You Make the Call: Feasibility of Computerized Aircraft Control

Richard D. Younger
Martin B. Linck
William P. Woesner
University of Colorado
Boulder, CO

Advisor: Anne M. Dougherty

Introduction

We investigate whether some of the work done by air traffic controllers (ATCs) could be handled by computers. Automated systems could act as watchdogs, heading off crises before they become catastrophes. Specifically, we investigate at what point an ATC must take charge of a situation to avoid catastrophe, what sort of decision must be made to remedy the situation, and how much stress is involved.

Objectives

- Define a minimum safe distance between aircraft.
- Develop a numerical model of air traffic around a busy airport.
- Assess system complexity and corresponding ATC workload under a variety of circumstances.
- Develop aircraft guidance algorithms that minimize controller stress.

The System

An ATC has three main tasks as an aircraft approaches an airport, all of which must be carried out as quickly as possible [Wood 1983]:

The UMAP Journal 21 (3) (2000) 285–300. ©Copyright 2000 by COMAP, Inc. All rights reserved. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice. Abstracting with credit is permitted, but copyrights for components of this work owned by others than COMAP must be honored. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior permission from COMAP.

更多数学建模资料请关注微店店铺“数学建模学习交流”
https://k.weidian.com/RHO6PSpA

- Ensure that the aircraft does not collide with another aircraft or any other obstacle.
- Ensure that the approaching craft is inserted smoothly into the traffic around the airport, with a minimum of disruption to the flight paths of other aircraft.
- Guide the aircraft onto a runway, again with a minimum of disruption to the rest of the traffic around the airport.

Special cases, such as aircraft experiencing mechanical malfunctions, medical crises, or fuel shortages, must be dealt with, and changing weather conditions must be taken into account.

To make the simulation concrete, we model Denver International Airport (DIA); our methods could be extended to nearly any air traffic control center.

Each ATC is assigned a specific type of task. Thus, one set of controllers assigns flight paths to incoming aircraft, another guides those aircraft to holding patterns or landing approaches, and yet another guides planes to a safe landing. As an aircraft passes from one controller to another, the pilots switch radio frequencies. Each frequency belongs to a specific controller, and each ATC watches a radar screen on which icons represent flights for which that ATC is responsible. The tower controllers, who are in charge of landings, can see the aircraft and so are not as dependent on radar information [FAA 1999].

At DIA, it is common practice to route all incoming flights on north-south and east-west vectors, since prevailing wind conditions usually favor these approaches [Wood 1983]. Departing flights must use the same runways as incoming flights but once airborne are routed out of DIA airspace on northeast-southeast and northwest-southwest vectors, to minimize conflicts between incoming and outbound aircraft. **Figure 1** shows a diagram of the general approach and departure vectors. To make its final approach and land on a runway, each aircraft has to pass a point in space approximately 5 mi from the end of the landing runway and approximately 2 mi (10,000 ft) above ground level.

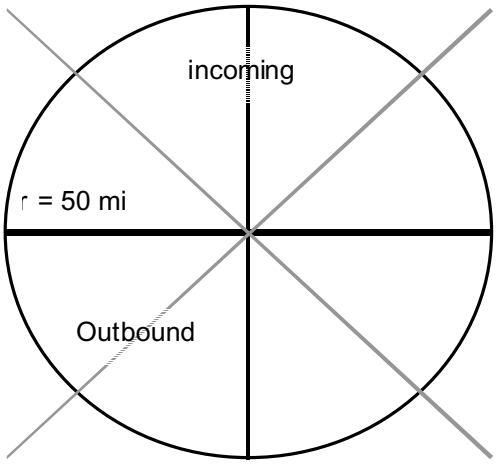

Figure 1. Airspace approach and departure diagram.

Assessing Safety

We use real data wherever possible; we consult the Federal Aviation Regulations (FAR) [FAA 1999] whenever technical questions arose.

Federally regulated Instrument Flight Rules (IFR) state that the minimum safe distance between adjacent aircraft is 1,000 vertical feet and 3 mi of horizontal distance, when aircraft are moving at landing speeds in close proximity to each other and to the airport. Since the runways at DIA are 4,330 ft apart, this minimum safe distance must be ignored on final approach and on runways.

Assessing Complexity, ATC Workload and Stress

Goode and Machol [1957], writing about large-scale queueing systems, describe complexity as "the extent to which any given attribute of a system will affect all the others if it is changed." In a complex system, all the variables are tightly linked; one could not, for example, change the position of an aircraft without immediately having to change some characteristic of most of the other aircraft in the system. Unfortunately, measuring this type of complexity is difficult, since there is no obvious set of measurable system variables for assessing how closely each variable depends on all the others.

