97139-01

Advanced Free Flight Planner and Dispatcher's Workstation

Preliminary Design Specification

J. Wilson
C. Wright
G. J. Couluris

Prepared for:

Advanced Air Transportation Technology (AATT) Program
National Aeronautics and Space Administration
Ames Research Center, Moffett Field, CA 94035

under

NASA Contract No. NAS2-14289(CDT)

November 1997

SEAGULL TECHNOLOGY, INC. 16400 Lark Avenue, Los Gatos, CA 95032-2547 • (408) 358-7100

Abstract

The National Aeronautics and Space Administration (NASA) has implemented the Advanced Air Transportation Technology (AATT) program to investigate future improvements to the national and international air traffic management systems. This research, as part of the AATT program, developed preliminary design requirements for an advanced Airline Operations Control (AOC) dispatcher's workstation, with emphasis on flight planning. This design will support the implementation of an experimental workstation in NASA laboratories that would emulate AOC dispatch operations.

The work developed an airline flight plan data base and specified requirements for: a computer tool for generation and evaluation of free flight, user preferred trajectories (UPT); the kernel of an advanced flight planning system to be incorporated into the UPT-generation tool; and an AOC workstation to house the UPT-generation tool and to provide a real-time testing environment.

A prototype for the advanced flight plan optimization kernel was developed and demonstrated. The flight planner uses dynamic programming to search a four-dimensional wind and temperature grid to identify the optimal route, altitude and speed for successive segments of a flight. An iterative process is employed in which a series of trajectories are successively refined until the UPT is identified. The flight planner is designed to function in the current operational environment as well as in free flight. The free flight environment would enable greater flexibility in UPT selection based on alleviation of current procedural constraints. The prototype also takes advantage of advanced computer processing capabilities to implement more powerful optimization routines than would be possible with older computer systems.

Acknowledgment

This work was performed for the National Aeronautics and Space Administration (NASA) as part of the Advanced Air Transportation Technology (AATT) program. Mr. Del Weathers, NASA Ames Research Center, provided technical monitoring, direction and coordination. Mr. David Schleicher, NASA Ames Research Center, provided technical support.

The research was performed by Seagull Technology, Inc., Los Gatos, CA. The Seagull project team developed preliminary design requirements for an experimental advanced Airline Operations Control (AOC) dispatcher's workstation Dr. John Wilson directed the design and development of the flight planner kernel. Dr. Chris Wright developed the software for the flight planner kernel. Dr. George Couluris coordinated project technical tasks. Dr. John Sorensen provided supervisory review and technical and documentation support.

Table of Contents

List of Figures

1. Introduction

This report describes the results of research performed as part of the National Aeronautics and Space Administration (NASA) Advanced Air Transportation Technology (AATT) program. This work investigates means by which dispatch operations of Airline Operations Control (AOC) facilities could support free flight concepts. The free flight concept promotes government and industry collaboration in the use of advanced technologies and procedural initiatives to improve air traffic system efficiency [1,2].

The objective of this research is the first step in the specification of a software component for an experimental advanced workstation for use in NASA laboratories that would emulate AOC dispatch operations. The advanced workstation would facilitate development and utilization of a user preferred trajectory (UPT) by each flight. The free flight UPTs would improve flight efficiency by taking advantage of the removal or relaxation of constraints imposed in current air traffic operations. Such constraints range from current procedural requirements which restrict airspace utilization flexibility, including restrictions to allowable routes and altitudes, to data processing limitations of current AOC automation, which may not be capable of fully implementing more-flexible procedures and trajectories.

1.1 AOC Dispatch Operation

The dispatcher's work functions may be distinguished according to pre-departure and post-departure flight phases:

- Flight planning
- Flight following and in-flight replanning

Flight planning focuses on the development of the preferred route, speed, and altitude profile for a flight in accordance with an optimization objective, such as minimizing fuel burn, flight time, or a combination of both, subject to schedule, aircraft weight limits and other requirements. Flight following involves monitoring the progress of a flight with respect to the flight plan, schedule, and safety and efficiency factors, and responding to contingencies and irregularities, which often relate to disruptions due to severe weather.

As shown in Figure 1, the flight planning and following functions require the dispatcher to interact with an extensive information exchange system. The system includes computer processing of static and dynamically updated databases, computerized data input and display interfaces, hard copy documents, and voice and data link communications. The dispatcher workstation accommodates these capabilities.

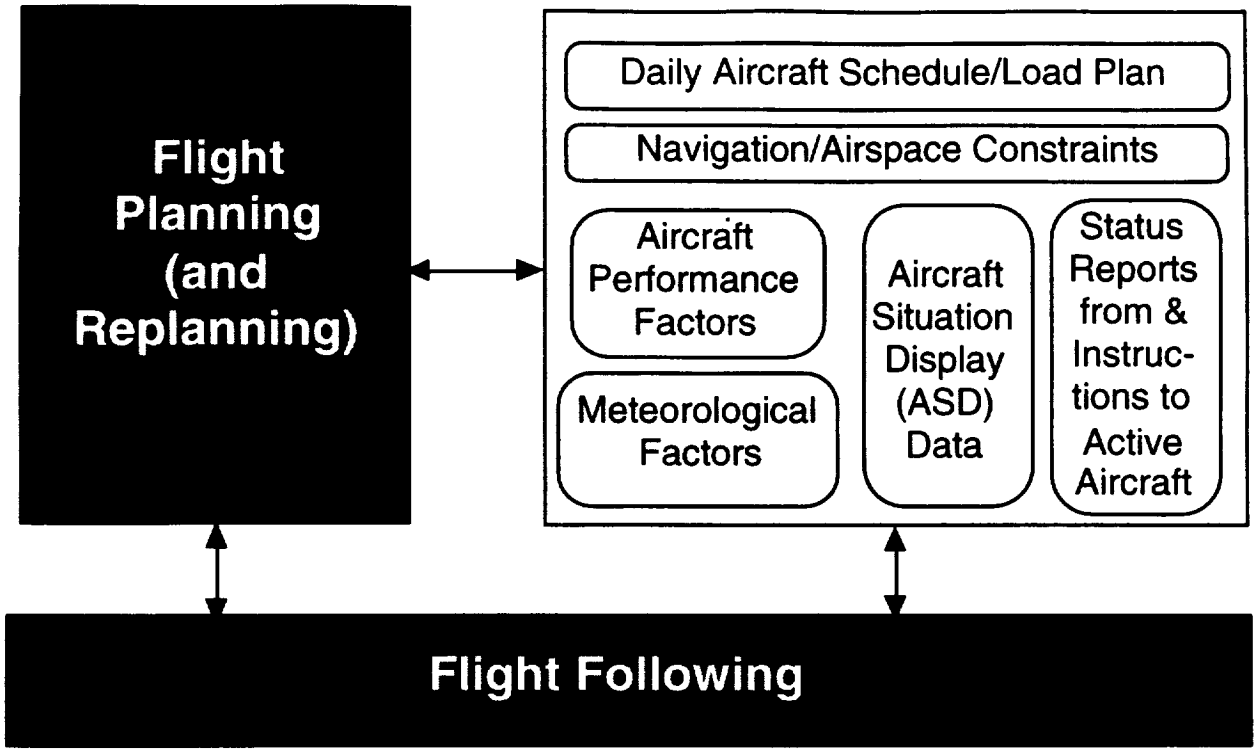

Figure 1. High level dispatch functions and data bases

Selected data processing requirements of the flight planning function are shown in further detail in Figure 2. As presented in reference [3], preparation for flight planning and preparation of the flight plan for release require the dispatcher to assemble, review, analyze, operate on, communicate, coordinate and confirm data concerning:

- Weather and field conditions at departure and arrival stations and alternate airport candidates for en route use;
- Airport and navigation aid system status, including terminal, runway, landing guidance, and navigation aid conditions;
- Aircraft limitations, including minimum equipment list (MEL), fuel load, take-off weight and landing weight constraints;
- Appropriate flight plan optimization objective in consideration of schedule, air traffic control, weather, navigation system, and safety factors;
- Fuel load, route, altitude profile, and airspeed calculations;
- Distribution of flight plan release data, including dissemination of fuel load data to the fueler, maximum payload data to load planning, and flight plan data to the pilot and air traffic control; and
- Finalization of the flight release with the pilot, pertinent company offices, and air traffic control authorities.

