Optimizing Settling Time for Swarm UAV Formation Control In the Presence of Obstacles

Raj Shah, M.S. Student (Mechanical)
Supreet Kurdekar, M.S. Student (Mechanical)
Chris Gnam, Ph.D. Student (Aerospace)
Department of Mechanical and Aerospace Engineering University at Buffalo Buffalo, New York, 14261

Abstract

UAVs are used in a wide range of applications such as land surveys, search and rescue operations, remote sensing, disaster monitoring to and security applications. A group of UAVs greatly enhances the performance and robustness in fulfilling difficult tasks. Multiple agents increase the chances of completing a mission as other agents can continue the mission even if one or more agents fail. Many of these applications require maintaining a desired formation. Therefore it is essential that any deviations from formation keeping (such as obstacle avoidance) be kept brief. We present a method for optimizing settling time in formation control of UAV swarm navigation in the event of obstacle avoidance. Decreased settling time in response to obstacle avoidance means less interruption to nominal operations. This can lead to increased performance or increased quality of data collection, depending on the objective of the drone swarm.

1 Introduction

Formation flying can allow for more robust application of UAVs to a wide variety of applications. When placed in real world environments, these swarms need to be able to react to obstacles. During an avoidance maneuver, the formation of the drones is broken, which for many applications means that the system is no longer able to conduct the work it has set out to do. As such, it is highly desirable to return to formation as quickly as possible after evading an obstacle.

We will be looking primarily at a four agent square formation, that comes across a cylindrical obstacle (analogous to a tree, pole,or many other real-world obstacles). Our goal is not to design a new control scheme, but rather to optimize existing methodologies. As such we reviewed standard approaches for controlling drones flying in formation.

Reynolds [1] showed decentralized flock motion scheme is implented to simulate the motion of an entire flock. This is accomplished by having each agent follow three simple rules.

1. **Avoidance:** avoid collisions with nearby flockmates.
2. **Velocity Matching:** attempt to match velocity with nearby flockmates.
3. **Flock Centering:** attempt to stay close to nearby flockmates.

Tanner[2] introduces a simple methodology for stable flocking configurations, utilizing potential functions for mediating inter-agent formation keeping maneuvers. While this

method proved powerful for both acquiring and maintaining a stable flocking formation, it did not consider obstacles in its formulation.

Leonard and Fiorelli[3], introduce the concept of a virtual leader. A virtual leader is an artificial point that maintains formation shape by attracting individual agents towards itself. This also proves advantageous as the motion of the entire swarm can be affected by controlling the virtual leader.

Finally, Han [4] demonstrates the use of potential functions for mediating obstacle avoidance. A repulsive potential is created around the obstacle, while potentials around each agent are used to mimic a spring like force to obtain a desired distance. Typically, each agent has a potential applied to it and the control input for that agent uses a steepest gradient policy so as to minimize the potential function value. Minimizing this potential function results in the agents both avoiding each other and obstacles, while also achieving the desired formation.

For the purposes of this project, we will be simulating 4 agents in 2-d space, capable of controlling their position and velocity via applied accelerations. The agents will be following a virtual leader, which itself is following a predetermined trajectory through the environment. The environment will have a single circular obstacle that the agents must avoid, while still following the virtual leader. The virtual leader will pass straight through this obstacle, and the agents will have to react (potentially breaking formation), and then return to formation as quickly as possible, as shown in Fig. 1.

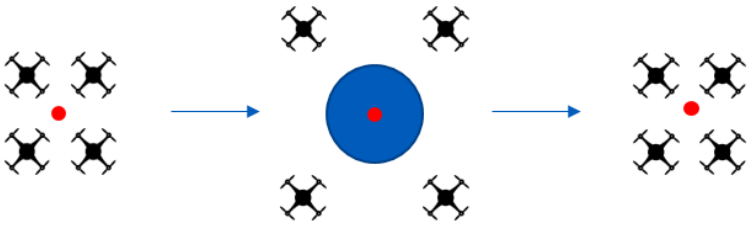


Figure 1: Virtual leader shown in red, and drones shown in formation around it. As the virtual leader passes through the obstacle, the drones can be seen splitting apart to avoid the obstacle before coming back together once they have passed the obstacle.

