MODELING AIRCRAFT MOVEMENTS USING STOCHASTIC HYBRID SYSTEMS

Arne Bang Huseby
University of Oslo

Modern simulation techniques and computer power makes it possible to simulate systems with continuous state spaces. In this paper we consider the problem of simulating aircraft movements. Such simulation studies are of interest in order to evaluate the risk of collision or near collision events in areas with heavy air traffic. Application areas include systems for automatic air traffic control (ATC) and airport traffic planning.
Risk elements in this area include arrivals of aircrafts into the area of interest, weather conditions, chosen runway and landing direction as well as arbitrary deviations from the normal flight trajectories. A realistic simulation model, incorporating all or most of these elements, require a combination of different tools. A popular framework for such modeling is stochastic hybrid systems.
A risk event occurs when two (or more) aircrafts are too close to each other in the air. When such an event occurs, the aircrafts will attempt to change their trajectories in order to avoid fatal accidents. Fortunately, risk events are fairly rare. Thus, in order to obtain stable results for the probability of a collision or near collision it is necessary to run the model for a substantial amount of time.
In the paper we propose a methodology for simulating aircraft movements based on ordinary differential equations and counting processes. The models and methods are illustrated with some simulation examples.

1 INTRODUCTION

The air traffic has experienced an incredible worldwide growth. As a result the current air traffic control systems are being pushed to the limit. The computer technology used in most control systems is often obsolete, placing heavy workload on the air traffic controllers. Thus, there is a growing need for improving the utilization of the airspace using more advanced technology and control procedures.

Due to the severe security considerations, new control systems must be tested and optimized within some sort of simulation environment before they are implemented. Modern simulation techniques and computer power makes it possible to carry out such simulations very efficiently and realistically. Among the pioneer works in this area are (Koo et al. 1997) and (Tomlin et al. 1998). Typical for these and similar works is the use of deterministic hybrid systems, i.e., simulation systems with mixed discrete event and continuous time dynamics.

While deterministic models may work very well as a description of the individual aircrafts, the overall performance of an air traffic control system depends on its ability to operate in a stable way in a stochastic environment. More recent approaches to the problem such as (Bernadsky et al. 2004) and (Glover and Lygeros 2004) use stochastic differential equations to model aircraft trajectories. One major risk driver not included in these works, however, is the *traffic density*. A control system that works perfectly under normal traffic may break down if the workload grows. Thus, when evaluating a system, it is important to include

In the recent paper (Bayen et al. 2006) air traffic is modeled using a network flow model. Instead of modeling the movements of each individual aircraft, they describe the evolution of the traffic density by using partial differential equations. This approach, which is also used in highway traffic models, enables them to take a more global perspective on the air traffic rather than looking at the air traffic in some limited area. Still this model does not include any stochastic elements.

In the present paper we focus on building and analyzing models for air traffic control in a stochastic environment. Contrary to (Bernadsky et al. 2004) and (Glover and Lygeros 2004) we use ordinary differential equations instead of stochastic differential equations for the aircraft movements. Such models are somewhat simpler and less noisy, while at the same time being sufficiently realistic. Thus, our approach is closer to the framework used in (Koo et al. 1997) and (Tomlin et al. 1998). The stochastic elements in our model include traffic density as well as uncertainty about the points where the aircrafts enter and leave the airspace. We believe that these elements are the most important sources of uncertainty in this context.

The main purpose of this paper is to show how different issues regarding air traffic control can be handled within such a framework. In order to keep the technical details at a modest level, the models are simplified as much as possible.

Using the proposed framework one can e.g., analyze how different aircraft trajectories can affect the security level. This is useful in the process of improving designated trajectories with respect to risk. One can also determine the maximal acceptable traffic density satisfying a certain security criterion.

In the present paper we have studied the risk reducing effect of a given flight deviation procedure. According to this procedure, the aircrafts are responsible for avoiding risk events. However, this procedure can easily be made much more sophisticated by including support from an air traffic control center. Although this leads to a more complex model, it is not difficult to extend the framework to facilitate this.

A common issue in air traffic control is that the trajectories for take-off and landing may depend on the weather conditions, especially the wind speed and direction. As a result the movements of the different aircrafts present become correlated. By incorporating a weather model into the picture, such effects can be studied in detail. The proposed framework can easily be extended to include such elements.

2 CONCEPTUAL MODEL

Throughout this paper we consider some airspace of interest, denoted $\mathcal{A}$. For simplicity we assume that $\mathcal{A}$ is sufficiently small so that all movements within $\mathcal{A}$ can be described using Euclidian geometry. Thus, in particular the shortest possible trajectory between any two points in $\mathcal{A}$ is the straight line between them. As time goes by, aircrafts enter, fly through and eventually leave $\mathcal{A}$ according to some suitable random process. The randomness includes both the arrival times as well as the trajectories of the aircrafts.