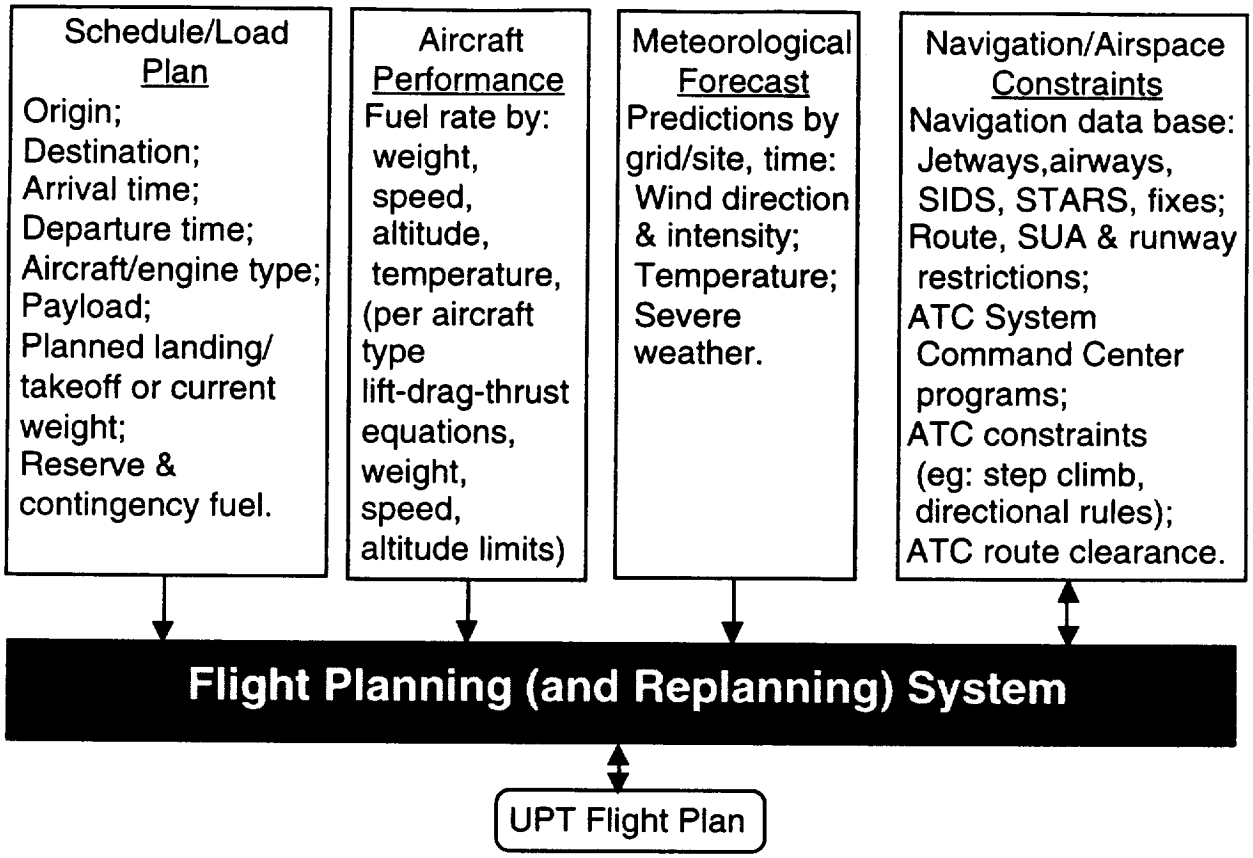

Figure 2. Flight planning: Representative data interfaces

Figure 3 shows further details of selected data processing requirements of the flight following function. Flight following requires the dispatcher to assemble, review, analyze, operate on, communicate, coordinate and confirm data concerning [3]:

- Monitoring of critical event times, including terminal gate pushback, wheels up, en route position reports, expected further clearances in the case of holding or deviations, touchdown, and gate arrival;
- Monitoring the condition of a flight with respect to the flight plan, including review of fuel reports for gate departure, takeoff, en route position reporting, hold entry, top of descent, and arrival events, recalculation of fuel requirements due to route or altitude deviations, and replanning for changing wind, turbulence, icing and other contingencies;
- Distribution of information that might affect flight safety or invalidate the flight release;
- Interaction with ATM automation for en route replanning, change in destination, and arrival scheduling;
- Interaction with flight deck/pilot for status reports and updating of flight plan and weather information; and

- Establishing closure on revisions to the flight plan with the pilot, pertinent company offices, and air traffic control authorities.

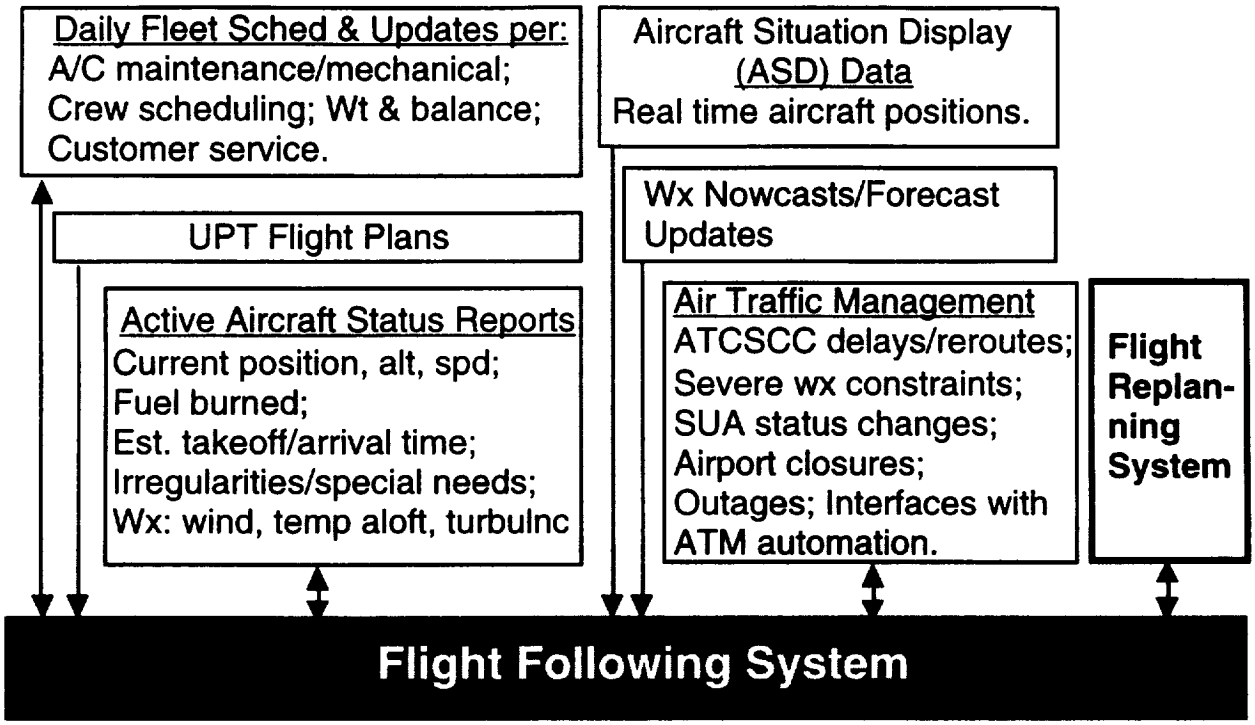

Figure 3. Flight following: representative data interfaces

1.2 AOC System Specification

Figure 4 illustrates the central importance of the flight planning kernel to the overall AOC dispatch operation. As a consequence, this research effort emphasizes initial design of the flight planning function in initiating the specification of software for an advanced AOC workstation. Accordingly, the research has emphasized developing a prototype kernel for an advanced flight plan optimization tool.