The flock formation will be parameterized by a *formation vector*, which is composed of all the inter-agent and agent-virtual leader relative velocities. Once the sum of the magnitude of all of these values are equal to zero (within some tolerance), then the system is said to be settled. We will also be assuming perfect state knowledge of all the agents, and perfect knowledge of the environment. There will be no delay or limitations on sensor speed. However, actuation authority and velocity limitations will be applied so as to accurately

capture the dynamics.

In order to accomplish this, we will be defining an objective function for the settling time, with respect to the control gains and the reaction distance from the obstacle. The first constraint is that the vehicles must not collide with either each other or with the obstacle. Secondly, the vehicles will have a maximum allowable acceleration (applied actuation), and will have a maximum allowable traveling velocity.

2 Methodology

Our optimization approach can be broken into two main components: The simulation, and the optimization itself. The simulation, although simplified, aims to accurately capture the dynamics of a swarm of drones. The simulation was then used as a penalized cost function for the optimization routines. Its inputs were the design variables, and the output was the settling time (which was to be minimized). If undesired behavior was observed in the simulation (such as two drones colliding, or a drone collided with the obstacle), it returned a large settling time in order to penalize the optimizer.

2.1 Simulation Overview

The dynamics were treated as a simple two-day planar case of drone dynamics. We ignored drag and other perturbing forces, and so utilized the assumption that the only force acting on the drones was their own thrust. The dynamics were propagated using a 4th order fixed step size runge-kutta numerical integrator and the following state space model:

$$d\vec{X}_i = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{r}_i \\ \vec{v}_i \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \vec{u}_i = \begin{bmatrix} \vec{v}_i \\ \vec{u}_i \end{bmatrix} \tag{1}$$

where $\vec{v}_i$ is the velocity vector of the i^{th} drone, and $\vec{u}_i$ is the control input of the i^{th} drone.

2.1.1 Constraints on Dynamics

We have limited the maximum speed and maximum acceleration of the vehicles to be consistent with real world drone dynamics. In addition, we have given the drones a size, and are not treating them as simple point particles. This enables to us to check if drones have collided with one another, and if they have, the objective function penalizes the optimizer.

2.1.2 Controller Design

The controller is run for each drone independently, and utilizes five separate error states, along with their associated gains. The control acceleration input is given as:

$$u = k_{r_{ia}} e_{r_{ia}} - k_{v_{ia}} e_{v_{ia}} - k_{r_{vl}} e_{r_{vl}} - k_{v_{vl}} e_{v_{vl}} + k_{r_{obs}} e_{r_{obs}} \tag{2}$$

The five error states are then given by:

$$e_{r_{ia}} = \sum_j \frac{\vec{r} - \vec{r}_j}{||\vec{r} - \vec{r}_j||} \tag{3}$$

$$e_{v_{ia}} = \sum_j \vec{v} - \vec{v}_j \tag{4}$$

$$e_{r_{vl}} = \frac{\vec{r} - \vec{r}_{vl}}{||\vec{r} - \vec{r}_{vl}||} \left(||\vec{r} - \vec{r}_{vl}|| - d_0\right) \tag{5}$$

$$e_{v_{vl}} = \vec{v} - \vec{v}_{vl} \tag{6}$$

$$e_{r_{obs}} = \frac{\vec{r} - \vec{r}_{obs}}{||\vec{r} - \vec{r}_{obs}||} \frac{d_{react}}{||\vec{r} - \vec{r}_{obs}||^4} \tag{7}$$

where $\vec{r}$ and $\vec{v}$ are the position and velocity of the drone (with $\vec{r}_j$ and $\vec{v}_j$ indicate the position and velocity of another drone), d_0 is the desired distance from the virtual leader, $\vec{r}_{vl}$ and $\vec{v}_{vl}$ are the position and velocity of the virtual leader, $\vec{r}_{obs}$ is the position of the obstacle, and d_{react} is the reaction distance from the obstacle.