The aircrafts arrive $\mathcal{A}$ according to a suitable counting process $\{N(t)\}$, where $N(t)$ denotes the number of aircrafts arrived at time t. An aircraft flying through $\mathcal{A}$ enters into the airspace at a point, referred to as its *arrival point*, and leaves at another which we will call its *departure point*. The arrival and departure points of the ith aircraft are denoted respectively by $\boldsymbol{X}_{a,i}$ and $\boldsymbol{X}_{d,i}$, $i = 1, 2, \ldots$. We assume that $(\boldsymbol{X}_{a,i}, \boldsymbol{X}_{d,i})$, $i = 1, 2, \ldots$ are independent random vectors with a common distribution. We also introduce the time of arrival and departure for each aircraft denoted respectively $T_{a,i}$ and $T_{d,i}$, $i = 1, 2, \ldots$, where $T_{a,i}$, $i = 1, 2, \ldots$ correspond to the jumps of $\{N(t)\}$. That is, we have:

$$N(t) = \sum_{i=1}^{\infty} \mathrm{I}(T_{a,i} \leq t). \tag{1}$$

We may think of such a model as a queueing system where the clients are the aircrafts arriving at times $T_{a,i}, i = 1, 2, \ldots$, and the processing times are given by $T_{p,i} = T_{d,i} - T_{a,i}$, $i = 1, 2, \ldots$. The ith aircraft is present in the queue at time t if $T_{a,i} \leq t \leq T_{d,i}$, $i = 1, 2, \ldots$. The queueing process, denoted $\{Q(t)\}$, describes the length of the queue as a function of time, where $Q(t)$ is given by:

$$Q(t) = \sum_{i=1}^{\infty} \mathrm{I}(T_{a,i} \leq t < T_{d,i}). \tag{2}$$

We also introduce the process $\{M(t)\}$ describing the number of processed aircrafts at time t, where $M(t)$ is given by:

$$M(t) = \sum_{i=1}^{\infty} \mathrm{I}(T_{d,i} \leq t). \tag{3}$$

We observe that $Q(t) = N(t) - M(t)$.

The position of the ith aircraft at time t is denoted $\boldsymbol{x}_i(t)$, $i = 1, 2, \ldots$. Thus, the trajectories of the ith aircraft is given by $\{\boldsymbol{x}_i(t) : T_{a,i} \leq t \leq T_{d,i}\}$, with the boundary conditions $\boldsymbol{x}_i(T_{a,i}) = \boldsymbol{X}_{a,i}$ and $\boldsymbol{x}_i(T_{d,i}) = \boldsymbol{X}_{d,i}$, $i = 1, 2, \ldots$.

The velocity of the ith aircraft at time t is denoted $\dot{\boldsymbol{x}}_i(t)$, and we have:

$$\boldsymbol{x}_i(t) = \boldsymbol{X}_{a,i} + \int_{T_{a,i}}^{t} \dot{\boldsymbol{x}}_i(u) du. \tag{4}$$

for $i = 1, 2, \ldots$. Initially we simply assume the ith aircraft flies with constant speed s_i along a straight line between $\boldsymbol{X}_{a,i}$ and $\boldsymbol{X}_{d,i}$. In this case we have for $i = 1, 2, \ldots$:

$$\dot{\boldsymbol{x}}_i(t) = \frac{\boldsymbol{X}_{d,i} - \boldsymbol{X}_{a,i}}{\|\boldsymbol{X}_{d,i} - \boldsymbol{X}_{a,i}\|} s_i. \tag{5}$$

We say that a *risk event* occurs when two (or more) aircrafts are too close to each other in the air. More precisely we define a critical distance C. At each point of time t we let $J(t)$ denote the index set of aircrafts present in $\mathcal{A}$ at t and compute:

$$D(t) = \min_{i,j \in J(t), i \neq j} \{\|\boldsymbol{x}_i(t) - \boldsymbol{x}_j(t)\|\}. \tag{6}$$

A risk event occurs at time t if $D(t) < C$.

In order to avoid risk events there are several options to consider. If possible one could allow the aircrafts to fly through $\mathcal{A}$ along designated trajectories where $D(t)$ is always greater than C. While this obviously is a very safe solution, this typically leads to less than optimal utilization of the airspace. Moreover, if $\mathcal{A}$ is close to an airport, it may be impossible to find such trajectories. A more realistic strategy would be to look for trajectories where the risky areas are reduced as much as possible, while still allowing the aircrafts to reach their intended destinations. When designing such trajectories, one must take into account that different weather conditions may trigger different needs for trajectories.

Assuming that the allowed trajectories have been chosen, the next option is to place restrictions on the number of aircrafts allowed to be present at the same time in the airspace. In our model this implies controlling the properties of the counting process $\{N(t)\}$. An extreme solution is simply to allow only one aircraft in $\mathcal{A}$ at each point of time. Since this reduces the capacity drastically, this is rarely an acceptable solution. Thus, one may try to maximize the airspace capacity while keeping the frequency of risk events below a certain level.

Finally, given the allowed trajectories and airspace capacity, the air traffic control center may instruct the aircrafts to change their trajectories in order to avoid risk events. With increasing air traffic, this becomes more and more complex. Thus, having automatic procedures is desirable.