As suggested in Figs. 1-4, many other components exist as part of the AOC workstation. These must be specified to enable subsequent design of a comprehensive AOC workstation simulator to support real time research with the dispatcher in the loop. This then is the second emphasis of this work – to describe the various functional components of the AOC workstation to meet free flight needs.

The remainder of this report consists of three additional sections and two appendices:

- Overview of the AOC workstation functional specifications;
- Flight planner kernel description; and

- Recommendations for continued workstation development to support NASA free flight research.

The flight planner section describes our newly-developed prototype for an advanced flight planner. More details concerning its design are presented in Appendices A and B.

The recommendations address additional tasks needed and software data requirements for further developing an advanced AOC workstation simulation to support NASA free flight laboratory experimentation. The workstation software would incorporate flight planning, flight following, communications with ATM and flight deck, data base management and other AOC automation functions.

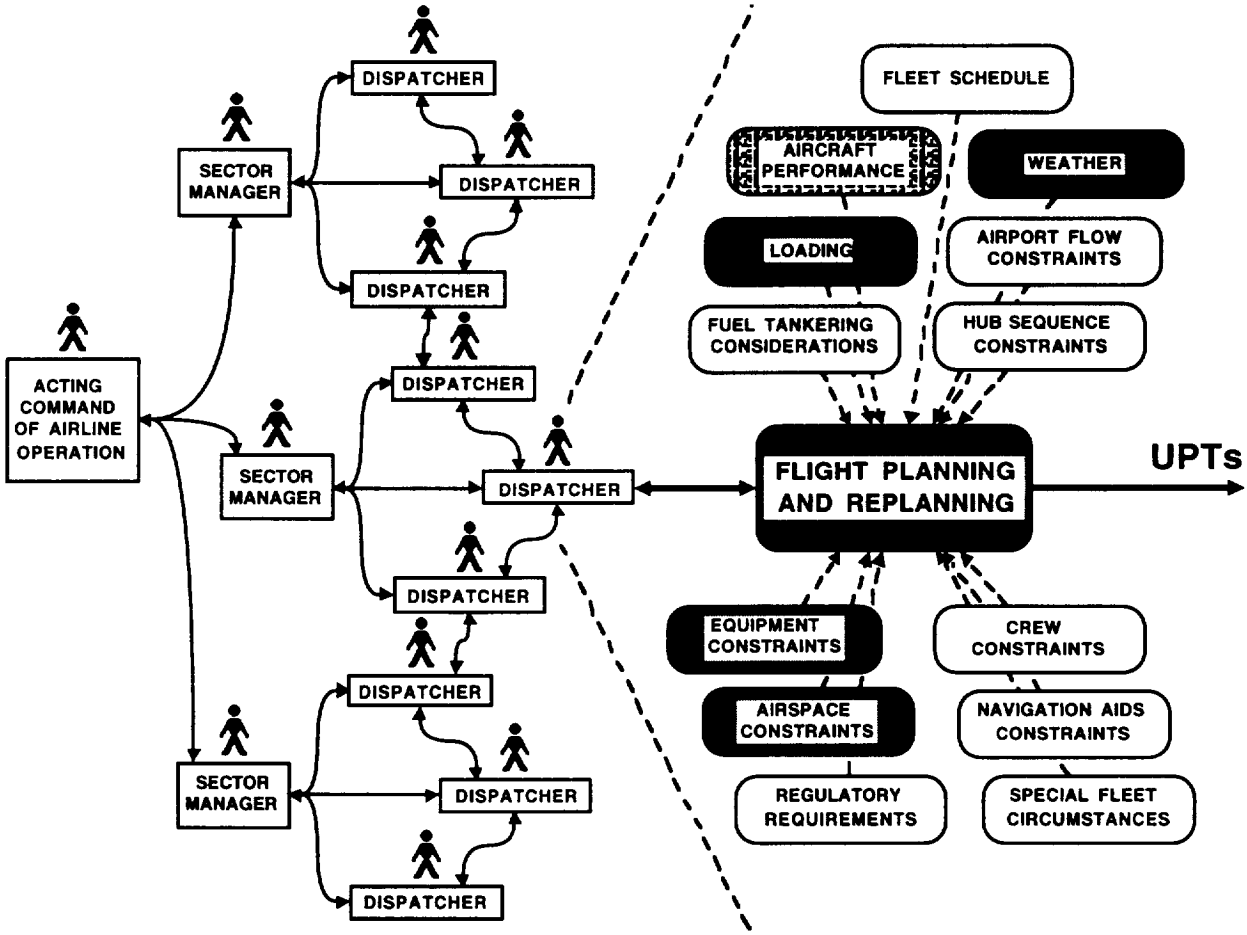

Figure 4. Components of an AOC representative operating structure which interact with the flight planning function

2. Flight Plan Database and AOC Workstation Specifications

We conducted a one-year effort to develop an airline flight plan data base and to specify the requirements for (a) a computer tool for generation and evaluation of free flight, user preferred trajectories (UPT); (b) an advanced flight planning system (FPS) to be incorporated into the UPT-generation tool, and (c) an AOC workstation to house the UPT-generation tool and to provide a real-time testing environment. This AOC workstation will enable including the dispatcher in subsequent real-time free flight experiments at NASA. This work consisted of four tasks which are now summarized. Subsequent work needed to implement these requirements is summarized in Section 4.

2.1 Free Flight Trajectory Data Base

The purpose of this task was to begin to develop a data base of UPTs, sub-optimal flight plans, and the associated data that characterize aircraft type performance and prevailing weather conditions on the days the flight plans are generated. These data were developed at American Airlines using their existing flight planning capabilities. They contain all the operational factors that affect the definition of a UPT for any given day, such as:

1. Flight identification
2. Origin and destination airports and alternative destinations
3. Aircraft Type
4. Planned fuel burn
5. Planned reserve fuel, alternative airport fuel, contingency fuel, hold fuel, added discretionary fuel and ferry fuel load
6. Planned payload, takeoff weight and landing weight
7. Planned flight route and performance by flight segment:
   - Segment end point names (including top of climb, bottom of descent), latitude and longitude
   - Wind vector, temperature, tropopause height and turbulence indicator
   - Flight level
   - Flight course and heading
   - Mach number, true air speed and ground speed
   - Distance
   - Flight time
   - Fuel burn
8. Aircraft weight and payload at departure gate

9. Scheduled gate departure time
10. Taxi-out and Taxi-in time and fuel burn

This data base has the following purposes:

1. These trajectories and the accompanying flight data (weather field, aircraft types) can be used as realistic flight plan input to a flight simulation program such as TAAM for the subsequent study of the interaction of UPTs.
2. The trajectories can be used as models for the desired output of the UPT-generation kernel that was specified and designed under Tasks 2 and 3. That is, the flight plan generation process can be tuned to produce the same trajectories under the same operational conditions.
3. The trajectories can be used to quantify the value, or benefit, of different levels of flight plan optimization (e.g., constrained en route flight levels vs. cruise-climb) as operations evolve to the free flight environment.

