

Fast aircraft separation calculations for gradient based optimization of airspace simulations

Tristan A. Hearn*

NASA Glenn Research Center, Cleveland, OH, 44135

Simulations of airspace operational concepts can play a significant role in determining future paradigms that would allow for a safe increase in airspace density. In particular, airspace simulations which are capable of handling large numbers of aircraft act as an enabling capability for the testing of proposed airspace operational concepts. Simulations allowing for gradient based optimization methods are particularly attractive, since they would potentially allow for an efficient and empirical means to derive best operational practices. These could also allow for vehicle multidisciplinary design and optimization studies to include air traffic management considerations as a discipline.

But any large scale simulation of airspace operations must include some methodology for addressing airspace separation requirements, which in the most direct sense would be tracked in a manner that computationally grows as a quadratic function of the number of simulated aircraft. Efficient indirect methods have been developed in certain contexts to address this limitation. However, any means of addressing separation requirements in a gradient based optimization context should be implemented by functions which provide analytic derivative information to maximize numerical precision and computational efficiency.

In this paper, a fast and differentiable separation metric is described in application to gradient based optimization of airspace operations. Rather than computing the separation distance between every pair of aircraft in a simulation, this method effectively reduces the problem to a smaller relevant set using a geometric decomposition. This method guarantees that the smallest distance at all points in simulated time is determined exactly. When used in an optimization constraint context, this guarantees that a required separation is maintained between all pairs of aircraft.

The presented metric has logarithmic computational growth with respect to the number of simulated aircraft, and is shown to perform well in a series of notional 2D airspace optimization problems when used to enforce specified airborne separation constraints. Results show that this is notably faster than a direct pairwise distance computing metric for optimizations involving both small and large numbers of aircraft, yet enforces separation requirements to the same tolerance. It is shown that this favorable scalability is an enabling capability for more sophisticated air traffic management conceptual studies.

I. Introduction

The demand for air transportation is expected to substantially increase in the coming decades[1]. This increase is due to many factors, including the introduction of a number of new airborne transportation services, such as On-Demand Mobility (ODM), and from Unmanned Aircraft Systems (UAS)[2]. To cope with an expected increase of demand, future airspaces must be able to handle an increase in vehicle density. This will require the development and deployment of completely new paradigms for air traffic management and control services[3, 4].

A. Airspace simulation and numerical optimization

The evolution of increasingly complex airspace operational concepts require the means to rigorously evaluate them across a wide variety of contexts. Thus, simulations and numerical models of airspace operations have a potentially vital role to play in determining CONOPS (Concepts Of Operations) for the future NAS (National Aerospace System). By creating models that can incorporate (for instance) different airspace control mechanisms, or include levels of control

*Aerospace Engineer, Propulsion Systems Analysis Branch, 21000 Brookpark Rd MS 5-11.

centrality, these methods can be quantitatively vetted for effectiveness at maintaining flight requirements and robustness to uncertainty in the environment.

One key flight requirement is for aircraft to maintain a defined amount of separation between themselves and other aircraft, an amount prescribed by aviation authorities. Maintaining this separation while airborne minimizes the risk of a collision between aircraft operating within the same airspace. When two aircraft actually are closer to each other than specified in the defined separation criteria, this is called a "loss of separation" (LOS). Separation requirements are typically designated by a required amount of lateral, vertical, and temporal distance. Preventing and mitigating LOS events is the primary purpose of air traffic control systems, and thus the ability to recognize and address airborne separation must be a core consideration of any multi-vehicle airspace management simulation.

There are a variety of types of airspace models and simulations that have been used in past and in ongoing ATM research. For instance, human-in-the-loop studies have involved the integration of multiple aircraft simulators in real time[5, 6]. While these studies are high fidelity at the aircraft performance level and capable of evaluating mixed human-machine performance, they unfortunately do not conceptually scale to large numbers of simultaneously controlled aircraft, or to large numbers of repeatable iterations.

Other types of simulation frameworks operate directly with numerical representations of aircraft equations of motion, which can be integrated in simulated time steps. These have no human directly in the loop of the simulation, and can simulate long duration air traffic scenarios quickly[7, 8]. These typically involve implementing parameterized trajectories which are determined through a fixed set of rules representing a particular air traffic control scenario. This allows for inclusion of large numbers of simulated vehicle "agents". These simulated scenarios are run to completion, and statistics related to metrics of interest (such as loss of airborne separation) can then be computed, averaged over large numbers of simulations using methods such as Monte Carlo sampling. While this can be very effective for evaluating a particular control strategy or a fixed set of ATM concepts, it may be difficult to use such an approach to point towards specific alternatives to the ones that are implemented, or even to refine a concept (or set of integrated concepts) for better performance. If there are a large amount of degrees of freedom to a problem (e.g., a large set of parameters relating to control concepts such as deconfliction strategies or control delegation), it may also be difficult to address determining an optimal strategy through a sampling strategy.

Large scale Multidisciplinary Design Optimization (MDO) methods provide a potential means to address this[9, 10]. MDO methods provide a generative means for determining a "most favorable" configuration for a particular model, that scales to numerical problems with large numbers of design variables and constraints. Air traffic management studies with large numbers of agents and parameterized control rules certainly fit this description. Following well-established methodology for trajectory level optimization[11], recent developments in the field of MDO research have demonstrated the ability to couple aircraft trajectory analysis to a variety of other disciplines in a large scale optimization context[12]. These include the coupled optimization of vehicle trajectories to aeroacoustic modeling of noise [13], thermal modeling of electrical and fueled propulsion systems [14, 15], and other vehicle design level considerations. Joint multidisciplinary optimization studies such as these are invaluable at providing information about possible improvements to system level performance that are can be realized when considering the interaction of multiple subsystems in a holistic manner.

Thus, in a similar manner to integrating trajectory operation with vehicle level design, it is desirable to apply MDO methods to study and refine concepts for future air traffic management. A simple example concept (roughly matching a centralized ATM paradigm) would be to simultaneously minimize the collective flight time (or perhaps net collective energy consumption) of a set of simulated aircraft trajectories according to their respective vehicle equations of motion, subject to a set of numerical constraints relating to operational requirements (such as airborne vehicle separation and fleet scheduling).

A simple single-integrator type optimization problem that drives the simulation model this way has the form given in Table 1. In this case, the positional vectors x and y of all aircraft are integrated over simulated time with respect to their velocity vectors $\dot{x}$ and $\dot{y}$, which operate as simplified controls. Representing the system of equations using the collocation method, the state of the model is then converged simultaneously (rather than sequentially) for all vehicles over all points in discretized time, in order to efficiently optimize the collective flight time of all vehicles[11, 12].

Solving large air traffic management problems in an optimization context introduces practical considerations that must be taken into account, since different optimization methods generally come with their own set of requirements of the models that they are applied to. Numerical optimization algorithms can generally be classified into two categories, gradient-based and gradient-free. Gradient-based optimization methods use mathematical derivatives of a model to iteratively perturb parameter values in a direction which improves the value of an objective function until a terminating criteria is reached (typically based on the KKT optimality conditions). This contrasts with gradient-free methods, which may use a variety of sampling and local search strategies to determine optimal parameter values for a model[16]. There

Table 1 An example minimum-time formulation for an airspace operation optimization.

