Event-triggered Consensus Control Approach for Guaranteed Finite Time Convergence – *Progress Report*

Liam Dale[1], n9741283
Dr. Aaron McFadyen[2]

Abstract **– Motivated by the incipience of urban air mobility this report investigates event-triggered consensus of multi agent systems guaranteeing finite time convergence. This resolves how large systems should efficiently and effectively reach a shared goal state. The event-triggering concept is introduced to avoid unnecessary communications between agents. State-dependent and state-independent triggering protocols are tested. They are both augmented with open loop estimation techniques to maintain effectiveness in a moving system. The former is found to reduce transmissions and upholds performance more successfully. Sliding mode control is explored to guarantee finite time convergence. This has the additional benefit of robustness against disturbances. A design framework is introduced to ensure that this is compatible with general event-triggering strategies. Simulations verify efficacy, and demonstrate concepts through use of simplified real-world models and a sky-highway use case**

I. Project Introduction

A century of aerospace innovation is culminating to the development of urban air mobility (UAM). This constitutes fully automated vertical take-off and landing aircraft (VTOL) for intra-city transportation [1]. When compared to road transport this technology may alleviate congestion and decrease travel time while faring at par or better regarding fuel costs and carbon dioxide emissions [2]. This has piqued interest from many entities including Uber and Airbus, which are already rolling out services on a small scale [3, 4]. Due to its infancy there are still many open questions regarding regulations and technical implementations [5].

Successful UAM realisation will see an increase in vehicle density along major aerial routes and highways which will eclipse the capability of contemporary air traffic management (ATM) systems. Thus, a challenge is posed to provide technologies which autonomously maintain safety and efficiency for dense multi-agent systems (MASs) with networking capabilities [1]. This will require all agents coordinate to observe a collective behaviour.

This project will focus on the topic of consensus, which typically refers to the problem of reaching an agreement among a group despite their initial state trajectories [6]. This is an integral component of group decision making and motion coordination. A typical example is platooning on highways, where vehicles negotiate a collective speed and heading to maintain a safe separation [6]. This project report focusses on decentralised event-triggered consensus as this has practical implications relevant to UAM.

This report is structured as follows. Section II provides a background and literature review to further introduce the topic. Findings inform the research problem and design program, which are outlined within sections III and IV. Multiple event-triggering protocols are tested in V with simple linear dynamics. Section VI builds on this by introducing nonlinear control with gain-scheduling and sliding mode technique. Finally findings are discussed.

II. Background/Literature Review

Information sharing between MAS participants is essential for consensus such they may all agree on a common goal [6]. It may occur in a centralised manner, which assumes all agents communicate global knowledge over a fully connected network and solve the problem with a central node. This approach introduces a single point of failure and as communication topologies are rarely fully connected becomes increasingly impractical as the number of agents scales [7]. Alternatively there is a decentralised scheme, where information may only be shared between neighbours and nodes perform individual decision making. This method is pragmatic by virtues of robustness and scalability, however is more complex in structure and organisation [7].

Continuous communication between agents is unrealistic for practical applications, as it is an unnecessary drain on limited resources [8]. Alongside corresponding computation and actuator updates it consumes energy to drain batteries and curtail flight time. Network constraints such as packet loss, latency and throughput worsen with congestion, endangering performance and stability [8]. Drawbacks are exacerbated as the network size scales. As such, it is important to conservatively transmit information while preserving the overall system performance [9]. Event triggered control is a solution where transmissions are spread sporadically based on current system measurements and an event-triggering threshold. This on-demand strategy significantly reduces transmission frequency while upholding performance [9].

While this investigation is tailored towards UAM the applications extend to networked control systems where resource conservation is important. Subterranean environments are critical to modern infrastructure. Due to harsh physical conditions and unforeseeable dangers such as fire or collapse it is preferable to use autonomous vehicles for many related operations [10]. A challenge is the limited communication infrastructure. In [10] autonomous mobile radio nodes traverse a collapsed mine to form a daisy chain

Commented [LD1]: NASA https://www.nasa.gov/aeroresearch/programs/iasp/aam

Commented [LD2]: How work?

Commented [LD4]: V Underground mining

Commented [LD3]: For finite time convergence

[1] **Liam Dale** is with Science and Engineering Faculty (SEF), Queensland University of Technology (QUT), Australia. This report is in partial fulfilment of EGH400-1 unit assessment requirements and submitted on 06-September-2020. liam.dale@connect.qut.edu.au.
[2] **Dr. Aaron McFadyen** is also with Science and Engineering Faculty (SEF), Queensland University of Technology (QUT), Australia. He is the supervisor for this research project.

and create an ad-hoc communications network. This enables tele-operated robots to clear rubble. An event-triggered solution would crucially extend node lifetime and preserve network resources.

A. *Multi-Agent Consensus*

The network topology plays an integral role in the consensus of a multi-agent system. It determines the rate of convergence, negotiated result and whether consensus is possible [6]. The topology is not necessarily fixed, being affected by vehicle motion and communication dropouts. Suppose there are n agents where the communication topology is modelled by a directed graph $G_n \triangleq (V_n, E_n)$. $V_n = \{1, 2 \dots n\}$ is the set of vertices and $E_n \subseteq V_n \times V_n$ is the set of edges. This system must have negotiations arbitrated by a consensus algorithm.

The most prevalent continuous-time consensus algorithm [6] is given as

$$\dot{x}_i(t) = -\sum_{j=1}^{n} a_{ij}(t)\big[x_i(t) - x_j(t)\big], \qquad i = 1, \dots, n$$

where $a_{ij}(t)$ is an entry in the associated adjacency matrix. This is linked to the network graph G_n through the value of edge (j, i). Variables i and j respectively denote receiving and transmitting agents. State x is the information state, which relates to the consensus variable of interest. The corresponding matrix form is

$$\dot{x}_i(t) = -L_n(t)x(t)$$

where $L_n(t)$ corresponds to the G_n associated Laplacian matrix. Consensus is achieved when for each initial state $x_i(0)$ and for all $i, j = 1, \dots, n$ the difference of states $|x_i(t) - x_j(t)| \to 0$ as $t \to \infty$.

The continuous time algorithm is discretised with a difference equation [6]

$$x_i[k+1] = \sum_{j=1}^{n} d_{ij}[k]x_j[k], \qquad i = 1, \dots, n$$

where $d_{ij}[k]$ is an entry in the associated row-stochastic matrix D, and k denotes a sampling event. Information states are held constant between triggers. The corresponding matrix form is

$$x[k+1] = D[k]x[k]$$

Similarly, consensus is achieved when for each initial state $x_i[0]$ and for all $i, j = 1, \dots, n$ the difference of states $|x_i[k] - x_j[k]| \to 0$ as $k \to \infty$.

For both continuous and discrete time with invariant topologies and constant a_{ij} gains consensus is achieved when the directed topology has a directed spanning tree, or the undirected topology is connected. In these scenarios the Laplacian matrix eigenvalues $\lambda_i(L_p)$ are positive semidefinite, starting from zero and increasing. As such, non-zero eigenvalues $\lambda_i(-L_p)$ are all negative. The second smallest $\lambda_2(L_p)$ is the algebraic connectivity and defines the asymptotic convergence rate [6].

These algorithms force x_i to be driven towards the information state of its neighbours. Equilibrium is determined by vehicles which are the root of a directed spanning tree, equal to a weighted average of their initial states [6]. This property gives rise to leaderless, and leader-follower strategies. When there is a single leader it becomes a reference for all followers, which is often the case in platooning. For leaderless networks if a reference is desired an intangible virtual leader may be created to emulate this behaviour [6]. For undirected connected graphs all nodes are leaders and followers giving average consensus [6].

With controllers being connected over a network, the consensus should be robust to wireless channel constraints and stochasticity [7]. Issues such as fading, variable latency, packet misses and quantisation cause the system and controller inputs to be an approximation of measured values. These issues have the potential to degrade performance and stability [7], and alongside network topology changes are not addressed in the general algorithm mentioned.

B. *Consensus for Single-Integrator Dynamics*

Consider agents where the information state dynamics are given as the control input [6].

$$\dot{\xi}(t) = u_i(t), \qquad i = 1, \dots, n$$

The fundamental consensus algorithms are then altered as follows

$$\mathrm{u_i(t)} = -\sum_{j=1}^{n} a_{ij}(t)\big[\xi_i(t) - \xi_j(t)\big], \qquad i = 1, \dots, n$$

$$\xi_i[k+1] = \sum_{j=1}^{n} d_{ij}[k]\xi_j[k], \qquad i = 1, \dots, n$$

and in matrix forms

$$\dot{\xi} = -|L_n(t)| \otimes |I_m|\xi$$
$$\xi[k+1] = -|D_n[k] \otimes I_m|\xi[k]$$

where $\otimes$ denotes the Kronecker product. The concepts for stability and convergence are analogous to before.

This algorithm may be extended to guarantee that information states converge to a set difference Δ_{ij}, such that $(\xi_i - \xi_j) \to \Delta_{ij}(t)$ as $t \to \infty$ [6]. The revised controller is given as follows.

$$\mathrm{u_i(t)} = \dot{\delta}_\iota - \sum_{j=1}^{n} a_{ij}\big[(\xi_i - \xi_j) - (\delta_i - \delta_j)\big], \qquad i = 1, \dots, n$$

Here $\Delta_{ij} \triangleq \delta_i - \delta_j, \forall i \neq j$ where δ_i and δ_j describes the relative state reference of corresponding agents. This algorithm is particularly useful for maintaining relative vehicle positions, which has applications in formation control.

C. *Event-Triggered Consensus Protocols*

For resource conservation communications and corresponding actuator updates may be spread over discrete time intervals, between which measured values are locked and held. There must be a sufficient frequency to effectively capture state changes and preserve performance goals. Conversely there must be a guaranteed lower bound on inter-event execution times to exclude Zeno behaviour, where infinite events happen on a finite time horizon [11].

The sampled-data approach is where transmissions are delayed by a fixed period, although improving utilisation this still results in an over-provisioning of resources. High frequency triggers required to maintain stability and provide new information in the transient phase are excessive as the

Commented [LD10]: Often this is the leader in a platoon, but may not be the case for UAM/UTM

Special case with single root node

Virtual leaders – pg. 56

Commented [LD5]: A general strategy for developing….

Commented [LD11]: How can we translate/extrapolate out to an aerial network, underground mining network etc.

Always good to add some context. For example, at the end of the paragraph you might: For example, for connected vehicles in an underground mine or fixed aerial network, this means/could denote etc.

Commented [LD8]: Link to G and L matrices

Commented [LD14]: 2.4 NOTES
Discrete time references

Commented [LD15]: ~~Information states which imply fixed relative displacements are suitable for steady-state separation; a candidate could be momentum, but not position. If separation guidelines are violated there are no mechanisms to widen the gap.~~

Comment: elaborate some more here

Commented [LD16]: Define delta_i to be reference state of ith agent etc

Commented [LD17]: Other ways

Commented [LD18]: True - we have stability concerns that may be impacted by sampling time, form of hold (ZOH, 1st order etc.) depending on the dynamics and prevalence of T in the state equations

Commented [LD9]: How though?

system converges [11]. A dynamic approach is required to overcome this limitation.

An event-triggered protocol transmits information on demand to reduce wastage without jeopardising performance. A core component is the error measurement e between the current states x and latest broadcast states $\hat{x}$

$$\hat{x}_i(t) = x_i(t_k^i), \qquad t \in [t_k^i, t_{k+1}^i)$$
$$e(t) = \hat{x}(t) - x(t)$$

where $t_0^i, t_1^i, \ldots$ are the triggering instants for agent i. This provides a measure on how valuable the state of an agent is to maintaining overall closed loop behaviour. Events are triggered when this crosses a dynamic threshold e_T and so at all times the following condition holds.

$$error \leq threshold \triangleq e(t) \leq e_T(t)$$

On triggering agents broadcast their states and alongside receivers update their control inputs accordingly. As controllers use periodically broadcast values the input is redefined as

$$\mathrm{u_i}(t) = -\mathrm{K}\sum_{j=1}^{n} a_{ij}(t)\left[\hat{x}_i(t) - \hat{x}_j(t)\right], \qquad i = 1, \ldots, n$$

where K is a gain matrix and again agent i receives from agent j. A basic operating procedure is represented in Figure 1 below. The error threshold is selected to preserve underlying consensus stability concepts such as Lyapunov stability, input-to-state stability (ISS) and L_2 stability. The latter two are preferred in practical systems due to their robustness against external disturbances.