We collected a complete set of American Airlines' flight plans for four days of operation. Days included the following:

1. A normal summer day with light winds;
2. A day with severe thunder storm disruption;
3. A winter day with strong west-east jet stream activity; and
4. A winter day with snow storm disruption.

The set of over 10,000 flight plans generated with today's operational constraints serve as the baseline set.

A second smaller set was generated to represent the ideal, or UPT flight plans for these four sample days.

Task 1 Deliverable: The product of this task was the beginning of a flight plan data base that is accessible on a Unix workstation. This data base was delivered to NASA.

2.2 User Preferred Trajectory Generation Specifications

In this task, Seagull worked with the American Dispatchers Federation (ADF), American Airlines (AAL), Delta Airlines (DAL) and NASA and reviewed revenant documentation [4,5,6,7,8] to develop the specifications for a tool to generate and analyze UPTs between any US city pairs. This future computer tool could be used by NASA to:

1. Define collections of UPTs for any given weather condition to extend the database created in Task 1. These data could be used for subsequent free flight analysis by input to TAAM or other suitable flight analysis program.
2. Analyze individual UPTs in terms of fuel burn, time, and direct operating cost (DOC) savings relative to more restricted (less optimal) flight plans.

3. Generate UPT flight plans on a real-time basis as part of a dispatcher workstation simulation that emulates the dispatcher environment and his/her role relative to the pilot and air traffic manager.

Seagull worked closely with American Dispatchers Federation and American Airlines to define UPT-generation tool requirements from the airline operations perspective. UPT-generation is encompassed in the flight planning process; this process uses input from several sources as depicted in the Figure 5 sketch.

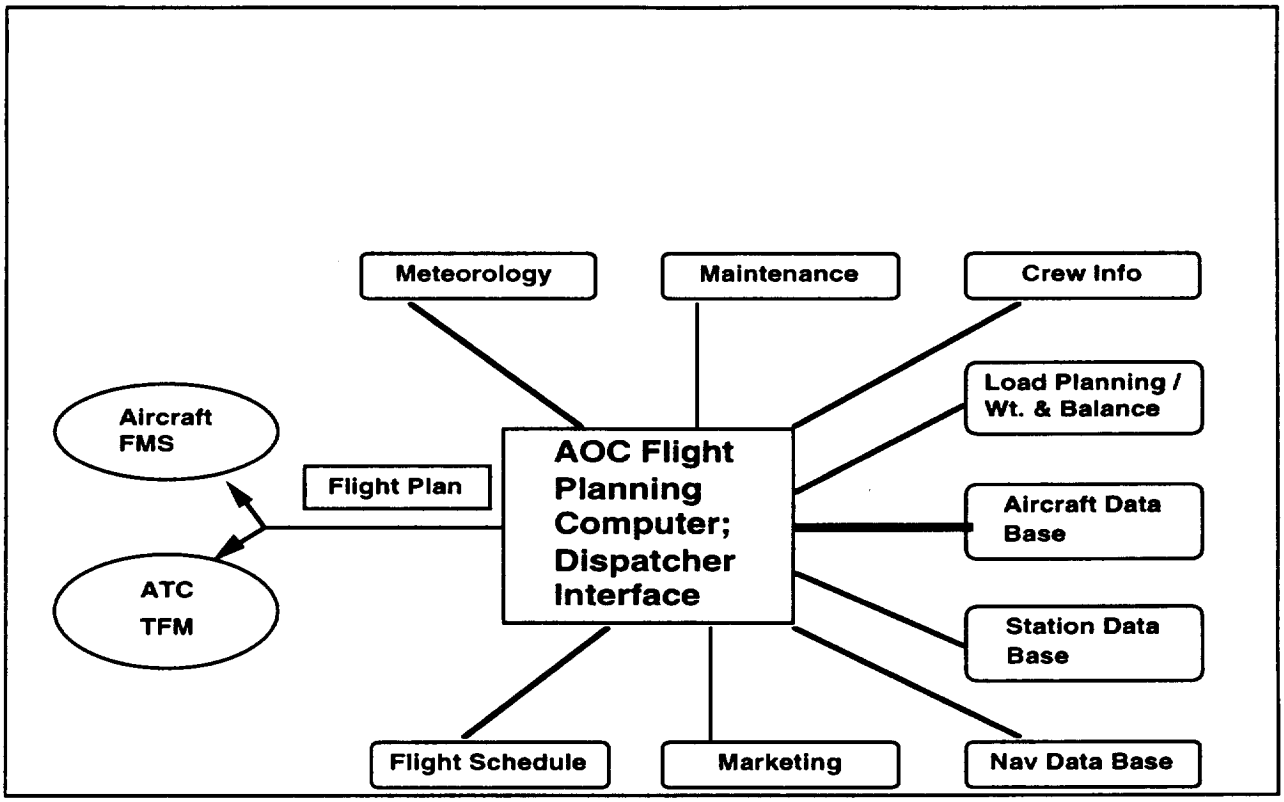

Figure 5. Elements of the flight planning process

Elements of this process that are included in the UPT-generation tool specifications are data bases defining the following:

1. The collection of city pairs that are to be serviced. This includes any airspace constraints affecting flight between these city pairs.
2. The flight schedule that serves these city pairs. Included are the schedule structure of banks of incoming and outgoing aircraft from a large hub operation. This includes the connection priorities of the flights at their various destinations.
3. Data representing the types of aircraft assigned to the schedule. Each type is specified by aerodynamic and propulsion performance characteristics, weight and flight envelope constraints, and passenger/payload data that affect schedule priority.

4. Data representing a spectrum of weather days including summer convective weather and winter snow storm conditions. Also included are the associated wind and temperature fields and movement of jet stream. These data represent the operational scenarios for normal and weather-impacted (irregular operation) days.
5. Data representing other flights (city pairs, schedules, flight profiles) that compete for the same airspace and runway occupancy.
6. Parameters that affect the flight cost that are used to optimize the flight plan. Cost of flight time (including maintenance, crew and depreciation costs), cost of fuel (per airport), and requirement for controlled time-of-arrival are included.

Other input requirements are:

1. Other input that dispatchers and AOC personnel use during the course of their decision making relative to the operation of their flights.
2. The requirements for interface with the aircraft/FMS/pilot and the ATM automation system/air traffic controller/manager. This includes input of data that affects creation of the flight plans and output of data that distributes the flight plan and other information useful to aircraft and ATM automation for facilitating free flight.
3. The options required by the dispatcher for optimizing the flight profile in the process of creating the flight plan that defines the UPT. Options include optimizing fuel load, altitude, speed, horizontal path, time of flight, and their various combinations. This is explained more fully in the next subsection.

ADF dispatchers were polled to expand the details of these specification elements. We defined information and computation facility required to generate flight plans that are both responsive to the UPT requirement and the ease of their generation. AAL and DAL AOC managers were polled to determine that all considerations from their operational control perspective are addressed.

Task 2 Deliverable: The product of this task is the preliminary specification for a UPT-generation tool presented in Appendix A. The specification is an engineering design document that was used to guide the subsequent flight planning kernel software design. The requirements includes provisions to meet the need to preserve the airline flight schedule as well as to optimize the individual flight profiles.