	Variable/Function	Description
minimize	$\mathcal{T}$	Net flight time for all vehicles
with respect to	$x, y, \dot{x}, \dot{y}$	ODE states
subject to	$\mathcal{G}_1 = 0$	Collocation converged for ODE solution
	$\mathcal{G}_2 = 0$	All aircraft fly from their origins to their destinations
	$\mathcal{G}_3 < 0$	Aircraft separation maintained at all times

are practical advantages and disadvantages to both types of optimization algorithms. Broadly speaking, gradient-based optimization, though it is susceptible to convergence onto local optima, uses far less function evaluations compared to gradient-free methods[17].

For large-scale computationally-expensive models with a large number of design variables (such as multi-agent trajectory-based airspace models), this is a significant advantage, particularly when combined with adjoint methods. As the name implies, gradient-based optimization algorithms rely on gradient information to achieve their reduced computational cost. Finite-difference approximation methods are often used to compute derivatives, but this method scales poorly as the number of design variables increases. Finite-difference approximations are also sensitive to the step size used to estimate them, especially when a model has one or more internal solvers converging a nonlinear system of equations. For these reasons, tools that provide analytic derivatives in addition to their analysis outputs are particularly suitable for efficient optimization with gradient-based methods.

The choice of problem statement (quantities to be optimized, the parameters to be considered, or the constraints to be satisfied) can also have important structural implications on the scalability of an optimization problem. In particular, determining and satisfying airborne separation requirements via a non-linear optimization constraint can pose a scalability challenge that must be addressed. Even in a non-optimization context, robustly checking separation between large numbers of simulated agents is computationally challenging.

B. Computing vehicle agent separation

Computing the quantities of interest that verify separation requirements in a simulation involves determining if any aircraft violate the separation constraint at any point in discretized time. The most direct means of doing this is by computing the distances (e.g. lateral, vertical, temporal, or any other form of distance specified) between all pairs of aircraft across time.

For example, consider a 2D airspace model containing n aircraft. x_i and y_i are the positional vectors for vehicle i in an 2D airspace model. Each pair of aircraft can then have their lateral separation determined by the Euclidean metric

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2},$$

for each point in discretized time t in the transcription of the ODE describing the vehicle flight models. As required by gradient based optimization algorithms, the derivatives of $d_{i,j}$ with respect to the position vectors can then be computed as

$$\frac{\partial d_{i,j}}{\partial x} = \frac{2(x_i - x_j)}{d_{i,j}}$$

and

$$\frac{\partial d_{i,j}}{\partial y} = \frac{2(y_i - y_j)}{d_{i,j}}.$$

Thus, enforcing an airborne separation requirement in a model would be equivalent to specifying

$$d_{i,j} \leq \delta \text{ for all } 0 \leq i \leq n, j \leq n, \quad (1)$$

where (an example) $\delta > 0$ is some uniformly applied separation requirement. In the case of a higher dimensional model or one where there are multiple types of separation that may not be applied uniformly (lateral, vertical, wake turbulence, etc.), this requirement simply would have the condition in Equation 1 taken over the appropriate set of vehicle indices i and j .

We can then form a set of optimization constraints to enforce vehicle separation. For each time point t , we specify that

$$\mathcal{G}_{t,i,j} = \delta - d_{t,i,j} \leq 0 \quad (2)$$

with the needed constraint derivatives with respect to the vehicle positional state vectors computed as

$$\frac{\partial \mathcal{G}_{t,i,j}}{\partial x_t} = -\frac{2(x_{t,i} - x_{t,j})}{d_{t,i,j}}$$

and

$$\frac{\partial \mathcal{G}_{t,i,j}}{\partial y_t} = -\frac{2(y_{t,i} - y_{t,j})}{d_{t,i,j}}.$$

Thus, if $\mathcal{G}_{t,i,j} \leq 0$ for all time points t and all vehicle pairs i and j , then there are no separation conflicts. If it is positive, then a conflict exists at point in time t between vehicles i and j . This direct approach of iteratively checking all possible vehicles pairs at each point in simulated time to generate a set of optimization constraints G will be called the **pairwise** method for separation constraint formulation.

For the pairwise method, as the number of aircraft n within a simulated airspace is increased, the number of possible pairings of aircraft n_{pairs} grows quadratically as

$$n_{\text{pairs}} = \frac{n(n-1)}{2} = O(n^2),$$

and performing this calculation for each point in simulated time of a model gives the pairwise method the an overall complexity $O(n^2k)$, where k represents the length of the temporal discretization for the vehicle ODEs. Thus, as an example, if $n = 1,000$ and $k = 30$, then 14,985,000 separate distance calculations must be performed in order to determine if there is any loss of separation (LOS) events in a single iteration of an airspace operations optimization problem. Though there is some opportunity for parallelization, if a direct pairwise distance method is used to constrain a problem that is complex enough to require a large number of iterations, a significant amount of computational resources may be needed to converge to a feasible solution that satisfies separation requirements.

A number of different approaches have been taken to address this issue in past simulation studies. In [18] and [19], the computational workload of checking separation is reduced by subdividing the airspace with spatial decompositions into discrete grid structure. In [20], N-D polyhedral intersection methods are applied to vehicle trajectories with a fast tree-based data structure to determine potential conflicts.

These methods are each shown to be an effective means of reducing the computational workload of computing vehicle separation within the problem domains that they present. However, the manner in which they operate (largely through binning and accumulation operations) are discontinuous with respect to the parameters in the problem (vehicle states and controls). This unfortunately precludes the use of gradient-based optimization methods. Thus, a new approach to efficiently enforce vehicle separation is needed that is both fast yet still differentiable.

II. Differentiable geometric decomposition based separation metric

For the purpose of simplicity, consider the case where a single lateral separation δ in two dimensions is required (as in Equation 1).

Note that for a single point in simulated time, specifying that all $d_{i,j} \leq \delta$ simultaneously for all pairs of agents i and j is equivalent to specifying the same condition on the minimum separation that exists,

$$\min_{i,j} d_{i,j} \leq \delta.$$

If we could determine this minimum separation for each point in time efficiently, then one could form a reduced set of optimization constraints to enforce vehicle separation. For each time point $0, \dots, t \leq k$, we specify that

$$\mathcal{G}_t = \delta - \min_{i,j} d_{t,i,j} \leq 0 \quad (3)$$

Just as in the constraint formulation for the pairwise method in Equation 2, if $\mathcal{G}_t \leq 0$ for all time points t , then there are no separation conflicts. If $\mathcal{G}_t > 0$, then a conflict exists at point in time t . Using this concept as a starting point, a number of opportunities and challenges immediately become apparent.

A. The effects of constraint aggregation

One positive consequence of this formulation is that the number of constraints to compute and provide to the optimization algorithm decreases from $k \frac{n(n-1)}{2}$ in the pairwise formulation of the constraint in Equation 2 to just k in Equation 3. Setting aside the actual process of computing these minimum quantities for a moment, it should be noted that there is an opportunity for an increase in efficiency in this change alone. Optimization problems that have a large number of input parameters (e.g. vehicle states and controls) relative to the number of output parameters (such as objectives or constraints) can enhance their performance using methods such as adjoint derivative methods[9, 21]. This is conceptually identical to the backpropagation method used in optimization of machine learning architectures in their training phases[22, 23]. By substantially reducing the number of separation constraints, adjoint methods could offer significant improvements in convergence speed for the same number of vehicles. Grouping constraints into a smaller number for improved efficiency is often referred to in the optimization literature as "constraint aggregation"[24, 25]. Aggregation can be an effective way of improving the performance of highly constrained optimization problems, it typically comes with other considerations for tuning parameters of the aggregation process, and respecting the fidelity of the original problem. Thus, the constraint $\mathcal{G}_t$ formulated in Equation 3 is equivalent to a simple form of constraint aggregation on the larger set of pairwise constraints $\mathcal{G}_{t,i,j}$ in Equation 2.

