Cranfield University

Karen M. Feigh

An Airspace Simulator for Air Traffic Management Research

School of Engineering

Master of Philosophy Thesis

Cranfield University

School of Engineering

Department of Power, Propulsion, and Aerospace Engineering

Master of Philosophy Thesis

Submitted in Academic Year 2003-2004

by

Karen M. Feigh

An Airspace Simulator for Air Traffic Management Research

Under the Supervision of Professor David J. Allerton

November 2003

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Philosophy.

ABSTRACT

Presently the airspace system has reached capacity. Radical changes are required to take advantage of emerging technologies (to provide the dramatic increases in airspace capacity) to accommodate the projected increase in air traffic demand. Conventional commercial airspace simulators are too structured to simulate some of the unconventional proposals to increase airspace capacity. This has lead the development of dedicated simulators by NASA and EUROCONTROL to evaluate the potential capacity benefits of radical changes in the airspace structure. These proprietary, high-fidelity simulation tools require dedicated equipment and specially trained operators.

This thesis describes the construction of a high-level, low-fidelity non-proprietary airspace simulator for the purpose of conducting exploratory research into radical new approaches to enhance airspace capacity. The simulator developed is capable of simulating over 300 aircraft using the BADA v3.3 PTF performance database, while mimicking FMS guidance to navigate over a spherical earth. In addition, the simulation includes an observed winds aloft model, and an air traffic control model which provides conflict detection and avoidance guidance as implemented by an air traffic service provider.

Three scenarios (North Atlantic Endurance, Landing at Gatwick, Simple Europe) have been simulated to verify and evaluate the capabilities of the simulator. These scenarios show that the simulator has successfully implemented the BADA performance database and great circle navigation over a spherical earth. The scenarios also demonstrate that the impact of the wind model was small but noticeable. The impact of the air traffic control model is significant. The scenarios illustrated that the air traffic control model is capable of solving most of the conflicts it detects. However, the simulator's ability to merge arrival streams was met with limited success.

The simulator presented has the advantages over existing ATM simulators of being able to run on a single PC and in fast time. Being open source, it facilitates novel research through the software organisation and data structures. Additional advantages include implementing an observed wind field model, data link simulation and natural language style control logic.

ACKNOWLEDGEMENTS

I would like to extend gratitude to the many individuals who shared their time and expertise with me. Without their help this work would not have been possible. I am also grateful to the Marshall Commemoration Commission whose generosity enabled this work.

"A close accord between our two countries is essential to the good of mankind in this turbulent world of today, and that is not possible without an intimate understanding of each other. These scholarships point the way to the continuation and growth of the understanding which found its necessity in the terrible struggle of the war years."

George C. Marshall

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOTATION

A	Amplitude of the assumed wave four structure
D	Drag
ϵ	Phase angle
g	Earth's gravitational constant
h	Altitude above mean sea level
λ	Longitude
$\dot{\lambda}$	Time rate of change of longitude
m	Mass
ϕ	Latitude
$\dot{\phi}$	Time rate of change of latitude
ψ	Heading with respect to true north
R_o	Radius of the Earth
t	Time
Δt	Time step
T	Thrust
θ	Flight path angle
$\bar{U}$	Mean wind velocity over the longitude range of the original NPN region
$\bar{U_o}$	Mean wind magnitude for wave four wave structure
V_{CAS}	Calibrated airspeed
V_{TAS}	True airspeed
V_D	Up-down component of true airspeed, aircraft frame (Down is positive)
V_D^i	Up-down component of true airspeed, inertial frame (Up is positive)
V_E	East-West component of true airspeed, aircraft frame (East is positive)
V_E^i	East-West component of true airspeed, inertial frame (East is positive)
V_N	North-South component of true airspeed, aircraft frame (North is positive)
V_N^i	North-South component of true airspeed, inertial frame (North is positive)
V_{W_D}	Up-down component of wind, inertial frame (Up is positive)
V_{W_E}	East-West component of wind, inertial frame (East is positive)
V_{W_N}	North-South component of wind, inertial frame (North is positive)
x	Position
$\dot{x}$	Velocity
x_t	Position at time t

$\dot{x}_t$	Velocity at time t
x_{t+1}	Position at time t+1

CHAPTER 1

Introduction and Motivation

1.1 Air Transportation Capacity

In its short history, ATM (Air Traffic Management) has experienced several major paradigm shifts in response to capacity crises. The responses have come through the introduction of new technologies such as light guns, radio communication, radio beacons and radar [1]. The technological evolution has resulted in an airspace system comprised of a network of fixed routes and finite capacity. As a consequence the continued growth of the air transport industry has resulted in an overloaded air traffic system[2, 3]. Zeghal and Hoffman state,"The major challenge facing ATC (Air Traffic Control) is to enhance air traffic capacity and flight efficiency while providing safety improvements."[4] According to a capacity study carried out by Donohue and Laska in 2001, most European and U.S. airports are currently operating at or above 50% of their stated capacity[1], at which point ATM systems begin to experience significant delays, i.e. greater than 15 minutes [5]. When operating close to capacity limits it is common for demand to exceed the availability of runway and airway slots, resulting in delays which are attributed to ATM systems [6]. These delays can be found in all flight segments [7]. In addition to airport congestion, which dominates in the U.S., en route congestion currently accounts for 75% of delays in Europe [8]. These delays grew by 40% from 2000 to 2001 even though air traffic during this period only increased by 5% [8]. In order to reduce these delays, the capacity of the entire air transport system, not just at airports, must be increased [9].

Over the past decade, considerable effort has been expended on different methods to increase the overall capacity of the air transportation system [1, 10]. "The evolutionary path that ATC must follow is one of steadily improving the accuracy and integrity of the information base and making the decision process more optimal - until at some point the 'physical' capacity of airspace as defined by minimum separation criteria is reached."[3] Until the mid 1990s, these studies had focused

[1]It should be noted that European stated capacity is calculated based on IMC (Instrument Meteorological Conditions), and U.S. stated capacity is calculated based on VMC (Visual Meteorological Conditions).

primarily on improving the system (ground based and aircraft based) within the constraints of its current structure.

In ground based systems, research has been conducted on automation to aid controllers to regulate air traffic more efficiently [10, 11, 12, 13, 14, 15]. The two most extensive projects in this area are the CTAS (Center-TRACON Automation System) begun in 1989 in a joint program with NASA Langley and NASA Ames Research Centner, and the URET (User Request Evaluation Tool) developed by the MITRE Corporation. "CTAS is an integrated system... [compromised of three tools to] assist controllers in achieving greater efficiency in the management and control of arrival traffic in the extended terminal area, as well as assist in the conflict prediction and resolution of traffic along airway and user-preferred trajectories."[9] URET provides the controller with a 20-minute look-ahead conflict probe, and URET allows controllers to play what-if scenarios to avoid conflicts early on with minimal flight path impact [10].

In aircraft based systems, research efforts focused on the development of autopilot and FMS (Flight Management System) to enable more precise and fuel efficient navigation [16, 17, 18], as well as demonstrating the feasibility of data link systems to connect the autopilot and FMS to ground controllers and navigation aides [11, 19, 20]. Advances in these areas led the ICAO (International Civil Aviation Organisation) to embrace the concept of CNS/ATM, a concept which "envisages the use of data link communications, satellite navigation, and the ADS (Automatic Dependent Surveillance)."[21] Despite the advances in onboard avionics to improve aircraft fuel efficiency and navigational capability, the current air traffic management system is not capable of exploiting these advances [22].

A handful of studies, however have investigated a radical departure from the current airspace structure. Both NASA and several U.S. airlines conducted studies into the cost savings and feasibility of user preferred routing [23]. In addition EUROCONTROL initiated the PHARE (Program for Harmonised Air Transport Management Research) in 1989. In the mid 1990s Zegal and Hoffman stated, "the forecasted traffic density growth in Europe and in the United States over the next fifteen years suggests that solely improving ground systems might not be sufficient to achieve the required capacity at appropriate safety levels."[4] It had become apparent that improvements are also needed to the air route structure. "This network [of jet routes] imposes restrictions which lead to less efficient routing than director or wind-optimal trajectories."[24, 25] The current route structure also limits the

impact new technologies can make in the areas of tracking, prediction and communication [26]. One of the concepts that has gained considerable attention in the past decade is the the concept widely known as *free flight*.

In 1995, free flight was highlighted by the RTCA (Radio Technical Commission for Aeronautics), whose members include aircraft manufacturers, operators, air traffic controllers, and government agencies, as the way to increase capacity of the air transport system [27]. The report put forward by RTCA defined free flight as:

> ... a safe and efficient flight operation capability under IFR (Instrument Flight Rules) in which the operators have the freedom to select their path and speed in real time. Air traffic restrictions are only imposed to ensure separation, to preclude exceeding airport capacity, to prevent unauthorised flight through SUA (Special Use Airspace), and to ensure safety of flight. Restrictions are limited in extent and duration to correct the identified problem. Any activity which removes restrictions represents a move toward free flight.

The acceptance of RTCA's definition of free flight affirmed that the fundamental constraint of ATC is the safe separation of aircraft, and that any other constraints placed upon aircraft navigation were viewed unfavourably by the industry [28]. However, some researchers have claimed that there is a lack of quantifiable evidence in the public domain justifying free flight [29]. The definition put forward in the report was purposefully vague and open for interpretation. The report also failed to settle questions not only about the feasibility of mature free flight, but more importantly about, the transition to a mature free flight state.

After almost a decade of active research into free flight, its feasibility and implications, three overarching research areas have emerged.

- the technical aspects of airborne separation assurance and airspace optimisation
- the procedural aspects of the delegation of separation assurance and the redefinition for the responsibilities of aircrews and ATSP (Air Traffic Service Provider)s
- the human factors and workload aspects of the emerging roles of both air traffic controllers and aircrews

Research in these three areas has indicated the feasibility of free flight, although a range of free flight implementations has lead to various degrees of endorsement [8, 25, 29, 30, 31, 32, 33]. Ratcliffe showed that free flight in Europe would

result in fewer potential collisions than the current route structure, but that collisions in the current route structure are easier to resolve with altitude restrictions [29]. Magill also found that free flight would result in fewer potential collisions due to a reduction in airspace density [31]. However a study by Andrews & Welch [32] concludes that the variability in sector loading will be higher in free flight and that "simultaneous multiple conflicts will be a routine occurrence when traffic flow is not highly conditioned and may impose a limitation on the tolerable traffic loading."

The primary area of contention in free flight's ambiguity is the human factors element, where minor variations in the definition of free flight can lead to major effects on its effectiveness. Andrews & Welch conclude "reasonable assumptions regarding the amount of workload relief provided by decision-support tools leads to the conclusion that only limited growth can be accommodated by this means."[32] However, this view is refuted by Hoekstra et al. [33] who concluded, "none of the studies [both off-line and man-in-the-loop studies of no-ATC, mature free flight] could refute the feasibility of an airborne separation assurance concept." Standardised airways simplify the job of the controller by limiting the trajectories that aircraft may follow, thus also limiting the number of trajectories liable to conflict. Eby concludes that "Indeed, without the use of airways, the current ATC system would simply be unable to handle typical daily traffic."[25] Several major programs and projects have investigated variations of the free flight concept in both Europe and the U.S.

In Europe most of the research has been conducted by EEC (EUROCONTROL Experimental Center). A major program conducted by the EEC throughout the last decade was PHARE. The free flight concept behind PHARE was an evolutionary approach increasing the accuracy and availability of information, leading to a greater precision in aircraft position and thus increasing the capacity of the system [3]. Under PHARE, each aircraft in the EUROCONTROL region would be assigned a 4D trajectory in the form of a tube-in-the-sky, which it would follow to a specific navigational precision [34]. The PHARE program provided the basis for the European Commission's AFAS (Aircraft in the Future ATM System) program. The free flight concept behind the AFAS program is a time-based ATM where "first-come, first-served" scheduling is replaced by first-planned, first served scheduling in which equipped aircraft receive priority [35]. The notion of *equipped* aircraft is important because of the extensive range of aircraft avionic capabilities. These range from simple magnetic compasses to sophisticated flight management systems which integrate satellite and inertial navigation systems. One of the key

issues facing any change in airspace structure is how to accomplish the transition with a fleet of mixed equipage. One solution put forth is to restrict large sections of airspace to aircraft with the proper equipage for free flight.

In the U.S., both NASA and the FAA (Federal Aviation Administration (U.S.A.)) have conducted research with their own versions of free flight. NASA's approach to free flight is based on DAG-TM (Distributed Air-Ground Traffic Management) [36]. The premise behind this version of free flight is that aircraft will not be autonomous in a small region segregated airspace, set aside for free flight, but that distributed responsibility for traffic management would be between aircraft and ground-based controllers enabling free flight to be applied to a larger area of airspace [36]. The FAA is overseeing the development of a set of tools to aid in the transition towards free flight called FFPI (Free Flight Phase I), which concluded at the end of 2002. The free flight concept behind FFPI is yet uncertain, as it is primarily concerned with developing a set of core technologies to enable a variety of free flight concepts to be implemented [37].

It is clear from the current literature that radical changes in the fundamental airspace infrastructure are underway in both Europe and the U.S. However it is yet unclear what form the future structure will take as the FAA, NASA, Boeing and EUROCONTROL are each pursuing their own vision [38, 39, 40]. This structure will be greatly influenced by the emergence and maturation of new technologies, such as data link. The next section will provide some background information on the technologies currently deployed, and Section 1.3 will discuss the future technologies likely to be deployed to replace the current generation of technology.

1.2 Current Air Traffic Management Environment

The current ATM system consists of a rigid route structure where aircraft are constantly under ground based control. The system operates in the following manner. A commercial transport aircraft will begin its flight by advising the control tower of its intention to execute a flight plan, which has been scheduled in advance with the ATSP. On most modern aircraft, this flight plan is also loaded into the aircraft FMS. The flight will then be given clearance by the tower to taxi and take-off following a SID (Standard Instrument Departure). The flight is controlled by the tower until it has cleared tower airspace, at which point, control of the aircraft is handed off to the TRACON (Terminal Area Controller) to complete the climb-out and turn onto its initial jet route. Once clear of terminal air space, the aircraft enters centre

airspace and is now under the control of a sector controller. The aircraft remains in centre airspace, under the control of different sector controllers as it traverses its flight path from one jet route to another. At sector boundaries and the convergence of airways, the aircraft may be issued with speed and altitude restrictions. Following the established jet routes, the aircraft follows a flight path, which may not necessarily be a direct routing. When the aircraft approaches its destination, it reenters TRACON airspace and begins a STAR (Standard Terminal Arrival Route), where it is merged into a single stream of arriving aircraft before being cleared to land. When the number of arriving aircraft is too great, the aircraft may be fed into a holding pattern. The holding pattern is used to evenly space randomly arriving aircraft in order to maximise runway throughput. A rigid route system has been historically necessary to maintain aircraft separation; however it comes at the cost of inefficiency resulting in added flight time.

ATM can be generally characterised by three components: communication, navigation and surveillance, often referred to collectively as CNS. All three components are currently ground-based and the collaboration between aircraft or air carriers and ATSPs is limited [41]. The different facets of the CNS (Communication, Navigation and Surveillance) system are available to and are utilised by ATSPs, flight crews, and AOC (Airline Operational Centers)s, who share responsibility for safe, economic and efficient air transport. A major limitation of the current air transport system is that the three key participants in the ATM system does not have access to the same set of information. For example, radar track information is known to the ATSPs, but not the aircrew or the AOCs. The limitations imposed by these different facets, coupled with the rigid route structure and limited capacity of the current ATM system, have led to a very conservative use of airspace. The following three sections will describe each aspect of the CNS system.

1.2.1 Communication

Ground-to-ground voice exchange is carried out over telephone lines, but ground-to-ground data exchange is far more complicated. In Europe, two different systems are currently available for ground-based data exchange, the AFTN (Aeronautical Fixed Telecommunications Network), and the SITA (Société Internationale de Telecommunications). In the United States AFTN is also used along with AviNet. AFTN has been available for several decades, and is capable of providing ICAO-format flight plan data to even the smallest airports. However, smaller airports tend

to have older versions of AFTN hardware, and as a consequence are "not capable of supporting the efficient exchange of data required by the present ATS (Air Traffic Services) system."[42] AFTN is used primarily by the area control centres and air traffic service providers, but not airline operation centres.

In Europe AOCs prefer to use the SITA network. SITA provides a "global network for data communications" and allows the data to be formatted by the user [42]. In the U.S. AOCs and airports often use AviNet, which provides data communications, message switching and content services using a variety of customisable data formats [43]. Because of the number of different systems available, there are effectively two different sets of data being employed in ATM, one for ATSPs and one for AOCs. As each network carries a different set of information, both ATSPs and AOCs could benefit from a unified communication system.

Air-to-ground/ground-to-air voice communication includes HF (High Frequency), VHF (Very-high Frequency), and UHF (Ultra-high Frequency) analogue and satellite digital signals. Civilian aircraft primarily utilise the VHF spectrum and reserve the HF spectrum for long distances over water [1]. The UHF spectrum is reserved for use by military aircraft. Air crews must tune their radios to the correct frequency for each ATSP along their route [42]. In modern aircraft equipped with a FMS, the radios are tuned automatically. While tuned to a specific frequency, the air traffic controller and all the air crews on that frequency and in range, hear all messages passed over the channel. All communication on the channel is available to anyone listening to the channel. This 'party-line' effect is the main source of situational awareness for pilots. In order to minimise miscommunication, both controllers and air crew follow a standard set of procedures and phraseology [1].

The ACARS (Airborne Communications Addressing and Reporting System) is the only widespread air/ground data link, but is not approved for flight critical messaging. ACARS utilises both VHF radio and satellite communications links and the information is fed into the appropriate ground-to-ground data network SITA or ARINC depending on location.[42]

1.2.2 Navigation

Traditionally aircraft have relied on ground based radio beacons and inertial navigation systems to navigate. Modern aircraft combine traditional methods with newer technologies and navigational aids including satellite based radio navigation

to provide greater accuracy and a higher level of redundancy [7].

The primary method of navigation for commercial aircraft consists of beacon-to-beacon navigation. The U.S. maintains a system of federal airways, compromising VOR (VHF Omnidirectional Range), DME (Distance Measuring Equipment) systems, TACAN (Tactical Air Navigation) and combinations thereof. Depending on altitude, these highways-in-the-sky are designated as either 'victory' airways up to 18,000ft MSL (Mean Sea Level) or jet routes between 18,000ft to 45,000ft MSL. Similar air routes used in Europe are designated airways [44]. Currently VOR/DME navigation is approved by the ICAO as the primary means of terrestrial navigation [1].

1.2.3 Surveillance

ATM currently uses a combination of primary and secondary radar for surveillance purposes. Primary radar provides the controller with aircraft range, azimuth, and in some cases elevation, whereas secondary radar includes the aircraft's altitude and transponder identification, which enables each aircraft to be uniquely identified. Secondary radar utilises a variety of modes, A, C, and S [7]. Mode C and S provide the pressure altitude referenced to 29.92in (1013mm) of mercury in 100ft increments from 1,000-126,700ft [7]. Mode S is the current standard for civil transport aircraft [45].

1.3 Future Air Traffic Management Environment

The capacity crisis, which currently threatens both European and U.S. airspace, reached a temporary climax (in the U.S.) in the summer of 1998 [46] and has become the impetus for the transition from airspace-based ATM, where route structures are based on fixed ground stations, to trajectory based ATM, where route structures are more fluid and can take advantage of area navigation [22]. This transition is being facilitated by advances in CNS technologies which are blurring the traditional boundaries between communication, navigation and surveillance as most of these new technologies are designed to serve multiple CNS functions.

1.3.1 Flight Management Systems

In order to increase both efficiency and safety, modern aircraft are routinely equipped with FMS to guide aircraft along a desired flight plan. According to Kayton & Fried the FMS has three specific capabilities relevant to ATM [7]. The first is its ability to accurately guide aircraft along a specific path through a series of two- or three-dimensional (2D or 3D) way points. The second is minimising the cost of a flight by selecting optimal speeds or altitudes for fuel efficiency. The third is its ability to meet a RTA (Required Time of Arrival), often referred to as four-dimensional (4D) navigation capability.

Flight management systems hold databases of static information such as locations and frequencies of navigational beacons, dynamic information such as the current flight plan, and predicted wind velocities along the flight path. The FMS integrates data from avionics systems to acquire current position, velocity and attitude, and generates altitude, heading and throttle commands to guidance systems. FMS architectures have been developed with data link interfaces to allow the FMS to transmit current location, flight plan and wind data and to receive new flight plans [7, 35]. However most of the data link capabilities of FMS are under utilised by current ATM systems. With the increase of dataflow in the future airspace system, however, FMS is likely to become more critical to all three areas of CNS.

1.3.2 Navigation

Navigation technology is based on satellite radio navigation augmented by onboard internal navigation systems coupled with a flight management system. The main form of satellite navigation to date is the GPS (Global Positioning System) established and maintained by the U.S. Department of Defense. GPS will soon be reinforced by the Europe's Galileo system, consisting of twelve medium earth orbit satellites. Ochieng et al. conducted a simulation study of the combined accuracy and integrity of the combined GPS/Galileo system and concluded that the horizontal error can be reduced to under 6m, and vertical error can be reduced to under 10m respectively. Even more important however is the improvement in integrity monitoring, where the a combination of GPS and Galileo systems, leaves, "only a few isolated points having availability below 99% and a majority of areas having between 99.7 and 100% availability." [47] This capability will allow aircraft to use satellite navigation as a primary means of navigation, which will in turn allow radio

beacons to be removed [48].

Equally as important as changes to navigation aids is a change to the manner in the way that navigation requirements are specified. The new concept is known as RNP (Required Navigation Performance). RNP "is a statement of the required navigation accuracy required for operation within a defined airspace."[7] RNP type is based on navigation performance accuracy, i.e. the difference in lateral distance between true aircraft position and measured aircraft position, and is stated in nautical miles [49]. In addition to accuracy RNP also specifies the integrity, continuity and availability requirements [47].

Changing to RNP will allow aviation authorities to certify the overall navigation system accuracy instead of specific pieces of navigation equipment; however, an aircraft can only be RNP certified if it has an FMS capable of calculating the total system error of the aircraft.

1.3.3 Surveillance

In addition to primary and secondary radar, future surveillance will also depend on information transmitted by aircraft. This technology is known as ADS (Automatic Dependent Surveillance). Two versions are currently proposed: ADS-A for communication addressed to specific aircraft and ADS-B for broadcast communication. ICAO specifies that ADS systems must transmit at least the 3D position of the aircraft, time of day and aircraft identification [7]. The difference between ADS and the current surveillance systems is that the quality of the data processed by a surveillance system is highly dependent on the quality of the aircraft's sensors [7]. For older, more poorly equipped aircraft, relying on the ADS-B may decrease the accuracy of surveillance data, but for modern aircraft, it will most likely improve it, as "the navigation accuracy of today's state of the art aircraft well exceeds that of ground based radar."[35] ADS also provides a suitable medium to transmit the information gathered by FMS to both ATSPs and AOCs. ADS reporting intervals are expected to range from 10s in enroute phases of flight to 4s in terminal areas [7].

1.3.4 Communication

The ATN (Aeronautical Telecommunications Network) is proposed for the future of both ground-to-ground and air-to-ground communications [7, 42]. The ATN will allow ground, air-ground and avionics sub-networks to communicate via data

link using the "International Organisation of Standardisation (ISO) open-system interconnection (OSI) 7-layer protocol architecture."[7] This system should enable AOCs, ATSPs and aircrews to access the same set of information, thus ensuring more efficient operations. The airborne segment of the ATN is proposed to provide ADS, which will broadcast current aircraft state, and possibly the next trajectory change point or wind data, to both ATSPs and AOCs. The technology used to implement ADS or the broadcast version, ADS-B (Automatic Dependent Surveillance Broadcast) is still under discussion in Europe; however, in 2002 the FAA published a series of reports endorsing 1090ES (Mode-S Extended Squitter) and UAT (Universal Access Transceiver) as it's choice for ADS-B technology [21, 50, 51].

Voice communications will also be included in the ATN data network. "The controller-pilot data link communications (CPDLC) service is planning to use the ATN and is designed to replace the controller/pilot dialogue under certain conditions." [52] After conducting data link experiments onboard NASA's 737, Knox and Scanlon concluded that the use of data link for routine tactical and strategic communications with ATC resulted in a perceived reduction in pilot workload and a reduction in crew communication errors [20]. It is believed therefore, that messages exchanged via data link will "reduce communication errors and relieve overloaded ATC voice radio frequencies."[20] Dieudonne, Joseph, and Cardosi performed a comparative risk analysis between today's voice system and the proposed data link using CPDLC (Controller-Pilot Data Link Communications) and similarly concluded that, "the probability of an error occurring via the mechanism of delivery has been reduced by orders of magnitude."[53] However concerns have been raised about the potential workload implications and loss of situational awareness caused by the lack of the party-line that such a transition may cause [54]. These concerns lead Sigmore and Hong to investigate the possibility of adding party line data to a data link network, which they deemed feasible. Additionally mixing voice and data link communications poses its own problems. Lozito et al. found that when time pressure was applied to flight crews using a mixed voice data link environment resulted in longer transmission times and more instances of missed message loading [55].

1.4 Existing Airspace Simulators

The widespread implementation of satellite navigation, flight management systems and automatic dependent surveillance systems is the foundation for fundamental

change to the air traffic system. However, as the cardinal tenet of ATM is to maintain or improve the current safety level, [4, 7] while maintaining the efficiency of the air transportation network, it is not feasible to "test out" new ATM schemes without extensive simulation [4, 7]. Currently two simulation packages have been developed commercially, to enable ATSPs to develop new procedures.

1.4.1 Commercial ATM Simulation Tools

Two simulation packages are commercially available for air traffic management research: TAAM (Total Airspace & Airport Modeler) and RAMS (Reorganised ATC Mathematical Simulator). TAAM is a commercial package developed by The Preston Group (TPG) in association with the Australian Civil Aviation Authority, and is maintained by TPG, currently a subsidiary of The Boeing Corporation. RAMS was developed by EUROCONTROL and is maintained under the name RAMS Plus by ISA Software. Both TAAM and RAMS are discrete event simulators capable of running in FAST-TIME (Simulating one second in less than on second of real time), and both are widely used by industry.

TAAM

TAAM is used primarily as a planning tool to conduct analysis and feasibility studies of ATM concepts. TAAM has the ability to simulate most ATM functions in detail including ground movement and handling. Additionally, it can generate scenarios for real-time ATC simulators. TAAM specialises in simulations that cover the entire ATM process from gate to gate [56].

Due to the size and complexity of performing a large scale simulation of an Air Traffic system, TAAM requires comprehensive input data files describing the entire air traffic system. TAAM allows these files to be configured for simulation of varying levels of fidelity to allow, "better modelling of critical areas."[56] The inputs can be specified on either a system or sector wide basis.

TAAM is capable of generating 2D or 3D graphical visualisation of the simulation, viewable in several independently scalable windows simultaneously. TAAM also provides a range of output data choices, which can be tailored for specific research needs. TAAM's main strengths lie in its ability to perform ground and terminal area simulations. The latest version of TAAM allows for limited weather

modelling using winds aloft in sectors, SIGMET (Significant Meteorological Information) and METAR (Meteorological Airfield Report). TAAM's principal areas of application have been: [56]

—Airport capacity (gate, taxiway, runway capacity)
—Planning airport improvements, extensions
—Noise impact
—Impact of severe weather
—Design of terminal area procedures (SIDs/STARs)
—Design of terminal area ATC sectors Controller workload assessment
—Impact of new ATC rules, e.g. reduced vertical separation.

RAMS

RAMS Plus is a fast-time discrete-event simulation software package which, like TAAM, is used as a planning tool to conduct analysis and feasibility studies of ATM concepts. RAMS has placed special emphasis on being able to study and analyse airspace structures, ATC systems and future ATC concepts, but lacks the ability to model ground and terminal areas to the same level of fidelity as TAAM [57]. RAMS has the ability to be run in real time as well as FAST-TIME, thus allowing human interaction [58]. Like TAAM, RAMS requires comprehensive input data files describing the entire air traffic system. Input files necessary to run RAMS include an airspace description, flight plan description, a rule based ATM system, workload analysis and weather pattern files.

RAMS is capable of generating a 2D graphical visualisation of the simulation, in a single scalable window. RAMS also provides a range of output data choices, which can be tailored for specific research needs. RAMS' main strengths lie in its ability to perform rule-based conflict detection and avoidance during the en-route phase of flight. The latest version of RAMS includes a limited convective weather model represented as dynamic forbidden-zones [58]. RAMS' principal areas of application have been: [58]

—ATC Workload
—Free routing investigation
—Free flight investigations
—Airspace capacity, density

Limitations

Many compelling arguments exist for using either TAAM or RAMS for ATM research. They are both high-fidelity simulation tools capable of simulating large numbers of aircraft. Both tools have been extensively verified for accuracy directly by the developer and indirectly by the end users. However, these tools also present many challenges to researchers as a consequence of their inherent complexity. Donohue and Laska found that TAAM and RAMS, "require significant amounts of data that are sometimes difficult to obtain."[5] They also stated, "Learning to use these models takes considerable time and effort limiting their use to specialised individuals."[5] Additionally, RAMS and TAAM are closed source tools, thus eliminating the possibility of extending their capabilities to novel research applications. Consequently, it is possible neither to simulate data link equipped aircraft nor to pass messages between system. This limitation prevents RAMS and TAAM from begin used for work such as that suggested by Wichman, Carlsson and Lindberg, who conducted experiments in feeding back observed wind data between pairs of aircraft to improve the performance of an experimental FMS [35]. Furthermore TAAM is prohibitively expensive for independent researchers.

1.4.2 Non-commercial Simulation Tools

In order to investigate innovative concepts researchers also use non-commercial ATC simulation tools. Unfortunately tools of this nature are difficult to find, and the ones mentioned in the literature, which are not proprietary, often have undesirable characteristics. First, as most of these simulation tools are written for a specific research application, they are often not portable nor readily extendable. Secondly, they often come with no technical support or documentation, and these simulation tools are often of unknown quality.

In the literature review undertaken for this research, several airspace and ATM simulators were identified. However many of the smaller and lower fidelity simulation tools were only referenced as background to conflict detection and avoidance studies, such as the general purpose research flight simulator used by Funabiki and Muraoka to investigate potential human error in the cockpit [59] or the Complete Air Traffic Simulator used by Alliot and Bosc to study the impact of reduced vertical separation minimums (RVSM) [60]. Other references outline methods for

developing simulation tools, such as the Aircraft Construction Tool (ACT) developed by CAE Electronics, which developed a general aircraft model to simulate different aircraft types from data available in *Jane's All the World's Aircraft*. These simulations appear to lack documentation in the public domain.

Even references to some of the larger, high fidelity tools currently in development at NASA, EUROCONTROL and the NLR (National Aerospace Laboratory (The Netherlands)) are obscure and difficult to obtain. It is apparent from the literature available that NASA is developing a high fidelity FFSIM (Free Flight Simulation) which will incorporate a number of smaller simulation tools with the intent of modelling the future of free flight airspace in the U.S. [61, 62]. However, according to Schleicher and Davis, FFSIM lacks a wind model and is not yet complete [62]. It seems however that NLR has already developed such a simulator, called NARSIM (NLR's Air Traffic Control Research Simulator), which they have been using to simulate new operational procedures and new controller assistance tools, and to develop human machine interfaces for the new Amsterdam Advanced ATC system [63]. EUROCONTROL also has its own airspace simulator called EUROCONTROL Simulation Capability and Platform for Experimentation (ESCAPE), which has similar capabilities to NARSIM. All of these systems, are restricted to the current state of the art and are complex pieces of software which run over multiple computers, in a dedicated setup. Consequently, they are not suitable for exploratory research applications.

1.5 Case for a Non-proprietary, Open Source Airspace Simulator

After considering all the airspace simulators found in the literature, it is clear that a high-level, low-fidelity airspace simulator did not exist. Moreover, it was also apparent from the literature that research into radical changes to increase airspace capacity were needed and on-going. This need had sparked the creation of several non-commercial airspace simulators by NASA, NLR and EUROCONTROL, as both TAAM and RAMS were found to be too restrictive to investigate novel methods to improve capacity [64]. However, the complexity, and somewhat proprietary nature of these non-commercial (academic) simulation tools, limits their use is restricted to large research programs associated with these organisations. There is, therefore, a need for a simpler, more flexible and more accessible simulator for exploratory

research into radical changes to increase airspace capacity that is especially suited to small research teams.

1.6 Simulation Constraints

In order to take full advantage of creating a high-level, low-fidelity simulator, it was important to meet the following constraints:

- ❖ Run on a single personal computer (PC)
- ❖ Run faster than real-time
- ❖ Be Open source
- ❖ Be Nonproprietary

The first two computational constraints make the creation of efficient data structures and algorithms essential. Having a simulator that can run faster than real time, but requires several computers working in parallel would limit the usefulness of the simulator. Conversely having a simulator that can run on a single PC but takes several days to compute a six hour simulation period would also be less than ideal. The second two constraints have little impact on the simulation's computational aspects, but require a clear modular design so that the simulation may be easily adapted to different research applications.

1.7 Research Objectives

The research presented in this thesis has a single overarching aim:

> To create a new simulation tool for the purpose of conducting exploratory research into radical new approaches to improve airspace capacity with the following capabilities and characteristics:
>
> - ❖ Develop discrete event simulator
> - ❖ Simulate atmospheric conditions
> - ❖ Simulate data link communications
> - ❖ Simulate ATC guidance
>
> while meeting the constraints outlined above.

CHAPTER 2

Airspace Simulator Requirements

Due to the complexity of existing simulation tools, a high level, low fidelity simulator was developed to meet the objectives outlined in Section 1.7. This simulator is designed to model possible ATM structures in the 2020 time frame. Section 2.1 in this chapter outlines the simulator's requirements in terms of capability, fidelity, and speed. Section 2.2 describes the performance database employed. Section 2.3 describes the programming language and computer platform. Section 2.4 outlines the simulator's organisational structure. The overall assumptions upon which the simulator is based are presented in Section 2.5. Finally, Sections 2.6 and 2.7 describe the inputs and outputs of the simulator.

2.1 Simulation Requirements

The simulation requirements can be organised into three areas: simulation capabilities, speed and fidelity. The simulation capabilities summarise the functional requirements of the simulation tool and are discussed in Section 2.1.1. The speed and fidelity requirements address the simulation performance requirements and are discussed in Section 2.1.2. Table 2 summaries the simulation requirements discussed below.

2.1.1 Capabilities

The air traffic simulator has four distinct functional requirements, covering the simulation of specific aircraft and the simulation of the natural and manmade environments in which the aircraft operate:

- *To efficiently simulate aircraft performance through an adequate dynamics model.* As the simulation tool is intended to facilitate research into future air space concepts, this requirement must include the simulation of the performance characteristics of multiple aircraft simultaneously. Additionally, the

simulation tool must to allow the manipulation of the aircraft dynamic models while the simulation is in progress through guidance updates. This element of the simulation tool is critical in the evaluation of information feedback through data link systems. Information feedback over data link will play an important role in future airspace systems as the information passed will provide the primary source of information for communications and surveillance.

- *To simulate the airspace environment in which the simulated aircraft will be operating.* The aircraft within the airspace need to be organised so that the they may be monitored and controlled by an air traffic control algorithm. Additionally, the simulation tool needs to be capable of simulating real-time broadcast and monitoring of simulation parameters. The simulation tool must be able to locate individual aircraft so that data-link transmissions can be transmitted between aircraft. Additionally, the simulation tool needs to broadcast a much more diverse set of information than RAMS or TAAM over a simulated data-link network. In addition to the information such as current position, velocity and heading, the current wind readings, intent information, and flight phase must also be provided.[20, 65, 66].
- *To realistically simulate the atmospheric environment in which the aircraft operate.* The atmospheric environment includes both natural phenomenon, e.g. wind fields and weather phenomena. Variability and uncertainty in wind fields have been shown in the literature to cause significant variation in aircraft performance [12, 67, 68]. These environmental factors therefore need to be simulated to enable the effect of atmospheric variation to be investigated.
- *To simulate the air traffic management structure, which includes airspace sectorisation, airport location, and air traffic control procedures such as altitude and speed restrictions.* The air traffic management procedures need to be flexible and capable of handling a wide range of existing and experimental procedures.

2.1.2 Speed and Fidelity

In a simulation environment, speed refers to the number of computations the simulation is capable of running during a specified period of time. This speed is governed by several factors including the time step used, the data structures used, the logical structure of the simulation, and the computer hardware i.e. the processor speed.

Table 2: Simulator Requirements

Simulator Requirements	Level of Necessity
Handle 300 aircraft simultaneously	Desired
Handle 4000 aircraft total	Desired
Capable of running FAST-TIME	Necessary
Capable of running on a standard PC	Necessary
Allows data link simulation	Necessary
Allows intent broadcast	Necessary
Allows aircraft state broadcast	Necessary
Allows current wind broadcast	Necessary
Allows flight phase broadcast	Desired
Allows data link feed back	Necessary
Allows wind field simulation	Necessary
Includes conflict detection and avoidance algorithms	Necessary

The time step refers to the amount of time covered in one simulation step. The choice of time step is directly related to the fidelity of the simulator; the smaller the time step the greater the fidelity. However, the speed of simulation does not effect the fidelity of the simulator. The speed of a simulator is classified as either fast, real or slow time.

In simulators which run in *real time*, one second of simulated time is equivalent to one second of real time, regardless of the time step chosen. A real time simulator that has a time step of one second computes the calculations required for each time step, once per second, but a real time simulator with a time step of half a second will compute the calculations twice each time step. Consequently, for a simulator to run in *fast time* with a time step of half a second, then the calculations need to be computed faster than every half second such that one second of simulated time is computed faster than one second of real time. Similarly simulators which run in *slow time* compute one second of simulated time slower than one second of real time.

The fidelity of a simulator is a relative description of how accurately it models the motion of the entity it is simulating. The fidelity level of a simulation is often tied to the performance model used, which in turn dictates the time step allowable. The time step chosen must be sufficiently small so that error introduced by the discretization of the simulation can be neglected. Hoffman summarises the following performance model classification system suggested by Renteux:[69, 70]

- ❖ Class A: Full flight dynamical models, utilising six degrees of freedom (DoF) equations of motion incorporating translational and rotational motion.
- ❖ Class B: Point mass models, utilising three DoF equations of motion covering translational motion.
- ❖ Class C: Parametric models, where the actual forces, i.e. lift, drag, thrust, are not considered, but the resulting acceleration and speeds are modelled.
- ❖ Class D: Fixed models, where horizontal and vertical speeds are extracted from tables as functions of altitude and flight phase.

Using this classification system, Class A simulators afford the highest level of fidelity and consequently have the slowest speed and require the smallest time step. Fidelity impacts on speed because of number of calculations computed during a single time step increases with the level of fidelity. While increasing fidelity decreases speed, several other aspects of a simulation also impact on speed. Any increase in the number of calculations per time step, decreases the simulation speed. Therefore additional time can be added by inefficient data structures which require multiple calculations to locate and access data required. Consequently, a simulator with a higher fidelity level and more efficient data structures and architecture can run faster than a simulator with a lower fidelity level simulator with less efficient data structures and architecture.

The simulation tool required for this research programme needs to be sufficiently powerful to simulate large numbers of aircraft simultaneously while running in fast-time. It decided that the minimum requirements for the simulation should be derived from the projected 2020 traffic level for the busiest airport in Europe, London's Heathrow Airport, as the that airspace would be the most congested airspace in Europe. It was further decided that the minimum number of concurrently simulated aircraft should be equivalent to two hours of the 2020 Heathrow daily peak traffic. Additionally, the minimum number of total simulated aircraft should be equivalent to 12-16 hours of 2020 Heathrow traffic. Specifically the simulator needs to be able to model at least 4,000 aircraft in total and 300 aircraft simultaneously over a 12-16 hour period [6].

Although the requirement for speed is in direct opposition to the level of fidelity of the simulator, the widespread use of FMS, allows the individual performance characteristics of aircraft to be captured with relatively few parameters, i.e. mass, true air speed at cruise, rate of climb or descent, and fuel flow rate as a function of altitude. By assuming an aircraft is equipped with an FMS the assumption

is made that an aircraft will behave in a uniform fashion because the FMS will handle all of the lower level control functions which would otherwise vary among aircraft types. It was concluded that aircraft under FMS control, could be adequately modelled using either a Class B, (3 DoF (Degrees of Freedom)) [71], Class C (parametric) or Class D (tabular) model [72].

2.2 Performance Database: BADA

EUROCONTROL has developed and maintains a performance database of the coefficients for a Class B (3DoF) and Class D simulation. This database contains tabulated performance information for over 186 different aircraft types and is called the BADA (Base of Aircraft Data) dtabase [72]. The BADA database is based on the energy formulation outlined below:

$$(T - D)V_{TAS} = mg\dot{h} + mV_{TAS}\frac{dV_{TAS}}{dt} \quad (1)$$

where T is Thrust; D is Drag; V_{TAS} is the true airspeed; m is aircraft mass; $\dot{h}$ is the time rate of change of altitude and g is the Earth's gravitational constant. Since its inception BADA has become the industry standard for aircraft performance modelling in Europe [72]. The current version of the BADA database, 3.3 incorporates 186 aircraft types [73], which constitute over 90% of the aircraft using EUROCONTROL airspace [74]. As the BADA database is readily available, and well documented in the literature [42, 33, 74, 75, 76], it was decided to use the BADA 3.3 as the basis for the aircraft performance model.

The BADA database is made up of three separate text files for each supported aircraft type:[72]

- OPF (Operations Performance File)
- APF (Airline Procedure File)
- PTF (Performance Table File)

The OPF contains the performance data specific to the aircraft type such as aerodynamic parameters, flight envelope, thrust, fuel consumption and ground handling, which allow a 3DoF model to be constructed [72]. The APF "specifies recommended speed procedures for climb, cruise, and descent conditions," for each

aircraft type. The OPF and APF contain the basis of a 3DoF simulation and they are used to derive the PTF, which "specifies cruise, climb, and descent performance at different flight levels,"[72] for each aircraft type. These three files form an aircraft performance model. Depending on the type of simulation required different BADA files are required. For example, a Class B simulation uses the APF and OPF files, and a Class D simulation includes the PTF for each aircraft type used. The format of these files is given in the BADA User's Manual [72]. To minimise the computational load, it was decided that the simulation should use the PTF portion of the BADA database to create a Class D dynamics model.

2.3 Programming Language and Operating System

The efficiency of software is affected by the capabilities of the language in which it is written. The simulator created for the proposed research is written in the Modula-2 programming language. Modula-2 provides several advantages over programming languages such as C, C++, or FORTRAN. Modula-2 allows the use of specialised data types, modular design, and pointers similar to C and C++, but with much stricter syntax and data type compatibility requirements. Arguably, Modula-2 is easier to learn and less prone to programming errors. In addition both the flight simulators at Cranfield University are written in the Modula-2 language, compiled under the MS-DOS operating system with a compiler known as Stony Brook Modula-2 (SBM). Because DOS has been phased out in recent versions of the Microsoft's Windows operating systems, and Stony Brook Modula-2 only supports a 16-bit environment, it was decided to use the GPM (Garden's Point Modula-2) compiler under the Linux operating system. Linux is an open source operating system, which is inexpensive and widely available.

2.4 Simulation Organisation

The simulation architecture is modelled according to the functionality required by the research specified in Section 2.1. The research proposed requires a multi-aircraft performance model, a weather model, an airspace model, and an ATC model. Therefore the overall system structure includes primary modules corresponding to

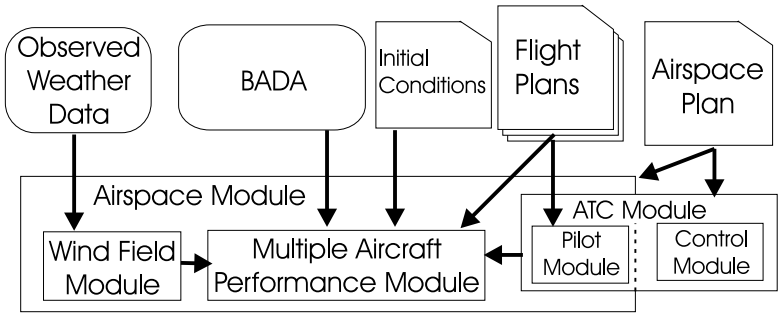


Figure 1: Simulation Architecture

each modelling requirement. The modularity is designed to accommodate future changes and enhancements to the simulator. This structure is shown in Figure 1.

2.4.1 Aircraft Performance Model

The aircraft performance module forms the basis of the simulation tool. Its objective is to model aircraft equipped with 3D or 4D FMS navigating over a spherical earth using either conventional or satellite navigation to follow a flight plan made up of set way points. It takes in data from the BADA input files and the weather module. The ATC module interacts with the performance model by modifying the individual aircraft flight plans. It outputs the aircraft position, altitude, velocity, heading, rate of climb or descent.

2.4.2 Atmospheric Environment

The atmospheric environment module supplements the aircraft performance module by generating the wind velocity components which are added to the aircraft's velocity (which is relative to the air) to obtain the aircraft's inertial velocity, or velocity relative to the Earth-fixed frame. A wind field model is required in order to accurately simulate aircraft motion, because the variability in winds aloft causes trajectory deviation [9, 30]. The wind velocity is imported from external wind files and is output to the aircraft performance model.

2.4.3 Airspace Model

The airspace environmental module includes the atmospheric and aircraft performance modules. The airspace module's objective is to efficiently organise all aircraft in the simulator in terms of their geographic location to facilitate data-link transmissions and ATC control. It is necessary to simulate data link transmissions because future ATM systems will rely on data link information as their primary means of communication and surveillance. Providing an efficient data link simulation is essential to ensure that only the correct aircraft receive the appropriate messages. Without such a system the computational load required to simulate a data link network would be prohibitive.

2.4.4 Air Traffic Control

The ATC module monitors the airspace module and maintains the safe separation of aircraft. An ATC module is required to accurately model the ATM environment, which operates under ground control and to allow different types of air traffic control to be simulated. The ATC module uses the data structure maintained by the airspace module to organise the aircraft into discrete control sectors. The ATC module takes in data from the ATC plan and issues the appropriate commands to the airspace module to alter specific aircraft flight plans. It is important to note that this module appears split in Figure 1. This model has two distinct functions, which correspond to the functions of a pilot and a controller. The Control Module is completely separate from the Airspace Module, and only interacts with the Airspace Module through a simulated data link with the Pilot Module. The Pilot module therefore serves as an interface for the Control Module to access the Airspace Module in order to implement the commands that the Control Module generates.

2.5 Simulation Assumptions

Throughout the simulation the assumption is made that aircraft will operate under FMS guidance. It was felt that this was a legitimate assumption for the 2020 time frame, as most air carriers actively encourage the use of FMS to minimise fuel burn and because future RNAV requirements will require an FMS. Thus it is assumed that each aircraft behaves in a fairly uniform fashion because the lower level control loops, which are specific to each aircraft type, are assumed controlled

by the autopilot function as a part of the flight management system. Consequently, all turns were modelled as rate-one turns (3^0 per second), and all transitions from cruise to climb or descent were modelled as instantaneous.

2.6 Simulator Initialisation

The simulation tool requires three different input file types: initial conditions file, the airspace plan file and the flight plan files. Depending on the simulation application, the number of the initial condition files and flight plan files will vary. A separate initial condition file is required for each flight plan used during the simulation run. Additionally a separate flight plan file is required for each aircraft type and flight plan used during a simulation run. For example a simulation set up that includes two different flight plans and three different aircraft types will have two initial condition files and six flight plan files. These initialisation files will allow aircraft to be automatically created in the correct proportions throughout the simulation run. Examples of these input files can be found in Appendix B.

2.7 Simulator Outputs

The output capabilities required by the simulation tool are straight forward, as they involve standard metrics such as aircraft state, creation and removal times. The simulation tool records the position, altitude, velocity, heading, rate of climb or descent of each aircraft at each time step. Additionally the simulation needs to record the actions of the ATC module, and the commands that are issued. These outputs allow researchers to gather the data necessary to determine the effectiveness of different airspace structures and ATC schemes.

CHAPTER 3

Aircraft Performance Module

The core module of the simulation is the Multiple Aircraft Performance Module, as every other module interacts with it. This chapter will describe the development and evaluation of the module. The first two sections present the module's requirements and assumptions. The next section discusses the necessary data structures. The fourth section presents the logical structure of the module, and the final section provides a validation of the module's performance abilities. The organisation of this module is illustrated in Figure 2.

3.1 Requirements

The Multiple Aircraft Performance module needs to meet four requirements. First, it must accurately implement the BADA performance database. Second, it must be capable of simulating multiple aircraft types simultaneously. Third, it must be capable of simulating aircraft flying multiple flight plans simultaneously. Fourth, the multiple aircraft performance module needs to model spherical navigation, so that the aircraft may fly as they would under FMS guidance.

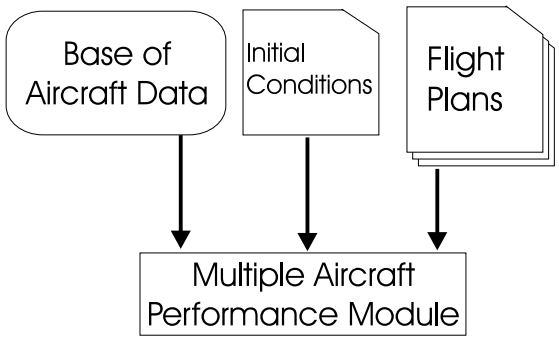


Figure 2: Simulator Architecture

3.2 Data Structures

The core of the Multiple Aircraft Performance Module is the individual aircraft performance model, which requires a comprehensive set of aircraft dynamic coefficients or a database of performance characteristics. This section will describe four different types of data structures for storing these performance characteristics: the text-based input files, the initialisation data structures, the operational data structures and the binary storage data structures.

3.2.1 Input Files

The simulator requires that the BADA files be read in for each aircraft type used. In addition to the parametric parameters, the simulator also requires a flight plan and a set of initial conditions for each aircraft to be simulated. The format of these files can be seen in Appendix B. The data from both the BADA and the initial condition files are stored in a set of initial data structures which are described in Section 3.2.2.

3.2.2 Initialisation Data Structures

The initialisation data read from the text files is stored in five separate static structures for use throughout the simulation. Since the number of different types of aircraft and the different flight plans to be used in each simulation are known prior to simulation and will not change during simulation, it was not necessary to use a dynamic data structure to store the initialisation data. The first three structures are arrays corresponding to the three sets of BADA data, as illustrated in Figure 3. The length of the arrays corresponds to the number of different types of aircraft to be used in the simulation. In the case of Figure 3 there are five different aircraft types.

