Minimizing network utilization in event-triggered control of multi-agent system

Anton Hässler

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6159
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2023

Abstract

The exact solution to a set of equations modeling the event-triggered control of a multi-agent system is derived and computed up to a specified error on the time between events. An attempt to minimize the number of events of the solution subject to bounds on overshoot and convergence time is made using a differential evolution method. Parameter points which result in a low number of events of the bounded solution are found.

Acknowledgements

I want to thank Stefan Ristevski, Lara Laban, Kristian Soltesz and Fredrik Bagge Carlson for their feedback and support when working on this project.

Contents

1

Introduction

The idea with the control studied in this project is to have a multi-agent system move in a formation, where each agent only communicates with its neighbors. Commands to change position are given to only one or a few agents of the system, and the rest of the agents should follow these "leader agents" while retaining formation. Communications are event-triggered, meaning that agents communicate if and only if a certain event-triggering condition is met (in contrast to for example communications happening with a predetermined, constant frequency). The neighbors of each agent are defined through a communication graph, where the links are undirected, meaning that neighbor relations are mutual. The communication graph is seen as a constant in this project, but one can imagine this graph also being a variable. The specific example studied in this project is 1-dimensional, where the communication graph is a line graph, the number of agents is 4 with one leader and the command sent to the leader is a step function. However, everything done in the project can be directly generalized to 3 dimensions, any communication graph and any number of agents (the transition to a continuous time command for example might not be as straightforward, but one can always approximate such using step functions). The formation is thought to exist in the non-varying dimensions, but this is mainly to avoid introducing unnecessary variables, as specifying a formation does not introduce complications mathematically. To aid in visualizing the control, the agents will be thought of as drones in a drone swarm. The control studied comes from the article *An Event-Triggered Distributed Control Architecture for Scheduling Information Exchange in Networked Multiagent Systems* [1], which presents two versions of the control, one we call the sampled version and the other the solution predictor version. It also contains a stability proof of both versions of the control and some simulated and real life demonstrations. In this project, primarily the sampled version is studied as it's arguably the simpler version of the two, but many of the ideas can likely be extended to the solution predictor version.

As with most systems depending on a set of parameters, designed to complete a certain task, one wants to tune the parameters such that the system performs the task as satisfactory as possible. There is often a balance between performance and

cost: improving performance often increases cost and vice versa. Thus, the subject of finding good or even optimal parameters often becomes a subject of study, and developing reliable methods for tuning such parameters is often interesting. In this case, the cost is the number of communication events between the drones, and the performance is bounded by requiring the dynamics to stay inside bounds on overshoot and convergence time. The goal with the project is to minimize this cost.

The initial optimization idea relied on the assumption that the theoretical dynamics changes continuously with the parameters, which allows for gradient based optimization which is very developed and commonly used. However, as it turns out the dynamics as a function of the parameters has a high density of discontinuities, and is in general very "spiky". This makes the gradient less useful. The source of the spiky shape is believed to be discontinuities at trigger order changes of neighboring drones. Instead of using a gradient based approach, a derivative free differential evolution method was used.

The implementation of the theoretical model used in [1] is a multi-threaded Euler-approximation, with one thread for each drone. In this project, a method of evaluating the theoretical dynamics to high accuracy is developed. This is used to compare the theoretical model with the implementation used in [1]. As a reference, a single-threaded Euler approximation is used to illustrate the convergence of the Euler approximation to the theoretical model as $dt \to 0$. The function that is optimized is the number of events of the theoretical model, which can be seen as the number of events in the implementation in [1] as the sampling frequency goes to infinity. Many sets of parameters resulting in a low number of events for the theoretical model are found. However, in order for the found parameters to also result in a low number of events for the implementation in [1] and the single-thread Euler approximation, dt has to be made sufficiently small.

There are many interesting points in this project. One is that it is shown how the theoretical model for the dynamics of an event-triggered system of this type can be evaluated to both high precision and efficiency. Efficient computation allows for better optimization, and precise computation means that the output is very close to the theoretical model. One reason the ability to precisely evaluate a model can be desirable, is that it can give a better understanding of the model, as we can be sure that the particular output of the computation for a set of parameters is not in large due to error in the approximation method. Thus it might give hints on what to change in the underlying model if the output is undesirable, for example. The observation that the dynamics as a function of the parameters can be discontinuous at points which result in simultaneous events of neighboring drones is also interesting, as it could be a quite general fact for systems which have a distributed event-triggered control where state-derivatives change discontinuously at events. Finally, the concept of letting some property of the system dictate when the agents

communicate by using an event-triggering condition is interesting, as this allows for a communication frequency which adjusts itself as the control progresses. In principle, this should allow for a lower amount of network utilization than for example having communications occur with a predetermined constant frequency.

Some examples of possible applications of drone swarms include search and rescue operations, military applications and surveillance. To explore the search and rescue application into a bit more detail, imagine that a person has gone missing. To find the person, a search is conducted. In this search, one can imagine having a large number of drones, say 1000, fly in a line formation, each with a camera filming the terrain below. The footage can be analyzed by an AI, and if the AI deems that there is a high likelihood that the person is on a part of the footage, this can be alerted to the search and rescue team. The usefulness in this is that many drones flying in formation can quickly cover a large area.

The presentation of this work is structured into four parts. The first part describes the dynamical system and the approach taken to compute the dynamics up to a specified error on the time between events. The approach is to first analytically solve the dynamics between events, then to use this solution to compute the time to the next event where a parameter of the control is a line, and finally to use this time as a lower bound to the time to the next event in an iterative procedure. It concludes with plots of the dynamics and plots of the number of events as a function of two varying parameters, keeping the others fixed. The purpose of this part of the work is to efficiently and precisely compute the dynamics of the theoretical model resulting from a set of parameters. The second part describes the bounds set on performance and how the optimization was performed. The third part consists of a discussion, where some ideas for future research are presented. The main findings in the project are summarised in the fourth part, which concludes the presentation of the project. An appendix containing details on how all computations were performed is found last in the document.

All computations were done using the programming language Julia.

2

Description and computation of the dynamics

2.1 Description of the dynamics

In the section "Foundation", the graph defined by Laplacian L is assumed to be connected and undirected. See the section "Properties of the Laplacian" in the appendix for definitions of these properties and proofs of the statements made in this section.

Foundation

The control in [1] is based on the dynamics defined by:

$$\dot{\mu}_i = \alpha(\sum_{j\sim i} \mu_j - \mu_i), \tag{2.1}$$

where α is a scalar with unit s^{-1}. The notation $j \sim i$ means that node j is a neighbor of node i, so that the sum is over all neighbors of node i. In the analysis presented in the following sections, these indices will refer to drones in the swarm, such that each drone has a unique fixed index. In vector form (2.1) can be written as:

$$\dot{\mu} = -\alpha L\mu, \tag{2.2}$$

where L is the Laplacian matrix of the communication graph. The dynamics defined by this equation converges to a vector of ones times the mean of the initial value of μ. Equations (2.1) and (2.2) can be used as a model for many different processes. An illustrative example is the diffusion of a substance between a set of nodes on a graph, where the outflow from a node is proportional to the concentration at the node and evenly distributed over the outgoing links. The variable μ_i would then denote the concentration at node i. The variable μ_i can also be the position of a

drone in a drone swarm, and to specify a formation for the swarm, one can modify the equation using formation parameters h_i as:

$$\dot{\mu}_i = \alpha(\sum_{j\sim i}(\mu_j - h_j) - (\mu_i - h_i)). \tag{2.3}$$

This results in a shift h_i of coordinate i of the fixed point. In the diffusion example, this would correspond to shifting where 0 is on the gauges measuring the concentrations, so that the gauge at node i returns μ_i when the concentration actually is $\mu_i - h_i$. This implies that μ_i is what the gauge at node i is returning, not the actual concentration at node i. The fixed point can be set to c times a vector of ones by modifying the dynamics to:

$$\dot{\mu}_i = \alpha((\sum_{j\sim i}\mu_j - \mu_i) + k_i(c - \mu_i)), \tag{2.4}$$

where $k_i > 0$ if i is the index of the leader drone and zero otherwise. We call such c a command. In the diffusion example, this can be visualized as connecting a source node with constant concentration c and infinite supply to the nodes corresponding to $k_i > 0$. To specify a formation and a command, one can use the following equation:

$$\dot{\mu}_i = \alpha([\sum_{j\sim i}(\mu_j - h_j) - (\mu_i - h_i)] + k_i[c - (\mu_i - h_i)]), \tag{2.5}$$

which converges to $\mathbf{c} + h$, where $\mathbf{c} = c\mathbf{1}$ and $\mathbf{1}$ is a vector of ones. In matrix form, (2.5) can be written as:

$$\dot{\mu} = -\alpha((L+K)(\mu - h) - K\mathbf{c}) \tag{2.6}$$

where K is diagonal and $K(i,i) = k_i$. An intuitive way of looking at the control in [1], is to see it as a way of approximating the trajectories defined by these equations using an event-triggered and distributed control. For more details on the dynamics originating from (2.1), the reader is referred to [2].