However, one challenge to any constraint is that this method, like other efficient separation metrics mentioned, it does present a challenge for use in gradient based optimization in its simplest form. It is not a continuous process with respect to the parameters of the problem, and this discontinuity can present challenges for numerical treatment. For example, as two aircraft are perturbed from being too close (and thus violating the required separation δ) into a feasible state $d_{i,j} > \delta$, then the constraint $\mathcal{G}_t$ will discontinuously shift to represent a new value representing a completely different pair of aircraft, the newest closest pair. Some means of smoothing this discontinuity is need in order for this process to be suitable for gradient based optimization. This is an issue recognized and addressed in popular forms of constraint aggregation, such as the **Kreisselmeier–Steinhauser (KS)** method [24, 26, 27]. In the KS method, the maximum value of a vector is approximated by a tunable nonlinear function in order to smooth out the derivatives of the aggregation process. Applying KS aggregation to the set of pairwise constraints $\mathcal{G}_{t,i,j}$

$$KS_t(\mathcal{G}_{t,i,j}) = \max_{i,j} \mathcal{G}_{t,i,j} + \frac{1}{\rho} \sum_{i,j} e^{\rho(\mathcal{G}_{t,i,j} - \max_{i,j} \mathcal{G}_{t,i,j})},$$

where $\rho > 0$ is a smoothness parameter. This function can be differentiated, and is smooth enough for numerical optimization. Specifically, the gradients of the KS aggregation process applied to the pairwise constraints are given as

$$\frac{\partial KS_t(\mathcal{G}_{t,i,j})}{\partial x_t} = \frac{\partial KS_t(\mathcal{G}_{t,i,j})}{\partial \mathcal{G}_{t,i,j}} \cdot \frac{\partial \mathcal{G}_{t,i,j}}{\partial x_t} = -\frac{\rho e^{\rho(\mathcal{G}_{t,i,j} - \max_{i,j} \mathcal{G}_{t,i,j})}}{\rho \sum_{i,j} e^{\rho(\mathcal{G}_{t,i,j} - \max_{i,j} \mathcal{G}_{t,i,j})}} \cdot \frac{2(x_{t,i} - x_{t,j})}{d_{t,i,j}}$$

and

$$\frac{\partial KS_t(\mathcal{G}_{t,i,j})}{\partial y_t} = \frac{\partial KS_t(\mathcal{G}_{t,i,j})}{\partial \mathcal{G}_{t,i,j}} \cdot \frac{\partial \mathcal{G}_{t,i,j}}{\partial y_t} = -\frac{\rho e^{\rho(\mathcal{G}_{t,i,j} - \max_{i,j} \mathcal{G}_{t,i,j})}}{\rho \sum_{i,j} e^{\rho(\mathcal{G}_{t,i,j} - \max_{i,j} \mathcal{G}_{t,i,j})}} \cdot \frac{2(y_{t,i} - y_{t,j})}{d_{t,i,j}}.$$

Of course, using KS aggregation in this way still clearly requires that all distance pairs be computed and does not mitigate the $O(n^2)$ complexity issue, but it would allow the optimization process to proceed with efficient adjoint derivative methods as the number of parameters of the problem grow.

B. Efficiently solving the Closest pair problem

Focusing on the complexity issue, if the closest vehicles at each point in time can be determined in a manner more efficiently than $O(n^2)$, then it is possible to verify whether the airspace is conflict-free or not more efficiently than the

pairwise method. This is an example of what is known in computational geometry as a "Closest-point problem"[28]. This is a class of problems that have a variety of researched solution methodologies in different application areas, including querying and database retrieval[29], computer graphics [30], field effects in physics modeling [31], among others.

The most common class of solution methods are recursive divide-and-conquer methods, that split the search space uniformly at each step[32]. To follow this approach and apply it to a simulated airspace model, it is necessary to first sort the vehicles in the airspace by one of two spatial dimensions, for instance the horizontal (y) dimension. The vehicles are then split into two halves about the midpoint of the airspace. Then, the search for the closest pair of vehicles is performed recursively on each of these two sets, with the case where there are 3 vehicles or less operating as the recursive base-case. Finally, at each recursion level, vehicles that exist within a thin strip-shaped region of width equal to the closest distance determined at that level to the boundary of the two halves of the spatial separation is checked. A strict upper limit to the number of vehicles that needs to be checked within this strip exists as long as the subset of vehicles is iterated over in sorted order in the second (e.g. y vertical) spatial dimension[28].

Interestingly, it has been shown that if a stable sorting algorithm is used to precompute the sorted representations of the spatial vectors x and y in each dimension, then this method can determine the aircraft i and j that are closest to each other with complexity $O(n \log(n))$ [28]. The sorting steps required to preprocess the data can be done with $O(\log(n))$ complexity, which would give the GD method a complexity of $O(k \cdot n \log(n))$ for the entire process of finding the closest pair of aircraft at all time steps in an optimization problem.

Algorithm 1 lists the pseudocode for the primary portion of this algorithm, applying the recursive divide and conquer geometric decomposition method. This includes the base-case brute force search Algorithm 2 and the strip iteration 3.

C. Formulating the proposed separation constraint

Using Algorithm 1, the process for computing the constraints $\mathcal{G}_t$ in Equation 3 can be done efficiently for each point in time, without requiring checking all pairs of aircraft.

In order to smooth out this process to be suitable for gradient based optimization, an adaptation is made to the closest pair search. A cache is passed and maintained for each level of the recursion, and every distance calculation performed that violates the separation requirement in the brute search or strip algorithms 2 and 3 are saved and returned, in addition to the identified minimum distance.

The separation constraint is formulated by adding together the quantity representing the worst-offender minimum distance together with any other observed distances that violate the required separation. For the remainder of this paper, this method for computing separation constraints for trajectory optimization will be referred to as the **Geometric Decomposition** (or **G.D.**) method. This constraint can be written as

$$\mathcal{GD}_t = \sum_{(i,j) \in \Theta} (\delta - d_{t,i,j}) \leq 0,$$

where Θ represents the cache of aircraft pairings that were observed to violate the separation constraint, which if non-empty, necessarily includes the closest pair of aircraft. The gradients of this constraint are computed simply as a summation of gradients over the same set,

$$\frac{\partial \mathcal{GD}_t}{\partial x_t} = - \sum_{(i,j) \in \Theta} \frac{2(x_{t,i} - x_{t,j})}{d_{t,i,j}}$$

and

$$\frac{\partial \mathcal{GD}_t}{\partial y_t} = - \sum_{(i,j) \in \Theta} \frac{2(y_{t,i} - y_{t,j})}{d_{t,i,j}}.$$

Note that there is no guarantee that all separation violations will be captured by this constraint in every iteration of an optimization. However, it is guaranteed that $\mathcal{GD}_t > 0$ if there exists any violation of separation requirements, and contain no loss of separation if $\mathcal{GD}_t \leq 0$.