Further, we are interested not merely in the complexity of the system but in the stress and fatigue that the system is likely to cause its ATCs. Surprisingly, there is no accepted set of factors that cause ATC stress, fatigue, and error. Some researchers, such as Redding [1992], have concluded that the number of incidents was highest when ATC workload was actually moderate to intermediate. On the other hand, Morrison and Wright [1989], reviewing NASA data, report that ATCs make more mistakes when the workload is at its highest. One innovative study of the phenomenon of ATC fatigue is that by Brookings et al. [1996]. Their test subjects were Air Force ATCs who were asked to play an air traffic control simulation and were exposed to scenarios of varying difficulty. One scenario was an "overload scenario," in which they were asked to coordinate the movements of 15 aircraft at once. As the ATCs attempted to deal with each scenario, their heart rates, blink rates, and brain activity were monitored. Brookings et al. concluded that there was a correlation between workload and operator stress and that the likelihood of error—in the form of separation errors (not enough room between planes), fumbled approaches, and botched handoffs—was directly linked to ATC stress. As a result, we assume for our simulation that ATC stress should be minimized and that there is no "optimal stress level" [Redding 1992] at which ATCs should operate.

Based on our literature survey, we conclude that there are four measurable factors that influence ATC stress levels:

- n, the number of total planes in the airspace around the airport.
- f, the number of separation errors currently occurring in the airspace around the airport. We assume that a single separation error causes as much stress

as 20 extra aircraft in the airspace.

- d, the *smallest* distance between planes currently on the map.
- a, the *average* distance between aircraft. If all the aircraft are well distributed throughout the airspace of the airport, the ATCs are likely to experience less stress than they would if they were all clustered together.

Our formula for the stress-causing complexity experienced by ATCs is

$$C = n + 20f + 100/d + 100/a.$$

Queueing Theory, Stochastic Input, and Algorithm Design

The airport is a queueing system. A queueing system has servers that handle input, perform some function on the input, and then pass the input to some other part of the system. Input is not created or destroyed in the system. The servers are usually referred to as *channels*. The five active runways of DIA are the channels of the system.

When the number of servers is inadequate to the amount of input they are called upon to handle, a queue develops. In the case of an airport, the holding patterns in which aircraft wait for clearance to land are the queues of the system.

When the amount of traffic is stochastic (determined by a probabilistic distribution) and the input is discrete, the input density is often described by a Poisson distribution. The amount of time for each channel to process an input need not be constant. The standard numerical approach to modeling would involve setting up an input generator, which would send us airplanes according to a density function. We would then set up our channels, and the amount of time to process each plane would vary, probably according to a normal distribution. We would also set up holding patterns, to which our planes could be sent when there are no runways available. We could then let the program run and see how queues develop as time passes.

Unfortunately, this simple approach doesn't allow us to investigate all the questions posed by the problem statement. The objective of the simulation should be to develop and test algorithms to monitor the position and speed of aircraft and to alter flight paths to minimize the workload and stress of ATCs. If we treat airplanes simply as inputs, without them becoming objects maneuvering in space, we cannot adequately test those algorithms. So, a more ambitious approach is called for, one that allows us to "create airplanes" stochastically, maneuver them, send them to holding patterns, land them, and assess the value of stress-causing complexity as the simulation runs.

The Model

An airport is a continuous system; each airplane continuously changes position and velocity. However, there are discrete events that characterize the system, such as takeoffs, landings, and handoffs. Both continuous and discrete characteristics of the system can be described by a discrete model provided the time resolution is fine enough. Recognizing this, we develop three numerical simulations of aircraft behavior, which test:

- the minimum-safe-distance assumption,
- guidance algorithms on aircraft landing, and
- guidance algorithms for aircraft entering and maintaining a holding pattern.

Minimum-Safe-Distance Simulation

Consider two aircraft traveling on parallel flight paths at an airspeed of 300 mph, well below the cruising speed of commercial airliners. Let the vertical axis be the z-axis, the axis parallel to the flight path be the x-axis, and the axis perpendicular to both be the y-axis. Wind and other factors perturb the velocity vectors of the planes according to (we assume) a normal distribution with mean zero. A large standard deviation might correspond to the planes flying in heavy weather, a small one to calm weather.