For all options it is necessary to evaluate the solutions in some way or the other. In the present paper we will focus on the latter two options, and show how these problems can be analyzed using simulation. The solutions will be evaluated with respect to several measures. The first, and most fundamental measure is the limiting fraction of the time where the minimum distance between two aircrafts is below the critical value C. This measure is denoted R, and given by:

$$\bar{R} = \lim_{t\to\infty} \frac{1}{t} \int_0^t \mathrm{I}(D(u) < C)du. \tag{7}$$

In relation to this, we also estimate the asymptotic average minimum distance between two aircrafts in $\mathcal{A}$, that is:

$$\bar{D} = \lim_{t\to\infty} \frac{1}{t} \int_0^t D(u)du. \tag{8}$$

Secondly we consider the process $\{M(t)\}$ counting the number of processed aircrafts as a function of time and estimate the asymptotic average throughput:

$$\bar{M} = \lim_{t\to\infty} \frac{M(t)}{t}. \tag{9}$$

Finally, we estimate the asymptotic average processing time:

$$\bar{T}_p = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} T_{p,i}. \tag{10}$$

3 SOFTWARE IMPLEMENTATION

The models described in this paper have been implemented in *java*. The object structure of the software

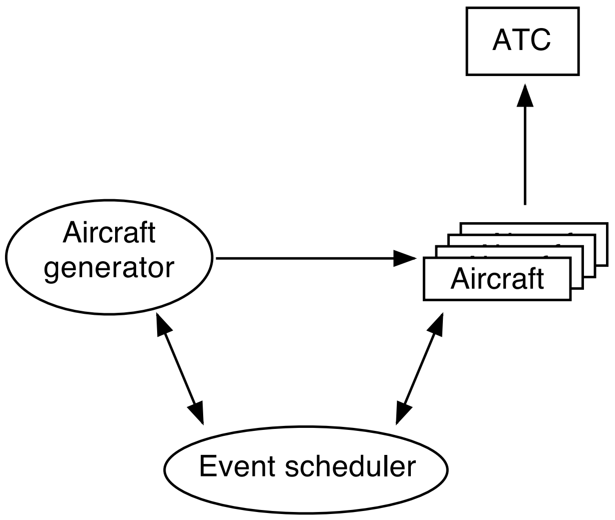

Figure 1: *Object Structure*

is outlined in Figure 1. The *aircraft* objects are generated by the *aircraft generator* according to to the chosen counting process $\{N(t)\}$. The aircraft generator also generates the arrival and departure points for each of the aircrafts. When generated the aircraft register at the *ATC* (air traffic control) which maintains a list of objects in the airspace $\mathcal{A}$. The aircraft objects keeps track of their positions $\boldsymbol{x}_i(t)$ and velocities $\dot{\boldsymbol{x}}_i(t)$. Initially the position of the ith aircraft is of course equal to its arrival point, $\boldsymbol{X}_{a,i}$, while the velocity is given by (5), $i = 1, 2, \ldots$. Each time the position and velocity of an aircraft is updated, it checks if the departure point is reached (within a suitable ϵ). If so, the aircraft is removed from the aircraft list.

The aircraft list is made available by the use of public static methods. Thus, any object has access to all the aircrafts in $\mathcal{A}$, their positions and velocities. This model allows the air traffic control to be handled either centralized by the ATC object, or locally by the aircraft objects. In our implementation, however, we have chosen to handle everything locally. That is, it is the aircraft objects that must find a way to avoid risk events.

The *event scheduler* handles both *discrete* events related to the aircraft generator as well as *dense time* events such as position and velocity updates for the aircrafts. Thus, each aircraft is generated as a response to a "GENERATE" event. When this event is processed by the aircraft generator, it samples the waiting time to the next "GENERATE" event as well, and sends this to the event scheduler. Similarly, position and velocity updates are handled by the aircraft objects as a response to an "UPDATE" event. When an aircraft processed an "UPDATE" event, it computes the time for the next "UPDATE" event and sends this to the event scheduler. Note that since all "UPDATE" events are handled locally by the aircrafts, it is possible to use different update intervals depending e.g., on the distance to the closest aircraft. Thus, if an aircraft travels in a "safe" area, it may use a long update interval, while a short update interval is used when the distance to the closest aircraft is small. By using this technique it is possible to accelerate the simulations considerably without compromising on the numerical precision.

4 ANALYSIS

4.1 *Analysis of airspace capacity*

In this section we present results from the simulations starting out with an analysis of the airspace capacity. Throughout this section we simplify the problem by assuming that all aircrafts fly at the same height, say 10,000 ft (3,048 meters). The airspace $\mathcal{A}$ is chosen to be a square with sides of length 10 nautical miles (18,520 meters). Positions in $\mathcal{A}$ are specified relative to a coordinate system with metric units, and with origin at the center of $\mathcal{A}$. Thus, $\mathcal{A}$ is the set of vectors $\boldsymbol{x} = (x_1, x_2, x_3)$, such that $x_j \in [-9,260; 9,260]$, for $j = 1,2$ and $x_3 = 3,048$.

At this stage we also assume that all aircrafts fly at constant speed along straight lines from their arrival points to their departure points. In the simulations the speed was chosen to be 250 knots (128.6 m/s).

All flights through $\mathcal{A}$ are assumed to be either *northbound* or *eastbound*. For *northbound* flights the arrival points are sampled within an interval I_S of the southern border of $\mathcal{A}$ and departure points within an interval I_N of the northern border of $\mathcal{A}$. Similarly, for *eastbound* flights the arrival points are sampled within an interval I_W of the western border of $\mathcal{A}$ and departure points within an interval I_E of the eastern border of $\mathcal{A}$. When an aircraft is generated, we let $\Pr(\text{Flight is northbound}) = \Pr(\text{Flight is eastbound}) = 1/2$. We then sample the arrival and departure points uniformly from the corresponding intervals. In the simulations we let all the intervals have lengths of 200 meters centered at the midpoint of their respective borders. With such short intervals the angles between any crossing trajectories will be approximately 90 degrees.