The control input can be thought of as modeling fictitious forces between the drone, virtual leader, and the obstacle. This is shown more clearly in Fig. 3. These forces can be thought as following the gradient of a given potential function, which follows Hooke's law ($F = kx$). The only one which deviates from this is the interaction between the drones and the obstacle, which is outlined in Eq. (7), where a power of 4 is introduced. The purpose of this

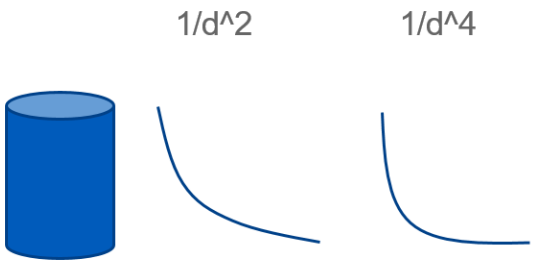


Figure 2: Potential field comparison between $1/r$ and $1/r^4$

In addition, a sliding mode type control is utilized, where the drones have a different set of gains when in formation, as opposed to when performing obstacle avoidance. Therefore, we have two sets of gains: one for the *cruise mode*, and another for the *obstacle avoidance mode*.

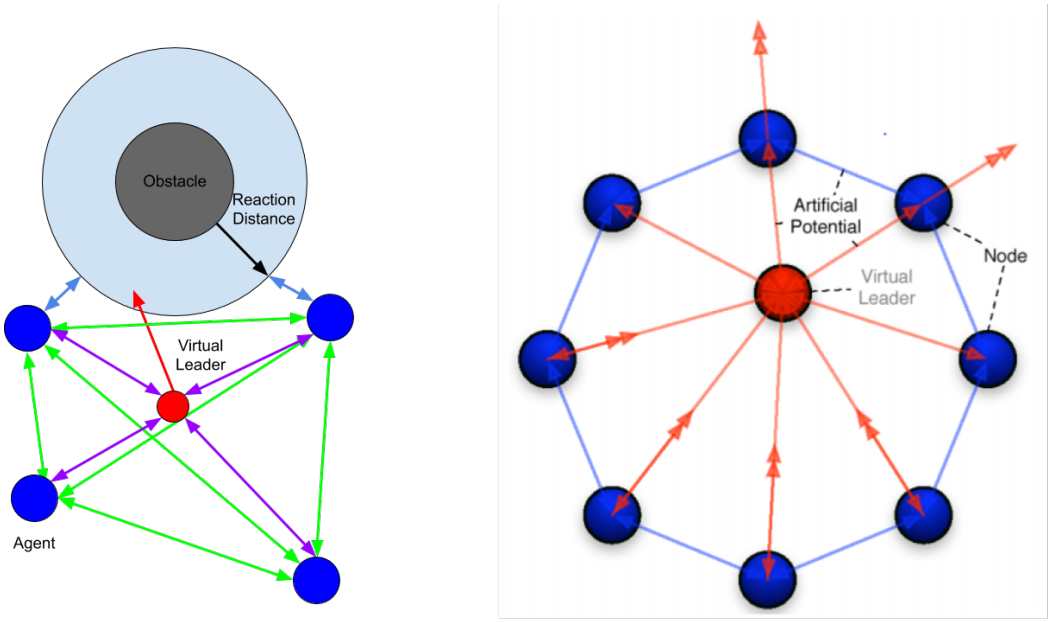


Figure 3: Diagram of problem interactions. **Left:** All interaction forces which define the control input. Blue forces are agent-obstacle interactions, purple are agent-virtual leader interactions, and green are inter-agent interactions. **Right:** Shows more clearly how the interaction forces are modeled as spring forces.[5]