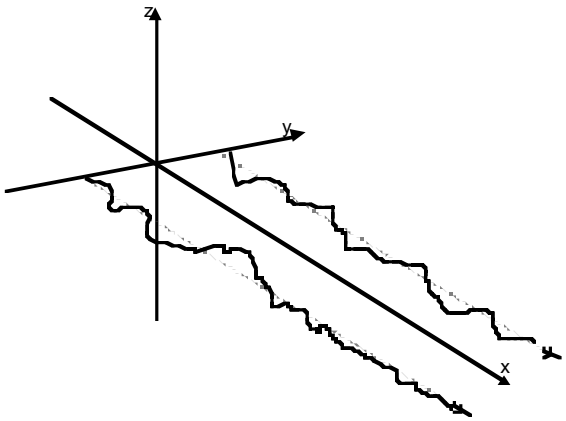

Figure 2. Aircraft separation simulation.

If we did not correct the aircraft's course, the plane would diverge from its flight path and move in a random walk as time passes. Our program determines the distance between the aircraft and its flight path and changes the velocity vector to bring the plane back on course. Airplanes have maximum rates of acceleration in any direction, and we assume that the aircraft can change velocity in any direction by no more than 10 mph/s. The result is a corrected random walk, with the step size normally distributed and a finite correction to each step.

Since the wingspan of an airliner is approximately 200 ft and turbulence effects surround the aircraft, we assume that if the airliners come within 250 ft of one another, they collide. We assume for the sake of simplicity that this holds true in the vertical direction as well. To test our minimum-safe-distance assumption, we fly planes next to each other for 100 h; if the likelihood of collision is less than 0.05%, we consider the distance safe for the given weather conditions.

Developing Aircraft Guidance Algorithms

A flow diagram for our airport is shown in **Figure 3**.

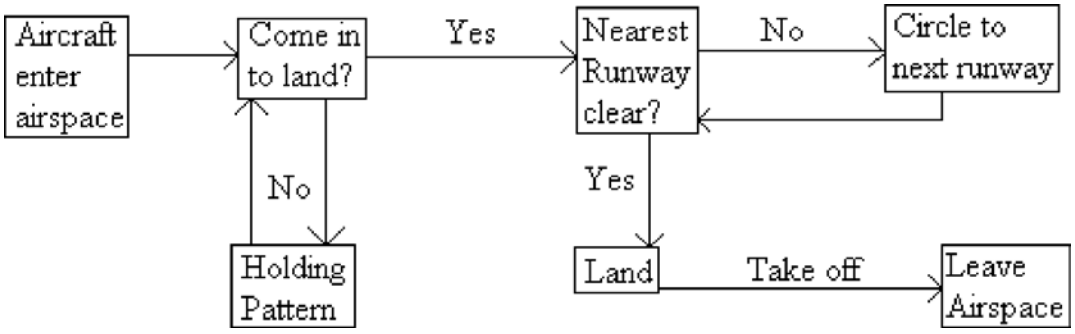

Figure 3. Flow diagram for an airport.

Airplanes enter the airspace according to a stochastic distribution. If a landing approach is free, they proceed towards it. If no runway is clear or if the airspace is too crowded, they are sent to a holding pattern (our queue). The amount of time in the holding pattern depends on the rates at which planes land and planes enter the airspace.

We assume that the scheme used to select the next aircraft cleared for approach to landing is FIFO (first in, first out). If we wished to take into consideration factors such as fuel or other flight emergencies, each plane would have to be assigned a priority and planes would be pulled from the queue according to priority.

Once an aircraft has been cleared for approach, it must select a runway. If the nearest runway is not free, the aircraft must circle until one is. Once it finds a clear runway, it may land. The runway is then occupied for some (perhaps stochastic) amount of time. The aircraft then spends some (perhaps stochastic) amount of time on the ground and takes off again.

The process can be divided into boxes, or areas. How much stress airplanes in a given area cause to the ATCs depends on the number of airplanes and the extent to which guidance algorithms manage to control the aircraft. Aircraft just entering or just leaving the airspace are unstressful, since they are presumably spaced out along a very large circumference. Aircraft within 5 mi of a runway that are coming in to land are admitted only when a runway is clear; they are handled by ATCs who monitor those flights visually as well as via radar. The

consequences of a mistake in this function are so high that we can safely assume that humans will handle this task for the foreseeable future.

There remain two areas in which our algorithms might ease controller stress:

- aircraft that have received clearance and must line up for a landing. **Figure 4** shows runway checkpoints, through one of which an aircraft must pass to land.
- aircraft that have not been cleared to land and must be sent to a queue. **Figure 5** shows the set of checkpoints that comprise the holding pattern. An aircraft can enter the holding pattern at any checkpoint, and, if properly guided, proceeds to the next checkpoint in the sequence. It cycles through the sequence until given clearance to approach. All aircraft move through the sequence in the same direction, to avoid head-on collisions.

We design algorithms to maintain aircraft spacing while guiding those aircraft, either toward a runway checkpoint, or through the circular set of checkpoints that form the queue.