For the arrival process $\{N(t)\}$ we chose a counting process with i.i.d. waiting times sampled from a censored exponential distribution. More specifically, if $W_1, W_2, \ldots$ denote the waiting times between arrivals, we assume that $W_1, W_2, \ldots$ are sampled as:

$$W_i = \max(\kappa, U_i),\ i = 1, 2, \ldots, \tag{11}$$

where κ is a nonnegative constant, and $U_1, U_2, \ldots$ are independent and identically exponentially distributed with mean μ. Here μ^{-1} represents the unrestricted traffic intensity per unit time, while κ is a control parameter which can be used to limit the traffic into $\mathcal{A}$. For a given μ the objective is to find a suitable κ such that the probability of a risk event is small. In the simulations we let $\mu = 90$ seconds, while κ was varied between 10 and 50 seconds.

According to current aviation practice (see e.g., (Tomlin et al. 1998)), each flying aircraft should be given a certain protected zone shaped as a *virtual cylinder* centered around the aircraft, where no other aircrafts should be allowed. The cylinder should have a radius of 2.5 nautical miles (4,630 meters). The height of the cylinder varies from 1,000 ft to 4,000 ft depending on the location of the aircraft. In our context we say that a risk event occurs whenever an aircraft enters the protected zone of another aircraft. Thus, if two aircrafts fly through $\mathcal{A}$ at the same time, a risk event occurs when the distance between the aircrafts is shorter than 4,630 meters.

We simulated the model 2000 times. In each iteration we observed the air traffic in $\mathcal{A}$ for two hours. With the given arrival process this was sufficient to obtain satisfactory convergence of the various measures. The main results are listed in Table 1.

Table 1: *Estimated criticality and performance measures as a function of the control parameter κ.*

κ	10	20	30	40	50
$\bar{R}$	0.250	0.239	0.227	0.066	0.020
$\bar{D}$	12,108	12,288	12,548	12,840	13,196
$\bar{M}$	0.660	0.650	0.632	0.611	0.588
$\bar{T}_p$	144	144	144	144	144

We observe that the asymptotic risk fraction of time, $\bar{R}$, is obviously unacceptable when κ is 30 or lower. When κ is increased to 40 or 50, however, the risk fraction drops respectively to 6.6 percent and 2.0 percent. Taking into account that nothing has been done to the flight trajectories at this stage, this is far better. The sudden change in risk fraction is related to the heavy left-hand tail of the exponential distribution. For such a distribution the left censoring has a very noticeable effect.

Turning to the asymptotic average minimum distance, $\bar{D}$, we see that is increasing with increasing κ. This is of course hardly surprising since fewer aircrafts arriving into $\mathcal{A}$ implies that $\mathcal{A}$ will be less crowded. Still this effect is far less dramatic than the change in $\bar{R}$.

The results for the asymptotic average throughput, $\bar{M}$, listed as number of aircrafts processed per minute, indicate a decreasing tendency as κ increases. Again this is natural since this is directly related to the arrival process. With fewer aircrafts arriving $\mathcal{A}$, clearly the throughput must go down as well. Yet we notice that the effect on the throughput is modest.

Since nothing has been done to the flight trajectories at this stage, the asymptotic average processing time is constant for all κ.

4.2 *Avoiding risk events*

We now turn to the problem of avoiding risk events by modifying the trajectories dynamically. To simplify the problem slightly, we assume that the jth aircraft is flying at a constant speed of s_j meters per second as long as it is present in $\mathcal{A}$. Ideally each aircraft would prefer to fly from its arrival point to its departure point along a straight line. However, when other aircrafts are present in $\mathcal{A}$, it may be necessary to deviate from this path. Developing an automatic procedure for handling this is a complex task. One issue that makes this difficult, is that when more than two aircrafts are present in $\mathcal{A}$, handling one risk event may trigger another event. In our model we handle this issue by taking very small steps at a time. In each step each aircraft looks only for the closest aircraft in $\mathcal{A}$, and choses a safe path accordingly. Information about other aircrafts further away is discarded, at least until the next step.

The algorithm we are about to present, is similar to the "left", "straight", "right" algorithm discussed in (Tomlin et al. 1998). However, we allow the directions to be adjusted at each step of time. The main structure of the algorithm can be described as follows:

Algorithm 4.1 *For each point of time t and for each $j \in J(t)$ do the following:*

STEP 1. *Calculate the "ideal" velocity for the jth aircraft as:*

$$\dot{\boldsymbol{x}}_j(t) = \frac{\boldsymbol{X}_{d,j} - \boldsymbol{x}_j(t)}{\|\boldsymbol{X}_{d,j} - \boldsymbol{x}_j(t)\|} s_j. \tag{12}$$

STEP 2. *Identify the closest aircraft, and determine whether it is necessary to modify the trajectory in order to avoid a risk event.*