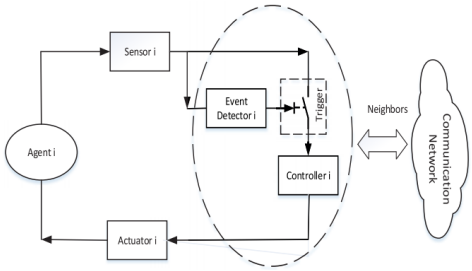


Figure 1 Event Triggering Process [11]

An early event triggered consensus strategy for a linear multi-agent system of N agents is considered in [12] for undirected, connected topologies. The solution is extensible to high order dynamics, described via

$$\dot{x}_i(t) = Ax_i(t) + Bu_i(t), \qquad i = 1,2, \ldots N$$

assuming

$$rank(AB) = rank(A)$$

It is also robust to switching topologies, and enforces a lower boundary on the inter-execution time to exclude Zeno behaviour for all agents. The event triggering-threshold is given as

$$e_T(t) = k_i|z_i(t)|, i = 1,2, \ldots N$$

which has the following elements.

$$0 < k_i < 1/\max\left(\lambda_N[L_p(t)]\right)$$

$$z_i(t) = \sum_{j=1}^{N} a_i(x_i(t) - x_j(t))$$

This relaxes the actuation update requirements, however $z_i(t)$ still requires constant knowledge of neighbouring states and global topology – contradicting motivations and leaving continuous communication as a requirement. Communication is relaxed in [13] to use only broadcasted states at the expense of a more complex triggering function. Assumptions of linearity and an undirected topology simplify event triggering, however, raise practical concerns.

An alternative strategy using a state-independent threshold for relative error is proposed in [14] for an undirected, connected topology. The triggering threshold is given as

$$e_T(t) = c_0 + c_1 e^{-\alpha t}$$

with constants

$$c_0 \geq 0, c_1 \geq 0, c_0 + c_1 > 0$$
$$0 < \alpha < \lambda_2(L_p)$$

By only needing local information to compute the state-triggering condition the need for continuous communication is eliminated. Design choices are required for c_0, c_1 and α. If the choice for $c_0 \neq 0$ then consensus is not exact but bounded by a radius but Zeno behaviour is automatically excluded. Details are also given on extensions to accommodate for network stochasticity and double-integrator agents. The assumptions of linear dynamics and network topologies are again impractical.

These strategies will continue to trigger when consensus is reached unless agents are at an equilibrium point. The error thresholds trend towards zero as either the agents converge or time elapses. When they are arbitrarily small if any agent deviates from its latest broadcast value the error is enough to trigger an event [15]. These triggers are not crucial to maintaining consensus, and so are a contradictory wastage of resources. In many autonomous vehicle scenarios this is a hard limitation as agents are constantly moving.

D. Model-Based Event-Triggered Consensus Protocols

Model-based event-triggered control protocols employ observer estimation techniques to the predict error and reduce triggering frequency [16]. There is a computational cost as agents must be equipped with estimators for themselves and each of their leaders. If the network is not homogenous agents must also communicate their dynamics. As the network scales these burdens grow [15]. There are open loop and closed loop estimation techniques.

In the open loop estimation approach the latest broadcasts are projected forwards without control inputs. They may be rewritten as

$$\hat{x}_i(t) = e^{A\left(t - t_k^i\right)} x_i(t_k^i), \qquad t \in [t_k^i, t_{k+1}^i)$$

where A is a linear state dynamics matrix for agent i. This redefines the control input and error [16]

$$\mathrm{u_i}(t) = -\mathrm{K}\sum_{j=1}^{n} a_{ij}(t)\left[e^{A_i\left(t - t_k^i\right)} x_i(t_k^i) - e^{A_j\left(t - t_k^j\right)} x_j(t_{k'}^j)\right]$$

$$e_i(t) = e^{A\left(t - t_k^i\right)} x(t_k^i) - x(t)$$

Commented [LD22]: own event and neighbour event

Commented [LD19]: OLD : An event-triggered protocol transmits information on demand to reduce wastage without jeopardising performance. This is characterised by an event-triggering condition which compares the error between the current and most recently transmitted information measurement $\hat{x}$ (state or output) to a set threshold e_T [30]. $error \geq threshold \triangleq |x(t) - \hat{x}(t)| \geq e_T$ Here t_j describes the j^{th} consecutive sampling instant, $t \in [t_j, t_{j+1})$ for $j = 0,1, \ldots, \infty$. This provides a measure on how valuable the state of an agent is to maintaining overall closed loop behaviour and adapts tasks as required. A basic operating procedure is represented in Figure 1 below. The error threshold is selected to preserve underlying consensus stability concepts such as Lyapunov stability, input-to-state stability (ISS) and L_2 stability. The latter two are preferred in practical systems due to their robustness against external disturbances.

Commented [LD23]: other limitations in overview

Commented [LD24]: For single integrator agents the trigger protocol is given.

Commented [LD21]: protocol

where t_k^j, denotes a triggering instant for agent j. State dependent and independent protocols utilising this technique are studied in [16] and [17] respectively.

A discretised version of this technique is presented in [17]. The estimation is given recursively from the latest broadcast as

$$\hat{x}_i[k] = G^{(k-k_q^i)}x_i[k_q^i], \qquad k \in [k_q^i, k_{q+1}^i)$$

where G is the discrete state space dynamics matrix and k_q^i denotes triggering iterations of agent i. The event threshold is taken as

$$e_T(k) = c\alpha^{kT}$$

with T as the discrete time step and the following constants.

$$c > 0$$
$$\max(|\lambda(\Xi)|) < \alpha < 1$$

For N agent states and n network agents matrix $\Xi \in \mathbb{C}^{(n-1)N\times(n-1)N}$ is given as

$$\Xi \triangleq I_{n-1} \otimes G + \Delta \otimes HK$$

where H is the discrete time input matrix and I is an identity matrix. Finally, $\Delta \in \mathbb{C}^{(n-1)\times(n-1)}$ is the submatrix of the Jordan canonical form of $I_n - D_{in}$ without row and column one. D_{in} is the topology in-degree matrix (see Appendix I matrix preliminaries). Compared to the previously surveyed state-dependent protocol this threshold requires more calculation. An advantage is that agent dynamics are considered. Discrete sampling inherently provides a lower boundary on inter-event times to exclude Zeno behaviour and so there is no bounding constant needed. Perfect consensus is enforced.

The closed loop approach makes estimates with a control input. It follows that agents must additionally broadcast their inputs on event triggers. This requires drastic alteration of the generic strategy; if $\hat{x}$ reflects the measured states there is no error to trigger. The strategy is considered in [18] where agents are triggered by virtual edge systems which consider the difference in states and inputs of their two vertices. Advantages are only seen in the transient phase. When consensus is reached control inputs are zero, and this technique becomes analogous to the open loop estimation. Input broadcasting merits further review and consideration. If agents can estimate the states of their neighbours the continuous monitoring requirements of protocols surveyed in the previous subsection are relaxed.

E. Finite Time Convergence

The convergence rate is a significant performance metric for consensus control, so a typical problem is to develop controllers which drive the system in as little time as possible [6]. Most control schemes use Lipschitzian dynamics, leading to exponential convergence with infinite settling time [19] and dependence on initial state conditions [11]. With an event-triggering protocol which aims to reduce control updates, this may suffer further [11]. This motivates practical MAS design to reach consensus in finite time. Research in this area is attracting attention and evolving rapidly, with recent approaches in [20, 21, 22, 23].

An early event-triggered finite time consensus control scheme is proposed in [24] for a mobile sensor network. This designs a finite-time consensus algorithm and determines a state-dependent event-triggering rule to preserve stability. It models a linear agent i with state derivatives

$$\dot{x}_i(t) = v_i(t), \qquad \dot{v}_i(t) = u_i(t)$$

where x_i and v_i respectively denote position and velocity. The controller is given as

$$u_i = sig\left(\sum_{j=0}^{n} a_{ij}\left(\beta(x_j - x_i) + \gamma(v_j - v_i)\right)\right)^{\alpha} + \sum_{j=0}^{n} a_{ij}(v_j - v_i)$$

where $0 < \alpha < 1$, $\beta > 0$, and $\gamma > 0$. These are each used to tune the rate of convergence. The function $sig(r)^{\alpha} = [sign(r_1)|r_1|^{\alpha} \ \dots \ sign(r_m)|r_m|^{\alpha}]^T$. The triggering condition has increased complexity, and still requires continuous communication. Results show that compared to a standard benchmark performance is improved while reducing computational resource usage and control updates.

III. Research Problem

The aim of this project is to investigate a practical approach for a MAS to achieve consensus. As a foundational concept in unmanned traffic management (UTM) this is quintessential for the successful realisation of future UAM.

This research will simulate average consensus amongst homogeneous agents under an undirected network. Communication will be event triggered and guarantee convergence within finite time to achieve a balance of feasibility and performance. This will lay the foundation to explore disturbance robustness.

Even minor disturbances can greatly affect the event-triggering performance and consensus. When the triggering threshold is low they can cause unnecessary events and undercut design goals. Alternatively with bounded consensus and sporadic triggers the system precision is reduced.

Network artefacts such as transmission latency, dropouts and quantisation are also relevant to large MASs. They will not be considered within this paper however as their effects are abated by avoiding continuous communication. This reiterates the importance of minimising unnecessary disturbance related events.

IV. Design Program

Working towards a practical implementation requires that a realistic scenario be modelled. This section details some practical considerations towards the final outcome.

A. Vehicle Dynamics

Most physical systems are governed by nonlinear dynamics. Two such vehicle models are defined within this subsection. They are similar, although have different physical states and levels of actuation.

These models may be linearized to enable to the use of simple linear control techniques. This requires calculation of the Jacobian state and input matrices (see Appendix I.C). These are denoted respectively as $A(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$ and $B(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$, where $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{u}}$ are state and input vectors which linearization has occurred around. These may be calculated at different operating points, provided that the system is measured or completely observable.

Commented [LD28]: Sliding mode shit

An event-triggered finite time consensus control scheme is proposed and experimentally tested in [] to show promising performance. This designs a novel event-triggered finite-time integral sliding mode controller for formation consensus.

Commented [LD27]: An input-based triggering approach to leader-following problems

Distributed model based event-triggered control for synchronization of multi-agent systems

$\hat{\boldsymbol{x}}_i(t) = \boldsymbol{e}^{A(t-t_k^i)}\boldsymbol{x}_i(t_k^i) + \int_{t_k^i}^{t} \boldsymbol{e}^{A(t-\tau)}Bu_i(\tau)\,d\tau, \ t \in [t_k^i, t_{k+1}^i)$

Commented [LD29]: This will lay the foundation to explore a pressing design challenge which may come under one of the succeeding threads.

a)Robustness to disturbances
b)Robustness to network stochasticity
c)Relaxing network assumptions
d)Robustness to hardware limitations

The latter component is intentionally left open due to the rapid evolution of research in this realm.

The nonlinear model may be approximated in the state space form as follows.

$$\dot{\boldsymbol{x}} = A(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})\boldsymbol{x} + B(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})\boldsymbol{u}$$

The approximation will worsen as the actual points diverge from $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{u}}$. An inherent limitation of the representation accuracy is the discrete time sampling interval.