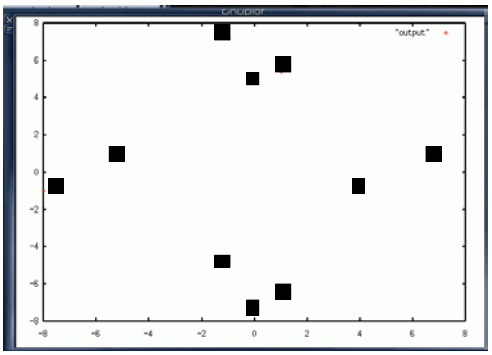

Figure 4. Runway checkpoints.

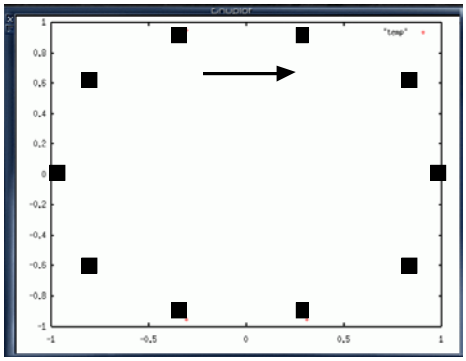

Figure 5. Queue checkpoints. Arrow indicates flight direction.

The Algorithms

Single-Avoidance

This algorithm determines the distance between each aircraft and that aircraft's next checkpoint and orients the aircraft toward the checkpoint. It also evaluates the distance between that aircraft all other aircraft; if any distance is equal to or smaller than the minimum distance (3 mi horizontal, 1,000 ft vertical), it then orients the aircraft directly away from the dangerously close airplane (the aircraft in the airspace longer changes course) without changing speed. Once the distance again exceeds the minimum safe distance, the aircraft that had to change its course looks for its nearest objective (which need not be

the same objective that it was originally approaching) and changes course toward that objective. An example of aircraft being guided toward a landing checkpoint by the single-avoidance algorithm is shown in **Figure 6**.

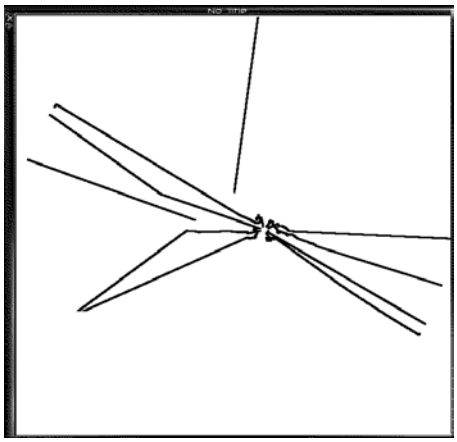

Figure 6. Aircraft converging on runway checkpoints while guided by the Single-Avoidance Algorithm.

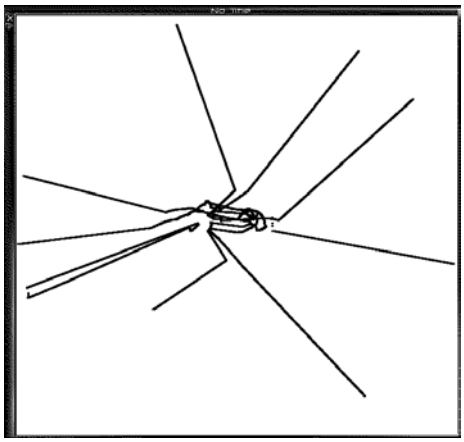

Figure 7. Aircraft moving toward landing checkpoints under the Double-Avoidance Algorithm.

Double Avoidance

If the distance between two aircraft drops below the minimum safe distance, *both* aircraft head away from each other, changing courses equally and in opposite directions without changing speeds. **Figure 7** shows an aircraft being guided toward a landing checkpoint by the Double-Avoidance Algorithm.

Single-Vector Repulsion

Unlike the first two algorithms, this algorithm constantly measures the distance between the aircraft under its control and that aircraft's nearest neighbor. It alters the course of that aircraft before a separation error can occur. The scheme used to correct the aircraft's flight path is as follows.

In each time step, the algorithm determines the direction in which the aircraft must fly to reach the nearest checkpoint. It thus establishes a velocity vector $\vec{V}$ for the aircraft, whose magnitude cannot change but whose direction is corrected in every time step. It then finds the vector $\vec{D}$ that connects the craft under guidance with its nearest neighbor and calculates a correction vector $\vec{A}$ whose direction is the same as that of $\vec{D}$ but whose magnitude is

$$||\vec{A}|| = \frac{b}{||\vec{D}||^2},$$

where b is a scaling constant. If b is large, the correction is severe, even at large distances; if b is very small, we run the risk that flight paths will not be adjusted

quickly enough and the aircraft will come to close to each other.