Event-triggered approximation of the Laplacian dynamics

What follows is a description of the event-triggered control presented in [1]. There are 6 parameters, γ_1, γ_2, ε, ϕ_f, ϕ_0, and κ, and four state variables, x, x_m, μ and $\hat{\mu}$ (meaning that they are functions of time and determined by the dynamical equations of the model). The variable x denotes the positions of the 4 drones, and the purpose of the other 3 is to get a stable distributed event-triggered control. The variable μ can be seen as a reference trajectory seeking to approximate the Laplacian trajectory, and $\hat{\mu}$ are broadcasts of the reference trajectory to neighboring drones. Each drone has access to its own values of x, x_m, μ and $\hat{\mu}$, and access to the values of $\hat{\mu}$ of neighboring drones. The derivatives of x, x_m and μ are defined as:

$$\dot{x}_i = -\gamma_1(x_i - \hat{\mu}_i), \tag{2.7}$$

$$\dot{x}_{mi} = -\gamma_1(x_{mi} - \mu_i), \tag{2.8}$$

$$\dot{\mu}_i = -\gamma_2((\sum_{j\sim i} \hat{\mu}_i - \hat{\mu}_j) + k_i(\hat{\mu}_i - c)), \tag{2.9}$$

where $k_i = 1$ if i is the index of the leader drone and 0 for all other i. Equation (2.9) can be written in vector form using the Laplacian as:

$$\dot{\mu} = -\gamma_2((L+K)\hat{\mu} - K\mathbf{c}). \tag{2.10}$$

In this case, c will be a step function and the graph a line graph, meaning that L is defined as:

$$L = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}. \tag{2.11}$$

Drone 1 will be the leader drone, which means that K is defined as:

$$K = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{2.12}$$

The states are required to satisfy the following inequality, called the event-triggering condition:

$$|\mu_i - \hat{\mu}_i| < \varepsilon|x_i - x_{mi}| + \phi(t), \tag{2.13}$$

where $\phi(t) = (\phi_0 - \phi_f)e^{-\kappa t} + \phi_f$, where t is elapsed time counting from the initial time of the dynamics $t_0 = 0$. If this equality is not satisfied for some drone, this drone triggers and communicates an update of $\hat{\mu}$ to its neighbors. For the update, there are two versions of the control. One is the sampled version, and the other the solution predictor version. In this project, mainly the sampled version of the control was studied, which simply sets $\hat{\mu}_i = \mu_i$ when drone i triggers. This will be the assumed version henceforth. Given a command c, parameter values and an initial condition, equations (2.7) to (2.13) together with the update rule for $\hat{\mu}$ determine the dynamics of the system. A stability proof for this control is found in [1]. The proof shows that for certain parameter values, the drones will converge to a bounded region centered at c, i.e. $|x_i - c| \leq d$ as $t \to \infty$, where $d \geq 0$ is a finite number. The idea of the proof is to show the existence of a Lyapunov function.

2.2 Computation of the dynamics

Exact solution to the dynamical equations between events

Between events, it is straightforward to solve the dynamical equations. This gives the states as functions of time elapsed from the previous event and is valid until the next event, which can either be the trigger of a drone or a change in the step function c. Let the variable τ be the time counting from any initial time that is after the most recent event, and view the states as functions of τ. Until the next event, it then holds that $\hat{\mu}$ is constant, $\mu(\tau)$ a line:

$$\mu_i(\tau) = \mu_i(0) - \tau\gamma_2((\sum_{j\sim i}\hat{\mu}_i - \hat{\mu}_j) + k_i(\hat{\mu}_i - c)) = \mu_i(0) + q_i\tau = \mu_{i0} + q_i\tau, \quad (2.14)$$

and $x(\tau)$ a decaying exponential:

$$x(\tau) = (x_0 - \hat{\mu})e^{-\gamma_1\tau} + \hat{\mu}, \quad (2.15)$$

where $q_i = -\gamma_2((\sum_{j\sim i}\hat{\mu}_i - \hat{\mu}_j) + k_i(\hat{\mu}_i - c))$, $\mu_0 = \mu(0)$ and $x(0) = x_0$. Using the expression for μ in the differential equation for x_m (equation (2.8)), we get:

$$\dot{x}_m = -\gamma_1(x_m - (q\tau + \mu_0)). \quad (2.16)$$

Multiplying both sides with $e^{\gamma_1\tau}$ and using the product rule, we get:

$$\frac{d}{d\tau}(e^{\gamma_1\tau}x_m) = \gamma_1(q\tau + \mu_0)e^{\gamma_1\tau}. \quad (2.17)$$

The integral of $q\tau e^{\gamma_1\tau}$ can be computed using partial integration:

$$\int_0^\tau qse^{\gamma_1 s}ds = [\frac{qs}{\gamma_1}e^{\gamma_1 s}]_0^\tau - \int_0^\tau \frac{q}{\gamma_1}e^{\gamma_1 s}ds = \frac{q\tau}{\gamma_1}e^{\gamma_1\tau} - [\frac{q}{\gamma_1^2}e^{\gamma_1 s}]_0^\tau$$

$$= \frac{q\tau}{\gamma_1}e^{\gamma_1\tau} + \frac{q}{\gamma_1^2}(1 - e^{\gamma_1\tau}) = \frac{q}{\gamma_1^2} + e^{\gamma_1\tau}(\frac{q\tau}{\gamma_1} - \frac{q}{\gamma_1^2}). \quad (2.18)$$

Thus, integrating both sides of (2.17) gives:

$$x_m(\tau)e^{\gamma_1\tau} - x_m(0) = \mu_0 e^{\gamma_1\tau} - \mu_0 + \frac{q}{\gamma_1} + e^{\gamma_1\tau}(q\tau - \frac{q}{\gamma_1}) \implies$$

$$x_m(\tau) = (x_{m0} - \mu_0 + \frac{q}{\gamma_1})e^{-\gamma_1\tau} + q\tau + \mu_0 - \frac{q}{\gamma_1}, \quad (2.19)$$

where $x_{m0} = x_m(0)$. Between events, the dynamics can be updated exactly according to the model using (2.14), (2.15) and (2.19).

Equations for the time to the next trigger event

Let τ be the time elapsed counting from a chosen initial time after the most recent event. The time to the next trigger event is then the smallest τ such that the event-triggering equation:

$$|\mu_i - \hat{\mu}_i| = \varepsilon |x_i - x_{mi}| + \phi, \tag{2.20}$$

is satisfied for at least one drone. Using (2.14), (2.15) and (2.19), and dropping the index i, the event-triggering equation can be written as:

$$|\mu_0 + q\tau - \hat{\mu}| = \varepsilon |(x_0 - \hat{\mu})e^{-\gamma_1 \tau} + \hat{\mu} - (x_{m0} - \mu_0 + \frac{q}{\gamma_1})e^{-\gamma_1 \tau} - q\tau - \mu_0 + \frac{q}{\gamma_1}| + \phi(\tau). \tag{2.21}$$

Define $\tilde{\mu}_0 = \mu_0 - \hat{\mu}$, $\tilde{x}_0 = x_{m0} - x_0$ and $z = \tilde{\mu}_0 - \tilde{x}_0 - \frac{q}{\gamma_1}$. In these variables the equation becomes:

$$|\tilde{\mu}_0 + q\tau| = \varepsilon |z e^{-\gamma_1 \tau} - q\tau + \frac{q}{\gamma_1} - \tilde{\mu}_0| + \phi(\tau). \tag{2.22}$$

The time Δ to the next trigger event, is the smallest τ such that this equation is satisfied for at least one drone. Note that everything except ϕ and τ is constant in this equation. One way of computing Δ is to, for each drone, compute the smallest time which satisfies its event-triggering equation, and pick the smallest of the solutions. This can be done by noting that the solution δ to the equation for one drone satisfies one of the following equations:

$$q\delta + \tilde{\mu}_0 = \varepsilon (z e^{-\gamma_1 \delta} - \tilde{\mu}_0 + \frac{q}{\gamma_1} - q\delta) + \phi(\delta), \tag{2.23}$$

$$q\delta + \tilde{\mu}_0 = -\varepsilon (z e^{-\gamma_1 \delta} - \tilde{\mu}_0 + \frac{q}{\gamma_1} - q\delta) + \phi(\delta), \tag{2.24}$$

$$-(q\delta + \tilde{\mu}_0) = \varepsilon (z e^{-\gamma_1 \delta} - \tilde{\mu}_0 + \frac{q}{\gamma_1} - q\delta) + \phi(\delta), \tag{2.25}$$

$$-(q\delta + \tilde{\mu}_0) = -\varepsilon (z e^{-\gamma_1 \delta} - \tilde{\mu}_0 + \frac{q}{\gamma_1} - q\delta) + \phi(\delta). \tag{2.26}$$

If we solve each one of these equations, we know that our sought δ is one of the four solutions. To pick the correct solution, we impose the condition that the quantities which were previously inside the absolute values are positive. In other words, the sought solution will be the smallest $\delta > 0$ which satisfies the three conditions:

$$s_l(q\delta + \tilde{\mu}_0) \geq 0, \tag{2.27}$$

$$s_r \varepsilon (z e^{-\gamma_1 \delta} - \tilde{\mu}_0 + \frac{q}{\gamma_1} - q\delta) \geq 0, \tag{2.28}$$

where s_l is either 1 or -1, s_r is either 1 or -1 and:

$$s_l(q\delta + \tilde{\mu}_0) = s_r \varepsilon (z e^{-\gamma_1 \delta} - \tilde{\mu}_0 + \frac{q}{\gamma_1} - q\delta) + \phi(\delta). \tag{2.29}$$

Equation (2.29) can be written as:

$$a\delta + be^{c\delta} + d = \phi(\delta), \tag{2.30}$$

where $a = (s_l + s_r\varepsilon)q$, $b = -s_r\varepsilon z$, $c = -\gamma_1$ and $d = s_l\tilde{\mu}_0 + s_r\varepsilon(\tilde{\mu}_0 - \frac{q}{\gamma_1})$. There are many different possible approaches to solving equations of this form. The approach taken in this project is to first express the exact solution to (2.30) when ϕ is a line in terms of the Lambert W function, and then to use this solution to compute lower bounds to the time to the next event when ϕ is an exponential. This leads to a method for computing the time to the next event up to a specified error when ϕ is an exponential.

Solution when ϕ is a line

Let $\phi(\delta) = \alpha\delta + \beta$. Equation (2.30) then becomes:

$$a\delta + be^{c\delta} + d = \alpha\delta + \beta. \tag{2.31}$$

By defining $\tilde{a} = a - \alpha$ and $\tilde{d} = d - \beta$, it can be rewritten as:

$$\tilde{a}\delta + be^{c\delta} + \tilde{d} = 0. \tag{2.32}$$

Subtracting $be^{c\delta}$ from both sides and then multiplying with $-\frac{c}{\tilde{a}}e^{-c\delta - c\tilde{d}/\tilde{a}}$ gives:

$$-(c\delta + \frac{c\tilde{d}}{\tilde{a}})e^{-(c\delta + c\tilde{d}/\tilde{a})} = \frac{bc}{\tilde{a}}e^{-c\tilde{d}/\tilde{a}}. \tag{2.33}$$

By defining $g(\delta) = -(c\delta + \frac{c\tilde{d}}{\tilde{a}})$ and $\xi = \frac{bc}{\tilde{a}}e^{-c\tilde{d}/\tilde{a}}$, the equation becomes:

$$g(\delta)e^{g(\delta)} = \xi. \tag{2.34}$$

The function $f(x) = xe^x$ is plotted in Figure 1 for $x \in [-4, 0.5]$. The plot illustrates the fact that $xe^x = y$ has no real solution if $y < -1/e$, two real solutions if $-1/e \leq y < 0$, and one real solution if $y \geq 0$. The Lambert W function has the property

$$xe^x = y \implies x = W(y), \tag{2.35}$$

for one of the branches of W [3]. In other words, $W(y)$ returns the solution x to $xe^x = y$. For real arguments, the branches of W are W_0 and the W_{-1}, where $W_0(z)$ is defined for $z \geq -1/e$, and $W_{-1}(z)$ is defined for $-1/e \leq z < 0$. The branches W_0 and W_{-1} can be seen as the x-values corresponding to the blue curve in Figure 2.1 right and left of $x = -1$, respectively.