2.3 Optimized Flight Planning Process

In this task, we expanded the specification of the UPT-generation tool to define the details of an advanced flight planning (FP) process. This task expanded the Appendix A specification by creation of a flight planning kernel. Examples of its operation are presented in Section 3. The current FP software design details are documented in Appendix B.

The FP kernel design creates provision for several computational options that can be set to meet particular flight requirements of the UPT. The FP process is structured

so that it calls this primary inner kernel to generate the reference altitude and speed profile, route (the "flight path"), and fuel burn schedule. These are used, in turn, to produce the flight planning output table and dispatch release information.

The FP path generation process is divided into a series of computation steps grouped into modules that are called by the AOC workstation executive. The executive brings in the appropriate data, makes the appropriate computations, and provide the required output for various user requirements. The computational sequence consists of a series of hierarchical steps and options set by the dispatcher's choices and the demands of the FP needed to connect the city pair. Each of these steps is to be implemented as a separate module with well defined functionality and user interfaces.

The FP logic is organized so that the user will be able to test and compare the outcome of using the various options. This will enable comparison to determine (a) the quantitative benefit that a more complex UPT has in terms of saved fuel or direct operating cost (DOC), and (b) the computational penalty (time, memory, etc.), if any, that the implementation of the more advanced method could have on the overall flight planning process.

Some airline routes are pre-defined as a single path between the city pair. Other city pairs have the intermediate options of defining what is the best route (horizontal path) that connects the two given airports based upon the wind conditions, type of aircraft, payload, and route constraints that apply.

The following discusses some of the options that are included in the flight planning process.

Single Fixed Route. For a single fixed route, there is a hierarchy of computation options depending upon the constraints involved:

a. Fixed cruise altitude and speed. These profiles occur when air traffic control and the volume of other flight are such that only very constrained flight profiles can be used. For example, the flight speed and altitude are fixed when flying on one of the North Atlantic Track (NAT) paths. For constrained cruise altitude and cruise speed, there is no option to optimize. However, the FP computation provides the accurate fuel burn requirement and the optimum top-of-descent (TOD) point.

b. Fixed cruise speed; free cruise altitude. Here, the cruise speed is predetermined to be some fixed or scheduled value such as long range cruise speed, maximum speed, or some speed schedule possibly determined by airline fuel policy. A minimum time flight plan is computed using the maximum allowed cruise speed. The optimization process determines the best cruise altitude, where the optimum step climb or step descent points are between multiple cruise altitudes, and the accurate fuel load and TOD point. "Dynamic programming" is the technique that is used to determine the optimum altitude profile.

c. Fixed cruise altitude; free cruise speed. Air traffic control can constrain the altitude level that the aircraft can fly along but leave the cruise speed free. In this

case, the optimization process determines the best speed to minimize overall fuel burn or DOC. Speed will change from segment to segment as the wind vector changes along the route. A local speed optimization method is used to find the best cruise speed along any given segment. "Segment" is used here to represent a section of the cruise path between consecutive waypoints but no longer than some fixed length such as 100 nmi.

d. Free cruise altitude and cruise speed. Here, the FPS computation finds the best flight levels, step climb/descent points, speed along the segments, TOD point, and fuel load as a combined optimization process. The optimization process combines the dynamic programming and segment speed optimization techniques to find the overall best profile.

Multiple Routes. Several levels of sophistication are made available for selection of the 3D optimal flight plan route. The dynamic programming search process determines the best path as is discussed in Section 3. The multiple route options include:

a. Multiple single path routes. Here, several single fixed routes are pre-defined by the AOC that serve as possible paths to transition between the given city pair. The FP generates an optimal trajectory for each of these paths and then rank orders the possibilities in terms of flight cost. Sometimes a non-optimal path might be preferred because it can avoid regions of turbulence or heavy traffic. If five or more possible paths exist, the search begins with the middle path and works outward so that the search does not have to continue search over paths with unfavorable winds. Also, the dispatcher has the option to eliminate some of the paths graphically from the set by examining the overlying wind field and other constraints. In addition, the multiple routes may have several segments in common. The logic is structured so that it does not duplicate computing the flight plan portion over the common segments.

b. Constrained random routes. "Constrained" means that the routes have been defined by an array of pre-defined navigation aid locations, jet or victor route segments, oceanic checkpoints, SIDs or STARs. This array has some geometric boundary used to limit the search, and it is defined by the location of the city pair. This boundary can be thought of as having an elliptical shape with its major axis being the great circle are connecting the city pair. The further apart the city pair, the larger the lateral boundary of the array. The dispatcher has the option of eliminating certain waypoints, segments of routes or airspace within the array to confine the search. The search process has the further option of (a) being constrained to a search over a fixed altitude, or (b) being a full 3D search where the route and profile are determined simultaneously. Option (a) separates the horizontal route search from the vertical profile search. It has a computational speed advantage but often misses finding the true optimal 3D path. However, it is retained as an option to full 3D planning.

c. Unconstrained random routes. "Unconstrained" means that the waypoints that define the route have not been pre-defined but are computed as part of the 3D route search process. These can be considered full free flight trajectories[4]. A

great circle arc is defined to connect the city pair. Elliptical arcs are defined on either side of the center arc such that the distance between the arcs at the midpoint is a specific fraction of the center arc length. Four or more arcs on either side of the center arc (for a total of at least nine elliptical arcs) define the trial paths. Evenly spaced nodes (e.g., at every 50 nmi) are define along the arcs. These nodes define the trial waypoints. These nodes are further specified at 3 or more flight levels to define the 3D grid field for the search. The number of flight levels and their altitudes are a function of the total range of flight. Again, the dispatcher may eliminate regions of this grid network to limit the search. After the search has found the best path through the initial field of node points, a new set of nodes is defined about this path with the lateral boundary limited to the original distance between two adjacent elliptical arc lines. The search is repeated to refine the path. Then, a third set of nodes is defined about the second path with the lateral boundary further limited by the distance between adjacent lines in the second field. The search is again repeated, and this process is continually refined until the 3D path is defined that takes advantage of a finer mesh wind field forecast.

d. Combination routes. Some of the routes may have combinations of the previous options with portions that are fixed to either a single route segment, multiple non-overlapping paths (e.g., the NAT structure, SIDs), a web of overlapping route segments (e.g., the Pacific flex-track system), or constrained random routing (e.g., Miami to Madrid). The flight planning process is designed to sort out this search process to customize the plan to the appropriate constraints that apply. As unconstrained UPT becomes possible, this capability will also be blended in.

Cruise Altitude Optimization. Cruise altitude is optimized with two options: today's flight level constraints and cruise climb.

a. Constrained flight levels. Today's flight level constraints vary over the world's airspace. The constraints may be timed, such as for European flow control or for the NAT entrance; they may be regionally defined such as the US hemispherical rule, and they may have metric units. The FP process determines the optimum altitudes for the different segments of flight and the points where step climbs and descents should be made.

b. Cruise-climb. The other option is to use cruise-climbs for following the best wind profile as fuel is burned. This type of vertical profile is continuously changing in altitude, and it is analogous to the unconstrained random route in the horizontal plane.

Climb and Descent Optimization. Two options are available from which to compute the climb and descent portions of the FP. The accuracy gained here helps define more precisely the fuel burn requirement. The climb/descent profiles can be included in the optimization process. Options include a basic profile computation and climb-descent speed optimization.

a. Basic profile computation. The basic design integrates the climb and descent equations of motion of the aircraft using point mass assumptions and the drag

polars and thrust-fuel flow performance models of each aircraft. This method has been perfected [4] during previous flight management, flight planning, and traffic management system designs. The climb profile is based on using maximum climb thrust and typical airline specified climb speed (Mach-CAS schedule) profile for the particular aircraft type. The descent profile is based on using idle or partial thrust with the airline descent speed schedule. The speed schedules can either be fixed for a given aircraft type or they can be set by the dispatcher.