2.2 Settled State

The settled state is taken to be when the drones are no longer moving with respect to one another or with respect to the virtual leader. To calculate this, the normalized relative velocities between the drones and the virtual leaders were calculated, and then added up. Because is value will never truly equal zero, we selected a total tolerance of 0.05 m/s. That is, if the sum of all of the normalized errors was less than the tolerance, after the vehicles had entered the obstacle avoidance maneuver, the formation was assumed to be settled. We selected this number heuristically after observing several runs and identifying when the formation seemed reasonably settled.

2.3 Optimization

2.3.1 Design Variables

The design variables we use are the five gains given in Eq. (2), as well as the *reaction distance*, d_{react}. d_{react} is the distance from the obstacles at which the control gains switch from the cruising mode to the obstacle avoidance mode.

It was noticed through testing that these design variables, as they are defined above as simple scalars, span several orders of magnitude. As such, a scaling factor for each gain was added to ensure that the optimizer was treating each appropriately. We found heuristically a reasonable scaling to be applied was as follows:

$$\frac{k_{r_{ia}}}{10} \quad \frac{k_{v_{ia}}}{100} \quad \frac{k_{r_{vl}}}{10}$$

2.4 Constraints

Analytical derivations of the constraints were attempted, but due to the highly stochastic transient nature of the problem, they grew to be far too complicated and did not end up working for our particular problem. Instead, since all of constraints were on the drone dynamics and interactions (such as avoiding collisions, and having maximum allowable velocities and accelerations), we elected to embed all of the constraints in our actual simulation, as a *penalized cost function*. If any of these transient constraints were violated, the cost function would return an extremely high settling time to indicate an in feasible solution, which would penalize the optimizer. This was found to be extremely successful in all of our optimization tests.

2.5 Interior Point Algorithm Approach

Our primary method of optimization was using the interior point algorithm, as implemented by MATLAB's built in `fmincon` function. The simulation outlined above as used as the objective function, with no constraints explicitly given to the optimizing routine. Due to the simulation only taking a second at most to evaluate for a given set of gains, this optimization routine would take only a minute or two to produce results.

2.6 Genetic Algorithm Approach

Due to our objective function being highly multimodal, we also attempted to use a Genetic Algorithm (GA) as implemented in MATLAB to find a superior minimum. This approach took significantly longer, with the `GA` taking around two hours to converge, and its converged value matched closely with that found by `fmincon`.

3 Results and Discussion

3.1 Performance of Optimized Gains

Both optimization methods yielded us with similar performance however they both produced wildly different gains despite starting with the same initial conditions. Fig. 4 shows a comparison between the total error between the initial gains (which were selected at random), and the gains optimized for by each method outlined previously. The interior point solution settled in exactly 34 seconds, while the GA solution settled in 34.7 seconds. Both of these were a steep improvement over the randomly selected gains, which had a settling time over 140 seconds.