STEP $3a$. *If no action is needed, use the ideal velocity in the interval $[t, t+dt)$, and update the position to:*

$$\boldsymbol{x}_j(t+dt) = \boldsymbol{x}_j(t) + \dot{\boldsymbol{x}}_j(t)dt. \tag{13}$$

STEP $3b$. *If an action is needed, make a "turn". That is, replace the ideal velocity by:*

$$\dot{\boldsymbol{x}}'_j(t) = \Lambda \dot{\boldsymbol{x}}_j(t), \tag{14}$$

where Λ is the 3×3 horizontal rotation matrix given by:

$$\Lambda = \begin{bmatrix} \cos(\lambda) & -\sin(\lambda) & 0 \\ \sin(\lambda) & \cos(\lambda) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{15}$$

and where λ is the angle between $\dot{\boldsymbol{x}}_j(t)$ and $\dot{\boldsymbol{x}}'_j(t)$. Use the modified velocity in the interval $[t, t+dt)$ instead. Thus, in this case we get:

$$\boldsymbol{x}_j(t+dt) = \boldsymbol{x}_j(t) + \Lambda\dot{\boldsymbol{x}}_j(t)dt. \tag{16}$$

Note that since Λ is an *orthonormal* matrix, the speed of the aircraft is not altered by the multiplication. In order to make a *left* turn, one must choose λ between zero $\pi/2$. Similarly, a *right* turn is obtained by choosing λ between zero and $-\pi/2$.

In order to specify the algorithm further, we need to find a criterion (Step 2) for when an action is needed. Moreover, we must determine the matrix Λ as a function of the relative position and velocity of the two aircrafts (Step 3b).

To do so, we consider the jth aircraft denoted a_j at time t_0, and assume that we have identified the closest aircraft to a_j, say a_k. At this stage it is convenient to work with the relative positions and velocities of a_j as seen from a_k, denoted $\boldsymbol{x}_{j,k}(t)$ and $\dot{\boldsymbol{x}}_{j,k}(t)$ respectively, and given by:

$$\begin{aligned} \boldsymbol{x}_{j,k}(t) &= \boldsymbol{x}_j(t) - \boldsymbol{x}_k(t), \\ \dot{\boldsymbol{x}}_{j,k}(t) &= \dot{\boldsymbol{x}}_j(t) - \dot{\boldsymbol{x}}_k(t). \end{aligned}$$

The distance between a_k and a_j at time t_0 can then be written as $\|\boldsymbol{x}_{j,k}(t_0)\|$. If this distance is large, there is no need to modify the trajectory of a_j, at least not yet. Thus, as part of the procedure, we introduce an *alert distance*, denoted C_A. Obviously, in order to avoid risk events C_A must be chosen to be greater than the critical distance C. In the simulations C_A was chosen to be twice the size of C, i.e., 5 nautical miles (9,260 meters).

In the following we consider the case where $\|\boldsymbol{x}_{j,k}(t_0)\| \leq C_A$. We must then determine if the two aircrafts get closer or further apart as time goes by if no action is taken. That is, we consider the distance $\Delta(t) = \|\boldsymbol{x}_{j,k}(t)\|$. If no action is taken, we have:

$$\boldsymbol{x}_{j,k}(t) = \boldsymbol{x}_{j,k}(t_0) + \dot{\boldsymbol{x}}_{j,k}(t_0)(t - t_0). \tag{17}$$

Inserting this into $\Delta(t)$ and differentiating with respect to t, we find the point of time when $\Delta(t)$ is minimized. Denoting this by t_1 we get that:

$$t_1 = t_0 - \frac{\boldsymbol{x}_{j,k}(t_0)^T \dot{\boldsymbol{x}}_{j,k}(t_0)}{\dot{\boldsymbol{x}}_{j,k}(t_0)^T \dot{\boldsymbol{x}}_{j,k}(t_0)}. \tag{18}$$

From this it follows that $t_1 > t_0$, i.e., that the aircrafts are getting closer to each other, if and only if:

$$\boldsymbol{x}_{j,k}(t_0)^T \dot{\boldsymbol{x}}_{j,k}(t_0) < 0. \qquad (19)$$

The scalar product of $\boldsymbol{x}_{j,k}(t_0)$ and $\dot{\boldsymbol{x}}_{j,k}(t_0)$ is negative if and only if cosine of the angle between these two vectors is negative, i.e., if this angle is between $\pi/2$ and $3\pi/2$.

By inserting the expression for t_1 into (17) we get that the position where a_j is closest to a_k, is given by:

$$\boldsymbol{x}_{j,k}(t_1) = \boldsymbol{x}_{j,k}(t_0) - \dot{\boldsymbol{x}}_{j,k}(t_0)\frac{\boldsymbol{x}_{j,k}(t_0)^T \dot{\boldsymbol{x}}_{j,k}(t_0)}{\dot{\boldsymbol{x}}_{j,k}(t_0)^T \dot{\boldsymbol{x}}_{j,k}(t_0)}. \qquad (20)$$

Moreover, the minimal distance between a_j and a_k, $\Delta(t_1) = \|\boldsymbol{x}_{j,k}(t_1)\|$. If $t_1 \leq t_0$ or $\Delta(t_1) > C$, it is not necessary to change the trajectory of a_j, at least not yet. On the other hand, if $t_1 > t_0$ and $\Delta(t_1) \leq C$, some action is needed to avoid a risk event. As already mentioned, our approach to this is to make a "turn" by rotating the velocity vectors λ radians. To determine λ, we introduce the normal vector to the relative position vector, $\boldsymbol{x}_{j,k}(t_0)$, which we denote by $\boldsymbol{y}_{j,k}(t_0)$, obtained by rotation $\boldsymbol{x}_{j,k}(t_0)$ $\pi/2$ radians counterclockwise.

