$

where $T_i^{r,right}$ and $T_i^{r,left}$ indicate that motor i is located at right or a t left of r-axis.

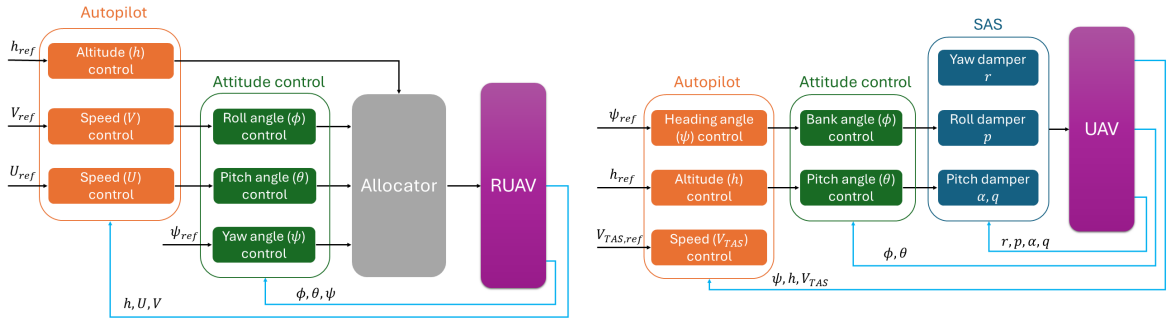
3. Control Systems

To make the simulator ready to use in path planning problems, each kind of aircraft is equipped with a customizable controller. The fixed-wing UAV control system is based on classic control [11], implementing the Stability Augmentation System (SAS) to enhance aircraft stability, the Control Augmentation System (CAS), to follow a reference signal in terms of attitude.
Figure 3 shows the control systems schemes for both types of vehicles. The controllers for UAV and RUAV differ in how the control action is actuated. In a fixed-wing aircraft, the vehicle state is changed by modifying the thrust or the deflections of the mobile surfaces. In a rotorcraft, the state of the system is changed by modifying the thrust of each propeller. The RUAV considered in this work has four motors and a cross structure, as in figure 2, with the rotors 1 and 3 rotating clockwise and 2 and 4 anticlockwise. In order to use SISO (Single Input Single Output) controllers with the RUAV model an allocation function is necessary. This function converts the commands on thrust, yaw, pitch or roll in thrust commands for every motor. The allocation function is shown in equation 11.

$$\begin{bmatrix} T1 \\ T2 \\ T3 \\ T4 \end{bmatrix} = \begin{bmatrix} T - \psi + \theta + \phi \\ T + \psi + \theta - \phi \\ T - \psi - \theta - \phi \\ T + \psi - \theta + \phi \end{bmatrix} \tag{11}$$