The other two structures are two-dimensional matrices corresponding to the set flight plan and initial condition data derived from the initial condition files, as illustrated in Figure 4. The matrices have dimensions of the number of different aircraft types to be used (five) by the number of different set flight plans to be used during the simulation (three). It is necessary to have flight plans tailored for each aircraft type because the current version of the simulator has not implemented logic

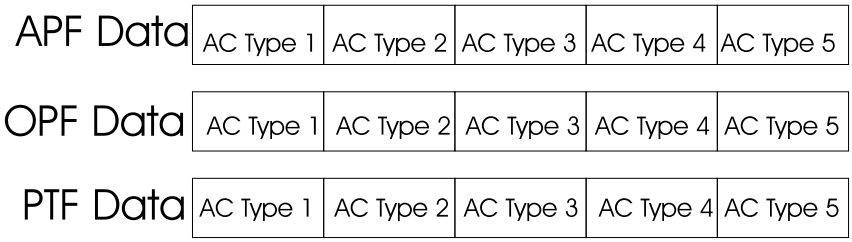


Figure 3: Initial Structure for BADA Data

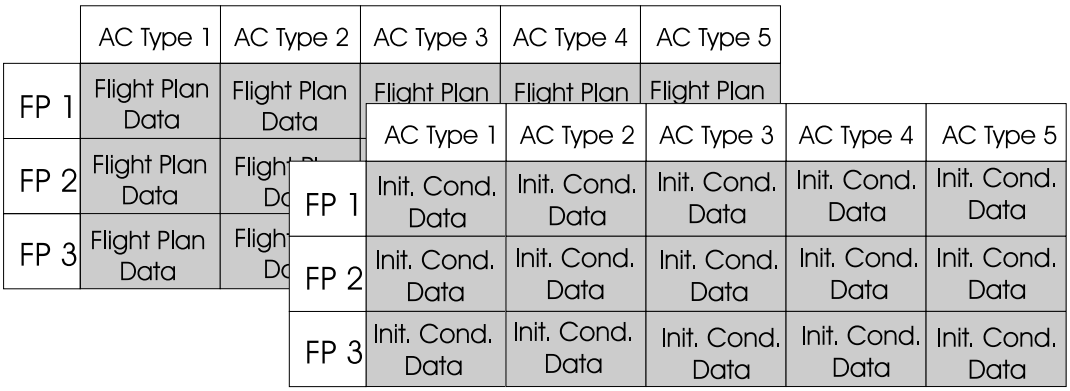


Figure 4: Initial Structure for Flight Plan and Initialisation Data

to automatically determine TOD (Top of Descent) or BOC (Bottom of Climb) points to meet ATC restrictions.

3.2.3 Operational Data Structures

Operational data structures include the structures necessary to store the data generated and used during each simulation time step. The operational data structures are created during the initialisation phase of the simulation and consist of seven data arrays. Table 3 lists the arrays which make up the operational data structures as well as their length and function. The length of the arrays corresponds to the number of concurrent aircraft allowable, as each array entry corresponds to the same simulated aircraft in each array. The exception to this rule is the Master Record array which keeps a log of all aircraft simulated over the course of the simulation, and thus has a length equal to the total number of aircraft simulated. This number

Table 3: Operational Data Structure Description

Array Name	Length	Function
Master Record	Total No. of Aircraft	Record array position and start/finish time for aircraft
Current Master Record	No. of Concurrent Aircraft	Array of active aircraft
Aircraft State	No. of Concurrent Aircraft	Record aircraft position, velocity and heading
Command State	No. of Concurrent Aircraft	Record aircraft guidance commands
Predicted Wind State	No. of Concurrent Aircraft	Record predicted wind determined by aircraft
Flight Plan List	No. of Concurrent Aircraft	Hold aircraft flight plan
Way Point List	No. of Concurrent Aircraft	Hold list of way points derived from Flight Plan List

may exceed the total number of concurrent aircraft allowable. As aircraft are removed from the simulation upon their arrival at their destination, the data at that index is deleted in all arrays except the Master Record array, and the space is then available to store data for a newly created aircraft.

With the exception of the Flight Plan List and Way Point List arrays, data in each array is arranged in a static data structure known in Modula-2 as a record. The Flight Plan List and Way Point List arrays, however are arrays of linked lists derived from the initial flight plan text file. It was felt that to keep the simulation as simple as possible, only the Flight Plan List and Way Point List arrays warranted a dynamic structure. It was not possible to use a dynamic structure for the Master Record array as the length of the Master Record array is required at initialisations to leave enough space in the permanent storage file.

Master Record & Current Record Array

Both the Master and Current Master Record arrays are arrays of the same data structure. This record holds the aircraft type, the flight plan, the start and finish time of simulation for the aircraft, the aircrafts's generation number, and the array index in which it is stored. The Master and Current Master Record arrays serve two distinct purposes. The Current Master Record array contains all of the aircraft actively being simulated, whereas the Master Record array contains all of the aircraft simulated up to that point in the simulation. When an aircraft is created it is added

to both the Master and Current Master Record arrays, however when an aircraft is removed, its entry is removed only from the Current Master Record array, and the Master Record array records the time of removal. At the culmination of a simulation the Master Record array is transferred to the permanent storage file.

Aircraft & Command State Arrays

Each entry in the Aircraft State array contains the set of data necessary to completely describe the position, heading, velocity and mass of a specific aircraft. This array is initialised during the initialisation phase of the simulation, and is updated at each time step, as described in Section 3.3.

Each entry in the Command State array contains the set of data necessary to guide an aircraft along its given trajectory. The Command State array serves as temporary storage for parameters which are derived in one section of the simulation, and will be used in a later section. The Command State array consists of both specific guidance commands, i.e. specifying a particular rate of climb, and a number of flags, which trigger commands such as to turn to new way point and to remove an aircraft. For further explanation of these arrays please refer to Section 3.3.

Flight Plan Record & Way Point List Arrays

The Flight Plan Record is read in as a single data structure, which contains phase of flight information, altitude commands, and way point information as a single flight plan. It is subsequently split into two separate linked-list structures to allow more flexibility in execution and alteration of the flight plan during simulation: one for the vertical and one for the lateral navigation components. The Flight Plan list is derived from the Flight Plan record and serves as the vertical navigation component. This component controls the phase of flight (climb, cruise, descent) and the altitude desired for each phase. The vertical navigation component is governed by a set of six switches (time, command, latitude, longitude, altitude, heading) which trigger the switch to the next vertical navigation segment. These switches allow the vertical profile of the aircraft to be tied to other aspects of the aircraft state such as lateral position. The Way Point list is derived from the Flight Plan list and serves as the lateral navigation component, providing a basis for trajectory calculation. Both the Flight Plan list and the Way Point list are organised into arrays whose entries correspond to the entries in the Current Master Record array.

3.2.4 Storage Data Structures

Storage data structures include the structures necessary to temporarily and to permanently store data derived from the simulation. After the initial time step, the temporary data storage is created then subsequently added to after each additional time step, until it is cleared and begun again. The permanent data storage structure consists of a single random access file, which is opened during initialisation, written to over the duration of the simulation and closed at termination.

The temporary data storage structure is a two dimensional matrix, the dimensions of which are the number of concurrent aircraft allowable by the number of simulation updates or time steps allowable before sending data to permanent data storage. Thus a simulation with 100 concurrently simulated aircraft running for one hour (3,600s) has dimensions of 100x3600. The matrix is built up of a number of arrays which record the aircraft information for each time step, and is held in local memory. The simulation is capable of handling a variety of resolutions, which effect the frequency that the data is placed into temporary storage. For example the same simulation with 100 concurrent aircraft running for one hour, might have a resolution of 5s, which would reduce the dimension of the permanent data storage to 100x750. As with the operational data structures, each entry in the array corresponds to an individual aircraft. The data contained in each entry is a amalgamation of data from the state operational arrays condensed into a data block and is discussed further in Section 3.3.2.

The permanent data storage structure is a random access file which holds three separate data structures. The first is an eight byte block of file data, which includes the number of total aircraft simulated, number of concurrent aircraft allowable, the resolution, the size the master record and the size of the data block. The second is the master record array, with a length of the total number of aircraft processed in the simulation. The second is a two dimensional matrix with dimensions of the number of concurrent aircraft allowable by the total number of simulation updates run during the entire simulation. A graphical illustration of the permanent data storage structure can be seen in Figure 5, where a total of n aircraft were simulated with m aircraft simulated simultaneously. The data held in the file is in binary format to minimise the amount of space required for storage.

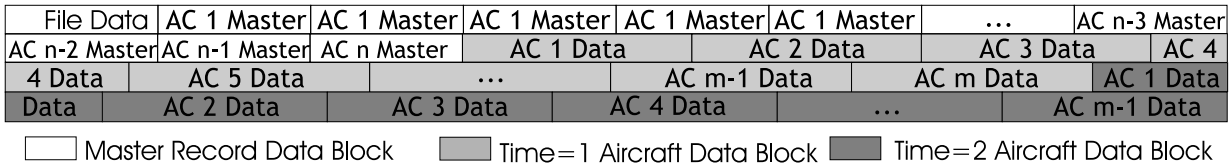


Figure 5: Permanent Storage Structure

3.2.5 Storage Size

The total number of aircraft the simulator can handle is a direct consequence of the size of the permanently stored data blocks, the frequency with which the data is recorded (time step size), and the resolution of the output data required, e.g. every 5 seconds. The maximum number of concurrent aircraft the simulator can handle is also a function of data block size and time step, as well as a function of the frequency with which the data blocks are transferred to permanent storage. As these parameters are arbitrary, to determine these simulation parameters it was necessary to consider any overriding limits on storage size. For permanent data storage, it was decided that each simulation run should be capable of being stored on a single CD, thus setting a maximum storage size of 700 MB (Megabyte). In addition it was decided that the maximum amount of storage used by the temporary data structure should remain under 10 MB for portability to other machines. It is important to note that the limits on data storage are parameters in the simulation and can be changed. However for the majority of cases it is believed that these limits are appropriate.

With these two limits set, the number of total aircraft allowable for a given resolution and simulation period were calculated and are presented in Table 4. From these calculations it is obvious that in order to simulate the large numbers of aircraft desired, the data blocks would need to be condensed under 76 B (Byte). However it is important to note that Table 4 assumes that a single aircraft will be flying for the duration of the simulation period to illustrate the amount of memory such long flights require. It is possible to simulate 4264 aircraft in a 24 hour simulation if the aircraft fly routes which average three hours in length and the resolution is set to five seconds.

To determine the frequency with which the temporary data structure needed to be transferred to the permanent storage structure file, an investigation was carried out to determine the time impact of transfer frequency. In other words, to determine if a time penalty would be incurred by transferring less data at a higher frequency,

Table 4: Total Number of Aircraft Allowable Assuming Data Block Size and Continuous Flight

Resolution	DB Size	Number of Hours Simulated 24	20	12	6	3
Once a Second	128B	63	76	127	253	506
	76B	107	128	213	426	853
Every Two Seconds	128B	12	150	252	500	1000
	76B	127	152	253	506	1013
Every Five Seconds	128B	375	375	630	1250	2500
	76B	533	640	1066	2132	4264

than transferring more data at a lower frequency. The investigation timed the transfer of data blocks of either 265, 384 or 512 bytes for 100, 200, 300, 400 and 500 aircraft at a time with update rates of 1, 10, 30, 60, 300, 1,800, 3,600s. For comparison, the times were then normalised to the time to transfer 3,600 data blocks per 100 aircraft. The resulting averages and standard deviations can be seen in Figure 6.

As expected, the larger the data block size, the longer the transfer time is. However instead of increasing transfer time, increasing the frequency of transfers actually decreases it slightly. Therefore it was decided that to provide as much flexibility as possible, to allow any combination of total number of aircraft, maximum

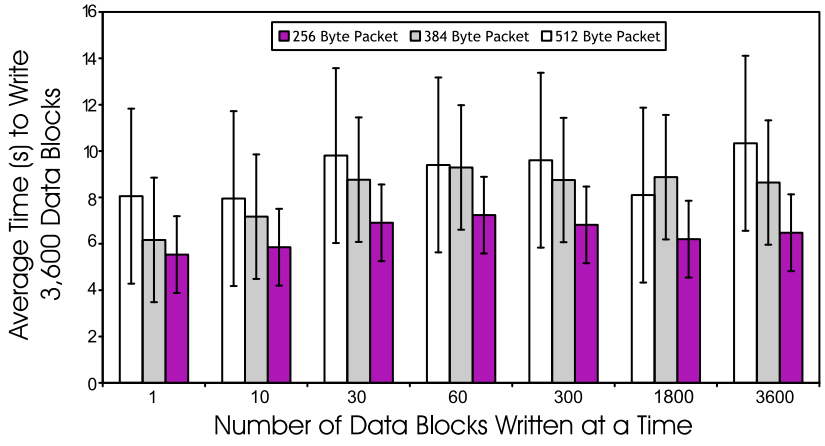


Figure 6: Data Transfer Time Comparison

number of concurrent aircraft and data block size which meet both the temporary (10 MB) and permanent (700 MB) size restrictions. Information regarding total aircraft simulated, concurrent aircraft simulated, data block size and resolution is thus stored at the beginning of each permanent storage file so that it can be read at a later date.

3.3 Logical Design

The logical design for the Multiple Aircraft Performance Module has three distinct phases: Initialisation, Simulation Loops, and Shutdown. The initialisation phase includes reading in required initialisation data, placing it into the appropriate initialisation data structures, generating the aircraft, and linking them with the appropriate operational data structure entries. The simulation loop phase includes all of the navigation and guidance calculations, condensing of data, recording data to both temporary and permanent data storage structures, the dynamic creation and removal of aircraft from active simulation, and hooks for broadcasting and downloading data. The shutdown phase records all remaining data in temporary data storage to permanent storage, deallocates all dynamically allocated data and terminates the program. Figure 7 illustrates the control flow diagram for the Multiple Aircraft Performance Module.

3.3.1 Initialisation Phase

The initialisation phase consists of a linear flow of data from text files to the initialisation data structures. Once the data has been successfully imported, then the initial aircraft are pseudo-randomly generated from a specified distribution of aircraft type and assigned a specific flight plan from a specified distribution of flight plans and placed into the Current Master Record array. If there are fewer aircraft initially generated than the maximum number of concurrent aircraft allowable, then the remaining current master record entries will be given an aircraft type of EMPTY. The simulation loop phase will only take into account aircraft with an aircraft type other than EMPTY.

Consequently, each aircraft generated will have an assigned aircraft type and an assigned flight plan. These two assignments link the aircraft described in a particular Current Master array entry to the appropriate entries in the initialisation

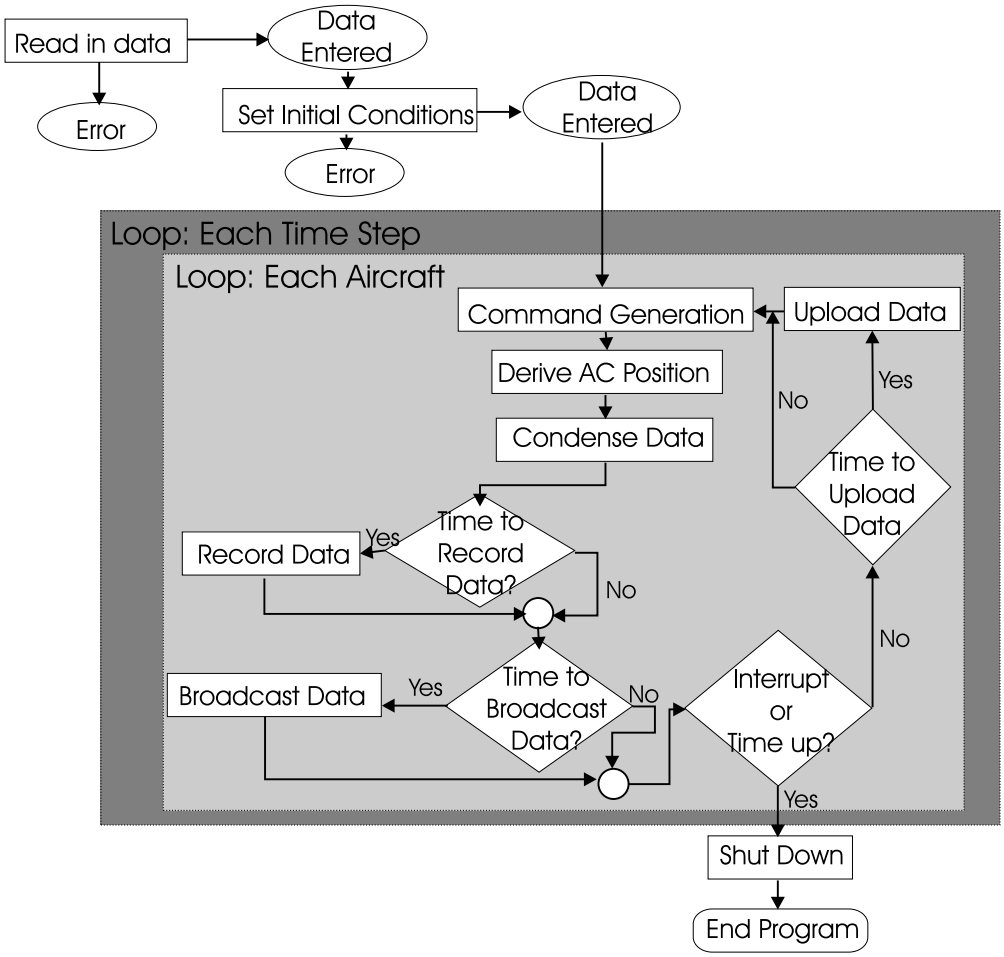


Figure 7: Multiple Aircraft Performance Module Control Flow

data structures. An illustration of this linkage can be seen in Figure 8. With the Current Master array linked correctly, the Aircraft State, Command State, Flight Plan list, and Way Point list operational arrays can be initialised. The final step in the initialisation phase is to open the random access file for permanent data storage.

3.3.2 Simulation Loop Phase

The simulation loop phase consists of two nested loops. The inner loop cycles through each aircraft in the Current Master array with an aircraft type other than EMPTY. The outer loop cycles through time and is incremented by a preset time

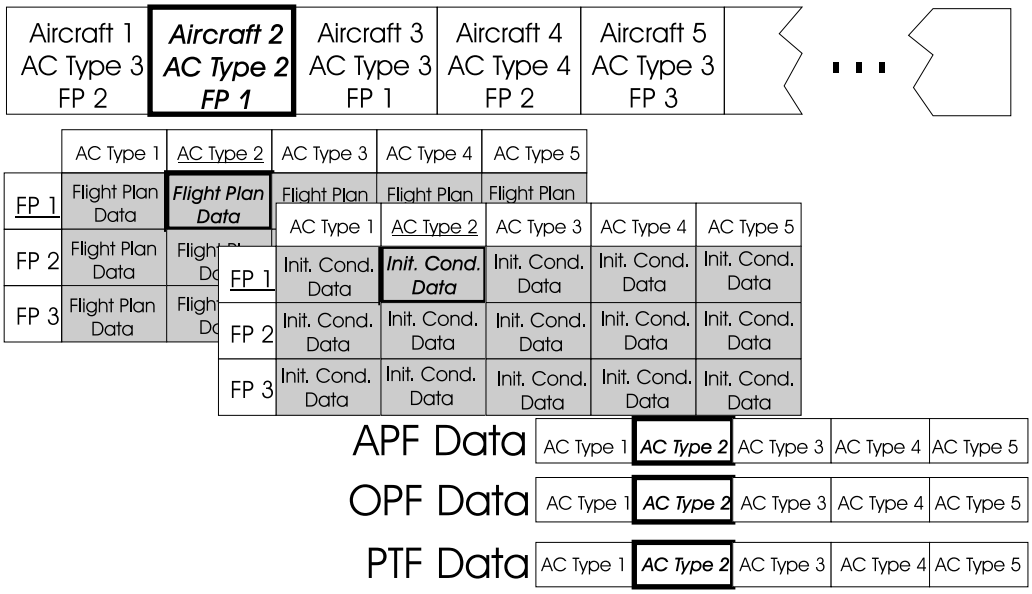


Figure 8: Aircraft Generation Linkages

step. Inside the loops a set of commands are generated for each aircraft, the aircraft's position is updated and the data generated is condensed and temporarily or permanently stored.

Random Aircraft Generation

There are two primary ways to generate aircraft in simulation. The first is to generate them randomly according to some preset distribution. The second is to generate them according to some input file or pre-established order. In order to reduce the number of input files required by this simulation, it was decided to implement a pseudo random generation algorithm. The random aircraft generation occurs between the outer time loop and the inner command generation loop. At pseudo-random time intervals an aircraft is generated. The time intervals have a Gaussian distribution specified by an average generation frequency and deviation from that frequency. The aircraft type and flight plan are pseudo-randomly chosen according to a distribution provided to the algorithm from a specified array of aircraft types and flight plans. During generation the aircraft is added to the Current Master and Master arrays, and both the Flight Plan and Way Point lists are created.

Navigation & Command Generation

The command generation phase determines the guidance commands to follow a great circle route from one way point to another. The aircraft will climb, cruise, or descend according to the prescribed flight plan. Additionally the command generation phase also determines if an aircraft has reached its destination and if so, removes the aircraft from active simulation. The overall control flow diagram for the command generation phase can be seen in Figure 9.

The first step in the command generation phase is to independently update the current flight plan segment and current way point if necessary. During this procedure two separate flags are set. The `NewWP` flag is set if the current way point has been updated, and the `RemoveAC` flag is set if the aircraft has reached its destination. The `NewWP` flag is part of the lateral guidance logic and triggers the calculation of the distance out from the current way point to begin the turn to the next way point. The `RemoveAC` flag is part of the dynamic aircraft creation and removal logic and triggers the removal of the aircraft from the active simulation. During the simulation aircraft can only be removed by reaching their destination.

The next three procedures make up the vertical guidance component of the

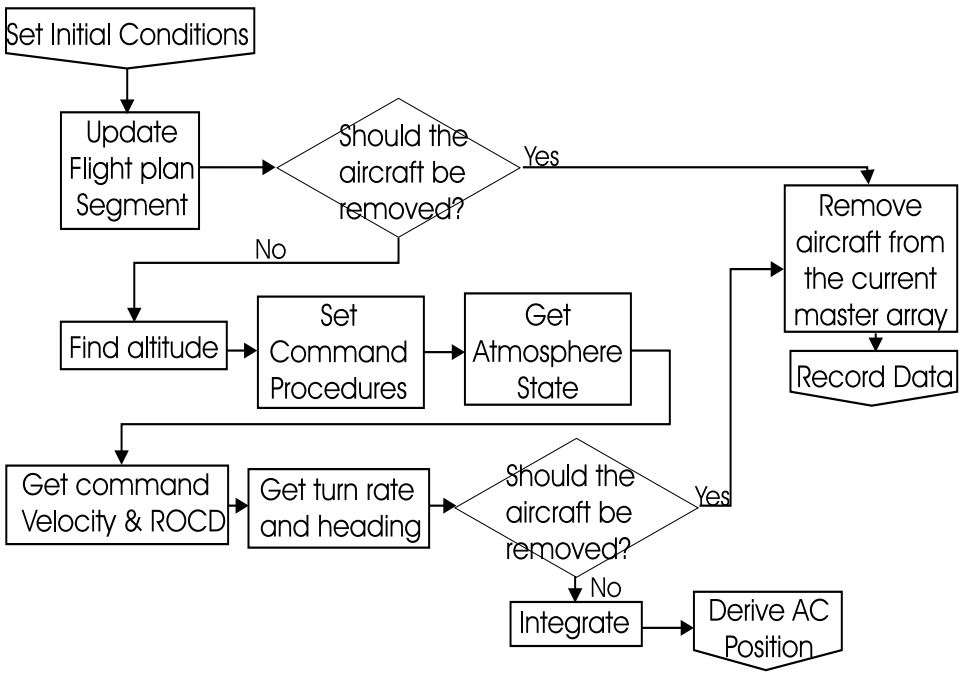


Figure 9: Command Generation Control Flow Diagram

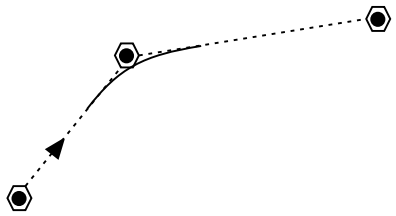


Figure 10: Inside Turn

command generation phase. They set the altitude command, the procedure mode (climb, cruise, descent), the velocity and rate of climb/descent from the current flight plan, aircraft state, and BADA PTF. The present simulator assumes perfect accuracy in the aircraft's knowledge of its altitude. Later versions may benefit from the addition of an altitude uncertainty model. Currently the data is linearly interpolated from the BADA PTF. Changes in rate of climb and descent are assumed instantaneous so no logic is provided to increase or decrease the aircraft's rate of climb/descent before a vertical maneuver is executed.

The following two procedures (shown as one procedure in Figure 9) make up the lateral guidance component of the command generation phase. They set the turn rate and heading from the way point list and the aircraft state. The lateral guidance assumes a spherical earth and great circle routing between way points. The aircraft follow the great circle heading to parameterised accuracy, allowing in effect FMS accuracy to be modelled. Additionally in order to better mimic the flight path taken by the aircraft under FMS guidance, all turns are assumed to be rate one, inside turns, see Figure 10, executed at a turn rate of 3^o a second. The turn rate is modelled as linearly increasing turn rate until a rate of 3^o a second is achieved, and then a linear decrease in turn rate until a rate of 0^o a second is achieved.

Position Update

The position update phase uses the command data generated previously and increments the aircraft by one time step using numerical integration. The control flow diagram for the position update phase is illustrated in Figure 11. A time step of one second is used by the simulator, thus allowing the simulator to run in fast time while maintaining a high level of accuracy. Due to the relatively small time steps involved a simple Euler numerical integration scheme shown in Equation 2 is used

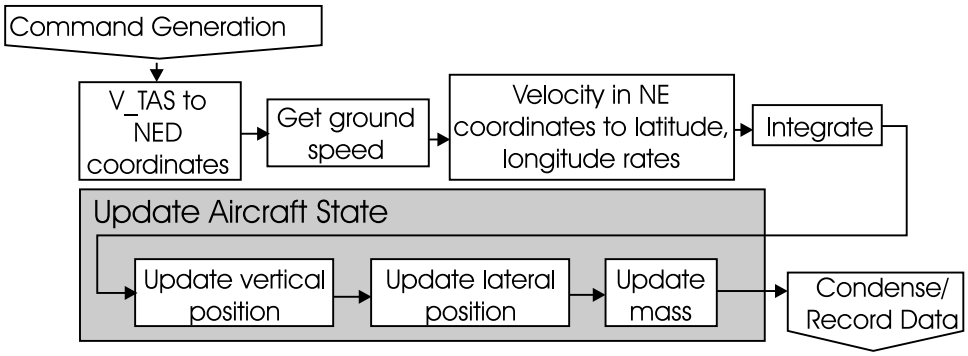


Figure 11: Position Update Control Flow Diagram

throughout the position update phase.

$$x_{t+1} = (\dot{x}_t \times \Delta t) + x_t \tag{2}$$

The first step in this phase is to break the aircraft's TAS (True Air Speed) into its components of North, East, Down (NED) with respect to the air, as shown in Equations 3 - 5 where θ is the flight path angle, and ψ is the heading from true north.

$$V_D = V_{TAS} \sin\theta \tag{3}$$
$$V_N = V_{TAS} \cos\theta \sin\psi \tag{4}$$
$$V_E = V_{TAS} \cos\theta \cos\psi \tag{5}$$

From these components the ground speed can then be calculated by transforming the from the air frame to the Earth-centred, Earth-fixed frame (ECEF), This transformation is accomplished by adding in the wind vector, as shown in Equations 6 - 8.

$$V_D^i = -V_D + V_{W_D} \tag{6}$$
$$V_N^i = V_N + V_{W_N} \tag{7}$$
$$V_E^i = V_E + V_{W_E} \tag{8}$$

Then the ground speed components are transferred into latitude and longitude rates

using Equations 9 and 10 [7].

$$\dot{\phi} = \frac{V_N^i}{R_o + h} \tag{9}$$

$$\dot{\lambda} = \frac{V_E^i \sec L}{R_o + h} \times \cos L \tag{10}$$

The final step in the update position phase is to integrate the latitude and longitude rates along with the rate of climb/descent and the aircraft mass using Equation 2. The new aircraft position and mass transferred to the aircraft state record. Additionally commanded velocities, rates of climb/descent, turn rates and heading are also transferred to the aircraft state record.

Record Data

The Record Data phase of the Simulation Loop transfers data from the operational arrays such as the aircraft state array, first to temporary data storage structure and finally to a permanent data storage structure. The first step in the data recording process is to condense the data from the three operational array records into a single record structure, called a data block, while minimising the record's memory usage. Table 5 illustrates a sample data block structure and illustrates how transformations prior to data storage are used to reduce storage size. For instance both latitude and longitude are stored as `integers`, but are in reality `reals` with a precision of 6 decimal places. A `real` requires 8 bytes or 64 bits of storage space, whereas a `integer` only requires 4 bytes or 32 bits of storage space.

Once the data has been condensed into a data block, the block is placed into the temporary storage array. This process is repeated throughout the aircraft loop and then over a specified number of time steps. At each time step an array is added to the temporary storage matrix. The number of time steps in the temporary storage matrix depends on the size of the data block as it must remain small enough to ensure the temporary storage matrix stays less than 10MB. The size of the storage blocks is left as a variable parameter in the simulator.

Once the maximum number of time steps allowable has been reached the temporary storage matrix is transferred to the permanent storage file. The data is transferred as raw data from the temporary storage matrix to the random access file one block at a time. The starting position in the random access file is positioned to leave space for the eight bytes of file data and for the Master Record array to be

Table 5: Data Block Structure

Variable	Modula-2 Data Type	Range Necessary	Multiplier	Bytes Used
		Master Information		
ac_{type}	AC-TYPE	type of aircraft		4
$time$	CARDINAL	current time (s) from sim beginning		4
		AC State		
lat	INTEGER	[-1.5708 : 0.000005 : 1.5708] rad	*1,000,000	4
$long$	INTEGER	[-3.1416 : 0.000005 : 3.1416] rad	*1,000,000	4
alt	SHORTCARD	[0 : 1 : 14000] m		2
$mass$	CARDINAL	[0 : 1 : 300000] kg		4
V_{TAS}	SHORTCARD	[0 : 0.1 : 400] $\frac{m}{s}$	*10	2
V_{G_N}	SHORTCARD	[0 : 0.1 : 400] $\frac{m}{s}$	*10	2
V_{G_E}	SHORTCARD	[0 : 0.1 : 400] $\frac{m}{s}$	*10	2
$heading$	SHORTREAL	[-0.0873 : 0.0018 : 0.0873] rad	*1,000	4
$ROCD$	SHORTREAL	[-50 : 0.1 : 50] $\frac{m}{s}$		4
		Vertical Guidance		
alt_{cmd}	SHORTCARD	[0 : 1 : 14000] m		2
$proc_{cmd}$	C-PROCEDURE	commanded procedure mode		4
$ROCD_{cmd}$	SHORTREAL	[-50 : 0.1 : 50] $\frac{m}{s}$		4
$V_{TAS_{cmd}}$	SHORTCARD	[0 : 0.1 : 400] $\frac{m}{s}$	*10	2
		Lateral Guidance		
$turn_{rate_{cmd}}$	INTEGER	[-0.0873 : 0.001745 : 0.0873] $\frac{rad}{s}$	*1,000,000	4
$heading_{cmd}$	SHORTREAL	[-0.0873 : 0.0018 : 0.0873] rad		4
		Predicted Wind		
$time_ahead$	CARDINAL	[0:14400] s		4
$wind_{North}$	INTEGER	[-400 : 400] $\frac{m}{s}$		4
$wind_{East}$	INTEGER	[-400 : 400] $\frac{m}{s}$		4
$wind_{Down}$	INTEGER	[-400 : 400] $\frac{m}{s}$		4
			Total	72

written at the beginning of the file after the simulation has been concluded.

3.3.3 Shutdown Simulation

The final phase of the simulation is the Shutdown phase. Figure 12 illustrates the control flow diagram for the Shutdown phase. The first step in this final phase is to transfer the remaining data in temporary storage structures over to the random access file for permanent storage. Next the eight bytes of file data and the Master Record are written to the file and the random access file is closed. Finally all of the dynamically allocated memory for the Way Point list and Flight Plan list arrays are

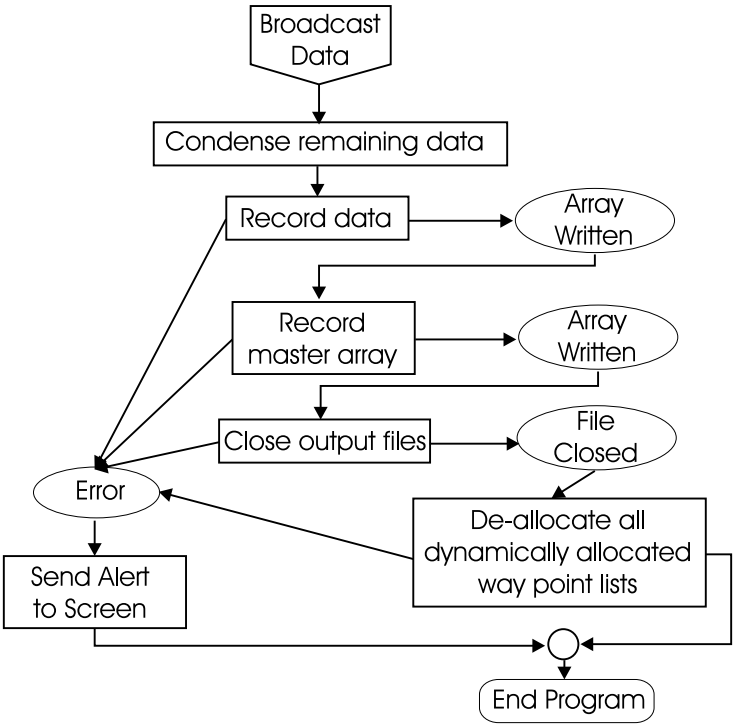


Figure 12: Shutdown Control Flow Diagram

de-allocated and the simulation program is terminated.

3.4 Validation

Before employing the aircraft performance model, it was necessary to validate it. The validation conducted consisted of three separate evaluations. The first sought to verify that the simulator correctly implemented version 3.3 of BADA. The second sought to verify that the simulator could correctly guide the aircraft along a great circle route, and the third sought to verify that the simulator was capable of simulating the required number of aircraft concurrently and in total.

3.4.1 Correct Implementation of BADA v3.3 Data

This evaluation is aimed at eliminating any trivial problems arising from an incorrect implementation of the BADA data. The investigation consisted of running five separate aircraft types through three separate maneuvers at or between specified flight levels to allow a comparison to be made to the data given in the BADA PTF. The comparison was made by determining the average velocity of the aircraft from

Table 6: Implementation of BADA v3.3 Data Evaluation Results

Aircraft Type	Time For Maneuver		%
	Simulator	BADA	Error
Cruise for 2000km			
A320	8675	8643	0.37%
B737-300	9143	9108	0.38%
B737-800	8580	8548	0.37%
B747-200	8307	8275	0.39%
MD80	8913	8879	0.38%
Climb From FL290 To FL310			
A320	215	215	0%*
B737-300	192	192	0%*
B737-800	148	148	0%*
B747-200	177	177	0%*
MD80	203	203	0%*
Descent From FL330 To FL290			
A320	112	112	0%*
B737-300	123	123	0%*
B737-800	137	137	0%*
B747-200	145	145	0%*
MD80	233	233	0%*

*To 1s accuracy

the flight time produced by the simulation using Equation 11.

$$\overline{V_{TAS}} = \frac{Distance}{FlightTime} \tag{11}$$

The results are given in Table 6. It can be seen that the aircraft climbs and descends exactly as the BADA directs. The minor error seen in the Cruise segment can be attributed to the navigation error and numerical error and is well within current aircraft navigational tolerances [7].

3.4.2 Correct Implementation of Great Circle Navigation

This evaluation is aimed at verifying that the navigation and guidance logic used by the simulator is capable of controlling aircraft along great circle paths. The investigation consisted of simulating five different aircraft types navigating between two way points at a constant altitude for approximately 1080Nmi (2,000km), over

Table 7: Great Circle Navigation Evaluation Results

Aircraft Type	Comparison Point Average Percent Error					
	East	West	North	South	South-West	North-East
A320	5.79×10^{-8}	5.79×10^{-8}	7.39×10^{-7}	9.66×10^{-7}	2.15×10^{-5}	1.77×10^{-4}
B737-300	5.79×10^{-8}	5.79×10^{-8}	1.90×10^{-6}	1.79×10^{-6}	2.08×10^{-5}	1.69×10^{-4}
B737-800	5.79×10^{-8}	5.79×10^{-8}	1.82×10^{-6}	1.90×10^{-6}	2.12×10^{-5}	1.80×10^{-4}
B747-200	5.79×10^{-8}	5.79×10^{-8}	1.98×10^{-6}	1.88×10^{-6}	2.21×10^{-5}	1.86×10^{-4}
MD80	5.79×10^{-8}	5.79×10^{-8}	1.72×10^{-6}	1.79×10^{-6}	2.12×10^{-5}	1.72×10^{-4}

six different directions. The trajectories were compared with the great circle joining the two way points every 108Nmi (200km) for a total of 10 comparison points. The comparison was carried out using MATLAB^{TM}to first calculate the great circle route and then to compare the trajectories of the aircraft as recorded by the simulator. The results are listed in Table 7. The table indicates that the guidance imbedded in the simulator provides an accuracy of approximately half a kilometre over a 2000km flight, which is within specifications for RNAV-1 [7, 77].

3.4.3 Verification of Capacity

In addition to the implementation and navigation verification, a capacity verification was also conducted. Aircraft were flown on two distinct flight paths with an approximate length of 600Nmi (1,117km) for a simulated period of 18 hours. New aircraft were created every of 15s which resulted in an average of 321 aircraft being concurrently simulated and 4,000 aircraft being simulated in total, thus verifying that the simulator is capable of meeting the capacity requirements established in Chapter 2.

3.5 Summary

The requirements and capabilities of a set of simulation tools necessary to complete the research proposed have been discussed. The simulator's architecture was discussed in detail and both the data structure and the control flow architectural aspects were covered. The adherence of the simulator to the BADA database has been verified, and the simulator has also been shown to have the required capacity of 300

concurrent and 4,000 total aircraft simulated. Additionally the simulator's navigation algorithm has been verified, and its accuracy falls within the RNP requirements for enroute navigation.

CHAPTER 4

Atmospheric Environment

4.1 Model Motivation

In order to accurately simulate aircraft flights, it is necessary to provide an accurate wind field model as the variability in winds aloft is a primary contribution to trajectory uncertainty [9, 30]. Existing simulations such as RAMS, TAAM, and NARSIM all allow some form of wind simulation [56, 63, 78]. NARSIM and TAAM use METAR and METAF (Terminal Aerodrome Forecasts)s to generate wind. RAMS uses an add-on model called ATMOS (Atmospheric Weather Model) to model winds aloft using a set of data provided by the FAA for the continental United States at a single point in time and with a resolution of 1^0 of latitude and longitude and 1,000ft of altitude. The purpose of the wind field model developed for this simulator is to supply wind data to the multiple aircraft performance model as shown in Figure 13.

4.2 Model Requirements

The wind field model is required to supply the wind vector necessary for each aircraft to compute its ground speed. The form of the wind model includes a series of uniform

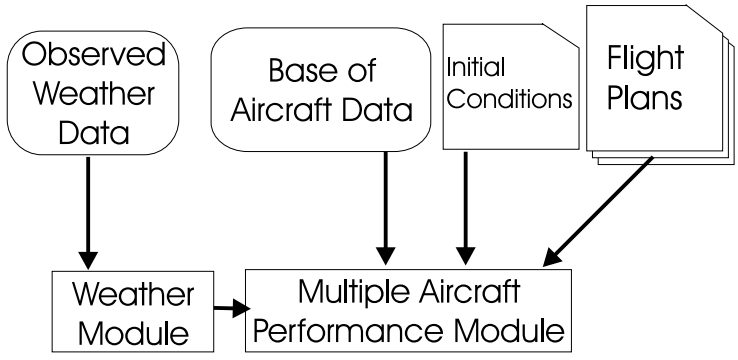


Figure 13: Simulator Architecture with Wind Field Model

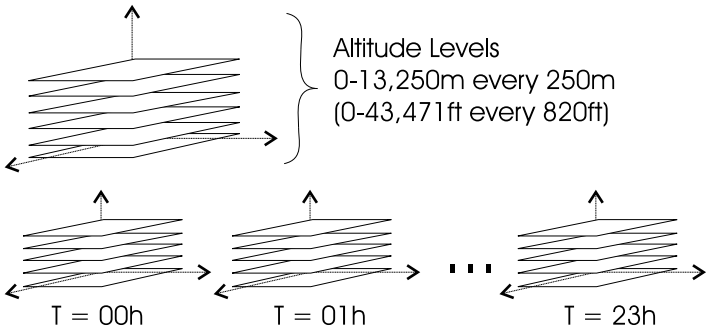


Figure 14: Grid Data Visualisation

grids over a series of time intervals and altitude levels as shown in Figure 14. By using a set of uniform grids, the computation required during the simulation run can be minimised by using a simple linear interpolation algorithm. The wind field needs to cover a large geographic area and a large time span so that multiple aircraft can be subjected to the same wind field. As there is no data source capable of generating a wind model over the entire globe, a more restricted wind model was used. An average flight in both Europe and the US is 470Nmi (∼870 km) or 80 minutes in duration and traverses 5 or 3 sectors respectively [6]. Thus the minimum size wind field required should cover the equivalent area of 3 U.S. sectors, 312,351Nmi^2 (1,071,334km^2).

Additionally, the range and scale of altitudes covered needs to be sufficient to capture the variability in wind magnitude and direction experienced both on climb out and on approach. The primary cruising altitudes for commercial transport aircraft are located between 30,000ft to 43,000ft, as high-bypass turbojet engines are most efficient at these pressure altitudes. Consequently, the wind field required needs to extend from ground level to 43,000ft (13,106m) to capture the majority of flight levels used by commercial transport aircraft.

4.3 Data Source

In order to construct a wind model to calculate the wind vectors at aircraft locations, a suitable source of wind data was required, which met the requirements set out in Section 4.2. The purpose of the data source is to construct the uniform grids which make up the model. The fidelity of the resultant model depends on the resolution of the grid, i.e. the greater the number of data points the greater the resolution, and the frequency with which data is available at those sites. Thus a data source

with 15 data collection sites which records data every hour will result in a model of higher fidelity than a data source with 5 data collection sites which records data every 6 hours.

A number of different sources of data for the wind field model are available, several of which are compared in Table 8. The majority of wind data available to aircrews consists of surface observations and is distributed in the form of METARs. METAFs and forecasted winds-aloft are based on sophisticated weather models based on observed winds-aloft data from discrete locations around the globe are also available.

METARs, however, do not provide the range of altitude for wind data necessary for the required wind field model. Additionally, as has been shown in the literature, the predicted winds aloft used to generate METAFs and forecasted winds-aloft are often inaccurate [67]; it was felt that predicted winds-aloft data is not a suitable source on which to base the model. Consequently it was decided that the model should be based on actual observed winds aloft instead of forecasted data, if possible.

	Area Covered	Area Scale	Altitude Range	Altitude Scale	Duration	Duration Scale	Observed/ Forecasted	Collection Method
METAR	World Wide	Airfield	Surface	n/a	Constant	Hourly	Observed	Ground Observation
METAF	World Wide	Airfield	Surface	n/a	Constant	Every 6 Hours*	Forecasted	Ground Observation?
Forecasted Winds Aloft	World Wide	Major Airfields	$1e10^3$m – $1.3e10^4$m (MSL)	1,000m	Constant	Every 8 Hours*	Forecasted	Computer Weather Model
Statistical Recreation	World Wide	Strategic Locations	$1e10^3$m – $2.7e10^4$m (MSL)	1,000m	Several Years$^+$	Every 12 hours	Observed	Weather balloon
NPN	$2e10^6$ km^2	$7e10^4$ km^2	2000m – $1.6e10^4$m (AGL)	250m	Constant	Hourly	Observed	Clear Air Radar

*for the U.S.A. $^+$[79]

Table 8: Sources of Wind Data

Unfortunately observed winds-aloft data is primarily gathered by weather balloons, which are not launched in close proximity. Extensive statistical wind velocity and direction data was collected by NASA in the 1960s and was subsequently compiled and published [79, 80]. Due to the low number of observation locations available through weather balloon launches, this type of data was considered too crude to construct a wind field model.

Alternatively, the NOAA (National Atmospheric and Oceanographic Agency (The United States)) NPN (NOAA Profiler Network) collects observed winds aloft data every six minutes and reports it hourly from a series of radar stations located over a large portion of the central U.S. The NPN was the only set of observed winds aloft which satisfied the scale and range criteria for geographic area, altitude and time, and was consequently chosen as the basis for the wind field model.

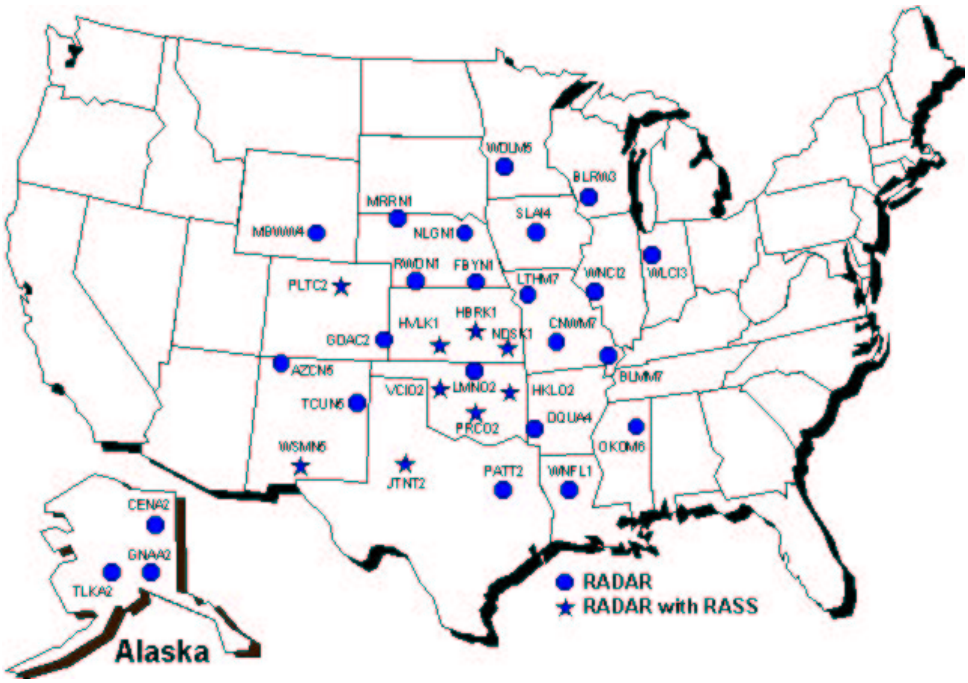


Figure 15: NPN Geographic Site Location

The NPN covers 33 collection sites, with 30 sites located in the region from the Mississippi river valley west to the eastern edge of the Rocky Mountains in the continental United States (CONUS) and three are located in Alaska, as shown in Figure 15. From these sites, data is available for an area from the 33rd to the 44th parallel and from the 87^o to 107^o longitude, covering an area of over 2 million square kilometres.

Each station records the wind velocity and direction at every 250m AGL (Above Ground Level) to around 16,000m and reports the data hourly. The wind data is collected by using 404 MHz clear air radar with a wave length of 74cm, capable of detecting, "fluctuations in the atmospheric density, caused by turbulent mixing of volumes of air with slightly different temperature and moisture content."[81] The mean wind is calculated from the fluctuations in the index of refraction. The NPN radar is capable of operating in the presence of clouds and moderate precipitation [81] and has an accuracy of 1m/s wind velocity.[81]

The NPN data had the additional benefit of being available to the public through an interactive website [82]. This website provides the data collected by each station in a variety of different text formats. The availability and text formatting of the NPN data allows researchers to create as many wind field models (under different observed weather conditions) for as long a duration as desired, by downloading different data sets. The public availability of weather data further adds to the accessability of this simulation to researchers.

4.4 Data Preparation

In order to create the wind field model, the data collected from each station over a period of 24 hours needs to be mapped onto a uniform grid that the simulated aircraft could then interpolate between grid points to calculate the wind at their location. Additionally, as the grid does not cover the entire globe, it was necessary to transpose and reflect the data through vertical and horizontal planes in a patch work fashion to simulate global coverage. To demonstrate how the data from the NPN networks is prepared for use by the simulator, data collected on May 8^{th} 2002 will be used as an example, and the following section will outline its preparation.

Of the 30 stations located in CONUS, four were not functioning on May 8^{th} 2002 when the data sample was down-loaded. These non-functioning stations can be seen in Figure 16 as the crossed out locations. Twenty-four hours of data was downloaded in units of KNOTS (nautical miles per hour) for wind speed and

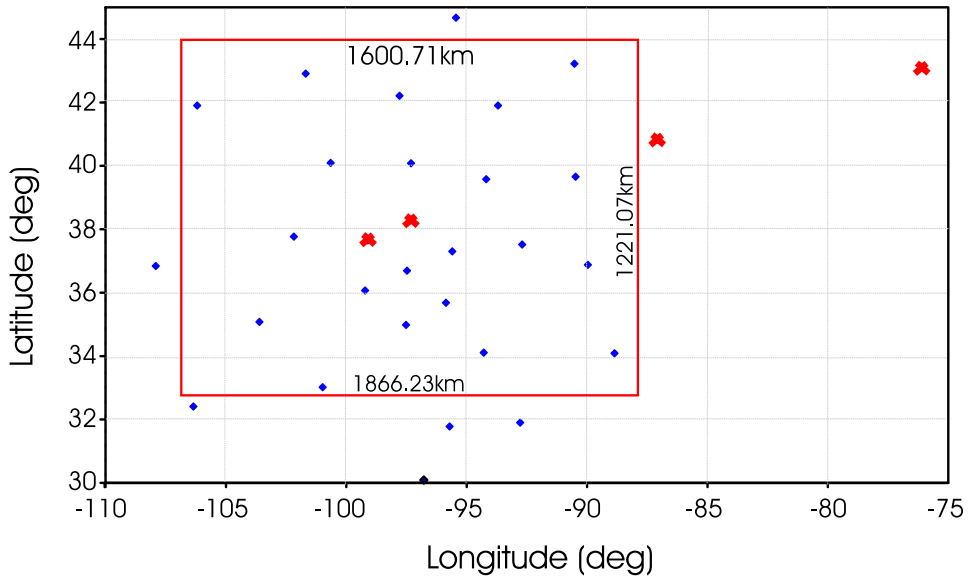


Figure 16: Operational Stations

degrees relative to true North for wind direction [83] from the NPN website [82] in 26 separate formatted text files (one per station). To configure the data into a useful data structure it was necessary to first interpolate the data from each hour from the unique altitude levels of each station location onto a uniform set of altitude levels. For the purposes of this research the data is only needed from ground level to 43,000ft (13,250m). Unfortunately, each station has a unique altitude and consequently records data at slightly different altitudes because the data readings that are taken every 250m AGL. The elevation of the highest recording station was 1,900m and accordingly, the uniform altitude levels begin at 2,000m and increase by 250m until 13,250m. All altitude levels below the 2,000m are duplicates of the 2,000m level. This interpolation was completed with MATLAB^{TM}using a cubic-spline interpolation.

Next, the uniform data levels are transferred from the discrete geographic data-collection-locations onto the uniform grid mentioned above and shown in Figure 14. This transformation is also performed in MATLAB^{TM}, using a method that is

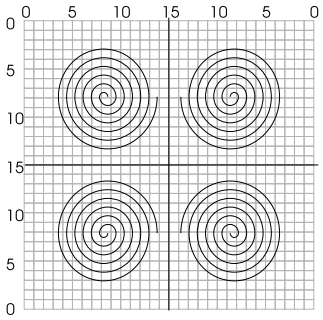


Figure 17: Wind Filed Mirroring Across Lines of Latitude and Longitude

based on biharmonic spline interpolation [84]. This method was chosen because it handled the non-smooth nature of the wind data better than other interpolation options such as nearest neighbour, triangle-based linear, or triangle-based cubic which were all based on a Delaunay triangulation of the data. The resulting data is formatted as a set of grids for each altitude level and each hour as shown in Figure 14 on page 48. The grid size chosen was 10^0 of latitude by 15^0 of longitude, which is the largest grid that would provide complete coverage of the globe. The data is stored in a large matrix comprising 46 by 24 data grids.

To cover the entire globe, the wind data is mirrored across lines of latitude and longitude by first incrementing then decrementing the lateral portion of the grid, such that the wind field on either side of the edge form a mirror image, as shown in Figure 17. This enables one patch of wind to provide a uniform and smooth wind field over the entire earth. This tiling also introduces a source of error for aircraft completing journeys that cover more than a single wind field patch. The extent of the error introduced by tiling the wind field is discussed in Section 4.6. As the NOAA Profiler Network expands, however, the individual grids will can be enlarged and this error will be reduced.