The velocity vector $\vec{V}$ is corrected to $\vec{V}_c = \vec{V} - \vec{A}$. The result is that every aircraft is repelled by the aircraft nearest to it, with increasing intensity as the distance between the adjacent aircraft becomes smaller. At the same time, the aircraft remains attracted to its objective.

If there are only two aircraft in the simulation, and both are headed for the same objective, they behave like a pair of negatively charged ions approaching a large positively charged sphere (this analogy breaks down when there are more than two aircraft). **Figure 8** shows aircraft maneuvering toward an objective while guided by the Single-Vector Repulsion Algorithm. Note that most of the aircraft maintain much better spacing than with the previous algorithms.

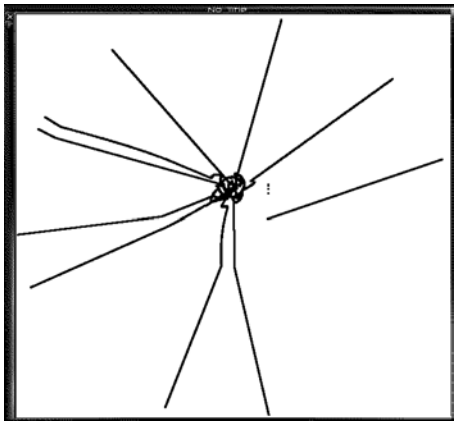

Figure 8. Aircraft maneuvering toward landing checkpoints under the guidance of the Single-Vector Repulsion Algorithm.

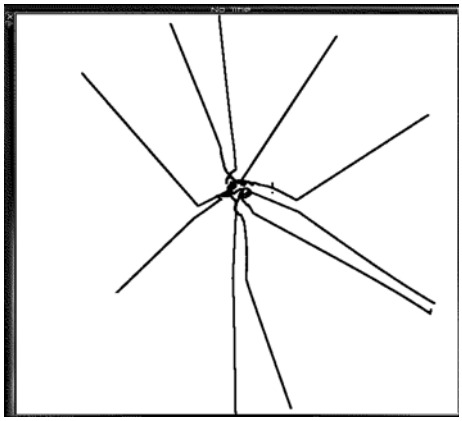

Figure 9. The Multiple-Vector Repulsion Algorithm.

Multiple-Vector Repulsion

The Multiple-Vector Repulsion Algorithm has the same vector mechanics as the Single-Vector Repulsion Algorithm, but each airplane is repelled not just by its nearest neighbor but by every other airplane in the airspace. The behavior of a number of aircraft headed toward a single objective would be roughly analogous to a group of negatively charged ions heading toward a large positively charged sphere. With more than one objective, however, the analogy is less apt, since each "ion" is attracted only to the nearest sphere. **Figure 9** shows 10 aircraft approaching a checkpoint under the guidance of the Multiple-Vector Repulsion Algorithm.

Testing the Algorithms

The Landing Approach Test

In this test, aircraft enter the airspace according to a normal distribution with mean 120 s and a standard deviation of 60 s. The points on the circumference of the airspace are evenly and randomly distributed. The airspace is centered on the airport and has a radius 50 mi. Each aircraft enters the airspace at an altitude of 6 mi (close to cruising altitude) and descends to one of 10 runway checkpoints (each of the 5 runways can be approached from either end), each located 5 mi from the end of a runway and at an elevation of 2 mi. Once an airplane reaches a runway checkpoint, its velocity instantly becomes zero. This ensures that no further aircraft can reach that checkpoint while the plane remains in place. After a set period of time (1 min in our simulation, based on the FAR [FAA 1999]), the airplane disappears and can be described as having landed. The runway is then clear and a new aircraft can occupy that checkpoint. The value for ATC stress is calculated at each time step.

The Queueing Test

Aircraft enter the airspace just as they do in the landing approach test. They descend to a set of queueing checkpoints and maneuver around those checkpoints in a holding pattern. Each holding pattern has a saturation level of airplanes, equal to the perimeter of the pattern divided by twice the minimum allowable distance between aircraft. Aircraft are added to the holding pattern stochastically until saturation is reached. We calculate the amount of ATC stress that the holding pattern generates. **Figure 10** shows 10 aircraft descending toward and entering the holding pattern under the control of the Single-Vector Repulsion Algorithm.