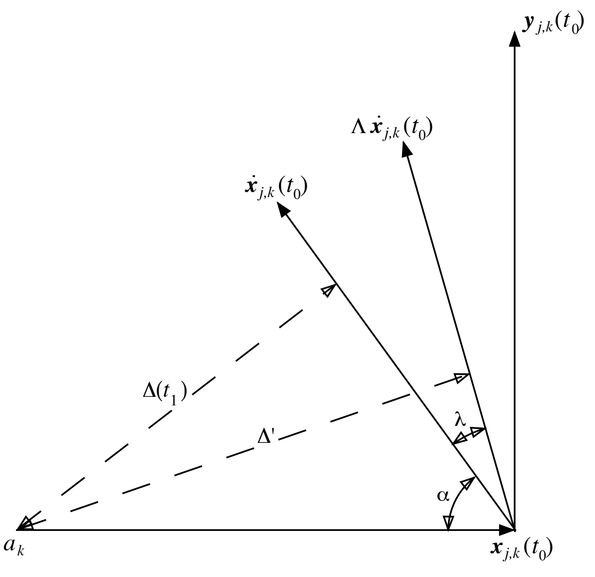

Figure 2: *Rotating the relative velocity*

We now consider three cases. In the first case, which is illustrated in Figure 2 we assume that $\dot{\boldsymbol{x}}_{j,k}(t_0)^T \boldsymbol{y}_{j,k}(t_0) > 0$. This corresponds to the situation where the angle between $\boldsymbol{x}_{j,k}(t_0)$ and $\dot{\boldsymbol{x}}_{j,k}(t_0)$ is between $\pi/2$ and π. In this case we let α be the difference between π and this angle. Thus, α is between 0 and $\pi/2$.

By rotating $\dot{\boldsymbol{x}}_{j,k}(t_0)$ clockwise λ radians, the minimal distance between a_j and a_k is increased from $\Delta(t_1)$ to some acceptable number $\Delta' > \Delta(t_1)$. We then have:

$$\sin(\alpha) = \frac{\Delta(t_1)}{\|\boldsymbol{x}_{j,k}(t_0)\|},$$

$$\sin(\alpha + \lambda) = \frac{\Delta'}{\|\boldsymbol{x}_{j,k}(t_0)\|}.$$

Hence, we get that:

$$\lambda = \arcsin(\frac{\Delta'}{\|\boldsymbol{x}_{j,k}(t_0)\|}) - \arcsin(\frac{\Delta(t_1)}{\|\boldsymbol{x}_{j,k}(t_0)\|}). \qquad (21)$$

In the second case we assume instead that $\dot{\boldsymbol{x}}_{j,k}(t_0)^T \boldsymbol{y}_{j,k}(t_0) < 0$. This corresponds to the situation where the angle between $\boldsymbol{x}_{j,k}(t_0)$ and $\dot{\boldsymbol{x}}_{j,k}(t_0)$ is between π and $3\pi/2$. In this case we let α be the difference between this angle and π. Thus, α is between 0 and $\pi/2$ this time as well. As in the previous case we can increase the minimal distance between a_j and a_k to some acceptable number $\Delta' > \Delta(t_1)$ by rotating $\dot{\boldsymbol{x}}_{j,k}(t_0)$ λ radians, where λ is given by (21). However, in this case the rotation must be done *counterclockwise.*

The final case covers the situation where $\dot{\boldsymbol{x}}_{j,k}(t_0)^T \boldsymbol{y}_{j,k}(t_0) = 0$, i.e., when the angle between $\boldsymbol{x}_{j,k}(t_0)$ and $\dot{\boldsymbol{x}}_{j,k}(t_0)$ is π. In this case it does not matter if we do the rotation clockwise or counterclockwise. As a convention we use a counterclockwise rotation here.

To complete the specifications we must choose the acceptable distance $\Delta' > \Delta(t_1)$. Obviously, Δ' should be greater than the critical distance, C. In the simulations we used the following rule:

$$\Delta' = \frac{\Delta(t_0) + C}{2}. \qquad (22)$$

Finally, note that so far we have only computed the rotation angle, λ, for the *relative* velocity vector $\dot{\boldsymbol{x}}_{j,k}(t_0)$. In order to achieve this rotation by modifying the trajectories of the aircrafts, we must rotate $\dot{\boldsymbol{x}}_j(t_0)$ and $\dot{\boldsymbol{x}}_k(t_0)$ the same angle. That is, we replace $\dot{\boldsymbol{x}}_j(t_0)$ and $\dot{\boldsymbol{x}}_k(t_0)$ respectively by $\Lambda\dot{\boldsymbol{x}}_j(t_0)$ and $\Lambda\dot{\boldsymbol{x}}_k(t_0)$, where the matrix Λ is given by (15). In Algorithm 4.1, however, only $\dot{\boldsymbol{x}}_j(t_0)$ is rotated. The reason for this is that the algorithm loops through all $j \in J(t)$. Thus, the rotation of $\dot{\boldsymbol{x}}_k(t_0)$ is handled when the trajectory of aircraft a_k is considered. Assuming that a_j is the aircraft closest to a_k, both velocity vectors will eventually be rotated correctly. When many aircrafts are present in $\mathcal{A}$, however, this assumption may not hold. Thus, as we shall see, Algorithm 4.1 works best when $\mathcal{A}$ is not too crowded.