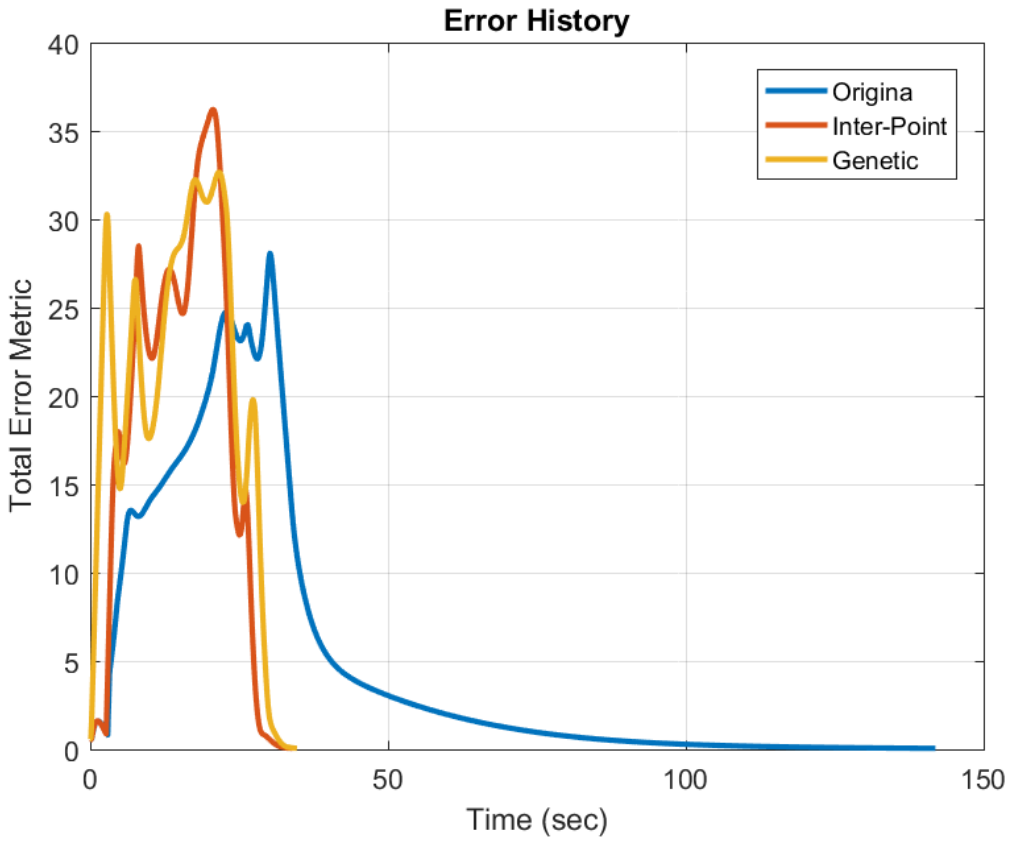
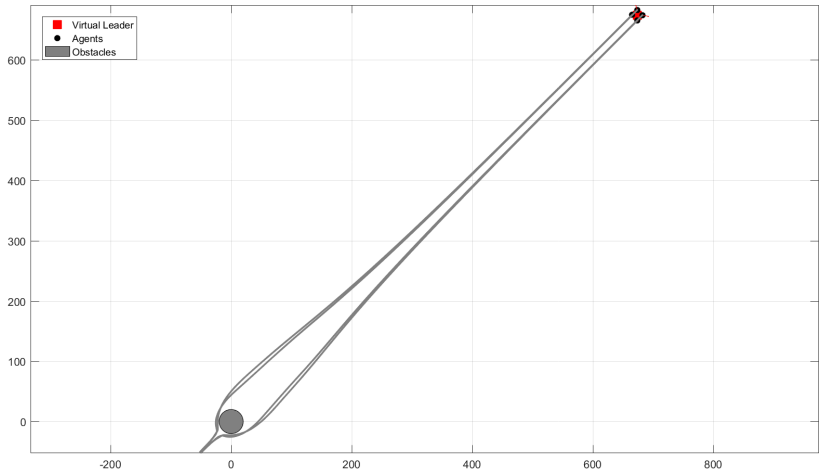
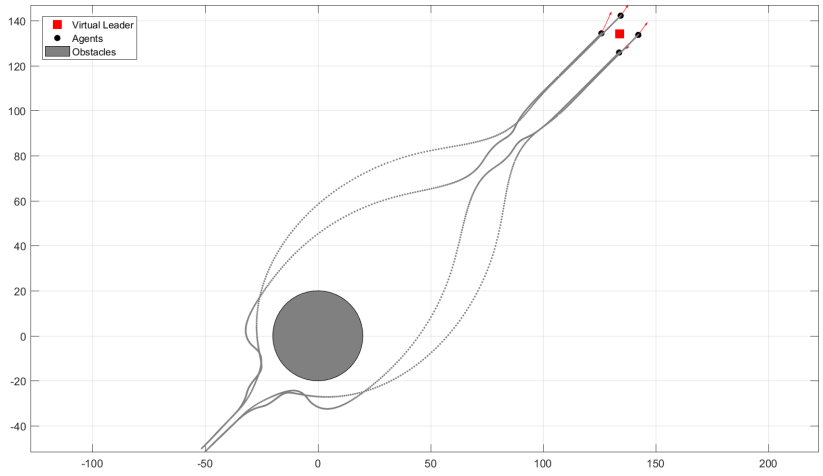