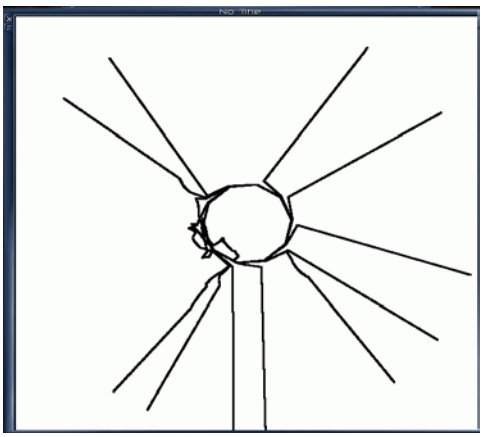

Figure 10. Aircraft being maneuvered in to a holding pattern by the Single-Vector Repulsion Algorithm.

Simplifying Assumptions in Testing the Algorithms

- All aircraft behave like commercial air carrier aircraft.
- The velocity of all aircraft is 300 mph.
- The minimum vertical separation between aircraft is 1,000 ft.
- The minimum horizontal separation between aircraft is 3 mi.
- The turning radius of all aircraft is negligibly small compared to the overall airspace.
- An aircraft has reached a checkpoint when it has passed within 1 mi of that checkpoint.
- Weather conditions do not affect the behavior of aircraft and are not considered.
- No aircraft is given any special priority over any other; fuel and emergency considerations are therefore ignored.
- There is no coordinating intelligence at work. The human ATCs can watch the simulation (and be stressed by it), but all actions of the aircraft are determined by the algorithm being tested.
- Aircraft are incapable of acting without instructions from the control algorithms. In essence, the pilot of each aircraft blindly and unquestioningly follows the orders given by the algorithm.

The following assumptions apply only to the Landing Approach test for all algorithms:

- A runway approach checkpoint remains occupied for 1 min.
- Any aircraft that reaches an approach checkpoint lands; there are no failed landing attempts.
- Once an aircraft has landed, it ceases to interact with any other aircraft and disappears from the simulation.
- Outbound aircraft do not interact with inbound aircraft and hence do not appear in the simulation. **Figure 11** shows that with our current checkpoint system, corridors develop in which there is no inbound traffic. So we assume that outbound traffic passes through these corridors and need not be accounted for.

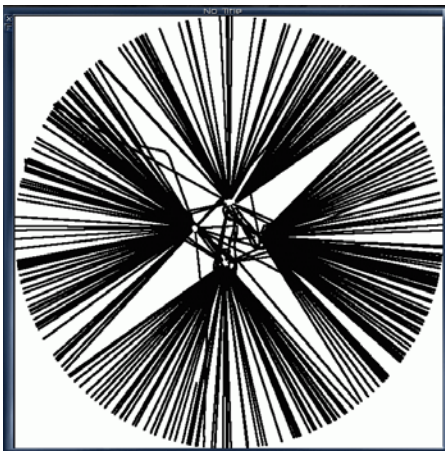

Figure 11. Typical traffic pattern when aircraft head toward the checkpoints from the periphery. Note the open corridors, along which outbound traffic can be routed.

Results and Commentary

Minimum-Safe Distance Simulation

Figure 2 shows the flight paths of two aircraft as they travel next to each other for 5 min, together with ideal flight paths that are separated by 3 mi. The results of many such simulations are given in **Figure 12**. Each data point represents the average of 150 runs, with each run lasting 100 h. The components of each aircraft's velocity are each disturbed by components with a mean of zero and with standard deviations as noted in the legend. The disturbances correspond to moderate, heavy, and severe weather conditions.

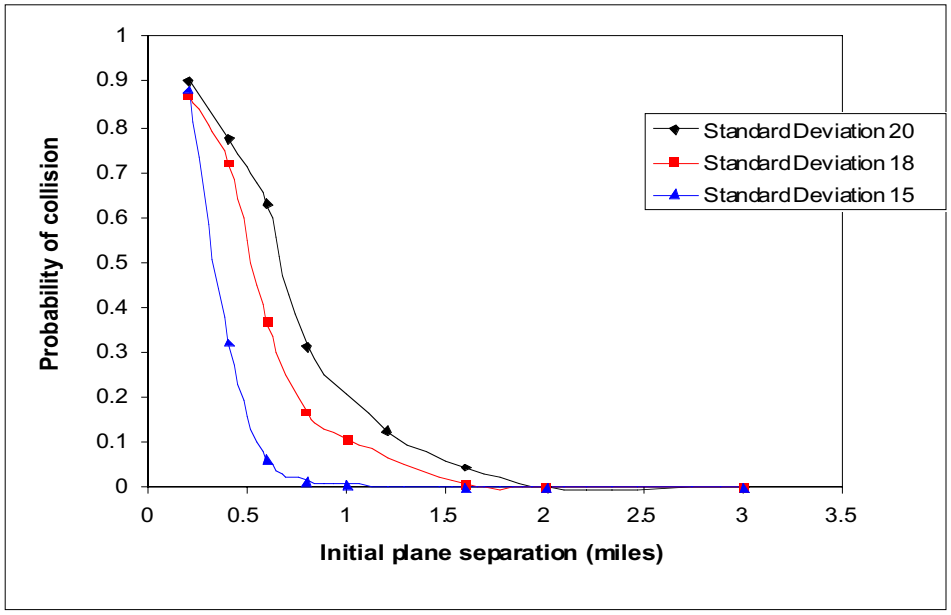

Figure 12. Probability of collision in a 100 h run as a function of initial separation.