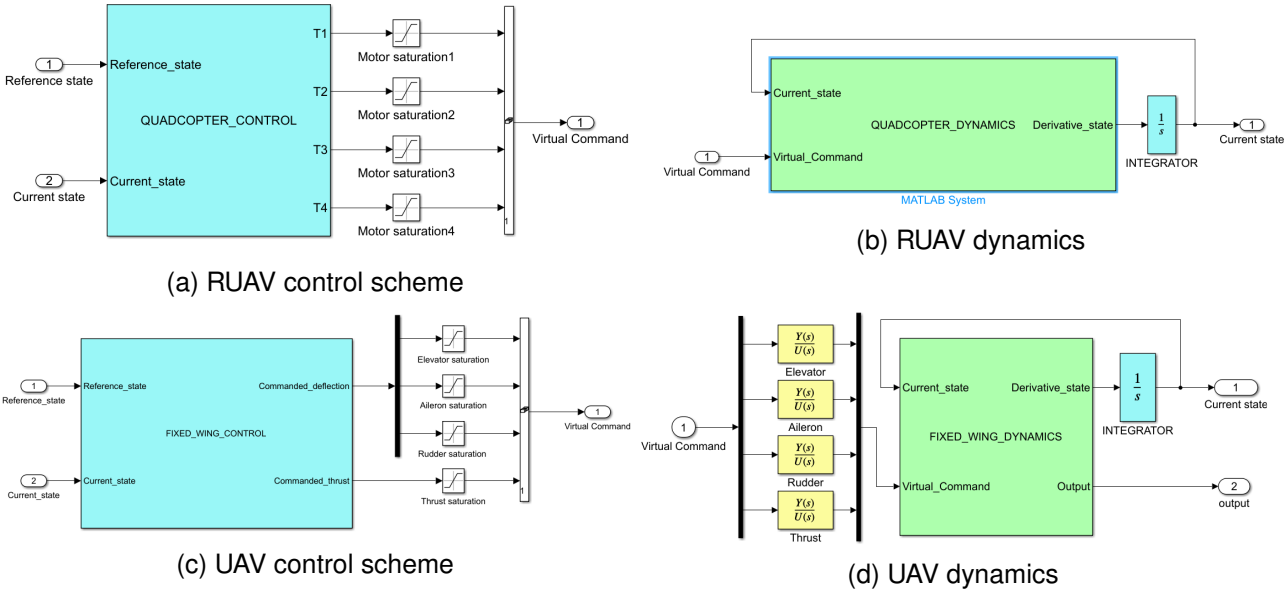
4. Architecture

To simplify the operations and make the simulator versatile, the implementation is based on the Matlab system class, which is cross-compatible between both Matlab and Simulink, making the simulator runnable both in the Matlab-only or Simulink environments.



(a) RUAV. (b) UAV.

Figure 3 – Schematic representation of the autopilots for both RUAV and UAV.



(a) RUAV control scheme (b) RUAV dynamics

(c) UAV control scheme (d) UAV dynamics

Figure 4 – Dynamic and control schemes for single RUAV and single UAV.

In this section the structure of the simulator is analyzed following a top-down approach, starting from the single-vehicle blocks which are the base of the final multi-vehicle blocks.

4.1 Single vehicle blocks

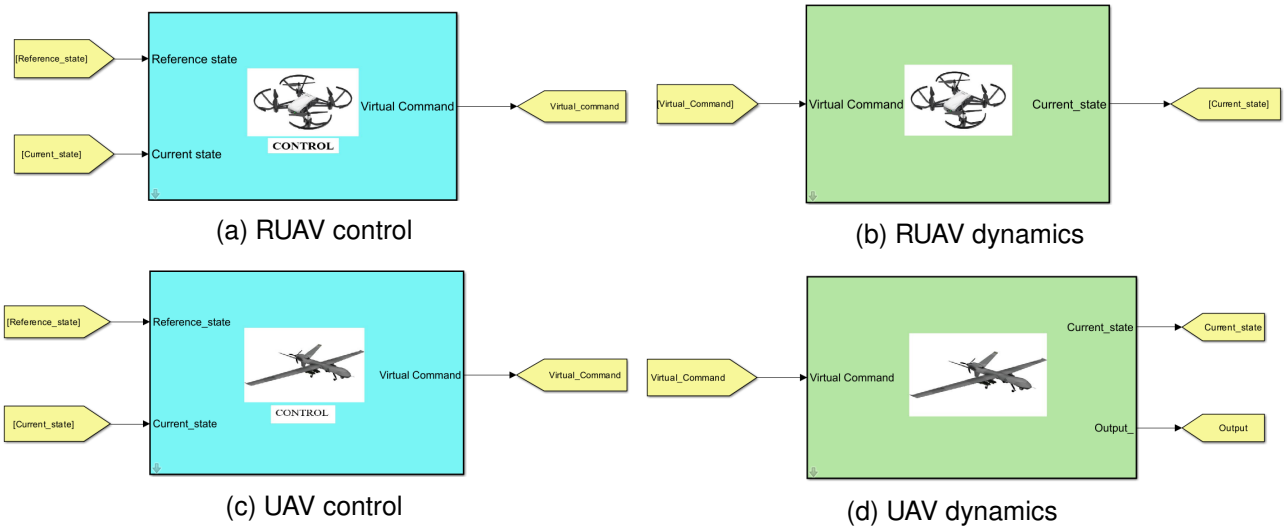
Figure 4 presents the appearance of the four schemes implemented for the two types of single-vehicle: UAV Control, UAV Dynamic Model, RUAV Control and RUAV Dynamic Model. The description of the Input/Output signals connecting blocks is represented in table **??**.
The details of each of the single-vehicle schemes are described in the following:

- **Control blocks**
  Both the UAV Control block and the RUAV Control block are composed of a Matlab system and four saturation blocks, as shown in figure 4a and figure 4c. The saturations of the UAV impose limits on the deflection angles of the control surfaces and on the thrust. The saturations of the RUAV impose limits on each propeller speed. For both blocks, The Matlab system accepts as input a reference signal vector $Reference\_state$ and a vector $Current\_state$ representing the state at time step t. The output of the block is a vector $Virtual\_Command$ representing the control actions. An integrator block is not needed because the implemented controllers are digital, thus the integration is carried out in the Matlab system. In the Matlab system, the UAV and RUAV controls are implemented as discussed in section 3.with the properties of the class representing the characteristics of the particular controller. Lastly, in **??** is shown the composition of the vectors present in figure 4a and figure 4c.

- **Dynamic Model blocks**
  The UAV Dynamic Model block consists of four transfer functions, representing the actuators' dynamics, a Matlab system and an integrator block, as shown in figure 4b. The RUAV Dynamic Model block is made of a Matlab system and an integrator block, as shown in figure 4d. For both blocks, The Matlab system accepts an input signal vector $Virtual\_Command$ and a vector $Current\_state$ representing the state at time step t. The output of the block is a vector $Derivative\_state$ representing the state derivative. The integrator is used to calculate the state from the state derivative. Moreover, the integrator holds the starting condition of the state $Current\_state(0)$. In the Matlab system, the UAV and RUAV models are implemented as discussed in section 2. with the properties of the class representing the characteristics of the particular vehicle.

4.2 Multi vehicle blocks

The Multi-vehicle blocks final appearance, as presented to the users, is shown in figure 5. These blocks are the generalization of the single-vehicle systems and are useful to avoid cluttering the workspace. Every multi-vehicle block works with matrices in input and output, where different vehicles are represented by different columns. The Input/Output signals of the multi-vehicle blocks have the same structure of the single-vehicle ones, but different dimensions. The multi-vehicle blocks take in input matrices, having the same number of rows as the single-vehicle column vectors and as many as the number of vehicles considered.