Figure 4: Time history comparison between the total calculated error between original gains and optimized gains

A comparison of the drone trajectories through space are shown below in Fig. 5. It is clear that the drones using the un-optimized gains are much slower to converge. In addition, it is also clear that the drone using the optimized gains maneuver more aggressively around the target, and converge back to formation much more quickly.



(a) Drone trajectories using un-optimized gains



(b) Drone trajectories using gains optimized via interior point algorithm

Figure 5:

3.2 Interior Point Algorithm Results

Variable	Initial	Optimized
$k_{r_{ia}}$	20	24.0777
$k_{v_{ia}}$	30	0.0021
$k_{r_{vl}}$	50	66.2946
$k_{v_{vl}}$	20	16.8299
$k_{r_{obs}}$	30	22.0434
d_{react}	50	50.8013

Table 1: Optimized Gains and Reaction Distance using `fmincon`

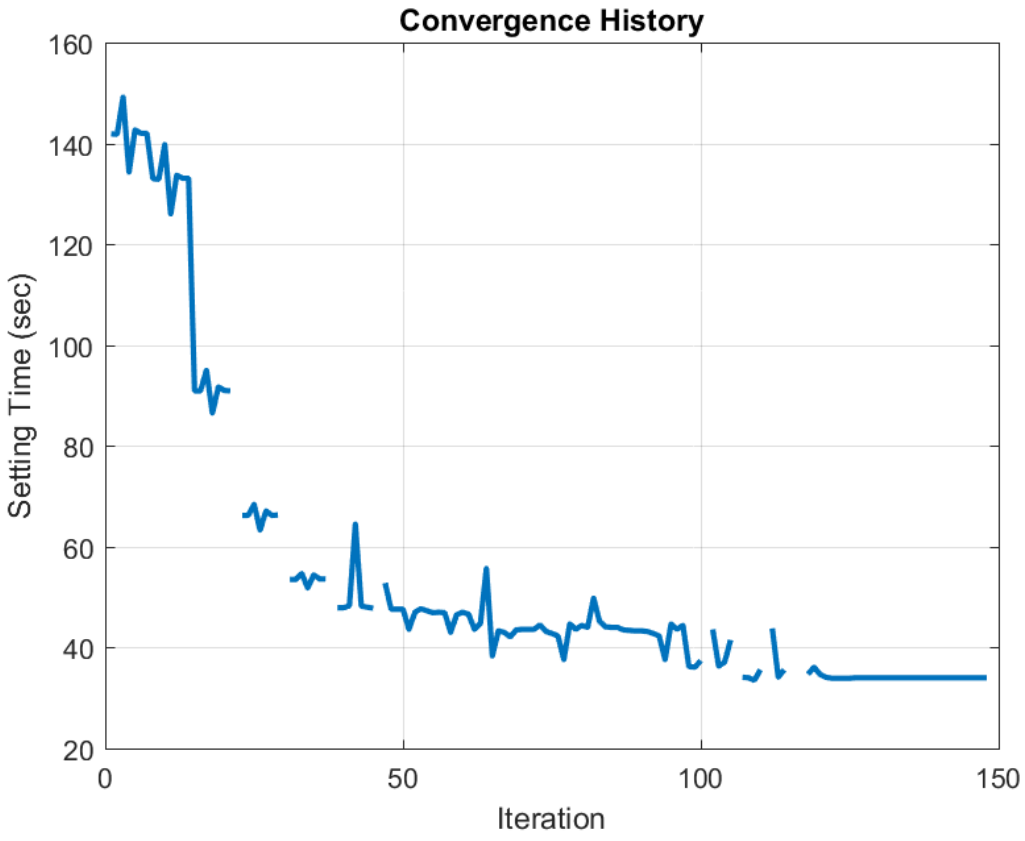