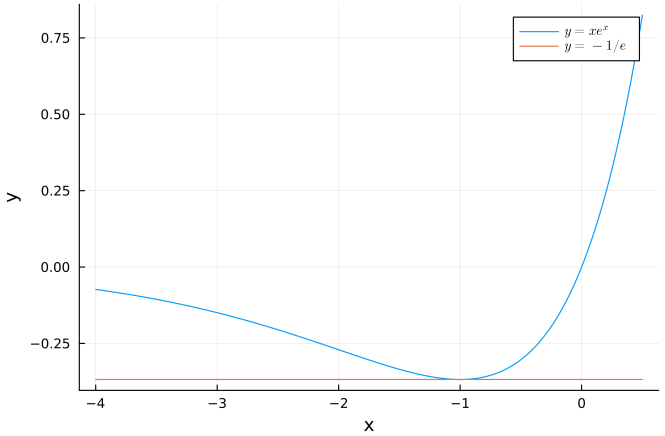

Figure 2.1 Plot of the function $y = xe^x$. The Lambert W function $W(y)$ returns the solution x to the equation $y = xe^x$, given that a branch of W is specified (i.e. which solution to return).

Using the Lambert W function, the solution to (2.31) can be expressed as:

$$g(\delta) = W(\xi) \iff -(c\delta + \frac{c\tilde{d}}{\tilde{a}}) = W(\xi) \tag{2.36}$$

$$\iff \delta = -\frac{d-\beta}{a-\alpha} - \frac{1}{c}W(\xi). \tag{2.37}$$

The procedure used for computing the time to the next trigger event for one drone when $\phi(\delta) = \alpha\delta + \beta$ can now be explained as follows:

- For the four combinations given by $s_l = \pm 1$ and $s_r = \pm 1$, compute all real solutions to (2.29) by evaluating the right hand side of (2.37).
- This means that for each combination of s_l and s_r, if $\xi \geq 0$ one candidate solution is obtained by evaluating $W_0(\xi)$, if $-1/e \leq \xi < 0$ two candidate solutions are obtained by evaluating both $W_0(\xi)$ and $W_{-1}(\xi)$, and if $\xi < -1/e$ no candidate solution is obtained.
- Pick the smallest positive candidate solution which satisfies inequalities (2.27) and (2.28) as the time to the next trigger event for the drone.

The time to the next trigger event when $\phi(\delta) = \alpha\delta + \beta$ will then be the smallest obtained time out of all the drones.

Iteration step for ϕ exponential

The event-trigger inequality can be rewritten as:

$$|\mu_i - \hat{\mu}_i| - \varepsilon|x_i - x_{mi}| < \phi. \tag{2.38}$$

Between events, this inequality is always satisfied. By defining $h_i(\tau) = |\mu_i(\tau) - \hat{\mu}_i| - \varepsilon|x_i(\tau) - x_{mi}(\tau)|$, the time to the next trigger event for drone i can be seen as the first positive point of intersection between $h_i(\tau)$ and $\phi(\tau)$. It is also true that if ϕ is a decaying exponential and T is a tangent to ϕ, then $T(\tau) \leq \phi(\tau)$ for all τ. Thus, computing the time to the next trigger event where ϕ is replaced with a tangent to ϕ will always yield a lower bound to the time to the next trigger event. Using this fact and assuming that the errors in all function evaluations, including evaluations of W, are negligible, the exact dynamics can be computed up to the error E on the time between events by using the following iteration step:

- Let t be the current point of time in the iteration of the dynamics.
- Let T_t be the tangent to ϕ at t.
- Compute the time to the next trigger event with ϕ replaced by T_t as outlined in the section "Solution when ϕ is a line". Denote this time $\tilde{\Delta}$.
- If $c(t+\tilde{\Delta}) \neq c(t)$, let dt be the time to the next change in c and update the states to time $t+dt$ using (2.14), (2.15) and (2.19).
- If $c(t+\tilde{\Delta}) = c(t)$, let $dt = \tilde{\Delta} + E$ and update the states to $t+dt$ using (2.14), (2.15) and (2.19). For all i which $|\mu_i - \hat{\mu}_i| - \varepsilon|x_i - x_{mi}| \geq \phi(t+dt)$, set $\hat{\mu}_i = \mu_i$.
- Update t to $t+dt$.
- Repeat until $t \geq t_{end}$, the end time of the dynamics.

The idea of the iteration process is illustrated in Figure 2.2.

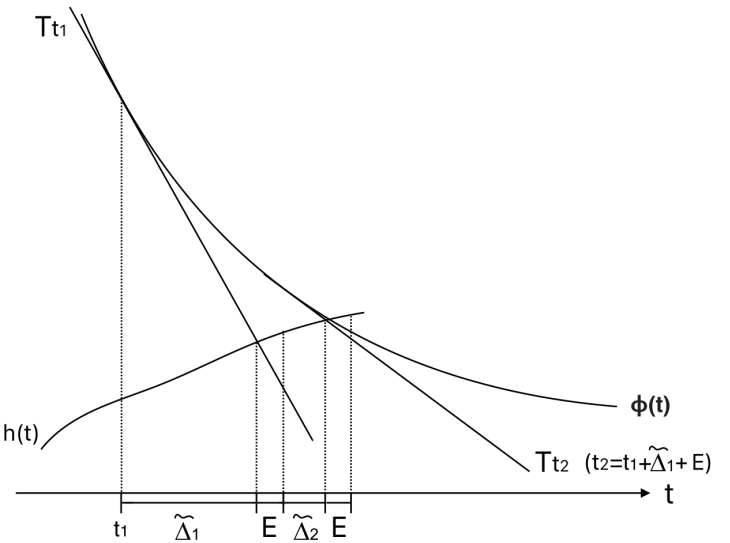

Figure 2.2 Illustration of the iteration. Computing the time to the next event with ϕ replaced with its tangent at the current t gives a lower bound $\tilde{\Delta}$. The dynamics are updated to the time $t+\tilde{\Delta}+E$, where E is the specified error on time between events. The proportions between quantities in the figure are not representative of the typical scenario to help illustrate the idea. In particular, ϕ is almost equal to its tangent and $E \ll \tilde{\Delta}$ for most computations.

An example of the dynamics of the theoretical model is shown in Figure 2.3. Figure 2.4 shows a 3D-plot of the number of events for varying γ_1 and γ_2, keeping the other parameters fixed. The plots were computed using the method outlined in this section, with $E = 10^{-11}$ s. For initial conditions, exact parameter values and all other details required to reproduce these plots, see the section "Simulation specifications" in the appendix. This section contains all specific information required to reproduce the data generated in the remainder of this report.

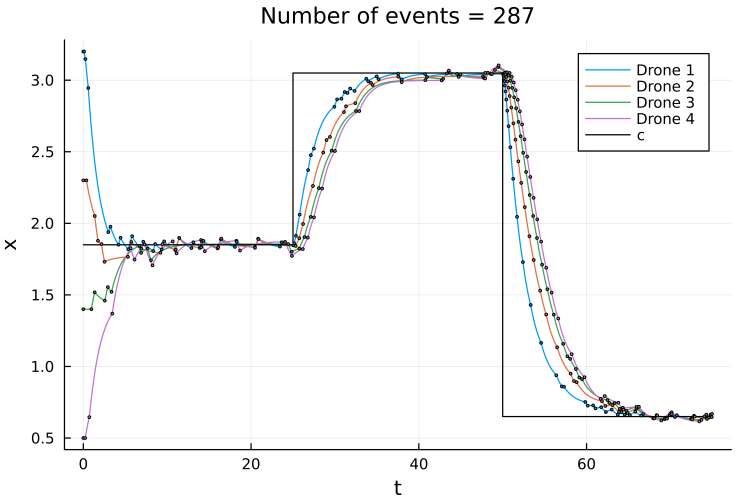

Figure 2.3 Example of dynamics resulting from computation of the theoretical model. Trigger events for each drone are marked with small dots on the trajectory of each drone.

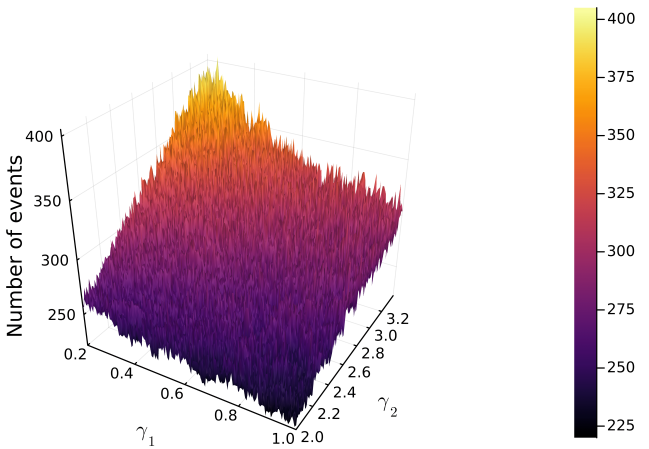

Figure 2.4 3D plot of number of events for $\gamma_1 \in [0.2, 1]$ and $\gamma_2 \in [2, 3.4]$. The other parameters are kept fixed. Yellower colors correspond to more events, and darker to less events. The function is very spiky, and we note that there is an "average gradient". The direction of the gradient is towards higher γ_2-values and smaller γ_1-values in this case.