Therefore, a fully converged airspace optimization problem with $\mathcal{GD}_t \leq 0$ will be conflict free, even if not every conflict is detected at every iteration of the optimization. This introduces a re-balancing of the computational effort of the problem. Successful convergence of an optimization problem with a GD constraint may be expected to take more

iterations than one with a pairwise constraint, but the iterations themselves should be expected to be less computationally expensive.

Algorithm 1: Geometric decomposition method for separation constraint calculation

Result: Return closest distance, indices of closest distance, and cache of all calculations

Function $MinDist(a_x, a_y, idx_x, idx_y)$:

```

/* Base case: small number of vehicles */
if  $k \leq 3$  then
    | return  $BruteSearch(a_x, idx_x)$ 
end
/* Split airspace into partitions along the x-axis */
 $Q_x \leftarrow$  lower half of  $a_x$ ;
 $R_x \leftarrow$  lower half of  $a_x$ ;
 $Q_{idx_x} \leftarrow$  lower half of  $idx_x$ ;
 $R_{idx_x} \leftarrow$  lower half of  $idx_x$ ;
 $m \leftarrow$  midpoint of  $a_x$ ;
 $Q_y, R_y, Q_{idx_y}, R_{idx_y} \leftarrow [], [], [], []$ 
for  $i \leftarrow 0$  to  $k$  do
    | if  $a_y[i] \leq m$  then
    |     |  $Q_y.insert(a_y[i])$ 
    |     |  $Q_{idx_y}.insert(idx_y[i])$ 
    | else
    |     |  $R_y.insert(a_y[i])$ 
    |     |  $R_{idx_y}.insert(idx_y[i])$ 
    | end
end
/* Recursively search each partition for the closest pair */
 $(p_1, q_1, mi_1) \leftarrow MinDist(Q_x, Q_y, Q_{idx_x}, Q_{idx_y})$ 
 $(p_2, q_2, mi_2) \leftarrow MinDist(R_x, R_y, R_{idx_x}, R_{idx_y})$ 
if  $mi_1 \leq mi_2$  then
    |  $d \leftarrow mi_1$ 
    |  $mn \leftarrow (p_1, q_1)$ 
else
    |  $d \leftarrow mi_2$ 
    |  $mn \leftarrow (p_2, q_2)$ 
end
/* Check the vertical strip nearest to partition boundary */
 $(p_3, q_3, mi_3) \leftarrow MinDistStrip(a_x, a_y, idx_y, d, mn)$ 
if  $d \leq mi_3$  then
    | return  $mn, d$ 
else
    | return  $(p_3, q_3), mi_3$ 
end

```

III. Simulations and benchmarks

The proposed GD separation constraint was benchmarked in several ways against the direct pairwise (PW) method. All benchmarks for analysis and optimizations were performed on a Macbook Pro with 3.1 GHz Intel Core i5 processor and 8 GB 2133 MHz LPDDR3 memory.

Algorithm 2: Brute force search for closest pair of vehicles

Result: Return closest distance, indices of closest distance, and cache of all calculations

Function *Brute* (ax, id_{x_y}):

/ Check each pair directly via iteration*

**/*

/ Return closest distance, indices of closest distance, and cache of all calculations*

**/*

$m_i \leftarrow$ distance between first two points in list

$m_{i_{p_1}}, m_{i_{p_2}} \leftarrow$ index of first and second points in list

for $i \leftarrow 0$ **to** $k - 1$ **do**

for $j \leftarrow i + 1$ **to** $k - 1$ **do**

$d \leftarrow$ distance between $a[0]$ and $a[1]$

$dx, dy \leftarrow$ derivative of d in x and y directions

if $d \leq m_i$ **then**

$m_i \leftarrow d$

$m_{i_{p_1}} \leftarrow id_{x_x}[i]$

$m_{i_{p_2}} \leftarrow id_{x_x}[j]$

end

end

end

return $m_{i_{p_1}}, m_{i_{p_2}}, m_i$

Algorithm 3: Checking vertical strip region for a closest pair of points

Result: Return closest distance, indices of closest distance, and cache of all calculations

Function *MinDistStrip* ($p_x, p_y, id_{x_y}, delta$):

$m_i \leftarrow$ distance between first two points in list

$m_{i_{p_1}}, m_{i_{p_2}} \leftarrow$ index of first and second points in list

for $i \leftarrow 0$ **to** $length\ of\ p_y - 1$ **do**

if $p_y[i]$ *is not within a distance delta of the boundary* **then**

 Skip this iteration;

end

for $j \leftarrow i + 1$ **to** $\min(length\ of\ p_y - 1, i + 7)$ **do**

if $p_y[j]$ *is not within a distance delta of the boundary* **then**

 Skip this iteration;

end

$d \leftarrow$ distance between $p_y[i]$ and $p_y[j]$

if $d \leq m_i$ **then**

$m_i \leftarrow d$

$m_{i_{p_1}} \leftarrow id_{x_x}[i]$

$m_{i_{p_2}} \leftarrow id_{x_x}[j]$

end

end

end

return $m_{i_{p_1}}, m_{i_{p_2}}, m_i$

A. Benchmarking the closest pair calculation for a single time point

To begin, the process of determining the closest pair of aircraft in notional 2D airspace for a single point in time was performed in order to compare the PW and GD method in a non-optimization context. This was done to capture the behavior of the full GD method, which includes pre-sorting steps that may have a measurable impact on the total run time of the algorithm. This benchmark was performed by randomly generating a set of positional x and y points in a plane with values between 0 and 100. Then each method for finding the minimum separation can be benchmarked for various number of aircraft.

Figures 1 and 2 first illustrate how the PW and GD methods differ in this benchmark, among a generated set of 15 and 75 vehicles respectively. For 15 vehicles, the PW method requires 105 separate distance calculations to determine the minimum, while the proposed GD method requires only 23. Note that the pairs that are actually computed with the GD method (shown as thin connected lines in each figure) are specifically those vehicles which are relatively close together, as would be expected given the recursive geometric composition. That is, vehicles which are across the domain from each other and unlikely to conflict are skipped by the process altogether. It's exactly these computed pairs which will contribute to the GD optimization constraint and the corresponding derivative of that constraint.

For 15 vehicles, the difference is not particularly noteworthy. However, for 75 vehicles the difference both visually and quantitatively noticeable, with the number of distance calculations needed to correctly identify the same minimum distance dropping from 2775 with PW to just 248 with GD.

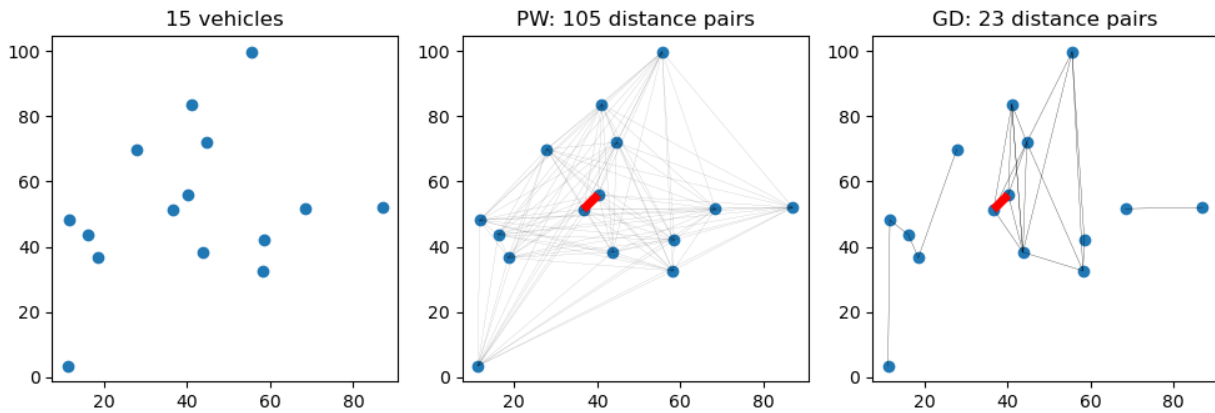


Fig. 1 Finding the closest pair within a set of 15 aircraft in a simulated space using the PW and GD method. In this case, the PW method is shown to (second figure) require computing 105 distance pairs to locate the closest pair. In the third figure, it is shown that the GD method only requires 23 distance calculations. In both cases, a line connecting the points for all computed distance pairs is plotted, with the identified closest pair highlighted in red.