4.5 Interpolation Scheme

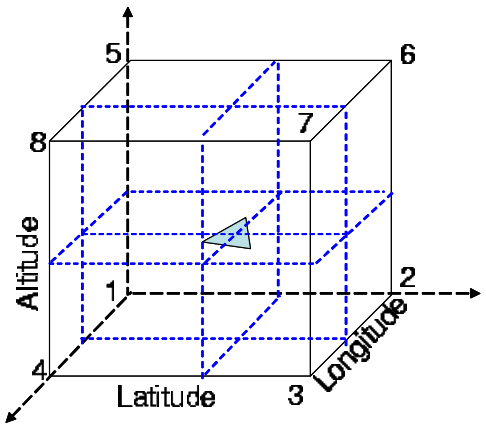


Figure 18: Interpolation Cube

The wind model is implemented by interpolating the wind direction and magnitude data stored in the data grids. The interpolation scheme used is a tri-linear interpolation, whereby each dimension is linearly interpolated independently of the other dimensions. A linear interpolation was chosen to minimize the computational load induced by the wind field model. The first step in the tri-linear interpolation scheme is to define the interpolation cube around the aircraft's current position. The interpolation cube is illustrated in Figure 18 based on the latitude, longitude, altitude coordinate set. The eight vertices are numbered starting from the back lower left-hand corner and proceeding clockwise around the lower face continuing to the back upper left-hand corner and proceeding clockwise around the upper face. The percentage of linear distance between each set of vertices is calculated for each dimension. Then the values of wind magnitude and direction are combined using the linear weighting to produce a fully interpolated value. This value is used by the simulation to determine the actual wind encountered by the aircraft. It is calculated at each time step and incorporated into the position calculation as described in Equations 6 - 8 in Chapter 3.

4.6 Wind Field Accuracy

The accuracy of the new interpolated set of grids for the May 8^{th} sample data has been checked by interpolating back to the original set of points using a tri-linear interpolation method. The average error and standard deviation over 24 hours and each flight level for the points reproduced from the grids are shown in Figure 19. If the data collection error and the interpolation error are assumed to be independent, then the average overall accuracy of the wind field model is approximately 2.5m/s.

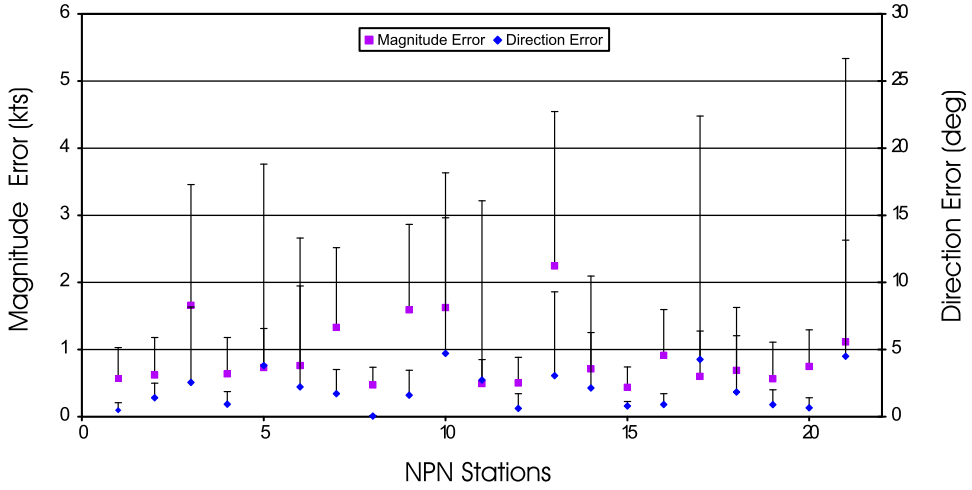


Figure 19: Interpolation Error by Magnitude and Direction

In order to determine the accuracy of the weather model over a region larger than the original wind field data, a comparison of the wind variability given by the model should be made with the variability of the actual wind field. The day-to-day variability (variance) of longitudinally averaged winds obtained by tiling the NPN wind field longitudinally is likely to be significantly greater than the corresponding variability for actual longitudinally averaged winds, so it is desirable to quantify the relationship between the variances derived from the two approaches. A direct comparison to the real wind field is not possible however, due to the lack of such data. Therefore it is necessary to assume a simplified model of the upper atmosphere and use this model for comparison. Palmen and Newton state that the predominant wave form for the upper atmosphere (between 20,000-40,000ft) has an average wave

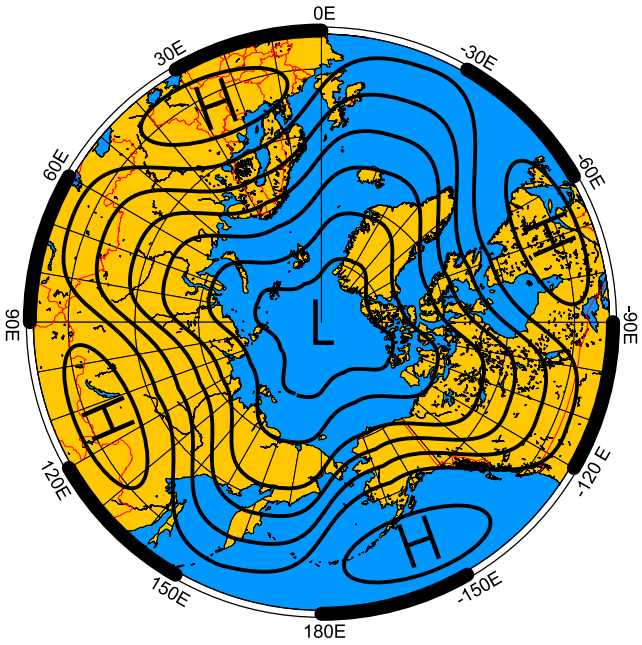
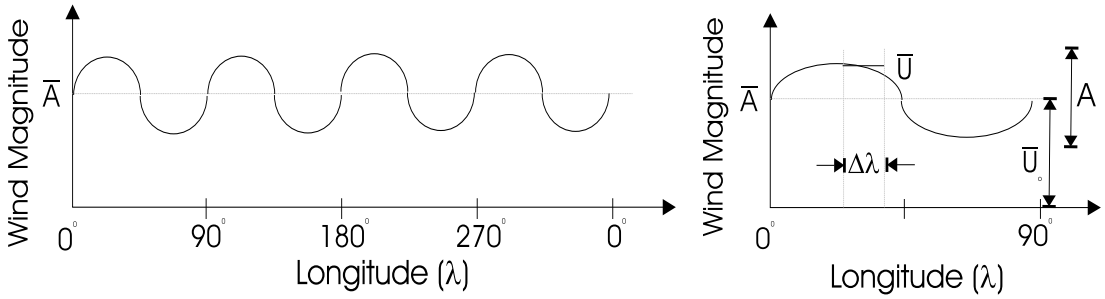


Figure 20: Upper Atmospheric Wave Four Pattern



(a) Complete Wave

(b) Single Wavelength

Figure 21: Upper-level Wave Pattern

length of 90^0, as shown in Figures 20 and 21 [85].

This wave pattern is designated wave four and is an appropriate model to analyse the impact of tiling the NPN wind field across lines of longitude. The NPN model can be assumed to represent a small fraction of the whole wave, as shown in Figure 21 (b), where the $\Delta\lambda$ represents the longitudinal width of the NPN region. Using the wave four model, a generalised equation can be derived to describe the

wave four wind structure, see Equation 13:

$$\bar{A} = \bar{U_o} + A \tag{12}$$

$$\bar{U} = \bar{U_o} + A \int_{\lambda - \frac{\Delta\lambda}{2}}^{\lambda + \frac{\Delta\lambda}{2}} \sin(4\lambda + \epsilon)\, d\lambda \tag{13}$$

where $\bar{U}$ is the mean wind velocity over the longitude range of the original NPN region data at a specific latitude and altitude level; $\bar{U_o}$ is the mean wind magnitude for wave four wave structure; A is the amplitude of the assumed wave four structure; λ is the longitude; and ϵ is the phase angle. By deriving the parameters in this equation from the NPN region data (upon which the wind model is derived), it is possible to generalise the NPN model to a corresponding wave four structure. $\bar{U}$ can be found by examining the data from a set of wind measurements. By calculating $\bar{U}$ over a several day period the variance of $\bar{U}$, $VAR(\bar{U})$, can also be found because over several days ϵ can be assumed to take all values with an approximately uniform density distribution. The $\bar{U}$ in the sample is proportional to the value of A. Therefore A can be derived by varying ϵ to match the highest and lowest values of $VAR(\bar{U})$. Similarly $\bar{U_o}$ can be found by taking the average of the $\bar{U}$ values.

$$\bar{U_R} = \bar{U_o} + A \int_{\lambda_1}^{\lambda_2} \sin(4\lambda + \epsilon)\, d\lambda \tag{14}$$

Once the coefficients for the wave four wind field are known, then the generalised wave form can be used to investigate the variance of the wave four structure over specific longitudinal distances using Equation 14, where the mean wind magnitude over the range of longitude is designated $\bar{U_R}$. The variance for the wave four wind field, $VAR(\bar{U_R})$ can be calculated by varying the phase angle, ϵ in Equation 14. The variance for the NPN model, however, is not a function of longitudinal distance, as it is a repeated pattern and therefore has the same variance regardless of longitudinal distance. The variance for the NPN model is derived directly from the data.

The variance in flight time when using the wave four model can be compared to the flight time variance when using the NPN model using Equations 15 and 16.

$$FlightTime_{wavefour} \approx \frac{Distance}{Velocity^2}(Velocity + \bar{U_R}) \quad (15)$$

$$FlightTime_{NPNmodel} \approx \frac{\frac{Distance^2}{NPNRegionWidth}}{Velocity^2}(Velocity + \bar{U}) \quad (16)$$

4.7 Summary

This chapter developed the requirements for the wind field model required by a high-level, low-fidelity airspace simulation tool. As ground observations and the available forecasted winds aloft data sets proved inadequate, it was decided to base the wind field model on observed winds aloft data provided by the NOAA's Profiler Network of wind radar stations. A method was then presented to transform this data into a set of uniform grids which formed the database for the wind model. The tri-linear wind model used to generate the required wind vector for each simulated aircraft was then described and its overall accuracy verified to within 2.5m/s.

CHAPTER 5

Airspace Module

5.1 Data Link Equipage Assumption

In the future aircraft will rely on high-speed data-links with other airspace users and ground control stations for communication, surveillance, and in conjunction with Air Traffic Controllers, navigation [52, 86, 87, 88, 89]. As described in the introduction, several different technologies are being developed and evaluated to provide these services. The FAA recently announced its endorsement of Mode-S extended squitter technology which is now in operation at the Miami Centre [88]. As the industry is now committed to implementing data-link throughout the U.S. and Europe to ease congestion, it is vital that any airspace simulation have the ability to model aircraft with data link capabilities. Therefore it was decided that this simulation tool would assume all simulated aircraft to be data-link enabled.

5.2 Purpose

Simulating airspace with a complex data-link network poses a computational challenge for real and fast time simulation. It is necessary to develop a data structure to locate aircraft within the range of both airborne and ground ADS-B transmitters. For example, the broadcast range of air-to-air data-link messages is on the order of 10 miles in the TMA and 40 miles en route [90]. Consequently the aircraft within the range of the signal must first be located from the set of all aircraft before the message can be sent, which is a 2-combination problem [91] as shown by Equation 17.

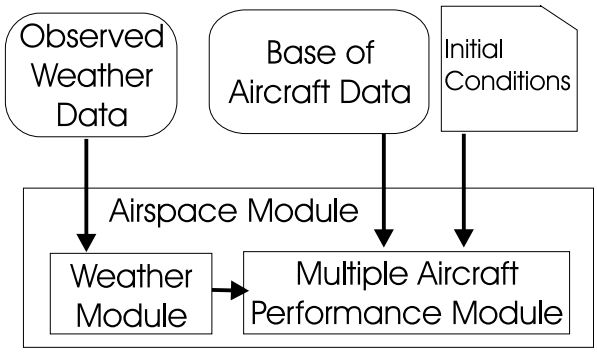


Figure 22: Airspace Module Architecture

$$C(n, 2) = \frac{(n!)}{2(n-1)!} \tag{17}$$

In this equation n is the total number of aircraft. If the total airspace contains 5,000 aircraft, the worst case scenario for sorting through them once requires comparing 12.5 million aircraft pairs, which is a significant number of computations to complete, considering that each comparison may take up to $1\mu s$ and that the objective of this simulation tool is to simulate airspace faster than real-time, i.e. to simulate one second of simulated time in less than one second of real time. However, by properly designing an appropriate data structure and efficient algorithms, this worst-case estimate can be reduced. Additionally, the structure must satisfy Air Traffic Control constraints, such as aircraft separation. In order to simulate both the ATC and data link requirements, a data structure needs to be created to organise the aircraft simulated in the multiple aircraft performance module, into an airspace module as shown in Figure 22

5.3 Requirements

To properly simulate a data-link network, the data structure must support three primary algorithms:

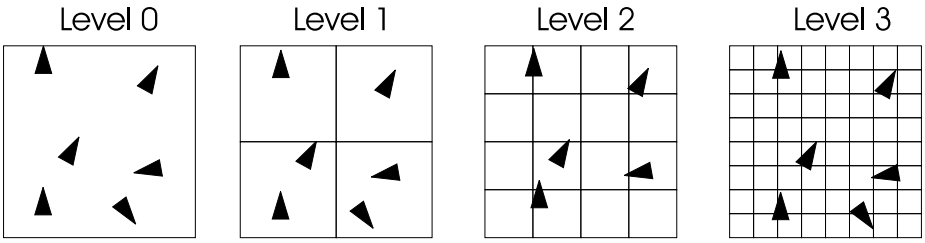


Figure 23: Three Levels of decomposition of a Rectangular Area

- ❖ Locate aircraft within a specified distance of a given static location
- ❖ Locate aircraft within a specified distance of a given aircraft
- ❖ Compute of inter-aircraft distances

The data structure needs to support these functions while minimising the storage size of the structure and maximising the efficiency of the algorithms necessary to complete these functions. An ideal data structure would incorporate the geographic nature of the data into the form of the structure itself, thereby reducing the computational time necessary to locate aircraft within the structure.

5.4 Data Structure Options

While it is possible to arrange the aircraft position data in a number of basic data structures such as an array or a linked list, such a simple structure would not improve upon the worst case estimates described above. Consequently these structures were quickly discounted. The standard data structures which take advantage of the geographic aircraft locations are binary trees, quadtrees, or octtrees [92, 93]. "The term quadtree is used to describe a class of hierarchical data structure whose common property is that they are based on the principle of [recursive] decomposition of space."[93] Consequently any geographic region can be subdivided repeatedly, until a desired level of resolution is reached, as can be seen in Figure 23.

For the application of aircraft geographic location, a quadtree is most appropriate because it allows a two dimensional representation of the surface of the earth

Table 9: Geographic Area and Number of Aircraft Covered in each Leaf Node for Equal Area Point-Region Quadtrees of Varying Levels

Level	Number of Regions	Area per Region km^2	Maximum number of Aircraft per Flight Level
0	1	$5130E+14$	–
1	4	$1.283E+14$	–
10	1,048,576	975.05	~ 20
11	4,194,304	243.76	~ 6
12	16,777,216	60.94	~ 2

to be broken up into reasonably sized regions with a tree of only 10-12 levels. Table 9 gives properties of a Quadtree with 10-12 levels. The area per region is approximated by dividing the area of the surface of a spherical earth by the number of regions in a given quadtree type, as described by Tobler and Chen [94]. The maximum number of aircraft per flight level is approximated by calculating the number of circular regions representing the minimum horizontal aircraft spacing (5 miles) fit into an average size region. Although binary trees can be used to represent geographic regions, they are less appropriate because they generate more levels to represent the same geographic region. Oct-trees, which would allow individual flight levels to be represented by distinct tree regions, would require significantly more regions, as can be seen in Equations 18 and 19:

$$QuadtreeRegions = 4^L \quad (18)$$

$$OcttreeRegions = 8^L \quad (19)$$

where L is the number of levels of decomposition. For example, from Equations 18 and 19 it can be deduced that a ten level octree would have approximately 1024 times as many regions as a ten level quadtree, while representing the same geographic region.

5.5 Quadtree Options

Quadtrees are well documented and have been used extensively for the representation of geographic and spatial information, for location of obstacles in robotic motion [92, 93] or for efficient storage and access of three-dimensional graphics [95]. Figure 24 illustrates the structure of a small quadtree and the region which it represents. Quadtrees consist of three types of tree node: Parent Node, Leaf Node or Empty Node. The total number of nodes in a quadtree is given by Equation 20, where L is the number of levels of decomposition.

$$QuadtreeNodes = \sum_{i=1}^{L} 4^L \tag{20}$$

It is important to note that each Parent Node contains a maximum of four Child nodes, which can be either a Leaf Node or an Empty Node. Each Leaf Node corresponds to a specific geographic region, as illustrated in Figure 24, and contains data about the region or about objects inside the region, e.g. aircraft location. Each Empty node also corresponds to a specific geographic region, but no data is stored in it. Quadtrees can be classified by the method by which they are traversed and the method by which the region is decomposed.

5.5.1 Traversal Method

Pointer quadtrees

Pointer quadtrees contain pointers to the Parent and four Child Nodes in addition to data stored about the region the node represents. By including the pointer information in the node itself the tree can be traversed by following the pointers up to the first common ancestor and then down to the Child Node required. The advantage of the Pointer quadtree is that no external indexing is required to traverse the tree. The disadvantages of the Pointer quadtree are the increasing storage space

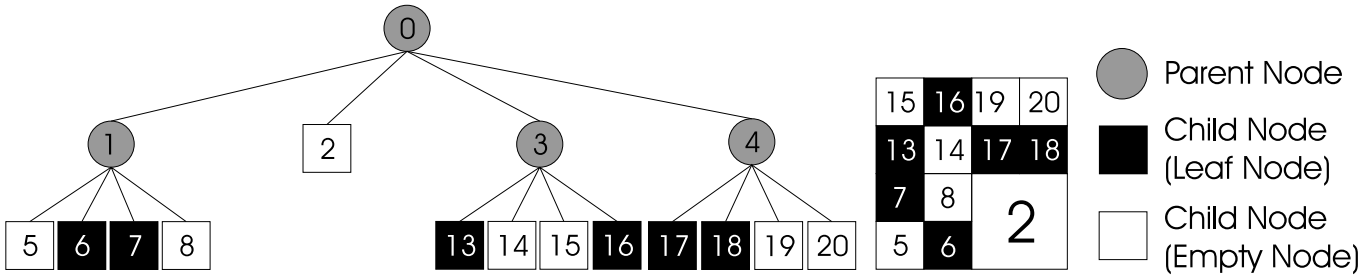


Figure 24: Quadtree Representation of a Rectangular Area

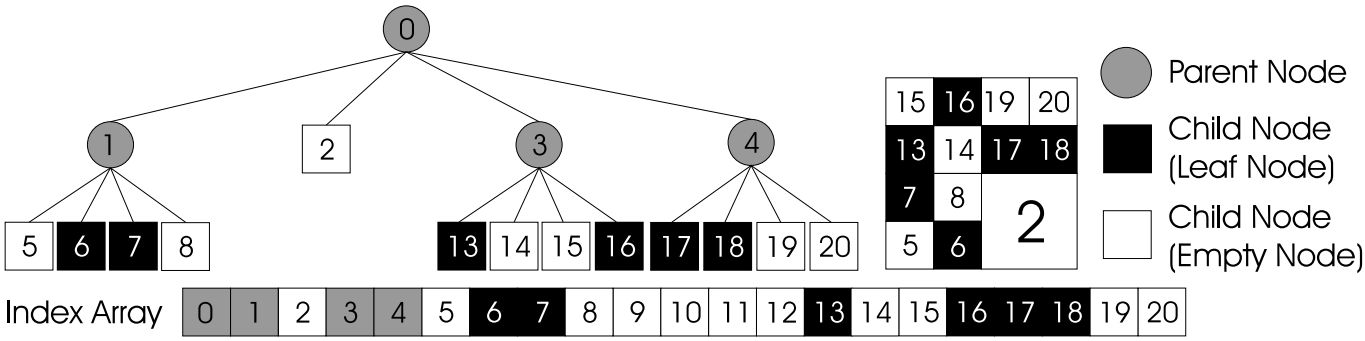


Figure 25: Indexed Quadtree Representation of a Rectangular Area using Morton(Z-Index) Ordering

necessary, 5^n for n levels, and the additional traversal computations required. In a Pointer quadtree, the storage space required increases because each node holds pointers to five additional nodes: four Children and a Parent. However, for sparse quadtrees in which only a small number of nodes contain data, for example air traffic simulation, Pointer quadtrees can provide substantial memory reduction in comparison with Indexed quadtrees, as illustrated in Table 10. However, traversing Pointer quadtrees requires additional computation because each node along a path must be accessed to determine the next parent or child node, e.g. a node three levels up cannot be accessed in one step, but requires three steps.

Indexed/Linear quadtrees

Indexed/Linear quadtrees contain only data stored about the region each Leaf Node represents. The structure of the tree is contained in a separate Index Array, whereas in Pointer trees the structure is contained in the nodes via the parent and sibling pointers. The creation of the Index Array requires that the two dimensional regions represented by the tree nodes be mapped into the one dimensional array. A number of different methods, called tile indexing, are suggested by Samet [96, 93]. Tile index options are more thoroughly discussed in Section 5.5.4. For this comparison, it will be assumed that the tree nodes are numbered sequentially, as illustrated in Figure 25 such that Child and Parent Nodes can be accessed via equations 21-22 where I_C is the Child Index, I_P is the Parent Index, and $n \in [1, 4]$:

$$I_C = (I_P * 4) + n \tag{21}$$

$$I_P = (I_C - n)/4 \tag{22}$$

The disadvantage of the Indexed quadtree is the additional storage space required by the Index Array, which can be quite substantial for sparse trees covering large geographical areas. However this additional storage space is inconsequential for dense trees covering large geographic areas because it is only a small fraction of the space required by the five pointers held in each node.

To illustrate the differences in memory required for airspace simulation, Table 10 compares the memory required for a Pointer and an Indexed/Linear quadtree containing 5,000 Leaf Nodes over 10-12 levels. In these calculations each Leaf Node is assumed to require 16B of storage, and each pointer to require 4B of storage. The Indexed Array size is a function of the quadtree level (see Equation 20), regardless of the number of Leaf Nodes. The Indexed Tree size is obtained by adding the memory required for 5,000 pointer-less Leaf Nodes (16B) to the Index Array size. The Pointer Tree size is the memory required for 5,000 pointer Leaf Nodes (36B) plus the approximate number of nodes necessary to create 5,000 Leaf Nodes in the

Table 10: Memory Required for Quadtrees of Varying Levels with 5,000 Leaf Nodes

Tree Depth Levels	Index Array Size (MB)	Indexed Tree Size (MB)	Pointer Tree Size (MB)
10	5.6	5.77	0.28
11	22.4	25.2	0.30
12	89.5	180.2	0.36

specified number of levels. It is clear that the Indexed quadtree requires substantially more storage space than the Pointer quadtree.

The main advantage of the Indexed quadtree is the computational improvements traversing the quadtree via the Index Array. For example, to traverse three levels up the tree, only two steps are required because the node index of the current node is known and from it the index of the required node can be calculated. For the air traffic application it is clear that the Pointer Quadtree offers substantial memory reduction. However, the Indexed Quadtree size is within the range of modern desktop computers, and affords reduced computation time for node location. It was therefore decided to use an Indexed/Linear quadtree in this simulation.

5.5.2 Decomposition Method

Indexed/Linear quadtrees can be further categorised by the geographic area represented by the quadtree and can be decomposed by several different methods as discussed by Samet [96, 93], which can be broadly classified as either region or vector. A region quadtree is illustrated in Figure 23, where region size is dependent upon tree level and independent of aircraft location. A vector quadtree is illustrated in Figure 26, where region size is independent of tree level and dependent upon aircraft location.

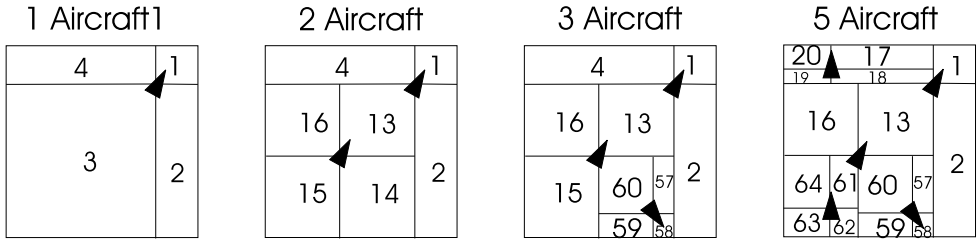


Figure 26: Vector Quadtree Decomposition

As one of the main purposes of the proposed quadtree structure is to quickly locate aircraft within a given geographic region, and given the number of aircraft and their relative locations are variable, the most germane of these options is the Point-Region (PR) quadtree, a specialised form of Region quadtrees designed specifically to store point data, as opposed to vector data. The PR quadtree is most appropriate because the geographic area represented by each quadtree child is one fourth of the area of its parent. Additionally, in PR quadtrees the final structure of the quadtree is independent of order of the node addition, which is necessary since aircraft will be created and removed from the quadtree randomly, and the number of aircraft present each moment will vary throughout the simulation [92].

5.5.3 Spherical Decomposition

It was necessary to develop a method for adapting PR quadtree decomposition to spherical geometry. Tobler and Chen studied three different solutions in [94] to the decomposition problem; two use map projections and one uses platonic solids to approximate spherical geometry. They dismiss both map projection techniques due to excessive complexity and dismiss the use of platonic solids due to poor approximation of the sub-facet edges to great circle distances and the variability in child region sizes. Tobler and Chen [94] devise a system of using authalic coordinates as

described in Equations 23 and 24,

$$\lambda_o = \frac{1}{2}(\lambda_2 + \lambda_1) \tag{23}$$

$$\phi_o = sin^{-1}\big(\frac{1}{2}(sin(\phi_2) + sin(\phi_1))\big) \tag{24}$$

where λ_o is the longitude division point, and ϕ_o is the authalic latitude division point. Authalic coordinates have the advantage of making use of pre-existing spherical gridding system of latitude and longitude by only introducing an authalic latitude to maintain equal area regions during decomposition.

5.5.4 Region Indexing for Linear Quadtrees

To gain the computational savings facilitated by a Linear/Indexed quadtree, it is necessary to map the 2D region represented by a quadtree onto a 1D array. The results of these mappings are known as a space-filling curves [92, 93]. According to Samet [93], the most germane space-filling curves for quadtrees are Morton and Peano-Hilbert ordering, as illustrated in Figure 27 because they complete a quadrant or subquadrant before leaving it, which is consistent with the recursive decomposition inherent in the quadtree structure. In quadtrees, Morton Ordering has several advantages over Peano-Hilbert Ordering such as symmetry and simple recursive generation [93, 92].

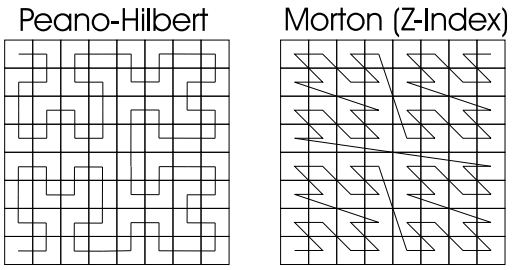


Figure 27: Space-filling Curves

Morton Ordering is primarily implemented using bitsets, which require two to three bytes per decomposition level. This allows the position of each region to be determined by "interleaving the bits of the x and y coordinates"[93] of that region's index. However bitset operations can be complicated for trees with many levels as the bit sets become very long. Additionally, the authalic coordinate system chosen is recursively defined, and therefore difficult to derive an explicit function to map it to a binary index. Without this function, the benefits of using bitset based Morton ordering are negated. It was therefore decided that the Morton ordering would be implemented with `CARDINAL` based indices such that Child and Parent Nodes can be accessed via equations 21-22, where $n = 1$ for the Southwest quadrant, $n = 2$ for the Southeast quadrant, $n = 3$ for the Northwest quadrant and $n = 4$ for the Northeast quadrant, as illustrated in 25.

5.5.5 Point-Region Quadtree Implementation

The quadtree structure implemented in this simulation tool is an Indexed Point-Region Quadtree. The quadtree therefore has two primary structures, an Index Array and a `Leaf_Node Record`. The Index Array is an array of a variant record, `Index Record`, which contains either a pointer to an individual `Leaf_Node Record`, in the case of Leaf Nodes, or nothing, in the case of Parent Nodes or Empty Nodes. In this way the storage space necessary for the Index Array can be minimised, as the Array is a fixed size regardless of the number of `Leaf_Node Record`s. Switching between node types is achieved by switching between `CASE` types.

Each aircraft is represented by a `Leaf_Node Record`. `Leaf_Node Record`s contain pointers to the Aircraft State Array, the Master State Array and to a linked list of other `Leaf_Node Record`s containing aircraft in the same quadtree region, as illustrated in Figure 28. Linking the `Leaf_Node Record`s to the Aircraft State Array with pointers allows the trees to be updated as the aircraft move without the need for additional computation. As the Quadtree is a PR quadtree, only the lowest level Leaf Nodes contain more than one aircraft, i.e. all leaf nodes which occur at a

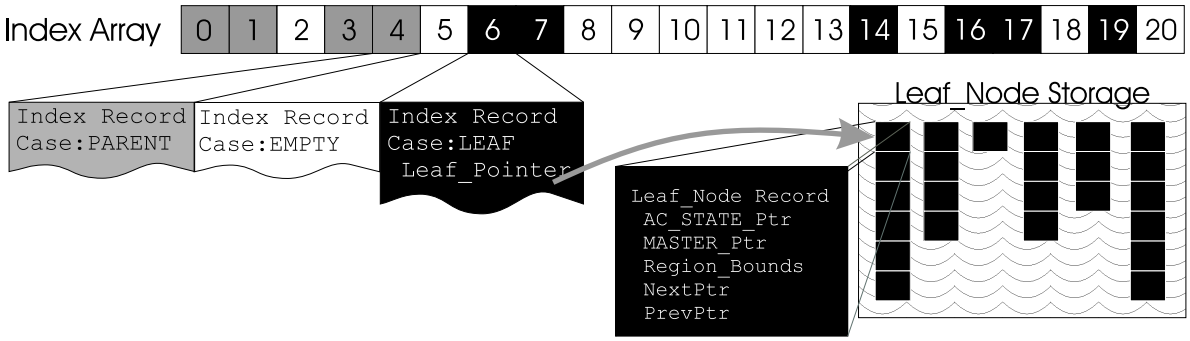


Figure 28: 3-Level Linear Point-Region Quadtree Implementation Example

higher levels will only contain one aircraft because the addition of a second aircraft into that region will trigger a decomposition into two nodes at a lower level. These `Leaf_Node Record`s are held in a linked list, which must be accessed through the Quadtree Index Array.

5.6 Geographic Location Algorithms

The Point Region Quadtree using authalic coordinates for space decomposition described in Section 5.2 allows the three primary functions of neighbour node location, neighbouring aircraft distance computation, and regional aircraft location required for data link and ATC simulation to be efficiently conducted. The following subsections will describe the algorithms developed to perform each of these primary functions, and where applicable, different algorithms developed for the same function will be compared.

5.6.1 Neighbour Node Location

One of the primary functions required to simulate ATC and Data link functionality is the location of `Leaf_Node Record`s (representing aircraft) which reside in

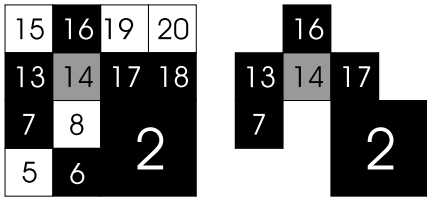


Figure 29: Neighbouring Nodes to Node 18

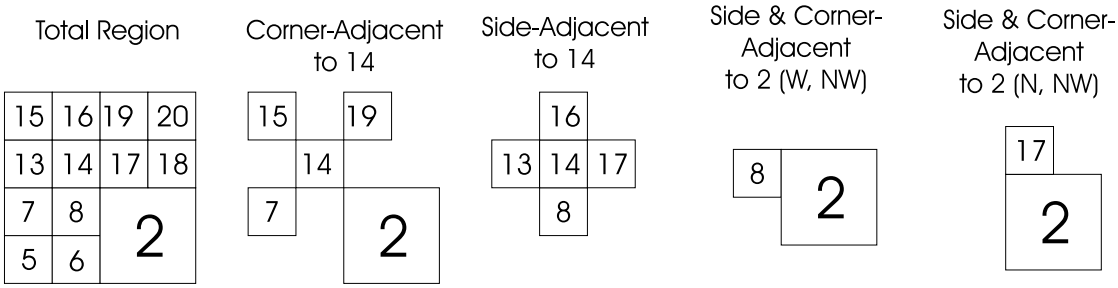


Figure 30: Graphic representation of Samet's Basic Algorithms

neighbouring regions, as illustrated in Figure 29. To locate all nodes which are geographically adjacent to a specified node, three different algorithms are proposed. They employ the same set of basic recursive algorithms for finding nodes adjacent to a corner, a side, or a side and a corner, as outlined by Samet [96] and as illustrated in Figure 30. In Samet's algorithms, the only neighbouring Leaf Nodes at or above the current node level are returned. The proposed algorithms modify Samet's algorithms so that all of the neighbouring nodes are returned, regardless of whether it is an Empty or Leaf Node. This modification was necessary to find neighbouring nodes below the current node level.

Algorithm 1 locates all four corner-adjacent nodes. Starting with the Northeast corner and moving South, all nodes which are adjacent to the East side of the *specified node* are identified using the adaptation of Samet's `CORNER_ADJ_NEIGHBOR` [96] until the Southeast corner node is found. Proceeding clockwise, the same process is repeated on each side of the *specified node*. As the nodes are identified, the Leaf Nodes are entered into an array, which is returned at the end of the algorithm.

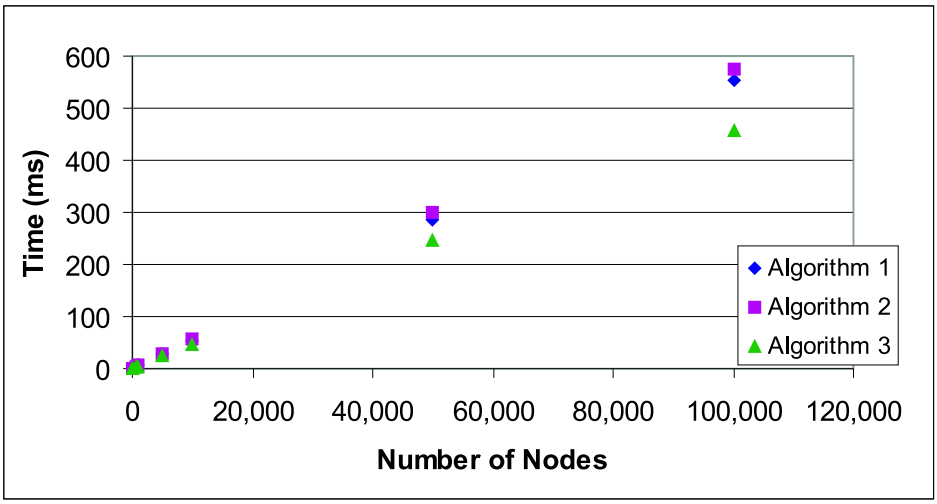


Figure 31: Time to Find Neighbouring Leaf Nodes for a Randomly Generated Quadtree for Varying Numbers of Leaf Nodes

Algorithm 2 locates all four corner-adjacent nodes. Starting with the East side, all the nodes adjacent to the East side of the *specified node* are located using an adaptation of Samet's basic algorithm `GTEQUAL_ADJ_NEIGHBOR` [96]. Proceeding Clockwise, this procedure is repeated for each of the four sides. As the nodes are identified, the Leaf Nodes are entered into an array, which is returned at the end of the algorithm.

Algorithm 3 begins by locating all four corner-adjacent nodes. Starting with the East side, all the nodes adjacent to the side are located using an adaptation of Samet's basic algorithm, `GTEQUAL_ADJ_NEIGHBOR` [96]. Proceeding clockwise, this procedure is repeated for each of the four sides. Unlike Algorithm 2, all the nodes are entered into an array, even if they are not Leaf Nodes. After the array is complete, it is filtered to remove any Empty Nodes and any duplicated Leaf Nodes.

To determine the most efficient of the three algorithms, the algorithms were tested by generating an array of neighbouring leaf nodes for a randomly generated tree containing different numbers of leaf nodes. The trees were consistent for all the algorithms and were limited to ten levels. The results from these trials is shown in Figures 31 and 32, where the time required to complete the task, in milliseconds (ms), is plotted against the number of leaf nodes in a given tree. It is clear from

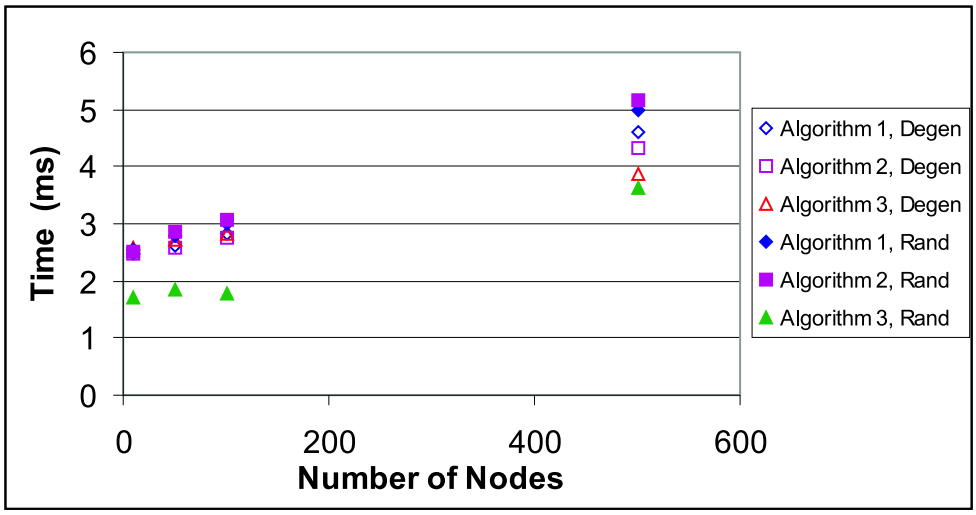


Figure 32: Comparison in Time to Find Neighbouring Leaf Nodes between Degenerate and Randomly Populated Quadtrees

the results that Algorithm 3 is the most efficient, and that the efficiency increases as the number of nodes in a tree is increased. In practice, based on likely number of aircraft to be simulated, the variance between the three algorithms is less than 1 ms.

In addition to testing the efficiency of the algorithms for randomly generated quadtrees, the performance was also evaluated for degenerate trees, as the simulator will be used to simulate sections of airspace considerably smaller than the globe. A degenerate tree is any tree whose leaves are not evenly distributed. For this analysis degenerate trees were created that had at most two children on any level. An identical test was performed on a degenerate set of quadtrees. Memory considerations and the ten level tree restriction reduced the maximum size of quadtree to 500 nodes. The results from these trials can be compared with the timings with randomly generated trees in Figure 32, where the time required to complete the task is plotted against the number of nodes in a given tree. It is also clear from the results that Algorithm 3 is the most efficient, for trees with at least 500 Leaf Nodes. However, unlike the randomly generated case, the differences between the three algorithms becomes negligible in degenerate trees with fewer than 500 nodes.

5.6.2 Inter-Aircraft Conflict Detection

To maintain proper inter-aircraft spacing, it is necessary to determine if aircraft are within a specified horizontal or vertical distance from one another. To minimise the computation time necessary to calculate the distance between each set of possible aircraft conflicts, the number of aircraft pairs are first minimised using the Quadtree by only comparing the aircraft in neighbouring nodes. First intra-node aircraft pairs are computed for the Leaf Node with the lowest Quadtree Index, which is designated the Comparison Node. Then inter-node aircraft pairs are computed for each neighbouring node with a Quadtree Index higher than the Comparison Node. All aircraft pairs which are within a specified distance are added to an array. Then the Leaf Node with the next lowest Quadtree Index becomes the Comparison Node and the process is repeated. In this way the distance between any two aircraft pairs is computed only once, and only aircraft in neighbouring nodes are compared. The distances are calculated using the equation for a great circle distance given in Equation 25 as given in [7], where d is the angular distance given in radians, $\Delta\phi$ is the difference in Latitude, and $\Delta\lambda$ is the difference in Longitude.

$$d = \sqrt{\Delta\phi^2 + cos(\phi)cos(\lambda)\Delta\lambda^2} \tag{25}$$

5.6.3 Location of All Aircraft within a Circle

To simulate data-link transmissions it is necessary to locate all aircraft within a circular search region specified by a centre point and a radius. To determine if an aircraft is within the region it is necessary to calculate the distance between the aircraft and the centre of the region.

To minimise the computation time, the number of aircraft to be so tested are first reduced using the quadtree structure. The trapezoid representation of the circular region is tested to determine if it is contained within the Quadtree Node in which the centre of the required region falls, as illustrated in Figure 33. If it does, as

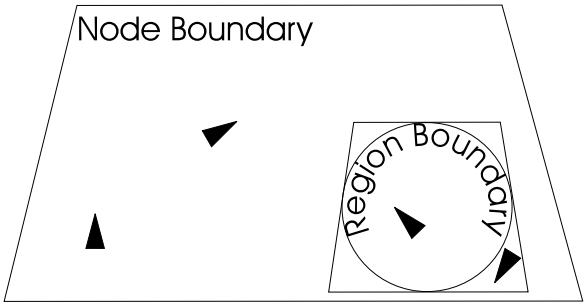


Figure 33: Search Region within Node Boundaries

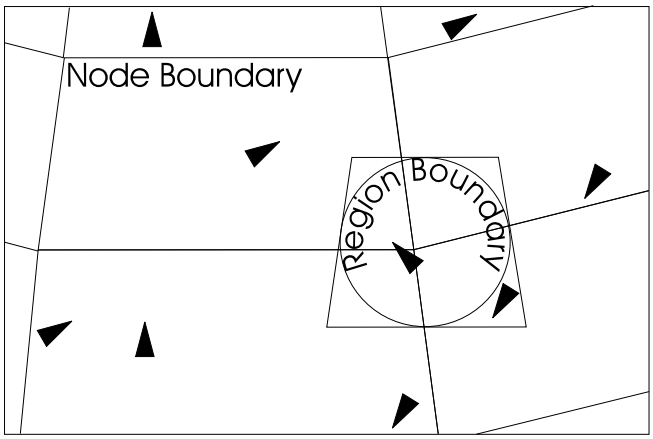


Figure 34: Search Region Spanning Several Nodes in Search Algorithm

seen in Figure 33, then all of the aircraft within that Node are logically tested to see if they fall within the trapezoid representation of the circular region. The distance is computed only for aircraft falling within this trapezoid to determine which of these aircraft are within the search region.

However if the trapezoid representation of the circular region is not fully contained within the Quadtree Node, then all eight of the neighbouring nodes are located, as illustrated Figure 34. The level of the neighbouring nodes is determined by the radius of the search region, so that the neighbouring nodes will always fully contain the search region.

5.7 Quadtree Primitives

As the quadtree is a dynamic data structure, where aircraft are randomly added and removed, and aircraft cross from one region into another at random, it is important

to develop a set of primitives which are computationally efficient for the construction, maintenance and disposal of the PR quadtree. Three such primitives are described below.

5.7.1 Quadtree Creation and New Node Insertion

The Quadtree is created during the initialisation phase of the simulation, from the Current Master Array. When new aircraft are added to the Quadtree, a new `Leaf_Node Record` is dynamically allocated, the appropriate `Index Record` in the Quadtree Index Array is switched to a Leaf Node, and a pointer to the new `Leaf_Node Record` is set. Using the aircraft's location, the index is found by traversing down the Index Array until either a Leaf Node or an Empty Node is found. If the node is empty, then the `Index Record` is switched to Leaf Node case and the pointer is set to the new Leaf Node. For example, if an aircraft were added to the region labelled 5 in the Quadtree represented in Figure 25, the Index Array search would begin at Parent Node 0 then traverse to Parent Node 1, then to Empty Node 5, which would then be converted into a Leaf Node.

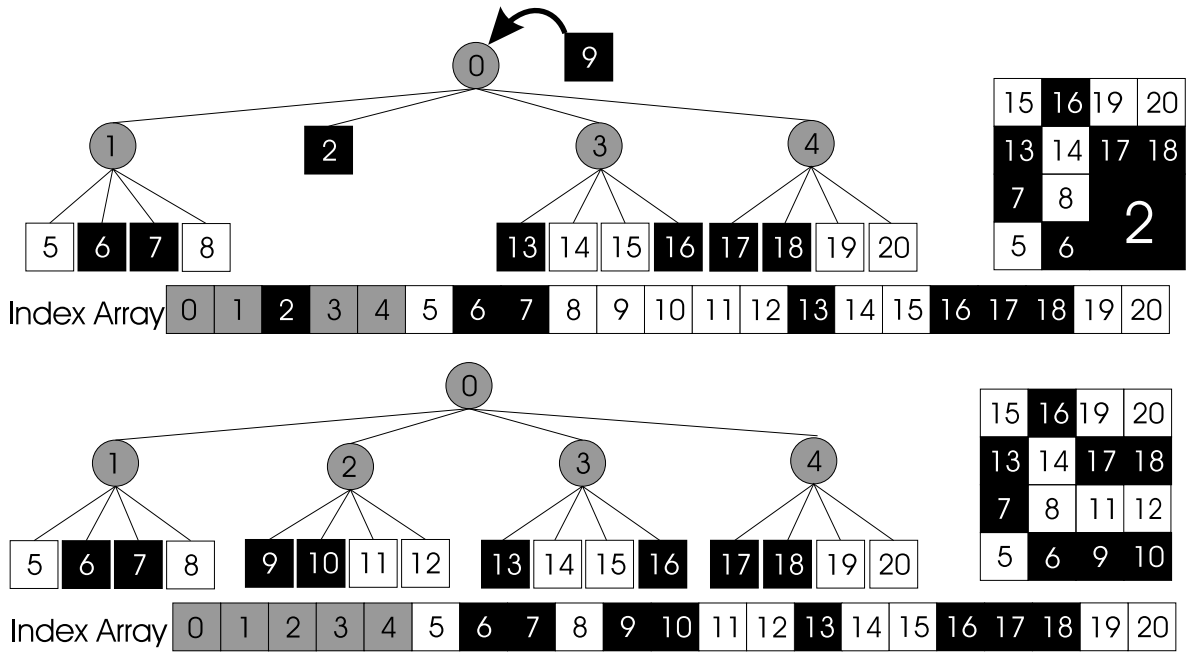


Figure 35: Insertion of New Node into an Indexed Point Region Quadtree

If, however, the terminating node is already occupied by a Leaf Node, then the `Leaf_Pointer` to the `Leaf_Node Record` is set to `NIL` and the Index Array entry is switched to a Parent Node. The algorithm is called starting from the new Parent Node to place the former Leaf Node into a lower level in the Quadtree following the process outlined above. Once the former Leaf Node has been successfully placed, then the algorithm is again called starting from the new Parent Node to place the new Leaf Node. For example, an aircraft with location in region 9 is added to an existing Quadtree, as shown in Figure 35. The new Leaf Node traverses down the tree from Parent Node 0 to Leaf Node 2. Upon reaching Leaf Node in region 2, the Leaf Node is transformed into a Parent Node and the former Leaf Node is moved down a level to region 10. Then the new Leaf Node is placed by traversing down a level from Parent Node 2 and is placed in to an Empty Node in region 9. However, if the old Leaf Node which was found in region 2, was subsequently placed into region 9, then the process of replacing the current Leaf Node and continuing to try to place the new Leaf Node would continue until the maximum number of levels in the Quadtree had been reached.

If a Leaf Node currently resides at the lowest allowable level in the tree, then the new `Leaf_Node Record` is added to the end of linked list, as illustrated by Figure 28. These `Leaf_Node Record`s must be referenced through the linked list structure by existing `Leaf_Pointer` in the Quadtree Index Array.

5.7.2 Updating the Quadtree

Once the quadtree has been created, it must be periodically updated so that as the aircraft move, the tree will continue to accurately represent their position. The update frequency is a function of the accuracy required by all the functions using the quadtree. For example to match current ATC en-route radar accuracy the tree would need to be updated every 12s. However, if there are few aircraft travelling relatively slowly, then this update rate may be decreased if the impaired accuracy is not found to impact on the ATC or Data-link functions. Updating the Quadtree involves

three steps. The first is to search the Quadtree to find all aircraft which have left the boundaries of their region. The second step is to remove these aircraft and place them in a temporary list. The final step is to re-insert each aircraft in the list back into the Quadtree. This update is accomplished by removing the `Leaf_Pointer`s from the Quadtree Index Array and placing them into a temporary list, and then removing them from the list and placing them back into the Quadtree Index Array. During the entire process the actual `Leaf_Node Record`s are untouched, only the `Leaf_Pointer`s which point to them are re-organised.

5.7.3 Removing Aircraft

To minimise the number of small allocations of memory, `Tree_Node Record`s are allocated upon aircraft creation and deallocated upon aircraft removal from the simulation. Thus, an aircraft, represented by a `Tree_Node Record`, is only removed from the Quadtree when it has reached its destination or when the simulation is terminated. As aircraft removal can happen at any time, the removal of an aircraft from the Quadtree may require that the tree be updated first so that all aircraft will be in their appropriate regions.

The first step in removing an aircraft is to locate it in the Quadtree. If the aircraft is not found, it must have moved out of the quadtree region since the last quadtree update. The Quadtree is then updated and the search process begins again. After the aircraft is found, the corresponding `Tree_Node Record` is deallocated and the `Leaf_Pointer` in the Quadtree Index Array is set to `NIL` and the corresponding `Index Record` switched to an Empty Node.

However, if the aircraft removed was the only aircraft in a node, i.e. the `Leaf_Node Record` removed was the only `Leaf_Node Record` in the linked list, and there is only one other sibling (which is now an only child), then that sibling node must be moved up the quadtree until it is no longer an only child. For example, if the aircraft represented by the `Tree_Node Record` in region 9, as illustrated in Figure 36 is to be removed, then the `Tree_Node Record` in region 10 will then be

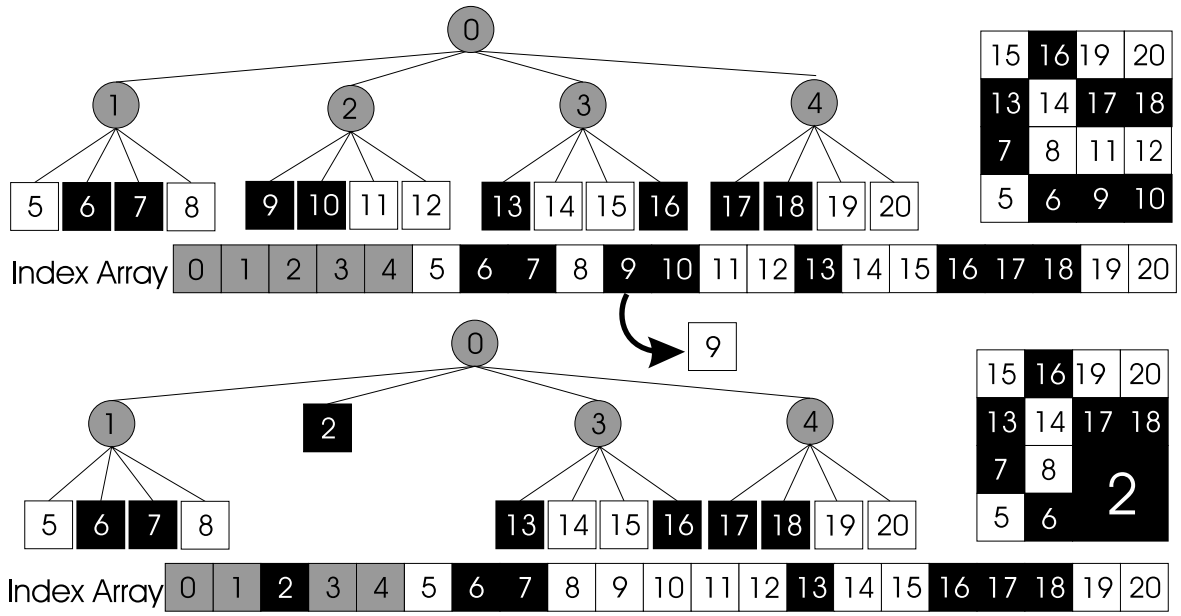


Figure 36: Deletion of an Aircraft in an Indexed Point Region Quadtree

an only child. If that region contains only one `Leaf_Node Record` or aircraft, in its corresponding linked list then the `Leaf_Node Record` in region 10 needs to be moved up one level. At this point the `Index Record` in Index 10 is switched to Empty and the `Index Record` in Index 2 is switched to a Leaf Node which will now point to the `Leaf_Node Record` representing the remaining aircraft.