2.3 Discontinuous points

As is apparent in Figure 2.4, the number of events as a function of the parameters can be described as "spiky". One likely source of this shape is the fact that the dynamics as a function of the parameters has discontinuities at trigger-order changes of neighboring drones. What follows is an investigation of the origins of these discontinuities.

Assume two drones, say drones 1 and 2, are neighbors. Assume also that the two drones trigger almost at the same time, but with drone 1 triggering slightly before drone 2. Now we perturb the parameters slightly so that they trigger exactly at the same time, and then a tiny bit more so that drone 2 triggers slightly before drone 1. This results in a trigger-order change, from drone 1 being first to drone 2 being first, and the question is: Do the dynamics change continuously as the trigger order changes? To answer this question, we look at the trigger inequality of drone i:

$$|\mu_i - \hat{\mu}_i| - \varepsilon|x_i - x_{mi}| < \phi. \tag{2.39}$$

Denote $h_i(t) = |\mu_i - \hat{\mu}_i| - \varepsilon|x_i - x_{mi}|$. When drone 2 triggers, it communicates an update of $\hat{\mu}_2$ to drone 1 since they are neighbors. This does not change the value of h_1, as h_1 only depends on state variables of drone 1. But $\dot{\mu}_1$ depends on $\hat{\mu}_2$, since $\dot{\mu}_i = -\gamma_2((\sum_{j\sim i}\hat{\mu}_i - \hat{\mu}_j) + k_i(\hat{\mu}_i - c))$, and thus the trigger of drone 2 causes a discontinuous change of $\dot{h}_1$. Say that the drones trigger exactly at the same time, and the simultaneous event occurs at time t_{sim}. Then, as drone 2 is made to trigger infinitesimally before drone 1, if the discontinuous change in $\dot{h}_1(t_{sim})$ that happens as a consequence of the trigger of drone 2 results in $\dot{h}_1(t_{sim}) < \dot{\phi}(t_{sim})$, h_1 will not intersect ϕ at $t = t_{sim}$, i.e. there is a finite change in the dynamics resulting from an infinitesimal perturbation in the parameters, in other words a discontinuity. (Note that as $h_1(t) < \phi(t)$ for $t < t_{sim}$ and $h_1(t) \approx \phi(t)$ for $t \approx t_{sim}$, h_1 will intersect ϕ at $t = t_{sim}$ if $\dot{h}_1(t_{sim}) > \dot{\phi}(t_{sim})$, and only if $\dot{h}_1(t_{sim}) \geq \dot{\phi}(t_{sim})$.) Thus, any point in parameter space that results in there being a simultaneous event of neighboring drones at any point in the dynamics can be a point of discontinuity. Figure 2.5 illustrates one of these discontinuities. The only difference in input when computing the two plots is a difference in γ_2 of 10^{-10}.

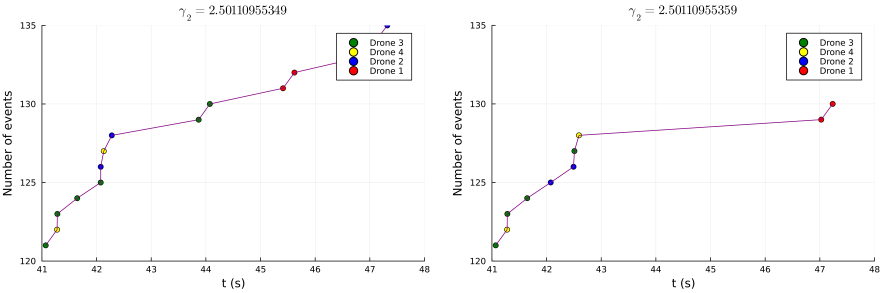

Figure 2.5 In the left plot, drones 2 and 3 (blue and green respectively) have an almost simultaneous event around $t \approx 42.1$. In this plot, drone 3 is triggering slightly before drone 2. In the right plot, γ_2 is permuted by 10^{-10}, which causes drone 2 to trigger slightly before drone 3. This changes the time to the next trigger of drone 3 by a finite amount, resulting in a discontinuity.

2.4 Convergence analysis of implementation

To test the model in a real world experiment, [1] uses a multi-threaded Euler approximation of the theoretical model, with one thread for each drone and $dt = 0.1$ s. For more details, see [1]. One question is how well this implementation corresponds to the theoretical model. To investigate this, the convergence of this implementation to the theoretical model was studied for decreasing dt. The convergence of a single-thread Euler approximation with constant time-step was also studied as a reference, see "Simulation details" in the appendix for details on the computation. To illustrate the correspondence for decreasing dt between the two approximations and the theoretical model, the time between events as a function of event number is plotted in Figure 2.6. The plot indicates that the correspondence between all 3 models improves as dt is decreased to roughly 10^{-3}. When dt is decreased further, the correspondence between the single-thread Euler approximation and the theoretical model improves greatly. This does not seem to be the case for the implementation in [1], as it seems to break down around $dt = 10^{-5}$. See the caption for a detailed description of the plot.

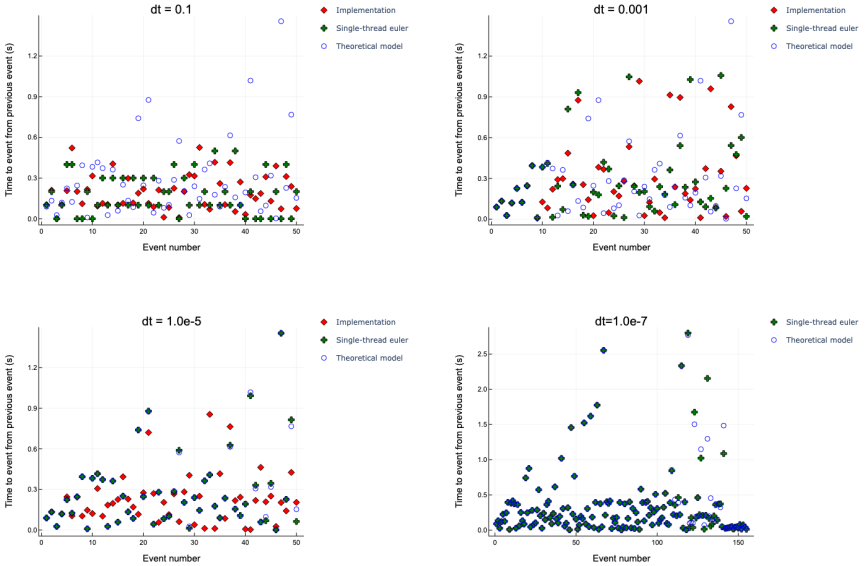

Figure 2.6 Plots of time between events for decreasing dt in the two approximations. The unit on dt is seconds. Data points coming from the implementation in [1], single-thread Euler and the method outlined in Section 2.2 are named "Implementation", "Single-thread Euler" and "Theoretical model", respectively. When going from $dt = 0.1$ to $dt = 0.001$, the correspondence between the 3 models is greatly improved for the 10 first events. However after these 10 first events, significant deviations start to occur. When going to $dt = 10^{-5}$, the correspondence between the single-thread Euler approximation and the theoretical model is greatly improved, as is expected. However this is not the case for the implementation in [1], as it seems to break down around these values of dt. For this reason, only the single-thread Euler and the theoretical model are compared in the bottom right plot. As expected, the Euler approximation corresponds even closer to the theoretical model for $dt = 10^{-7}$.

3

Optimization

3.1 Bounds

To get a well-posed optimization problem, some constraints on performance are necessary. It was decided that suitable constraints in this case were bounds on overshoot and convergence time. The overshoot for one drone was defined as the maximum deviation of the position of the drone from c after it had passed the current value of c. In other words, for a step function with 3 steps, each drone gets 3 overshoots, where each overshoot is the maximum deviation from the command after the command is passed. The overshoot of the dynamics was then defined as the maximum over the overshoots of all drones. To define convergence time, a convergence condition was defined as a function of two parameters d_c and τ_c. For a given value of c, the condition was the following: If all drones are within the distance d_c from c for time τ_c, then they have converged to c. This definition of convergence is illustrated in Figure 3.1. The convergence time for a given c was then defined as the time from the last change of c to the time the drones had converged. For a step function with 3 steps, this results in 3 times. The convergence time was taken as the maximum out of the three. If the drones didn't converge to c for any of the steps before c was changed, the convergence time was set to a large number greater than the bound on convergence time. The overshoot was computed by computing the deviation from c of all drones after each step of the iteration, which is enough information since $\dot{x}_i$ does not change sign between events. The convergence time was computed by analytically computing the intersection points of all drones with the convergence zone in every iteration step. This information was then used to check if the convergence condition was fulfilled in the taken step, and if so when it was fulfilled.

To enforce the bounds, barrier functions were used. The barrier functions were simply set to 0 inside the bounds and a large value (10^6) outside the bounds. In other words, the cost function to be minimized was defined as:

$$f(x) = Ev(x) + B_{os}(x) + B_{ct}(x), \tag{3.1}$$

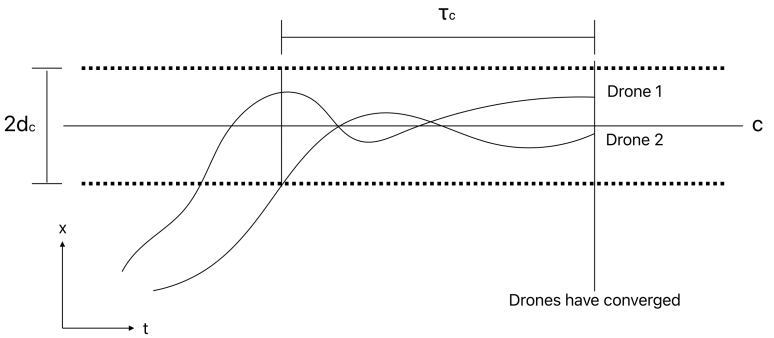

Figure 3.1 The convergence time was defined as a function of the two parameters d_c and τ_c.

where $Ev(x)$ is the number of events for parameters x and:

$$B_{os}(x) = \begin{cases} 10^6, & \text{if overshoot} \geq osb \\ 0, & \text{otherwise} \end{cases}, \tag{3.2}$$

$$B_{ct}(x) = \begin{cases} 10^6, & \text{if convergence time} \geq ctb \\ 0, & \text{otherwise} \end{cases}. \tag{3.3}$$

The overshoot bound osb was set to 0.1 m and the bound on convergence time ctb was set to 20 s. The parameters used to define convergence were set to $\tau_c = 2$ s and $d_c = 0.05$ m. Evaluation of f was done by using the iteration scheme described in Section 2.2. In Figure 3.2, a heatmap of $f(x)$ is compared with a heatmap of the number of events.