4.3 *Simulation results using Algorithm 4.1*

In the remaining part of this section we illustrate the use of Algorithm 4.1 by presenting some simulation results. In Figure 3 we have plotted the trajectories of two flights, one northbound and one eastbound, both starting at the same point of time. Thus, if no action is taken, the aircrafts will collide at the center of $\mathcal{A}$. When the distance between the aircrafts becomes shorter than the alert distance, C_A, both aircrafts make left turns. As a result the trajectories follow a curved path until the distance between the aircrafts become large enough. When this happens, the aircrafts follow their ideal paths towards their respective departure points.

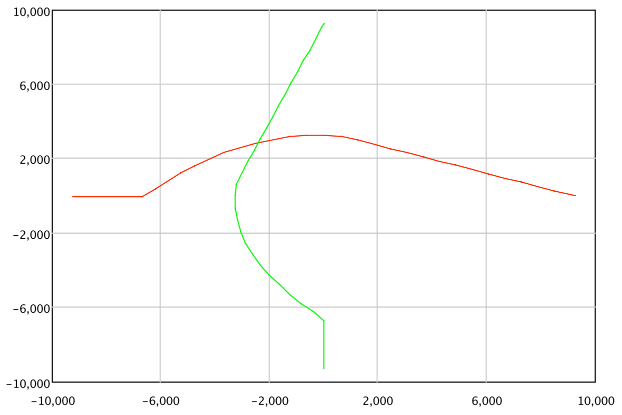

Figure 3: *Trajectories of two flights*

If the aircrafts do not start at the same time, the minimum distance between the aircrafts given no action is longer. As a result the modified trajectories become less curved, and the processing times become shorter. We ran six different simulations where we varied the time between the two flights from 0 to 50 seconds. In Figure 4 we have plotted the distances between the aircrafts as a function of time for these six simulations. The leftmost curve represent the case where the two flights start at the same point of time. All six curves has a minimum value equal to the critical value C, i.e., 4630 meters. Thus, when only two aircrafts are present in $\mathcal{A}$, the risk events are avoided completely by using Algorithm 4.1.

In the last simulations we ran the same scenarios as we did in Subsection 4.1. This time, however, we used Algorithm 4.1 to avoid risk event. The results are given in Table 2. Compared to corresponding numbers in Table 1 we see that the use of this algorithm has a substantial impact. When the control parameter κ is small, the algorithm actually has a strong negative effect. In particular, if $\kappa \leq 30$, the asymptotic risk fraction of time, $\bar{R}$, is close to 50 percent, which of course is completely unacceptable. However, if κ is increased to 40, $\bar{R}$ drops down to 0.6 percent. Moreover, if $\kappa = 50$, the occurrence of risk events is zero.

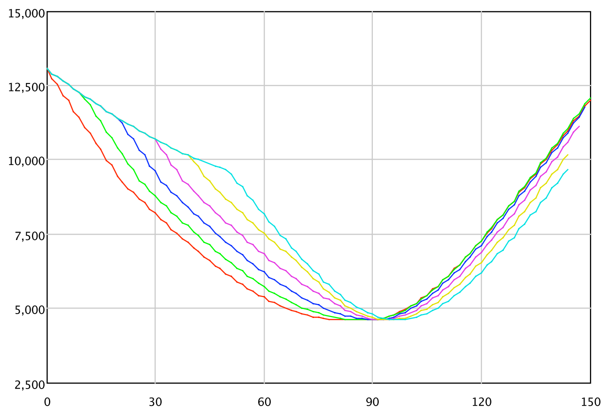

Figure 4: *Distance (meters) between flights as a function of time*

Thus, the arrival process has a dramatic effect on the usefulness of Algorithm 4.1.

A similar pattern holds for the asymptotic average minimum distance, $\bar{D}$. When $\kappa \leq 30$, $\bar{D}$ is below 9,000 meters, which is more than a 25 percent decrease compared to the corresponding numbers in Table 1.

For the asymptotic average throughput, $\bar{M}$, the effects are much less noticable. Still for $\kappa \leq 30$ the throughput is slightly lower than for case with unmodified trajectories. This is significant, though, since this indicates that processing does not keep up with the arrival of aircrafts into $\mathcal{A}$.

Finally, we observe that the asymptotic average processing time increases dramatically when $\kappa \leq 30$ compared to the case with unmodified trajectories.

Table 2: *Estimated criticality and performance measures as a function of the control parameter* κ.