This scaling was investigated for varying numbers of aircraft, from 2 up to 500. In each case, the total number of distance pairs computed was saved as well as the amount of time needed to perform the calculation (in seconds). This process was repeated 10 times for each number of aircraft in order to properly average the compute timings, to correct for background processes on the computer. The results of these benchmarks are plotted in Figure 3. These results show dramatically different scaling behavior between the PW and GD method for increasing numbers of vehicles, in terms of both the number of distance calculations needed as well as the actual compute time. This provides computational validation of the difference of their theoretical complexity. It should be noted that the scale on the y-axis of the compute times, on first examination, may seem to imply that the difference is not very computationally significant. At the highest end the pairwise method for 500 vehicles can correctly identify the closest pair of vehicles in only 0.325 seconds compared to the G.D. method's 0.012 seconds, both of which seem to be very short durations. But it should be noted that this is only for the task of identifying the closest pair for a single moment of time. An actual dynamic airspace optimization will include many of these calculations for each point in time, for many possible iterations of the optimization algorithm during the process of convergence. A difference of 0.313 seconds can be significant when taken over all of these steps of the optimization process.

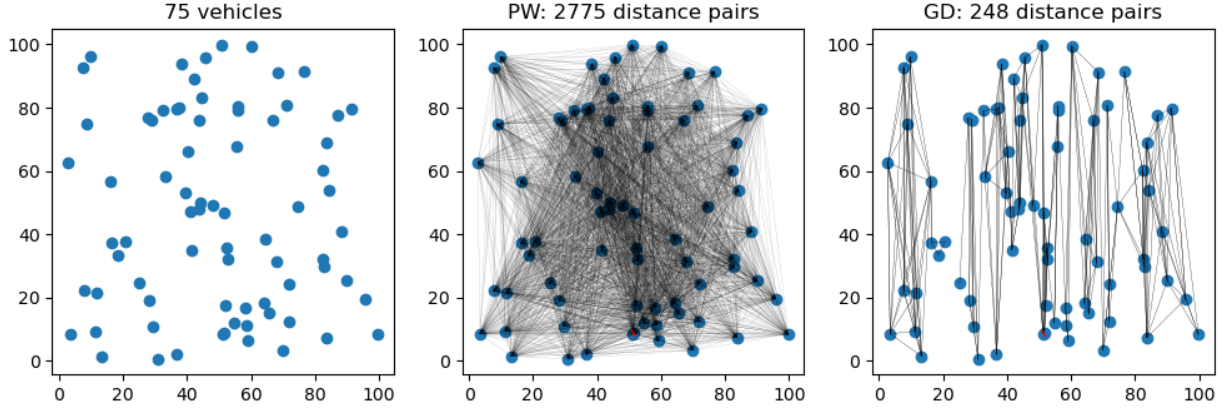


Fig. 2 Finding the closest pair within a set of 75 aircraft in a simulated space using the PW and GD method. In this case, the PW method is shown to (second figure) require computing 2775 distance pairs to locate the closest pair. In the third figure, it is shown that the GD method only requires 248 distance calculations. In both cases, a line connecting the points for all computed distance pairs is plotted, with the identified closest pair highlighted in red.

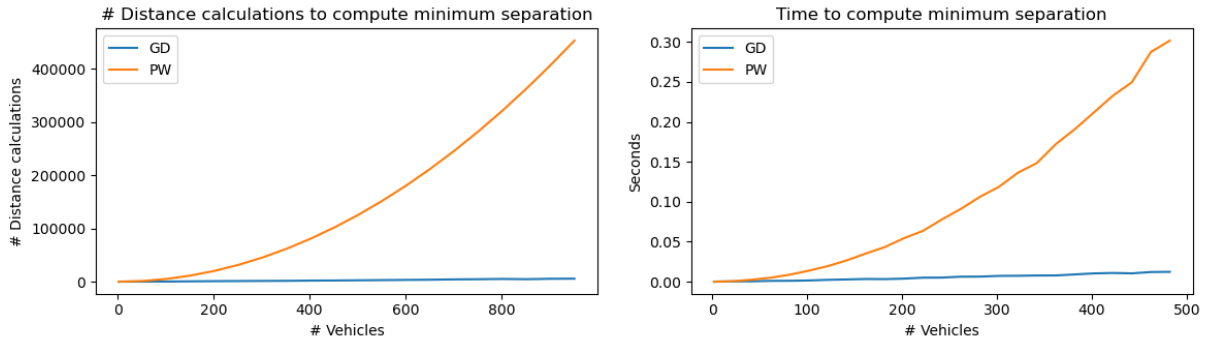


Fig. 3 Comparisons between the PW and GD methods for determining the closest pair of aircraft in a randomly generated airspace, for increasing numbers of vehicles between 2 and 500. The first plot (left) shows the total number of distance pairs needed to find the minimum separation (similar to Figures 1 and 2), while the second plot shows actual compute timings, in seconds. As expected, the GD method is seen to scale much more favorably than the PW method measured in both the number of calculations needed and the total time needed for those calculations.

B. Benchmarking optimizations with separation constraints

Next, full dynamic trajectory optimization problems were formulated in order to compare the PW and GD methods for formulating separation constraints. In addition to these, optimizations were also performed using KS aggregation applied to the PW constraints, reducing the total number of constraints to a number identical to the number provided by the GD method. This was done in order to separate performance improvements because of the actual reduction in number of distance calculations from performance gains due to the aggregation into a reduced number of constraints. It was expected that these aspects would each have their own separate effect on computational performance of the optimization process.

The optimization problem implemented was the minimum flight time problem as described in Table 1, notionally as a set of small UAS (Unmanned Aircraft System) drones operating within compact airspaces. The vehicle positions were specified as dynamical states to be solved by the optimizer. Each vehicle begins simultaneously at prespecified points, and are required to fly to specific destinations, respecting set limits on velocity of 22 meters/second. All vehicles were required to uniformly maintain a specified minimum separations between 25 and 50 meters (depending on the specific scenario) from each other at all times. Each airspace simulation was developed in the Python programming language

as an optimization problem using the OpenMDAO framework[33] and the optimal control library Dymos[12]. The SNOPT optimizer was used to converge all optimization problems, which makes use of a scalable and efficient quadratic programming method suitable for problems with a large number of design variables and constraints[34].