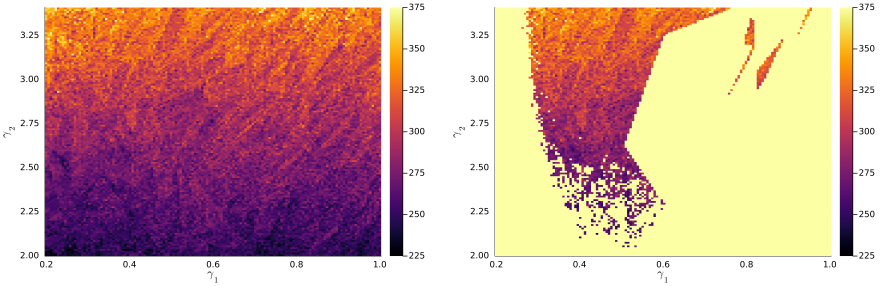

Figure 3.2 Left: Heatmap of the number of events for $\gamma_1 \in [0.2, 1]$ and $\gamma_2 \in [2, 3.4]$, keeping the other parameters fixed. Right: Same plot with bounds on overshoot and convergence time. Yellow areas are out of bounds.

When conducting the optimizations, it was found that the optimizer sometimes got stuck at certain parameter points. The reason was that the number of events tended to infinity at these points. This is not unexpected to happen, as if the time to the next event goes to 0, the number of events can go infinity. Specifically, if $|x - x_m| \to 0$, $\phi \to 0$ and $\frac{d}{dt}|\mu - \hat{\mu}| > \frac{d}{dt}(\varepsilon|x - x_m| + \phi)$, then it seems plausible that the number of events goes to infinity. To deal with these points, a maximum number of events was set to 2000, so that if the number of events was greater than this number at some parameter point, then this point was considered out of bounds.

3.2 Optimization of $f(x)$

After testing some different optimization ideas and optimization packages written for Julia, the methods which seemed best performing were a set of differential evolution methods found in the package BlackBoxOptim.jl [4].

The optimization which resulted in the best set of parameters was conducted as follows: A large search region was defined as the 6D hypercube with sides $(0.05, 5)$. For this search region, the DE method "*adaptive_de_rand_1_bin_radiuslimited*" with default settings from the BlackBoxOptim package was set to minimize $f(x)$ for 8 hours. The best parameters found in this optimization and the corresponding number of events are displayed in Table 3.1. After this optimization was performed, another optimization was conducted over a narrower search region centered at the found point using the same method. The result is displayed in Table 3.2. The dynamics of the theoretical model for the best parameters found are plotted in Figure 3.3.

Table 3.1 Optimization over a large search region using one of the DE-methods of BlackBoxOptim.

Search region	$\gamma_1 \in [0.05,5]$, $\gamma_2 \in [0.05,5]$, $\varepsilon \in [0.05,5]$, $\phi_f \in [0.05,5]$, $\phi_0 \in [0.05,5]$, $\kappa \in [0.05,5]$
Method	*adaptive_de_rand_1_bin_radiuslimited* (default settings)
Termination criterion	Optimize for 8 hours
Best point	$\gamma_1 = 0.747785$, $\gamma_2 = 2.81739$, $\varepsilon = 0.0863887$, $\phi_f = 0.289035$, $\phi_0 = 1.52821$, $\kappa = 1.43711$
Number of trigger events	108

Table 3.2 Optimization over a narrow search region roughly centered at the best point in Table 3.1 using one of the DE-methods of BlackBoxOptim. The only difference in the resulting best point is an increase in γ_2 of 10^{-4}.

Search region	$\gamma_1 \in [0.7,0.8]$, $\gamma_2 \in [2.76,2.86]$, $\varepsilon \in [0.04,0.14]$, $\phi_f \in [0.24,0.34]$, $\phi_0 \in [1.47,1.57]$, $\kappa \in [1.39,1.49]$
Method	*adaptive_de_rand_1_bin_radiuslimited* (default settings)
Termination criterion	Optimize for 500 seconds
Best point	$\gamma_1 = 0.747785$, $\gamma_2 = 2.81749$, $\varepsilon = 0.0863887$, $\phi_f = 0.289035$, $\phi_0 = 1.52821$, $\kappa = 1.43711$
Number of trigger events	102

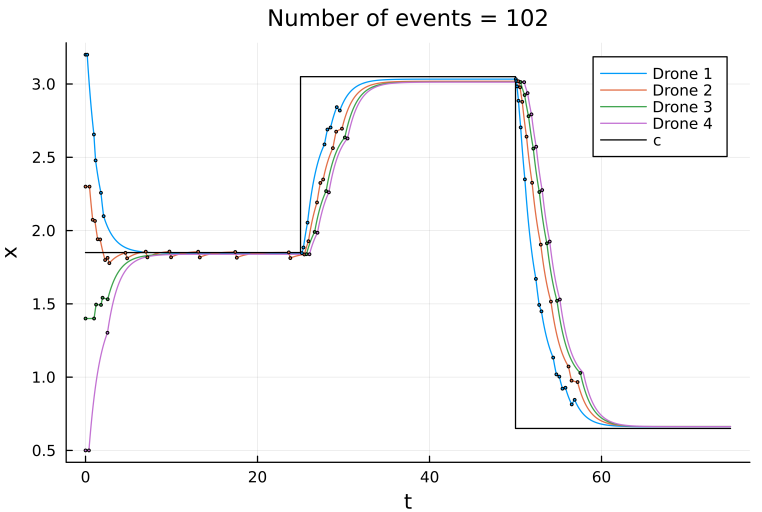

Figure 3.3 Dynamics of the theoretical model for the parameters "best point" in Table 3.2. The convergence time is 12.93s and the overshoot is 0.07233m (as defined in Section 3.1). It is worth noting that the trajectories are very similar to the trajectories in Figure 2.3, while the number of events is significantly less.

Let p_{opt} be the set of parameters denoted "best point" in Table 3.2. For the parameters p_{opt}, the number of events resulting from the single-thread Euler approximation and the implementation in [1] as a function of decreasing dt are shown in Figure 3.4. The implementation in [1] again seems to break down around $dt \approx 10^{-4}$. The single-

thread Euler approximation however results in 102 events for $dt = 10^{-7}$. The plot suggests that for these parameters and this experimental setup, if one were to use an Euler approximation to implement the theoretical model, a sampling frequency of greater than roughly 10^5 Hz might result in a good correspondence between the implementation and the theoretical model.

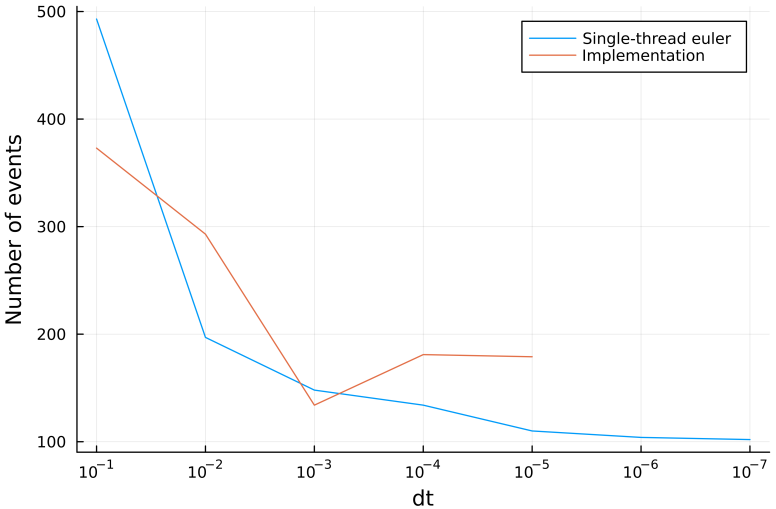

Figure 3.4 Number of events as a function of decreasing dt for the implementation in [1] and the single-thread Euler approximation for the parameters p_{opt}. For these parameters, the correspondence between the single-thread Euler approximation and the theoretical model seems to be very good for $dt \leq 10^{-5}$. The number of events for $dt = 10^{-7}$ is 102 for the single-thread Euler approximation.

It should be noted that other optimizations were performed using the DE-methods from the BlackBoxOptim package. The methods generally found a point with a number of events in the $110 - 150$ range quite quickly, say in less than 5 minutes. The points were however generally not located close to p_{opt}. This indicates that there likely exists many points which are close to optimal. There is also no reason to believe that p_{opt} is a global minimum.

4

Discussion

4.1 Sensitivity of system

While the overall dynamics of the theoretical model might be reasonably robust to changes in input, it is apparent that this is not the case for the number of events. To illustrate this sensitivity, we define a sensitivity measure at point x as the following expectation:

$$\text{Sensitivity at point x} = E(|g(x) - g(x+\Delta)|), \tag{4.1}$$

where the perturbation Δ is a random variable and g is the function we are measuring. To study the sensitivity of the number of events with respect to the parameters at p_{opt}, we let $x = p_{opt}$, $g(s) = Ev(s)$ and the perturbation distribution be a 6D multivariate normal distribution with covariance matrix $\Sigma = \sigma I$ for a scalar σ. We cut of the tails of the perturbation distribution in each dimension at 2σ to avoid complications that arise from having tails that go to infinity. For 200 evenly spaced values of σ between 10^{-6} and 10^{-2}, the expectation is estimated using $N = 500$ samples for each σ. The result is plotted in Figure 4.1, which indicates that the number of events of the theoretical model is very sensitive with respect to changes in the parameters at the point p_{opt}. This makes it so that optimizing the number of events in any real life implementation of the model might be difficult, as the number of events might not be robust to noise and other imprecisions that make the implementation deviate from the model. One interesting research topic could be to investigate where this sensitivity comes from, and if it is possible to change the model so that not only the dynamics are robust, but also the number of events. Perhaps the sensitivity is related to the fact that the model has discontinuous trigger order changes. Another consequence of the high sensitivity is that if one wants to optimize the number of events in an implementation of this theoretical model, it might be a necessary to create a close replica of the implementation to get a function to optimize which corresponds well enough with the implementation. The reason being that all the details in the implementation are important due to the high sensitivity.