This method has several advantages over the table lookup method used by current airline flight planning tools:

(a) It accurately accounts for the wind profile at each altitude during climb or descent including effects of turns in the approach or departure route. This is essential to compute an accurate top of descent;

(b) It can take into account local ATC procedures such as stepping up or down to fixed altitude levels during climb/descent, passing over feeder fixes at fixed altitude and speed (e.g., 250 kt at 12000 ft), etc.; and

(c) It provides the option of changing the climb or descent speed schedule and determining the overall effect on the FP time and fuel requirements.

b. Climb-descent speed optimization. This option allows the dispatcher to optimize the climb or descent profile by choice of the speed schedule followed during the profile. Two methods are available – Mach/CAS speed parameter optimization and energy state optimization [4]. This allows a finely tuned flight plan to more nearly resemble the profile that would be generated by a flight management system.

Additional Optimization Options. The 2D or 3D flight plans are designed so that some overall cost parameter can be optimized. If minimum time is desired, then the flight profile and route are selected to minimize flight time between the city pair by flying at the maximum speed possible. If fuel burn is to be minimized, then the flight segment sequence is chosen to provide overall minimum fuel. If minimum DOC is desired, then a "cost function" consisting of the sum of fuel multiplied by fuel cost plus time multiplied by cost-of-time is minimized.

For a hub operation, it is often advantageous to determine the minimum fuel flight plan that also achieves a required time-of-arrival (RTA). This is done by minimizing the DOC cost function with fuel cost fixed and the time cost variable. For this case, the search process is extended to search on the time cost parameter (in this case, an artifact of the optimization process) that drives the flight time to meet the desired schedule.

Takeoff or Landing Weight Constraints. The flight plan profile is constrained by both takeoff and landing weight constraints. For various reasons, the computation of this profile may need to be done in both forward and backward time.

In one case, the landing weight (including dry aircraft weight, payload, and reserve/alternate fuel requirements) are specified. Then, the flight plan is computed

in backwards time starting with this landing weight to determine what fuel load is required for this flight. This produces a required takeoff weight. If this takeoff weight exceeds maximum takeoff weight, then the payload must be limited so that the takeoff limit is met. For this situation, the flight plan must be recomputed in forward time beginning with the maximum takeoff weight.

In another case, the takeoff weight is specified to be maximum so that maximum payload can be carried. The objective is to determine how much fuel is required to carry this initial weight so that the initial fraction of fuel and payload can be specified. This requires first computing the flight plan in forward time. The resulting landing weight must be checked to see that the landing limit is not exceeded. If it is, then the flight plan must be recomputed in backward time to establish what the maximum takeoff weight can be to achieve this landing limit.

Time of Arrival Iteration. As mentioned before, the technique that is used to minimize fuel and simultaneously fix time-of-arrival is to fix the fuel cost and to iterate computationally on the time cost in the optimization cost function for the given city pair. This is done by the executive repeatedly calling the inner reference path generation routine while iterating on time cost until convergence on RTA is achieved. This iteration is straight forward for the single fixed route case described earlier. For the multiple route or random route cases, it requires an extensive amount of computation so that considerable care must be taken in the code design to constrain the computation time to within a reasonable limit.

Random Routing Iteration. As described earlier, the search for the unconstrained random route requires a series of searches through a series of increasingly finer mesh arrays of nodal points. The FP executive computes the points in each nodal array before calling the 3D FP computation to optimize the path through this array. This iteration is repeated until convergence onto the optimal path is reached.

En Route Refile Analysis. The ability to re-compute the flight plan from any point along the flight profile is provided. This capability is used to re-compute the plan if a weather disruption, or bust, is experienced to determine if an alternate destination, refueling stop, or alternate remaining route is required. This application is also used to compute the point of re-dispatch for refiling to limit the reserve fuel requirements. Another application is to use this capability to determine the alternate flight plan in case of degraded engine performance. This includes the engine-out drift down phase if required.

Task 3 Deliverable: The product of this task is the software design for an advanced flight planning kernel that is the heart of the UPT-generation tool. This is described further in Section 3 and Appendices A and B.

2.4 Advanced Dispatchers' (AOC) Workstation

The final task was the development of plans to create an AOC workstation that can be used by NASA as part of future real-time simulations of the free flight environment. The AOC workstation serves to provide the dispatcher environment just as NASA uses cockpit and ATC controller plan view display (PVD) simulators today. This workstation is to be hosted on a Unix machine of NASA's choice. It will contain the UPT-generation tool and advanced flight planning capability described in Tasks 2 and 3. It will also contain provision for the dispatcher to interact with the air traffic controller/manager and ATM decision support tools such as the Center-TRACON Automation System (CTAS).

The AOC workstation also serves two other primary functions that will be required in NASA real-time simulations of the future free flight environment. These are:

1. The communication hub that connects the dispatcher directly to

   (a) the aircraft flight management system (FMS)/data link interface;

   (b) the ATM traffic management unit;

   (c) internal airline data sources such as passenger reservations and aircraft performance records; and

   (d) other external sources of data such as the National Weather Service.

2. The flight following capability that the dispatcher uses to track the flight and interact with the pilot and ATM after takeoff.

Figure 6 is a sketch of the AOC workstation data flow from systems that were developed for the Oceanic Development Facility (ODF) and the Human Factors Laboratory (HFL) at FAA Technical Center (FAATC) [7]. These workstations are being used in real-time experiments by the FAA as part of developing the Oceanic ATM Automation System. The ODF workstation is on an HP machine, and the HFL workstation is on a Sun machine. The dispatcher interface from these designs will be adapted to NASA's needs to address US domestic flight.

As part of this task, we questioned several dispatchers at both AAL and DAL to learn what features they deemed important or would like to have as part of an AOC workstation of the future. As of the time of our visits to the AOCs (Winter 1995-6), the airlines were involved in continuing upgrades to the tools available to their dispatchers. The dispatchers commented that the tools they had were much better than what had been available in years past, and that these tools had dramatically improved the quality of the job that they were able to do.

Typical of the tools that AOC dispatchers now have at each individual workstation are the following.