The likelihood of collision is well under our 0.05% criterion when the separation distance is 3 mi. In fact, in the course of all 450 runs of 100 h, over all three disturbance levels, no collisions occur. The likelihood of collision increases exponentially as the horizontal separation distance decreases and becomes appreciable at 1.8 mi, where the first collision occurs. We conclude that FAA regulations give a secure safety margin and so abide by them for the rest of the simulation.

Holding Pattern and Landing Approach Simulations

There are no statistically significant differences among the stress levels for our four test algorithms, neither for holding pattern nor landing approach simulations. The holding pattern simulation achieves equilibrium with little to no stress and then the stress levels out. Since the landing simulation is much more complicated, one would expect sharp spikes and peaks in the stress, and this is what we see. **Figures 13** (generated using the Single-Vector Repulsion Algorithm) and **14** (generated using the Multiple-Vector Repulsion Algorithm) show sample graphs of stress vs. time for the queueing and landing simulations, respectively.

In **Figure 13**, the stress rises as planes are added; when the pattern is saturated, the stress level becomes steady. There are few collision warnings; the rise in stress is caused by the decreasing proximity of the planes as they approach the holding pattern.

Figure 14, however, is fascinating. For $0 < t < 5{,}000$, the graph is piecewise continuous, but the rest is totally discontinuous. In our definition of stress, there are two continuous terms and two discrete terms. Over the first 5,000 s of the simulation, the continuous terms play an important role in stress; but during the latter 5,000 s, the continuous terms die off, leaving the discrete terms to determine the stress.

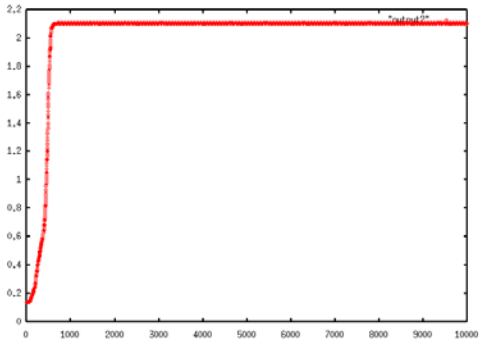

Figure 13. Stress vs. time for a queueing simulation.

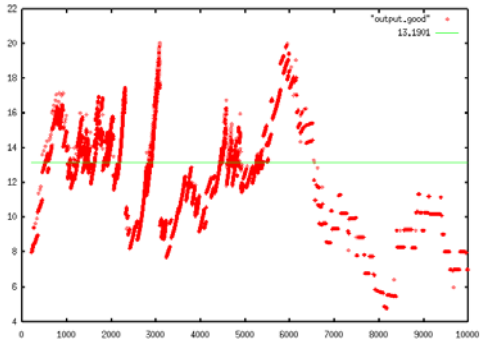

Figure 14. Stress vs. time for a landing simulation.

Table 1 gives summary statistics for both simulations run with each of the controlling algorithms. Each scenario was repeated 100 times. The seed for

the pseudorandom number generator was held constant across the controlling algorithms, so that each algorithm received the same input.

Table 1.
Descriptive statistics from the simulations.

	Landing Simulation		Queueing Simulation	
	Mean	SD	Mean	SD
Single Avoidance	12.8	0.8	4.1	1.9
Double Avoidance	14.2	1.7	4.1	1.9
Single-Vector Repulsion	14.2	1.7	4.1	1.9
Multiple-Vector Repulsion	13.2	1.6	4.8	2.2

The most interesting result is that the Single Avoidance Algorithm performs better than the other more clever algorithms. Here we can draw a parallel to the phenomenon from computer science known as *deadlock*. Deadlock occurs when two or more processes cannot continue executing because each process is requesting resources owned by the other process [Nutt 1999]. Despite much research and many clever algorithms, the standard way to handle deadlock in a computing environment is to just pick a winner, usually the process that has been waiting the longest. The Single Avoidance Algorithm is analogous: When two planes request the same airspace, only one can be awarded it; the algorithm picks a winner and vectors the loser away.