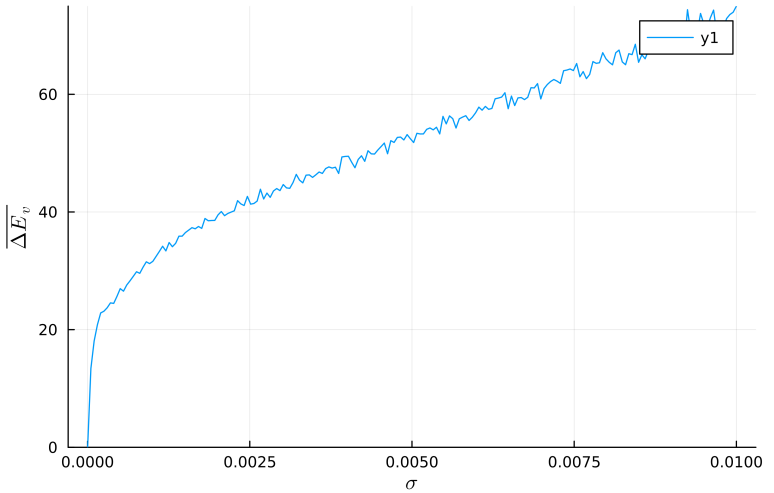

Figure 4.1 $\overline{\Delta E_v} = \sum_k |Ev(p_{opt}) - Ev(p_{opt} + \Delta_k)|/N$ where Δ_k is the k'th sample of the perturbation distribution, plotted as a function of increasing σ. It is clear from the plot that small changes in the parameters lead to large changes in the number of events.

4.2 Discontinuities

A topic which might be related to the sensitivity is the fact that the model has a high density of discontinuities, originating from trigger order changes of neighboring drones. One interesting research topic could be to investigate if it is possible to make a similar model which doesn't have these discontinuities. Maybe this would lead to a less sensitive model, which also is easier to optimize. If such a model is found, it might be possible to use a gradient based optimization.

4.3 Generalisations

There are many possible generalisations to what has been done in this project. One interesting extension could be to study the dynamics and number of events as a function of the number of drones. Maybe the sensitivity decreases as the number of drones increases. Some other extensions are to let c be a more general step function, 3 spatial dimensions instead of 1 and to let the initial conditions be arbitrary within some bounds. Regarding the optimization, one can also let the graph defined by the Laplacian L be a parameter and optimize with respect to L as well. However, it would of course be helpful if it is possible to make a model which is less sensitive and easier to optimize before trying to optimize the model.

4.4 Emphasizing robustness

A possible solution to the high sensitivity and the spiky cost landscape is to add a perturbation to the input to the model, for example the parameters, and define the cost as the expected value of f, defined in Section 3.1. In this way, we are not searching for just one very good point, but instead for a neighborhood of good points. This makes sense from many different perspectives: Practically, we want the dynamics to be robust to changes in input, as in a real world implementation there are always sources of noise. Theoretically, this should lead to a less spiky and smoother cost function, and hence the function should be easier to optimize. However, evaluating the cost function becomes more difficult, as a brute-force estimate of the expectation will require many simulations for each point in parameter space. In principle, an optimal implementation of the framework presented in this project or a similar framework should result in a very fast computation of the dynamics. Perhaps the computation can be made fast enough such that it is feasible to optimize a brute-force estimate of the expectation. If not, maybe there are other ways of estimating the expectation. In any case, the idea of optimizing an expectation could be an interesting topic for future study. [1]

4.5 Increasing simulation time

For this experimental setup, the number of trigger events as a function of the parameters has a high density of discontinuities in parameter space. An interesting observation is that this density of discontinuities likely goes to infinity if the simulation time goes to infinity. It could be interesting to study what function the number of events or number of events per unit time converges to as the simulation time goes to infinity, and perhaps optimize this function instead.

4.6 A slightly changed model

In this section, a few changes to the studied model are proposed. The proposed changes are based on intuition.

The equations for the studied model are:

$$\dot{x}_i = -\gamma_1(x_i - \hat{\mu}_i), \tag{4.2}$$

$$\dot{x}_{mi} = -\gamma_1(x_{mi} - \mu_i), \tag{4.3}$$

$$\dot{\mu}_i = -\gamma_2((\sum_{j\sim i} \hat{\mu}_i - \hat{\mu}_j) + k_i(\hat{\mu}_i - c)), \tag{4.4}$$

[1] One can imagine also minimizing the variance for example, if one wants to put more emphasis on the robustness of the solution.

Equation (4.4) can be written in matrix form as:

$$\dot{\mu} = -\gamma_2((L+K)\hat{\mu} - K\mathbf{c}). \tag{4.5}$$

The event triggering condition is:

$$|\mu_i - \hat{\mu}_i| < \varepsilon|x_i - x_{mi}| + \phi(t). \tag{4.6}$$

In this model, each drone knows its own values on the variables x, x_m, μ and $\hat{\mu}$. The variable μ can be seen as a reference trajectory approximating the Laplacian trajectory, and if we want our drones to follow this reference trajectory as closely as possible, it makes sense to swap x_m and x. This makes sense for other reasons as well, for example swapping x_m and x makes it so that $\dot{x}$ changes continuously at events instead of discontinuously. Another intuitive change is to change the differential equation for μ, i.e. (4.5), to:

$$\dot{\mu} = -\gamma_2((D+K)\mu - A\hat{\mu} - K\mathbf{c}), \tag{4.7}$$

where A is the adjacency matrix and D is diagonal with $d_i = \sum_j A_{ij}$. This makes it so that the control for each drone uses the actual μ-values of the drone, instead of the broadcasted values. Finally, it seems more intuitive to have ϕ reset at events, instead of having ϕ decay over the course of the entire simulation. With these changes, the equations would be:

$$\dot{x}_i = -\gamma_1(x_i - \mu_i), \tag{4.8}$$

$$\dot{x}_{mi} = -\gamma_1(x_{mi} - \hat{\mu}_i), \tag{4.9}$$

$$\dot{\mu} = -\gamma_2((D+K)\mu - A\hat{\mu} - K\mathbf{c}), \tag{4.10}$$

$$|\mu_i - \hat{\mu}_i| < \varepsilon|x_i - x_{mi}| + \phi(\tau), \tag{4.11}$$

where τ is time counting from the most recent event and $\phi(\tau) = (\phi_0 - \phi_f)e^{-\kappa\tau} + \phi_f$. An example of this model is shown in Figure 4.2, where the parameters are the same as in Figure 2.3 except for $\phi(\tau)$ which was set to $\phi(\tau) = \frac{1}{2}e^{-5\tau}$. The sampled version for the broadcast was used.

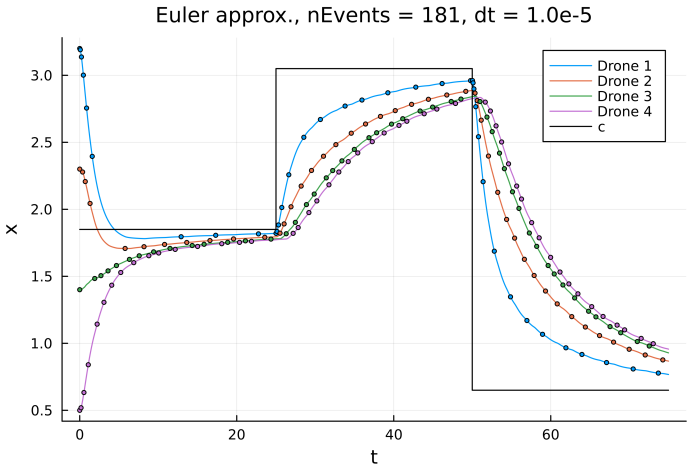

Figure 4.2 Euler approximation with $dt = 10^{-5}$ s of the slightly changed model. The values on γ_1, γ_2 and ε are the same as in Figure 2.3, and $\phi(\tau)$ was defined as $\phi(\tau) = \frac{1}{2}e^{-5\tau}$.

The convergence in Figure 4.2 is a bit slow, and to improve convergence, one idea could be to increase the value on k_i, so that $k_i > 1$ for the leader drone. This since then, the flow from the source node to the network increases. Setting $k_1 = 4$ gives the plot in Figure 4.3.

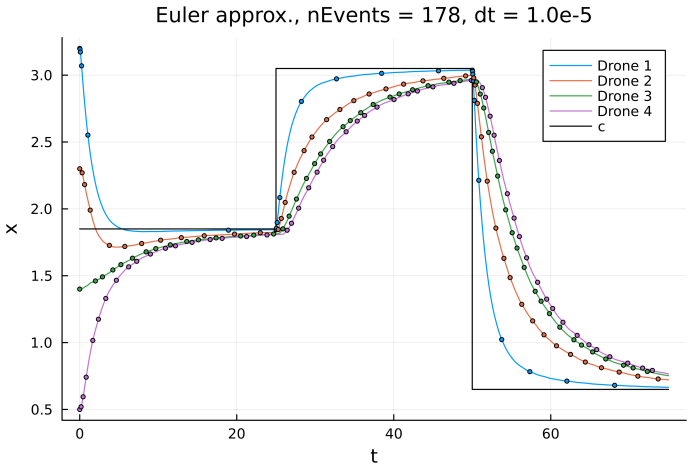

Figure 4.3 Same setup as in Figure 4.2, except for k_1 which was set to $k_1 = 4$.