5.8 Summary

This chapter presented a data structure for organising the aircraft simulated in the multiple aircraft performance module in order to facilitate the simulation of a data linked network and airspace structure. A Point Region quadtree data structure was chosen because it was independent of the order of aircraft addition. The indexed quadtree traversal method was chosen over the pointer method because although it required more storage space the index structure offered computational benefits. The quadtree decomposition uses authalic coordinates to maintain equal area regions. Algorithms for PR indexed quadtree creation, maintenance and destruction have been presented. Algorithms for nearest-neighbour node finding, inter-aircraft

conflict detection and the location of all aircraft within a circle have also been presented. These algorithms form the basis for the airspace module.

CHAPTER 6

Air Traffic Control Module

Thus far, this thesis has concerned itself with the creation of a discrete event aircraft simulation capable of simulating multiple aircraft following multiple flight plans. Similar to their real counterparts, these simulated aircraft operate in isolation, as they have no knowledge of the whereabouts and movements of other aircraft.

In reality however, aircraft do need to concern themselves with other aircraft, and due to their inherent lack of information about other airspace users, they rely upon Air Traffic Control Service Providers (ATCSPs) to monitor the airspace and to to provide safe separation for all airspace users. Unlike individual aircraft, the ATCSPs are aware of all of the airspace users and the airspace structure in which they are operating. Their primary function is to monitor the airspace for possible conflicts and to communicate any pertinent information to the aircrews of the aircraft under their control. The pilots in turn also communicate with the ATCSPs, providing them with position updates and intentions. ATCSPs accomplish this function of aircraft separation by establishing and maintaining an airspace structure and a set of standard operating procedures to be used inside that airspace structure.

This thesis has not yet considered the representation of the airspace structure and the rules under which the air traffic operate. The airspace structure contains two elements: the air route structure which is based on a series of ground based radio navigation beacons and airspace divisions called centres and sectors; and the Air Traffic Management (ATM) procedures implemented by ATCSPs. In order to complete the airspace simulation tool, a simulation equivalent of the airspace structure with the associate ATSP is required.

As the airspace simulation module is implemented as a discrete event simulation, it follows that the airspace structure, or the ATC Module should also be

implemented as a discrete event simulation capable of interacting with the air traffic simulation by monitoring aircraft positions and issuing instructions to specific aircraft.

Additionally, recent advances in navigation technology and the desire to increase system capacity have resulted in a high degree of uncertainty in the evolution of the air space structure over the next 20-30 years [3, 10, 97, 98, 99]. Thus an airspace simulator which can flexibly implement different ATM strategies will be very useful to the research community as it seeks to develop new strategies to accommodate the changing air space structure.

This chapter outlines the control module's requirements and describes the implementation of the simulation of controller functions. Section 6.1 will explain the requirements of the control module and the need for a flexible and efficient implementation of controller logic. Section 6.2 will discuss the architecture used to implement the controller logic, and how the requirements and functions were divided up between the controller modules and pilot module. Section 6.3 will provide background on Linux interprocess communication tools, and relate them to the control module communications requirements. Section 6.4 will describe the controller module implementation and data structures. It will also include an introduction to the controller vocabulary, script interpreter, stack creation, and stack machine execution of controller logic. Section 6.5 will define the control module's interaction with the individual aircraft simulation, and the implementation of the commands issued by the control module. Concurrent operation of the simulation modules will be addressed in Section 6.6. Section 6.7 will discuss the performance characteristics of the control module, including the time responsiveness of the controller to detecting conflicts, and to responding to updates. A simple example is given in Section 6.8 to illustrate the control module's response time. Section 6.9 will summarise the controller module and its role in the air space simulation.

6.1 Control Module Requirements

6.1.1 Functional Requirements

The overall function of the Air Traffic Control module is to implement a system which would make the simulated aircraft behave as if they were under positive ATC guidance. Accordingly, the control system is responsible for the safe separation of aircraft, the avoidance of convective weather conditions and the avoidance of restricted airspace while maintaining efficient traffic flow. This task involves the ATC monitoring the aircraft movements, invoking some set of logic to simulate controller behaviour, issuing commands and modifying aircraft behaviour to comply with those commands. These commands include altitude clearances, altitude and speed restrictions, and flight plan alterations. Presently, the ATC module is only required to maintain aircraft separation and is not responsible for maintaining efficient traffic flow, avoiding severe weather, or avoiding restricted airspace.

6.1.2 Communications Requirements

In reality the air traffic controllers work very closely with the aircrews and and are in constant communication with one another. This constant communication is necessary because without it, neither the pilot or the controller have the information they require. The controller is dependent upon the aircraft to provide current position passively via secondary surveillance radar, and on the pilot to provide intent actively via verbal communication. In turn the pilot is dependent upon the controller to provide altitude and speed clearances to maintain safe separation. Simulating both the passive and the active forms of communication between the airspace module and the ATC module efficiently is therefore a key requirement of the ATC module.

6.1.3 Modelling Controller Behaviour Requirements

As discussed in Section 6.1.1 the ATC module has two distinct functions. The first is to detect any possible loss of separation, formulate a resolution strategy and implement the resolution strategy by issuing flight plan corrections. The second function is to execute the resolution strategy corrections issued by altering the flight plan of the affected aircraft. These functions correspond to the role of the air traffic controller and the flight crew.

As this simulation is a high-level airspace simulation, it is not interested in modelling a controller's cognitive process, but rather in modelling the results of those processes. Therefore, to fulfill its first function of maintaining aircraft separation, the ATC module must employ an algorithm to mimic the results of a controller's behaviour rather than the distinct behaviours themselves. This algorithm needs to be devised and implemented in a flexible way to address the uncertainty in future airspace constructs. Additionally, it needs to be efficiently applied as it will be called repeatedly during the simulation. Since a controller's goal is to maintain safe separation and the controller is employing a standard set of procedures to achieve this goal, it is conceivable to conceptualise an algorithm to encode the underlying logic used by the controller. This leads to the idea of encoding the logic at initialisation and executing it when necessary, to increase efficiency. Implementing a set of logic in such a way introduces two distinct requirements. The first requirement is to encode the logic, and the second is to execute the logic.

6.1.4 Modelling Pilot Behaviour Requirements

To fulfill its secondary task of implementing commands issued by the ATC agent, the ATC must incorporate a module to manipulate the flight plans of the aircraft affected. As with the requirements to model controller behaviour, it is beyond the scope of this simulation tool to model flight crew behaviour beyond a time delay model, i.e. the pilot behaviour is treated as an agent who completes the task set without error in a finite period of time. As such, the requirement for modelling

the pilot's behaviour is reduced to correctly implementing the flight play alterations contained in the ATC module command.

6.1.5 Response Time Requirements

To correctly simulate the airspace structure the ATC module would need to carry out all of its functions within a representative time window, in order to correlate with reality. Previous studies into controller response time are not available in the literature, however studies into the response time of pilots to controller commands have shown that on average pilots can respond to a verbal command in 10s, or to a data link command in about 25s [54, 100]. However these timings do not cover the time taken by the controller to maintain her *mental model* of the airspace, the time to locate any possible conflicts, or the time to determine a solution. As these timing studies only cover the pilot's half of the overall response function, it is conceivable that the overall response time, i.e. the time from the aircraft position update until the time of pilot command execution, could be on the order of 10-50s. Consequently one of the requirements of the ATC module is to implement its functions within this time window.

6.1.6 Requirement Summary

The following points summarise the ATC control module requirements:

- ❖ The controller modules must maintain a current representation of the geographic location of the aircraft under their control.
- ❖ The controller modules must monitor the aircraft positions for possible loss of separation.
- ❖ The controller modules must devise a plan to maintain separation distances and develop commands to implement the plan.
- ❖ The controller modules must communicate the commands to the appropriate

aircraft.

- ❖ The ATC module must efficiently simulate the transfer of data between the controller modules and the airspace module.
- ❖ The controller module must separately encode the controller logic at initialisation.
- ❖ The controller must execute the control logic repeatedly during the simulation.
- ❖ The pilot module must correctly alter the aircraft flight plans in accordance with commands issued by the controller module.
- ❖ The ATC module must provide a controller response time between 10-50s.

6.2 ATC Module Architecture

Following the requirements discussed above, the implementation of the control module led to two distinct sub-modules: a controller module and a pilot module, as shown in Figure 37. The controller module is implemented as a separate process to the air traffic module; in practice the controller is a separate entity operating in its own time frame and with a limited knowledge of the airspace environment. Implementing the controller as a separate process ensures that the only knowledge the controller has of the airspace environment must be explicitly passed to it. The pilot module is, however, implemented as an addition to the air traffic module because it must interact closely with the air traffic performance module. The controller module and the pilot module communicate with one another via a form of inter process communication, which is represented in Figure 37 by solid arrowheads. These communication channels simulate information being passed between the aircraft and controller either actively via VHF radio or data-link communications or passively via secondary surveillance radar. Universal time is maintaining by the airspace simulation and is accessible by the controller module. The dashed lines indicate access to the information in the direction indicated by the arrowhead. For example the controllers can access the universal time, but cannot alter it.

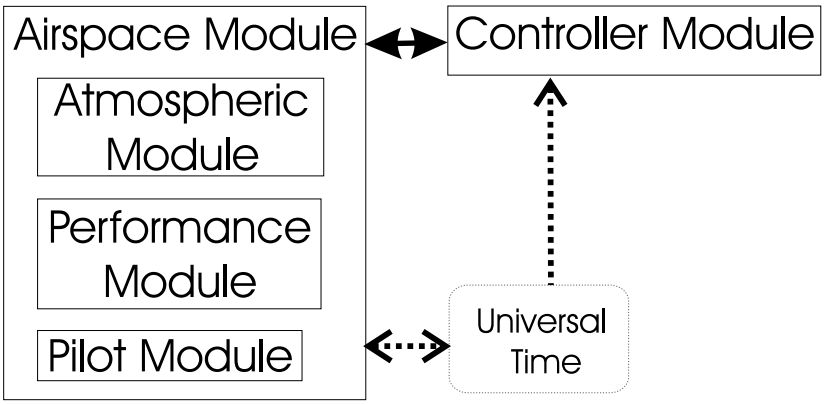


Figure 37: Control Module Architecture

6.3 Interprocess Communication Via Message Queues

It has been established in Section 6.2 that providing autonomy for the controller module from the rest of the airspace module is desirable to more accurately simulate the separate airspace players. However, this autonomy poses a challenge to provide the controller with the appropriate data at the appropriate time and doing so across multiple separate Linux processes. Thus it is necessary to develop a set of messages to pass the appropriate data between the autonomous controller modules and the airspace module, and to develop or utilise a method of passing the messages between processes.

Six standard messages were developed to allow the airspace module to communicate with the controller modules. These messages are listed in Table 11. It is important to note that, from the standpoint of the controller module, all of the messages (with the exception of the Update Aircraft message) are sent asynchronously.

In order to allow an autonomous process to communicate, it was necessary to develop a suitable form of interprocess message passing. The Linux operating system provides a set of tools which allow information to be passed between separate processes, the most basic of which is a pipe. Pipes transfer the output of one process to the input of another process, providing the two different processes have been started by a common ancestor process [101]. To connect unrelated processes

Table 11: Interprocess Message Description

Message Name	Purpose	Sender Module	Recipient Module	Time Stamp	Blocking Mode
Initialisation	Sends airspace initialisation information	Airspace	Controller	No	Yes Airspace
New Aircraft	Sends aircraft initialisation information	Airspace	Controller	No	Yes Airspace
Update Aircraft	Sends updated aircraft state information	Airspace	Controller	Yes	No
Remove Aircraft	Removes aircraft from controller data structures	Airspace	Controller	No	Yes Airspace
Terminate	Shutdown the controller module	Airspace	Controller	No	Yes Airspace
WILCO	Understand & Will Comply acknowledgement of last message	Airspace, Controller	Airspace, Controller	No	Yes Controller
Command	Commands aircraft to alter flight plan or transfer controller	Controller	Airspace	Yes	No

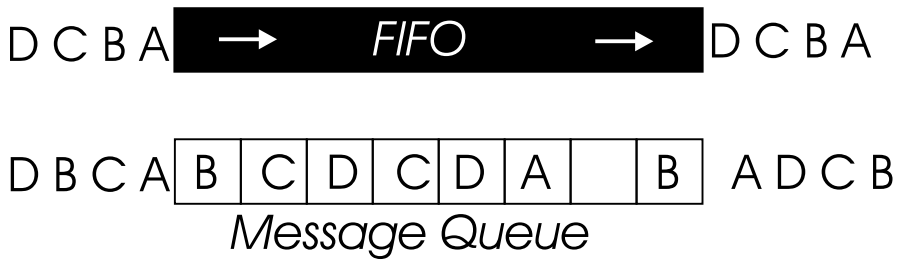


Figure 38: Interprocess Communication Tools

it is necessary to use a a named pipe or FIFO, which is external to either of the processes it is connecting. Unfortunately pipes work on the principle of a first-in first-out queue, as illustrated in Figure 38 so unless the order of the information that is being passed into the pipe is known, then the receiving process cannot correctly process the data. For example if message A contains eight bytes but message B, C

and D contain only 4 bytes, the receiving process will not know how much data to receive from the pipe unless it what order the messages are sent in.

In a situation such as the case with the simulation tool where different types of messages are randomly sent, it is necessary to identify what type of message was coming through the pipe. A message queue, which is a more advanced FIFO, has the ability to identify messages by type and remove different typed messages from the queue on request [101]. This ability is illustrated in Figure 38 as a list of boxes, which the receiving process has access to. The receiving process has the ability to pull off the first message of a given type to receive messages in the order it requires. Message queues also allow a process to be suspended while it waits for a certain message type or to return to the calling function without retrieving any messages, known as non-blocking mode [101]. For example, in blocking mode if the receiving process was using the message que in Figure 38 and needed the next message it received to be of type E, then it would suspend the process because there are no messages of type E. The process would remain suspended until a message of type E was placed into the queue. This feature facilitates faster simulations by making more efficient use of processor time. Message queues were therefore implemented to pass data between the airspace and the ATC module.

To determine how efficient a message que would be at transmitting messages between the control modules and the airspace module, a representative message que was created and a message the same size as the Update Aircraft message was sent between two processes. The number of messages sent was varied between 100 and 1,000,000. The results of these trials are illustrated in Figure 39. The relationship is linear when plotted on a log-log scale. It is clear that to send a million messages would take just over 1 second. From this data the impact of using message ques to transmit data can be extrapolated. Assuming the simulation has 350 simultaneous aircraft each updating every second for a 24 hour simulation period, the impact to the overall simulation time would therefore be approximately 100s, which is less than 5% of the overall simulation run time for simulations running up to 50 times faster than real time.

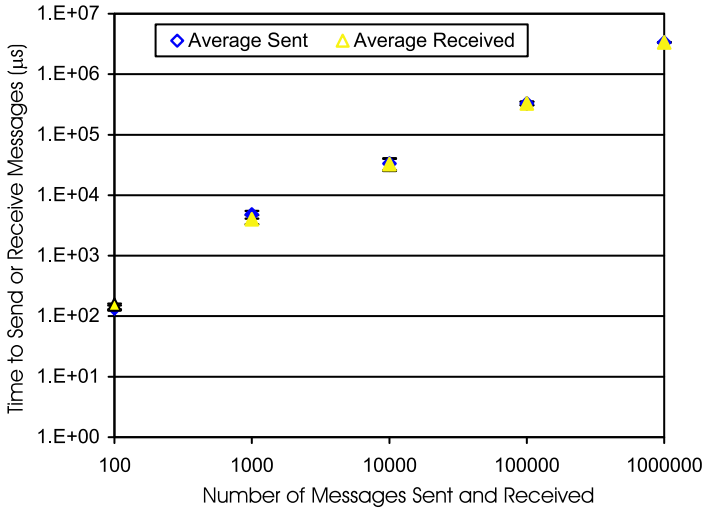


Figure 39: Message Queue Timing

6.4 Controller Module

The controller module is required to locate any potential loss of separation, to formulate a solution strategy and to implement the solution strategy by issuing commands to the appropriate aircraft. In so doing, the controller module must be flexible, i.e. it must be able to implement different control strategies for different sections of airspace. The controller module must also have a mechanism to easily change the control strategy in order to investigate the impact of different strategies on the air traffic flow. The following sections describe the data structures of the controller module.

6.4.1 Implementation Options

Without the emphasis on a flexible implementation for research purposes, the simplest way to implement the controller module ATM strategy would have been to create a standard set of ATM strategies, and allow the user to invoke an appropriate strategy. However since one of the main objectives for this simulation tool was to facilitate ATM research, it was felt that a simple set of standard ATM strategies

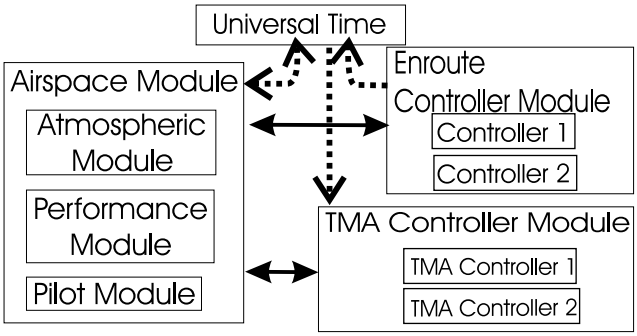


Figure 40: Detailed Control Module Architecture

would be too restrictive. Instead it was decided that a limited controller vocabulary could be assembled, similar to the control text files used in RAMS and TAAM. This approach would enable the ATM strategy to be imported from a text file and to be interpreted by the controller, thus fulfilling the requirements set out to establish the controller logic at initialisation and implement it during run time. By developing a controller vocabulary, the simulator could then facilitate many more combinations of ATM strategy. Additionally, the vocabulary could be easily expanded for future work.

To further increase the flexibility of the simulation and its ability to correctly simulate the airspace structure, it was necessary to implement different types of controllers. The present airspace structure employs many different types of controllers for different phases of flight, such as ground controllers, tower controllers, terminal area maneuvering (TMA) controllers, and enroute (radar) controllers [87]. Because this simulation tool is not concerned with simulating aircraft ground operations or low-level procedures it was not necessary to simulate ground controllers or tower controllers. It was felt that it was possible to adequately simulate the current airspace structure using two representative controller types: enroute controllers and TMA controllers. The TMA controllers would have control of an aircraft from initial approach through to landing, and the enroute controller would have control of all other phases of flight. The expanded control module architecture is illustrated in Figure 40.

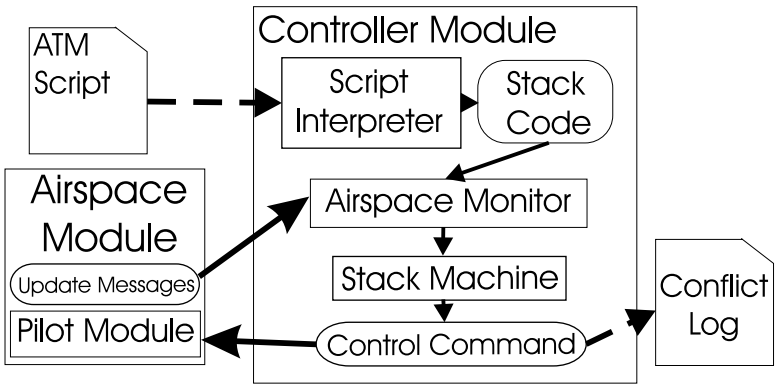


Figure 41: Generic Controller Module Architecture

6.4.2 Data Structure Overview

The controller module, which is tasked with maintaining adequate aircraft separation is implemented as three sections as illustrated in Figure 41: the script interpreter, the airspace monitor, and the stack machine, which correspond to its three requirements. At initialisation a script file is read in and converted to an array of logical instructions by the script interpreter to be used by the stack machine. During the simulation phase, aircraft update messages are passed from the airspace module to the airspace monitor, which maintains the controller module's "mental model" of the aircraft under its control. At specific intervals the control logic is invoked by executing the stack machine, which creates command messages and transmits them to the pilot module. The pilot module then executes the commands and transmits a WILCO message back to the controller. Once the stack machine has completed its execution of the array of logical instructions, the controller returns to its airspace monitoring state. These three sections of the controller are described in detail in subsections 6.4.3-6.4.7. The differences in the implementation of the two controller types are also discussed in the following subsections.

6.4.3 External Script Files

The purpose of the external script files is to allow a wide variety of ATM strategies and procedures to be implemented in a form which is understandable to both system

developers and professional controllers. To fulfill this requirement a set of controller vocabulary and syntax are developed to describe the logical procedures implemented in air traffic control procedures. These text files are written in a pseudo-code style where conditional `if-then-else` and `while-do` statements provide a logical structure. Appendix A contains all of the words which can be used in a script file along with their appropriate usage and implications. These air traffic control procedures are then stored as external script files. An example of a simple script file is presented in Figure 42.

The control vocabulary is made up of five different types of words: key words, calculation words, action words, internal variable words and external variable words. Key words occur at the start of each line and determine the action to be taken for that line. In the example, the following are key words: `set`, `get`, `s_while`, `increment`, `command`, `end`, `stop`. Internal variables are variables which can be set directly via the script file such as `conflict_radius` and `current_conflict`. Internal variables can be `set` and `incremented`. Conversely external variables are variables which cannot be set directly via the script file, such as `no_conflicts`, `ac1`, `ac2`, and `sector_id`. Calculation arguments are words which follow the key word `get`, such as `conflicts`. Action arguments are words which follow the key word `command`, such as `RESTRICT_LOWER_FL` and `RESTRICT_UPPER_FL`.

The sample script file begins by setting the conflict radius to 30 nautical miles. Then all conflicts within that conflict radius are acquired in an array, and the number of conflicts is returned in the external variable `no_conflicts`. For each

```
1: set conflict_radius 30 nm
2: get conflicts sector_id conflict_radius
3: s_while ((current_conflict < no_conflicts) and(no_conflicts > 0)) do
4:   increment current_conflict
5:   command RESTRICT_LOWER_FL ac1 (ac1_alt + 1000 ft) (30nm / ac1_vel)
6:   command RESTRICT_UPPER_FL ac2 (ac2_alt - 1000 ft) (30nm / ac2_vel)
7: end
8: stop
```

Figure 42: Sample ATM Script File

conflict found, the leading aircraft `ac1` is commanded to restrict its lower flight level to 1,000 feet above its current altitude for a variable number of seconds, and the trailing aircraft, `ac2` is commanded to restrict its upper flight level to 1,000 feet below its current altitude for a variable number of seconds. The number of seconds could have also been explicitly specified. This script example illustrates a simple vertical conflict resolution procedure.

6.4.4 Script Interpretation and Stack Compilation

During the initialisation phase of the simulation the external script file is interpreted by the script interpretation algorithm. The script interpreter produces an array of logical instructions to be used by the stack machine to implement the ATM procedures and strategy. This array of logical instructions will increase the computational efficiency of the control module by reducing the complex logic into stack machine logic. An illustration of the instruction array for the sample ATM script file in Figure 42 is shown in Figure 43. As shown, the first instruction is to set variable 1, `conflict_radius`, to 30Nmi (55,560m).

6.4.5 Stack Machine Execution

A stack machine or a stack computer is any software or hardware mechanism which employs last-in first-out (LIFO) storage mechanisms. In this application a stack machine is a very basic logical compiler which acts on a set of instructions and uses a single stack as a register. Items are pushed onto and popped off of the stack in compliance with the active instruction in the instruction array. A basic calculator is good example of a stack machine. Figure 44 illustrates the primary data structures found in a stack machine. In the preceding steps the quantity 5 was pushed onto the stack, followed by the quantity 7. Next acting on the `Add` instruction they were both popped off the stack, added together, and the resultant quantity, 12, was pushed back onto the stack. In the current step, 12 is popped off the stack, and in the next

```
 1: SET 1; 0; 0; 0 55560;        9999;        9999
 2: GET 3; 44; 1; 0       9999;        9999;        9999;
 3: PUSHIV 2;
 4: PUSHEV 19;
 5: SWAP
 6: COMP
 7: PUSHEV 19;
 8: PUSHC 0.000000000;
 9: COMP
10: AND
11: JF L   28;
12: INCREMENT 2
13: PUSHEV 3;
14: PUSHC 304.800000000;
15: ADD
16: PUSHC 55560.000000000;
17: PUSHEV 5;
18: DIV
19: COMMAND 5; 1; 0; 0||      9999;        9999;        9999
20: PUSHEV 10;
21: PUSHC 304.800000000;
22: SUB
23: PUSHC 55560.000000000;
24: PUSHEV 12;
25: DIV
26: COMMAND 4; 8; 0; 0||      9999;        9999;        9999
27: JUMPIN     3;
28: STOP
```

Figure 43: Sample ATM Instruction Stack

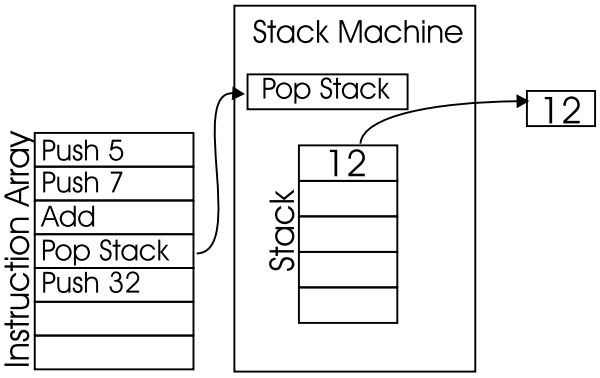


Figure 44: Illustration of a Simple Stack Machine

step the quantity 32 will be pushed onto the stack. The process continues as such until the end of the stack or a stop instruction is reached.

6.4.6 Enroute Controller

Section 6.4.1 established that the enroute controller module would be responsible for maintaining aircraft separation for all aircraft that have not commenced their approach into an airport. To add flexibility and realism to the simulation tool it is desirable to facilitate multiple enroute controllers and multiple airspace sectors, such that each controller module is only responsible for aircraft in its designated sector. In the present airspace structure, ATC centres and sectors are arbitrary constructs influenced by political boundaries, air route structures and real-time flow considerations [6, 64]. Often, over the course of a day, airspace sectors are combined or divided to maintain controller workload within an acceptable range. Since air route structures are both arbitrary and dynamic, it was decided that it was not necessary to implement them exactly as they exist in the current airspace environment. For the research purposes addressed by this simulator, only a representative structure is required. To simplify conflict detection algorithms, it was decided to base the control sectors on the quadtree structure which is used by the airspace module and was discussed in Chapter 5. A comparison between the current upper airspace structure and a similar quadtree based airspace structure can be seen in Figure 45. The advantage of using the quadtree structure over other representations is the improved efficiency in determining possible aircraft conflicts.

Each controller module constructs and maintains a quadtree representing the air traffic currently under its control. The information to populate this quadtree is passed at regular intervals from the individual aircraft to the corresponding controller via the appropriate message queue. Each controller module has a dedicated message queue, which simulates an ATC controller's dedicated frequency. The rate at which individual aircraft send state updates to the controller is aircraft dependent, which allows the simulation of aircraft with mixed equipage. This process of maintaining a "picture" of the airspace via updates in aircraft location is similar to the way secondary surveillance radar is used to update a radar controller's visual

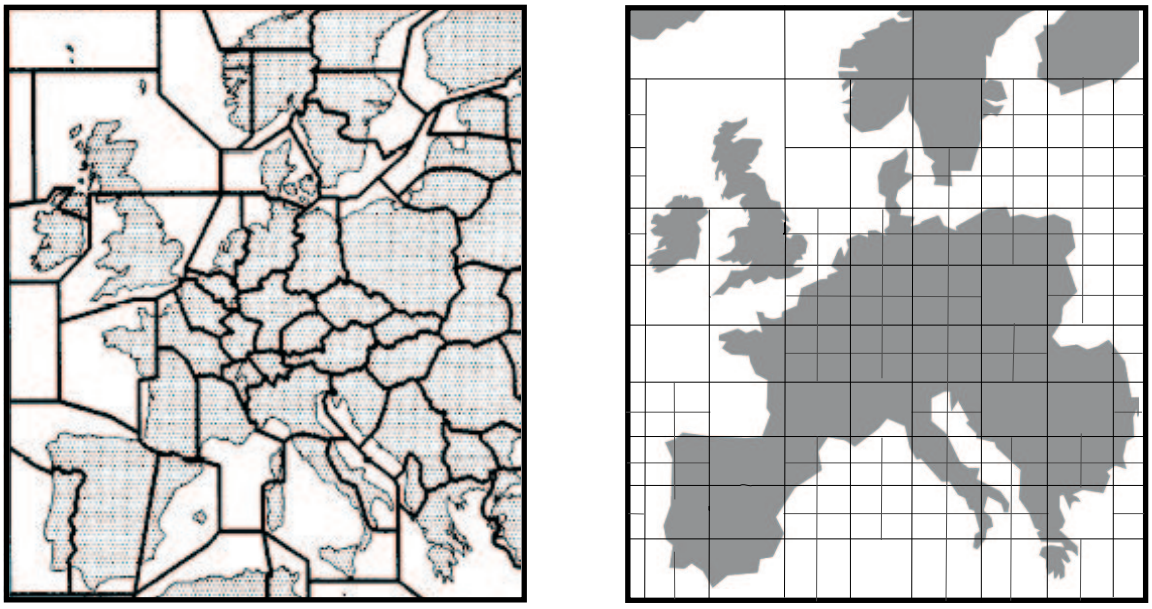


Figure 45: Upper-level Airspace Sectors and Level-7 Quadtree Comparison

display screen. To the enroute controller module, the quadtree is analogous to the mental image each ATC controller maintains of the aircraft under his control.

As the control logic is held in the instruction array created during initialisation, in addition to maintaining an accurate quadtree of the aircraft under its control, the enroute controller module must also execute the stack machine and thereby implement the ATM logic on the aircraft in the quadtree. The frequency with which the controller module executes the stack machine is set upon initialisation, and may differ between controller modules. To more closely approximate actual ATC environments if no new aircraft updates have been received during the intervening time period, then the stack machine will not be executed. During the stack machine execution command messages are created and sent to the pilot module. After each command has been sent, the controller module waits for an acknowledgement from the pilot module. Between stack machine execution calls, the controller module receives and acts upon all of the messages which arrive in order to maintain the quadtree representation of the air traffic within the controller module's sector.

6.4.7 Terminal Manoeuvering Area (TMA) Controller

Section 6.4.1 established that the TMA controller module is responsible for maintaining aircraft separation for all aircraft that have begun their approach into an airport. As each TMA controller is responsible for only one airport, each airport simulated is required to have a designated TMA controller. Although both controller modules have the same function, the methods that they use to maintain separation are very different because TMA conflicts must be resolved so that the aircraft are successfully merged while maintaining separation, not just successfully separated. The separation methods used by TMA controllers include, speed restrictions, lateral deviations and holding patters where aircraft complete a four minute circuit at a specific location.

The data structures needed to maintain separation for terminal manoeuvering areas are consequently different from the single quadtree used by an enroute controller module. However, similar to the enroute controller module, each TMA controller module maintains its own set of location arrays, through regular aircraft position updates from the air traffic module. The data structures used by the TMA controller module is illustrated in Figure 46. All aircraft under the TMA controller module's control are associated with either the initial approach array, the final approach array, or a stack array. Stack arrays represent individual holding patterns. The number of stack arrays is flexible, and is set at initialisation of the TMA controller. These location arrays are mutually exclusive, i.e. an aircraft cannot exist in more than one array at any time. In addition to these arrays, a comparison array is used to merge traffic, by using a copy of the aircraft position and estimated time of arrival at a specified position. Consequently, aircraft in the comparison array are also associated with another location array.

On commencing its initial approach phase of flight, an aircraft will be transferred from an enroute controller module to the appropriate TMA controller module, and placed into the initial approach array. The aircraft will remain in the initial

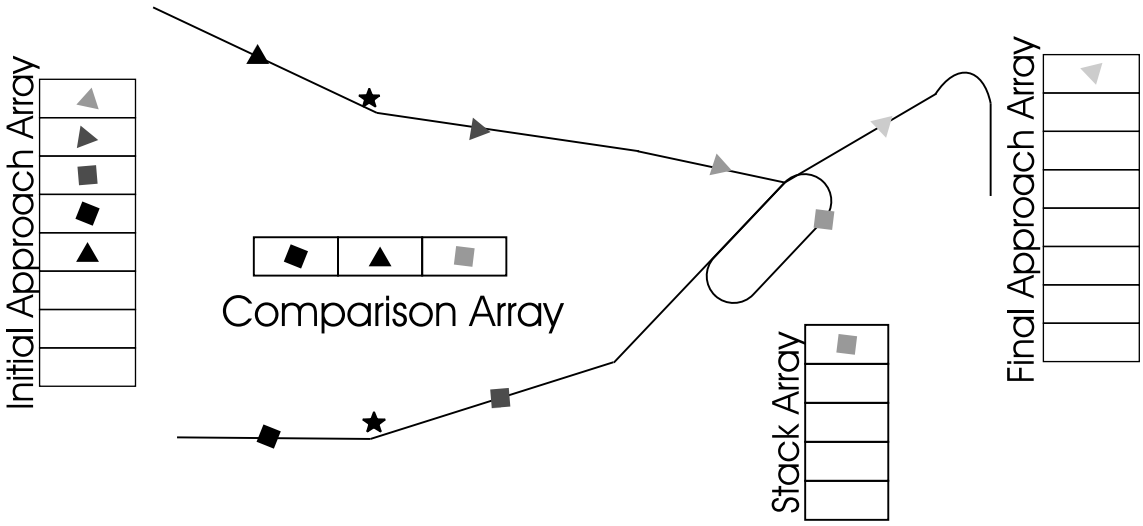


Figure 46: TMA Control Module Airspace Monitoring Data Structures

approach array until it is either transferred to a stack array or the final approach array. The final approach array is used to hold the fully merged aircraft stream. The stack arrays are used to hold aircraft in a particular holding stack. The stack array index corresponds directly to the stack altitude levels, and consequently only one aircraft is assigned to any stack level. In Figure 46 five aircraft are in the initial approach array, one aircraft is in the stack array and one aircraft is in the final approach array.

In this illustration, there are two arrival streams which must be merged. The comparison array is used to compare the aircraft in each stream which is closest to the star. Using the comparison array, the aircraft in each stream will be issued with instructions to continue, to reduce speed, to hold in the stack or some combination of these actions. Aircraft downstream of the star will have already been instructed. In general the specific logic implemented by the comparison array is dictated by the external script file in the same way that the control logic is implemented in the enroute controller module. Consequently, the vocabulary used to create the logic implemented in TMA controller modules differs slightly from the vocabulary used for enroute controller modules. Further clarification of this vocabulary is provided

in Appendix A.

6.4.8 Conflict Log

As the primary function of both controller modules is the separation of aircraft, it is desirable to record all of the potential conflicts detected by the controller using the control logic provided in the script files. Accordingly, whenever the controller detects a conflict, it is recorded in an external text file, known as the conflict log. The conflict log records the generation number, position and speed of the aircraft involved, the time of the incident, the stack instruction number and the controller identifier. The information is designed to allow a user to postprocess a simulation run and determine what conflicts were detected and by which instruction out of the instruction array compiled by the external script interpreter. To analyse a particular conflict, the time index provides a thorough examination of the full output data. The data allows for the evaluation of different airspace structures and the corresponding control logic.

6.5 Pilot Module

The function of the pilot module is to interpret command messages sent from the various controller modules and to execute the instructions. The pilot module checks for new commands at a set frequency, and if commands are present, they are responded to in the order in which they arrive. Depending upon the command, either the flight plan, the way point list or both will need to be altered. Once the alterations have been made, the pilot module will send a WILCO message acknowledging the receipt to the appropriate controller.

6.6 Process Harmonisation

As all simulation scenarios will involve at least one controller module, it is important to understand how the airspace module interacts with the controller modules to ensure that they are all running in the same time frame, i.e. that the controller modules are not sending commands for situations that happened in the past where the solutions are no longer relevant. It is also necessary to ensure that the simulation does not enter a deadlock situation, in which each process is waiting for information from another process. An example of a deadlock situation is where the airspace module is waiting on a message from an enroute controller module (meaning it is suspended), while the controller module on which it is waiting, is in turn waiting on a message from the airspace module (meaning it is also suspended). In this situation both processes are suspended with no means of becoming reactivated; they are considered hung.

The simulation tool employs two methods of processes harmonisation. The first is the provision of universal time which is generated by the airspace module and is kept in a shared memory space that is accessible by all of the processes. The controller modules use the universal time to trigger the execution of the stack machine. This ensures that, provided the stack machine execution takes less time than the execution frequency, the controllers will be regularly monitoring the airspace and generating commands as necessary. Figure 47 illustrates the dual time lines for the controller and airspace modules. The second process harmonisation tool is the use of process suspension by strategic message queue blocking, i.e. using the message queue receive function in blocking mode which commands the process to suspend until a message of the correct type is placed in the message queue. An example where the second process harmonisation method is implemented occurs after a controller issues a command; a blocking message receive function is invoked and the controller process is suspended until the pilot executes the changes and issues a WILCO message. Suspending the controller module ensures that the controller

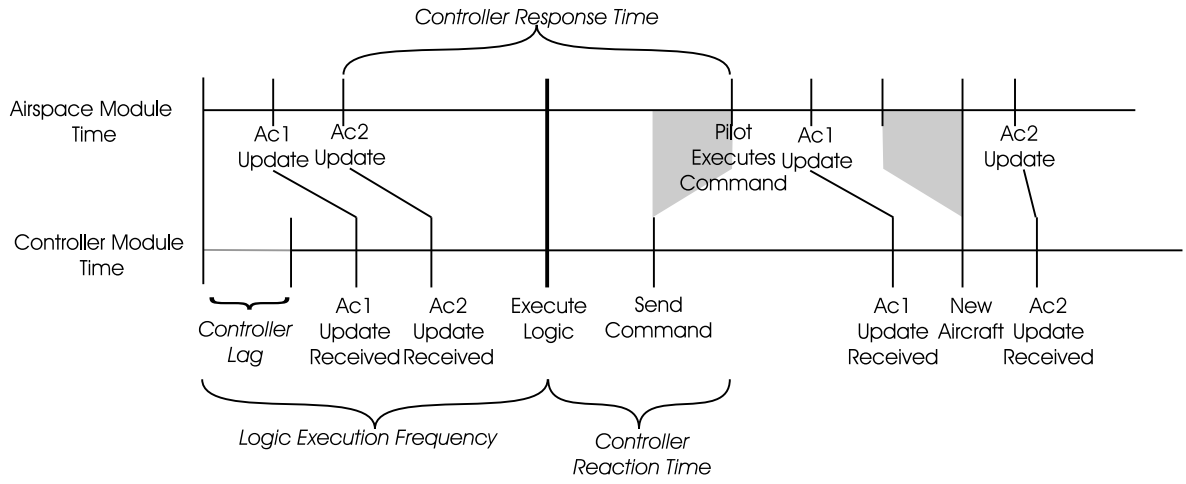


Figure 47: Airspace and ATC Module Time Line Illustration

module does not advance ahead of the airspace module. Earlier in this Chapter, Table 11 shows which types of messages are downloaded in blocking mode and which process they block.

6.7 Performance Characteristics

In order to meet the requirements set out in section 6.1 the control module must respond to all possible conflicts within a reasonable period of time. The response time, which can also be characterised as a time lag between the airspace module and the controller modules, is bounded by a combination of the two process harmonisation methods described previously. The best case, and an unrealistic scenario, would be zero time lag where the airspace module waited for each controller at every time step and vice versa. The controller module lag is bounded by the rate at which new aircraft are created and removed because, after each new aircraft is created, the airspace module waits for a WILCO signal from the specific controller module, effectively synchronising the two processes.

For simulations involving very few aircraft, the time between the creation and deletion of an aircraft may be of the order of several hundred seconds. However, the only way for controller lag to accumulate beyond the lag introduced by the message queue is for the execution of the stack machine and the processing of the

aircraft updates to take longer than time between execution calls, i.e. if stack machine execution frequency is every 30s of simulation time and the controller module takes 35s of simulation time to execute the stack machine then the controller will accumulate 5s of lag for every 30s of simulated time, until an aircraft is created or removed where the airspace module will wait for the controller to catch up. This is illustrated in Figure 47, where the time line of two aircraft, Ac1 and Ac2, is traced through two update cycles. Controller lag is present from the beginning eventually increasing the controller reaction and response time. Figure 47 illustrates how the controller lag is propagated until a new aircraft is generated, where it is reset to the lag introduced by the message queues. The time line begins just after the logic held in the instruction array has been executed by the stack machine. Thus the distance between the beginning of the time line and the next call to execute the stack machine represents the stack machine, or logic, execution frequency.

As the stack machine execution is based on universal time, which is, in turn, set by the airspace module, the actual time for a controller to respond to a possible conflict is bounded by the frequency of airspace module update message frequency in conjunction with the stack machine execution frequency plus any controller module lag. To ensure that the controller response times remain within an acceptable range, the command message is time stamped with the time of the most recent aircraft update message, which enables the controller lag to be calculated. If the response time becomes too long, then simulation parameters such as aircraft update frequency, or stack machine execution frequency can be adjusted accordingly. Airspace module lag is not possible because all controller module commands are issued in response to airspace module update messages.

6.8 Simple ATM Controller Example

To demonstrate how the control module works, the sample ATM script listed in Figure 42 has been implemented in a basic conflict scenario to illustrate the responsiveness of the controller, the effect on the involved aircraft, and the conflict log. In

the example two MD-80 aircraft were flown at the same altitude on an intersecting course. The simple ATM script logic was designed to detect potential aircraft conflicts at a range of 30Nmi, and to command one aircraft to climb 1,000ft and the other aircraft to descend 1,000ft to maintain separation. In this scenario, the aircraft positions were downloaded to the controller every 30 time steps, and command messages were uploaded from the controller every 5 time steps. The simulation time step is equal to one second. Additionally, since this was a simple demonstration, the frequency of aircraft generation was set to 500±10s. The controller invoked the stack machine every 30s.

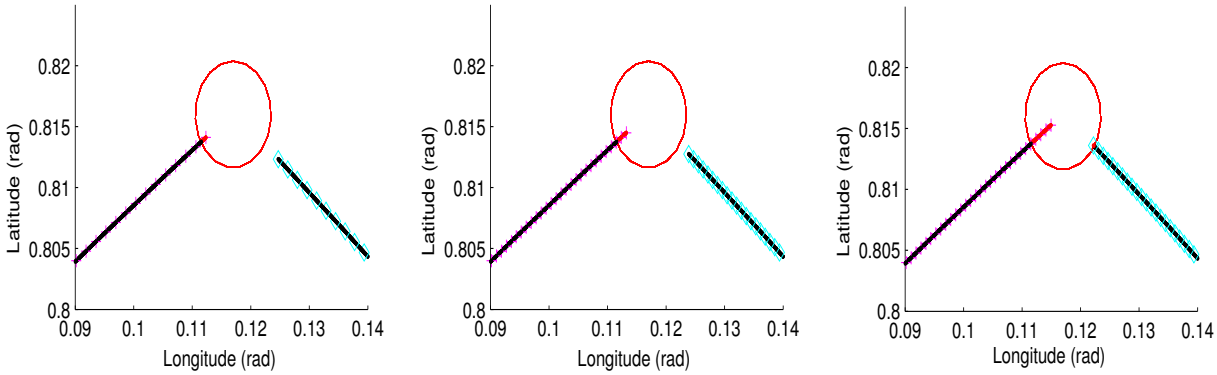
Figures 49 and 50 illustrate the conflict detection and its subsequent resolution. The 30Nmi boundary around the collision point is represented by the oval. Figure 49, (a) illustrates the position of both aircraft 2161 seconds into the simulation. At this point Aircraft 1, *Northwest bound*, downloads its position to the controller. Figure 49, (b) illustrates the position of both aircraft 2191 seconds into the simulation. At this point Aircraft 0, *Northeast bound*, transmits its position to the controller. After obtaining the position update from Aircraft 0, the stack machine is run, thus invoking the simple ATC logic. The controller detects a conflict between Aircraft 0 and Aircraft 1. The conflict is logged in the controller log, and commands are sent to Aircraft 0 and Aircraft 1.

Figure 48 shows an excerpt from the controller log, generated by the example. The first column holds the controller number, which corresponds to its quadtree index. The second column records the time of the last update which triggered the conflict alert. The third column is the number of the specific instruction in the instruction stack which generated the command. Columns 5-8 are the aircraft number, position in degrees, altitude in metres, and velocity in metres per second of the first aircraft, and Columns 9-13 are the aircraft number, position, and velocity of the second aircraft. The last column lists the separation distance between the two aircraft in metres. (1Nmi $\cong$ 1,800m)

It can be seen from this excerpt that, at 2191s into the simulation when the

Cnt	Time	SC	|AC1	lat	lon	alt	vel	|AC2	lat	lon	alt	vel	Dist
311	2191	19	1	46.54	7.14	3251	150	0	46.67	6.49	3251	150	51956.5
311	2191	26	1	46.54	7.14	3251	150	0	46.67	6.49	3251	150	51956.5

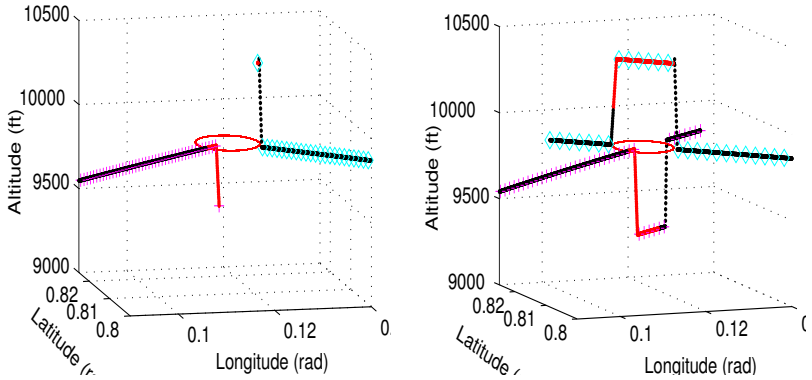
Figure 48: Sample Controller Log



(a) t = 2161 (b) t = 2191 (c) t = 2224

Figure 49: Conflict Detection Time Line

conflict was detected, the two aircraft were approximately 28Nmi apart. By examining the recorded aircraft positions for the two aircraft it can be determined that at 2224s the message was received by the pilot module and the altitude restrictions were implemented. At this point the aircraft were approximately 21.6Nmi apart. Figure 50 illustrates the flight path of the aircraft executing the solution. The response time for this scenario was 33s from receipt of the latest aircraft update until the execution of the controllers command. This scenario was run repeatedly for three different aircraft loadings and simulation durations. Table 12 shows the average controller response time for each scenario. As predicted the more frequent the aircraft generation (or the rate at which the aircraft are introduced into the simulation) the more responsive the control module becomes due to the increased synchronisation between the airspace module and the control module.



(a) t = 2250 (b) t = 3000

Figure 50: Conflict Solution Time Line

Table 12: Controller Module Response Time

Simulation	Aircraft Generation Frequency (s)		
Period (s)	125	250	500
5,000	19.5	32.0	49.5
10,000	21.9	38.1	41.6
15,000	18.4	36.7	50.6

6.9 Summary

In this chapter the requirements and implementation of the control module have been described. The control module meets its functional requirements by enabling the airspace simulator to implement conflict detection and avoidance strategies as well as positive control in terminal maneuvering areas using two distinct controller module types: enroute and TMA. The enroute and TMA controller modules meet the requirements of simulating the ATC controller behaviour by encoding the control logic strategy into a text file using the control vocabulary listed in Appendix A. Encoding the control logic in a text file using a high-level language allows individuals unfamiliar with the simulation tool to easily understand the control logic, and the large vocabulary allows considerable variability in the control logic used. The text file is then interpreted and stored as an array of stack machine instructions, which

allows the logic to be efficiently implemented via stack machine execution at regular intervals throughout the simulation.

As required, the controller modules are in constant communication with the airspace modules sending messages via dedicated message queues. Using multiple independent control modules that communicate via standard messages further enhances the flexibility and the realism of the control module. The commands issued by the controller module are implemented by a pilot module by altering the aircraft flight plans and thereby meeting the requirements of modelling the pilot behaviour.

The control modules must operate within a set response time of 10s - 50s. In this chapter a basic example of enroute control logic has been examined, with a response time of 34s. In addition it has been demonstrated that the control module response time can be improved by increasing the aircraft generation frequency, as this causes the the airspace module and control module time lines to synchronise more frequently. In Chapter 7 more complicated scenarios will be implemented, and the performance of the entire simulation tool including the control module will be evaluated further.

CHAPTER 7

Testing and Evaluation

Up to this point in the thesis the main modules have been introduced and individually evaluated. The entire simulation however, has yet to be evaluated as a whole. In order to determine if the simulation tool met the objectives established in Chapter 2, the whole simulator needs to be evaluated. In this chapter the simulator is evaluated using three different scenarios, each investigating different portions and phases of the overall simulation:

- ❖ North Atlantic Crossing
- ❖ Landing at Gatwick
- ❖ Simple Europe

The first scenario will evaluate the basic simulator's performance and the impact of the control module on runtime performance. The second scenario will evaluate the TMA control module's ability to merge arrival streams of aircraft. The final scenario will evaluate the control module's ability to separate aircraft and its impact on the overall simulation runtime performance. All together these scenarios cover the range of capabilities required by the simulation set forth in Chapter 2. During these scenarios different airspace and simulation metrics were varied to assesses their impact on the simulator's performance. The first two sections will introduce the metrics used in these evaluations. The third section will discuss the simulator's limitations. The fourth section will present the three evaluation scenarios, and the last section will summarise their findings.

7.1 Airspace Metrics

Airspace metrics are parameters which are inherent in the airspace structure and affect the simulation runtime performance. Airspace metrics include:

Airspace Density - The number of aircraft in a specific area. Airspace density greatly affects controller workload and is therefore used to determine when new airspace sectors should be created.

ATC Sectors - The number of enroute air traffic control sectors is directly related to the number of controllers needed to control the traffic. In practice, each sector has one to two controllers, controlling about 20 aircraft at any time [87].

Airports - The number of airports is directly related to airspace complexity, as each airport will have terminal maneuvering airspace and tower airspace.

Controller Behaviour - The complexity of the algorithm required to simulate the underlying logic used by controllers is directly related to the complexity of the airspace and the airspace density.

7.2 Simulation Metrics

Simulation metrics are parameters which either govern the implementation of individual modules in the simulation tool or translate airspace metrics into variables in the simulation tool. The first type of simulation metrics are designated implementation simulation metrics, and the second type are designated translation simulation metrics. Each translation simulation metric corresponds to an airspace metric listed in section 7.1.