1. Quadrotor

The idealised rigid-body dynamics of a quadrotor vehicle [25] are a candidate to contextualise this project within UAM. There are three degrees of freedom with cartesian x-y motion and rotation about a single axis. These are controlled by front and rear actuator forces F_F and F_R. This can be seen below in Figure 2. States considered are position x, momentum p_x, position y, momentum p_y, pitch angle θ, and finally angular inertia L. These are as follows.

$$\boldsymbol{z_1} \triangleq [x \quad p_x \quad y \quad p_y \quad \theta \quad L]^T$$
$$\boldsymbol{F_1} \triangleq [F_F \quad F_R]^T$$

Which are described with the following dynamical model.

$$\dot{\boldsymbol{z}}_1 = \begin{bmatrix} \dot{x} \\ \dot{p}_x \\ \dot{y} \\ \dot{p}_y \\ \dot{\theta} \\ \dot{L} \end{bmatrix} = \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{z}_4 \\ \dot{z}_5 \\ \dot{z}_6 \end{bmatrix} = \begin{bmatrix} m^{-1} z_2 \\ -(F_F + F_R)\sin(z_5) \\ m^{-1} z_4 \\ (F_F + F_R)\cos(z_5) - mg \\ J^{-1} z_6 \\ l_F F_F - l_R F_R - b_t J^{-1} z_6 \end{bmatrix}$$

Table 1. Quadrotor Variable Descriptions

Variable	Description
m	Vehicle mass
g	Gravity constant
J	Moment of inertia around centre mass
b_t	Aero-dynamic damping
F_F, F_R	Thrust force of front rotors, rear rotors
l_F, l_R	Distance from centre mass of front rotor, rear rotor thrust vectors

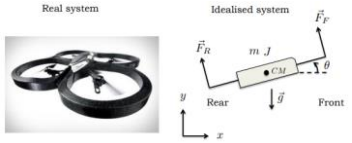


Figure 2 Quadrotor Model

The Jacobian matrices for this model are shown below.

$$A_1(\boldsymbol{z}, \boldsymbol{F}) = \begin{bmatrix} 0 & m^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \partial f_2/\partial z_5 & 0 \\ 0 & 0 & 0 & m^{-1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \partial f_4/\partial z_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & J^{-1} \\ 0 & 0 & 0 & 0 & 0 & -b_t J^{-1} \end{bmatrix}$$

$$\frac{\partial f_2}{\partial z_5} = -(F_F + F_R)\cos(z_5)$$

$$\frac{\partial f_4}{\partial z_5} = -(F_F + F_R)\sin(z_5)$$

$$B_1(\boldsymbol{z}, \boldsymbol{F}) = \begin{bmatrix} 0 & 0 \\ -\sin(z_5) & -\sin(z_5) \\ 0 & 0 \\ \cos(z_5) & \cos(z_5) \\ 0 & 0 \\ l_F & -l_R \end{bmatrix}$$

$$C_1 = \boldsymbol{I_6}, \qquad D_1 = \boldsymbol{0^{6\times 2}}$$

2. Hovercraft

A hovercraft model may be used for this application [26]. It is very similar to the surveyed quadrotor however instead of position and momentum, it considers momentum and acceleration. The system is also more actuated, having starboard, port and lift actuators F_S, F_P and F_L. States are momentum p_x, acceleration $\dot{p}_x$, momentum p_y, acceleration $\dot{p}_y$, pitch angle θ and angular velocity $\dot{\theta}$. These are represented as follows.

$$\boldsymbol{z_2} \triangleq \begin{bmatrix} p_x & \dot{p}_x & p_y & \dot{p}_y & \theta & \dot{\theta} \end{bmatrix}^T$$
$$\boldsymbol{F_2} \triangleq [F_S \quad F_P \quad F_L]^T$$

The following details state derivatives. The model is pictured in Figure 3.

$$\dot{\boldsymbol{z}}_2 = \begin{bmatrix} \dot{p}_x \\ \ddot{p}_x \\ \dot{p}_y \\ \ddot{p}_y \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{z}_4 \\ \dot{z}_5 \\ \dot{z}_6 \end{bmatrix} = \begin{bmatrix} z_2 \\ (F_s + F_P)\cos(z_5) - F_L \sin(z_5) - z_2 \\ z_4 \\ (F_s + F_P)\sin(z_5) + F_L \cos(z_5) - z_4 \\ z_6 \\ F_s - F_p - z_6 \end{bmatrix}$$

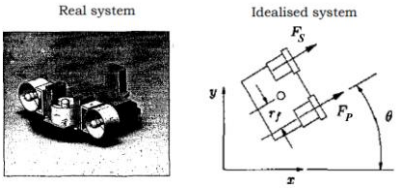


Figure 3 Hovercraft Model

For the purpose of linearisation, the Jacobian matrices are as follows.

$$A_2(\boldsymbol{z}, \boldsymbol{F}) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & \partial f_2/\partial z_5 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & \partial f_4/\partial z_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

$$\frac{\partial f_2}{\partial z_5} = -(F_s + F_p)\sin(z_5) - F_l \cos(z_5)$$

$$\frac{\partial f_4}{\partial z_5} = (F_s + F_p)\cos(z_5) - F_l \sin(z_5)$$

$$B_2(\boldsymbol{z}, \boldsymbol{F}) = \begin{bmatrix} 0 & 0 & 0 \\ \cos(z_5) & \cos(z_5) & -\sin(z_5) \\ 0 & 0 & 0 \\ \sin(z_5) & \sin(z_5) & \cos(z_5) \\ 0 & 0 & 0 \\ 1 & -1 & 0 \end{bmatrix}$$

$$C_2 = \boldsymbol{I_6}, \qquad D_2 = \boldsymbol{0^{6\times 3}}$$

Commented [LD32]: The idealised rigid-body dynamics of a quadrotor vehicle [26] are a candidate to contextualise this project within UAM. There are three degrees of freedom with cartesian x-y motion and rotation about a single axis. This can be seen below in Figure 2. States considered are x position, x momentum, y position, y momentum, pitch angle, and angular inertia. These are as follows.

Commented [LD33]: check consistency w/ state symbols and ensure each are defined

B. Use Case

Future UAM may be realised with a free flight style which is not governed by an organisational system [27]. Vehicles may be freely launched and compete for available airspace. They travel the shortest path to their destination and when anticipating collision employ sense and avoid technologies [27]. This will require minimal infrastructure to implement however presents significant safety challenges.

The air highway method of air traffic control provides a safer and more orderly alternative to free flight [27]. It emulates land freeways by geofencing air traffic paths for UAVs to follow. These are splits into directed lanes for travel, passing, emergency and damaged vehicles. Individual lanes will operate under different conditions, such as speed and minimum separation. Paths are carefully planned to preclude agents from crossing. With designated paths the route from source to destination is less efficient than free flight [27]. This idea has not been thoroughly tested or studied, so an optimal design is not known.

The project considers the air highway method of traffic control due to its safety merits. It models a sky highway separated into five lanes without speed limits. Using the cartesian XY plane lanes span the horizontal direction and are stacked vertically. Agents in each lane communicate with their two adjacent neighbours. They aim to reach a consensus on horizontal velocity while specifying a dynamic formation with minimum offsets to avoid loss of separation. New agents are periodically spawned around the entry point with random initial conditions. They merge into a designated lane. This causes the network topology to switch. Lane boundaries are predefined so agents do not need to use consensus techniques to negotiate the vertical position or velocity.

Simulations with different concepts from this report have been conducted for demonstrative purposes. Animations are available for viewing at https://github.com/ldale1/EGH400-Event-Triggered-Consensus/tree/master/Animations.

C. Disturbance Rejection

Aerial vehicles are susceptible to the effects of wind and other environmental conditions. This is especially pertinent for small UAVs, which may be blow off course and risk collision [27]. A wind model will be applied to agents to verify their robustness against exogenous disturbances. This will take the form of the U.S. Naval Research Laboratory Horizontal Wind Model, which is available through MATLAB [28]. It has meridional and zonal components. The chosen simulation location is Brisbane city.

The application of the wind model will be simplified. Causing acceleration it will singularly affect the velocity or momentum rate of change. Changes to pitch will not be considered despite the practical implications. Applied force F_W is calculated proportionally to the relative system speed as follows [29].

$$F_W = C(v_{SYS} - v_w)^2$$

Here v_{SYS} and v_w respectively denote the agent and wind velocities. C is an aggregation of constants such as the drag coefficient, effective aperture and fluid density.

When the same disturbance is applied to all agents it goes unnoticed within the consensus algorithm. Being affected an equal amount no inter-agent error appears for the controller to correct. This occurs for the wind model when consensus is achieved. This causes the arbitrated consensus to shift. One compensation strategy is to incorporate a disturbance observer and adjust the target appropriately. Another is to incorporate a virtual leader, which is unaffected by disturbance. This removes average consensus and instead enforces it around a setpoint or tracking reference.

D. Operating Regime

For each agent their operations are simulated with the following algorithm in discrete time. Samples t_k are synchronised for all agents, however this may not be the case in practice.

Algorithm 1. Network Simulation

```
01:  t_k ← 0
02:  x̂ ← x_0
03:  Broadcast x̂
04:  While t_k < t_sim
       Check for transmissions
05:    If (receiving transmission from any leader j)
06:      x̂_j ← x_j
07:      Recalculate control input u
08:    End if
       Check for an event
09:    If (‖e‖ >‖e_T‖)
10:      x̂ ← x
11:      Broadcast x̂
12:      Recalculate control input u
13:    End if
       Decide on control input
14:    If (recalculated control input)
15:      Update actuator
16:    Else
17:      Hold actuator
18:    End if
       Ready for the next sample
19:    Step agent
20:    If (model-based protocol)
21:      Project states x̂
22:      Project leader states x̂_j
23:    End if
24:    t_k ← t_{k+1}
25:  End while
```

This models a time-invariant topology. When this switches the new topology is modelled as a contiguous scenario. The algorithm restarts with the initial states as the final states of the previous topology.

Commented [LD34]: A preliminary simulation has been performed. Dynamics are those as discussed in section V, however these are extended with another actuator and independent dimension. A model-based state-dependent triggering strategy also discussed in V is used. Updates to actuators are coupled, however with the simplified dynamics they could be separated. Vertical position and velocity are not decided via consensus and so should be triggered independently of network status.

V. Consensus Algorithm Exploration

Consider a strongly connected network with five agents. They have simple linear dynamics described as follows.

$$\dot{x}_i = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x_i + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_i$$

where x_i and u_i denote states and control input for agent i. The Laplacian communications matrix and associated eigenvalues λ_L are as follows.

$$L = \begin{bmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & -1 & -1 & -1 & 4 \end{bmatrix}$$

$$\lambda_L = [0, 5, 5, 5, 5]$$

The system is modelled in discrete time, with a sampling period of 0.01 seconds. The corresponding discrete time dynamics are given as follows.

$$x_i[k+1] = \begin{bmatrix} 1 & 0.01 \\ 0 & 1 \end{bmatrix} x_i[k] + \begin{bmatrix} 0.0001 \\ 0.01 \end{bmatrix} u_i[k]$$

The row-stochastic communications matrix and associated eigenvalues λ_D are given below.

$$D = \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix}$$

$$\lambda_D = [0, 0, 0, 0, 1]$$

Initial state values are given as $x_1(0) = [5.5; 9.5]$, $x_2(0) = [-4.5; -6.5]$, $x_3(0) = [12.5; -0.5]$, $x_4(0) = [9.5; -1.5]$ and $x_5(0) = [-0.5; 2.5]$. Their controller gains are $K = [0.99, 1.72]$ as decided by a discrete linear quadratic regulator which penalises states and inputs equally.

A. State-Dependent Event Trigger

Using the state-dependent event trigger in [12] adapted for discrete time agent trajectories are pictured as follows. Consensus is successfully reached. The settling value for state two is 0.7, which is the average of initial values.