- Graphical displays of traffic using the aircraft situation display (ASD) feed
- Separate graphical displays of weather
- A text command window for flight plan generation

- A text window for weather reports and forecasts
- Paper printouts of flights to be planned on their shift, on which the dispatchers make notes
- Printed or electronic messages regarding status of airline operations
- A telephone/radio switchboard

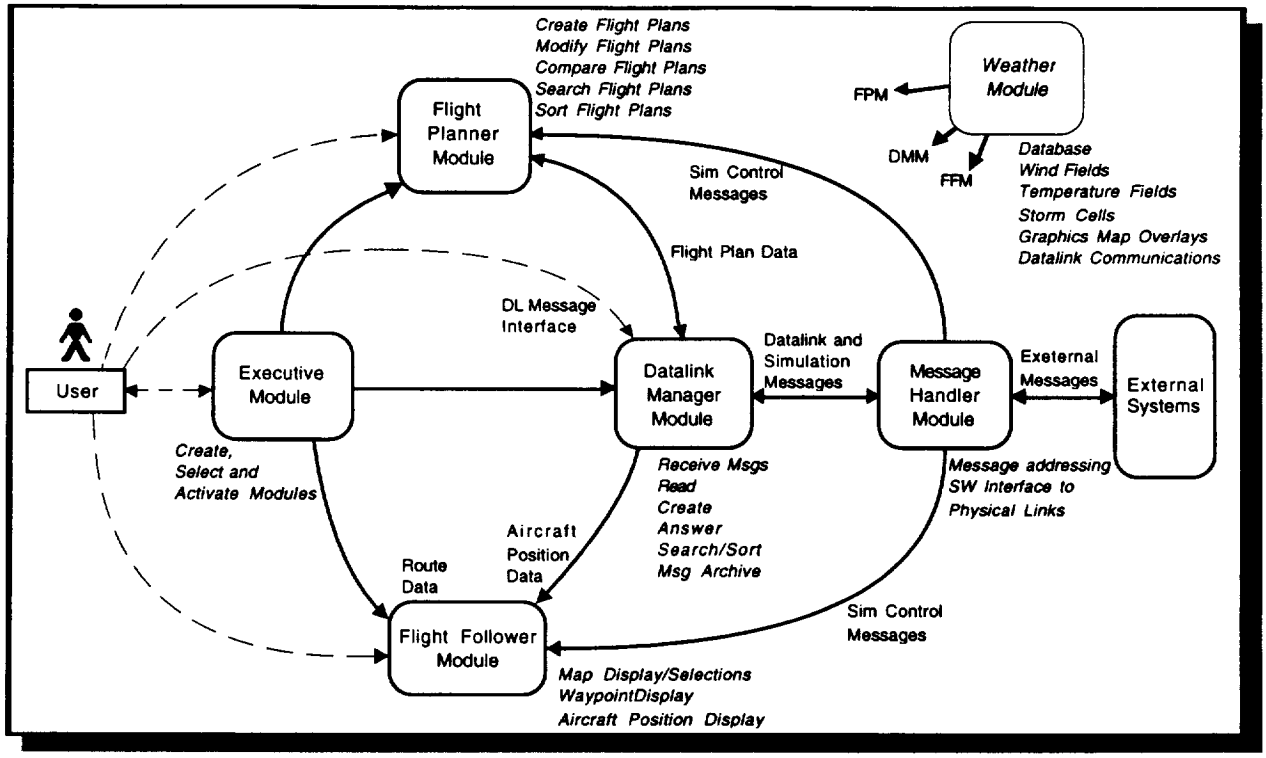

Figure 6. High level AOC workstation data flow

Among the capabilities most frequently mentioned as desirable next steps by the dispatchers (as of Spring of 1996) were the following.

- Graphical overlay of entire planned flight routes for selected flights on the ASD display (in addition to the current aircraft location the ASD provides)
- Selectable overlay of weather (particularly radar weather data) on the ASD display
- Selectable overlay of navigation aids and jet routes on ASD display
- Ability to build routes manually without lots of keystrokes (e.g. with graphical route-building capability)
- Ability to make specific adjustments to computer-generated flight plan with display of re-route impact on fuel and time
- Ability to point, click and drag graphical route depiction to alter the computer-generated flight plan route

- Ability to point and click on aircraft to display ASD data (particularly time en-route and fuel burned so far)
- Convenient flight replanning from arbitrary location to arbitrary destination
- Automatic updates of cargo and passenger loads for the flight from other cooperating airline databases directly into flight planning system
- Ability to choose from a list of flights rather than keying flight numbers in to start flight planning process

In addition, we discussed with NASA personnel the idea of using the AOC workstation to enable the dispatcher to interact with and affect the terminal approach scheduling process now implemented in the CTAS Traffic Management Advisor (TMA) logic. The TMA logic currently schedules approach traffic based on a first come-first served scheme. The AOC dispatcher would often prefer a different approach schedule that favors certain higher priority flights. These might include flights that are running late, flights with more critical connections, or flights with greater payload.

Provision is being made to place a CTAS TMA workstation in the AAL AOC to allow dispatchers to become familiar with the current scheduling process. A feature of the advanced AOC workstation will be to allow the dispatcher to change this schedule according to certain limits and constraints. This will be the subject of early experiments with the AOC workstation in the NASA AATT real time simulation facility.

Task 4 Deliverable: There have been two products that resulted from this task. The first is an the UPT-generation tool preliminary software data specification to incorporate the AOC workstation to represent the functions of a typical US major carrier operation. The second is development of recommendations for future creation for NASA of the advanced flight planner, the UPT-generation tool, and the AOC workstation, with features summarized previously. These are presented in Section 4.

3. Flight Planner Kernel

The prototype for an advanced flight plan optimization kernel has been designed to function in the current operational environment as well as in free flight. The free flight environment would enable greater flexibility in UPT selection based on alleviation of current procedural constraints. The prototype also takes advantage of advanced computer processing capabilities to implement more powerful optimization routines than would be possible with older computer systems.

The key features of the prototype are:

- Iterative, dynamic programming-based trajectory optimization is performed throughout the flight, accounting for horizontal, vertical and speed dimensions;
- Gridded forecasts of winds aloft are interpolated in space and time;
- Rapid optimization is performed using iterative, adaptive search grids;
- Modes of optimization include:
  - Minimum DOC, minimum flight time, or minimum fuel;
  - Forward or backward time processing to allow starting with a fixed takeoff weight or ending with a fixed landing weight;
- Cruise climb or step climb and descent procedures are selectable;
- Airspace restrictions, such as Special Use Areas (SUAs), severe weather regions, or congested areas, are enforceable;
- Mid-flight replanning is supported; and
- Object-oriented software design is implemented.

The features and the operation of the prototype are described in the following sections. The sections provide an overview of the flight planner, and describe the flight time and fuel burn calculation procedure used to model aircraft performance. The companion Appendix A provides a user's guide for program operation and the software structure.

3.1 Flight Planner Overview

The flight planner searches a four-dimensional wind and temperature grid to identify the optimal route, altitude and speed for successive segments of a flight. An iterative process is employed in which a series of trajectories are successively refined until the UPT is identified. Figure 7 is an example of three successively generated origin-to-destination trajectories projected on to the horizontal plane. Each of these three routes has a corresponding optimum vertical profile. Figure 7 also shows the final fourth vertical profile produced by the iterations. Each iteration produces a

segment-by-segment optimum speed schedule The wind field at one cruise flight level is shown for illustration.