In addition to the current airspace metrics listed above, there are inherent airspace metrics such as human reaction time and radar sweep frequency. These inherent airspace metrics are implemented in the simulation as an implementation simulation metric, e.g. frequency of controller stack machine execution. These inherent airspace metrics have not been included in the list of airspace metrics because

they are invariable aspects of the airspace system. In a high level simulation, it is not always necessary to match each inherent airspace metric with a simulation metric, instead it is enough to mimic their impact on the overall system. Inherent airspace metrics are implemented in the simulator through implementation metrics, such as the frequency of enroute controller module updates, which mimics the secondary surveillance radar system or ADS-B.

However, implementation simulation metrics do not necessarily correspond to inherent airspace metrics. Others are purely a function of the simulator's data structure, such as the frequency of quadtree updates. For example, the quadtree is a data structure used by the simulator to organise the simulated aircraft, and has no equivalent in the real airspace system.

7.2.1 Implementation Simulation Metrics

Simulation period is the amount of real time to be simulated.

Frequency of aircraft generation is the mean time between the introduction of a new aircraft into the simulation.

Number of quadtree levels is equivalent to the smallest possible sector size. As the number of quadtree levels increases, the sector resolution becomes finer and the size of the quadtree increases.

Frequency of quadtree updates is the simulated time between aircraft position updates in the quadtree structure.

Frequency of enroute controller module updates is the time between aircraft position updates issued to the enroute controller module. This parameter determines the resolution at which the enroute controller module is updated. In practice, an enroute radar has a refresh rate of 12s [7].

Frequency of TMA controller module updates is the time between aircraft position updates being issued to the TMA controller module. This parameter

governs the update rate of the TMA controller module. In practice, a TMA radar has a refresh rate of 6s [7].

Frequency of enroute controller stack machine execution corresponds to the rate at which the controller's logic is implemented. In practice controllers continually monitor their sector for potential conflicts.

Frequency of TMA controller stack machine execution corresponds to the rate at which the TMA controller's logic is implemented. In practice TMA controllers continually monitor their sector for possible loss of separation.

Frequency of data storage is the rate at which data is transferred from the simulation's active memory to the hard-drive .

Wind is whether or not the wind field model is included in the simulation.

Data resolution is the rate at which the simulation generates outputs, e.g. aircraft position, heading, which are subsequently recorded in an output file.

7.2.2 Translation Simulation Metrics

Aircraft Loading is the number of aircraft simulated concurrently. This metric corresponds to the airspace density and is calculated as the average length of all the flights simulated divided by the aircraft generation frequency.

Number of enroute control modules corresponds directly to the number of ATC sectors simulated.

Number of TMA control modules corresponds directly to the number of airports simulated.

Controller Logic Complexity is the number of instructions created in the controller instruction array by the ATC script interpreter. The more complex the logic the greater the number of instructions.

7.3 Simulation Limitations

As with all software, the simulation tool has limitations due to the data structures and algorithms contained in it. The simulation has been designed to provide up to 700MB of output to conform to the current data storage standards of a writable CD. Consequently, the combination of aircraft generation frequency, simulation period and resolution are constrained by this requirement. Additionally, the controller conflict detection algorithm is limited to latitudes below 65^0, as at latitudes greater than this the distance between lines of latitude are greatly exaggerated. Consequently flight plans which exceed this latitude limit will are not able to use any of the conflict detection or resolution algorithms. Further, several parameters which are not considered as simulation metrics, as they do not effect the simulation run time, are currently set at arbitrary levels which can be altered to accommodate additional situations. These parameters include the number of airports permissible in each sector, the number of stacks per sector, the number of aircraft per sector, the number of aircraft per quadtree region, etc.

7.4 Test Scenarios

7.4.1 Endurance, North Atlantic

Case Description

The purpose of this scenario is to demonstrate the capabilities of the airspace module alone while varying the frequency of data storage, the frequency of quadtree updates, and the level of aircraft loading. Additionally this scenario illustrates the effect of the addition of control modules and different control logic complexity levels on the overall simulation runtime.

Table 13 lists the metrics used in the test runs, and Figure 51 illustrates the flight plan followed by the aircraft. The specific input files, which include initial conditions and flight plans, are listed in Appendix B. In this scenario aircraft fly

Table 13: North Atlantic Crossing Metric List

Simulation Period	10 Hours (36,000s)
Flight Plans	1
Flight Path Crossings	0
Average Aircraft Loading	100, 150, 200
Control Modules	1,4
Enroute Control Logic	None, Simple
Number of Airports	1
Number of Hold Stacks	0
TMA Control Logic	None
Data Storage Frequency	300s, 450s
Quadtree Update Frequency	60s, 90s

from New York's JFK airport through Ebony fix, to the Dogal Fix off the coast of Ireland. The aircraft then commence their descent into Gatwick (EGKK) airport in the UK through a series of fixes specified by standard arrival route ASTRA 1B.

Testing Methodology

In this scenario the five simulation metrics varied were:

- ❖ the average aircraft loading
- ❖ the number of control modules used
- ❖ the control logic complexity employed
- ❖ the frequency of quadtree updates
- ❖ the frequency of data storage

They were varied as shown in Table 13. The resulting experimental design had 15 separate cases, which is fewer cases than a full factorial design. In a factorial experimental design, every level of each factor is combined with every level of each other factor. As this experiment included five independent variables, or factors, each with either 2 or 3 levels, a factorial design requires 24 separate cases to be run in order to isolate each variable. The cases which were eliminated were unrealistic in terms of aircraft separation criteria and control module limitations. For example

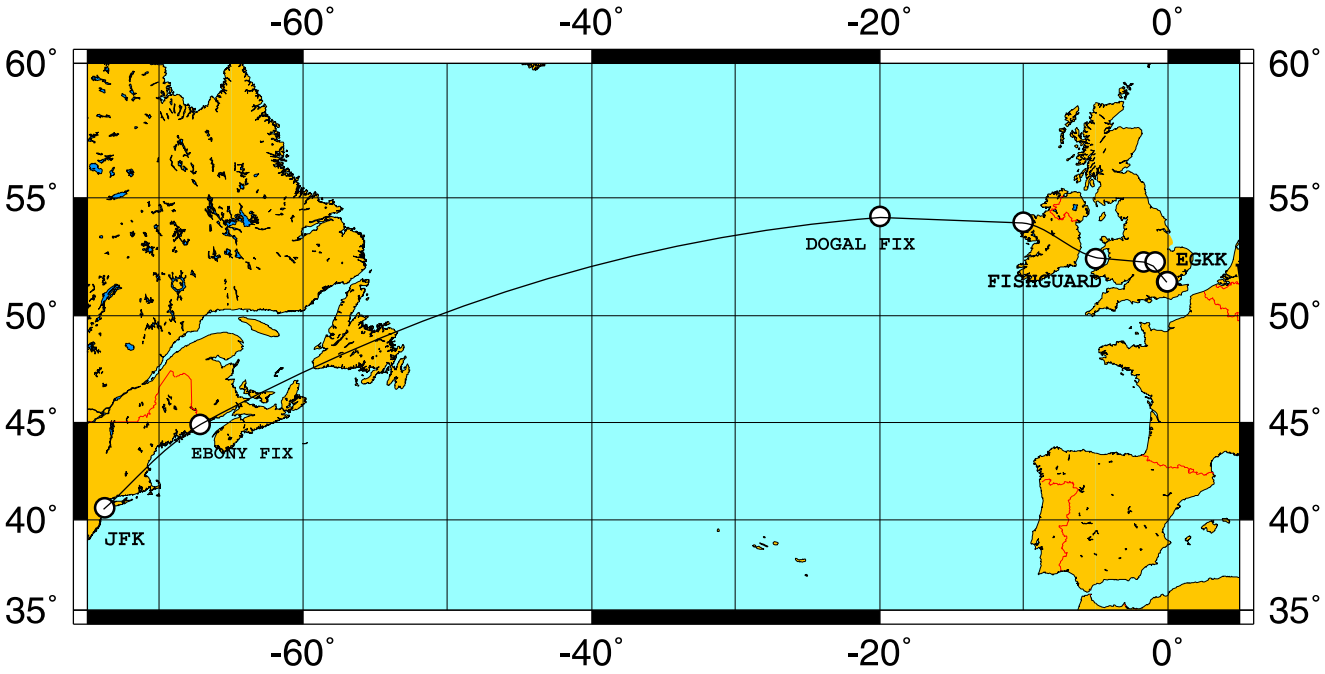


Figure 51: Representative North Atlantic Track

with an average aircraft loading of 200, aircraft were initialised within 125s of each other, which translates to a separation of only 15.5Nmi (assuming the aircraft have a cruising velocity of 230m/s). The separation criteria for North Atlantic crossings is 10Nmi instead of the normal 5Nmi, meaning that even slight deviations in velocity between aircraft would result in conflicts. In addition it was not realistic to use a single control module to control between 100 and 200 aircraft, as real controllers handle around 20 aircraft [87]. To observe the repeatability of the simulator, each case was run three times. This scenario is primarily concerned with the overall simulation time for the combination of the metrics used.

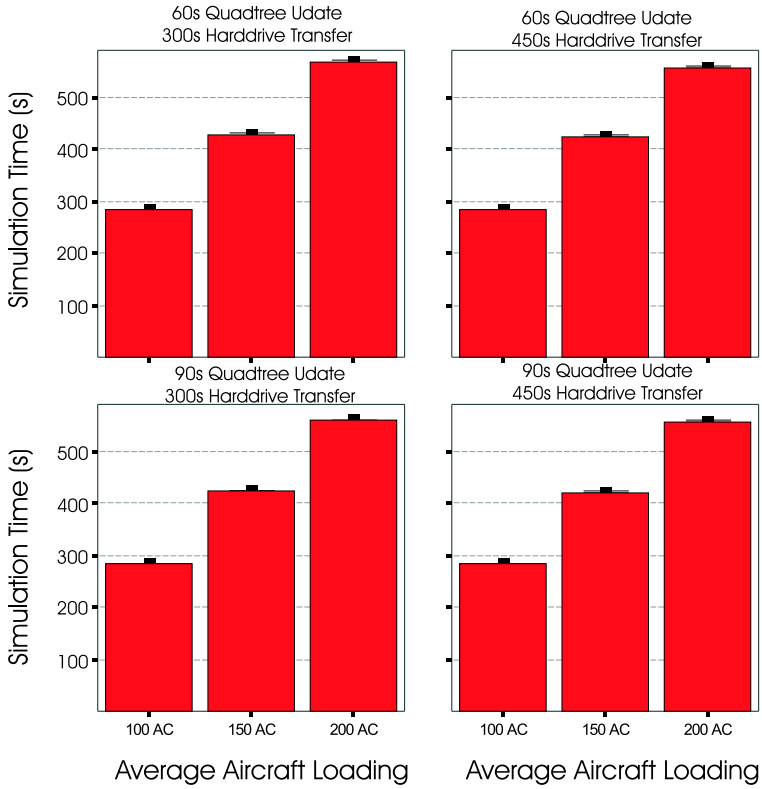


Figure 52: Impact of Aircraft Loading on Simulation Time

Results

The first objective of the North Atlantic Crossing scenario is to observe the overall simulation runtime for different average aircraft loadings and to evaluate the effects of data storage and quadtree update rates on the runtime. Figure 52 illustrates the results for the appropriate scenario cases. The bars represent the average from all three runs, with the error bars representing the standard deviation across all three runs. The quadtree update rate is varied by row, and the data storage rate is varied by column. It is clear that the affect of varying either of these metrics has little impact on the overall simulation time. It is also apparent that the primary metric that affects simulation time is the average aircraft loading. This is to be

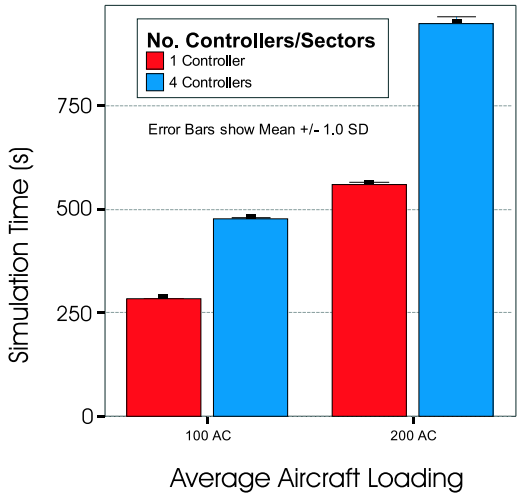


Figure 53: Impact of Aircraft Loading and Controllers on Simulation Time

expected because increasing the number of aircraft being concurrently simulated increases the number of calculations required per simulation time step. As algorithms were developed to avoid exponential growth characteristics (by substituting loops with recursive algorithms wherever possible), it is reassuring to observe that the relationship between increasing aircraft loading levels and time appears to be linear.

The second objective of the North Atlantic Crossing scenario is to observe the impact of the average aircraft loading, the number of control modules and the control module logic complexity on the overall simulation time. Figure 53 compares the average aircraft loading and the number of controllers, and Figure 54 compares the number of controllers and the control logic complexity. Again the bars represent the average across all three runs and the error bars represent the standard deviation.

It is apparent from Figure 53 that the addition of controllers has a significant impact on the overall simulation time. This is to be expected owing to the increased simulation overhead associated with each control module. Each module is a separate Linux process and maintains its own data structures to monitor the aircraft under its control.

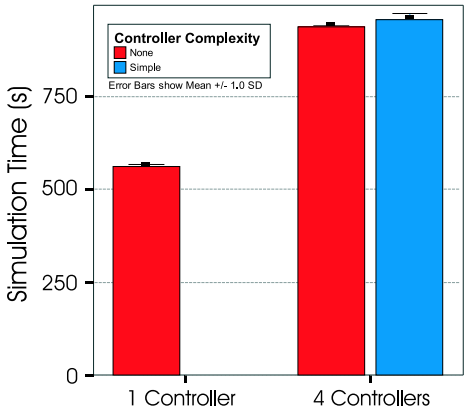


Figure 54: Impact of Controllers on Simulation Time

The computational overhead impact is further confirmed by Figure 54, which illustrates the variation in simulation runtime for different levels of control module logic complexity. There is very little difference in the overall simulation time between the simulation cases, which employ four controllers regardless of the control module logic. This is expected as the instruction array for the stack machine is pre-compiled. Additionally, the differences in complexity, which result in different length instruction arrays, have a negligible effect on the stack machine execution time when compared to the simulation overhead associated with running separate processes. In this scenario the simulation was running between 64 and 120 times faster than realtime.

7.4.2 Landing at Gatwick

Case Description

The purpose of this scenario is to demonstrate the capabilities of the TMA module logic while varying:

- the frequency of the TMA control module updates
- the level of aircraft loading
- the wind implementation

Table 14 lists the metrics used in these test runs, and Figure 55 illustrates the flight plans followed by the aircraft. The specific input files, which include initial conditions and flight plans, are listed in Appendix B. In this scenario aircraft fly from either the TRIPO or GURLU fixes to descend into Gatwick (EGKK) airport through a series of fixes based on standard arrival routes TIMBA 2B (TRIPO) or TIMBA 1H (GURLU). The TMA controller logic was either blank or was designed to merge the air traffic from the two streams at the TIMBA fix. If the lateral separation between the aircraft was not adequate, i.e. if left on their current speed and heading the aircraft would violate the standard wake vortex separation criteria, then the aircraft would be directed to a holding patter located at the TIMBA fix until an opening in the arrival stream would permit the aircraft to leave the stack.

Testing Methodology

In this scenario the four simulation metrics varied were:

- the average aircraft loading
- the control logic complexity
- the frequency of TMA control module updates
- the implementation of the wind field model

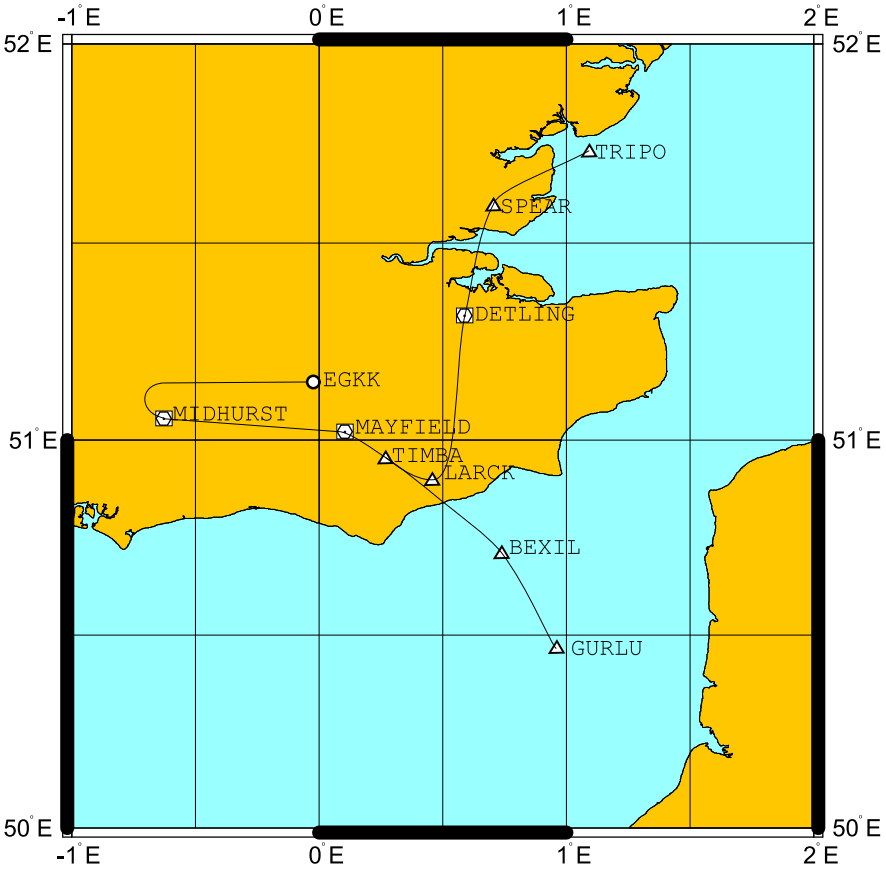


Figure 55: Standard Arrival Routes TIMBA 2B and TIMBA 1H

Table 14: Landing at Gatwick Metric List

Simulation Period	1.4 Hours (5,000s)
Flight Plans	2
Flight Path Crossings	1
Average Aircraft Loading	20, 25
Control Modules	2
Enroute Control Logic	None
Number of Airports	1
Number of Hold Stacks	1
TMA Control Logic	Complex
Frequency of Control Module Updates	15s, 30s

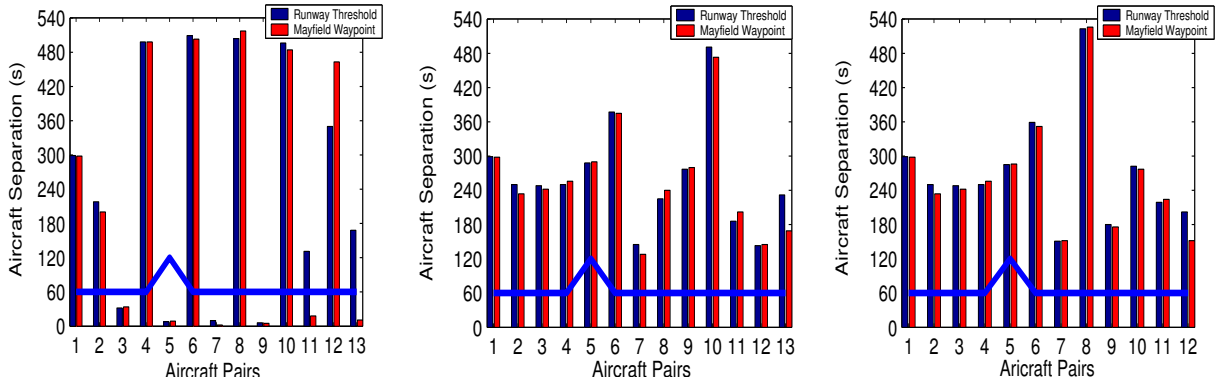
The first three of these metrics were employed as shown in Table 14. The wind field model was either implemented or not implemented. The TMA controller module stack machine execution frequency was set to 15s for each case. The resulting experimental design had 16 separate cases. This equates to a full factorial design.

The purpose of the TMA control module is to maintain the safe separation of aircraft while directing them to land. To test the effectiveness of the TMA control module to fulfil its purpose, the separation between aircraft pairs at two different thresholds was measured. For example if Aircraft 1 is flying the Timba 2B route and Aircraft 2 is flying the Timba 1H route, the separation between this aircraft pair will be measured as the time between Aircraft 1 passing the threshold and Aircraft 2 passing the threshold. The thresholds are located at the TIMBA fix and the runway threshold. Each case was run once because the controller's commands were highly variable between runs. This would lead to inconsistency between runs for the same case. The variability can be attributed to differences in the allocation of processing time by the operating system.

Results

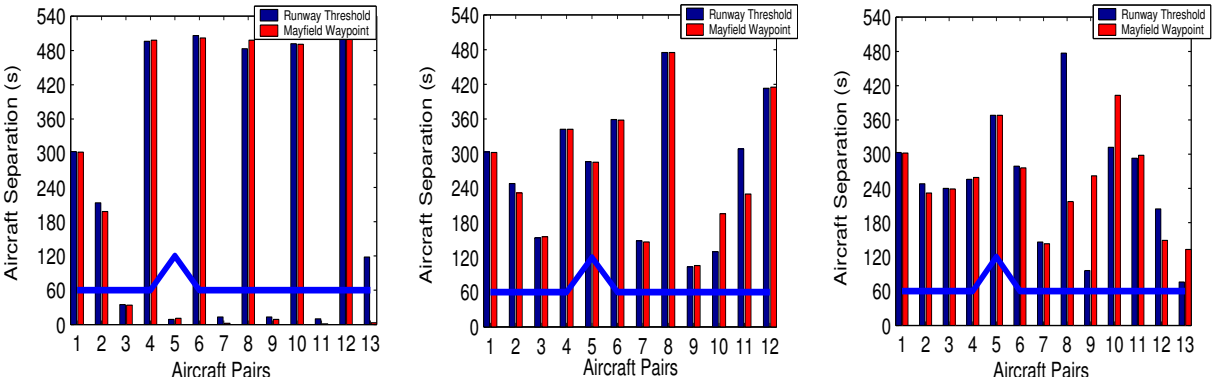
The primary objective of the Gatwick Landing scenario is to determine the effectiveness of the TMA controller, specifically at maintaining separation times to ensure the minimum separation distances as to avoid wake vortices. Figures 56-59 illustrate the separation between aircraft pairs for the two thresholds and the minimum separation times required at each point. As the simulation only included one aircraft with a wake vortex classification of heavy, there is only one point at which the minimum separation is 120s instead of 60s. The graph on the left of each figure is the control run, where no control logic was employed.

It is clear that the aircraft loading played a significant role in determining the natural separation between aircraft. This is due to the stochastic nature of the aircraft generation. The higher the aircraft loading the more frequently new aircraft are introduced into the simulation. Thus, while the control case with an average aircraft loading of 20, clearly violated the separation standards for more than half



(a) No Controller (b) 30s Controller Update (c) 15s Controller Update

Figure 56: Aircraft Separation Times in Wind for Aircraft Loading of 20

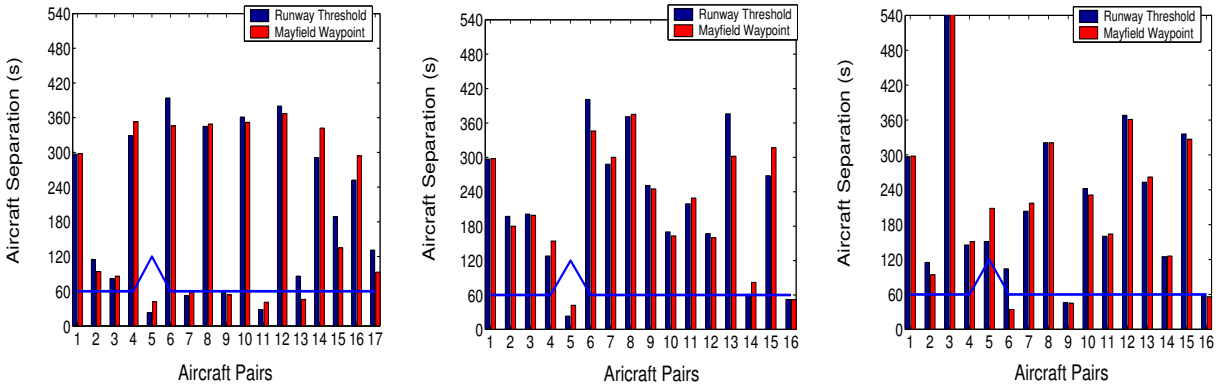


(a) No Controller (b) 30s Controller Update (c) 15s Controller Update

Figure 57: Aircraft Separation Times without Wind for Aircraft Loading of 20

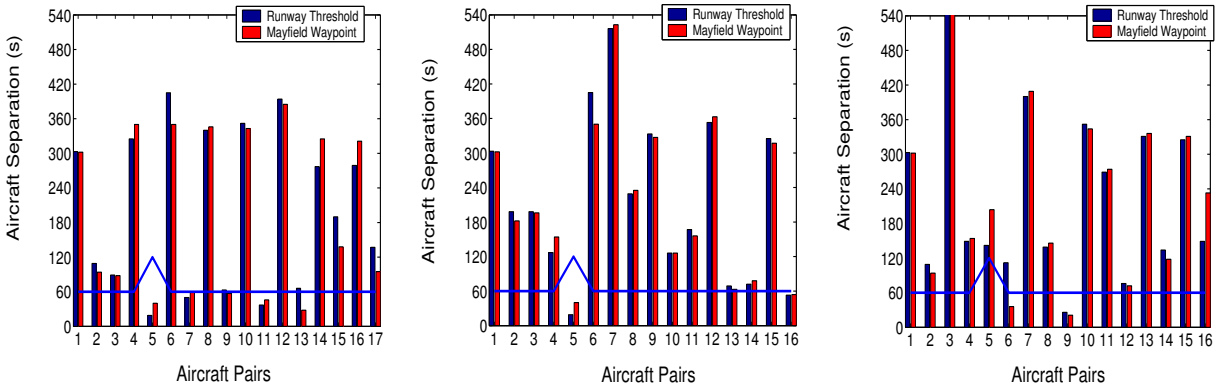
of the aircraft pairs, the control case with an average aircraft loading of 25 only violated the minimum separation criteria for a few aircraft pairs. It is important to compare the controller effectiveness in these control cases, and to note that the AC loading had a significant effect on the natural aircraft separation.

From observation of Figures 56-57, it is clear that the TMA control module is successful in separating all of the aircraft pairs. However, in Figures 58- 59 which illustrate the separation between aircraft pairs with a higher aircraft loading, it is clear that the TMA control module is less successful. It succeeds in separating most but not all of the aircraft pairs. It is important to note that although the



(a) No Controller (b) 30s Controller Update (c) 15s Controller Update

Figure 58: Aircraft Separation Times in Wind for Aircraft Loading of 25



(a) No Controller (b) 30s Controller Update (c) 15s Controller Update

Figure 59: Aircraft Separation Times without Wind for Aircraft Loading of 25

aircraft loading was higher for the cases shown in Figures 58- 59, the severity of the conflicts, i.e. the number of times the aircraft violated the minimum lateral separation criteria, is less.

Thus it would appear that the control module is more capable of separating aircraft flows which produce extreme conflicts, than it is separating aircraft flows which produce marginal conflicts. This is most likely due to limitations in the control logic script, please see Appendix B on page 184. This script includes a number of different intervention thresholds, which are defined as the minimum predicted lateral separation calculated by the controller. If the controller predicts that the lateral

separation between two aircraft is less than the specified threshold, it will take action to separate the aircraft. Further experimentation with these implementation thresholds may have improved the control module performance. Additionally more complex script capable of implementing intervention thresholds for a greater variety of scenarios might have improved the success of the TMA controller.

What is important to note is that where the control module is implemented the average separation of the aircraft pairs is increased and the variation in aircraft separation is reduced for the case with an aircraft loading of 20. Figure 60 illustrates the overall average of the aircraft separation. The TMA update frequency is varied by row. The controller complexity is varied by column, such that the first column represents the control cases and the second column represents the active controller cases. It appears that the controller affects the average separation time across all of

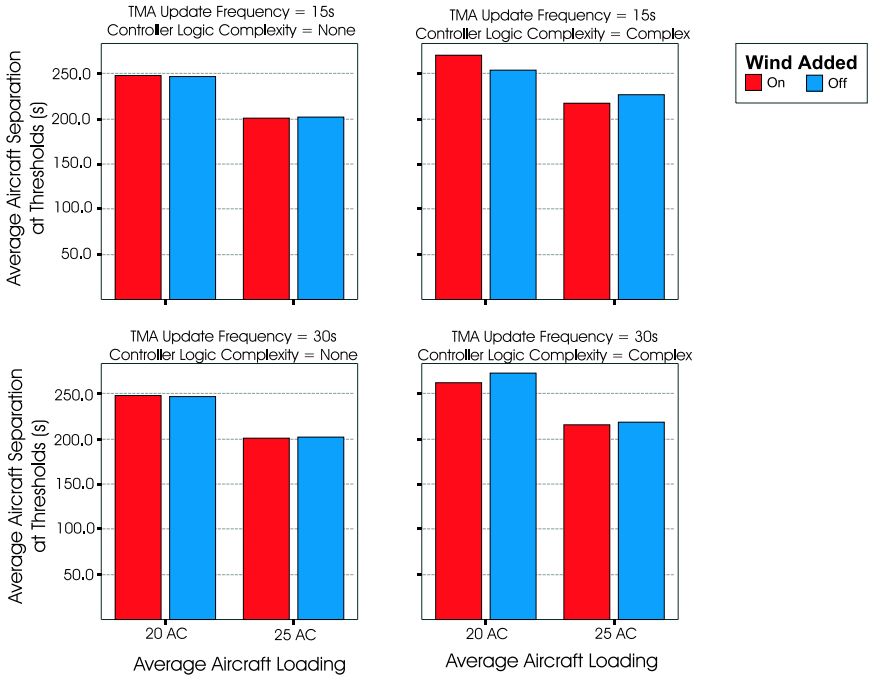


Figure 60: Comparison of Average Aircraft Separation for Varying Aircraft Loadings and TMA Update Frequency

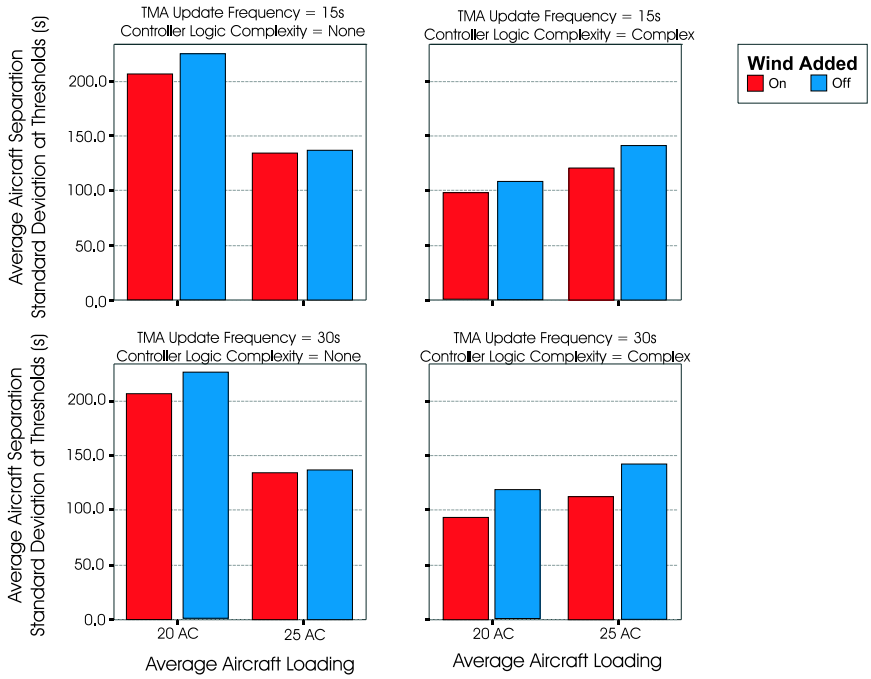


Figure 61: Comparison of Average Aircraft Separation Standard Deviation across Aircraft Pairs for Varying Aircraft Loadings and TMA Update Frequency

the aircraft pairs for both aircraft loading cases.

Figure 61 illustrates the average standard deviation for all of the cases. Similar to Figure 60 the TMA update frequency is varied by row, and the controller complexity is varied by column. This figure shows that the TMA control module is able to moderate the air traffic flow better for most of the cases.

From these two figures, it also appears that the TMA update frequency does not have any significant effect on the overall effectiveness of the TMA controller. While this is unexpected, it is possible that the standard TMA stack machine execution time of 30s negated any significant improvements provided by the more accurate aircraft positions. However the wind, did have a noticeable effect on the air traffic flow, which is expected as the addition of wind can significantly affect aircraft trajectories during altitude and heading changes [9].

7.4.3 Simple Europe

Case Description

The purpose of the Simple Europe scenario is to demonstrate the simulation of multiple flight plans, with multiple enroute control modules and multiple airports. Additionally the impact of wind and the effectiveness of the enroute control modules control logic on the overall simulation time and aircraft separation are investigated.

Table 15 lists the metrics used in the test runs, and Figure 62 illustrates the flight plans followed by the aircraft. The specific input files, which include initial conditions and flight plans are listed in Appendix B. In this scenario, aircraft flew from one of nine starting cities until they reached a cruising altitude of 37,000ft. Although some of these aircraft do not usually cruise at 37,000ft, a single level was chosen to maximise the number of aircraft in conflict with one another. One exception to this rule is the flight plan from Frankfurt to Amsterdam. For this flight there was insufficient time to reach the cruising altitude of 37,000ft, instead the aircraft climbed to 33,000ft Upon reaching their destination, the aircraft then descend as though landing. The flight plans followed for this scenario are not real, but are representative. For this scenario the TMA control logic was left blank, as the purpose was to investigate the effectiveness of the enroute controllers. The enroute control logic implemented a vertical and speed control strategy which is included in

Table 15: Simple Core Europe Metric List

Simulation Period	6 Hours (21,600s)
Flight Plans	9
Flight Path Crossings	8
Average Aircraft Loading	155, 233, 272, 320
Control Modules	8
Enroute Control Logic	None, Nominal
Number of Airports	9
Number of Hold Stacks	0
TMA Control Logic	None
Frequency of Controller Module Updates	15s, 30s
Frequency of Controller Stack Machine Execution	15s, 30s

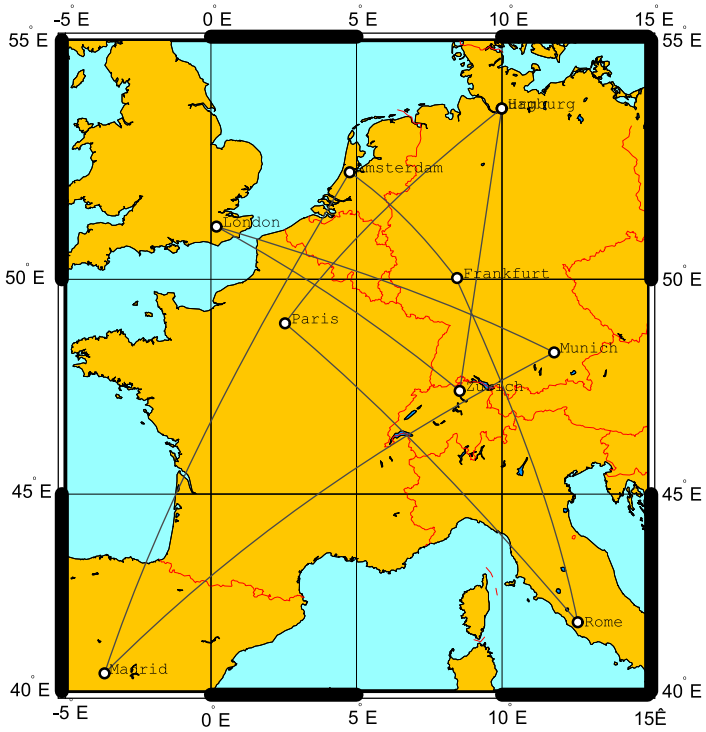


Figure 62: Representative European Flight Plans

Appendix B.

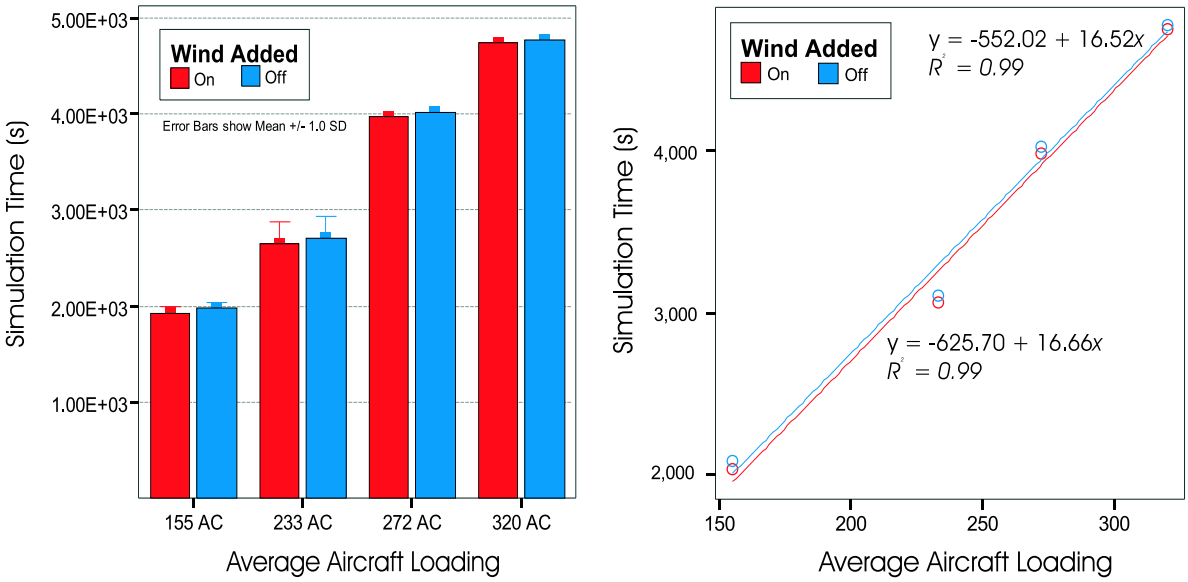
Testing Methodology

In this scenario the five simulation metrics varied were:

- ❖ the average aircraft loading
- ❖ the control logic complexity
- ❖ the frequency of enroute control module updates
- ❖ the frequency of the enroute control module stack machine execution
- ❖ the implementation of wind

the first four of which were employed as shown in Table 14. The wind model was either implemented or not implemented.

The resulting experimental design had 24 separate cases, which is fewer cases than a factorial design. The cases which were eliminated were unrealistic in terms of aircraft separation criteria. For example, with an average aircraft loading of 233,



(a) (b)

Figure 63: Simulation Time Relationships

aircraft were initialised from the same airport every 180s, which is almost at the minimum lateral separation of 5Nmi. The simulation runtime and the separation between aircraft pairs conflict resolution were generated by the simulator to determine the effectiveness of the enroute control module. Due to the time required to run and perform analysis, each case was only run once. However for completeness selected cases were run several times and similar results were obtained each time.

Results

The primary objective of the Simple Core Europe scenario was to again show that the simulation could handle over 300 aircraft simultaneously and to establish the relationship between the average aircraft loading and the simulation time. Figure 63 illustrates this linear relationship. Figure 63(a) shows the mean simulation time per average aircraft loading across all of the cases, and Figure 63(b) illustrates the linear relationship between the average aircraft loading and the simulation time. Figure 63 corresponds well with the results obtained from the North Atlantic Crossing scenario, where a linear relationship to average aircraft loading was also established.

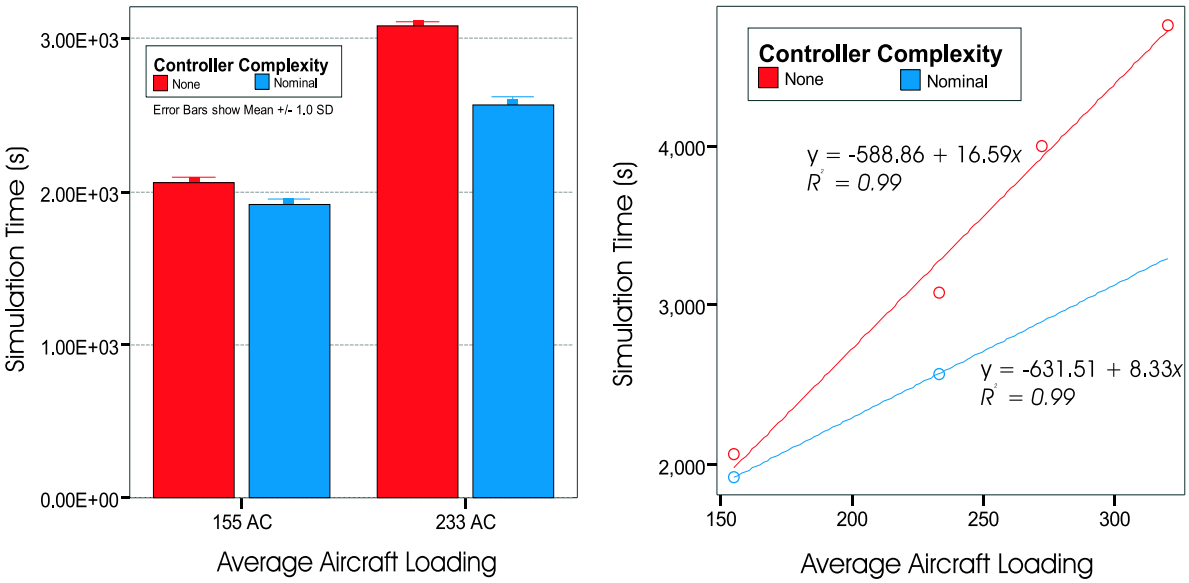


Figure 64: Comparison of Simulation Time for Different Controller Complexity Levels

The second objective of this scenario was to determine the impact of the controller logic on the simulation time. It was thought that the addition of the control module would marginally increase the simulation time. It was discovered that the relationship between average aircraft loading and simulation time is linear. However it was also discovered that the relationship is linear within controller complexity groupings, as shown in Figure 64. This raises the question, why would the simulation time be so much lower with the control module logic on, than without it?

The answer can be seen by examining the actual number of aircraft concurrently simulated instead of the average aircraft loading, because the average aircraft loading metric assumes that the average aircraft flight time is constant. In this case, the control logic is commanding aircraft to solve conflicts by either climbing or descending. However, aircraft are only commanded to climb if they are below the highest altitude allowable by the BADA performance tables. Figure 65 illustrates these performance tables, where Boeing 737-300 and 747-200 aircraft are currently

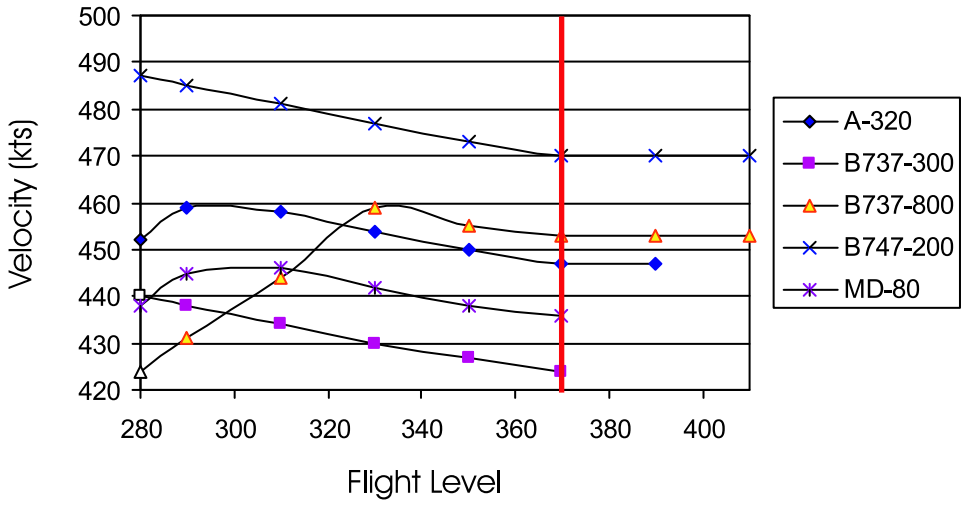


Figure 65: BADA 3.3 Performance Table Velocity v. Flight Level

at the highest allowable altitude. As the BADA performance data is linked to altitude, it can be shown that for the aircraft used in this simulation reductions in Flight Level lead to increases in velocity whereas increases in Flight Level result in no change in velocity. Thus, the aircraft under control module guidance complete their journeys faster than aircraft without control. This reduces the average flight time and the corresponding aircraft loading. Figures 66 and 67 illustrate this point by examining the number of aircraft simulated concurrently at the end of the simulation run, which is effectively a steady state point in the simulation.

Figure 66 shows the number of aircraft present at the end of the simulation run. It is important to notice that the error bars represent the standard deviation for all of the cases simulated. For average aircraft loadings of 155 and 233, the standard deviation across all of the cases is high. Figure 67 examines the number of aircraft present at the end of simulation, which reveals that the number of aircraft present in the cases with active control logic is less than the cases without control logic. In fact, by comparing the shape of the bars in Figures 64(a) and 67 it is clear that they are correlated. By normalising the simulation time by the number of aircraft present at the end of the simulation, it can be seen that the simulation time per aircraft is fairly constant across all of the average aircraft loadings and wind implementation as seen in Figure 68. The simulation time per aircraft is also fairly

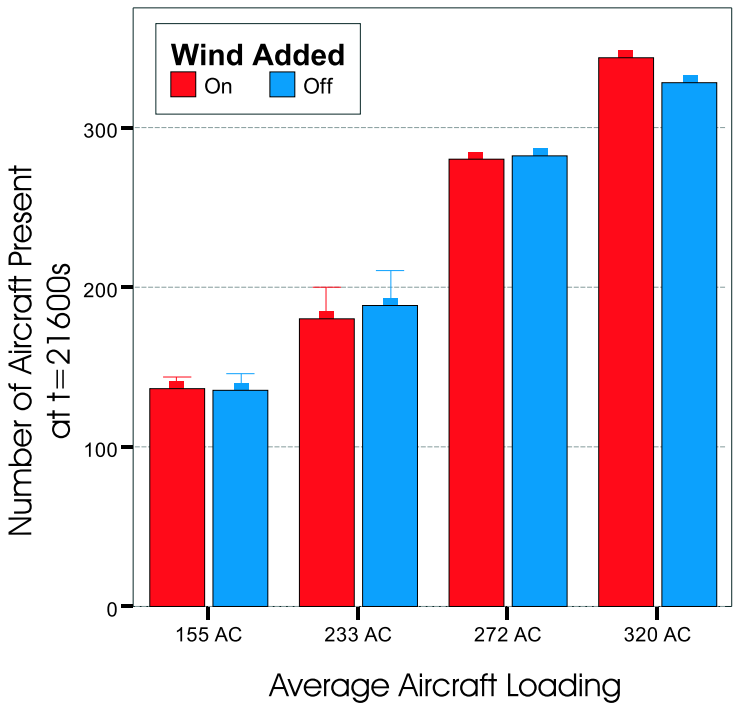


Figure 66: Number of Aircraft Present in Simulation at t=21600s

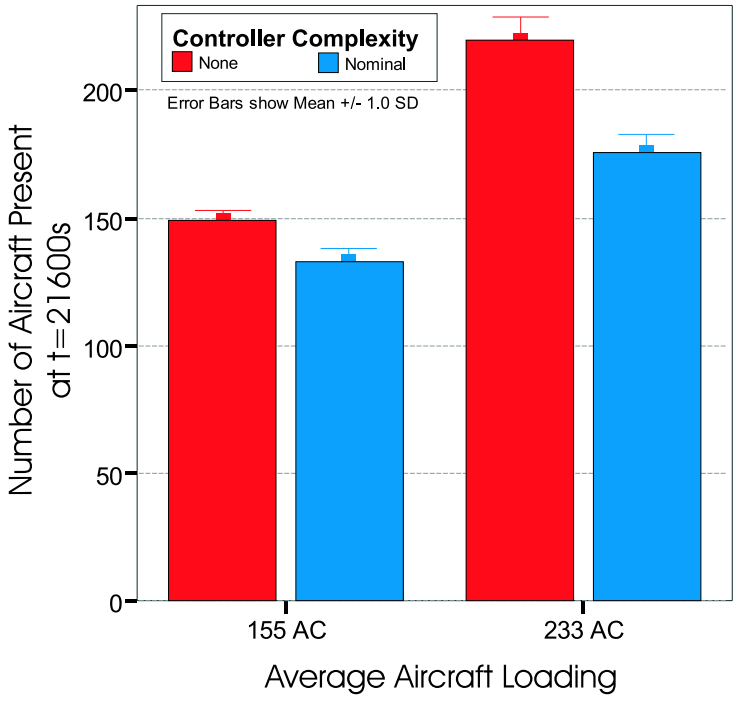


Figure 67: Comparison of the Number of Aircraft Present in Simulation at t=21600s for Different Controller Complexity Levels

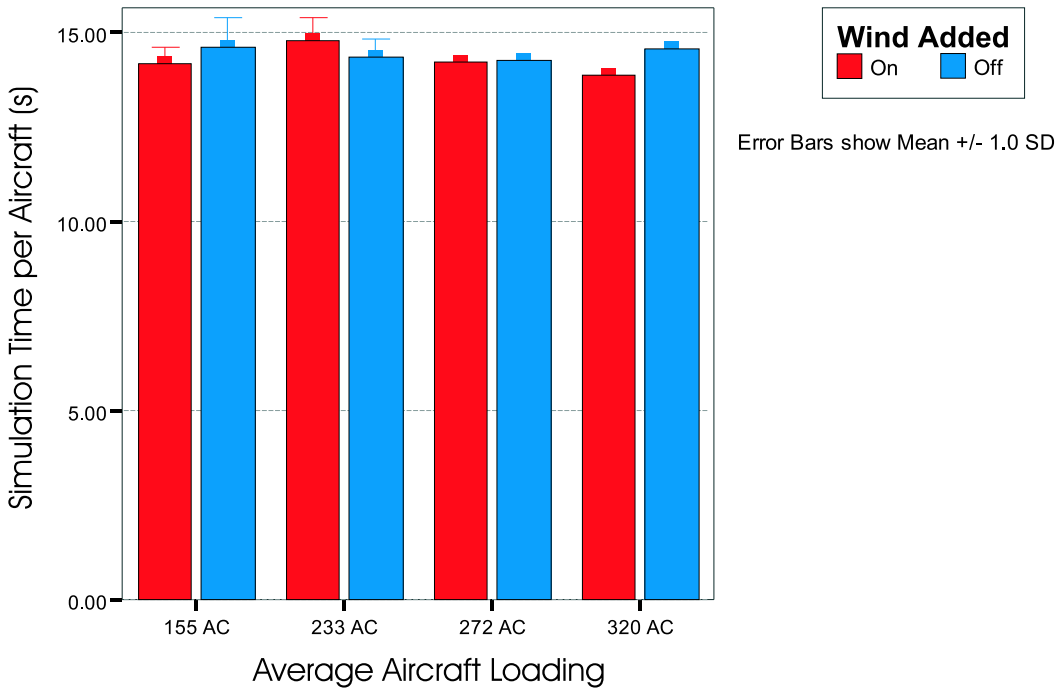


Figure 68: Simulation Time per Aircraft as a Function of Aircraft Loading

constant across different levels of control logic as seen in Figure 69. The implication of this consistency is that the control logic has indeed been implemented efficiently. It appears that the impact of controller logic complexity is minimal compared with that of the average aircraft loading on the overall simulation time, and that this impact is consistent over a range of aircraft loadings.