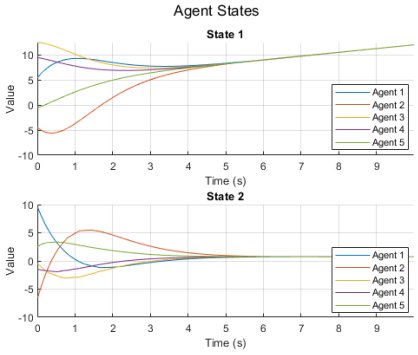


Figure 4 Scenario V.A State Trajectories

During the transient phase triggering instants observe intended behaviour. Agents one and two have the largest initial velocities. Their rapid movement causes frequent early triggers. As they adjust their speed this frequency reduces. When agents near consensus their triggering frequency increases rapidly, contradicting design goals. On average 40.7% of samples are triggered. Actuator updates are more common as they also relate to neighbour transmissions. This ratio will worsen as time draws out due to extreme steady state triggers.

Commented [LD35]: Actuator udpates

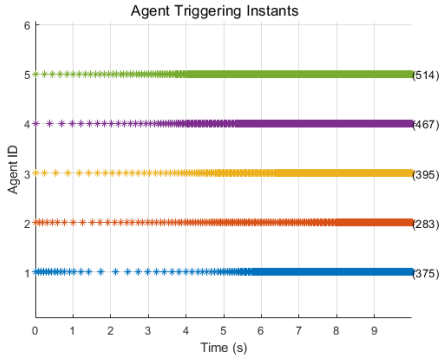


Figure 5 Scenario V.A Triggering Instants

Rapid triggering as consensus is approached is mirrored by an arbitrarily small error threshold. As state two settles on a nonzero value state one will always be drifting from its latest broadcast. Without a predictive factor the error grows and consistently triggers events. State one cannot be overlooked in triggering as it could be used to dictate the MAS formation. Magnitude of state two corresponds to steady state triggering frequency.

Commented [LD36]: Trigger error on both states? Or does velocity work?

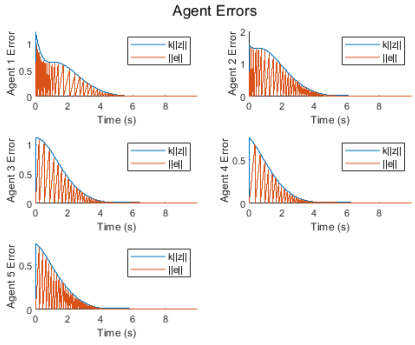


Figure 6 Scenario V.A Error and Threshold Norms

B. Model-Based State Dependent Event Trigger

This scenario uses the same event trigger, however predicts error with an open loop estimation approach. State trajectories are pictured below. Transient differences indiscernible and the agents settle at the same states.

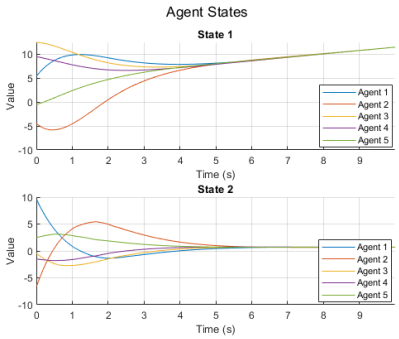


Figure 7 Scenario V.B State Trajectories

With the predictive factor agents stops triggering when consensus is reached. Net triggering has been reduced to 4.9% of all samples to better align with the intent. This ratio will improve as time draws out and the network is undisturbed.

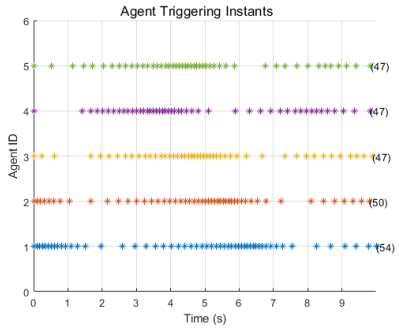


Figure 8 Scenario V.B Triggering Instants

C. Model-Based State Dependent Event Trigger under Switching Topologies

This scenario models a network which switches to a randomly generated, undirected topology every second. No restrictions are placed on their layout. The progression is shown in the succeeding figure.

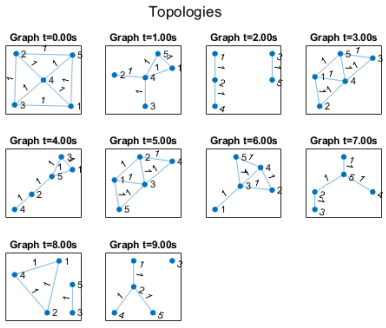


Figure 9 Scenario V.C Topology Progression

Agents and protocol are the same as in scenario V.B. Simulated trajectories are shown in the figure below. These have become jagged, as agent constantly switch leaders. When completely disconnected agents have zero control input and state two stays fixed. Consensus is still achieved; with enough random changes agents are inevitably included and transmit their states across the network. As switching frequency increases the result becomes more analogous to the previous scenario. This is due to the averaging of randomness. Settling has been drawn out, and the final state two value has dropped 17.1% to a value of 0.58.

Commented [LD37]: This could use some more

Commented [LD38]: spgr, then not and

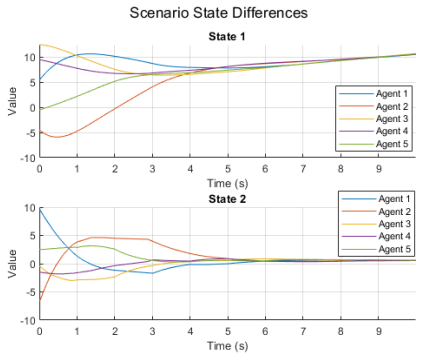


Figure 10 Scenario V.C State Trajectories

Topology changes trigger an event. Whether events keep triggering depends on state derivatives and updated error thresholds. When thresholds quickly drop agents adjust rapidly with frequent triggers.

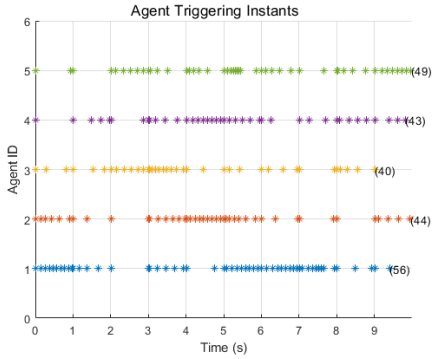


Figure 11 Scenario V.C Triggering Instants

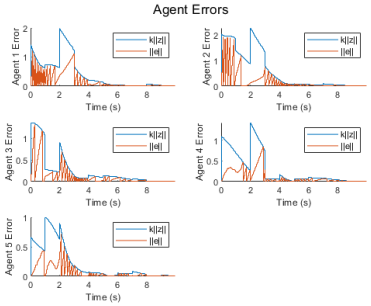


Figure 12 V.C Scenario Error and Threshold Norms

When there is a topology switch agent control inputs can change quite drastically. This is especially relevant when an agent suddenly becomes led by a single neighbour with different states. A mild example is seen in the succeeding figure at two seconds. A practical worry may be actuator switching speed and saturation.

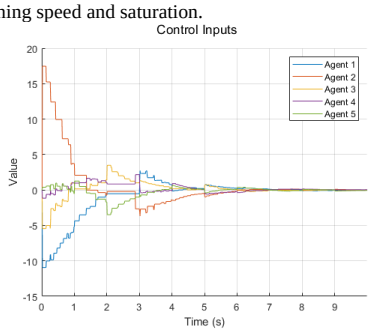


Figure 13 Scenario V.C Control Inputs

D. Model-Based State-Independent Event-Trigger

This scenario implements the model-based state-independent protocol surveyed in II.D with an invariant strongly connected topology. Constant c is chosen as 0.5 arbitrarily. The eigenvalues of matrix Ξ are all $0.9913 \pm 0.05i$ giving constant α as 0.992. State trajectories are similar to previous scenarios.

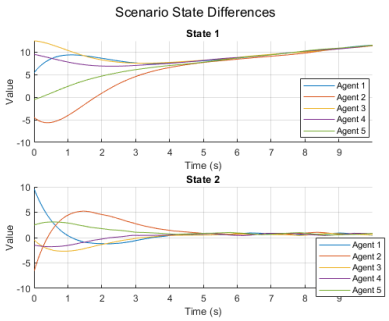


Figure 14 Scenario V.D State Trajectories

Event triggers are very infrequent as shown in the succeeding figure. As a consequence agents are not pushed to a precise consensus speedily; there is still a spread at ten seconds. This is reflected by the error threshold, which slowly decays uninfluenced by neighbours. Empirically the rate for perfect convergence is an inherent challenge for state-independent thresholds. While this may be affected by design choices if the exponential term decays too quickly the threshold goes to zero and appears as a fixed trigger. A similar effect is seen when lowering the static value of c. Given topology switches the threshold does not adapt.

Commented [LD39]: alter c does this behave better

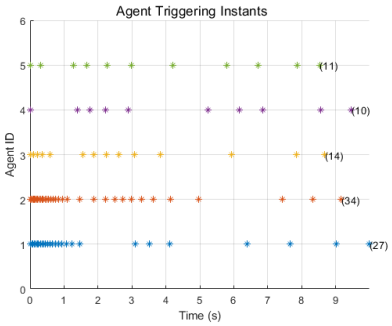


Figure 15 Scenario V.D Triggering Instants

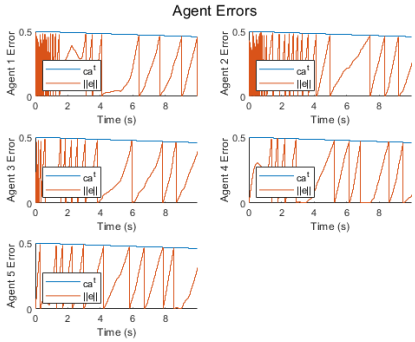


Figure 16 Scenario V.D Error and Threshold Norms

E. Algorithms Summary

This initial exploration has demonstrated that the state-dependent threshold more effectively guides the system toward consensus. A key metric for this is suggested as the error norm for networks, $\|E_{net}(k)\|$. This is given as

$$\|E_{net}(k)\| = \sqrt{\sum_{i=1}^{N} \|E_i(k)\|^2}$$

where N is the number of agents. The value $\|E_i(k)\|$ is the error norm of agent i

$$\|E_i(k)\| = \sqrt{\sum_{i=1}^{n} \|e_i(k)\|^2}$$

where n is the number of states. This is plotted in the succeeding figure where normalisation has occurred. High error does not correlate to poor outcomes. Given that performance and stability are upheld it instead indicates that event-triggers have only occurred as required. Before settling

the two state-dependent strategies tolerate high error. During this period the consensus deviation for all strategies (including fixed triggering) is comparable. This is shown in Figure 18. Tolerance drops once consensus is neared. The model-based variant compensates for this with its predictive approach. The state-independent strategy tolerates error to an upper limit based on its design parameters. Without incorporating neighbouring state information it cannot see the same success. Error stays high as exact consensus is not reached.

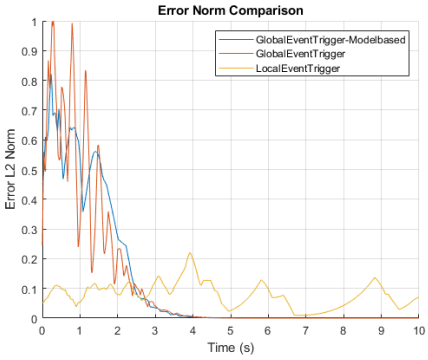


Figure 17 Network Error Norm

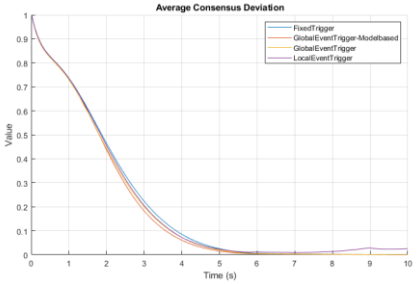


Figure 18 Consensus Standard Deviation

VI. Non-Linear Control

Most physical systems are governed by nonlinear dynamics, and so for a practical UAM implementation nonlinear control techniques must be studied. This section investigates the efficacy of state feedback, gain scheduling and sliding mode controllers using dynamical models from section IV.A. The model-based state dependent triggering rule is employed. The following two scenarios are used as demonstrations.