Figure 6: Convergence history for fmincon optimization

3.3 Genetic Algorithm Results

Variable	Initial	Optimized
$k_{r_{ia}}$	20	91.9202
$k_{v_{ia}}$	30	1.5019
$k_{r_{vl}}$	50	82.4238
$k_{v_{vl}}$	20	14.4036
$k_{r_{obs}}$	30	34.2422
d_{react}	50	409.6430

Table 2: Optimized Gains and Reaction Distance using `fmincon`

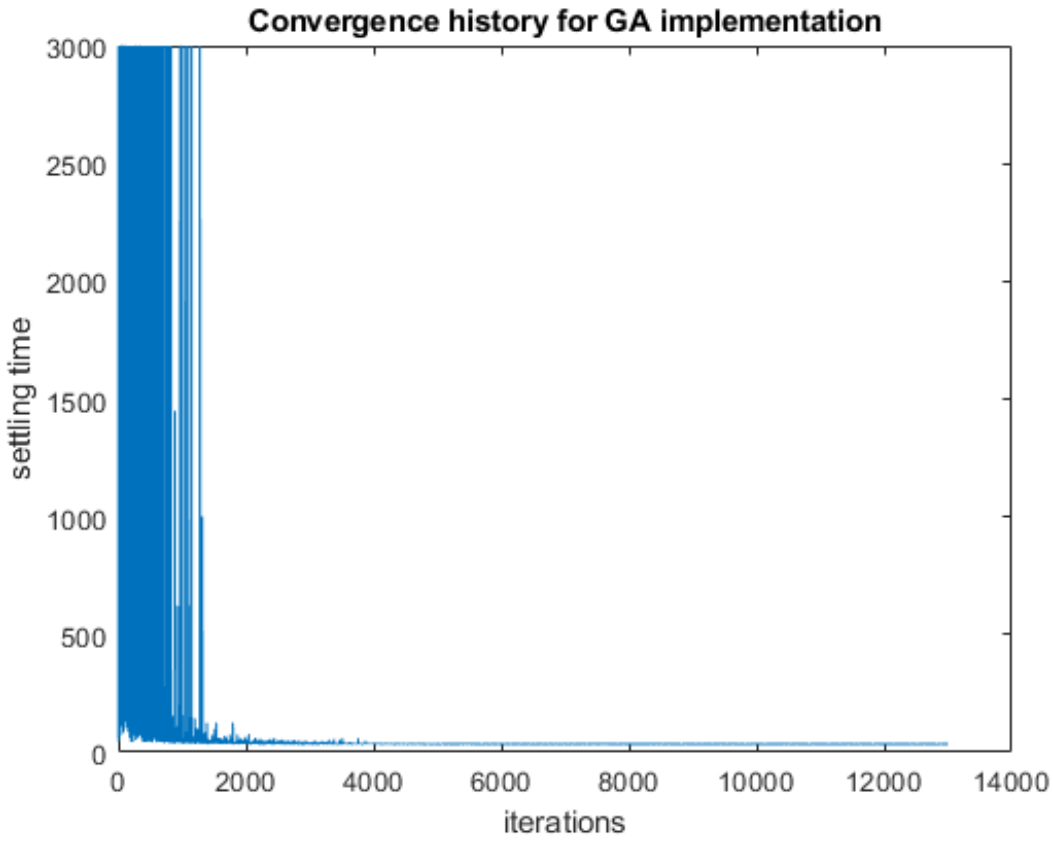


Figure 7: Convergence history for genetic algorithm (GA) approach

4 Conclusions

We drew several conclusions from this work. First and foremost, we learned thaht UAV swarm navigation based on a virtual leader and artificial potential field architecture can yield powerful results. These rather simple control rules yield to fantastic emergent swarming behavior, but tuning the gains for this type of system can be quite complicated. Optimization using standard optimization techniques can be a simple yet powerful way to tune these gains effectively.