To make the Multiple-Vector Repulsion Algorithm more efficient, we increase the repulsion factor between planes; if planes are kept farther apart, they will not incur collision warnings. The idea is good but the results are not encouraging. **Figure 15** is a plot of stress vs. time for the landing simulation using the Multiple-Vector Repulsion Algorithm with increased repulsion.

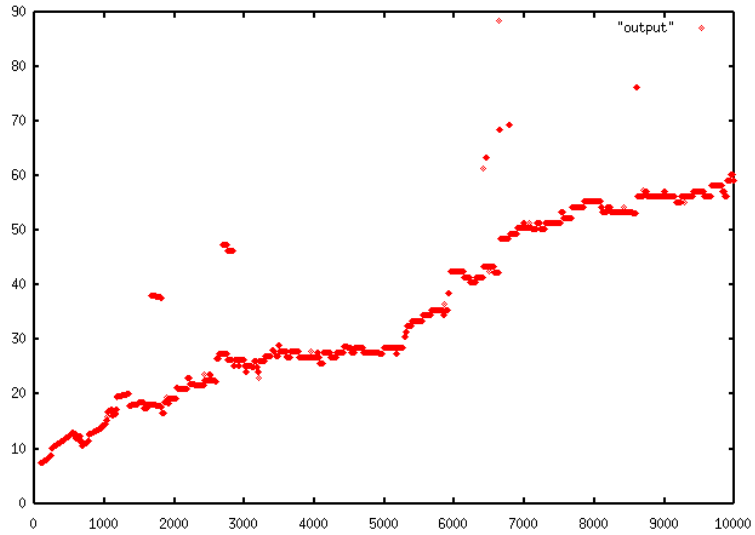

Figure 15. Stress vs. time for the landing simulation using the Multiple-Vector Repulsion Algorithm with increased repulsion.

We see a monotonically increasing stress level—certainly not the desired result. We can explain this phenomenon too in terms of deadlock. Imagine that two airplanes approach a checkpoint on opposite approach vectors. When they get close enough to each other, the repulsion factor makes them turn around. When they again get far enough away from each other, they turn around and fly back toward the checkpoint; and the cycle repeats. The planes become deadlocked, so very few planes can land. Hence, the number of planes in the airspace increases, leading to increasing stress.

The queueing simulations generate very little stress. Hence, software should be able to take control of a plane, vector it into a holding pattern, and keep it there.

Strengths and Weaknesses

More factors contribute to this system than we are able to consider in a weekend. However, our model is modularized to the point where modules can be run independently of each other, making it easy to focus on specific parts of the model.

It took our workstation approximately two hours to generate the data for our limited model, but parallel processors could each handle a set of planes.

Our model covers all of the major elements of an airport simulation; while it is based on DIA, it can easily be generalized.

Conclusions and Recommendations

We conclude from our simulation that *FAA guidelines for aircraft separation give a generous margin* for navigational, technical, and pilot error. Perhaps they could be relaxed, especially for well-controlled parallel landing situations in good to moderate weather.

The air traffic control problem exhibits many traits common to NP-complete problems:

- There is no deterministic way to pick the best solution.
- Humans appear to control air traffic much more effectively than computers could.
- Changes in the control algorithm for our simulation do not seem to impact the quality of the solution at all.

These features lead us to speculate that this problem is indeed NP-complete.

The computer does, however, handle the queueing problem fairly well; planes are vectored into the queue and held there with minimal stress. These results are promising; they suggest that *though computers should not replace humans as the primary means of air traffic control, computers might be capable of handling the more mundane tasks*.

References

Airports Data Base — Denver International Airport. 2000. `http://www.boeing.com/assocproducts/noise/denver.html` .

Brookings, J.B., G.F. Wilson, and C.R. Swain, 1996. Psychophysiological responses to changes in workload during simulated air traffic control. *Biological Psychology* 42: 361–77.

Federal Aviation Administration. 1999. *Federal Aviation Regulations*. Washington, DC: Government Printing Office.

Goode, Harry H., and Robert E. Machol. 1957. *System Engineering*. New York: McGraw-Hill.

Morrison, R., and R.H. Wright. 1989. ATC control and communication problems: An overview of recent ASRS data. In *Proceedings of the Fifth International Symposium on Aviation Psychology*, 902–907.

Nutt, Gary. 1999. *Operating Systems: A Modern Perspective*. Reading, MA: Addison-Wesley.

Redding, R.E. 1992. Analysis of operational errors and workload in air traffic control. In *Proceedings of the 36th Annual Meeting of the Human Factors Society*, 1321–1325.

Wood, Kenneth Ray. 1983. Airport activity simulation. Master of Science thesis, University of Colorado. Boulder, CO.