Scenario One

The quadrotor model is simulated with this scenario. There are four agents with the following initial conditions.

$$x_{01} = [1.95 \quad 0 \quad 3.36 \quad 0 \quad 0.73 \quad 0]^T$$
$$x_{02} = [-1.14 \quad 0 \quad 2.34 \quad 0 \quad 0.86 \quad 0]^T$$
$$x_{03} = [2.84 \quad 0 \quad -3.91 \quad 0 \quad 0.82 \quad 0]^T$$
$$x_{04} = [-4.50 \quad 0 \quad 3.34 \quad 0 \quad 0.93 \quad 0]^T$$

These are placed in a network under the following topology.

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

The sampling time is $5 \cdot 10^{-3}$ seconds. The control objective is to reach consensus on all states. Angular momentum L is virtually lead by a value of $0\ [kg \cdot m^2/s]$ to ensure that consensus pitch θ is fixed. This prevents steady-state rotation.

Scenario Two

This scenario instead considers the hovercraft model. The network contains three strongly connected agents, sampled at $5 \cdot 10^{-3}$ seconds. Initial conditions are given as follows.

$$x_{01} = [-3.71 \quad 0.60 \quad 0.54 \quad 2.31 \quad 0.85 \quad 0.01]^T$$
$$x_{02} = [-1.92 \quad 2.31 \quad 0.32 \quad -2.35 \quad 0.84 \quad -0.22]^T$$
$$x_{03} = [3.12 \quad -0.85 \quad -2.16 \quad -1.93 \quad -0.74 \quad 0.08]^T$$

This network is disturbed by the wind model discussed in section IV.C. This is applied to states p_x and p_y. To combat the consensus goal being pushed to the wind rate a virtual leader is introduced. This transmits these respective states with values of -2 and $2\ [kg \cdot m/s]$. This also aims to prevent steady state rotation by transmitting $\dot{\theta}$ as $0\ [s^{-1}]$.

A. State Feedback

State feedback controllers are prolific. They will be designed within this section to establish a performance baseline. With this the control law is given as

$$u(k) = -Kx(k)$$

where K is a gain matrix chosen to reposition system poles. Linear quadratic regulators will used for this purpose in succeeding examples. These are designed for linear systems, and so the nonlinear vehicle models must be linearised around an operating point to employ state feedback. See Appendix I.C for this purpose. As states diverge this linear approximation worsens, and the controller becomes less suitable.

Scenario One

To design the quadrotor controller the following state and input operating points are chosen.

$$\bar{x} = [0 \quad 0 \quad 0 \quad 0 \quad \pi/4 \quad 0]^T$$
$$\bar{u} = [1 \quad 1]^T;$$

The state vector $\bar{x}$ has been selected to represent the pitch range of 0 to $\pi/2$. Pitch is important, as it captures directionality of many of the states. Moving below this range the $\cos(\theta)$ terms flip direction, and moving above it the $\sin(\theta)$ terms flip direction. Practically this vehicle may operate in the range $\pm\pi/2$ after which it inverts. The state vector $\bar{u}$ has been selected arbitrarily. Values are non-zero to ensure the linear representation is controllable[3].

Considering the importance of pitch, the following linear quadratic regulator weightings have been chosen.

[33] For simulations involving the quadrotor and hovercraft zero control inputs are mid-rising quantised. Without this many terms disappear from Jacobian matrices and the system becomes uncontrollable.

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}, \qquad R = 1$$

This gives the following gain matrix

$$K = \begin{bmatrix} -0.19 & 259.4 & 9.43 & 269.7 & 15.15 & 14.63 \\ -9.44 & -269.7 & 0.19 & -259.4 & -15.15 & -14.63 \end{bmatrix}$$

which concludes the controller design.

When simulated this first scenario cannot be successfully stabilised. The state trajectories can be seen below. While the pitch angles all begin in the designed operating range, they quickly diverge when trying to reach consensus on other states. This quickly invalidates the linear approximation and so the controller becomes ineffective.

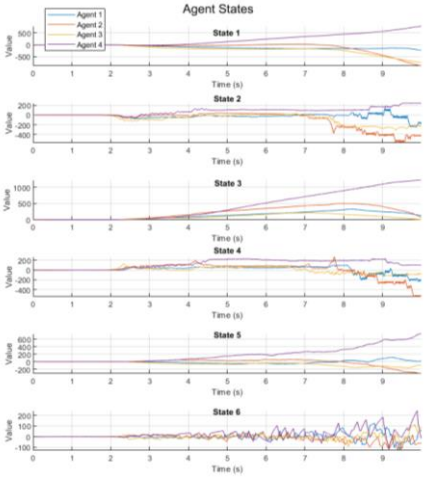


Figure 19 Quadrotor with Fixed Gain Feedback

Scenario Two

To design the controller for this scenario the following state and input operating points are chosen.

$$\bar{x} = [0 \quad 0 \quad 0 \quad 0 \quad \pi/4 \quad 0]^T$$
$$\bar{u} = [1 \quad 1 \quad 1]^T;$$

The rationale follows from scenario one. To get gain values linear quadratic regulator weightings are given as follows.

$$Q = \begin{bmatrix} 9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9 \end{bmatrix}, \qquad R = 1$$

These are chosen to place emphasis on states affected by disturbance. The angular velocity is also favoured again due to its effects on other system states. This gives the following gain matrix

$$K = \begin{bmatrix} 1.491 & 0.841 & 1.491 & 0.841 & 0.701 & 1.817 \\ 1.491 & 0.841 & 1.491 & 0.841 & -7.00 & -1.816 \\ -2.112 & -1.29 & 2.112 & 1.289 & 0 & 0 \end{bmatrix}$$

which finalises the controller.

Under simulation this scenario is successfully stabilised. State trajectories are displayed in the succeeding figure. States p_x and p_y converge at -1.50 and 2.40, missing setpoints by 25% and 20%. This steady state error could be eliminated by introducing integral states. The system settles within 2% of its final value at 7.2 seconds.

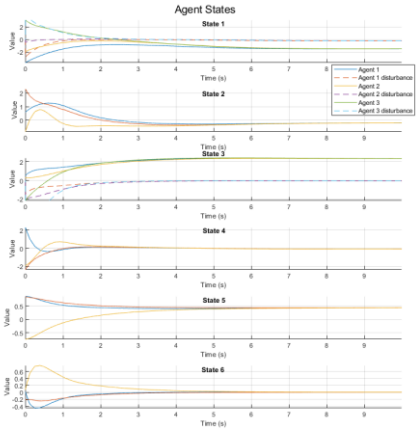


Figure 20 Hovercraft with Fixed Gain Feedback

One glaring issue is that event-triggers become rapid as the system settles. As the event-threshold e_T is reduced arbitrarily near zero even minor fluctuations in the wind model raise events. To resolve this a lower bound of δ may be imposed on the threshold. With this an augmented threshold $e_T{}'$ is given as

$$e_T'(t) = \max\left(e_T(t), \delta\right)$$

where $\delta \geq 0$. A similar concept is seen in the continuous state-independent strategy studied in II.C [14]. This implies that consensus is bounded rather than exact. Triggering instants with $\delta = 0$ and $\delta = 0.025$ are pictured below. Both appear to have a fixed triggering schedule. With the permissible bound however inter-event times are more spread. Ultimately transmissions after settling are wasted as the information variable is not changing.

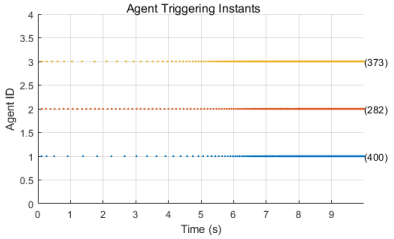


Figure 21 Fixed Controller Scenario 2- $\delta = 0$

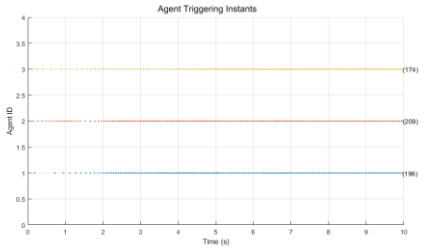


Figure 22 Fixed Controller Scenario 2- $\delta = 0.025$

B. Gain Scheduling

Gain scheduling is an adaptive control technique where controller parameters change with the plant operating conditions [30]. The high-level architecture is pictured in Figure 23. This scheme was introduced for missile guidance where feedforward gain would adjust to compensate for the varying effects of air pressure and Mach number [30]. It is particularly salient for an event-triggered design as the adaptation is minimal, and slowly varying. Leveraging distributed computing this is feasible for a MAS.

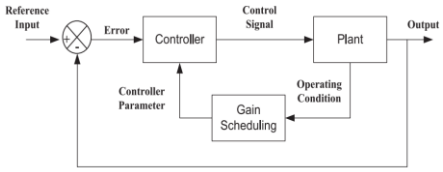


Figure 23 Gain Scheduling Architecture [31]

The standard design procedure of a gain scheduled controller can be generalised to three steps [30]. The first is to select a set of operating points covering the plant range. System values and nonlinearities are formulated as time-varying parameters. Following this a linear approximation is constructed for each point, and an appropriate controller is designed. Finally, gains are interpolated to ensure smooth transitions and create a global compensator. During operations the calculated gains are decided with real-time measurements.

This design scheme is less instinctive under event-triggering operations. Instead the system is linearised at its current operating point when an event is triggered or broadcast is received. A linear quadratic regulator penalising states and inputs equally calculates the gain for stability. No interpolation occurs. Owing to the nature of event triggering the system will be in a similar region and hence produce comparable gains. The updated controller design is accurate until the states diverge, at which time it will be redesigned through a new event. This design approach is dynamic, and less involved.

Scenario One

With the gain-scheduling controller this scenario can be stabilised. This is shown in the succeeding figure. The improvement is attributed to the repeated controller redesign. The region of stability is shifted with agent states and inputs. Consensus is achieved on position and velocity with a 2% spread 8 seconds into the simulation. This is slowed by fluctuations in the agent pitch.

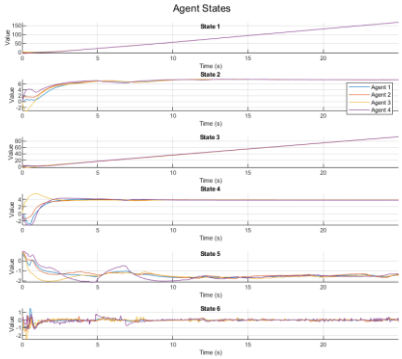


Figure 24 Gain Scheduling Quadrotor

Triggering instants are sparse, occurring 0.89% of all triggering instants. They are mostly caused by variations in pitch – which does not accurately settle. This is expected as angular acceleration is highly actuated. The trigger rate is deceptively small as exact consensus is not achieved. Arguably performance is acceptable; displacement and velocity are the most vital information states in avoiding loss of separation.

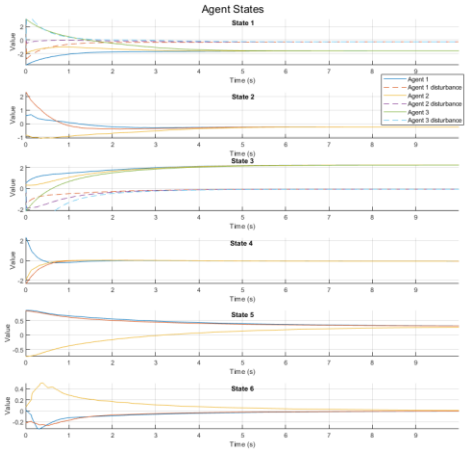
Figure 25 Gain Scheduling Quadrotor Triggers

Scenario Two

This scenario is again successfully stabilised. When compared with the fixed controller the system response is very similar. Smoother transitions with less overshoot are seen in the acceleration states $\dot{p}_x, \dot{p}_x$ and $\dot{\theta}$ to reduce errors. Despite this event-trigger behaviour is analogous. Again $\delta = 0.025$ was used as a lower bound on the event-triggering threshold. Without any mechanisms for disturbance rejection this controller still struggles with steady-state errors and redundant triggers.