The final objective of the Simple Core Europe scenario was to determine the effect of:

- the frequency of enroute controller update
- the frequency of the enroute control module stack machine execution
- the aircraft loading
- the implementation of the wind field model

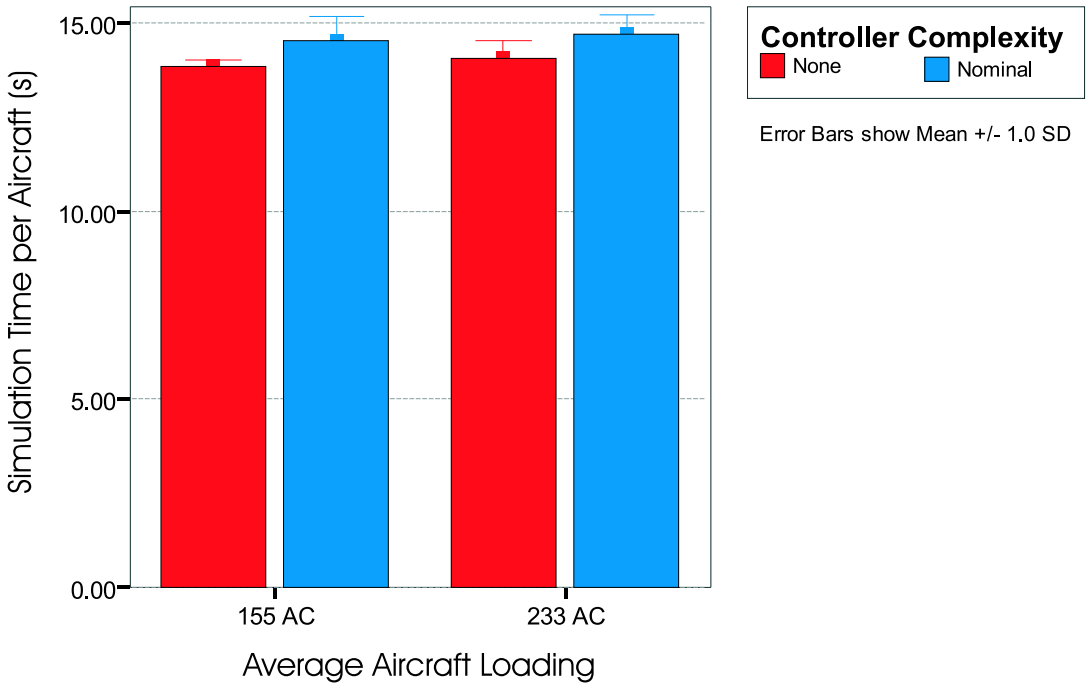


Figure 69: Comparison of Simulation Time per Aircraft for Different Controller Complexity Levels

on the effectiveness of the control module logic. Three different aspects of the control module effectiveness were examined:

- the control module response time
- the average aircraft separation at the point of intervention
- average flight time deviation

Figure 70 illustrates the average control module response time for all of the commands issued by the control module. The control module response time is the time from the last aircraft update to the execution of the control module command by the pilot module. It is important to note that for all of the cases tested the average control module response time was within the 10s-50s controller response time cited in the literature and set as a requirement of the control module. It is also

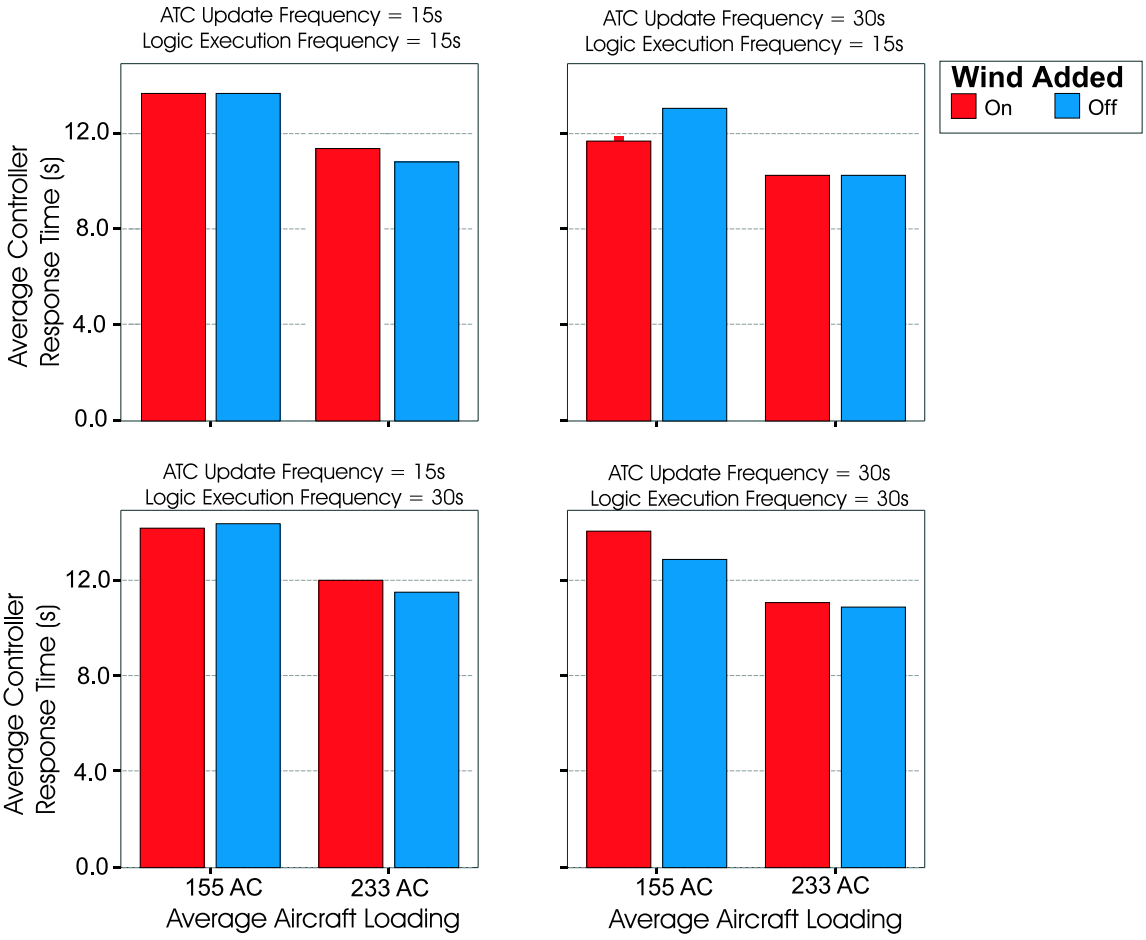


Figure 70: Comparison of Controller Response Times

interesting to note that as the aircraft loading increased, the average control module response time decreased. This is expected because, as explained in Section 6.6 on page 101, the controller module time line synchronises with the simulation time line at aircraft generation and removal. Consequently, the more frequently aircraft are generated or removed, the faster the control module response time becomes.

Figure 71 illustrates the average aircraft separation at the time of the control module intervention. The control module logic was designed to identify possible conflicts within a 30Nmi radius. It is important to note that the control module

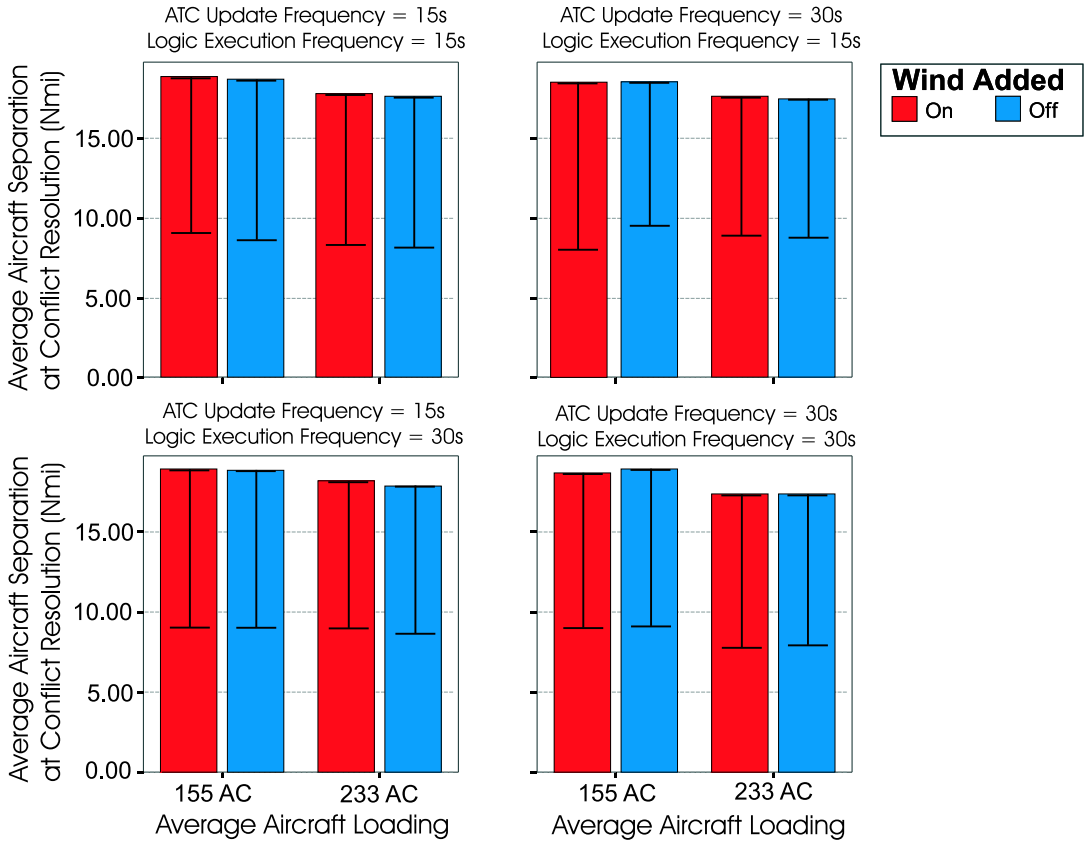


Figure 71: Comparison of Average Aircraft Separation at Controller Intervention

response time has translated into approximately 10Nmi of closure. However it is also important that the average aircraft separation is above the 10Nmi threshold. Upon closer inspection it is observed that the standard deviation for these calculations is between 8Nmi to 10Nmi, leading to a high number of cases without adequate separation. There are several reasons for this situation. Firstly, the control logic does not allow aircraft overtaking one another to be permanently reassigned to different altitude levels. Secondly, the control logic is limited to resolving two aircraft conflicts, thus the resolution of any conflict involving more than one aircraft may result in the creation of an additional conflict.

Figure 72 illustrates the average aircraft flight time deviation caused by the

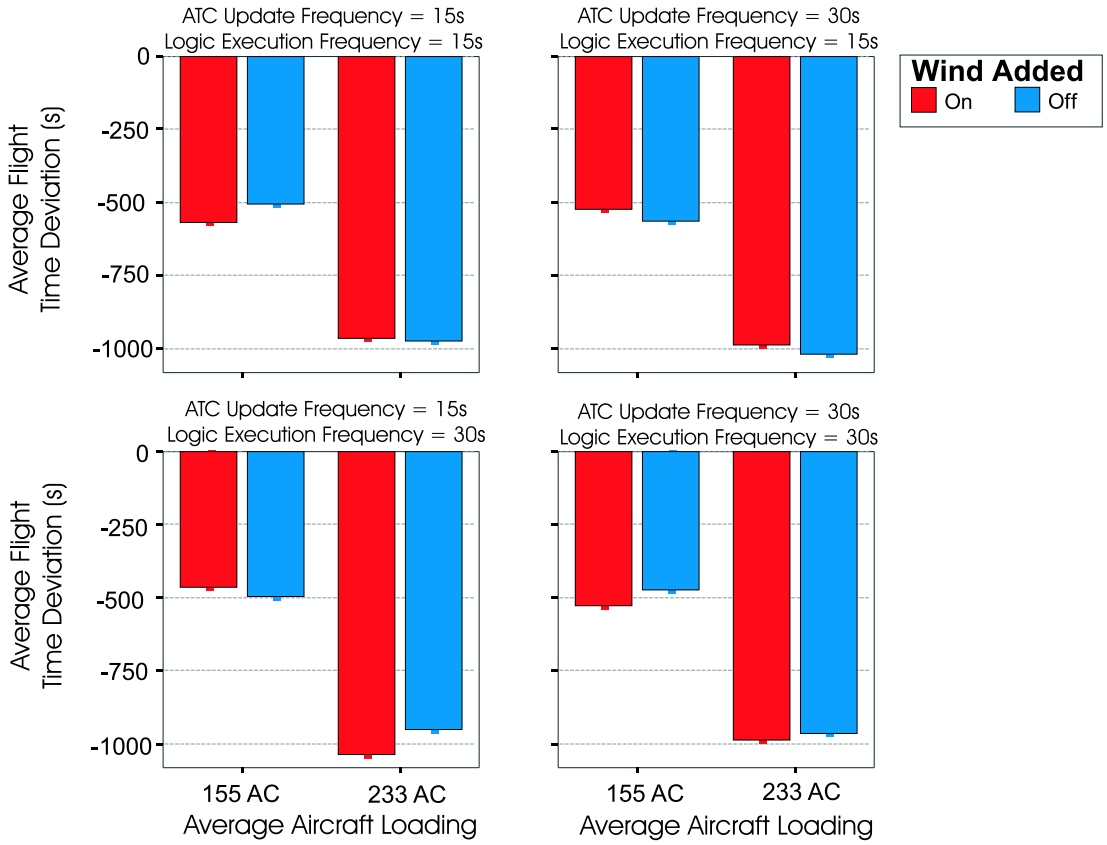


Figure 72: Comparison of Average Flight Time Deviation

control module logic. It is interesting to note that the introduction of the control logic decreased the average flight time between 8 and 15 minutes. This decreased flight time is due to increased use of lower flight levels for conflict avoidance, which allowed the aircraft to increase their velocity, see Figure 65. It is also interesting to note that the addition of wind had a small but noticeable impact on the flight time deviation, as can be expected.

On the other hand the differences in control module update rates and control module stack machine execution rates appear to have had little effect on the average flight time deviation. This insensitivity is to be expected as there was no noticeable difference in the average aircraft separation at conflict resolution caused by these

two metrics. For instance, if the conflict resolution occurs at the same distance, then the flight plan deviation will last for a similar period of time (whatever the time necessary to resolve the conflict successfully), regardless of the frequency of the updates and stack machine execution. It does appear, however, that the average aircraft loading has a significant effect on the average flight deviation. This is also to be expected as the number of aircraft overtaking one another in the higher aircraft loading cases will be significantly greater than in the lower aircraft loading cases, leading to an increase in the number of aircraft flying at lower flight levels for long periods of time. If aircraft fly at lower flight levels, their flight time will be reduced, increasing the flight time deviation.

7.5 Summary

Overall it appears that the simulation tool has achieved the objectives established in Chapter 2. The three scenarios described in this chapter have demonstrated the different capabilities of the simulation tool. The simulator has shown that it is capable of running fast time on a standard personal computer even under a load similar to the Simple Europe scenario. In the Simple Core Europe scenario presented here, the simulator ran between 5 and 11 times faster than real time depending on the case parameters. This scenario also demonstrated that the simulator has the capability to simulate over 300 aircraft simultaneously.

The control module has demonstrated the simulator's ability to simulate data link messages, which include both intent broadcasts and aircraft state information. Additionally, the simulation is capable of incorporating the wind field model. It has also shown that the simulator is capable of implementing ATC restrictions on speed, altitude and heading. The simulator has also demonstrated that it can achieve the desired response time even under a heavy aircraft loading. However, the simulator is less capable at successfully separating all aircraft conflicts, especially those which involve more than two aircraft.

CHAPTER 8

Discussion

8.1 Research Objectives

The primary objective of this research has been to create a high-level, low-fidelity simulation tool for the purpose of conducting exploratory research into radical new approaches to enhance the airspace capacity with the following capabilities and characteristics :

- *To model airspace characteristics appropriate to the 2020 time frame*
- *To run on a single personal computer (PC)*
- *To run in fast time*
- *To be open source*
- *To be nonproprietary*
- *To simulate atmospheric conditions*
- *To simulate data link communications*
- *To simulate* ATC *guidance*

The need for a simulator was deemed necessary due to the limited flexibility of existing commercial airspace simulators, such as RAMS and TAAM [64], and to model innovative airspace structures and procedures, which are required to increase airspace capacity. A search of the literature indicated that although such a simulation tool did not exist, research into increasing airspace capacity is needed and is currently being undertaken in both the U.S. and Europe. This research has sparked the creation of several non-commercial airspace simulators by NASA, NLR and EUROCONTROL. However, these simulators are high fidelity and require dedicated

facilities, extensive input data and months of training. The simulation presented in this thesis is aimed at filling a niche role created by these simulations: a low fidelity simulation that can be run on a single PC, runs in fast-time, and is open source. These objectives were then translated into a set of simulation requirements as listed in Table 16.

8.2 Simulation Review

This thesis has presented a high-level low fidelity airspace simulator for the purpose of investigating airspace capacity benefits derived from radical airspace structures. The simulator is compromised of four modules:

- ❖ Multiple Aircraft Performance Module (MAPM)
- ❖ Wind Field Module
- ❖ Airspace Module
- ❖ Control Module

which have been discussed in detail in Chapters 3 through 6. The structure of the simulator is illustrated in Figure 73, where the arrows indicate the direction of data flow, (solid arrowheads represent direct links and open heads representing data link). The top row illustrates the different input files required by each module. The middle row illustrates how the different modules interact with each other. Note that, although both the Pilot are Control module are contained within the ATC module, that the Pilot portion is held within the Airspace module whereas the Control module is not. This is because the Pilot module must share data with the Multiple Aircraft Performance module directly. The bottom row illustrates the output files generated by the simulation and the module which generates them. The modules are summarised individually in Sections 8.2.1 - 8.2.4.

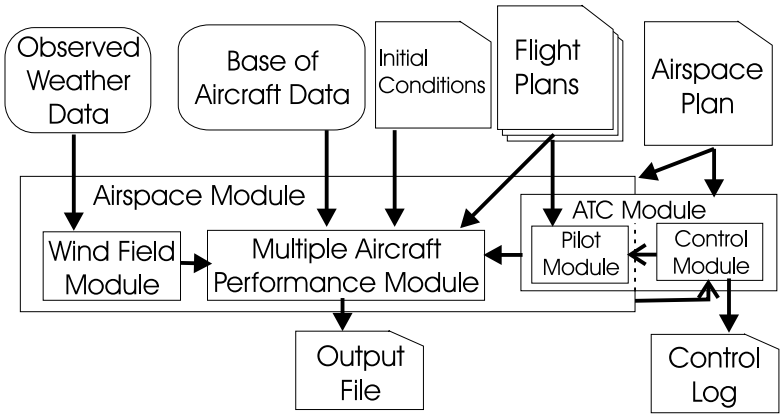


Figure 73: Simulation Architecture

8.2.1 Multiple Aircraft Performance Module

The multiple aircraft performance module is the heart of the simulation. It governs the creation, movement and removal of aircraft. Individual aircraft motion is modelled by the BADA v3.3 performance database. The simulation's adherence to the BADA database has been validated with an error of less than 0.5% over 2,000km. The simulation assumes that all aircraft are under FMS guidance due to the economic and navigation benefits provided by an FMS. To provide FMS guidance, the simulation assumes a spherical earth and employs great circle navigation between way points. The navigation and guidance algorithms have been tested and achieve an accuracy of less than half a kilometre deviation over a 2,000km flight, which is within the accuracy required by RNAV-1.

8.2.2 Wind Field Module

A wind field model is a key component of the airspace simulator. The wind field enables a wind field model from publicly available observed winds aloft data. Due to the significant impact that wind has on an aircraft's along-track navigation accuracy, the greater this accuracy, the greater the accuracy of conflict detection algorithms. The wind field model itself consists of a set of gridded wind data over a series of time and altitude levels. At each time step, the grids are interpolated to provide the wind

observed by each aircraft. Observed winds aloft data was chosen over forecasted winds aloft, which are traditionally used in airspace simulations, because of the discrepancies detected by Rodney et al. [67]. The National (US) Oceanographic and Atmospheric Administration's National Profiler Network was chosen as the wind field model data source because it was the only source of observed winds aloft that provided a consistent set of data over a large geographic region and over a long period of time. Additionally, it is freely available over the Internet enabling researchers to create different wind models for a range of weather conditions. The wind field model has been evaluated and found to have an average accuracy of 2.5m/s in magnitude and 5^0 in direction.

8.2.3 Airspace Module

The purpose of the airspace module is to organise the simulated aircraft to facilitate conflict detection and data link algorithms. These algorithms require the expeditious location of all aircraft within a given geographic region. The airspace module employs an indexed point-region quadtree to accomplish this function. The point-region style quadtree is chosen because the size of its regions and the structure of the tree are independent of the order aircraft are inserted or removed from the tree, which can occur asynchronously in an airspace simulation. The quadtree uses authalic coordinates to maintain equal area regions, thus avoiding, without the need to transform the coordinates from latitude and longitude, which would additional more computations. To further reduce the computational load required by the quadtree, the quadtree is indexed using a form of Morton ordering, which allows efficient quadtree traversal and minimising computation time.

8.2.4 Control Module

The purpose of the control module is to monitor the simulated airspace for the possible loss of aircraft separation, and upon detection of such a loss, to direct the

aircraft involved to resolve the conflict and restore the separation. The conflict detection and resolution logic is contained in an array of instructions which are implemented by a stack machine. This array of instructions is compiled from a plain-text, control logic file which is written in a pseudo code using a specialised control vocabulary. Utilising the control vocabulary provides a wide range of control options, and pre-compiling the instructions at initialisation maximises the runtime efficiency of the individual control modules. The control module is implemented as a set of independent Linux process, one per control sector. There are two distinct types of control modules: an enroute controller and a TMA controller. Enroute controllers are assigned to a sector of airspace, which corresponds to a quadtree region. Although using quadtree regions to represent airspace sectors is restrictive, it aids in the efficiency of the conflict detection and resolution algorithms. TMA controllers are responsible for merging aircraft on approach to a specified airport. The control module has been evaluated and is capable of resolving conflicts and merging arrival traffic. However, its effectiveness is highly dependent upon the control logic employed, the nature of the conflict (whether it involves more than two aircraft) and the frequency of the aircraft position updates and stack the machine execution.

8.3 Simulator Requirements: Origin & Satisfaction

As this simulation is aimed at modelling aircraft and air traffic in the 2020 time frame, it was necessary to simulate a large number of aircraft simultaneously and in total. The projected daily peak for the busiest European airport (London's Heathrow) in the year 2020 is 150 aircraft an hour, so it was decided that the simulator should be able to simultaneously simulate twice this number of aircraft, and should be able to sustain this level of aircraft for a simulated period of 12-16 hours. Therefore, the simulation should also be able to accommodate 4,000 aircraft in total.

Table 16: Simulator Requirements

Simulator Requirements	Achieved	Section
Capable of running fast time	Yes	7.4
Capable of running on a standard PC	Yes	7.4
Handle 300 aircraft simultaneously	Yes	7.4.3
Handle 4,000 aircraft total	Yes	3.4.3
Allows data link simulation	Yes	6.3
Allows intent broadcast	Yes	6.3
Allows aircraft state broadcast	Yes	6.3
Allows current wind broadcast	Yes	6.3
Allows flight phase broadcast	Yes	6.3
Allows data link feed back	Yes	6.5
Includes wind field simulation	Yes	7.4.2
Includes conflict detection and avoidance algorithms	Yes	6.8

Speed and Fidelity The simulator has met the requirement to run on a standard PC and to run in fast-time as illustrated by the results presented in Section 7.4. These scenarios also illustrate that the simulator run time is primarily a function of of the number of concurrently simulated aircraft and the number of airspace sectors and airports which are modelled. It is important to note though that the simulation run time has been shown by these scenarios to be a linear function of the number of concurrently simulated aircraft. A linear relationship indicates that the simulation structure and algorithms are efficiently implemented and implies that the simulation is capable of handling larger simulations, i.e. those with a greater number of concurrent aircraft without incurring significant time penalties.

Capacity Requirements The simulator met the requirement for simulating 300 aircraft simultaneously in the Core Europe scenario. The results are presented in Section 7.4.3 where an average aircraft loading of 320 aircraft was achieved over a period of 10, hours which produced a total of 1,500 aircraft. Although none of the scenarios presented in Chapter 7 have explicitly demonstrated that the simulator is capable of simulating a total of 4,000 aircraft, this capability was demonstrated earlier in Chapter 3. The total number of aircraft handled by the simulation is more

of an issue of storage space than of computational limitations caused by inefficient algorithms. Table 4 on page 34 has illustrates that the storage required by the simulator to store different amounts of aircraft depending on the average flight duration and the recorded resolution. If it is necessary to store more data than the default 700MB, the overall storage parameter needs to be changed to accommodate larger output files.

Wind Field Requirement Management of aircraft in airspace which includes an unpredictable wind field, is vital to increasing airspace capacity. Correct modelling of the variability of winds is, therefore, critical in a viable simulation. It has been suggested that future improvements to the airspace capacity will involve improved wind modelling through broadcasting wind field data [35]. To insure that the simulator is capable of modelling such an airspace structure, it was necessary to include a wind field model in the simulation. This model has been developed and integrated into the simulator as discussed in Chapter 4. Additionally the simulation is capable of broadcasting wind data observed by the aircraft. In both the Landing at Gatwick and the Simple Europe scenarios evaluated in Chapter 7, the effect of the wind field model was small but noticeable.

The wind field model used in the simulator is based on observed winds aloft data which is recorded over a section of the earth. In order to provide coverage over the entire earth, the data is repeated in a patchwork fashion. This repetition imposes limits on the validity of the wind field model for aircraft flying through more than one patch, and for patches near the poles where the distance between latitude and longitude are significantly distorted from their collection site.

Data Link Requirement In the future, data link systems will play an important role in transmitting information between the flight deck and other aircraft and between the flight deck and the air traffic service provider. To this end an airspace simulation needs the ability to model data link broadcasts. Although the U.S. has chosen a set of data link technologies to develop, the composition of the messages has yet to be decided. Consequently, the airspace simulation should be able to easily

alter the content of data link messages used by the simulator.

This simulator not only allows data link messages to be explicitly simulated through Linux message queues, it also provides a convenient method for modifying the content of the messages through the creation of new message types. In addition the pilot module, enables the simulation to implement feedback from the data link messages. The simulator's data link capability and its ability to implement feedback from data link messages can be seen in the Landing at Gatwick and Core Europe scenarios evaluated in Chapter 7, where seven different message types, were used to control aircraft.

Air Traffic Control Simulation Lastly, an airspace simulator needs to have a means of conflict detection and resolution as well as traffic flow management. This simulator has implemented its conflict detection and resolution algorithms in a two part control module, as described in Chapter 6 and summarised in Section 8.2.4. The effectiveness of the control module is dependent upon the frequency of the ATC updates sent by the aircraft, the frequency of the stack machine execution, the complexity of the control logic and the frequency of aircraft generation, as shown in Chapter 7.

The most significant of these metrics is the the frequency of aircraft generation, as it affects both the control module lag time and the probability of multiple aircraft being in conflict with one another. Clearly the shorter the control module lag time, the more effective the control module is likely to be. However, even without any lag, the control module only implements its logic at the frequency specified by the stack machine execution frequency. Furthermore, the control logic is only able to act on the most recent information that has been sent to it, which is governed by the ATC update frequency. These three metrics combine to form the control module response time, which is the time between the last update sent to the control module by an aircraft and the time any effected aircraft receives instructions back from the control module. The response time has been shown to be consistent with that of human controllers, between 10-50s.

However, even if the control module response time is on par with human controllers, it is only able to implement the logic which is encoded into the instruction stack. This means that the overall effectiveness of the control module is highly dependent on the complexity of the control logic, and the implementation of the control logic. Currently the control logic is bound by the vocabulary provided. For example the control logic is not capable of resolving conflicts involving multiple aircraft, nor is it capable of implementing any strategic plans because the current set of vocabulary does not cater for these situations. However, new vocabulary is easily implemented and will be included in future improvements.

8.4 Suggestions for Further Work

While there is always room for improvement in any work, the Airspace Simulator presented in this thesis is quite complete. It includes the basic concepts incorporated into more sophisticated models and a few advanced concepts not included in those models. There are however, some areas where further work would enhance the simulation. This section presents a number suggestions for further improvement of the software, for the specified modules. It also includes some discussion on the potential computational impact of these suggestions.

Overall Simulator Future improvements to the overall airspace simulation should simplify the scenario creation through the introduction of graphical user interfaces for both input file creation, simulation execution, and output file investigation. The inclusion of a database of current airports, airway boundaries, and restricted airspace would accelerate the scenario set up process. For example it would be convenient to enter airspace parameters via a graphical user interface.

Multiple Aircraft Performance Module In order to improve the fidelity of the simulation a higher fidelity performance model could be implemented. At present the model implements the performance model held in the performance table files which is a simplification of the overall performance model held in the operations

performance file and the airline procedure file. This improved model would improve the validity of the simulation for maneuvers which do not adequately correspond to the performance model held in the PTF. As this addition could have a significant impact on the performance runtime of the simulator, it would be best to include it as a parameter which can be set by the operator.

Airspace Module In order to simulate more realistic sector geometries, and proposed concepts such as super sectors it is suggested that the control module sectors should be decoupled from the quadtree structure. This decoupling would also enable the extension of control module availability to above the 65^{th} parallel. However, the decoupling would require the addition of another large data structure and has the potential to significantly increase the computation time required to find possible aircraft conflicts. This is because with the elimination of the quadtree structure it would be necessary to calculate the distance between each aircraft pair in a sector and possibly all aircraft pairs across several sectors. Yet, as the calculation of great circle distances is an efficient calculation, for sectors with small numbers of aircraft, computing the distance between each aircraft pair may not prove unsurmountable.

Wind Module Future improvements to the wind field model should include improving the interpolation schemes used to generate the gridded wind model to reduce the error incurred by transferring the raw data into the uniform grid model. The sensor accuracy could also be introduced to degrade the aircraft's wind measurement. This sensor degradation would enable the simulation of the wind measured by the aircraft's sensors and transmitted via data link as such a measurement would not match exactly the wind experienced by the simulated aircraft. Additionally, observed wind data from an expanded profiler network should also be incorporated. Ideally wind data would be available for the geographic area of interest. Finally, convective weather should be incorporated into the model parameter, and the error associated with the repetition of the same weather model to cover large regions of the earth should be quantified.

Control Module The simplest improvement that should be made to the control module is the expansion of the controller vocabulary to allow the control logic to handle multiple aircraft conflict scenarios, and to facilitate strategic traffic flow management. An enhanced vocabulary would enable a larger variety of airspace scenarios to be simulated, and concepts such as aircraft prioritisation within sectors to enable conflict resolution in free flight scenarios to be investigated.

Another suggestion to improve computational efficiency is to assess the feasibility and efficiency of keeping a central quadtree representation of the entire airspace in a shared memory region accessible to all control modules. It is possible by reducing the overhead associated with each control module maintaining its own quadtree, that the simulator performance might increase significantly. However, this could present a bottle neck with multiple control modules needing to access information simultaneously.

A further improvement would be to improve the control module conflict resolution capabilities by providing the control modules with a strategic view of the airspace they are controlling. For example if the control module knew that aircraft flying a certain route were due to land soon, it could command them to descend rather than climb in a conflict situation. In addition a way of including the past commands that each aircraft has been given into the conflict resolution strategy would also improve the module's conflict resolution capabilities. Both of these improvements would also help the control module cope with conflicts involving more than one pair of aircraft. However to fully handle multiple aircraft conflicts the control vocabulary would need to be expanded. This expansion should include a new command to issue permanent flight level clearances for a given sector.

An ATM issue illustrated by the simulation is the uncertainty introduced by the use of data link communications: that the controller and the pilot know what messages they have sent, but they may not be aware if those messages were received correctly. To overcome this uncertainty, in this simulation the controller waits for the pilot to respond to any commands sent to it before issuing other commands, but the pilot cannot be certain that the controller is issuing a command based on

the very latest position update that it sent. This uncertainty becomes even more important if the controller were to issue multiple commands to an aircraft in rapid succession. All of these commands might be issued before the pilot has a chance to execute any of them. Should the pilot execute each command in order, or only execute the last command received. These are real ATM issues that will need to be addressed before data link communications can be implemented system wide.

8.5 Airspace Simulator Benefits

The airspace simulation developed in this thesis has many benefits in the core airspace module and in the auxiliary modules. The major benefits of this simulator in comparison with other available airspace simulations is that it is nonproprietary and runs on a single PC. The simulation can be compiled and run under Linux which is also both nonproprietary and open-source. This allows the simulator to be accessible to a much larger number of researcher organisations. Additionally, both the BADA database and the wind data are used in the simulator are also freely available to academic research staff.

Additional beneficial aspects of this simulator are that it is a high-level, low-fidelity, open source simulator. These aspects reduce learning curve for users of the simulator and allow researchers to quickly prototype their ideas for new airspace structure or data link feed back. Additionally these aspects of the simulator enable it to run in fast time, which in turn allows results to be generated in a matter of days instead of weeks. Consequently a much higher number of cases can be studied.

The primary benefit of the wind field model is that it is based on observed winds aloft data instead of forecasted winds aloft, which makes it unique among the existing airspace simulators discussed. In addition the data on which the wind field model is based, is freely accessible and allows multiple wind models to be created for a variety of atmospheric conditions.

While not a novel concept, the rule based control logic, as discussed in Chapter 6 does provide significant benefits over rule based control implemented by other

simulators, such as RAMS, because the rule based control logic is implemented in a more intuitive style. This airspace simulator uses a set of control vocabulary to create a set of control logic which reads like a simplified natural language instead of a pseudo code language. Using natural language style allows the control logic to be created and understood by individuals with little or no knowledge of computer programming. The ability to include stake holders, such as experienced air traffic controllers, into the design and testing of novel new airspace structures is vital to their success and eventual implementation. In addition it provides a greater variety of implementable control scenarios.

APPENDIX A

Control Vocabulary

The controller module interprets a set of commands from a text file to construct the code for the stack machine, as outlined in Chapter 6. This text file is constructed from a specific set of vocabulary, which are described in the following sections. The control vocabulary consists of key words, command words, calculation words, and two different types of variables. The variables are distinguished between those which can be directly set in the control script, internal variables, and those which cannot be directly set, external variables. All of the other types of words act on either internal or external variables. The specifics of these words and their interaction is described in the following sections.

A.1 Key Words

Key words serve two primary functions. The first is to serve as logical operators which provide the framework around which the rest of the control script is constructed. The second is to serve as trigger words for the code interpretation algorithm to set the different variables and issue commands. The 21 key words are listed below along with their use. Please note that $< condition >$ implies a logical condition statement and $< expression >$ implies a mathematical statement which can be evaluated; a $< constant >$ implies a numerical constant; and $< time >$ implies an amount of time in seconds.

IF..THEN $IF < condition > THEN$

$< expression >$

END

ELSE $IF < condition > THEN$

$< expression >$

$ELSE$

$< expression >$

END

UNLESS $UNLESS < condition > DO$

$< expression >$

END

WHILE $WHILE < condition > DO$

$< expression >$

END

Pauses the control module until an update from the airspace module is received to change the condition.

REPEAT $REPEAT < expression > UNTIL$

$< condition >$

FOR $FOR < time > DO$

$< expression >$

END

WAIT $< condition >$

DELAY $< time >$

S_WHILE *Does not exit from execution upon failure.*

S_WHILE $< condition > DO$

$< expression >$

END

OR *Logical OR.*

AND *Logical AND.*

STOP *Stops script execution and maintains location in instruction stack.*

EXIT *Exits script execution and resets location in instruction stack.*

SAMPLE $< INTERNAL \quad VARIABLE >, <$ *constant* $>$

GET $< CALCULATION \quad WORDS >$

Sets EXTERNAL VARIABLES in the background

SET $< INTERNAL \quad VARIABLE >$

INCREMENT $< INTERNAL \quad VARIABLE >$

Except Sector_ID, Conflict_Radius, Update_Rate or Stack_Overflow

COMMAND $< ACTION \quad WORDS >, < EXTERNAL \quad VARIABLES >$

Sets EXTERNAL VARIABLES in the background

A.2 Internal Variables

Variables which can be directly set in the control script are designated internal variables. The 13 internal variables are listed below.

Conflict_Radius is the radius within which a conflict will be declared. Any possible conflicts detected with a conflict radius larger than the declared radius are ignored by the controller.

Current_Conflict is the index of the `conflict array` which is currently selected.

Update_Rate is the rate at which the aircraft send updates to the controller.

Current_Init_App is the index of the `initial approach array` which is currently selected.

Current_Stack is the index of the `stack monitor` which is currently selected.

Current_Nearest is the index of the `AC` which is currently selected.

Current_Final_App is the index of the `final approach array` which is currently selected.

Current_Comp is the index of the `comparison array` which is currently selected.

Temp_Value is the value which has been stored.

Temp_Value2 is the value which has been stored.

Nearest_Stack is the index of the `stack monitor array` which is nearest to a given aircraft.

Current_Selected is the index of the `stack level` of either the `nearest stack` or the `current stack` currently selected stack, which is currently selected.

Stack_Overflow is the index of the `stack monitor array` which is the overflow stack for the currently selected `stack monitor`.

Depending on the type of controller used, different internal variables are available to be incremented as shown below:

Controller Conflict_Radius, Current_Conflict

TMA Controller Conflict_Radius, Current_Conflict, Current_Init_App, Nearest_Stack, Current_Selected, Stack_Overflow

A.3 External Variables

Variables which cannot be directly set in the control script are designated external variables. These variables are set through the use of other control words externally in the controller. The 58 external variables are described below.

Sector_ID is the control sector identifier. It corresponds to the quadtree array index of the quadtree region within which this controller acts.

AC1, AC2 are the generation numbers designating specific aircraft.

AC1_PKT, AC2_PKT are the pointers to `aircraft state records` and to `master records` for specific aircraft.

AC1_alt, AC2_alt are the altitude in meters of AC1 and AC2.

AC1_heading, AC2_heading are the heading in radians of AC1 and AC2.

AC1_vel, AC2_vel are the velocity in $fracms$ of AC1 and AC2.

AC1_next_WP, AC2_next_WP are the pointers to the next waypoint for AC1 and AC2.

AC1_next_alt, AC2_next_alt are the altitude of the next flight plan segment for AC1 and AC2.

ACAC_Dist is the distance in meters between AC1 and AC2.

AC1_Boundary_Dist, AC2_Boundary_Dist are the distance in meters between AC1 and AC2 and the nearest control sector boundary.

Max_Vel_AC1, Max_Vel_AC2 are the maximum velocities for AC1 and AC2. By default they are 10% over normal velocity when climbing, 20% over normal velocity when cruising, and 25% over normal velocity when descending.

Wake_Type_AC1, Wake_Type_AC2 are the wake classifications (High, Mid, Low) for AC1 and AC2.

Max_Alt_AC1, Max_Alt_AC2 are the maximum altitude levels in meters for AC1 and AC2.

No_Conflicts is the number of conflicts found within the conflict radius.

Scenario is the conflict scenario: overtake, acute

Density is the airspace density in $fracnumberofaircraftsquaremeter$

Time is the time of the last update from the simulator. angle, right angle, obtuse angle, or head on.

No_Nearest_AC is the number of entries in `nearest array` ????

Nearest_AC is the generation number of the aircraft at `current nearest` index of `nearest array`.

Nearest_AC_Phase is the phase of flight (takeoff, climb, cruise, approach, hold, landing, ground) of the `nearest AC`

Nearest_AC_Hdist is the horizontal distance in meters of the `nearest AC` to the designated object.

Nearest_AC_Vdist is the vertical distance in meters of the `nearest AC` to the designated object.

Nearest_AC_Alt is altitude in meters of the `nearest AC`.

Nearest_AC_Heading is the true heading in radians of the `nearest AC`.

Nearest_AC_Wake is the wake classification of the `nearest AC`

Nearest_Apt is the index of the nearest airport to the designated aircraft in `airport array`.

Nearest_Stack is the index of the nearest stack to the designated aircraft in `stack array`.

No_Init_AC is the number of aircraft in the `initial approach array`

INIT_AC is the generation number of the aircraft at `current init app` index of the `initial approach array`

No_Final_AC is the number of aircraft in the `final approach array`

No_Nearest_Stack_AC is the number of aircraft in the `nearest stack`

Selected_AC is the generation number of aircraft at `current selected` index of the `selected array`

Selected_AC_Dist is the distance in meters of the aircraft `selected AC` from the runway threshold.

Selected_AC_Alt is the altitude in meters of the aircraft `selected AC` from the runway threshold.

No_Current_Stack_AC is the number of aircraft in the stack designated `current stack`.

No_Comp_AC is the number of aircraft in the comparison array designated `comp_ar`.

Comp_AC is the generation number of the aircraft in the comparison array at the index `current_comp`.

Comp0_AC is the generation number of aircraft in the comparison array at the index `current_comp` - 1.

Comp2_AC is the generation number of aircraft in the comparison array at the index `current_comp` + 1.

Comp_AC_Est is the estimated time of arrival of the aircraft in the comparison array at the index `current_comp` calculated by `GET TIME_EST`.

Comp_AC_Sep is the separation time required for the aircraft in the comparison array at the index `current_comp` to follow another aircraft.

Comp_AC_Alt is the altitude in meters of the aircraft in the comparison array at the index `current_comp`.

Comp_AC_Heading is the heading in radians of the aircraft in the comparison array at the index `current_comp`.

Final_AC is the generation number of the aircraft in the final approach array at the index `current_final_app`.

Final_AC_Timeout is the time aircraft in the final approach array at the index `current_final_app` has been in the final approach array.

Final_AC_Alt is the altitude in meters of the aircraft in the final approach array at the index `current_final_app`.

Final_AC_Heading is the heading in radians of the aircraft in the final approach array at the index `current_final_app`.

Depending on the type of controller used, different external variables are available to be incremented as shown below:

Controller

Sector_ID,
AC1, AC1_PKT, AC1_alt, AC1_heading, AC1_vel, AC1_next_WP, AC1_next_alt,
Wake_Type_AC1, Max_Alt_AC1, Max_Vel_AC1,
AC2, AC2_PKT, AC2_alt, AC2_heading, AC2_vel, AC2_next_WP, AC2_next_alt,
Wake_Type_AC2, Max_Alt_AC2, Max_Vel_AC2,
ACAC_Dist, No_Conflicts, Scenario,
Density, Time,
No_Nearest_AC, Nearest_AC, Nearest_AC_Phase, Nearest_AC_Hdist, Nearest_AC_Vdist,
Nearest_AC_Wake, Nearest_AC_Alt, Nearest_AC_Heading
Nearest_Apt

TMA Controller

Sector_ID,
AC1, AC1_PKT, AC1_alt, AC1_heading, AC1_vel, AC1_next_WP, AC1_next_alt,
Wake_Type_AC1, Max_Alt_AC1, Max_Vel_AC1,
AC2, AC2_PKT, AC2_alt, AC2_heading, AC2_vel, AC2_next_WP, AC2_next_alt,
Wake_Type_AC2, Max_Alt_AC2, Max_Vel_AC2,
ACAC_Dist, No_Conflicts, Scenario,
Density, Time,
No_Nearest_AC, Nearest_AC, Nearest_AC_Phase, Nearest_AC_Hdist, Nearest_AC_Vdist,
Nearest_AC_Wake, Nearest_AC_Alt, Nearest_AC_Heading
Nearest_Apt
Nearest_Stack, No_Nearest_Stack_AC, No_Current_Stack_AC,
No_Init_AC, INIT_AC, No_Final_AC
Selected_AC, Selected_AC_Dist, Selected_AC_Alt

No_Comp_AC, Comp_AC, Comp0_AC, Comp2_AC, Comp_AC_Est, Comp_AC_Sep, Comp_AC_Alt, Comp_AC_Heading

Final_AC, Comp_AC_TimeOut, Final_AC_Alt, Final_AC_Heading

A.4 Setting Internal Variables

Internal Variables are set by using the key word `SET` in the manner shown below:

$SET \quad < INTERNAL \quad VARIABLE >< expression >$

```
SET CONFLICT_RADIUS 30 nmi
```

However there are three exceptions to this rule and they are described below:

CURRENT_STACK

$SET \quad CURRENT\_STACK < STACK\_ID >$

```
SET CURRENT_STACK LARCK
```

NEAREST_STACK

$SET \quad NEAREST\_STACK < STACK\_ID >$

```
SET NEAREST_STACK LARCK
```

STACK_OVERFLOW

$SET \quad STACK\_OVERFLOW < INTERNAL \quad VAR >, < STACK\_ID >$

```
SET STACK_OVERFLOW (CURRENT_STACK, NEAREST_STACK), LARCK
```

A.5 Incrementing Internal Variables

Several Internal Variables represent indices in arrays, and as such may be incremented as well as set to a specific value. Internal variables are incremented by using the key word `INCREMENT` in the manner shown below:

$INCREMENT \quad < INTERNAL \quad VARIABLE >$

```
INCREMENT CURRENT_CONFLICT
```

Depending on the type of controller used, different internal variables are available to be incremented as shown below:

Controller CURRENT_CONFLICT, CURRENT_NEAREST

TMA Controller CURRENT_NEAREST, CURRENT_SELECTED, NEAREST_STACK, CURRENT_INIT_APP, CURRENT_COMP, CURRENT_FINAL_APP

A.6 Calculation Words

Calculation words are used in conjunction with the key word `GET` to call a series of functions in the controller module to set external variables. The specific usage of each of the 19 calculation words is listed below:

SECTOR_DENSITY

GET SECTOR_DENSITY<EXTERNAL VAR>

`GET SECTOR_DENSITY SECTOR_ID;` sets `DENSITY`

INTER_AC_DIST

GET INTER_AC_DIST<EXTERNAL VAR>,<EXTERNAL VAR>

`GET INTER_AC_DIST AC1 AC2;` sets `ACACDIST`

CONFLICTS

GET SECTOR_DENSITY<INTERNALVAR>,<INTERNALVAR>

`GET CONFLICTS SECTOR_ID CONFLICT_RADIUS;` sets `NO_CONFLICTS, CURRENT_CONFLICT` and *conflict array*

DIST_TO_BOUNDARY

GET DIST_TO_BOUNDARY<EXTERNAL VAR>

`GET DIST_TO_BOUNDARY (AC1, AC2, AC1_PTR, AC2_PTR);` sets `BOUNDARY_DIST`

NEAREST_TO_APT

GET NEAREST_TO_APT<AIRPORT_ID>,<EXPRESSION>

`GET NEAREST_TO_APT EGKK 30 nmi;` sets `NO_NEAREST_AC, CURRENT_NEAREST` and *nearest array*

NEAREST_APT

GET NEAREST_APT<EXTERNALVAR>

`GET NEAREST_APT (AC1, AC2, AC1_PTR, AC2_PTR);` sets `NEAREST_APT`

NEAREST_STACK

GET NEAREST_STACK<EXTERNALVAR>

`GET NEAREST_STACK (AC1, AC2, AC1_PTR, AC2_PTR);` sets `NEAREST_STACK`

LONGEST_ON_STACK

GET LONGEST_ON_STACK<INTERNALVAR>

`GET LONGEST_ON_STACK (NEAREST_STACK, CURRENT_STACK);` sets `SELECTED_AC`, `CURRENT_SELECTED` and *selected aircraft array*

LOWEST_ON_STACK

GET LONGEST_ON_STACK<INTERNALVAR>

`GET LONGEST_ON_STACK (NEAREST_STACK, CURRENT_STACK);` sets `SELECTED_AC`, `CURRENT_SELECTED` and *selected aircraft array*

HEAVY_IN_STACK

GET HEAVY_IN_STACK<INTERNALVAR>

`GET HEAVY_IN_STACK (NEAREST_STACK, CURRENT_STACK);` sets `SELECTED_AC`, `CURRENT_SELECTED` and *selected aircraft array*

MED_IN_STACK

GET MED_IN_STACK<INTERNALVAR>

`GET MED_IN_STACK (NEAREST_STACK, CURRENT_STACK);` sets `SELECTED_AC`, `CURRENT_SELECTED` and *selected aircraft array*

LIGHT_IN_STACK

GET LIGHT_IN_STACK<INTERNALVAR>

`GET LIGHT_IN_STACK (NEAREST_STACK, CURRENT_STACK);` sets `SELECTED_AC`, `CURRENT_SELECTED` and *selected aircraft array*

FIRST_AC

GET FIRST_AC <CONSTANT> <CONSTANT> <CONSTANT> <CONSTANT> < CONSTANT> <CONSTANT>

GET FIRST_AC (*deg min sec (latitude), deg min sec (longitude)*); places aircraft closest to the coordinates given into *comparison array* and sets COMP_AC

SEP_REQUIRED

GET SEP_REQUIRED<EXTERNALVAR> <EXTERNALVAR> <CONSTANT>

GET SEP_REQUIRED (COMP_AC, COMP2_AC, 18000 ft); determines the time separation required for when the second aircraft follows the first at that altitude. Places the answer into COMP_AC_SEP associated with the aircraft listed second.