Two separate types of configurations for the origin and destinations for each vehicle were formulated. First, a rectangular 1 km by 1 km airspace was generated, with each vehicle having an origin and destination randomly positioned within. This was done in order to simulate a set of vehicle missions with fairly low, more localized levels of interaction within the airspace. This way, the number of distance calculations truly needed to deconflict the airspace is far less than the total number of pairs of vehicles that could be compared. This is imagined as being a more likely scenario in reality than a scenario involving the active interaction of all aircraft, and it was expected that the G.D. method would be particularly performant in this case. An example of a 10 vehicle optimization with this airspace configuration is shown in Figure 4. This figure shows the trajectory paths for each vehicles, with and without the use of separation constraints to enforce airborne separation.

Second, a more complex high-interaction airspace was generated within a circle of radius 1 km. Each vehicle had an origin and destination point placed randomly along the edge of this circle, with limits specified such that the straight line connecting each origin to each destination nearly crossed the center of the circle. This scenario generates a high level of interaction between the vehicles during the middle of the simulation, making the number of relevant distance pairs at those times far less sparse than in the first scenario. This was an attempt at creating a worst-case scenario as a fair bound for the GD method, which may miss many potential losses of separation between aircraft in early iterations of the optimization process and require more iterations to successfully converge the model than the other two methods. Two plots shows example 10 vehicle optimizations with this configuration (both with and without separation constraints) are shown in Figure 5.

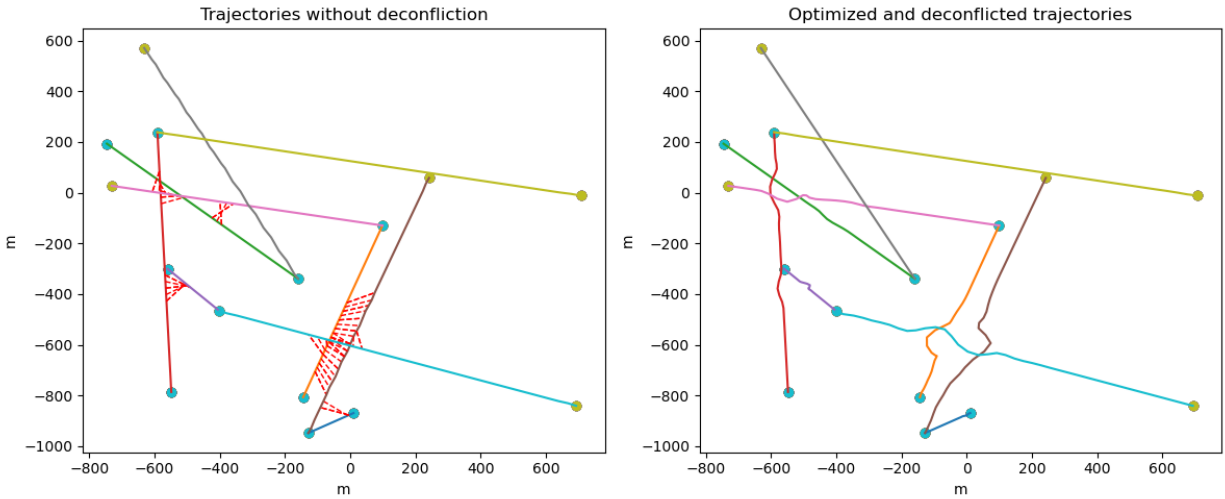


Fig. 4 Plot illustrating the setup of the rectangular airspace joint trajectory optimization problem. 10 vehicles have origins and destinations placed randomly within the airspace, which produces small areas of limited vehicle interaction distributed in the airspace. The left plot shows the trajectories optimized with respect to time when no separation constraint is added to the problem, resulting in multiple losses of separation over time that are highlighted with red lines. On the right, the results are shown from an optimization that has been optimized with a separation constraint included, that results in the removal of all conflicts.

For each of these airspace types, a set of optimization problems was performed to compare the P.W., P.W. with K.S. aggregation, and G.D. methods of specifying separation constraints. These problems were generated with for $n = 2, 4, 8, 15, 30, 50$, and 100 vehicles, with metrics averaged over 10 runs. These metrics included the number of iterations of the optimization algorithm needed for convergence and the total run time of the optimization. The results of these optimizations are shown in Table 2 for the rectangular low-interaction airspace configuration, and Table 3 for the circular high-interaction configuration.

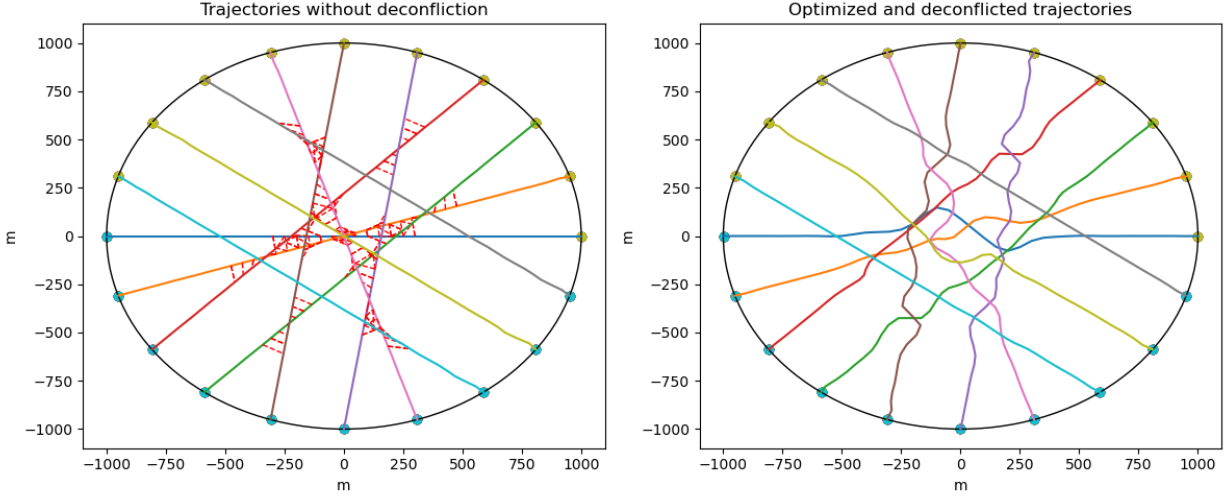


Fig. 5 Plot illustrating the setup of the circular airspace joint trajectory optimization problem. 10 vehicles have origins and destinations along the boundary of the circular airspace, which produces an area of high interaction in the center. The left plot shows the trajectories optimized with respect to time when no separation constraint is added to the problem, resulting in multiple losses of separation over time that are highlighted with red lines. On the right, the results are shown from an optimization that has been optimized with a separation constraint included, that results in the removal of all conflicts.

IV. Results and discussion

Table 3 lists the number of vehicles, number of optimization design variables, number of optimization constraints, the total compute time (in seconds) needed to converge the optimization problem, and total number of optimization algorithm iterations needed to converge the problem.

The number of constraints in this case includes not only the separation constraints, but also the nonlinear constraints from the collocation method used to transcribe the equations of motion as well as the boundary conditions on the trajectories that enforce fixed origin and destination points. From here, several trends are seen to emerge based on the number of vehicles in the model.