Commented [LD40]: Can you be more explicit about the reasons? Is this just due to a better model (i.e. linear model is now valid at each time due to updating)?

Also, hard to compare to previous approaches. Think the previously mentioned graph(s) will really help us to make strong comparisons and reduce the number of plots you may need to include.



C. *Sliding Mode Control*

Sliding mode control (SMC) is a non-linear technique which exhibits robustness, reduced order dynamics and finite-time convergence [32]. These properties make it attractive for practical UAM. Within some modifications it has also been used combat network artefacts such as packet loss and latency [33]. It is a type of variable structure system, which presents some challenges in its combination with consensus and the surveyed event-triggering algorithms.

In variable structure control systems the control law changes according to rules based on system states [34]. To illustrate this process, consider the double integrator system

$$\dot{x}(t) = u(t)$$

and the feedback control law

$$u(t) = -kx(t)$$

where k is a positive scalar. With this the relationship between $\dot{x}$ and x follows an ellipse according to

$$\dot{x}^2 + kx^2 = c$$

where c is an arbitrary initial condition. With this control law variables do not move toward the origin. This is shown in the following phase portraits with different gains k_1 and k_2 [34].

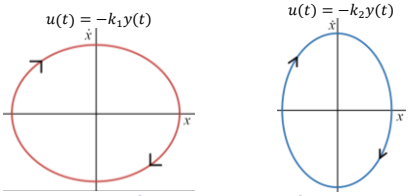


Figure 26 Simple Harmonic Motion Phase Portrait

Combining these the following switching law

$$u(t) = \begin{cases} -k_1\dot{x}(t), & if\ x\dot{x} < 0 \\ -k_2\dot{x}(t), & otherwise \end{cases}$$

where $0 < k_1 < 1 < k_2$ the system adopts a variable structure. This is seen in the phase portrait, which is split into distinct regions. The distance from the centre now monotonically decreases in a spiralling way [34].

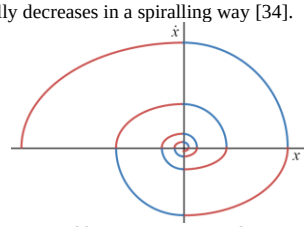


Figure 27 Variable Structure Motion Phase Portrait

A more emphatic example may be given with the following variable structure law [34].

$$u(t) = \begin{cases} -1, & if\ s(x,\dot{x}) > 0 \\ 1, & if\ s(x,\dot{x}) < 0 \end{cases}$$

Here $s(t)$ is a switching function given as

$$s(t) = \sigma_1 x(t) + \dot{x}(t)$$

where σ_1 is a positive scalar. This is known as the signum function, and is denoted as $sign(\sigma_1 x(t) + \dot{x}(t))$. The points where $s(t) = 0$ define a sliding surface. Given by

$$\dot{x}(t) = -\sigma_1 x(t)$$

this is represented on a phase portrait by a line with gradient $-\sigma_1$. The objective is to guide states to this surface and then cause a sliding motion along it by switching the control law at an extreme frequency. The system effectively acts with reduced order, and independently of the control while sliding [34].

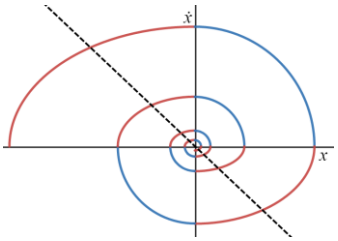


Figure 28 Sliding Surface Phase Portrait

When considering s as the sliding surface, and $\dot{s}$ as the rate of change towards it a sufficient condition for sliding to occur is given as $s\dot{s} < 0$. This is known as a reachability condition. This condition is enforced by the $sign(s(t))$ controller. Its main limitation is that it causes an asymptotic approach which delays the sliding motion. The system may also overshoot the sliding surface. The η-reachability condition

$$s\dot{s} < -\eta|s|$$

instead provides sliding in finite time by including a linear feedback component. It is given by

$$s(t) = \sigma x(t)$$

$$u(t) = -(\sigma B)^{-1}[\sigma A x(t) + \lambda sign(s)]$$

where σ is the designed manifold, λ is the switching gain, A and B are state space matrices.

The regular form design methodology describes a general approach for sliding plane design [35]. This first requires that state space matrices $A \in R^{n\times n}$ and $B \in R^{n\times m}$ are transformed into a more amicable state space representation.

Commented [LD41]: file:///C:/Users/liamd/OneDrive%20-%20Queensland%20University%20of%20Technology/EGH400-2%20Research%20Project%202/slidibg/9780429075933_webpdf.pdf

file:///C:/Users/liamd/Downloads/9781315136141_webpdf%20(4).pdf

https://link.springer.com/content/pdf/10.1007%2F978-981-15-6311-9.pdf

Commented [LD42]: equivalent control!!

$$\begin{bmatrix}\dot{\bar{x}}_1(t)\\ \dot{\bar{x}}_2(t)\end{bmatrix}=\begin{bmatrix}\bar{A}_{11} & \bar{A}_{12}\\ \bar{A}_{21} & \bar{A}_{22}\end{bmatrix}\begin{bmatrix}\bar{x}_1(t)\\ \bar{x}_2(t)\end{bmatrix}+\begin{bmatrix}0_{(n-m)\times m}\\ \bar{B}_2\end{bmatrix}\bar{\boldsymbol{u}}(\boldsymbol{t})$$

For this purpose there exists a transformation matrix T_r which is applied as follows.

$$\bar{x}(t)=Tr\cdot x(t)$$

$$T_r\cdot A\cdot T_R^T=\bar{A}=\begin{bmatrix}\bar{A}_{11} & \bar{A}_{12}\\ \bar{A}_{21} & \bar{A}_{22}\end{bmatrix}$$

$$T_rB=\bar{B}=\begin{bmatrix}0_{(n-m)\times m}\\ \bar{B}_2\end{bmatrix}$$

Matrix T_r may be computed through QR-decomposition [34]. Partitioning of the new system is done with the dimensions given below.

$$\bar{x}_1(t)\in R^{(n-m)\times 1},\qquad \bar{x}_2(t)\in R^{m\times 1}$$
$$\bar{A}_{11}\in R^{(n-m)\times(n-m)},\qquad \bar{A}_{12}\in R^{(n-m)\times m}$$
$$\bar{A}_{11}\in R^{m\times(n-m)},\qquad \bar{A}_{12}\in R^{m\times m}$$
$$\bar{B}_2\in R^{m\times m}$$

Combining the regular form system with the switching function the ideal sliding motion is given as follows.

$$\dot{\bar{x}}_1(t)=(\bar{A}_{11}-\bar{A}_{12}\bar{\sigma})\,\bar{x}_1(t)$$

By stabilising this through matrix $\bar{\sigma}\in R^{m\times(n-m)}$ the switching gain for the original system $\sigma\in R^{m\times n}$ is expressed as follows.

$$\sigma=[\bar{\sigma}\quad I_{m\times m}]\cdot Tr$$

This may be substituted into the desired controller.

Gao's reaching law is chosen for the discrete sliding mode implementation in this report. It is given as the following [33].

$$s(k+1)=(1-\tau\epsilon)s(k)-\tau\lambda sign\big(s(k)\big)$$

The value of τ is fixed as the sampling period. Variables ϵ and λ are design parameters which must satisfy $(1-\epsilon\tau)>0, \epsilon>0$ and $\lambda>0$. High values of the former speed up the reaching phase. The latter dictates the controller magnitude when sliding. Related to a discrete system by the derivation of the sliding variable

$$s(k)=\sigma x(k)$$
$$s(k+1)=\sigma x(k+1)$$

the reaching law is expressed through the following controller.

$$u(k)=-(\sigma H)^{-1}\big[\sigma Gx(k)-(1-\tau\epsilon)s(k)+\tau\lambda sgn\big(s(k)\big)\big]$$

Based on the continuous time η-reachability condition it reaches the sliding manifold in finite time.

1. Linear Sliding Mode Control

State feedback and sliding mode controllers may be simulated in a simple scenario to highlight their properties. Consider the simple linear model seen in section V. A sinusoidal disturbance is introduced to assess robustness, which alters dynamics as follows.

$$\begin{bmatrix}\dot{x}_1(k+1)\\ \dot{x}_2(k+1)\end{bmatrix}=\begin{bmatrix}1 & 0.01\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1(k)\\ x_2(k)\end{bmatrix}+\begin{bmatrix}0.0001\\ 0.01\end{bmatrix}u(k)-\begin{bmatrix}0\\ 0.5\end{bmatrix}\sin(0.01k)$$

A single agent with starting states $x_0=[4.97\quad -2.76]^T$ will be simulated with a sampling period of 0.01 seconds.

The state feedback benchmark is first designed with a linear quadratic regulator heavily weighted towards x_2. It is given as follows.

$$u(k)=-[0.98\quad 3.41]x(k)$$

When simulated the agent starts to converge asymptotically but cannot do so exactly. The states circle around the origin in accordance with disturbance. This is reflected in the phase portrait below.

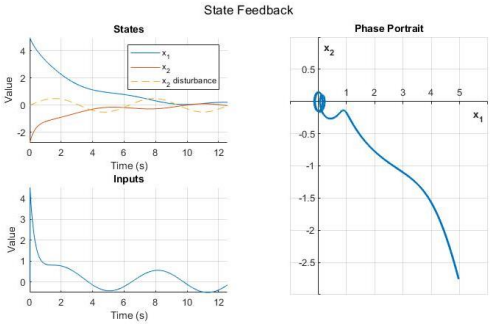


Figure 29 State Feedback States and Inputs (left) and Phase Portrait (right)

A sliding mode controller is designed instead to perform better under disturbance. Design parameter λ is chosen as 0.6 to eclipse peak disturbance and ϵ is arbitrarily picked as a small value of 1 to not slow the transient response. The sliding mode gain is decided with an equally weight linear quadratic regulator as $\sigma=[0.995\quad 1.00]$. With Gao's reaching law this gives the following controller.

$$u(k)=-0.990x_2(k)-0.597sgn(0.995x_1(k)+x_2(k)))$$

It is worth noting that this is reminiscent of the finite time controller studied in section II.E. When simulated the agent rejects the disturbance, and is successfully regulated. It begins in the reaching phase where the agent is pushed onto the sliding manifold. Once this occurs it progresses directly to the origin. This is pictured in the succeeding figure. The ideal sliding motion is slightly skewed when the disturbance is high. This is reflected in the agent input where the signum function stops switching and locks in the opposing direction. As the state feedback controller failed to regulate, the net control effort for these two controllers is similar.

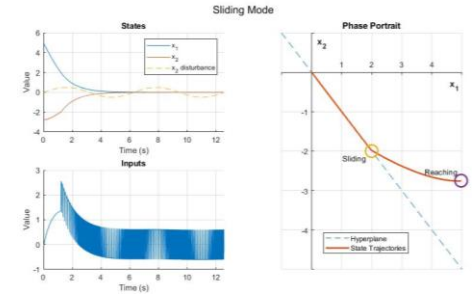


Figure 30 Sliding Mode States and Inputs (left) and Phase Portrait (right)

2. Event Triggering Design Considerations

The switching control law required to maintain sliding contradicts previously surveyed consensus and event-triggering techniques. The state-dependent strategies previously explored enforce perfect consensus by reducing

the triggering threshold to zero. Sliding mode control is obviously incompatible with this.