κ	10	20	30	40	50
$\bar{R}$	0.480	0.477	0.472	0.006	0.000
$\bar{D}$	8,558	8,672	8,813	12,881	13,223
$\bar{M}$	0.652	0.641	0.624	0.610	0.586
$\bar{T}_p$	219	221	224	145	144

By taking a closer look at the simulation results, it is easy to see that the poor results for $\kappa \leq 30$ is due to the increased traffic density in $\mathcal{A}$. As the airspace gets more and more congested, the aircrafts must fly longer within $\mathcal{A}$ to avoid other aircrafts. Thus, the processing time increases as well, which increases the number of aircrafts present in $\mathcal{A}$. In fact the process is not stable within the simulation time frame.

In Figure 5 we have plotted the average minimum flight distances as functions of time for $\kappa = 10$ and $\kappa = 50$. While the curve for $\kappa = 50$ becomes stable almost immediately, the curve for $\kappa = 10$ decreases

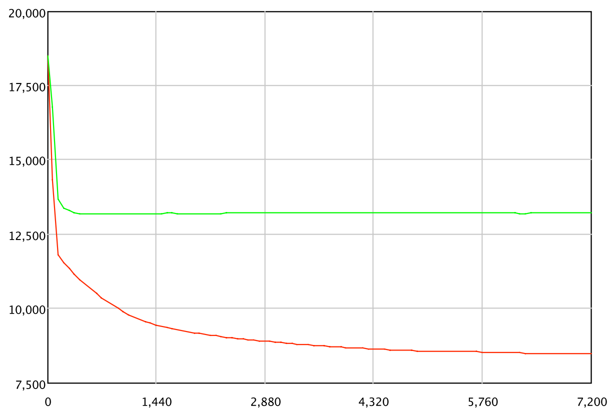

Figure 5: *Average minimum flight distances (meters) as functions of time for* $\kappa = 10$ *(lower curve) and* $\kappa = 50$ *(upper curve)*

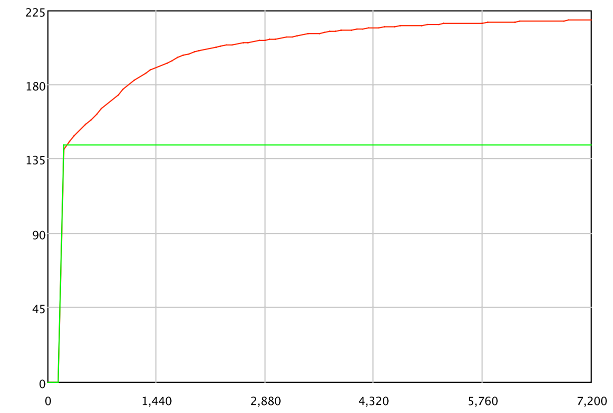

Figure 6: *Average processing times as functions of time for* $\kappa = 10$ *(upper curve) and* $\kappa = 50$ *(lower curve)*

throughout the simulation interval. This indicates that when $\kappa = 10$, the airspace gets more and more congested.

Similarly, in Figure 6 we have plotted the average processing time as functions of time for $\kappa = 10$ and $\kappa = 50$. Again the processing time is stable for $\kappa = 50$. For $\kappa = 10$, however, the processing time is increasing all the time. Thus, for $\kappa = 10$ the aircrafts tend to become more and more stuck within $\mathcal{A}$.

5 CONCLUSIONS

In this brief paper we have demonstrated how to build and analyse simulation models for aircraft trajectories from a risk perspective. The main sources of risk are the arrival process as well as the arrival and departure points within the airspace under consideration.

We have studied two issues related to air traffic control: managing the arrival process, and collision avoidance systems. One of the main findings in the simulation examples is that these two issues should be studied together. Developing a collision avoidance system without running this in an environment with stochstic traffic, may produce too optimistic conclusions. On the other hand analysing the risk related to the arrival process without implementing some sort of collision avoidance system may result in inefficient airspace utilization.

We would like to stress that the present paper is just a preliminary study. There are many model parameters that needs to be adjusted and optimized. Further work would include the study of different arrival processes and control mechanisms, as well as effects of different weather conditions. Moreover, the procedure used as a collision avoidance systems can be improved in many ways. In this paper we have used a distributed system where the aircrafts are responsible for choosing a sensible trajectory. By improving the coordination between the aircrafts, as well as including a central control unit managing arrivals and supporting the aircrafts may allow for a significantly improved utilization of the airspace.

REFERENCES

Bayen, A. M., R. L. Raffard, and C. Tomlin (2006, September). Adjoint-based control of a new eulerian network model of air traffic flow. *IEEE Transactions on Control Systems Technology 14*(5), 804–818.

Bernadsky, M., R. Sharykin, and R. Alur (2004). Structured modeling of concurrent stochastic hybrid systems. In *FORMATS/FTRTFT*, pp. 309–324.

Glover, W. and J. Lygeros (2004). A stochastic hybrid model for air traffic control simulation. In *Hybrid Systems: Computation and Control, Seventh Intl. Workshop*, pp. 372–386.

Koo, T.-K. J., Y. Ma, G. J. Pappas, and C. Tomlin (1997). Smartatms: A simulator for air traffic management systems. In *Winter Simulation Conference*, pp. 1199–1205.

Tomlin, C., G. J. Pappas, and S. Sastry (1998, April). Conflict resolution for air traffic management: a study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control 43*(4). Presented at the 1997 IEEE Conference on Decision and Control, San Diego.