(a) RUAV control (b) RUAV dynamics

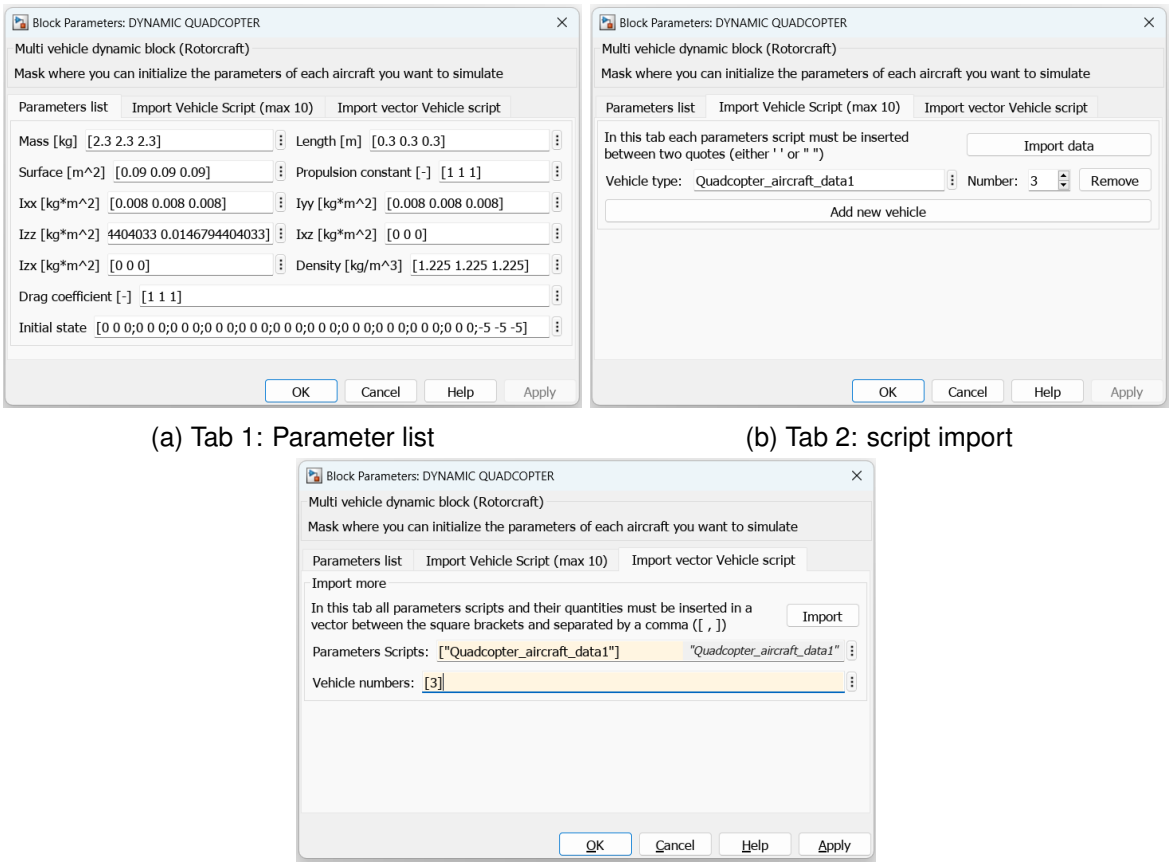
(c) UAV control (d) UAV dynamics

Figure 5 – Final appearance of the implemented dynamic and control blocks for multi-vehicle simulation.

Every multi-vehicle block is implemented with a Simulink *for each*, which makes use of the single-vehicle blocks presented in subsection 4.1 It is worth noting that the *for each* is also used to partition the vehicle parameters. These parameters should then be defined as a row vector or a matrix, where each column represents a different vehicle.
Figure 6 shows the interface of the RUAV dynamic block. To make things easier, each multi-vehicle block has an intuitive interface created with the Simulink mask editor, which presents three different modes to insert vehicle parameters: direct insert in the parameter list (figure 6a), interactive script import with spinner (figure 6b) and vector script import (figure 6c) by defining two arrays, one with the script's name and the other with the vehicle's number for each script. The vehicle details can be inserted in the first tab either by typing the values in full or by referring to MATLAB workspace variables. Another option is to write scripts containing the characteristics of different vehicles and import the data by using the name of the scripts and the number of vehicles for each script.

5. Use test-case

In this section, an example of the use of the simulator on Simulink is shown.



(a) Tab 1: Parameter list

(b) Tab 2: script import

(c) Tab 3: vector script import

Figure 6 – User interface to import the data for the RUAV dynamic block. The three different tabs give access to different import methods.

In particular, a scenario with three RUAV with the same characteristics and a starting altitude of 5 meters is used. To simulate the system, a few blocks are necessary: from the proposed library the dynamic and control blocks are picked; a zero-order hold is used to ensure that the digital controller time matches the time of the simulation; a delay is necessary to ensure data integrity during the process. Figure 7 represents the final scheme used to simulate the use case.

The use of the simulator in Matlab is equally intuitive. It is sufficient to initialize the desired number of vehicles by creating one instance of the class for each vehicle. Then, the step method can be used to carry out the calculation at each step.

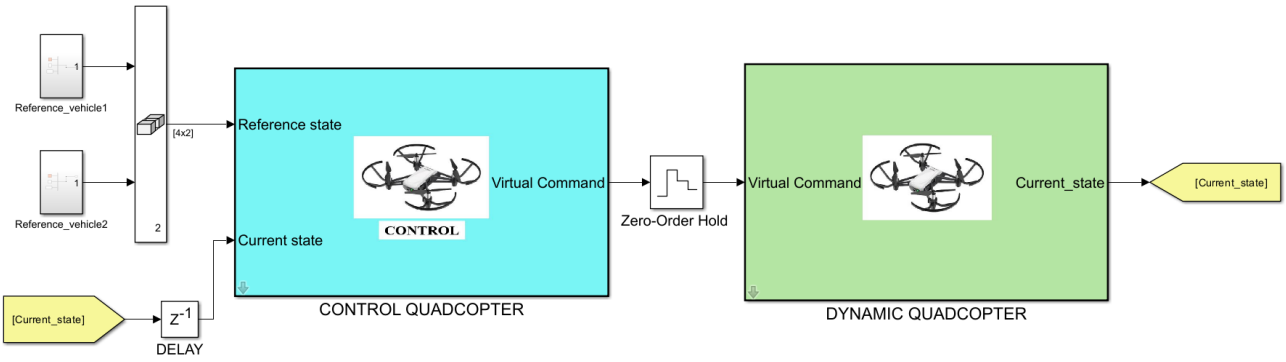


Figure 7 – Simulink scheme of an example scenario with 2 RUAV with different references to follow.