4.7 A second order model

One question is: Is there a similar model which doesn't have discontinuous trigger-order changes? A possible candidate is the following, written in matrix form:

$$\ddot{x} = -\gamma_2(\dot{x} - \gamma_1(-(D+K)x + A\hat{x} + Kc)), \tag{4.12}$$

where A is the adjacency matrix, D is diagonal with $d_i = \sum_j A_{ij}$, x the positions of the agents, K is diagonal specifying the leader agents and $\hat{x}$ are broadcasted estimates of x, updated at events. The event triggering condition can have the following form:

$$|\hat{x} - x| < \phi, \tag{4.13}$$

for some function ϕ. Note that the intuitive idea with (4.12) is to give the Laplacian velocity as a target for the velocity, instead of giving a target for the position through $\dot{x} = -\gamma_1(x - \hat{\mu})$ as in the studied model. The function ϕ in the event triggering condition should be viewed as an upper bound on the error between actual x-values and broadcasted x-values, such that broadcasts occur when the error is greater than ϕ. For the broadcasts $\hat{x}$, the sampled version would be setting $\hat{x}_i = x_i$ when drone i triggers. However, a better option similar to the solution predictor curve in [1] is to set $\hat{x}_i$ to an approximation to the solution of (4.12) for drone i when drone i triggers. The estimates $\hat{x}$ then becomes functions of time, and one possibility for the broadcast is to solve (4.12) exactly for an approximation of $\hat{x}$, and broadcast this approximate solution. To investigate this, we note that regardless of what $\hat{x}$ is, the homogeneous solution to (4.12) for drone i is:

$$x_i^H(\tau) = e^{-\frac{\gamma_2 \tau}{2}}(A_1 e^{q_i \tau} + A_2 e^{-q_i \tau}), \tag{4.14}$$

where $q_i = \sqrt{(\frac{\gamma_2}{2})^2 - \gamma_1\gamma_2(d_i + k_i)}$ and A_1, A_2 are constants. The general solution has the form $x_i(\tau) = x_i^H(\tau) + x_i^P(\tau)$ for any $x_i^P(\tau)$ which satisfies (4.12). One idea is to approximate $\hat{x}$ with a Maclaurin polynomial, and solve (4.12) for this approximate $\hat{x}$. If one uses the 0'th order polynomial, i.e. $\hat{x}(\tau) = \hat{x}(0)$, a particular solution is:

$$x_i^P = \frac{(\sum_{j\sim i} \hat{x}_j(0)) + k_i c}{d_i + k_i}, \tag{4.15}$$

where $\hat{x}_j(0)$ is the value of $\hat{x}_j$ at the event. For this version of the update, drones would broadcast A_1, A_2, q and x^P at events, which is all information necessary to define the function $\hat{x}_i(\tau) = x_i^H(\tau) + x_i^P(\tau)$. Some arguments for this model are the following:

- It is event-triggered and distributed.
- Since $\dot{x}$ changes continuously at events, the derivative of the LHS of (4.13) changes continuously at events. Thus, the model should not have discontinuous trigger order changes. Hence, the number of events as a function of the parameters might be less spiky, less sensitive and easier to optimize.

Figure 4.4 depicts the movement of the agents for the same experimental setup as the one studied in this project. The plots were computed using an explicit RK4 approximation, with $dt = 10^{-4}$s. The upper bound on the error ϕ between broadcasted and actual values was set to a decaying exponential, which was reset at events (and thus not as in the model studied in the project, where ϕ was not reset at events). The parameters were chosen somewhat arbitrarily. In the left plot, the sampled version for $\hat{x}$ was used, and in the right plot $\hat{x}_i(\tau) = x_i^H(\tau) + x_i^P(\tau)$ was used, where $x_i^P(\tau)$ is given by (4.15).

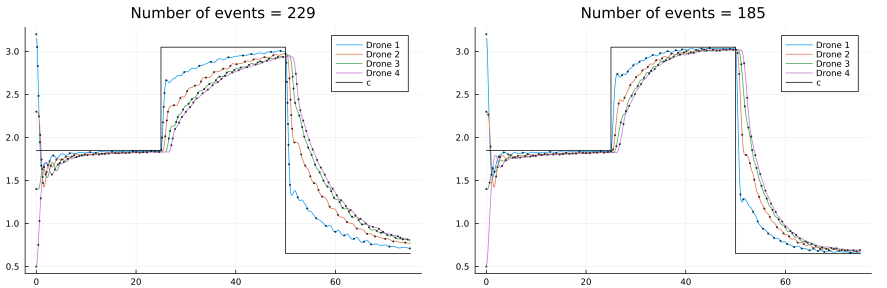

Figure 4.4 Left: Sampled version. Right: $\hat{x}_i(\tau) = x_i^H(\tau) + x_i^P(\tau)$, where $x_i^P(\tau)$ is given by (4.15) and $x_i^H(\tau)$ is given by (4.14).

5

Conclusion

The dynamics resulting from the sampled version of the control in [1] can be computed with high accuracy and efficiency by using the analytic solution to the dynamics between events and taking time steps which are lower bounds to the time to the next trigger event. The lower bounds to the time to the next trigger event can be computed by utilizing that the time to the next trigger event can be expressed analytically when ϕ is a line in terms of the Lambert W function. The number of communication events as a function of the 6 parameters $[\gamma_1, \gamma_2, \varepsilon, \phi_f, \phi_0, \kappa]$ in the sampled version of the control in [1] is highly discontinuous. The main source of the discontinuities is believed to be trigger order changes of neighboring drones. The number of communication events resulting from this control is also very sensitive with respect to changes in the parameters. Differential evolution methods seem to be suitable for finding parameters which significantly reduce the number of communication events of the sampled version of the control in [1]. The parameters $[\gamma_1, \gamma_2, \varepsilon, \phi_f, \phi_0, \kappa] = [0.747785, 2.81749, 0.0863887, 0.289035,\ 1.52821, 1.43711]$ result in 102 communication events for the sampled version, which is the lowest number of events found in this project for this particular experimental setup.

6

Appendix

6.1 Properties of the Laplacian

For this project, it is enough to only consider graphs that are connected and undirected. The definitions of these 2 properties are the following:

- Connected graph: For any two nodes A and B in the graph, there is a path from A to B, meaning that there is a set of links of the form $\{v_{Aj_1}, v_{j_1 j_2}, ..., v_{j_n B}\}$, where the v_{ij} denotes a link from node i to node j.
- Undirected: Links are two way, meaning that if there is a link from A to B, then there is a link from B to A

Let the adjacency matrix A of a graph be defined as $A_{ij} = 1$ if there is a link from node i to node j and $A_{ij} = 0$ if there is no link from node i to node j. Note that this implies that A is symmetric for an undirected graph. Now let $a_i = \sum_k A_{ik}$, and define $D = diag(a)$. Then the Laplacian L can be written as:

$$L = D - A. \tag{6.1}$$

The parameter α in Section 2.1 will be set to $\alpha = 1$ in the following proofs to lighten the notation. However, the proofs work for any $\alpha > 0$. Now for proofs of the statements in Section 2.1.

- Statement 1: For a connected and undirected graph, the dynamics defined by $\dot{x} = -Lx$ converges to a vector of ones times the mean of the initial value $x(0)$.

Proof: Since L is symmetric, the spectral theorem implies that L is diagonalized by an orthonormal basis of eigenvectors. Hence, the algebraic and geometric multiplicities of the eigenvalues of L coincide. A proof of this is as follows:

> **Proof that L symmetric implies that the algebraic and geometric multiplicities of L coincide**
>
> Let $m_g(\lambda)$ and $m_a(\lambda)$ denote the geometric multiplicity and algebraic multiplicity of eigenvalue λ, respectively. Since L is symmetric, L is diagonalizable by the spectral theorem. Thus, $m_g(\lambda) = dim(ker(L-\lambda I)) = dim(ker(S\Lambda S^T - \lambda I)) = dim(ker(\Lambda - \lambda I)) = m_a(\lambda)$, where the last equality follows from the fact that the eigenvalue matrix Λ is diagonal.

Since the algebraic and geometric multiplicities coincide for L, we refer to both as just the multiplicity of an eigenvalue. Now let $\mathbf{1}$ be a vector of ones. We note that $\mathbf{1}$ is an eigenvector of L with eigenvalue 0, since $L\mathbf{1} = 0$. A proof that the multiplicity of the eigenvalue $\lambda = 0$ is outlined below.

> **Proof that $\lambda = 0$ has multiplicity 1 for a connected and undirected Laplacian L**
>
> Assume that $Lv = 0$ for a vector v which is not parallel with $\mathbf{1}$. For such a v, there is a smallest element v_m such that $v_m \leq v_j \ \forall j \neq m$ with strict inequality for at least one j, and a largest element v_M such that $v_M \geq v_j \ \forall j \neq M$ with strict inequality for at least one j. Since the graph is connected, there is a path consisting of $n-1$ links connecting nodes m and M. Let the nodes on this path be $\{u_1, u_2, ..., u_n\}$, where $u_1 = v_m$ and $u_n = v_M$. There is a node on this path u_s such that $u_s \leq v_j$ for all neighbors of u_s and $u_s < v_d$ for at least one neighbor v_d of u_s.[1] Let a_s be the degree of node u_s, i.e. the number of neighbors of u_s. For such u_s, the derivative satisfies: $-\dot{u}_s = a_s u_s - \sum_{j\sim s} v_j \leq a_s u_s - (a_s - 1)u_s - v_d < u_s - u_s = 0$. In other words, $-\dot{u}_s < 0$ which contradicts $Lv = 0$. Hence $Lv = 0$ if and only if v is parallel with $\mathbf{1}$, which implies that the multiplicity of $\lambda = 0$ is 1.

From the Gershgorin circle theorem, it follows that all eigenvalues of L are greater than or equal to 0. Hence the matrix $-L$ has one eigenvalue equal to 0 with multiplicity 1 corresponding to the eigenvector $\mathbf{1}$, and all other eigenvalues are less than 0. From this, it follows that the dynamics defined by $\dot{x} = -Lx$ converges to $\frac{1}{N}\sum_k x_k(0)\mathbf{1}$. To see why, we use that L is diagonalized by an orthonormal basis of eigenvectors. This means that there is a matrix S whose columns are an orthonormal basis of eigenvectors of L, a diagonal matrix Λ with all eigenvalues of L on the

[1] The existence of u_s can be proven as follows: Since $u_1 < u_n$, at least one node u_j on the path from u_1 to u_n satisfies $u_j < u_{j+1}$. The first node u_j on this path which satisfies $u_j < u_{j+1}$ can be chosen as the node u_s, since $u_j = u_1$ and $u_j < u_{j+1}$.

diagonal and:

$$\frac{d}{dt}\tilde{x} = -\Lambda\tilde{x}, \tag{6.2}$$

where

$$\tilde{x} = S^T x. \tag{6.3}$$

Since the eigenvalues of L are all positive except for the eigenvalue $\lambda_0 = 0$ with multiplicity 1, all components of $\tilde{x}$ goes to 0 except for the component $\tilde{x}_{\lambda_0}$, corresponding to $\lambda_0 = 0$ and the eigenvector $\frac{1}{\sqrt{N}}\mathbf{1}$. This component is constant in time and equal to the projection of $x(0)$ on $\frac{1}{\sqrt{N}}\mathbf{1}$, i.e. $\tilde{x}_{\lambda_0} = \frac{1}{\sqrt{N}}\mathbf{1}^T x(0)$. Transforming back to the original coordinates through the equation $x = S\tilde{x}$, we get that $x(t)$ converges to $\frac{1}{N}\sum_k x_k(0)\mathbf{1}$ as $t \to \infty$.