The high-frequency jittering while sliding is bounded by a quasi-sliding mode band [33]. The size is defined according to

$$s^{+}(k) - s^{-}(k) \leq \frac{\tau \cdot \lambda}{1 - \tau \cdot \epsilon}, \qquad \mathrm{k} > \mathrm{k_s}$$

where $s^{+}(k)$ and $s^{-}(k)$ are respective upper and lower thresholds, and k_s is the sample where sliding is reached. In the previously example it is equal to $6.06 \cdot 10^{-3}$. This band can be seen in the succeeding figure, where the sliding variable zig-zags within its spread. While the sliding variable remains within this range the system is considered sliding. By keeping the values of λ and ϵ low fluctuations are kept tight. This design choice extends the transient response.

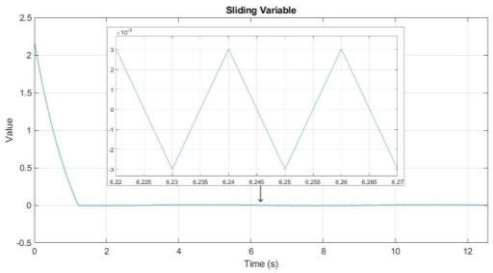


Figure 31 Progression of the sliding variable

Event-triggering strategies surveyed can be made to accommodate sliding mode control with some augmentations. While they may already be suitable for the smooth reaching phase another operating regime should be introduced for sliding. The controller simplifies during the latter.

$$\lim_{s \to 0} u(k) = -(\sigma H)^{-1}\left[\tau\lambda \cdot sgn\big(s(k)\big)\right]$$

This gives peak input values $u_{max}(k)$ as

$$u_{max}(k) = |(\sigma H)^{-1}| \cdot (\boldsymbol{\tau\lambda})_{n\times 1}, \qquad k > k_s$$

where k_s denotes the sample when sliding is reached. The maximum input spread Δ_{u-max} is determined from this as follows.

$$\Delta_{u-max} = 2 \cdot |(\sigma H)^{-1}| \cdot (\boldsymbol{\tau\lambda})_{n\times 1}, \qquad k > k_s$$

When combined with the input state space matrix H, bounds on maximum movement from control switching may be discerned. Labelled as Δ_{x-max}, it is given with the succeeding expression.

$$\Delta_{x-max}(k) = 2 \cdot |H||(\sigma H)^{-1}|(\boldsymbol{\tau\lambda})_{n\times 1}, \qquad k > k_s$$

This can be taken as a minimum bound on a redesigned event threshold $e_T{}'$

$$e_T' = \max(e_T(k),\ (1+\eta) \cdot \Delta_{x-max}(k) + \delta)$$

where $\eta > 0$ is a design parameter. It expands the triggering threshold to give some tolerance against imperfect sliding. An adverse effect of this is loosening the bounds on consensus. Again δ is provided as a disturbance baseline. Combined with open-loop estimation this provides an envelope for sliding state trajectories. This is predicated on the equivalent control being zero.

General operations must also be slightly modified. Event triggers no longer correspond to actuator updates – this instead must occur every sampling instant for sliding. They are only indicative of the consensus variable adjusting through a network transmission. This is motivated by redundant triggers in the previous hovercraft scenarios. Ideally between events agents can regulate themselves in the presence of disturbance. When sliding state broadcasts must be an equivalent representation, which is taken as a rolling average of their values. Projecting this more accurate representation is necessary, especially when open loop estimation is used to predict the consensus target.

The state-dependent model-based strategy from V.B may be augmented with proposed updates and simulated to evaluate the changes. Consider a four-agent system with the dynamics discussed in the previous sub-section. Starting states are $x_{01} = [0 \quad 2]^T$, $x_{02} = [-2.63 \quad 1.78]^T$, $x_{03} = [4.63 \quad 2.94]^T$ and $x_{04} = [0.21 \quad -3.37]^T$. This discrete time network matrix is given as follows.

$$D = \begin{bmatrix} 1.00 & 0.00 & 0.00 & 0.00 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

Note that agent one is a virtual leader used to establish a goal information variable. It only broadcasts its state on start-up. The tolerance parameter η is set as 0 for this simulation as the dynamics are very simple and perfect sliding is imaginable. The value of δ is set as 0.01 for the permissible disturbance bound. This is below the peak disturbance effect each sample. Setting $\sigma = [0.995 \quad 1.00]$, $\epsilon = 1$ and $\lambda = 5$ a controller is designed as the following.

$u(k) = -0.990x_2(k) - 4.975sgn(0.995x_1(k) + x_2(k)))$

This is similar to the controller in the previous sub-section, however λ is increased to magnify the sliding band and hence emphasise the properties of triggering changes. The state trajectories and event-broadcasting instants are displayed in the following. Agents converge quickly and accurately.

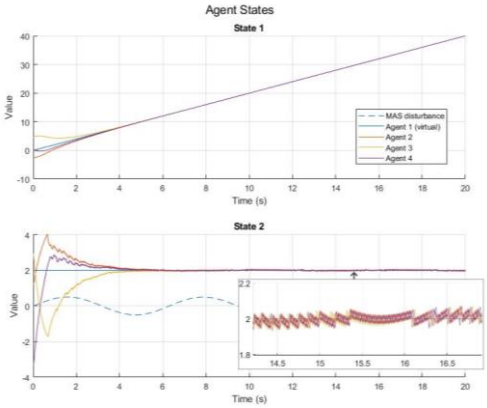


Figure 32 Linear SMC System States

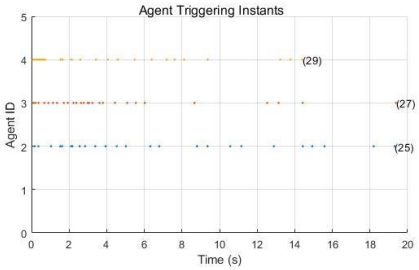


Figure 33 Linear SMC Event Triggers

Event triggers remain sporadic throughout the simulation, despite disturbances and switching. They occur on average 1.38% of sampling instants. Seeing the zoomed section in Figure 32 the disturbance causes a varying effect in sliding. This does not cause excessive triggering due to the δ tolerance threshold. Without the redesigned threshold this occurs every instant after sliding, totalling a much greater rate of 90.4%.

High frequency switching has additional practical consequences. It causes actuator wear and tear while potentially exciting unmodelled dynamics [34]. The energy consumed to do so may also be wasted – directly opposing one of the drivers for event-triggered control. Practically it may not be possible due to restrictions on actuator switching bandwidth. A more realistic way to model control updates is with a smooth, continuous function such as signum. This however further expands the sliding mode band. It will not be considered in this study.

3. Nonlinear Sliding Mode Control

The sliding manifold regular design approach requires a linear system representation. To keep this accurate the controller is redesigned every time an event is triggered. This is an evolution of the gain scheduling process. If the system is reaching the current states and inputs are used as the linearisation points. If the system is sliding equivalent states and inputs are used instead. Gains are picked with an equally penalising linear quadratic regulator.

Scenario One

The quadrotor is successfully stabilised. The controller uses parameters $\epsilon = 2$ and $\lambda = 5$. These have been tuned maximise convergence rate while keeping the sliding band and overshoot minimal. The augmented threshold uses $\eta = 2$ and $\delta = 0$. The value of η is selected to tolerate mild slippage in and out of sliding. This may occur when sliding gains are adjusted on controller redesigns. This effectively causes the manifold to shift. Without disturbance δ is simply selected as zero. The controller rapidly reaches consensus on position and velocity. This occurs at a comparable rate to the gain-scheduled controller, however the final information states are now closer to average consensus. This controller also reaches consensus on pitch.

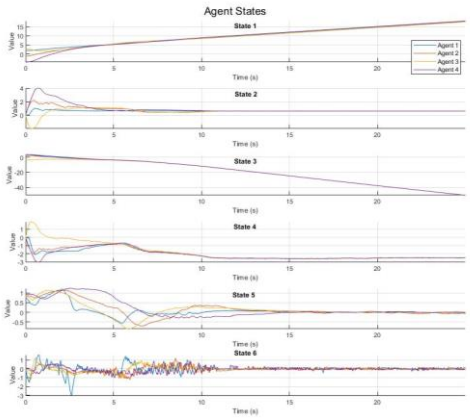


Figure 34 Quadrotor with Sliding Mode Controller

Redundant event-triggers are effectively avoided under the sliding mode controller and augmented event-triggering threshold. There is an initial flurry of triggers, however once the sliding manifold is reached these diminish. Occasionally they re-appear as the system adjusts. Practically the system will not remain at rest. Whether it is due to a moving reference or switching topology this equilibrium will inevitably be disturbed to result in more rapid triggers.

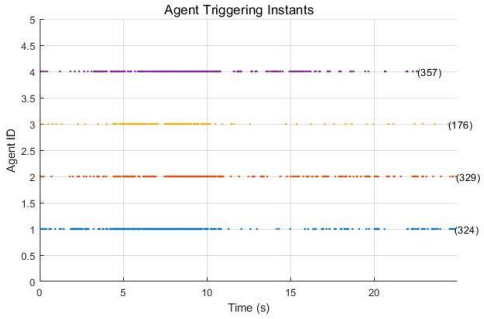


Figure 35 Quadrotor Triggers with Sliding Mode Controller

Scenario Two

The sliding mode controller can stabilise the hovercrafts. It uses parameters $\epsilon = 2$ and $\lambda = 20$. For disturbance rejection λ has been selected as a large value. Fast reaching is not critical so ϵ has been kept low. The augmented threshold uses $\eta = 2$ and $\delta = 0.025$. The selection of η follows the same rationale in scenario one. The value of δ is kept consistent with previous simulations. Consensus is reached rapidly - must faster than the feedback controllers. This is a consequence of finite time convergence and high gain. Large oscillations are seen as a repercussion. This controller can also successfully overcome disturbances to meet the virtual leader.

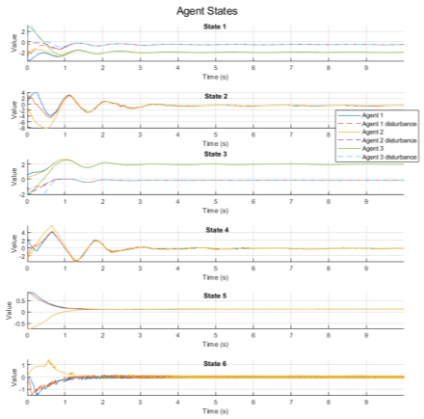


Figure 36 Hovercraft with Sliding Mode Controller

The sliding mode controller has been able to successfully reduce event triggers. Agents regulate between broadcasts, remaining within their event threshold. With a large switching gain this comes at the cost of a high control effort.

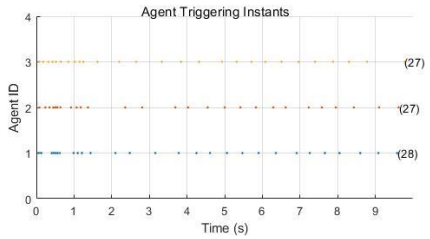


Figure 37 Hovercraft Triggers with Sliding Mode Controller

Inspecting the network error norm provides insight into the controller performance. There is a large initial disparity. This is reflective of the fast, finite time reaching. This continues until the sliding mode controller stops oscillating. There is a large error floor, which is caused by the updated triggering threshold. This means that exact consensus is not reached.

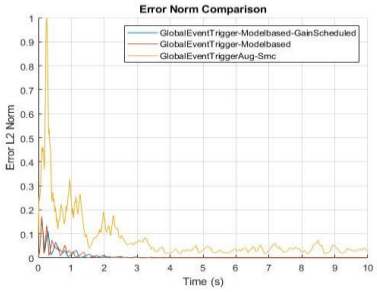


Figure 38 Scenario Two L2 Error Norms

VII. Conclusion

UAM and UTM will require control strategies which can efficiently and effectively arbitrate consensus among large systems. This report explores how event-triggering can reduce resource consumption while not compromising performance. It finds that generally state-dependent strategies outperform state-independent strategies. A form of state estimation is required for systems not at rest. Event triggering was trialled for nonlinear systems with state feedback, gain scheduled and sliding mode controllers. The latter exhibits properties of robustness and finite time convergence. This makes it the most appropriate for the context. An augment is suggested for this controller to be generally compatible with event-triggered systems. Future works will consider this combination in a less idyllic environment, with network artefacts, switching and reference tracking.