First, the number of design variables in the model grow linearly for all problems as the number of aircraft are increased, but the number of optimization constraints sharply diverges between the P.W. method and the K.S. and G.D. methods. This is expected based on the constraint aggregation inherent in both the P.W. + K.S. and G.D. methods, as discussed in Section II.A. From 2 vehicles to 100, the number of constraints for the P.W. method grow from 197 to 52901, while for the P.W. + K.S. and G.D. method they grow at a far slower rate from 197 to 7841. Since the number of design variables for all problems grows from 241 to 12001 between 2 and 100 vehicles, the P.W. + K.S. and G.D. methods always have a number of constraints that is less than the number of design variables. Therefore, the gradient values provided to the SNOPT optimizer will always be computed using the efficient adjoint method.

The average total compute times are fairly comparable for low numbers of vehicles between the three methods. In the case of two vehicles, the P.W. method takes the least amount of time to fully converge to a deconflicted solution (i.e. one without any violations of separation requirements), requiring just 0.643 seconds on average compared to 0.574 and 0.551 for P.W. + K.S. and G.D. respectively. However, with a small increase of the size of the airspace to just 4 vehicles, the differences in run times begin to contrast. The P.W. method requires 1.912 seconds to the P.W. + K.S. and G.D. methods 1.156 and 1.016 seconds, nearly half the run time of the P.W. method. The G.D. method is slightly faster than the P.W. + K.S. method at this stage, but both are fairly comparable in performance. The contrast for all 3 methods continues to develop as the number of vehicles increase. For 30 vehicles, the P.W. method requires almost 25 minutes to converge. Using K.S. aggregation brought this down to 11 minutes, and the G.D. method needed only about 81 seconds (1.3 minutes) in comparison. For 50 vehicles, the P.W. took about 3.5 hours compared to about 1.6 hours when using K.S. aggregation and 9.4 minutes when using the G.D. method. For 100 vehicles, only the optimizations using the G.D. method successfully converged, which took 634.769 seconds on average (about 10.5 minutes).

The number of iterations between the methods is notable, contrasting between the methods in a way nearly inverse to

# Vehicles	# Des. Vars.	# Constraints			Compute time (s)			# Iterations		
		PW	PW+KS	GD	PW	PW+KS	GD	PW	PW+KS	GD
.										
2	241	197	197	197	0.543	0.574	0.551	167	169	209
4	481	553	353	353	1.912	1.156	1.016	276	343	553
8	961	1745	665	665	10.313	4.997	3.764	481	754	919
15	1801	5371	1211	1211	83.687	39.062	11.655	1064	1501	1314
30	3601	19741	2381	2381	1481.120	652.609	81.546	1870	3640	2912
50	6001	52901	3941	3941	12644.514	5772.483	594.037	2977	8712	8604
100	12001	205801	7841	7841	.	.	634.769	.	.	13501

Table 2 Tabulated results from the optimizations performed on the rectangular, lower-interaction model. This table lists performance data from the airspace optimization problem specific in Table 1 for $n = 2, 4, 8, 25, 30, 50$, and 100 vehicles. The unaggregated direct pairwise distance metric (PW) is being compared to the use of the pairwise metric with K.S. aggregation (PW+KS) and to the geometric decomposition (G.D.) metric for the formulation of the vehicles separation constraints. The total number of design variables, number of constraints, overall compute time and number of major iterations needed to converge each optimization (averaged over 10 runs) are listed.

# Vehicles	# Des. Vars.	# Constraints			Compute time (s)			# Iterations		
		PW	PW+KS	GD	PW	PW+KS	GD	PW	PW+KS	GD
.										
2	241	197	197	197	0.651	0.664	0.660	404	404	385
4	481	553	353	353	2.01	1.305	1.406	317	621	1038
8	961	1745	665	665	13.985	5.835	4.311	1467	2323	2580
15	1801	5371	1211	1211	87.591	42.567	13.296	1043	1481	1272
30	3601	19741	2381	2381	1519.810	696.88	106.303	1345	2842	2422
50	6001	52901	3941	3941	.	6440.154	986.481	.	5908	22812
100	12001	205801	7841	7841	.	.	1691.576	.	.	28171

Table 3 Tabulated results from optimizations performed on the circular high-interaction airspace model, giving performance data from the airspace optimization problem specific in Table 1 for $n = 2, 4, 8, 25, 30, 50$, and 100 vehicles. The unaggregated direct pairwise distance metric (PW) is being compared to the use of the pairwise metric with K.S. aggregation (PW+KS) and to the geometric decomposition (G.D.) metric for the formulation of the vehicles separation constraints. The total number of design variables, number of constraints, overall compute time and number of major iterations needed to converge each optimization (averaged over 10 runs) are listed.

the run times. Regardless of the number of vehicles, the P.W. method required fewer total optimizer iterations than either of the other methods. For instance, the P.W. + K.S. and G.D. methods respectively ending up requiring 8716 and 8604 total iterations compared to 2977 for P.W. for 50 vehicles. This is likely due to the P.W. method, being unaggregated, providing more complete information to the optimizer at every iteration. Thus, the optimizer takes a more advantageous steps on average at each iteration towards the optimal solution with the P.W. method. However, the computational cost of producing the information needed to produce these quality iterations becomes rapidly more expensive as the number of vehicles increases. This is the essential tradeoff with using the K.S. or G.D. methods to reduce the number of constraints.

Table 2 tabulates the same information for the circular, high-interaction airspace model. The size of each optimization problem in terms of number of vehicles and resulting number of design variables and constraints is identical to the rectangular low-interaction configuration. But as expected, the models in the high-interaction airspace configuration were generally more difficult to converge, requiring more compute time and more optimizer iterations to converge successfully. This is particularly true for the G.D. method, which was expected to be less computationally performant than in the low-interaction configuration since large interacting clusters of vehicles will necessitate that more vehicle distance pairs will end up being computed in the recursive decomposition used to locate the closest vehicle pair. It is also possible that a large portion of vehicle pairs that violate the separation limits are missed in the early iterations of the optimization, necessitating even more iterations to properly converge to deconflicted feasibility for all trajectories. Yet the G.D. method is still the fastest method for converging the optimization problems for the problems involving 8 vehicles and higher.

The suitability for recursive geometric decomposition via a direct solution to the closest pair problem for handling separation constraints is clearly demonstrated in these results. The specific form of the separation criteria as a strict limit to the closest distance between any vehicles was a key element for the G.D. method to be applicable. Applying this approach to other elements of an airspace operations optimization problem that involve a potentially large number of vehicle interactions beyond simply their distance (such as constraints on inter-vehicle communications, wake modeling, joint aeroacoustics, etc.) would require a similar criteria, that may not have a direct analogy in terms of sorting and splitting the airspace with a geometric decomposition.

There are many further avenues for exploring how to handle dynamic optimization constraints involving pairwise vehicle interactions for trajectory-based airspace optimization problems. Further exploration of the discretized grid-based methods with novel differentiable formulations could provide a means for implementing a variety of types of airspace constraints as aggregations over appropriate models of field phenomena. For instance, instead of modeling separation as a limit to the closest distance between any two aircraft, it could be implemented as a limit within an aircraft density field modeled at a particular spatial resolution, applied across an airspace. There are significant challenges to implementing such a method in a differentiable fashion. But if it was feasible, this could easily extend beyond implementation of separation constraints to broader metrics, such as communications modeling, airspace complexity metrics, and trajectory flexibility metrics[35, 36].