We also discovered that while the genetic algorithm does a great job at working with highly multi-modal functions, it can be much more computationally expensive, and is clearly not guaranteed to converge to the global minimum. For this particular problem, it showed no benefits over the interior point algorithm.

In addition, for both of these optimization methods, this particular problem made formulating constraints on the optimizer quite difficult. However, many of these constraints (which exist on the transient behavior of the agents) lend themselves quite nicely to a penalized objective approach. By having the objective function penalize infeasible solutions, it not only simplified the problem formulation, but also yielded great results that we could be sure were physically consistent.

Finally, from a numerical perspective, we discovered that both the simulation step size, and the optimization step size play very important roles in the optimization of this type of problem. The simulation step size needed to be small enough to fully capture the behavior of the dynamics, and detect any potential collisions, but could not be too small which would increase the computation size. In addition, the optimization step size needed to be increased in order to ensure that the gains were moved around enough so that the local gradient of the objective could be identified.

Member Contributions

All three members were heavily involved in conducting literature review, and developing the dynamics and underlying models for all of the vehicles. All group members also contributed to the generation of results, as well as this report as well. Specific contributions for each group member are outlined below.

Supreet Kurdekar

Supreet focused on developing settling time criterion. He also aided in identifying the need for the step size increase, and scaling our design variables to get improved performance. He also heavily reviewed code written by the other group mates to ensure things were logically consistent and found several errors with our simulation model that saved us countless time. Finally, he also put together the first draft of our project abstract and presentation.

Raj Shah

Raj worked on initial code for the settling time calculation, as well as on exploring the potential use of a surrogate model. Our initial efforts to write the simulation were extremely slow to run and so Raj implemented a surrogate modeling technique using RBF. He discovered limitations in the surrogate modeling approach which, combined with improved vectorization of our simulation code, pushed our group in a different direction. Raj also implemented and ran the Genetic Algorithm (GA) approach we explored later in the project. Finally, he also put together the final draft of our presentation, and generated several figures and sections for the final report.

Chris Gnam

Chris primarily focused on the development of the simulation architecture and maintaining all of the code on GitHub. He also worked on vectorizing all of the code used to ensure fast runtime. In addition, he implemented several of the papers which were reviewed to get a better understanding of the problem. He also was primarily responsible for putting together the final formatting of the project report.

References

[1] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34, August 1987. ISSN 0097-8930. doi: 10.1145/37402.37406. URL `http://doi.acm.org/10.1145/37402.37406`.

[2] Herbert Tanner, A. Jadbabaie, and George Pappas. Stable flocking of mobile agents, part i: Fixed topology. volume 2, pages 2010 – 2015 Vol.2, 01 2004. doi: 10.1109/CDC.2003.1272910.

[3] N. E. Leonard and E. Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. In *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, volume 3, pages 2968–2973 vol.3, Dec 2001. doi: 10.1109/CDC.2001.980728.

[4] Ki Han, J Lee, and Yongsok Kim. Unmanned aerial vehicle swarm control using potential functions and sliding mode control. *Proceedings of The Institution of Mechanical Engineers Part G-journal of Aerospace Engineering - PROC INST MECH ENG G-J A E*, 222:721–730, 09 2008. doi: 10.1243/09544100JAERO352.

[5] G. H. Elkaim and R. J. Kelbley. A lightweight formation control methodology for a swarm of non-holonomic vehicles. In *2006 IEEE Aerospace Conference*, pages 8 pp.–, March 2006. doi: 10.1109/AERO.2006.1655803.