6. Conclusions

In this paper, we propose a complete solution to model heterogeneous drone formations in Simulink. The proposed simulator integrates elements of fidelity, reliability, soft real-time performance, flexibility, and ease of use, providing a practical means of facility simulation for research and development. The agent-based simulation technology is proven to be very successful in capturing the kind of decentralized, emergent behaviours of composite UAV formations. These agents act independently based on their internal rules and through dynamic interactions with the environment and other agents, which also make the setting similar to what is observed in the target system.
Using this in Simulink via Matlab system class makes a flexible and easy-to-use simulation framework for single and multiple vehicle simulations. Integration of dynamic models along with their own control system for both UAVs and RUAVs also prepares the simulator for path planning and advanced mission scenarios.
The simulator has a user-friendly interface with a Simulink mask editor for easy insertion of parameters and scenario setup, making the initial adoption of the simulator broadly accessible.
The ability to model a variety of formations, including fixed-wing UAVs, rotorcrafts, and ground vehicles, makes the simulator useful to simulate scenarios of disaster prevention, rapid response, and other life-saving applications. Additional work in the future will extend the simulator with complex environmental factors and across more domains.

7. Contact Author Email Address

For any further information, mail to: gennaro.raspaolo@unicampania.it

8. Copyright Statement

9. Acknowledgement

This work was supported by the research project - ID:P2022XER7W "CHEMSYS: Cooperative Heterogeneous Multi-drone SYStem for disaster prevention and first response" granted by the Italian Ministry of University and Research (MUR) within the PRIN 2022 PNRR program, funded by the European Union through the PNRR program.

References

[1] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying ad-hoc networks (fanets): A survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[2] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *nature*, vol. 521, no. 7553, pp. 460–466, 2015.

[3] E. D'Amato, V. A. Nardi, I. Notaro, and V. Scordamaglia, "A visibility graph approach for path planning and real-time collision avoidance on maritime unmanned systems," in *2021 International Workshop on Metrology for the Sea; Learning to Measure Sea Health Parameters (MetroSea)*, pp. 400–405, 2021.

[4] A. Ferraro and V. Scordamaglia, "A set-based approach for detecting faults of a remotely controlled robotic vehicle during a trajectory tracking maneuver," *Control Engineering Practice*, vol. 139, p. 105655, 2023.

[5] E. D'Amato, C. D. Capua, P. F. Filianoti, L. Gurnari, V. A. Nardi, I. Notaro, and V. Scordamaglia, "Ukf-based fault detection and isolation algorithm for imu sensors of unmanned underwater vehicles," in *2021 International Workshop on Metrology for the Sea; Learning to Measure Sea Health Parameters (MetroSea)*, pp. 371–376, 2021.

[6] M. Wooldridge, *An introduction to multiagent systems*. John wiley & sons, 2009.

[7] K. P. Sycara, "Multiagent systems," *AI magazine*, vol. 19, no. 2, pp. 79–79, 1998.

[8] Y. Mualla, A. Najjar, A. Daoud, S. Galland, C. Nicolle, A.-U.-H. Yasar, and E. Shakshuki, "Agent-based simulation of unmanned aerial vehicles in civilian applications: A systematic literature review and research directions," *Future Generation Computer Systems*, vol. 100, pp. 344–364, 2019.

[9] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The knowledge engineering review*, vol. 10, no. 2, pp. 115–152, 1995.

[10] V. A. Nardi, A. Ferraro, and V. Scordamaglia, "Feasible trajectory planning algorithm for a skid-steered tracked mobile robot subject to skid and slip phenomena," in *2018 23rd International Conference on Methods Models in Automation Robotics (MMAR)*, pp. 120–125, 2018.

[11] F. L. B. L. Stevens, *Aircraft Control and Simulation*. John wiley & sons, 1992.