APPENDIX I[4]

A. Graph Preliminaries

A team of p vehicles may have information exchanges modelled by graphs (V_p, E_p) which are either directed or undirected. Both cases have a node set $V_p = \{1, \dots, p\}$ and edge set $E_p \subseteq V_p \times V_p$. In the directed case edge (i, j) denotes that child node j may receive information from parent node i, although not vice versa. In the undirected case edge (i, j) signifies both nodes may receive information from one another. This may be categorised as a special directed graph case, where undirected (i, j) implies a directed couple (i, j) and (j, i). Self-edges are not allowed, unless stated. A weighted graph maps a weight to every edge.

Directed and undirected paths are sequences of edges (i_1, i_2), (i_2, i_3), … in directed and undirected graphs respectively. These are cycles instead if starting and ending at the same node.

A directed tree is where every node has a parent except for a root node, which consequentially has s directed path to its all its descendants. A directed graph is strongly connected if every node is the root of a directed tree reaching all other nodes. An undirected tree is where every pair of nodes is connected by a path. This is analogous to an undirected graph being connected.

B. Matrix Preliminaries

The adjacency matrix $\mathrm{A}_p = [a_{ij}] \in \mathbb{R}^{p \times p}$ of a graph G_n reflects the weights of edges (j, i). Value a_{ij} is 0 if $(j, i) \notin E_p$, as self-edges are not allowed this causes the diagonal to be zeros. For an undirected graph the adjacency matrix is symmetrical as $(j, i) \in E_p$ requires $(i, j) \in E_p$ causing $a_{ij} = a_{ji}$. They are also balanced, meaning $\sum_{j=1}^{p} a_{ij} = \sum_{j=1}^{p} a_{ji}$ for all i.

There is a corresponding Laplacian matrix $L_p = [l_{ij}] \in \mathbb{R}^{p \times p}$ which is defined as $L_p \triangleq D_{in} - \mathrm{A}_p$. Here the in-degree matrix $D_{in} = [d_{ij}] \in \mathbb{R}^{p \times p}$ is given as $d_{ij} = 0, i \neq j$ and $d_{ii} = \sum_{j=1}^{p} a_{ij}, i = 1, \dots, p$. This is not the common Laplacian matrix definition, however it has relevance to consensus algorithms.

There is also a corresponding row stochastic matrix $D_p = [d_{ij}] \in \mathbb{R}^{p \times p}$ which is nonnegative and has every row sum to one. It may be calculated as $D_p = (I + D_{in})^{-1} \cdot (I + A)$. The identity matrix $I = [i_{ij}] \in \mathbb{R}^{p \times p}$ is given as $i_{ij} = 0, i \neq j$ and $i_{ii} = 1$. D_{in} is the in-degree matrix as before. The product of these are still row stochastic. It is indecomposable and aperiodic (SIA) if $\lim_{k \to \infty} D^k = \mathbf{1} y^T$ for $y \in \mathbb{R}^n$. They have the eigenvalue 1 for eigenvector $\mathbf{1}_n$. This matrix is used to study consensus in the discrete time setting.

C. Linearisation

Consider a nonlinear equation of the following form.

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

Where f is a function which maps $\boldsymbol{R^n} \times \boldsymbol{R^m} \to \boldsymbol{R^n}$. This may be represented as a linear state space model at state $\bar{x}$ and input $\bar{u}$ operating points with Jacobian matrices.

$$A = \left(\frac{\partial f}{\partial x}\right)^T |_{x=\bar{x}, u=\bar{u}} \in \boldsymbol{R^{n \times n}}$$

$$B = \left(\frac{\partial f}{\partial u}\right)^T |_{x=\bar{x}, u=\bar{u}} \in \boldsymbol{R^{n \times m}}$$

These can be formulated as a system model for approximate states $\tilde{x}$ and inputs $\tilde{u}$.

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u}$$

[4] The work in this appendix is adapted exclusively from [6].

Commented [LD43]: reference proper

Commented [LD44]: This matrix is used to study consensus in settings where the topology evolves and is subject to disturbances.

REFERENCES

[1] A. Straubinger, R. Rothfeld, M. Shamiyeh, K.-D. Buechter, J. Kaiser and K. O. Ploetner, "An overview of current research and developments in urban air mobility – Setting the scene for UAM introduction," *Journal of Air Transport Management*, vol. 87, 2020.

[2] V. Bulusu, "Urban Air Mobility: Deconstructing the Next Revolution in Urban Transportation - Feasibility, Capacity and Productivity," ProQuest LLC, Ann Arbor, 2019.

[3] Uber Elevate, "Fast forwarding to the future of on-demand, urban aviation," Uber Technologies Inc, 2020. [Online]. Available: https://www.uber.com/in/en/elevate/uberair/. [Accessed 30 August 2020].

[4] Airbus, "Urban Air Mobility – the sky is yours," Airbus Group, 27 November 2018. [Online]. Available: https://www.airbus.com/newsroom/stories/urban-air-mobility-the-sky-is-yours.html. [Accessed 30 August 2020].

[5] X. Yang and P. Wei, "Scalable Multi-Agent Computational Guidance with Separation Assurance for Autonomous Urban Air Mobility," *American Institute of Aeronautics and Astronautics*, vol. 43, no. 8, pp. 1473-1486, 2020.

[6] W. Ren and R. W. Beard, Distributed Consensus in Multi-vehicle Cooperative Control, London: Springer, 2008.

[7] A. Bemporad, M. Heemels and M. Johansson, Networked Control Systems, Berlin: Springer, 2010.

[8] X. Wang and M. D. Lemmon, "Event-Triggering in Distributed Networked Control Systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 3, pp. 586-601, 2011.

[9] L. Ding, Q.-L. Han, X. Ge and X.-M. Zhang, "An Overview of Recent Advances in Event-Triggered Consensus of Multiagent Systems," *IEEE Transactions on Cybernetics*, vol. 48, no. 4, pp. 1110-1123, 2018.

[10] K. L. Moore, M. D. Weiss, J. P. Steele, C. Meehan, J. Hulbert, E. Larson and A. Weinstein, "Experiments with autonomous mobile radios for wireless tethering in tunnels," *Journal of Defense Modeling and Simulation*, vol. I, no. 9, p. 45–58, 2012.

[11] X. Ge, Q.-L. Han, D. Ding, X.-M. Zhang and B. Ning, "A survey on recent advances in distributed sampled-data cooperative," *Neurocomputing*, vol. 275, pp. 1684-1701, 2018.

[12] Z. Zhang, F. Hao, L. Zhang and L. Wang, "Consensus of linear multi-agent systems via event-triggered control," *International Journal of Control*, vol. 87, no. 6, pp. 1243-1251, 2014.

[13] S. S. Kia, J. Cortes and S. Martinez, "Distributed event-triggered communication for dynamic average," *Automatica*, vol. 59, pp. 112-119, 2015.

[14] G. S. Seyboth, D. V. Dimarogonas and K. H. Johansson, "Event-based broadcasting for multi-agent average consensus," *Automatica*, vol. 49, no. I, pp. 245-252, 2012.

[15] E. Garcia, P. J. Antsaklis and L. A. Montestruque, Model-Based Control of Networked Systems, New York: Springer International Publishing , 2014.

[16] Y. Liu and X. Hou, "Event-triggered consensus control of disturbed multi-agent systems using output feedback," *ISA Transactions*, vol. 91, pp. 166-173, 2019.

[17] D. Yang, X. Liu and W. Chen, "Periodic event/self-triggered consensus for general continuous-time linear multi-agent systems under general directed graphs," *IET Control Theory and Applications*, vol. 9, no. 3, p. 428–440, 2015.

[18] Y. W. Meng, L. Xie and R. Lua, "An input-based triggering approach to leader-following problems; Hongye Su; Zheng-Guang Wu," *Automatica*, vol. 75, p. 221–228, 2017.

[19] V. T. Haimo, "Finite Time Controllers," *SIAM Journal on Control and Optimization*, vol. 24, no. 4, pp. 760-770, 1986.

[20] J. Lui, Y. Yu, H. He and C. Sun, "Team-Triggered Practical Fixed-Time Consensus of Double-Integrator Agents With Uncertain Disturbance," *IEEE Transactions on Cybernetics*, vol. Early Access, pp. 1-10, 2020.

[21] X. Lu, "Distributed Event-Triggered Control for Prescribed Finite-Time Consensus of Linear Multi-Agent Systems," *IEEE Access*, vol. 8, pp. 129146 - 129152, 2020.

[22] C. Du, X. Liu, W. Ren, P. Lu and H. Liu, "Finite-Time Consensus for Linear Multiagent Systems via Event-Triggered Strategy Without Continuous Communication," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 19-29, 2020.

[23] J. Liu, Y. Zhang, Y. Yu and C. Sun, "Fixed-Time Event-Triggered Consensus for Nonlinear Multiagent Systems Without Continuous Communications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2221-2229, 2018.

[24] Q. Lu, Q.-L. Han, B. Zhang, D. Liu and S. Liu, "Cooperative Control of Mobile Sensor Networks for Environmental Monitoring: An Event-Triggered Finite-Time Control Scheme," *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4134-4147, 2017.

[25] T. Perez, "Modelling of Physical Dynamical Systems," Queensland University of Technology, Brisbane, 2017.

[26] Z. Jin, S. Waydo, E. B. Wildanger, M. Lammers, H. Scholze, P. Foley, D. Held and R. M. Murray, "MVWT-II: the second generation Caltech Multi-Vehicle Wireless Testbed," in *Proceedings of the 2004 American Control Conference*, Boston, 2004.

[27] A. Battista and D. Ni, "A Comparison of Traffic Organization Methods for Small Unmanned Aircraft Systems," *Transportation Research Record*, vol. 2672, pp. 21-30, 2018.

[28] MathWorks, "atmoshwm," 2021. [Online]. Available: https://www.mathworks.com/help/aerotbx/ug/atmoshwm.html. [Accessed 8 April 2021].

[29] P. Sachs, "Introduction," in *Wind Forces in Engineering*, Guildford, Biddles Ltd, 1978, pp. 1-8.

[30] W. J. R. J. S. Shamma, "Research on gain scheduling," *Automatica*, vol. 36, pp. 1401-1425, 2000.

[31] P. Swarnkar, S. K. Jain and R. Nema, "Adaptive Control Schemes for Improving the Control System Dynamics: A Review," *IETE Technical Review*, vol. 31, no. 1, pp. 17-33, 2014.

[32] Y. Shtessel, C. Edwards, L. Fridman and A. Levant, Sliding Mode Control and Observation, New York: Birkhäuser, 2014.

[33] D. H. Shah and A. Mehta, Discrete-Time Sliding Mode Control for Networked Control System, Singapore: Springer, 2018.

[34] C. Edwards and S. K. Spurgeon, Sliding Mode Control Theory and Applications, Boca Raton: Taylor & Francis Group, 1998.

[35] S. K. Spurgeon, "Hyperplane design techniques for discrete-time variable structure control systems," *International Journal of Control*, vol. 55, no. 2, pp. 445-456, 2007.

[36] P. Tabuada, "Event-Triggered Real-Time Scheduling of Stabilizing Control Tasks," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1680-1685, 2007.

[37] D. Liuzza, D. V. Dimarogonas, M. d. Bernardo and K. H. Johanssona, "Distributed model based event-triggered control for synchronization of multi-agent systems," *Automatica*, no. 73, pp. 1-7, 2016.