TIME_EST

GET TIME_EST<EXTERNALVAR> <CONSTANT> <CONSTANT> <CONSTANT> <CONSTANT> <CONSTANT> <CONSTANT> <CONSTANT> <CONSTANT>

GET TIME_EST (SELECTED_AC, COMP_AC, COMP2_AC, COMP0_AC) (*deg min sec (latitude), deg min sec (longitude)*), (altitude); estimates the time for the aircraft to get to the way point described by the latitude and longitude at the altitude specified. Places the answer into the corresponding *comparison array entry.*

LAST_ON_FINAL

GET FLAST_ON_FINAL

GET LAST_ON_FINAL; places generation number for the newest/last aircraft on the *final array* into FINAL_AC

ADDITIONAL_TIME_EST

GET ADDITIONAL_TIME_EST<EXTERNALVAR> <CONSTANT> <CONSTANT> < CONSTANT> <CONSTANT> <CONSTANT> <CONSTANT> <CONSTANT>

GET ADDITIONAL_TIME_EST (SELECTED_AC, COMP_AC, COMP2_AC, COMP0_AC) (*deg min sec (latitude), deg min sec (longitude)*), (altitude); estimate the time for aircraft to reach the way point associated with latitude, longitude

coordinates from the current *latitude and longitude coordinates used in* ***GET TIME_EST*** so that multiple way point paths can be estimated. Places the answer into the corresponding *comparison array entry.*

ORDERED_COMP_AR

GET ORDERED_COMP_AR

`GET ORDERED_COMP_AR;` orders the entries of *comparison array* in order of their `COMP_TIME_EST`

CLEAR_COMP_AR

GET CLEAR_COMP_AR <EXTERNALVAR>

`GET CLEAR_COMP_AR (COMP_AR, COMP0_AR, COMP2_AR);` removes the specified entry of the *comparison array*

A.7 Command Words

Action words are used in conjunction with the key word `COMMAND` to call a series of functions in the controller module to set external variables. The specific usage of each of the 14 calculation words is listed below:

CLIMB

COMMAND CLIMB<EXTERNAL VAR>,<EXPRESSION,ALT>

```
COMMAND CLIMB AC1 (AC1_ALT + 1000 ft);
```

DESCEND

COMMAND DESCEND<EXTERNAL VAR>,<EXPRESSION,ALT>

```
COMMAND DESCEND AC1 (AC1_ALT + 1000 ft);
```

RESTRICT_UPPER_FL

COMMAND RESTRICT_UPPER_FL<EXTERNAL VAR>,<EXPRESSION,ALT>,
<EXPRESSION,DURATION>

```
COMMAND RESTRICT_UPPER_FL AC1 (AC1_ALT - 1000 ft) 30 secs;
```

RESTRICT_LOWER_FL

COMMAND RESTRICT_LOWER_FL<EXTERNAL VAR>,<EXPRESSION,ALT>,

<EXPRESSION,DURATION>

```
COMMAND RESTRICT_LOWER_FL AC1 (AC1_ALT + 1000 ft) 30 secs;
```

RESTRICT_SPEED

COMMAND RESTRICT_SPEED<EXTERNAL VAR>,<EXPRESSION,SPEED>,

<EXPRESSION,DURATION>)

```
COMMAND RESTRICT_SPEED AC1 200 kts 30 secs;
```

INCREASE_SPEED

COMMAND INCREASE_SPEED<EXTERNAL VAR>,<EXPRESSION,SPEED>,

<EXPRESSION,DURATION>

```
COMMAND INCREASE_SPEED AC1 200 kts 30 secs;
```

ADD_WP

COMMAND ADD_WP<EXTERNAL VAR>,<EXPRESSION,HEADING>,

<EXPRESSION,DISTANCE>

```
COMMAND ADD_WP AC1 250 degs 30 km;
```

REPLACE_WP

COMMAND REPLACE_WP<EXTERNAL VAR>,<EXPRESSION,HEADING>,

<EXPRESSION,DISTANCE>

```
REPLACE_WP_WP AC1 250 degs 30 km;
```

CLIMB_LEVEL_BY

COMMAND CLIMB_LEVEL_BY<EXTERNAL VAR>,<EXPRESSION,ALT>,

<EXPRESSION,LAT>,<EXPRESSION,LONG>

```
CLIMB_LEVEL_BY AC1 30000 ft 46.2 degs 5.3 degs;
```

DESCEND_LEVEL_BY

COMMAND DESCEND_LEVEL_BY<EXTERNAL VAR>,<EXPRESSION,ALT>,

<EXPRESSION,LAT>,<EXPRESSION,LONG>

```
DESCEND_LEVEL_BY AC1 25000 ft 46.2 degs 5.3 degs;
```

DESCEND_LDG

COMMAND DESCEND_LDG<EXTERNAL VAR>,<EXPRESSION,ALT>,

<EXPRESSION,LAT>,<EXPRESSION,LONG>

```
DESCEND_LDG AC1 25000 ft 46.2 degs 5.3 degs;
```

TO_STACK

COMMAND TO_STACK<EXTERNAL VAR>,<STACK_ID>,<EXPRESSION,ALT>

```
TO_STACK SELECTED_AC LARCK 5000 ft;
```

LEAVE_STACK

COMMAND LEAVE_STACK<EXTERNAL VAR>,<STACK_ID>

```
LEAVE_STACK SELECTED_AC LARCK;
```

DATALINK_RATE

COMMAND DATALINK_RATE<EXPRESSION>

```
DATALINK_RATE 15 s;
```

APPENDIX B

Scenario Set-up Files

Appendix B contains the initialisation file for the three scenarios discussed in Chapter 7. The initialisation files are organised by scenario and then by initialisation file type. The flight plan files include the number of flight plan segments followed by a list of those flight plan segments. The ATC plans contain the simulation airspace set up. They include how many sectors, airports and holding patters (stacks) are to be simulated and a list detailing each one. The initial condition files contain the initialisation parameters for each aircraft type for a given flight plan. The control logic is included at the end. The control logic is pseudo code made up of the control vocabulary outlined in Appendix A.

B.1 North Atlantic Crossing

B.1.1 North Atlantic Crossing Flight Plans

```
# TEST program to fly JFK to GATWICK ;
#;
# Number of Segments;
0.800E01;
#;
#     PROCEDURE  THRUST                                  Waypoint Coords    Elapsed;
# Switch Mode   Mode   Phase     Altitude    Heading   Lat       Long       Time;
#;
# FINISH ;
LONG CLIMB_   CLIMB___    CL__ 3.7000000E04  000 +044.54.08 -067.09.23  00:00:00;
LONG CRUISE   CRUISE__    CR__ 3.7000000E04  000 +054.15.00 -020.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 3.3000000E04  000 +054.00.00 -010.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 2.7000000E04  000 +052.30.00 -005.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 2.3000000E04  000 +052.21.24 -001.39.41  00:00:00;
LONG DESC__   DESCENT_    APP_ 1.3000000E04  000 +051.27.10 -000.52.44  00:00:00;
LONG DESC__   DESCENT_    LND_ 2.0000000E03  000 +051.31.00 -000.04.00  00:00:00;
LONG DESC__   DESCENT_    LND_ 2.0000000E03  000 +051.31.00 -000.01.00  00:00:00;
#FI;
```

Figure 74: A320 JFK-EGKK Flight Plan

```
# TEST program to fly JFK to GATWICK ;
#;
# Number of Segments;
0.800E01;
#;
#     PROCEDURE  THRUST                                  Waypoint Coords    Elapsed;
# Switch Mode   Mode   Phase     Altitude    Heading   Lat       Long       Time;
#;
# FINISH ;
LONG CLIMB_   CLIMB___    CL__ 3.7000000E04  000 +044.54.08 -067.09.23  00:00:00;
LONG CRUISE   CRUISE__    CR__ 3.7000000E04  000 +054.15.00 -020.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 3.3000000E04  000 +054.00.00 -010.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 2.7000000E04  000 +052.30.00 -005.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 2.3000000E04  000 +052.21.24 -001.39.41  00:00:00;
LONG DESC__   DESCENT_    APP_ 1.3000000E04  000 +051.27.10 -000.52.44  00:00:00;
LONG DESC__   DESCENT_    LND_ 2.0000000E03  000 +051.31.00 -000.04.00  00:00:00;
LONG DESC__   DESCENT_    LND_ 2.0000000E03  000 +051.31.00 -000.01.00  00:00:00;
#FI;
```

Figure 75: B737-300 JFK-EGKK Flight Plan

```
# TEST program to fly JFK to GATWICK ;
#;
# Number of Segments;
0.800E01;
#;
#     PROCEDURE  THRUST                                  Waypoint Coords    Elapsed;
# Switch Mode   Mode   Phase     Altitude    Heading   Lat       Long       Time;
#;
# FINISH ;
LONG CLIMB_   CLIMB___    CL__ 3.7000000E04  000 +044.54.08 -067.09.23  00:00:00;
LONG CRUISE   CRUISE__    CR__ 3.7000000E04  000 +054.15.00 -020.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 3.3000000E04  000 +054.00.00 -010.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 2.7000000E04  000 +052.30.00 -005.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 2.3000000E04  000 +052.21.24 -001.39.41  00:00:00;
LONG DESC__   DESCENT_    APP_ 1.3000000E04  000 +051.27.10 -000.52.44  00:00:00;
LONG DESC__   DESCENT_    LND_ 2.0000000E03  000 +051.31.00 -000.04.00  00:00:00;
LONG DESC__   DESCENT_    LND_ 2.0000000E03  000 +051.31.00 -000.01.00  00:00:00;
#FI;
```

Figure 76: B737-800 JFK-EGKK Flight Plan

```
# TEST program to fly JFK to GATWICK ;
#;
# Number of Segments;
0.800E01;
#;
#     PROCEDURE  THRUST                                  Waypoint Coords    Elapsed;
# Switch Mode   Mode   Phase     Altitude    Heading   Lat       Long       Time;
#;
# FINISH ;
LONG CLIMB_   CLIMB___    CL__ 3.7000000E04  000 +044.54.08 -067.09.23  00:00:00;
LONG CRUISE   CRUISE__    CR__ 3.7000000E04  000 +054.15.00 -020.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 3.3000000E04  000 +054.00.00 -010.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 2.7000000E04  000 +052.30.00 -005.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 2.3000000E04  000 +052.21.24 -001.39.41  00:00:00;
LONG DESC__   DESCENT_    APP_ 1.3000000E04  000 +051.27.10 -000.52.44  00:00:00;
LONG DESC__   DESCENT_    LND_ 2.0000000E03  000 +051.31.00 -000.04.00  00:00:00;
LONG DESC__   DESCENT_    LND_ 2.0000000E03  000 +051.31.00 -000.01.00  00:00:00;
#FI;
```

Figure 77: B747-200 JFK-EGKK Flight Plan

```
# TEST program to fly JFK to GATWICK ;
#;
# Number of Segments;
0.800E01;
#;
#     PROCEDURE  THRUST                                  Waypoint Coords    Elapsed;
# Switch Mode   Mode   Phase     Altitude    Heading   Lat       Long       Time;
#;
# FINISH ;
LONG CLIMB_   CLIMB___    CL__ 3.7000000E04  000 +044.54.08 -067.09.23  00:00:00;
LONG CRUISE   CRUISE__    CR__ 3.7000000E04  000 +054.15.00 -020.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 3.3000000E04  000 +054.00.00 -010.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 2.7000000E04  000 +052.30.00 -005.00.00  00:00:00;
LONG DESC__   DESCENT_    CR__ 2.3000000E04  000 +052.21.24 -001.39.41  00:00:00;
LONG DESC__   DESCENT_    APP_ 1.3000000E04  000 +051.27.10 -000.52.44  00:00:00;
LONG DESC__   DESCENT_    LND_ 2.0000000E03  000 +051.31.00 -000.04.00  00:00:00;
LONG DESC__   DESCENT_    LND_ 2.0000000E03  000 +051.31.00 -000.01.00  00:00:00;
#FI;
```

Figure 78: MD-80 JFK-EGKK Flight Plan

B.1.2 North Atlantic Crossing ATC Plans

```
no_sectors  1;
no_airports 1;
no_stacks   0;
#;
# SectorID   MinLat    MaxLat    MinLon    MaxLon  Neighbors (N,E,S,W)  Script Logic;
#;
  16       3.00000E1 9.00000E1 -9.00000E1 0.00000E0  0   0   0   0     "blank.scr";
#;
# AirportID     Lat        Lon       Alt(ft)  Orientation(deg)   SectorID  Script Logic;
#;
   EGKK      051:28:39N  000:27:41E    80         90               16     blank.scr";
#;
# StackID  Lat    Lo AltMax(ft)  AltMin(ft)  Orientation(deg)   AirportID   Prev Stacks;
#;
#;
#FI;
```

```
no_sectors  3;
no_airports 1;
no_stacks   0;
#;
# SectorID   MinLat    MaxLat    MinLon    MaxLon     Neighbors (N,E,S,W)  Script Logic;
#;
  64       3.00000E1 4.86000E1 -9.00000E1 -4.50000E1  66   0   0   0        "v_sep.scr";
  66       4.86000E1 9.00000E1 -9.00000E1 -4.50000E1  0   67   64   0       "v_sep.scr";
  67       4.86000E1 9.00000E1 -4.50000E1 0.00000E0  0   0   66   0         "v_sep.scr";
#;
# AirportID     Lat        Lon       Alt(ft)  Orientation(deg)   SectorID  Script Logic;
#;
   EGKK      051:28:39N  000:27:41E    80         90            67         "blank.scr";
#;
# StackID Lat    Lon   AltMax(ft)  AltMin(ft)  Orientation(deg)   AirportID   Prev Stacks;
#;
#;
#FI;
```

B.1.3 North Atlantic Crossing Initial Conditions

B.1.4 TIMBA 2B

```
# Initial Condition File;
#;
# CONFIGURATION;
no_ac                0.500000E+01;
# TIME;
time_step            0.100000E+01;
start                0.100000E+01;
finish               0.720000E+04;
# DESTINATION;
ac_state.destination EGKK;
# A320;
# AIRCRAFT_STATE;
ac_state.lat         +040.39.00;
ac_state.long        -073.47.00;
#ac_state.lat         +055.39.00;
#ac_state.long        -025.47.00;
ac_state.alt         3.700000E+04;
ac_state.mass        0.605320E+05;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.231500E+03;
ac_state.V_CAS       0.200000E+03;
ac_state.V_G.N       0.162000E+03;
ac_state.V_G.E       0.000000E+00;
ac_state.heading     0.392699E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       CR__ ;
#;
# B733;
# AIRCRAFT_STATE;
ac_state.lat         +040.39.00;
ac_state.long        -073.47.00;
ac_state.alt         3.700000E+04;
ac_state.mass        0.605320E+05;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.219668E+03;
ac_state.V_CAS       0.200000E+03;
ac_state.V_G.N       0.162000E+03;
ac_state.V_G.E       0.000000E+00;
ac_state.heading     0.392699E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       CR__ ;
#;
# B738;
# AIRCRAFT_STATE;
ac_state.lat         +040.39.00;
ac_state.long        -073.47.00;
ac_state.alt         3.700000E+04;
ac_state.mass        0.605320E+05;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.234072E+03;
ac_state.V_CAS       0.200000E+03;
ac_state.V_G.N       0.162000E+03;
ac_state.V_G.E       0.000000E+00;
ac_state.heading     0.392699E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       CR__ ;
#;
# B742;
# AIRCRAFT_STATE;
ac_state.lat         +040.39.00;
ac_state.long        -073.47.00;
ac_state.alt         3.700000E+04;
ac_state.mass        0.285320E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.241000E+03;
ac_state.V_CAS       0.241000E+03;
ac_state.V_G.N       0.162000E+03;
ac_state.V_G.E       0.000000E+00;
ac_state.heading     0.392699E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       CR__ ;
#;
# MD80;
# AIRCRAFT_STATE;
ac_state.lat         +040.39.00;
ac_state.long        -073.47.00;
ac_state.alt         3.700000E+04;
ac_state.mass        0.605320E+05;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.225327E+03;
ac_state.V_CAS       0.200000E+03;
ac_state.V_G.N       0.162000E+03;
ac_state.V_G.E       0.000000E+00;
ac_state.heading     0.392699E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       CR__ ;
#FI;
```

B.1.5 North Atlantic Crossing TMA Controller Logic

```
set conflict_radius 10 nm
get conflicts sector_id conflict_radius
s_while ((CURRENT_CONFLICT < NO_CONFLICTS) AND (NO_CONFLICTS > 0)) do
  increment current_conflict
  if (ACAC_DIST < 10 nm) then
     if (ac1_alt { ac2_alt) then
        command RESTRICT_UPPER_FL AC1 (ac1_alt - 1000 ft) (10nm / ac1_vel)
        command RESTRICT_LOWER_FL ac2 (ac2_alt + 1000 ft) (10nm / ac2_vel)
     else
        command RESTRICT_LOWER_FL AC1 (ac1_alt + 1000 ft) (10nm / ac1_vel)
        command RESTRICT_UPPER_FL ac2 (ac2_alt - 1000 ft) (10nm / ac2_vel)
     end
  else
    if (scenario = 0) then ; overtake
     if (ac2_vel - ac1_vel > 100) then
         command RESTRICT_UPPER_FL ac2 (ac2_alt - 1000 ft) (30nm / ac2_vel)
     else
        command increase_speed ac1 ((ac1_vel + ac2_vel)*1.5) 900
        command restrict_speed ac2 ((ac1_vel + ac2_vel)*1.5) 900
     end
    end
    if (scenario = 1) OR (scenario = 2) OR (scenario = 3) then ; acute, right, obtuse
       command restrict_upper_fl ac1 (ac1_alt - 1000 ft) (30nm / ac1_vel)
    end
    if (scenario = 4) then ; head on
      if (MAX_ALT_AC1 } MAX_ALT_AC2) then
        command RESTRICT_LOWER_FL AC1 (ac1_alt + 1000 ft) (30 nm / (ac1_vel + ac2_vel))
        command RESTRICT_UPPER_FL ac2 (ac2_alt - 1000 ft) (30 nm / (ac1_vel + ac2_vel))
      else
        command RESTRICT_LOWER_FL AC2 (ac2_alt + 1000 ft) (30 nm / (ac1_vel + ac2_vel))
        command RESTRICT_UPPER_FL ac1 (ac1_alt - 1000 ft) (30 nm / (ac1_vel + ac2_vel))
      end
    end
  end
end
stop
```

B.2 Landing at Gatwick

B.2.1 Landing at Gatwick Flight Plans

```
# Number of Segments;
0.800E01;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
# GURLU-BEXIL-TIMBA-MAYFIELD-MIDHURST-GATWICK ;
LONG    CRUISE   CRUISE__   CR__   3.0000000E04   000    +050.28.40 +000.58.07  00:00:00;
LONG    DESC__   DESCENT_   APP_   2.5000000E04   000    +050.42.32 +000.44.13  00:00:00;
LONG    DESC__   DESCENT_   APP_   1.8000000E04   000    +050.56.42 +000.15.42  00:00:00;
LONG    DESC__   DESCENT_   APP_   1.5900000E04   000    +051.01.02 +000.06.58  00:00:00;
LONG    DESC__   DESCENT_   LND_   7.0000000E03   000    +051.03.14 -000.37.30  00:00:00;
HDG_    DESC__   DESCENT_   LND_   7.0000000E03   070    +051.08.53 -000.37.25  00:00:00;
LONG    DESC__   DESCENT_   LND_   2.6000000E02   000    +051.08.53 -000.11.25  00:00:00;
LONG    DESC__   DESCENT_   LND_   2.6000000E02   000    +051.08.53 -000.01.25  00:00:00;
#FI;
```

Figure 79: EGKK A320 TIMBA 2B Flight Plan

```
# Number of Segments;
0.800E01;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
# GURLU-BEXIL-TIMBA-MAYFIELD-MIDHURST-GATWICK ;
LONG    CRUISE   CRUISE__   CR__   3.0000000E04   000    +050.28.40 +000.58.07  00:00:00;
LONG    DESC__   DESCENT_   APP_   2.5000000E04   000    +050.42.32 +000.44.13  00:00:00;
LONG    DESC__   DESCENT_   APP_   1.8000000E04   000    +050.56.42 +000.15.42  00:00:00;
LONG    DESC__   DESCENT_   APP_   1.5900000E04   000    +051.01.02 +000.06.58  00:00:00;
LONG    DESC__   DESCENT_   LND_   7.0000000E03   000    +051.03.14 -000.37.30  00:00:00;
HDG_    DESC__   DESCENT_   LND_   7.0000000E03   070    +051.08.53 -000.37.25  00:00:00;
LONG    DESC__   DESCENT_   LND_   2.6000000E02   000    +051.08.53 -000.11.25  00:00:00;
LONG    DESC__   DESCENT_   LND_   2.6000000E02   000    +051.08.53 -000.01.25  00:00:00;
#FI;
```

Figure 80: EGKK B737-300 TIMBA 2B Flight Plan

```
# Number of Segments;
0.800E01;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
# GURLU-BEXIL-TIMBA-MAYFIELD-MIDHURST-GATWICK ;
LONG    CRUISE   CRUISE__   CR__   3.0000000E04   000    +050.28.40 +000.58.07  00:00:00;
LONG    DESC__   DESCENT_   APP_   2.5000000E04   000    +050.42.32 +000.44.13  00:00:00;
LONG    DESC__   DESCENT_   APP_   1.8000000E04   000    +050.56.42 +000.15.42  00:00:00;
LONG    DESC__   DESCENT_   APP_   1.5900000E04   000    +051.01.02 +000.06.58  00:00:00;
LONG    DESC__   DESCENT_   LND_   7.0000000E03   000    +051.03.14 -000.37.30  00:00:00;
HDG_    DESC__   DESCENT_   LND_   7.0000000E03   070    +051.08.53 -000.37.25  00:00:00;
LONG    DESC__   DESCENT_   LND_   2.6000000E02   000    +051.08.53 -000.11.25  00:00:00;
LONG    DESC__   DESCENT_   LND_   2.6000000E02   000    +051.08.53 -000.01.25  00:00:00;
#FI;
```

Figure 81: EGKK B737-800 TIMBA 2B Flight Plan

```
# Number of Segments;
0.800E01;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
# GURLU-BEXIL-TIMBA-MAYFIELD-MIDHURST-GATWICK ;
LONG    CRUISE   CRUISE__   CR__   3.0000000E04   000    +050.28.40 +000.58.07  00:00:00;
LONG    DESC__   DESCENT_   APP_   2.5000000E04   000    +050.42.32 +000.44.13  00:00:00;
LONG    DESC__   DESCENT_   APP_   1.8000000E04   000    +050.56.42 +000.15.42  00:00:00;
LONG    DESC__   DESCENT_   APP_   1.5900000E04   000    +051.01.02 +000.06.58  00:00:00;
LONG    DESC__   DESCENT_   LND_   7.0000000E03   000    +051.03.14 -000.37.30  00:00:00;
HDG_    DESC__   DESCENT_   LND_   7.0000000E03   070    +051.08.53 -000.37.25  00:00:00;
LONG    DESC__   DESCENT_   LND_   2.6000000E02   000    +051.08.53 -000.11.25  00:00:00;
LONG    DESC__   DESCENT_   LND_   2.6000000E02   000    +051.08.53 -000.01.25  00:00:00;
#FI;
```

Figure 82: EGKK B747-200 TIMBA 2B Flight Plan

```
# Number of Segments;
0.800E01;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode      Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
# GURLU-BEXIL-TIMBA-MAYFIELD-MIDHURST-GATWICK ;
LONG     CRUISE   CRUISE__   CR__   3.0000000E04   000    +050.28.40 +000.58.07  00:00:00;
LONG     DESC__   DESCENT_   APP_   2.5000000E04   000    +050.42.32 +000.44.13  00:00:00;
LONG     DESC__   DESCENT_   APP_   1.8000000E04   000    +050.56.42 +000.15.42  00:00:00;
LONG     DESC__   DESCENT_   APP_   1.5900000E04   000    +051.01.02 +000.06.58  00:00:00;
LONG     DESC__   DESCENT_   LND_   7.0000000E03   000    +051.03.14 -000.37.30  00:00:00;
HDG_     DESC__   DESCENT_   LND_   7.0000000E03   070    +051.08.53 -000.37.25  00:00:00;
LONG     DESC__   DESCENT_   LND_   2.6000000E02   000    +051.08.53 -000.11.25  00:00:00;
LONG     DESC__   DESCENT_   LND_   2.6000000E02   000    +051.08.53 -000.01.25  00:00:00;
#FI;
```

Figure 83: MD-80 TIMBA 2B Flight Plan

```
%#;
# Number of Segments;
1.000E01;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords  Elapsed;
# Switch    Mode      Mode       Phase     Altitude    Heading   Lat       Long     Time;
#;
# FINISH ;
LONG    CRUISE   CRUISE__   CR__   3.1000000E04   000    +051.42.47 +001.04.58  00:00:00;
LONG    DESC__   DESCENT_   APP_   3.0000000E04   000    +051.34.34 +000.42.01  00:00:00;
LONG    DESC__   DESCENT_   APP_   2.5500000E04   000    +051.18.14 +000.35.50  00:00:00;
LONG    DESC__   DESCENT_   APP_   1.8500000E04   000    +050.54.42 +000.26.48  00:00:00;
LONG    DESC__   DESCENT_   APP_   1.8000000E04   000    +050.56.42 +000.15.42  00:00:00;
LONG    DESC__   DESCENT_   APP_   1.5900000E04   000    +051.01.02 +000.06.58  00:00:00;
LONG    DESC__   DESCENT_   LND_   7.0000000E03   000    +051.03.14 -000.37.30  00:00:00;
HDG_    DESC__   DESCENT_   LND_   7.0000000E03   070    +051.08.53 -000.37.25  00:00:00;
LONG    DESC__   DESCENT_   LND_   2.6000000E02   080    +051.08.53 -000.11.25  00:00:00;
LONG    DESC__   DESCENT_   LND_   2.6000000E02   000    +051.08.53 -000.01.25  00:00:00;
#FI;
```

Figure 84: A320 TIMBA 1H Flight Plan

```
%#;
# Number of Segments;
1.000E01;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords  Elapsed;
# Switch     Mode      Mode        Phase      Altitude    Heading    Lat        Long      Time;
#;
# FINISH ;
LONG    CRUISE    CRUISE__    CR__    3.1000000E04    000     +051.42.47 +001.04.58  00:00:00;
LONG    DESC__    DESCENT_    APP_    3.0000000E04    000     +051.34.34 +000.42.01  00:00:00;
LONG    DESC__    DESCENT_    APP_    2.5500000E04    000     +051.18.14 +000.35.50  00:00:00;
LONG    DESC__    DESCENT_    APP_    1.8500000E04    000     +050.54.42 +000.26.48  00:00:00;
LONG    DESC__    DESCENT_    APP_    1.8000000E04    000     +050.56.42 +000.15.42  00:00:00;
LONG    DESC__    DESCENT_    APP_    1.5900000E04    000     +051.01.02 +000.06.58  00:00:00;
LONG    DESC__    DESCENT_    LND_    7.0000000E03    000     +051.03.14 -000.37.30  00:00:00;
HDG_    DESC__    DESCENT_    LND_    7.0000000E03    070     +051.08.53 -000.37.25  00:00:00;
LONG    DESC__    DESCENT_    LND_    2.6000000E02    080     +051.08.53 -000.11.25  00:00:00;
LONG    DESC__    DESCENT_    LND_    2.6000000E02    000     +051.08.53 -000.01.25  00:00:00;
#FI;
```

Figure 85: B373-300 TIMBA 1H Flight Plan

```
%#;
# Number of Segments;
1.000E01;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords  Elapsed;
# Switch     Mode      Mode        Phase      Altitude    Heading    Lat        Long      Time;
#;
# FINISH ;
LONG    CRUISE    CRUISE__    CR__    3.1000000E04    000     +051.42.47 +001.04.58  00:00:00;
LONG    DESC__    DESCENT_    APP_    3.0000000E04    000     +051.34.34 +000.42.01  00:00:00;
LONG    DESC__    DESCENT_    APP_    2.5500000E04    000     +051.18.14 +000.35.50  00:00:00;
LONG    DESC__    DESCENT_    APP_    1.8500000E04    000     +050.54.42 +000.26.48  00:00:00;
LONG    DESC__    DESCENT_    APP_    1.8000000E04    000     +050.56.42 +000.15.42  00:00:00;
LONG    DESC__    DESCENT_    APP_    1.5900000E04    000     +051.01.02 +000.06.58  00:00:00;
LONG    DESC__    DESCENT_    LND_    7.0000000E03    000     +051.03.14 -000.37.30  00:00:00;
HDG_    DESC__    DESCENT_    LND_    7.0000000E03    070     +051.08.53 -000.37.25  00:00:00;
LONG    DESC__    DESCENT_    LND_    2.6000000E02    080     +051.08.53 -000.11.25  00:00:00;
LONG    DESC__    DESCENT_    LND_    2.6000000E02    000     +051.08.53 -000.01.25  00:00:00;
#FI;
```

Figure 86: B737-800 TIMBA 1H Flight Plan

```
%#;
# Number of Segments;
1.000E01;
#;
#        PROCEDURE  THRUST                                      Waypoint Coords  Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long     Time;
#;
# FINISH ;
LONG   CRUISE   CRUISE__   CR__   3.1000000E04   000    +051.42.47 +001.04.58  00:00:00;
LONG   DESC__   DESCENT_   APP_   3.0000000E04   000    +051.34.34 +000.42.01  00:00:00;
LONG   DESC__   DESCENT_   APP_   2.5500000E04   000    +051.18.14 +000.35.50  00:00:00;
LONG   DESC__   DESCENT_   APP_   1.8500000E04   000    +050.54.42 +000.26.48  00:00:00;
LONG   DESC__   DESCENT_   APP_   1.8000000E04   000    +050.56.42 +000.15.42  00:00:00;
LONG   DESC__   DESCENT_   APP_   1.5900000E04   000    +051.01.02 +000.06.58  00:00:00;
LONG   DESC__   DESCENT_   LND_   7.0000000E03   000    +051.03.14 -000.37.30  00:00:00;
HDG_   DESC__   DESCENT_   LND_   7.0000000E03   070    +051.08.53 -000.37.25  00:00:00;
LONG   DESC__   DESCENT_   LND_   2.6000000E02   080    +051.08.53 -000.11.25  00:00:00;
LONG   DESC__   DESCENT_   LND_   2.6000000E02   000    +051.08.53 -000.01.25  00:00:00;
#FI;
```

Figure 87: B747-200 TIMBA 1H Flight Plan

```
%#;
# Number of Segments;
1.000E01;
#;
#        PROCEDURE  THRUST                                      Waypoint Coords  Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long     Time;
#;
# FINISH ;
LONG   CRUISE   CRUISE__   CR__   3.1000000E04   000    +051.42.47 +001.04.58  00:00:00;
LONG   DESC__   DESCENT_   APP_   3.0000000E04   000    +051.34.34 +000.42.01  00:00:00;
LONG   DESC__   DESCENT_   APP_   2.5500000E04   000    +051.18.14 +000.35.50  00:00:00;
LONG   DESC__   DESCENT_   APP_   1.8500000E04   000    +050.54.42 +000.26.48  00:00:00;
LONG   DESC__   DESCENT_   APP_   1.8000000E04   000    +050.56.42 +000.15.42  00:00:00;
LONG   DESC__   DESCENT_   APP_   1.5900000E04   000    +051.01.02 +000.06.58  00:00:00;
LONG   DESC__   DESCENT_   LND_   7.0000000E03   000    +051.03.14 -000.37.30  00:00:00;
HDG_   DESC__   DESCENT_   LND_   7.0000000E03   070    +051.08.53 -000.37.25  00:00:00;
LONG   DESC__   DESCENT_   LND_   2.6000000E02   080    +051.08.53 -000.11.25  00:00:00;
LONG   DESC__   DESCENT_   LND_   2.6000000E02   000    +051.08.53 -000.01.25  00:00:00;
#FI;
```

Figure 88: MD-80 TIMBA 1H Flight Plan

B.2.2 Landing at Gatwick ATC Plan

```
no_sectors  4;
no_airports 1;
no_stacks   3;
#;
# SectorID MinLat MaxLat   MinLon      MaxLon  Neighbors (N,E,S,W)  Script Logic;
#;
17324 5.00000E1 5.14000E1 -2.80000E0 0.00000E0  17330  0     0     20311   "blank.scr";
20311 5.00000E1 5.14000E1 0.00000E0  2.80000E0  20317  0     0     17324   "blank.scr";
17330 5.14000E1 5.43000E1 -2.80000E0 0.00000E0  0     20317 17324 0       "blank.scr";
20317 5.14000E1 5.43000E1 0.00000E0  2.80000E0  0     0     20311 17330   "blank.scr";
#;
# AirportID        Lat       Lon     Alt(ft)  Orientation(deg)   SectorID  Script Logic;
#;
   EGKK      051:08:35N  000:11:25W   16            245          17324  "EGKK_tester.scr";
#;
# StackID    Lat         Lon     AltMax(ft)  AltMin(ft) Orient(deg) AirportID St Prev Stacks;
#;;
MAYFLD    051:01:02N  000:06:58E    8000       5000         90        EGKK;
TIMBA     050:56:42N  000:15:42E   23000      17000        310        EGKK    1  BEXIL EMPTY;
BEXIL     050:42:32N  000:44:13E   19000      14000        310        EGKK    2;
#;
#FI;
```

B.2.3 Landing at Gatwick Initial Conditions

TIMBA 2B

```
# Initial Condition File;
#;
# CONFIGURATION;
no_ac                0.500000E+01;
# TIME;
time_step            0.100000E+01;
start                0.100000E+01;
finish               0.108000E+05;
# DESTINATION;
ac_state.destination EGKK;
# A320;
# AIRCRAFT_STATE;
ac_state.lat         +050.10.00;
ac_state.long        +001.30.00;
ac_state.alt         3.000000E+04;
ac_state.mass        0.136687E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.450000E+03;
ac_state.V_CAS       0.400000E+03;
ac_state.V_G.N       0.400000E+03;
ac_state.V_G.E       0.000000E+00;
ac_state.heading     5.582782E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       CR__ ;
#;
# B733;
# AIRCRAFT_STATE;
ac_state.lat         +050.10.00;
ac_state.long        +001.30.00;
ac_state.alt         3.000000E+04;
ac_state.mass        0.136687E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.450000E+03;
ac_state.V_CAS       0.400000E+03;
ac_state.V_G.N       0.400000E+03;
ac_state.V_G.E       0.000000E+00;
ac_state.heading     5.582782E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       CR__ ;
#;
# B738;
# AIRCRAFT_STATE;
ac_state.lat         +050.10.00;
ac_state.long        +001.30.00;
ac_state.alt         3.000000E+04;
ac_state.mass        0.136687E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.450000E+03;
ac_state.V_CAS       0.400000E+03;
ac_state.V_G.N       0.400000E+03;
ac_state.V_G.E       0.000000E+00;
ac_state.heading     5.582782E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       CR__ ;
#;
# B742;
# AIRCRAFT_STATE;
ac_state.lat         +050.10.00;
ac_state.long        +001.30.00;
ac_state.alt         3.000000E+04;
ac_state.mass        0.629023E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.450000E+03;
ac_state.V_CAS       0.400000E+03;
ac_state.V_G.N       0.400000E+03;
ac_state.V_G.E       0.000000E+00;
ac_state.heading     5.582782E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       CR__ ;
#;
# MD80;
# AIRCRAFT_STATE;
ac_state.lat         +050.10.00;
ac_state.long        +001.30.00;
ac_state.alt         3.000000E+04;
ac_state.mass        0.136687E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.450000E+03;
ac_state.V_CAS       0.400000E+03;
ac_state.V_G.N       0.400000E+03;
ac_state.V_G.E       0.000000E+00;
ac_state.heading     5.582782E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       CR__ ;
#FI;
```

TIMBA 1H

```
# Initial Condition File;
#;
# CONFIGURATION;
no_ac                 0.500000E+01;
# TIME;
time_step             0.100000E+01;
start                 0.100000E+01;
finish                0.108000E+05;
# DESTINATION;
ac_state.destination EGKK;
# A320;
# AIRCRAFT_STATE;
ac_state.lat          +052.04.47;
ac_state.long         +001.50.58;
ac_state.alt          3.100000E+04;
ac_state.mass         0.136687E+06;
ac_state.thrust       0.200000E+06;
ac_state.V_TAS        0.450000E+03;
ac_state.V_CAS        0.400000E+03;
ac_state.V_G.N        0.400000E+03;
ac_state.V_G.E        0.000000E+00;
ac_state.heading      5.582782E+00;
ac_state.ROCD         0.000000E+00;
ac_state.fp_seg       0.100000E+01;
ac_state.phase        CR__ ;
#;
# B733;
# AIRCRAFT_STATE;
ac_state.lat          +052.04.47;
ac_state.long         +001.50.58;
ac_state.alt          3.100000E+04;
ac_state.mass         0.136687E+06;
ac_state.thrust       0.200000E+06;
ac_state.V_TAS        0.450000E+03;
ac_state.V_CAS        0.400000E+03;
ac_state.V_G.N        0.400000E+03;
ac_state.V_G.E        0.000000E+00;
ac_state.heading      5.582782E+00;
ac_state.ROCD         0.000000E+00;
ac_state.fp_seg       0.100000E+01;
ac_state.phase        CR__ ;
#;
# B738;
# AIRCRAFT_STATE;
ac_state.lat          +052.04.47;
ac_state.long         +001.50.58;
ac_state.alt          3.100000E+04;
ac_state.mass         0.136687E+06;
ac_state.thrust       0.200000E+06;
ac_state.V_TAS        0.450000E+03;
ac_state.V_CAS        0.400000E+03;
ac_state.V_G.N        0.400000E+03;
ac_state.V_G.E        0.000000E+00;
ac_state.heading      5.582782E+00;
ac_state.ROCD         0.000000E+00;
ac_state.fp_seg       0.100000E+01;
ac_state.phase        CR__ ;
#;
# B742;
# AIRCRAFT_STATE;
ac_state.lat          +052.04.47;
ac_state.long         +001.50.58;
ac_state.alt          3.100000E+04;
ac_state.mass         0.629023E+06;
ac_state.thrust       0.200000E+06;
ac_state.V_TAS        0.450000E+03;
ac_state.V_CAS        0.400000E+03;
ac_state.V_G.N        0.400000E+03;
ac_state.V_G.E        0.000000E+00;
ac_state.heading      5.582782E+00;
ac_state.ROCD         0.000000E+00;
ac_state.fp_seg       0.100000E+01;
ac_state.phase        CR__ ;
#;
# MD80;
# AIRCRAFT_STATE;
ac_state.lat          +052.04.47;
ac_state.long         +001.50.58;
ac_state.alt          3.100000E+04;
ac_state.mass         0.136687E+06;
ac_state.thrust       0.200000E+06;
ac_state.V_TAS        0.450000E+03;
ac_state.V_CAS        0.400000E+03;
ac_state.V_G.N        0.400000E+03;
ac_state.V_G.E        0.000000E+00;
ac_state.heading      5.582782E+00;
ac_state.ROCD         0.000000E+00;
ac_state.fp_seg       0.100000E+01;
ac_state.phase        CR__ ;
#FI;
```

B.2.4 TMA Controller Logic: EGKK tester

```
;---------------put ac into comp_ar------------
; get the first ac with wp DETLING put it in comp_ar
GET FIRST_AC 51 18 14 0 35 50;
;---------------get the estimated time of arrival----------
SET CURRENT_COMP 1
if COMP_AC } 0 then
   GET TIME_EST COMP_AC 50 56 42 0 15 42 18000 ft
   INCREMENT CURRENT_COMP
end
; get the first ac with wp BEXIL put it in comp_ar
GET FIRST_AC 50 42 32 0 44 13;
;---------------get the estimated time of arrival----------
if COMP_AC } 0 then
   GET TIME_EST COMP_AC 50 56 42 0 15 42 18000 ft
   INCREMENT CURRENT_COMP
end ;----------what to do if there are 2 ac to be compared
with---- if (NO_COMP_AC > 1) THEN ;---compile the time estimates--
SET CURRENT_COMP 1

;........what to do if there are no ac in the stack..........
   if NO_CURRENT_STACK_AC = 0 then
      get ordered_comp_ar

      set TEMP_VALUE COMP_AC_EST
      SET CURRENT_COMP 2
      GET SEP_REQUIRED COMP0_AC COMP_AC 18000 ft
      if (COMP_AC_SEP > ((COMP_AC_EST - TEMP_VALUE)-72)) then
         SET CURRENT_STACK TIMBA
         COMMAND TO_STACK COMP_AC CURRENT_STACK 16000 ft
         GET CLEAR_COMP_AR COMP0_AC
      else
         GET CLEAR_COMP_AR COMP0_AC
      end

   ;.........what to do if there are ac in the stack.......
   else
      GET LONGEST_ON_STACK CURRENT_STACK
      GET TIME_EST SELECTED_AC 50 56 42 0 15 42 18000 ft
      ; should automatically create another entry in comp_ar
      GET LAST_ON_FINAL
      ; put's last on final in FINAL_AC
;
      ;check if the stack ac can be cleared
      SET CURRENT_COMP 3
      if FINAL_AC { 0 then
        SET CURRENT_STACK TIMBA
        COMMAND LEAVE_STACK SELECTED_AC CURRENT_STACK
        GET CLEAR_COMP_AR COMP_AC
      else
        GET SEP_REQUIRED FINAL_AC COMP_AC 18000 ft
        if (0 { (FINAL_AC_TIMEOUT + COMP_AC_EST - TIME - COMP_AC_SEP - 0)) then
          COMMAND LEAVE_STACK SELECTED_AC CURRENT_STACK
```

```
  ;set temporary value to the est time for the stack ac
  set TEMP_VALUE COMP_AC_EST
  ;
  ;check to see if any of the other two can be cleared
  SET CURRENT_COMP 2
  GET SEP_REQUIRED COMP2_AC COMP0_AC 18000 ft
  GET SEP_REQUIRED COMP2_AC COMP_AC 18000 ft
  ;     clear away the ac that was in the stack and order
  GET CLEAR_COMP_AR COMP2_AC
  GET ordered_comp_ar
  ;     get the sep required for the trailing ac
  SET CURRENT_COMP 2
  GET SEP_REQUIRED COMP0_AC COMP_AC 18000 ft
  ;     now all of the seps are in order
  SET CURRENT_COMP 1
  if ( COMP_AC_SEP < (TEMP_VALUE - COMP_AC_EST)) then
     ; clear it
     SET TEMP_VALUE2 COMP_AC_EST
     GET CLEAR_COMP_AR COMP_AC
     GET ordered_comp_ar
     if ( COMP_AC_SEP < (TEMP_VALUE2 - COMP_AC_EST)) then
        ; clear it
        GET CLEAR_COMP_AR COMP_AC
     else
        COMMAND TO_STACK COMP_AC CURRENT_STACK 16000 ft
     end
  else
     COMMAND TO_STACK COMP_AC CURRENT_STACK 16000 ft
     GET ordered_comp_ar
     if (COMP_AC_SEP < (TEMP_VALUE - COMP_AC_EST)) then
        ; clear it
        GET CLEAR_COMP_AR COMP_AC
     else
        COMMAND TO_STACK COMP_AC CURRENT_STACK 16000 ft
     end
  end
else
;ac in stack could not be used
  ;check to see if any of the other two can be cleared
  SET CURRENT_COMP 3
  ;     clear away the ac that was in the stack and order
  GET CLEAR_COMP_AR COMP_AC
  GET ordered_comp_ar
  ;     get the sep required for the leading ac
  SET CURRENT_COMP 1
  GET SEP_REQUIRED FINAL_AC COMP_AC 18000 ft
  ;     get the sep required for the trailing ac
  SET CURRENT_COMP 2
  GET SEP_REQUIRED COMP0_AC COMP_AC 18000 ft
  ;     now all of the seps are in order
  SET CURRENT_COMP 1
  if (COMP_AC_SEP < (FINAL_AC_TIMEOUT + COMP_AC_EST - TIME)) then
     ; clear it
     SET TEMP_VALUE2 COMP_AC_EST
```

```
               GET CLEAR_COMP_AR COMP_AC
               GET ordered_comp_ar
               if (COMP_AC_SEP < (TEMP_VALUE2 + COMP_AC_EST - TIME)) then
                  ; clear it
                  GET CLEAR_COMP_AR COMP_AC
               else
                  COMMAND TO_STACK COMP_AC CURRENT_STACK 16000 ft
               end
            else
               COMMAND TO_STACK COMP_AC CURRENT_STACK 16000 ft
               INCREMENT CURRENT_COMP
               GET SEP_REQUIRED FINAL_AC COMP_AC 18000 ft
               if (COMP_AC_SEP < (FINAL_AC_TIMEOUT + COMP_AC_EST - TIME)) then
                  ; clear it
                  GET CLEAR_COMP_AR COMP_AC
               else
                  COMMAND TO_STACK COMP_AC CURRENT_STACK 16000 ft
               end
            end
         end
      end
   end
   ;.........end of what to do if there are ac in the stack.......
   ;.........what to do if there is only one ac in the comp_ar.......
else
   SET CURRENT_STACK TIMBA
   if (NO_COMP_AC = 0) and (NO_CURRENT_STACK_AC > 0) THEN
      GET LONGEST_ON_STACK CURRENT_STACK
      GET TIME_EST SELECTED_AC 50 56 42 0 15 42 18000 ft
      ; should automatically create another entry in comp_ar
      GET LAST_ON_FINAL
      ; put's last on final in FINAL_AC
;
      ;check if the stack ac can be cleared
      SET CURRENT_COMP 3
      if FINAL_AC < 0 then
        SET CURRENT_STACK TIMBA
        COMMAND LEAVE_STACK SELECTED_AC CURRENT_STACK
        GET CLEAR_COMP_AR COMP_AC
      else
         GET SEP_REQUIRED FINAL_AC COMP_AC 18000 ft
         if (0 { (FINAL_AC_TIMEOUT + COMP_AC_EST - TIME - COMP_AC_SEP - 0))
;         and ( 120 }(COMP_AC_EST - TIME - COMP_AC_SEP))
         then
           COMMAND LEAVE_STACK SELECTED_AC CURRENT_STACK
           GET CLEAR_COMP_AR COMP_AC
         end
      end
   end
end stop
```

B.3 Core Europe

B.3.1 Flight Plans

Frankfurt, Germany - Amsterdam, The Netherlands

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +051.20.00 +007.11.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +051.20.00 +007.11.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +051.20.00 +007.11.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +052.18.31 +004.45.50  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +052.18.31 +004.45.50  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +052.19.31 +004.45.50  00:00:00;
#FI;
```

Figure 89: A320 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +051.20.00 +007.11.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +051.20.00 +007.11.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +051.20.00 +007.11.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +052.18.31 +004.45.50  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +052.18.31 +004.45.50  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +052.19.31 +004.45.50  00:00:00;
#FI;
```

Figure 90: B737-300 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +051.20.00 +007.11.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +051.20.00 +007.11.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +051.20.00 +007.11.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +052.18.31 +004.45.50  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +052.18.31 +004.45.50  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +052.19.31 +004.45.50  00:00:00;
#FI;
```

Figure 91: B737-800 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +051.20.00 +007.11.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +051.20.00 +007.11.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +051.20.00 +007.11.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +052.18.31 +004.45.50  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +052.18.31 +004.45.50  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +052.19.31 +004.45.50  00:00:00;
#FI;
```

Figure 92: B747-200 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +051.20.00 +007.11.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +051.20.00 +007.11.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +051.20.00 +007.11.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +052.18.31 +004.45.50  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +052.18.31 +004.45.50  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +052.19.31 +004.45.50  00:00:00;
#FI;
```

Figure 93: MD-80 Flight Plan

Hamburg, Germany - Paris, France

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +050.00.00 +003.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +050.00.00 +003.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +050.00.00 +003.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +049.00.46 +002.32.60  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.00.46 +002.32.60  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.00.46 +002.33.15  00:00:00;
#FI;
```

Figure 94: A320 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +050.00.00 +003.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +050.00.00 +003.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +050.00.00 +003.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +049.00.46 +002.32.60  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.00.46 +002.32.60  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.00.46 +002.33.15  00:00:00;
#FI;
```

Figure 95: B737-300 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +050.00.00 +003.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +050.00.00 +003.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +050.00.00 +003.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +049.00.46 +002.32.60  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.00.46 +002.32.60  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.00.46 +002.33.15  00:00:00;
#FI;
```

Figure 96: B737-800 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +050.00.00 +003.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +050.00.00 +003.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +050.00.00 +003.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +049.00.46 +002.32.60  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.00.46 +002.32.60  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.00.46 +002.33.15  00:00:00;
#FI;
```

Figure 97: B747-200 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +050.00.00 +003.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +050.00.00 +003.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +050.00.00 +003.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +049.00.46 +002.32.60  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.00.46 +002.32.60  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.00.46 +002.33.15  00:00:00;
#FI;
```

Figure 98: MD-80 Flight Plan

Munich, Germany - London, United Kingdom

```
# Number of Segments;
6.000E00;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode      Mode        Phase      Altitude    Heading    Lat        Long        Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +050.28.40 +000.58.07  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +050.28.40 +000.58.07  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +050.28.40 +000.58.07  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +051.08.53 -000.11.25  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +051.08.53 -000.11.25  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +051.08.53 -000.12.25  00:00:00;
#FI;
```

Figure 99: A320 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode      Mode        Phase      Altitude    Heading    Lat        Long        Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +050.28.40 +000.58.07  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +050.28.40 +000.58.07  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +050.28.40 +000.58.07  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +051.08.53 -000.11.25  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +051.08.53 -000.11.25  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +051.08.53 -000.12.25  00:00:00;
#FI;
```

Figure 100: B737-300 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +050.28.40 +000.58.07  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +050.28.40 +000.58.07  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +050.28.40 +000.58.07  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +051.08.53 -000.11.25  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +051.08.53 -000.11.25  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +051.08.53 -000.12.25  00:00:00;
#FI;
```

Figure 101: B737-800 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +050.28.40 +000.58.07  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +050.28.40 +000.58.07  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +050.28.40 +000.58.07  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +051.08.53 -000.11.25  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +051.08.53 -000.11.25  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +051.08.53 -000.12.25  00:00:00;
#FI;
```

Figure 102: B747-200 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +050.28.40 +000.58.07  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +050.28.40 +000.58.07  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +050.28.40 +000.58.07  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +051.08.53 -000.11.25  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +051.08.53 -000.11.25  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +051.08.53 -000.12.25  00:00:00;
#FI;
```

Figure 103: MD-80 Flight Plan

London, United Kingdom - Zurich, Switzerland

```
# Number of Segments;
6.000E00;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode      Mode       Phase      Altitude    Heading    Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +048.28.40 +007.58.07  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +048.28.40 +007.58.07  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +048.28.40 +007.58.07  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +047.27.53 +008.32.57  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +047.27.53 +008.32.57  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +047.26.53 +008.33.57  00:00:00;
#FI;
```

Figure 104: A320 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode      Mode       Phase      Altitude    Heading    Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +048.28.40 +007.58.07  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +048.28.40 +007.58.07  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +048.28.40 +007.58.07  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +047.27.53 +008.32.57  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +047.27.53 +008.32.57  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +047.26.53 +008.33.57  00:00:00;
#FI;
```

Figure 105: B737-300 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +048.28.40 +007.58.07  00:00:00;
ALT_  CLIMB_   CLIMB___    CL__    3.7000000E04    000    +048.28.40 +007.58.07  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +048.28.40 +007.58.07  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +047.27.53 +008.32.57  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +047.27.53 +008.32.57  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +047.26.53 +008.33.57  00:00:00;
#FI;
```

Figure 106: B737-800 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +048.28.40 +007.58.07  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +048.28.40 +007.58.07  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +048.28.40 +007.58.07  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +047.27.53 +008.32.57  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +047.27.53 +008.32.57  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +047.26.53 +008.33.57  00:00:00;
#FI;
```

Figure 107: B747-200 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +048.28.40 +007.58.07  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +048.28.40 +007.58.07  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +048.28.40 +007.58.07  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +047.27.53 +008.32.57  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +047.27.53 +008.32.57  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +047.26.53 +008.33.57  00:00:00;
#FI;
```

Figure 108: MD-80 Flight Plan

Amsterdam, The Netherlands - Madrid, Spain

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +041.30.00 -002.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +041.30.00 -002.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +041.30.00 -002.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +040.28.20 +003.33.39  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +040.28.20 +003.33.39  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.01.00 +002.34.15  00:00:00;
#FI;
```

Figure 109: A320 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +041.30.00 -002.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +041.30.00 -002.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +041.30.00 -002.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +040.28.20 +003.33.39  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +040.28.20 +003.33.39  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.01.00 +002.34.15  00:00:00;
#FI;
```

Figure 110: B737-300 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +041.30.00 -002.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +041.30.00 -002.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +041.30.00 -002.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +040.28.20 +003.33.39  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +040.28.20 +003.33.39  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.01.00 +002.34.15  00:00:00;
#FI;
```

Figure 111: B737-800 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +041.30.00 -002.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +041.30.00 -002.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +041.30.00 -002.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +040.28.20 +003.33.39  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +040.28.20 +003.33.39  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.01.00 +002.34.15  00:00:00;
#FI;
```

Figure 112: B747-200 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +041.30.00 -002.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +041.30.00 -002.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +041.30.00 -002.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +040.28.20 +003.33.39  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +040.28.20 +003.33.39  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +049.01.00 +002.34.15  00:00:00;
#FI;
```

Figure 113: MD-80 Flight Plan

Paris, France - Rome, Italy

```
# Number of Segments;
6.000E00;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode      Mode       Phase      Altitude    Heading    Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +043.30.00 +010.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +043.30.00 +010.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +043.30.00 +010.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +041.47.58 +012.35.42  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +041.47.58 +012.35.42  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +041.46.58 +012.35.42  00:00:00;
#FI;
```

Figure 114: A320 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode      Mode       Phase      Altitude    Heading    Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +043.30.00 +010.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +043.30.00 +010.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +043.30.00 +010.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +041.47.58 +012.35.42  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +041.47.58 +012.35.42  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +041.46.58 +012.35.42  00:00:00;
#FI;
```

Figure 115: B737-300 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +043.30.00 +010.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +043.30.00 +010.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +043.30.00 +010.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +041.47.58 +012.35.42  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +041.47.58 +012.35.42  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +041.46.58 +012.35.42  00:00:00;
#FI;
```

Figure 116: B737-800 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +043.30.00 +010.00.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +043.30.00 +010.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +043.30.00 +010.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +041.47.58 +012.35.42  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +041.47.58 +012.35.42  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +041.46.58 +012.35.42  00:00:00;
#FI;
```