- Statement 2: For a connected and undirected graph, the dynamics defined by $\dot{x}_i = \sum_{j\sim i}(x_j - h_j) - (x_i - h_i)$ converges to $\frac{1}{N}\sum_k (x_k(0) - h_k)\mathbf{1} + h$.

Proof:
Define $y = x - h$. From statement 1, it follows that the dynamics $\dot{y} = \frac{d}{dt}(x-h) = -L(x-h) = -Ly$ converges to $\frac{1}{N}\sum_k y_k(0)\mathbf{1} = \frac{1}{N}\sum_k (x_k(0) - h_k)\mathbf{1}$ as $t \to \infty$. Since $x = y + h$, we get that x goes to $\frac{1}{N}\sum_k (x_k(0) - h_k)\mathbf{1} + h$.

- Statement 3: For a connected and undirected graph with one or more leader agents, the dynamics defined by $\dot{x}_i = (\sum_{j\sim i} x_j - x_i) + k_i(c - x_i)$ converges to $c\mathbf{1}$, where $k_i > 0$ if i is the index of a leader drone and 0 otherwise.

Proof:
Define $\mathbf{c} = \mathbf{1}c$. Then, the dynamics can be written as $\dot{x} = -(L+K)x + K\mathbf{c}$, where $K = diag(k)$. We note that $\mathbf{c}$ is a fixed point of the dynamics, since $-(L+K)\mathbf{c} + K\mathbf{c} = -(0 + K\mathbf{c}) + K\mathbf{c} = 0$. Define $y = x - \mathbf{c}$. The dynamics for y are:

$$\dot{y} = -(L+K)y. \tag{6.4}$$

The matrix $L+K$ is positive semi-definite by the Gershgorin circle theorem. Hence, if it doesn't have the eigenvalue $\lambda = 0$, it is positive definite. What follows is a proof that $L+K$ doesn't have the eigenvalue $\lambda = 0$, using the quadratic form of the matrix. The quadratic form of L is:

$$Q = x^T L x = \sum_{j\sim 1} x_1^2 - x_1 x_j + \sum_{j\sim 2} x_2^2 - x_2 x_j + \ldots = \frac{1}{2}\sum_{v_{ij}\in E(G)} (x_i - x_j)^2, \tag{6.5}$$

where the notation $v_{ij} \in E(G)$ means that there is a link from node i to node j, so that the sum is over all links of the graph. The third equality in (6.5) follows from

that the graph is undirected.[2] For a connected graph, this quadratic form is zero if and only if $x_1 = x_2 = x_3 = ...$, i.e. $x \parallel \mathbf{1}$. One way of seeing this is by noting that $Q = 0$ implies that all nodes along any path in the graph have to have the same value, and since the graph is connected this implies that all nodes have to have the same value. The quadratic form of $L+K$ is:

$$Q_k = x^T(L+K)x = Q + \sum_i x_i^2 k_{ii}. \tag{6.6}$$

Since $\sum_i x_i^2 k_{ii} \geq 0$ and $Q \geq 0$, $Q_k = 0 \implies Q = \sum_i x_i^2 k_{ii} = 0$. As $Q = 0 \implies x = p\mathbf{1}$ for a scalar p, we get that $\sum_i x_i^2 k_{ii} = p^2 \sum_i k_{ii}$. Thus, since $k_{ii} \geq 0$ with strict inequality for at least one i, we get that $Q_k = 0$ if and only if $p = 0$, i.e. if and only if $x = 0$. Hence, $Q_k = 0 \implies x = 0$, which implies that the matrix $L+K$ does not have the eigenvalue $\lambda = 0$. Thus the matrix $-(L+K)$ is negative definite, which implies that $y \to 0$ as $t \to \infty$ for the dynamics defined by (6.4). As $y = x - \mathbf{c}$, this implies that $x \to \mathbf{c}$ as $t \to \infty$.[3]

- Statement 4: For a connected and undirected graph with one or more leader drones, the dynamics defined by $\dot{x}_i = (\sum_{j\sim i}(x_j - h_j) - (x_i - h_i)) + k_i(c - (x_i - h_i))$ converges to $\mathbf{c} + h$, where $k_i > 0$ if i is the index of a leader drone and 0 otherwise.

Proof:
Define $y = x - h$. The dynamics for y are:

$$\dot{y} = -(L+K)y + K\mathbf{c}. \tag{6.7}$$

Hence statement 3 implies that $y \to \mathbf{c}$ as $t \to \infty$. As $y = x - h$, this implies $x \to \mathbf{c} + h$ as $t \to \infty$.

6.2 Simulation specifications

This section contains specific details regarding how the computations were done. The Lambert W function was evaluated using the package LambertW.jl [6]. The error on time between events E was set to 10^{-11} s for all computations of the theoretical model. All computations use the same initial conditions, specified in Table 6.1, and the same command c, which was was defined as $c = 1.85$ for $t \in [0, 25]$, $c = 3.05$ for $t \in [25, 50]$, and $c = 0.65$ for $t > 50$. The end time t_{end} was set to 75

[2] See [5] for a detailed proof that $x^T L x = \frac{1}{2}\sum_{v_{ij}\in E(G)}(x_i - x_j)^2$. The factor $1/2$ comes from that one undirected link is in this case seen as two directed links with one link in each direction.

[3] That $-(L+K)$ is negative definite implies $y \to 0$ as $t \to \infty$ can be proven as follows: Define Lyapunov function candidate V as $V(y) = y^T y$. As $\frac{d}{dt}V = 2y^T\dot{y} = 2y^T(-(L+K))y \leq 0$, where $\frac{d}{dt}V = 0$ if and only if $y = 0$, it follows that $V \to 0$. Since $V = 0 \iff y = 0$, it follows that $y \to 0$.

(seconds) for all computations. Table 6.2 specifies the parameter values used for each plot in sections 2 and 3.

Table 6.1 Initial conditions.

$x_1(0)$	$x_2(0)$	$x_3(0)$	$x_4(0)$	$x_m(0)$, $\mu(0)$, $\hat{\mu}(0)$
3.2	2.3	1.4	0.5	$x(0)$

Table 6.2 Parameter values for each figure.

Figure	γ_1	γ_2	ε	ϕ_f	ϕ_0	κ
2.3	0.7	2.5	0.1	0.1	0.5	4/75
2.4	$\in [0.2,1]$	$\in [2,3.4]$	0.05	0.1	0.5	4/75
2.5	0.7	$2.50110955364 \pm 10^{-10}$	0.1	0.1	0.5	4/75
3.2	$\in [0.2,1]$	$\in [2,3.4]$	0.1	0.1	0.5	4/75
3.3	0.747785	2.81739	0.0863887	0.289035	1.52821	1.43711
3.4	0.747785	2.81739	0.0863887	0.289035	1.52821	1.43711

The single-thread Euler approximation was implemented as a function of dt by using the following iteration step:

- Let t be the current point of time in the iteration of the dynamics.
- Set $x(t+dt) = x(t) + \dot{x}(t)dt$, $x_m(t+dt) = x_m(t) + \dot{x}_m(t)dt$ and $\mu(t+dt) = \mu(t) + \dot{\mu}(t)dt$, where the derivatives are defined by equations 2.7, 2.8 and 2.9 respectively.
- For all i which $|\mu_i(t+dt) - \hat{\mu}_i(t)| - \varepsilon|x_i(t+dt) - x_{mi}(t+dt)| \geq \phi(t+dt)$, set $\hat{\mu}_i(t+dt) = \mu_i(t+dt)$.
- Update t to $t+dt$.
- Repeat until $t \geq t_{end}$, the end time of the dynamics.

Bibliography

[1] Stefan Ristevski, Tansel Yucelen, and Jonathan A Muse. “An event-triggered distributed control architecture for scheduling information exchange in networked multiagent systems”. In: *IEEE Transactions on Control Systems Technology* 30.3 (2021), pp. 1090–1101.

[2] Frank Lewis et al. *Cooperative Control of Multi-Agent Systems*. Springer, 2014.

[3] Thomas Dence. “A Brief Look into the Lambert W Function”. In: *Applied Mathematics* 4.6 (2013), pp. 887–892.

[4] *BlackBoxOptim*. `https://github.com/robertfeldt/BlackBoxOptim.jl/tree/master/src`. Accessed: 2023-02-03.

[5] Anne Marsden. “Eigenvalues of the Laplacian and their relationship to the connectedness of a graph”. In: *REU papers, University of Chicago* (2013), p. 6.

[6] *Evaluation of Lambert W function*. `https://github.com/jlapeyre/LambertW.jl`. Accessed: 2023-02-03.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
April 2023

Document Number
TFRT-6159

Author(s)
Anton Hässler

Supervisor
Stefan Ristevski, Combine, Sweden
Fredrik Bagge Carlson, Julia Computing, Sweden
Lara Laban, Dept. of Automatic Control, Lund University, Sweden
Kristian Soltesz, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Minimizing network utilization in event-triggered control of multi-agent system

Abstract

The exact solution to a set of equations modeling the event-triggered control of a multi-agent system is derived and computed up to a specified error on the time between events. An attempt to minimize the number of events of the solution subject to bounds on overshoot and convergence time is made using a differential evolution method. Parameter points which result in a low number of events of the bounded solution are found.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-43

Recipient's notes

Security classification

http://www.control.lth.se/publications/