References

- [1] Eck, J., “The Future of the NAS,” 2016.
- [2] Kopardekar, P., Rios, J., Prevot, T., Johnson, M., Jung, J., and Robinson, J. E., “Unmanned aircraft system traffic management (UTM) concept of operations,” 2016.
- [3] Mueller, E. R., Kopardekar, P. H., and Goodrich, K. H., “Enabling airspace integration for high-density on-demand mobility operations,” *17th AIAA Aviation Technology, Integration, and Operations Conference*, 2017, p. 3086.
- [4] Thipphavong, D. P., Apaza, R., Barmore, B., Battiste, V., Burian, B., Dao, Q., Feary, M., Go, S., Goodrich, K. H., Homola, J., et al., “Urban air mobility airspace integration concepts and considerations,” *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3676.
- [5] O’Connor, C. J., Kennedy, K. D., Underwood, M. C., Harrivel, A. R., Stephens, C., Comstock, J. R., and Last, M. C., “Development of a simulation platform to evaluate integration of UAM traffic into the NAS,” *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3850.
- [6] Prevot, T., Homola, J., and Mercer, J., “From rural to urban environments: human/systems simulation research for low altitude UAS Traffic Management (UTM),” *16th AIAA Aviation Technology, Integration, and Operations Conference*, 2016, p. 3291.
- [7] Xue, M., and Rios, J., “Initial study of an effective fast-time simulation platform for unmanned aircraft system traffic management,” *17th AIAA Aviation Technology, Integration, and Operations Conference*, 2017, p. 3073.
- [8] Zhao, Z., Luo, C., Zhao, J., Qiu, Q., Gursoy, M. C., Caicedo, C., and Basti, F., “A simulation framework for fast design space exploration of unmanned air system traffic management policies,” *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, IEEE, 2019, pp. 1–10.
- [9] Gray, J. S., Hearn, T. A., Moore, K. T., Hwang, J., Martins, J., and Ning, A., “Automatic evaluation of multidisciplinary derivatives using a graph-based problem formulation in OpenMDAO,” *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2014, p. 2042.
- [10] Hwang, J. T., Lee, D. Y., Cutler, J. W., and Martins, J. R., “Large-scale multidisciplinary optimization of a small satellite’s design and operation,” *Journal of Spacecraft and Rockets*, Vol. 51, No. 5, 2014, pp. 1648–1663.
- [11] Herman, A. L., and Conway, B. A., “Direct optimization using collocation based on high-order Gauss-Lobatto quadrature rules,” *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 3, 1996, pp. 592–599.
- [12] Falck, R. D., and Gray, J. S., “Optimal control within the context of multidisciplinary design, analysis, and optimization,” *AIAA Scitech 2019 Forum*, 2019, p. 0976.
- [13] Falck, R. D., Ingraham, D., and Aretskin-Hariton, E., “Multidisciplinary Optimization of Urban-Air-Mobility Class Aircraft Trajectories with Acoustic Constraints,” *2018 AIAA/IEEE Electric Aircraft Technologies Symposium*, 2018, p. 4985.
- [14] Falck, R. D., Chin, J., Schnulo, S. L., Burt, J. M., and Gray, J. S., “Trajectory optimization of electric aircraft subject to subsystem thermal constraints,” *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017, p. 4002.
- [15] Jasa, J. P., Mader, C. A., and Martins, J., “Trajectory Optimization of a Supersonic Aircraft with a Thermal Fuel Management System,” *2018 Multidisciplinary Analysis and Optimization Conference*, 2018, p. 3884.
- [16] Rios, L. M., and Sahinidis, N. V., “Derivative-free optimization: a review of algorithms and comparison of software implementations,” *Journal of Global Optimization*, Vol. 56, No. 3, 2013, pp. 1247–1293.
- [17] Conn, A. R., Scheinberg, K., and Vicente, L. N., *Introduction to derivative-free optimization*, SIAM, 2009.
- [18] Jardin, M., “Grid-based strategic air traffic conflict detection,” *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005, p. 5826.
- [19] Sridhar, B., and Chatterji, G., “Computationally efficient conflict detection methods for air traffic management,” *Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041)*, Vol. 2, IEEE, 1997, pp. 1126–1130.
- [20] Kuenz, A., *High performance conflict detection and resolution for multi-dimensional objects*, Vol. 2015, 2015.
- [21] Martins, J. R., Alonso, J. J., and Reuther, J. J., “A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design,” *Optimization and Engineering*, Vol. 6, No. 1, 2005, pp. 33–62.

- [22] Chauvin, Y., and Rumelhart, D. E., *Backpropagation: theory, architectures, and applications*, Psychology press, 1995.
- [23] Ng, S., Cheung, C., Leung, S., and Luk, A., “Fast convergence for backpropagation network with magnified gradient function,” *Proceedings of the International Joint Conference on Neural Networks, 2003.*, Vol. 3, IEEE, 2003, pp. 1903–1908.
- [24] Lambe, A. B., Kennedy, G. J., and Martins, J. R., “An evaluation of constraint aggregation strategies for wing box mass minimization,” *Structural and Multidisciplinary Optimization*, Vol. 55, No. 1, 2017, pp. 257–277.
- [25] Zhang, K.-S., Han, Z.-H., Gao, Z.-J., and Wang, Y., “Constraint aggregation for large number of constraints in wing surrogate-based optimization,” *Structural and Multidisciplinary Optimization*, Vol. 59, No. 2, 2019, pp. 421–438.
- [26] Kreisselmeier, G., and Steinhauser, R., “Systematic control design by optimizing a vector performance index,” *Computer aided design of control systems*, Elsevier, 1980, pp. 113–117.
- [27] Kreisselmeier, G., and Steinhauser, R., “Application of vector performance optimization to a robust control loop design for a fighter aircraft,” *International Journal of Control*, Vol. 37, No. 2, 1983, pp. 251–284.
- [28] Shamos, M. I., and Hoey, D., “Closest-point problems,” *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, IEEE, 1975, pp. 151–162.
- [29] Corral, A., Manolopoulos, Y., Theodoridis, Y., and Vassilakopoulos, M., “Algorithms for processing k-closest-pair queries in spatial databases,” *Data & Knowledge Engineering*, Vol. 49, No. 1, 2004, pp. 67–104.
- [30] Schneider, P., and Eberly, D. H., *Geometric tools for computer graphics*, Elsevier, 2002.
- [31] Callahan, P. B., and Kosaraju, S. R., “Algorithms for Dynamic Closest Pair and n-Body Potential Fields,” *SODA*, Vol. 95, 1995, pp. 263–272.
- [32] Tao, Y., Yi, K., Sheng, C., and Kalnis, P., “Efficient and accurate nearest neighbor and closest pair search in high-dimensional space,” *ACM Transactions on Database Systems (TODS)*, Vol. 35, No. 3, 2010, pp. 1–46.
- [33] Gray, J. S., Hwang, J. T., Martins, J. R., Moore, K. T., and Naylor, B. A., “OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization,” *Structural and Multidisciplinary Optimization*, 2019, pp. 1–30.
- [34] Gill, P. E., Murray, W., and Saunders, M. A., “SNOPT: An SQP algorithm for large-scale constrained optimization,” *SIAM review*, Vol. 47, No. 1, 2005, pp. 99–131.
- [35] Vogel, M., Schelbert, K., Fricke, H., and Kistan, T., “Analysis of airspace complexity factors’ capability to predict workload and safety levels in the TMA,” *Tenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2013)*, Chicago, 2013.
- [36] Idris, H., and Wing, D., “Improving separation assurance stability through trajectory flexibility preservation,” *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, 2010, p. 9011.