Figure 117: B747-200 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +043.30.00 +010.00.00  00:00:00;
ALT_  CLIMB_   CLIMB___    CL__    3.7000000E04    000    +043.30.00 +010.00.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +043.30.00 +010.00.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +041.47.58 +012.35.42  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +041.47.58 +012.35.42  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +041.46.58 +012.35.42  00:00:00;
#FI;
```

Figure 118: MD-80 Flight Plan

Rome, Italy - Frankfurt, Germany

```
# Number of Segments;
6.000E00;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode      Mode        Phase      Altitude    Heading    Lat        Long        Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +049.20.00 +010.15.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +049.20.00 +010.15.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +049.20.00 +010.15.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +050.01.35 +008.32.35  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +050.01.35 +008.32.35  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +050.01.35 +008.31.35  00:00:00;
#FI;
```

Figure 119: A320 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode      Mode        Phase      Altitude    Heading    Lat        Long        Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +049.20.00 +010.15.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +049.20.00 +010.15.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +049.20.00 +010.15.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +050.01.35 +008.32.35  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +050.01.35 +008.32.35  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +050.01.35 +008.31.35  00:00:00;
#FI;
```

Figure 120: B737-300 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +049.20.00 +010.15.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +049.20.00 +010.15.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +049.20.00 +010.15.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +050.01.35 +008.32.35  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +050.01.35 +008.32.35  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +050.01.35 +008.31.35  00:00:00;
#FI;
```

Figure 121: B737-800 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +049.20.00 +010.15.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +049.20.00 +010.15.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +049.20.00 +010.15.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +050.01.35 +008.32.35  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +050.01.35 +008.32.35  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +050.01.35 +008.31.35  00:00:00;
#FI;
```

Figure 122: B747-200 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +049.20.00 +010.15.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +049.20.00 +010.15.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +049.20.00 +010.15.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +050.01.35 +008.32.35  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +050.01.35 +008.32.35  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +050.01.35 +008.31.35  00:00:00;
#FI;
```

Figure 123: MD-80 Flight Plan

Madrid, Spain - Munich, Germany

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +047.30.00 +010.30.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +047.30.00 +010.30.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +047.30.00 +010.30.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +048.21.14 +011.47.10  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +048.21.14 +011.47.10  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +048.21.14 +011.48.10  00:00:00;
#FI;
```

Figure 124: A320 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +047.30.00 +010.30.00  00:00:00;
ALT_  CLIMB_   CLIMB___    CL__    3.7000000E04    000    +047.30.00 +010.30.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +047.30.00 +010.30.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +048.21.14 +011.47.10  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +048.21.14 +011.47.10  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +048.21.14 +011.48.10  00:00:00;
#FI;
```

Figure 125: B737-300 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +047.30.00 +010.30.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +047.30.00 +010.30.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +047.30.00 +010.30.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +048.21.14 +011.47.10  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +048.21.14 +011.47.10  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +048.21.14 +011.48.10  00:00:00;
#FI;
```

Figure 126: B737-800 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +047.30.00 +010.30.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +047.30.00 +010.30.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +047.30.00 +010.30.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +048.21.14 +011.47.10  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +048.21.14 +011.47.10  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +048.21.14 +011.48.10  00:00:00;
#FI;
```

Figure 127: B747-200 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +047.30.00 +010.30.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +047.30.00 +010.30.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +047.30.00 +010.30.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +048.21.14 +011.47.10  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +048.21.14 +011.47.10  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +048.21.14 +011.48.10  00:00:00;
#FI;
```

Figure 128: MD-80 Flight Plan

Zurich, Switzerland - Hamburg, Germany

```
# Number of Segments;
6.000E00;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode      Mode        Phase      Altitude    Heading    Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_     3.1000000E04    000     +052.14.00 +009.58.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__     3.7000000E04    000     +052.14.00 +009.58.00  00:00:00;
LONG CRUISE   CRUISE__    CR__     3.7000000E04    000     +052.14.00 +009.58.00  00:00:00;
LONG DESC__   DESCENT_    APP_     3.6400000E02    000     +053.37.49 +009.59.18  00:00:00;
LONG DESC__   DESCENT_    LND_     3.6400000E02    000     +053.37.49 +009.59.18  00:00:00;
LAT_ DESC__   DESCENT_    LND_     3.6400000E02    000     +053.38.00 +009.59.58  00:00:00;
#FI;
```

Figure 129: A320 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#         PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode      Mode        Phase      Altitude    Heading    Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_     3.1000000E04    000     +052.14.00 +009.58.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__     3.7000000E04    000     +052.14.00 +009.58.00  00:00:00;
LONG CRUISE   CRUISE__    CR__     3.7000000E04    000     +052.14.00 +009.58.00  00:00:00;
LONG DESC__   DESCENT_    APP_     3.6400000E02    000     +053.37.49 +009.59.18  00:00:00;
LONG DESC__   DESCENT_    LND_     3.6400000E02    000     +053.37.49 +009.59.18  00:00:00;
LAT_ DESC__   DESCENT_    LND_     3.6400000E02    000     +053.38.00 +009.59.58  00:00:00;
#FI;
```

Figure 130: B737-300 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +052.14.00 +009.58.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +052.14.00 +009.58.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +052.14.00 +009.58.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +053.37.49 +009.59.18  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +053.37.49 +009.59.18  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +053.38.00 +009.59.58  00:00:00;
#FI;
```

Figure 131: B737-800 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +052.14.00 +009.58.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +052.14.00 +009.58.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +052.14.00 +009.58.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +053.37.49 +009.59.18  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +053.37.49 +009.59.18  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +053.38.00 +009.59.58  00:00:00;
#FI;
```

Figure 132: B747-200 Flight Plan

```
# Number of Segments;
6.000E00;
#;
#        PROCEDURE  THRUST                                        Waypoint Coords    Elapsed;
# Switch    Mode     Mode       Phase     Altitude    Heading   Lat       Long       Time;
#;
ALT_ CLIMB_   TAKEOFF_    TOF_    3.1000000E04    000    +052.14.00 +009.58.00  00:00:00;
ALT_ CLIMB_   CLIMB___    CL__    3.7000000E04    000    +052.14.00 +009.58.00  00:00:00;
LONG CRUISE   CRUISE__    CR__    3.7000000E04    000    +052.14.00 +009.58.00  00:00:00;
LONG DESC__   DESCENT_    APP_    3.6400000E02    000    +053.37.49 +009.59.18  00:00:00;
LONG DESC__   DESCENT_    LND_    3.6400000E02    000    +053.37.49 +009.59.18  00:00:00;
LAT_ DESC__   DESCENT_    LND_    3.6400000E02    000    +053.38.00 +009.59.58  00:00:00;
#FI;
```

Figure 133: MD-80 Flight Plan

B.3.2 Initial Conditions

Frankfurt, Germany - Amsterdam, The Netherlands

```
# Initial Condition File;
#;
# CONFIGURATION;
no_ac                 0.500000E+01;
# TIME;
time_step             0.100000E+01;
start                 0.100000E+01;
finish                2.160000E+04;
# DESTINATION;
ac_state.destination EHAM;
# A320;
# AIRCRAFT_STATE;
ac_state.lat          +050.01.35;
ac_state.long         +008.32.35;
ac_state.alt          0.364000E+03;
ac_state.mass         0.136687E+06;
ac_state.thrust       0.200000E+06;
ac_state.V_TAS        0.150000E+03;
ac_state.V_CAS        0.150000E+03;
ac_state.V_G.N        0.750000E+02;
ac_state.V_G.E        0.750000E+02;
ac_state.heading      5.236000E+00;
ac_state.ROCD         0.000000E+00;
ac_state.fp_seg       0.100000E+01;
ac_state.phase        TOF_ ;
#;
# B733;
# AIRCRAFT_STATE;
ac_state.lat          +050.01.35;
ac_state.long         +008.32.35;
ac_state.alt          0.364000E+03;
ac_state.mass         0.119050E+06;
ac_state.thrust       0.200000E+06;
ac_state.V_TAS        0.150000E+03;
ac_state.V_CAS        0.150000E+03;
ac_state.V_G.N        0.750000E+02;
ac_state.V_G.E        0.750000E+02;
ac_state.heading      5.236000E+00;
ac_state.ROCD         0.000000E+00;
ac_state.fp_seg       0.100000E+01;
ac_state.phase        TOF_ ;
#;
# B738;
# AIRCRAFT_STATE;
ac_state.lat          +050.01.35;
ac_state.long         +008.32.35;
ac_state.alt          0.364000E+03;
ac_state.mass         0.136907E+06;
ac_state.thrust       0.200000E+06;
ac_state.V_TAS        0.150000E+03;
ac_state.V_CAS        0.150000E+03;
ac_state.V_G.N        0.750000E+02;
ac_state.V_G.E        0.750000E+02;
ac_state.heading      5.236000E+00;
ac_state.ROCD         0.000000E+00;
ac_state.fp_seg       0.100000E+01;
ac_state.phase        TOF_ ;
#;
# B742;
# AIRCRAFT_STATE;
ac_state.lat          +050.01.35;
ac_state.long         +008.32.35;
ac_state.alt          0.364000E+03;
ac_state.mass         0.617294E+06;
ac_state.thrust       0.200000E+06;
ac_state.V_TAS        0.150000E+03;
ac_state.V_CAS        0.150000E+03;
ac_state.V_G.N        0.750000E+02;
ac_state.V_G.E        0.750000E+02;
ac_state.heading      5.236000E+00;
ac_state.ROCD         0.000000E+00;
ac_state.fp_seg       0.100000E+01;
ac_state.phase        TOF_ ;
#;
# MD80;
# AIRCRAFT_STATE;
ac_state.lat          +050.01.35;
ac_state.long         +008.32.35;
ac_state.alt          0.364000E+03;
ac_state.mass         0.134923E+06;
ac_state.thrust       0.200000E+06;
ac_state.V_TAS        0.150000E+03;
ac_state.V_CAS        0.150000E+03;
ac_state.V_G.N        0.750000E+02;
ac_state.V_G.E        0.750000E+02;
ac_state.heading      5.236000E+00;
ac_state.ROCD         0.000000E+00;
ac_state.fp_seg       0.100000E+01;
ac_state.phase        TOF_ ;
#FI;
```

Hamburg, Germany - Paris, France

```
# Initial Condition File;
#;
# CONFIGURATION;
no_ac                0.500000E+01;
# TIME;
time_step            0.100000E+01;
start                0.100000E+01;
finish               2.160000E+04;
# DESTINATION;
ac_state.destination LFPG;
# A320;
# AIRCRAFT_STATE;
ac_state.lat         +053.37.49;
ac_state.long        +009.59.18;
ac_state.alt         5.300000E+01;
ac_state.mass        0.136687E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     3.926990E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B733;
# AIRCRAFT_STATE;
ac_state.lat         +053.37.49;
ac_state.long        +009.59.18;
ac_state.alt         5.300000E+01;
ac_state.mass        0.119050E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     3.926990E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B738;
# AIRCRAFT_STATE;
ac_state.lat         +053.37.49;
ac_state.long        +009.59.18;
ac_state.alt         5.300000E+01;
ac_state.mass        0.136907E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     3.926990E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B742;
# AIRCRAFT_STATE;
ac_state.lat         +053.37.49;
ac_state.long        +009.59.18;
ac_state.alt         5.300000E+01;
ac_state.mass        0.617294E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     3.926990E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# MD80;
# AIRCRAFT_STATE;
ac_state.lat         +053.37.49;
ac_state.long        +009.59.18;
ac_state.alt         5.300000E+01;
ac_state.mass        0.134923E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     3.926990E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#FI;
```

Munich, Germany - London, United Kingdom

```
# Initial Condition File;
#;
# CONFIGURATION;
no_ac                0.500000E+01;
# TIME;
time_step            0.100000E+01;
start                0.100000E+01;
finish               2.160000E+04;
# DESTINATION;
ac_state.destination EGKK;
# A320;
# AIRCRAFT_STATE;
ac_state.lat         +048.21.14;
ac_state.long        +011.47.10;
ac_state.alt         0.148700E+04;
ac_state.mass        0.136687E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     5.236000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B733;
# AIRCRAFT_STATE;
ac_state.lat         +048.21.14;
ac_state.long        +011.47.10;
ac_state.alt         0.148700E+04;
ac_state.mass        0.119050E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     5.236000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B738;
# AIRCRAFT_STATE;
ac_state.lat         +048.21.14;
ac_state.long        +011.47.10;
ac_state.alt         0.148700E+04;
ac_state.mass        0.136907E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     5.236000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B742;
# AIRCRAFT_STATE;
ac_state.lat         +048.21.14;
ac_state.long        +011.47.10;
ac_state.alt         0.148700E+04;
ac_state.mass        0.617294E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     5.236000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# MD80;
# AIRCRAFT_STATE;
ac_state.lat         +048.21.14;
ac_state.long        +011.47.10;
ac_state.alt         0.148700E+04;
ac_state.mass        0.134923E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     5.236000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#FI;
```

London, United Kingdom - Zurich, Switzerland

```
# Initial Condition File;
#;
# CONFIGURATION;
no_ac                0.500000E+01;
# TIME;
time_step            0.100000E+01;
start                0.100000E+01;
finish               2.160000E+04;
# DESTINATION;
ac_state.destination LSZH;
# A320;
# AIRCRAFT_STATE;
ac_state.lat         +051.08.53;
ac_state.long        -000.11.25;
ac_state.alt         0.196000E+03;
ac_state.mass        0.136687E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     2.618000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B733;
# AIRCRAFT_STATE;
ac_state.lat         +051.08.53;
ac_state.long        -000.11.25;
ac_state.alt         0.196000E+03;
ac_state.mass        0.119050E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     2.618000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B738;
# AIRCRAFT_STATE;
ac_state.lat         +051.08.53;
ac_state.long        -000.11.25;
ac_state.alt         0.196000E+03;
ac_state.mass        0.136907E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     2.618000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B742;
# AIRCRAFT_STATE;
ac_state.lat         +051.08.53;
ac_state.long        -000.11.25;
ac_state.alt         0.196000E+03;
ac_state.mass        0.617294E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     2.618000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# MD80;
# AIRCRAFT_STATE;
ac_state.lat         +051.08.53;
ac_state.long        -000.11.25;
ac_state.alt         0.196000E+03;
ac_state.mass        0.134923E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     2.618000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#FI;
```

Amsterdam, The Netherlands - Madrid, Spain

```
# Initial Condition File;
#;
# CONFIGURATION;
no_ac                0.500000E+01;
# TIME;
time_step            0.100000E+01;
start                0.100000E+01;
finish               2.160000E+04;
# DESTINATION;
ac_state.destination LEMD;
# A320;
# AIRCRAFT_STATE;
ac_state.lat         +052.18.31;
ac_state.long        +004.45.50;
ac_state.alt         0.000000E+00;
ac_state.mass        0.136687E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     3.926990E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B733;
# AIRCRAFT_STATE;
ac_state.lat         +052.18.31;
ac_state.long        +004.45.50;
ac_state.alt         0.000000E+00;
ac_state.mass        0.119050E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     3.926990E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B738;
# AIRCRAFT_STATE;
ac_state.lat         +052.18.31;
ac_state.long        +004.45.50;
ac_state.alt         0.000000E+00;
ac_state.mass        0.136907E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     3.926990E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B742;
# AIRCRAFT_STATE;
ac_state.lat         +052.18.31;
ac_state.long        +004.45.50;
ac_state.alt         0.000000E+00;
ac_state.mass        0.617294E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     3.926990E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# MD80;
# AIRCRAFT_STATE;
ac_state.lat         +052.18.31;
ac_state.long        +004.45.50;
ac_state.alt         0.000000E+00;
ac_state.mass        0.134923E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     3.926990E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#FI;
```

Paris, France - Rome, Italy

```
# Initial Condition File;
#;
# CONFIGURATION;
no_ac                0.500000E+01;
# TIME;
time_step            0.100000E+01;
start                0.100000E+01;
finish               2.160000E+04;
# DESTINATION;
ac_state.destination EDDM;
# A320;
# AIRCRAFT_STATE;
ac_state.lat         +040.28.20;
ac_state.long        -003.33.39;
ac_state.alt         0.200000E+04;
ac_state.mass        0.136687E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.523600E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B733;
# AIRCRAFT_STATE;
ac_state.lat         +040.28.20;
ac_state.long        -003.33.39;
ac_state.alt         0.200000E+04;
ac_state.mass        0.119050E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.523600E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B738;
# AIRCRAFT_STATE;
ac_state.lat         +040.28.20;
ac_state.long        -003.33.39;
ac_state.alt         0.200000E+04;
ac_state.mass        0.136907E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.523600E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B742;
# AIRCRAFT_STATE;
ac_state.lat         +040.28.20;
ac_state.long        -003.33.39;
ac_state.alt         0.200000E+04;
ac_state.mass        0.617294E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.523600E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# MD80;
# AIRCRAFT_STATE;
ac_state.lat         +040.28.20;
ac_state.long        -003.33.39;
ac_state.alt         0.200000E+04;
ac_state.mass        0.134923E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.523600E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#FI;
```

Rome, Italy - Frankfurt, Germany

```
# Initial Condition File;
#;
# CONFIGURATION;
no_ac                0.500000E+01;
# TIME;
time_step            0.100000E+01;
start                0.100000E+01;
finish               2.160000E+04;
# DESTINATION;
ac_state.destination EDDF;
# A320;
# AIRCRAFT_STATE;
ac_state.lat         +041.47.58;
ac_state.long        +012.35.42;
ac_state.alt         0.427000E+03;
ac_state.mass        0.136687E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     5.936000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B733;
# AIRCRAFT_STATE;
ac_state.lat         +041.47.58;
ac_state.long        +012.35.42;
ac_state.alt         0.427000E+03;
ac_state.mass        0.119050E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     5.936000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B738;
# AIRCRAFT_STATE;
ac_state.lat         +041.47.58;
ac_state.long        +012.35.42;
ac_state.alt         0.427000E+03;
ac_state.mass        0.136907E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     5.936000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B742;
# AIRCRAFT_STATE;
ac_state.lat         +041.47.58;
ac_state.long        +012.35.42;
ac_state.alt         0.427000E+03;
ac_state.mass        0.617294E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     5.936000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# MD80;
# AIRCRAFT_STATE;
ac_state.lat         +041.47.58;
ac_state.long        +012.35.42;
ac_state.alt         0.427000E+03;
ac_state.mass        0.134923E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     5.936000E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#FI;
```

Madrid, Spain - Munich, Germany

```
# Initial Condition File;
#;
# CONFIGURATION;
no_ac                0.500000E+01;
# TIME;
time_step            0.100000E+01;
start                0.100000E+01;
finish               2.160000E+04;
# DESTINATION;
ac_state.destination EDDM;
# A320;
# AIRCRAFT_STATE;
ac_state.lat         +040.28.20;
ac_state.long        -003.33.39;
ac_state.alt         0.200000E+04;
ac_state.mass        0.136687E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.523600E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B733;
# AIRCRAFT_STATE;
ac_state.lat         +040.28.20;
ac_state.long        -003.33.39;
ac_state.alt         0.200000E+04;
ac_state.mass        0.119050E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.523600E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B738;
# AIRCRAFT_STATE;
ac_state.lat         +040.28.20;
ac_state.long        -003.33.39;
ac_state.alt         0.200000E+04;
ac_state.mass        0.136907E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.523600E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B742;
# AIRCRAFT_STATE;
ac_state.lat         +040.28.20;
ac_state.long        -003.33.39;
ac_state.alt         0.200000E+04;
ac_state.mass        0.617294E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.523600E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# MD80;
# AIRCRAFT_STATE;
ac_state.lat         +040.28.20;
ac_state.long        -003.33.39;
ac_state.alt         0.200000E+04;
ac_state.mass        0.134923E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.523600E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#FI;
```

Zurich, Switzerland - Hamburg, Germany

```
# Initial Condition File;
#;
# CONFIGURATION;
no_ac                0.500000E+01;
# TIME;
time_step            0.100000E+01;
start                0.100000E+01;
finish               2.160000E+04;
# DESTINATION;
ac_state.destination EDDH;
# A320;
# AIRCRAFT_STATE;
ac_state.lat         +047.27.53;
ac_state.long        +008.32.57;
ac_state.alt         0.141600E+04;
ac_state.mass        0.136687E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.174500E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B733;
# AIRCRAFT_STATE;
ac_state.lat         +047.27.53;
ac_state.long        +008.32.57;
ac_state.alt         0.141600E+04;
ac_state.mass        0.119050E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.174500E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B738;
# AIRCRAFT_STATE;
ac_state.lat         +047.27.53;
ac_state.long        +008.32.57;
ac_state.alt         0.141600E+04;
ac_state.mass        0.136907E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.174500E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# B742;
# AIRCRAFT_STATE;
ac_state.lat         +047.27.53;
ac_state.long        +008.32.57;
ac_state.alt         0.141600E+04;
ac_state.mass        0.617294E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.174500E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#;
# MD80;
# AIRCRAFT_STATE;
ac_state.lat         +047.27.53;
ac_state.long        +008.32.57;
ac_state.alt         0.141600E+04;
ac_state.mass        0.134923E+06;
ac_state.thrust      0.200000E+06;
ac_state.V_TAS       0.150000E+03;
ac_state.V_CAS       0.150000E+03;
ac_state.V_G.N       0.750000E+02;
ac_state.V_G.E       0.750000E+02;
ac_state.heading     0.174500E+00;
ac_state.ROCD        0.000000E+00;
ac_state.fp_seg      0.100000E+01;
ac_state.phase       TOF_ ;
#FI;
```

B.3.3 Core Europe ATC Plan

```
no_sectors  8;
no_airports 9;
no_stacks   0;
#;
# SectorID   MinLat    MaxLat    MinLon    MaxLon     Neighbors (N,E,S,W)  Script Logic;
#;
1098 4.86000E1 5.43000E1 -1.12500E1 0.00000E0 0    1269 1076 0    "v_sep_core.scr";
1076 4.34000E1 4.86000E1 -1.12500E1 0.00000E0 1098 1247 1074 0    "v_sep_core.scr";
1074 3.84000E1 4.34000E1 -1.12500E1 0.00000E0 1076 1245 0    0    "v_sep_core.scr";
1269 4.86000E1 5.43000E1 0.00000E0  1.12500E1 0    1270 1247 1098 "v_sep_core.scr";
1247 4.34000E1 4.86000E1 0.00000E0  1.12500E1 1269 1248 1245 1076 "v_sep_core.scr";
1245 3.84000E1 4.34000E1 0.00000E0  1.12500E1 1247 1246 0    1074 "v_sep_core.scr";
1248 4.34000E1 4.86000E1 1.12500E1 2.25000E1  0    0    1246 1247 "v_sep_core.scr";
1246 3.84000E1 4.34000E1 1.12500E1 2.25000E1  1248 0    0    1245 "v_sep_core.scr";
#;
# AirportID     Lat        Lon       Alt(ft)  Orientation(deg)   SectorID;
#;
   LSZH 047:27:53N  008:32:57E    1416        245             1247   "blank.scr";
   EDDH 053.37.49N  009.59.18E      53         50             1269   "blank.scr";
   EHAM 052.18.31N  004.45.50E       0         10             1269   "blank.scr";
   LFPG 049.00.46N  002.32.60E     392         90             1269   "blank.scr";
   EDDF 050.01.35N  008.32.35E     364         70             1269   "blank.scr";
   LIRA 041.47.58N  012.35.42E     427        160             1246   "blank.scr";
   EGKK 051.08.53N  000.11.25W     196        260             1098   "blank.scr";
   EDDM 048.21.14N  011.47.10E    1487         80             1248   "blank.scr";
   LEMD 040.28.20N  003.33.39W    2000        150             1074   "blank.scr";
#;
# StackID       Lat        Lon       AltMax(ft)  AltMin(ft)  Orientation(deg)   AirportID;
#;
#;
#FI
```

B.3.4 TMA Controller Logic: v sep core

```
set conflict_radius 30 nm
get conflicts sector_id conflict_radius
s_while ((CURRENT_CONFLICT < NO_CONFLICTS) AND (NO_CONFLICTS > 0)) do
  increment current_conflict
  if (ACAC_DIST < 10 nm) then
     if (ac1_alt { ac2_alt)   then
        if (max_alt_ac2 } (ac2_alt + 1000 ft)) then
          command RESTRICT_UPPER_FL AC1 (ac1_alt - 1000 ft) (10nm / ac1_vel)
          command RESTRICT_LOWER_FL ac2 (ac2_alt + 1000 ft) (10nm / ac2_vel)
        else
          command RESTRICT_UPPER_FL AC1 (ac1_alt - 1000 ft) (10nm / ac1_vel)
        end
     else
        if (max_alt_ac1 } (ac1_alt + 1000 ft)) then
          command RESTRICT_LOWER_FL AC1 (ac1_alt + 1000 ft) (10nm / ac1_vel)
          command RESTRICT_UPPER_FL ac2 (ac2_alt - 1000 ft) (10nm / ac2_vel)
        else
          command RESTRICT_UPPER_FL ac2 (ac2_alt - 1000 ft) (10nm / ac2_vel)
        end
     end
  else
    if (scenario = 0) then ; overtake
     if (ac2_vel - ac1_vel > 100) then
         command RESTRICT_UPPER_FL ac2 (ac2_alt - 1000 ft) (30nm / ac2_vel)
     else
        command increase_speed ac1 ((ac1_vel + ac2_vel)*0.5) 900
        command restrict_speed ac2 ((ac1_vel + ac2_vel)*0.5) 900
     end
    end
    if (scenario = 1) OR (scenario = 2) OR (scenario = 3) then ; acute, right, obtuse
       command restrict_upper_fl ac1 (ac1_alt - 1000 ft) (30nm / ac1_vel)
    end
    if (scenario = 4) then ; head on
      if (MAX_ALT_AC1 } MAX_ALT_AC2) AND (MAX_ALT_AC1 > ac1_alt) then
         command RESTRICT_LOWER_FL AC1 (ac1_alt + 1000 ft) (30 nm / (ac1_vel + ac2_vel))
         command RESTRICT_UPPER_FL ac2 (ac2_alt - 1000 ft) (30 nm / (ac1_vel + ac2_vel))
      else
         command RESTRICT_LOWER_FL AC2 (ac2_alt + 1000 ft) (30 nm / (ac1_vel + ac2_vel))
         command RESTRICT_UPPER_FL ac1 (ac1_alt - 1000 ft) (30 nm / (ac1_vel + ac2_vel))
      end
    end
  end
end
stop
```

Glossary

ACARS *Airborne Communications Addressing and Reporting System*

ADS *Automatic Dependent Surveillance*

ADS-B *Automatic Dependent Surveillance Broadcast*

AFAS *Aircraft in the Future ATM System*

AFTN *Aeronautical Fixed Telecommunications Network*

AGL *Above Ground Level*

AOC *Airline Operational Centers*

APF *Airline Procedure File*

ATC *Air Traffic Control*

ATM *Air Traffic Management*

ATMOS *Atmospheric Weather Model*

ATN *Aeronautical Telecommunications Network*

ATS *Air Traffic Services*

ATSP *Air Traffic Service Provider*

B *Byte*

BADA *Base of Aircraft Data*

BOC *Bottom of Climb*

CNS *Communication, Navigation and Surveillance*

CPDLC *Controller-Pilot Data Link Communications*

CTAS *Center-TRACON Automation System*

DAG-TM *Distributed Air-Ground Traffic Management*

DME *Distance Measuring Equipment*

DoF *Degrees of Freedom*

EEC *EUROCONTROL Experimental Center*

FAA *Federal Aviation Administration (U.S.A.)*

fast-time *Simulating one second in less than on second of real time*

FFPI *Free Flight Phase I*

FFSIM *Free Flight Simulation*

FMS *Flight Management System*

GPM *Garden's Point Modula-2*

GPS *Global Positioning System*

HF *High Frequency*

ICAO *International Civil Aviation Organisation*

IFR *Instrument Flight Rules*

IMC *Instrument Meteorological Conditions*

knots *nautical miles per hour*

MB *Megabyte*

METAF *Terminal Aerodrome Forecasts*

METAR *Meteorological Airfield Report*

MSL *Mean Sea Level*

NARSIM *NLR's Air Traffic Control Research Simulator*

NLR *National Aerospace Laboratory (The Netherlands)*

NOAA *National Atmospheric and Oceanographic Agency (The United States)*

NPN *NOAA Profiler Network*

OPF *Operations Performance File*

PHARE *Program for Harmonised Air Transport Management Research*

PTF *Performance Table File*

RAMS *Reorganised ATC Mathematical Simulator*

RNP *Required Navigation Performance*

ROCD *Rate of climb or descent*

RTA *Required Time of Arrival*

RTCA *Radio Technical Commission for Aeronautics*

SID *Standard Instrument Departure*

SIGMET *Significant Meteorological Information*

SITA *Sociètè Internationale de Telecommunications*

SSR *Secondary Surveillance Radar* utilises a variety of modes, A, C, and S

STAR *Standard Terminal Arrival Route*

SUA *Special Use Airspace*

TAAM *Total Airspace & Airport Modeler*

TACAN *Tactical Air Navigation*

TAS *True Air Speed*

TOD *Top of Descent*

TRACON *Terminal Area Controller*

UHF *Ultra-high Frequency*

UREA *User Request Evaluation Tool*

URET *User Request Evaluation Tool* ¡/gdf¿

VFR *Visual Flight Rules*

VHF *Very-high Frequency*

VMC *Visual Meteorological Conditions*

VOR *VHF Omnidirectional Range*

BIBLIOGRAPHY

[1] Nolan, M. S. *Fundamentals of Air Traffic Control*. Wadsworth Publishing Company, Belmont, CA, 1990.

[2] (ATAG), A. T. A. G. European Air Traffic Forecasts. Web Page, 2000. Location: http://www.atag.org/ETF/index.htm, was accessed on 14 March, 2000.

[3] Schröter, H. PHARE Project Overview. Web Page, 1998. Location: http://www.eurocontrol.int/phare/documentation/overview/overview.htm, was accessed on 14 March, 1998.

[4] Zeghal, K., and Hoffman, E. ICAS 2000 Congress. In *Delegation of Separation Assurance to Aircraft: Towards a Framework for Analysing the Different Concepts and Underlying Principles*, 2000. August, 2000-30 September, 2000. Harrogate, UK.

[5] Donohue, G. L., and Laska, W. D. Air Transportation Systems Engineering. In Donohue, G. L., and Zellweger, A. G., editors, *United States and European Airport Capacity Assessment Using the GMU Macroscopic Capacity Model*, volume 193 of *Progress in Astronautics and Aeronautics*, pages 27–47. AIAA, Reston, VA, 2001.

[6] Liang, D., Marnane, W., and Bradford, S. Air Transportation Systems Engineering. In *Comparison of U.S. and European Airports and Airspace to Support Concept Validation*, volume 193 of *Progress in Astronautics and Aeronautics*, pages 27–47. AIAA, Reston, VA, 2001.

[7] Kayton, M., and Fried, W. R. *Avionics Navigation Systems*. John Wiley & Sons Inc., New York, NY, 2nd edition, 1997.

[8] Hoekstra, J. M., Ruigrok, R. C. J., and van Gent R. N. H. W. Air Transportation Systems Engineering. In Donohue, G. L., and Zellweger, A. G., editors, *Free Flight in a Crowded Airspace*, volume 193 of *Progress In Astronautics and Aeronautics*, pages 533–545. AIAA, Reston, VA, 2001.

[9] Williams, D. H., and Green, S. M. Flight Evaluation of Center-TRACON Automation System Trajectory Prediction Process. Nasa/tp-1998-208439, NASA, 1998.

[10] Wickens, Christopher D. andMavor, A. S. a. R., and McGee, J. P. *The Future of Air Traffic Control: Human Operators and Automation*. National Academy Press, Washington, D.C., 1998.

[11] Abbott, T. S. Flight Assessment of a Data-Link Based Navigation -Guidance Concept. Tm - 84493, NASA Langley, 1983.

[12] Green, S. M., Davis, T. J., and Erzberger, H. AIAA Guidance, Navigation and Control Conference. In *A Piloted Simulator Evaluation of a Ground -Based 4D Descent Advisor Algorithm*, AIAA 87-2522, pages 1173–1180, Reston, VA, 1987. AIAA. 17 August, 1987-19 August, 1987. Monterey, California.

[13] Erzberger, H., Davis, T. J., and Green, S. M. Machine Intelligence in Air Traffic Management. In *Design of Center-TRACON Automation System*, AGARD CP-538, pages 11.1–11.11. NATO, 1993. 11 May, 1993-14 May, 1993. Berlin, Germany.

[14] Brudnicki, D. J., and McFarland, A. L. 1st USA/Europe ATM R&D Seminar. In *User Request Evaluation Tool (URET) Conflict Probe Performance and Benefits Assessment*, Bretingy-sur-Orge, 1997. EUROCONTROL. June, 1997. Saclay, France.

[15] Administration, F. A. FAA Free Flight Technology Used Daily at Kansas City Center. Electronic Citation, 2001. Location http://www.faa.gov/apa/pr/pr.cfm?id=1467v was accessed on 10 February, 2002.

[16] Lee, H. Q., Neuman, F., and Hardy, G. H. 4D Area Navigation System Description and Flight Test Results. Nasa tn d - 7874, Washington D.C., 1975.

[17] F., N., and Kreindler, E. Minimum-Fuel, Three-Dimensional Flight Path Guidance of Transport Jets. Nasa tp - 2326, Washington D.C., 1984.

[18] Tobias, L., Alcabin, M., Erzberger, H., and O'Brien, P. J. Simulation Studies of Time-Control Procedures for the Advanced Air Traffic Control System. Nasa tp - 2493, NASA Ames, 1985.

[19] Parker, J. F. J., and Duffy, J. W. A Flight Investigation of Simulated Data-Link Communications during Single Pilot IFR Flight: Volume II – Flight Evaluations. Nasa cr - 3653, NASA Langley, 1982.

[20] Knox, C. E., and Scanlon, C. H. Flight Tests with a Data Link used for Air Traffic Control Information. Nasa tp - 3135, NASA Langley, 1991.

[21] Gustavsson, N. AIAA/IEEE Digital Avionics Systems Conference. In *VDL Mode 4 / STDMA – a CNS Data Link*, pages 111–116, Reston, VA, 1996. AIAA. 27 October, 1996-31 October, 1996. Atlanta, GA, 15th Proceedings.

[22] Nilim, A., Ghaoui, L. E., Hansen, M., and Duong, V. 4th USA/Europe ATM R&D Seminar. In *Trajectory-based Air Traffic Management (TB-ATM) under Weather Uncertainty*, Internet, 2001. FAA/EUROCONTROL. 3 December, 2001-7 December, 2001. Sante Fe, NM.

[23] McDonald, J. A., and Zhao, Y. Time Benefits of Free-Flight for Commercial Aircraft. http://www.aem.umn.edu/research/atc/download/AIAANote.pdf.

[24] Isaacson, D. R., and Erzberger, H. AIAA/IEEE Digital Avionics Systems Conference. In *Design of a Conflict Detection Algorithm for The Center/TRACON Automation System*, 2, pages 9.3/1–9.3/9, NJ, 1997. IEEE. 26 October, 1997-30 October, 1997. Irvine, CA.

[25] Eby, M. S. A Self-Organizational Approach for resolving Air Traffic Conflicts. *The Lincoln Laboratory Journal*, 7(2):239–253, 1994.

[26] Paielli, R. A., and Erzberger, H. Conflict Probability Estimation for Free Flight. *Journal of Guidance Control and Dynamics; 35th Aerospace Sciences Meeting and Exhibit*, 20(AIAA 97-0001):588–596, 1997.

[27] RTCA, I. Final Report of the RTCA Task Force 3, Free Flight Implementation. Technical report, Washington D.C., 1995.

[28] Jardin, M. R. Proceedings of the American Control Conference. In *Ideal Free Flight through Multiple Aircraft Neighboring Optimal Control*, pages 2879–2855. American Automatic Control Council, 2000. June, 2000. Chicago, IL.

[29] Ratcliffe, S. Free-Flight in Europe, Problems and Solutions. *The Journal of Navigation*, 54(2):213–221, 2001.

[30] Warren, A. W., and Schwab, R. W. A Methodology and Initial Results Specifying Requirements for Free Flight Transitions. *Air Traffic Control Quarterly*, 5(3):133–156, 1997.

[31] Magill, S. A. N. Air Transportation Systems Engineering. In Donohue, G. L., and Zellweger, A. G., editors, *Effect of Direct Routing on Air Traffic Control Capacity*, volume 193 of *Progress in Astronautics and Aeronautics*, pages 385–396. AIAA, Reston, VA, 2001.

[32] Andrews, J. W., and Welch, J. D. USA/Europe ATM R&D Seminar. In *Workload Implications of Free Flight Concepts*, 1997. June, 1997. Saclay, France.

[33] Hoekstra, J. M., van Gent, R. N. H. W., and Rugrok, R. C. J. AIAA Guidance, Navigation and Control Conference. In *Conceptual Design of Free Flight with Airborne Separation Assurance*, pages 807–817, Reston, VA, 1998. AIAA. 10 August, 1998-10 August, 1998. Boston, MA.

[34] Wilson, I. A. B. PHARE: Definition and Use of Tubes. Doc 96-70-18, Brussells, 1996.

[35] Wichman, K. D., Carlsson, G., and Lindberg, L. G. V. IEEE Digital Avionics Systems Conference. In *Flight Trials: "Runway-To-Runway" Required Time of Arrival Evaluations for Time-Based ATM Environment*, NJ, 2001. IEEE. 14 October, 2001-18 October, 2001.

[36] Ballin, M. G., Wing, D. J., Huges, M. F., and Conway, S. R. AIAA Guidance, Navigation and Control Conference. In *Separation Assurance and Traffic Management: Research of Concepts and Technology*, AIAA 99-3989, pages 313–324, Reston, VA, 1999. AIAA. 6 August, 1999-9 August, 1999. Portland, OR.

[37] FAA. About Free Flight Phase I. Web Page. Location: http://ffp1.faa.gov/about/about_ffp1.asp, was accessed on 14 March, 2002.

[38] Post, J., and Knorr, D. 5th USA/Europe ATM R&D Seminar5th USA/Europe ATM R&D Seminar. In *Free Flight Program Update*, Bretigny, France, 2003. Eurocontrol. 23 June, 2003-27 June, 2003. Budapest, Hungary.

[39] Williams, A., Mondoloni, S., Liang, D., Bradford, S., and Jehlen, R. 5th USA/Europe ATM R&D Seminar. In *The Big Iron*, Bretigny, France, 2003. Eurocontrol. 23 June, 2003-27 June, 2003. Budapest, Hungary.

[40] Haraldsdottir, A., Schoemig, E. G., Schwab, R. W., Singleton, M. K., Sipe, A. H., and van Tulder, P. A. 5th USA/Europe ATM R&D Seminar. In *BOEING Capacity Increasing ATM Concept for 2020*, Bretigny, France, 2003. Eurocontrol. 23 June, 2003-27 June, 2003. Budapest, Hungary.

[41] Hu, X., Wu, S.-F., and Jiang, J. AIAA Guidance, Navigation and Control Conference. In *Genetic Algorithm Based On-Line Real-Time Optimization of Commercial Aircraft's Flight Path for a Free Flight Strategy*, AIAA 2001-4234, pages 1153–1163, Reston, VA, 2001. AIAA. 6 August, 2001-9 August, 2001. Montreal, Canada.

[42] Mykoniatis, G., and Martin, P. Study of the Acqusition of Data from Aircraft Operators to Aid Trajctory Prediction Calculation. Eec note no. 18/98, Bretigny-sur-Orge, France, 1998.

[43] ARINC. AviNet Data Communications Services. Web Page. Location: http://www.arinc.com/products/voice_data_comm/avinet/, was accessed on 24 April, 2002.

[44] Leighton, S. J., McGregor, A. E., Lowe, D., et al. GNSS Guidance for All Phases of Flight: Practical Results. 54(1):1–13, 2001.

[45] FAA. Guidelines for Design Approval of Aircraft Data Communications Systems. Statute, 1999. AC 20-140.

[46] Week, A. Special Issue of Air Travel in Crisis. *Aviation Week*, 151(17), 1999.

[47] Ochieng, W. Y., Sauer, K., Cross, P. A., et al. Potential Performance Levels of a Combined Galileo/GPS Navigation System. *The Journal of Navigation*, 54(2):185–197, 2001.

[48] FAA. Frequently Asked Questions: What is the role of current ground-based navigation aids (ILS, VOR/DME, NDB, LORAN, etc.) in the future successful implementation of GPS technology? Web Page.

[49] Trotter-Cox, A. Required Navigation Performance (RNP) Another Step Towards Global Implementation of CNS/ATM. Electronic Citation, 1999. Location http://www.aviationmanuals.com/articles/article3.html was accessed on 12 February, 2002.

[50] Scardina, J. Overview of the FAA ADS-B Link Decision. Technical report, Washington, D.C., 2002.

[51] Scardina, J. The Approach and Basis for the FAA ADS-B Link Decision. Technical report, Washington, D.C., 2002.

[52] Sigmore, T. L., and Hong, Y. IEEE Digital Avionics Systems Conference. In *Party-line Communications in a Data Link Enviornment*, pages 2E4–1–2E4/8, Piscataway, NJ, 2000. IEEE. 7 October, 2000-13 October, 2000. Philadelphia, PA 19th Proceedings.

[53] Dieudonne, J., Joseph, M., and Cardosi, K. IEEE Digital Avionics Systems Conference. In *Is the Proposed Design of the Aeronautical Data Link System likely to Reduce the Miscommunications Error Rate and Controller/Flight Crew Input Errors*, pages 5E3–1–5E3/9, Piscataway, NJ, 2000. IEEE. 2000. Philadelphia, PA, 19th Proceedings.

[54] Fan, T. P., and Kuchar, J. K. IEEE Digital Avionics Systems Conference. In *Evaluation of Interfaces for Pilot-Air Traffic Control Data Link Communications*, pages 4A5–1–4A5/8, Piscataway, NJ, 2000. IEEE. 2000. Philadelphia, PA, 19th Proceedings.

[55] Lozito, S., Verma, S., Martin, L., Dunbar, M., and McGann, A. 5th USA/Europe ATM R&D Seminar. In *The Impact of Voice, Data Link and Mixed Air Traffic Control Environments on Flight Deck Procedures*, Bretigny, France, 2003. Eurocontrol. 23 June, 2003-27 June, 2003. Budapest, Hungary.

[56] Project, M. A. M. TAAM: Total Airspace & Airport Modeller. Web Page, 1996. Location: http://web.mit.edu/aeroastro/www/labs/AATT/reviews/taam.html, was accessed on 10 October, 2002.

[57] Software, I. RAMS Plus 3.0 User's Manual. Technical report, 2000.

[58] Project, M. A. M. RAMS: Reorganized ATC Mathematical Simulator. Web Page. Location: http://web.mit.edu/aeroastro/www/labs/AATT/reviews/rams.html, was accessed on 10 October, 2002.

[59] Funabiki, K., Muraoka, K., and Tanaka, K. AIAA Flight Simulation and Technology Conference. In *A Flight Simulation for Human Error Study*, AIAA 95-3410, pages 171–178, Reston, VA, 1995. AIAA. August 7-10. Baltimore, MD.

[60] Alliot, J. M., Bosc, J. F., Durand, N., and Maugis, L. AIAA/IEEE Digital Avionics Systems Conference. In *CATS: A Complete Air Traffic Simulator*, Piscataway, NJ, 1997. IEEE. 1997. Irvine, CA.

[61] Miles, E. S., C., D. P., and Wing, D. J. AIAA Modeling and Simulation Technologies Conference. In *Development of a Free Flight Simulation Infrastructure*, AIAA 99-4193, pages 307–316, Reston, VA, 1999. AIAA. 9 August, 1999-11 August, 1999. Portland, OR.

[62] Schleicher, D. R., Davis, P. C., Wallace, D. E., Shah, S., D. S., Mueller, T., Jones, E., Krozel, J., Couluris, G. J., and Dow, D. Free Flight Simulation Infrastructure. C175.21, Ames, CA, 2002.

[63] NLR. NARSIM Home Page. Web Page. Location: http://www.nlr.nl/public/narsim2/, was accessed on 13 August, 2003.

[64] Green, S. M., and Vivona, R. A. AIAA Guidance, Navigation and Control Conference. In *Local Traffic Flow Management Concept for Constrained En Route Airspace Problems*, AIAA 2001-4115, pages 578–588, Reston, VA, 2001. AIAA. 6 August, 2001-9 August, 2001. Montreal, Canada.

[65] Mueller, K. T., and Krozel, J. AIAA Guidance, Navigation and Control Conference. In *Aircraft ADS-B Intent Verification Based on a Kalman Tracking Filter*, pages 523–533, Reston, VA, 2000. AIAA. 14 August, 2000-17 August, 2000. Denver, CO.

[66] Coppenbarger, R. A., Kanning, G., and Salcido, R. 4th USA/Europe ATM R&D Seminar. In *Real-Time Data Link of Aircraft Parameters to the Center-TRACON Automation System (CTAS)*, Internet, 2001. FAA/EUROCONTROL. 3 December, 2001-7 December, 2001. Sante Fe, NM.

[67] Cole, R. E., Green, S. M., Jardin, M. R., Schwartz, B. E., and Benjamin, S. G. 3rd USA/Europe ATM R&D Seminar. In *Wind Prediction Accuracy for Air Traffic Management Decision Support Tools*, page A3, Bretigny-sur-Orge, France, 2000. EUROCONTROL. 13 June, 2000-16 June, 2000. Napoli, Italy.

[68] Warren, A. W. 3rd USA/Europe ATM R&D Seminar. In *Trajectory Prediction Concepts for Next Generation Air Traffic Management*, page E1, Bretigny-sur-Orge, France, 2000. EUROCONTROL. 13 June, 2000-16 June, 2000. Napoli, Italy.

[69] Hoffman, E. Contribution to Aircraft Performance Modeling for ATC. *Air Traffic Control Quarterly*, 2(2):103–130, 1994.

[70] Renteux, J.-L. Aircraft Modelling Standards for future ATC Systems. Doc. 872003, Brussells, Belgium, 1987.

[71] Green, S. M., Grace, M. P., and Williams, D. H. 3rd USA/Europe ATM R&D Seminar. In *Flight Test Results: CTAS and FMS Cruise/Descent Trajectory Prediction Accuracy*, page E1, Brussels, 2000. EUROCONTROL. 13 June, 2000-16 June, 2000. Napoli, Italy.

[72] Nuic, A. User Manual for the Base of Aircraft Data (BADA). Eec note no. 20/00, Bretigny-sur-Orge, France, 2000.

[73] Gill, B., and Maddock, B. Prediction of Optimal 4D Trajectories in the Presence of Time and Alitutde Constraints. Doc 97-70-09, Brussels, 1997.

[74] Sheehan, C. Coverage of 2000 European Air Traffic for the Base of Aircraft Data. Eec note no. 18/01, Bretigny-sur-Orge, 2001.

[75] Alliot, J.-M., Durrand, N., and Granger, G. USA/Europe ATM R&D Seminar. In *FACES: A Free-flight Autonomous and Coordinated Embarked Solver*, http://atm-seminar-98.eurocontrol.fr/finalpapers/track2/alliot1.pdf, 1998. 1 December, 1998-4 December, 1998. Orlando, FL.

[76] Weidner, T. J., and Green, S. M. 3rd USA/Europe ATM R&D Seminar. In *Modeling ATM Automation Metering Conformance Benefits*, page D3, sur-Orge, France, 2000. EUROCONTROL. 12 June, 2000. Napoli, Italy.

[77] Warren, A. W., Schwab, R. W., Geels, T. J., and Shakarian, A. Conflict Probe Concepts Analysis in Support of Free Flight. Nasa cr 201623, Hampton, VA, 1997.

[78] Software, I. ATMOS Weather Model 1.0 Users Manual. Technical report, 2000.

[79] Daniels, G. E., and Smith, O. E. Scalar and Component Wind Correlations between Altitude Levels for Cape Kennedy, Florida, and Santa Monica, California. Nasa tn d - 3815, Washington D.C., 1968.

[80] Vaughan, W. W. Interlevel and Intralevel Correlations of Wind Components for Six Geographical Locations. Nasa tn d - 561, Washington D.C., 1960.

[81] Oceanographic, N., and Agency, A. About NOAA Profiler Sites. Web Page, 2002. Location: http://www.profiler.noaa.gov/jsp/aboutNpnProfilers.jsp, was accessed on 4 June, 2002.

[82] NOAA. NPN Data Text Format. Web Page. Location: http://www.profiler.noaa.gov/test/servlet/TextDump, was accessed on 25 August, 2003.

[83] Benjamin, L. NPN accuracy? Electronic Mail. Sent to Margot Ackley and Karen Feigh on 30 May, 2002.

[84] Sandwell, D. T. Biharmonic Spline Interpolation of GEOS-3 and SEASAT Altimeter Data. *Geophysical Research Letters*, (2):139–142, 1987.

[85] Palmen, E., and Newton, C. W. *Atmospheric Circulation Systems*. International Geophysics. Academic Press, 1 edition, 1969.

[86] CAASD, M. Controller Pilot Data Link Communication – Capabilities. Web Page, 2001. Location: http://www.mitrecaasd.org/proj/cpdlc/capabilities.html, was accessed on 12 October, 2001.

[87] Nolan, M. S. *Fundamentals of Air Traffic Control*. Wadsworth Publishing Company, Pacific Grove, CA, 1999.

[88] Fiornio, F. Controller-Pilot Data Link Goes Live in Miami. *Aviation Week & Space Technology*, 157(16):40–41, 2002.

[89] Yuchnovicz, D. E., Novacek, P. F., Burgess, M. A., Heck, M. L., and Stokes, A. F. Use of a Data-LinkedWeather Information Display and Effects on Pilot Navigation Decision Making in a Piloted Simulation Study. Nasa/cr-2001-211047, Langley, VA, 2001.

[90] EUROCONTROL, F. . 1090 MHz Extended Squitter Assessment Report. 1090-wp-12-05, Washington, D.C., 2002.

[91] Zwillinger, D. E. *CRC Standard Mathematical Tables and Formulae*. CRC Press LLC, New York, NY, 2000.

[92] Oosterom, P. J. M. V. Reactive Data Structures For Geographic Information Systems. In *Overview of Spatial Data Structures*, pages 25–48. Leiden University, 1990, Netherlands.

[93] Samet, H. *Design and Analysis of Spacial Data Structures*. Samet, Hanan, College Park, Maryland, 2002.

[94] Tobler, W., and Zi-tan, C. A Quadtree for Global Information Storage. *Geographical Analysis*, 18(4):360–371, 1986.

[95] Samet, H., and Webber, R. E. Heirarchical Data Structures and Algorithms for Computer Graphics. *IEEE Computer Graphics & Applications*, pages 48–68, 1988.

[96] Samet, H. Neighbor Finding Techniques for Images Represented by Quadtrees. *Computer Graphics and Image Processing*, 18(1):37–57, 1982.

[97] Galotti Jr., V. P. *The Future of Air Navigation System (FANS)*. Ashgate Publishing company, Cambridge, UK, 1998.

[98] Nickum, J. D. Air Transport Avionics Cost Estimation Related to Future Communication Transitions: Coorination Draft. Wn00w0000026, McLean, VA, 2000.

[99] Grimaud, I., Hoffman, E., and Zeghal, K. SAE/AIAA World Aviation Congress. In *Evaluation of Delegation of Sequencing Operations to the Flight Crew from a Controller Perspective – Preliminary Results*, 2000-01-5566. SAE International, 2000. October, 2000. San Diego, CA.

[100] Vormer, F. J., Mulder, M., van Paassen, R. M., and Mulder, B. J. A. AIAA Guidance, Navigation and Control Conference. In *Design and Preliminary Evaluation of a Segment-based Routing Methodology*, Reston, VA, 2002. AIAA. 5 August, 2002-8 August, 2002. Monterey, CA.

[101] Stones, R., and Matthew, N. *Linux Programming.* Wrox Press Ltd, Birmingham, 2nd edition, 2001.