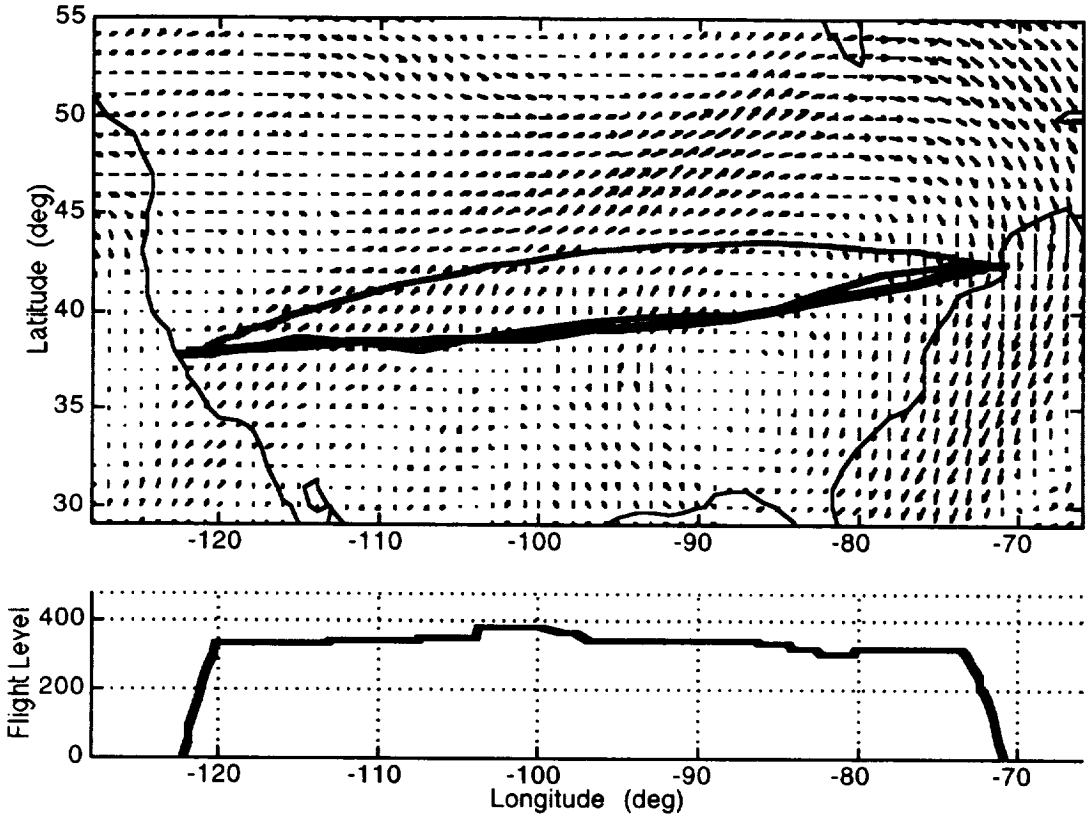

Figure 7. Example Boston to San Francisco free flight trajectory with wind field overlay

The following summarizes the steps of the flight planning process:

1. A weather processor updates and maintains meteorological information. In this version of the software, the weather processor performs a single read in of 18 hours of weather forecast data, which are provided in 3 hour increments. A forecast is re-issued every 12 hours. Other background information is also defined, such as a database of navigation aid identifications and locations.

   After this initialization a central loop of logic is entered. Within this loop a flight plan is requested by the user, an optimal flight plan is generated by the program, and program results are reported to the user.

2. At the start of the main loop, flight plan parameters are specified by the user. A flight plan consists of departure and destination points, a departure and arrival schedule, aircraft type information, aircraft loading and reserve fuel constraints, and cost function parameters. In addition, the user may specify search grid parameters to study program behavior or tune to a desired search characteristic.

Restricted airspaces may be specified to account for special use areas (SUAs) or areas of severe convective weather or other unfavorable conditions.

3. Once the flight plan parameters are specified, the automated search for a best route begins. This is done in an iterative fashion. At each iteration a current best route is used as the basis for building a search grid for determining the next iteration's best route. The horizontal and vertical resolution of the search grid in each iteration is set by user-defined parameters, but the iterative process is designed to use smaller grid point spacings in each successive grid. The initial route would be a sequence of great circle arcs along the great circle from origin to destination. Alternatively, the initial route may be a sequence of great circle arcs starting at the departure point, traveling through any target waypoints specified by the user, and ending at the destination point. Once an iteration's grid is generated, then a dynamic programming algorithm based on Dijkstra's "flooding" algorithm [9] is used to search for the best trajectory, including speed schedule, through the connections in this grid.

   Four iterations are used:

   - The first iteration is a simple vertical profile and speed schedule optimization along the great circle route, which provides a quick indicator of flight time and fuel requirements. For this iteration, a very rough vertical search grid is constructed along the great circle. An example vertical grid is shown in Figure 8 for the Boston-to-San Francisco great circle route. This great circle route is the northern-most track shown above in Figure 7.

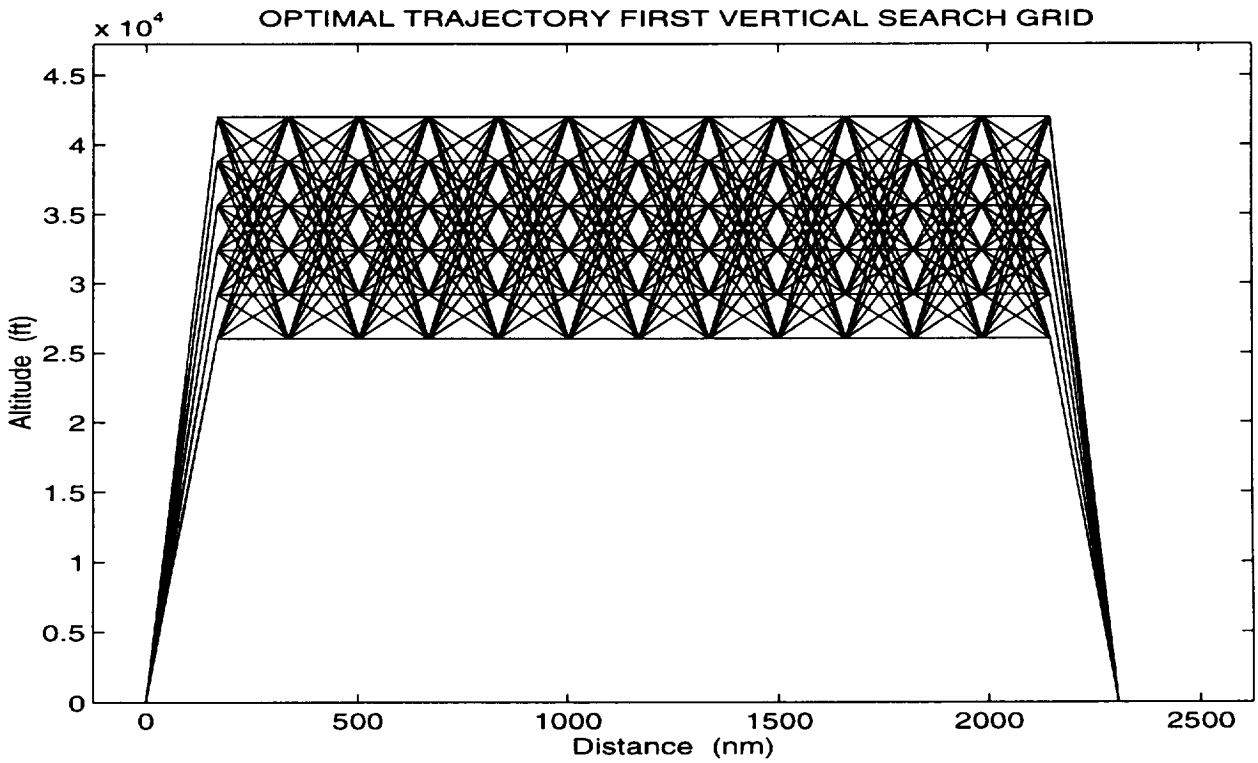

Figure 8. Great circle iterative vertical search grid for Boston to San Francisco flight

- The second iteration constructs a wide network of arcs with fairly rough horizontal and vertical resolution to ensure that all possible reasonable regions for the best route are examined. This wide grid is centered around the great circle-based solution, as illustrated in Figure 9 and 10 for horizontal and vertical components. The wide grid is used to generate the second trajectory solution for horizontal, vertical and speed components.

- The third iteration constructs a finer horizontal and vertical grid, as shown in Figures 11 and 12. This grid is centered around second trajectory solution in order to refine the path in the most promising region. The irregular shape of the horizontal and vertical envelop of this grid conforms to the route and profile of the second trajectory solution. The fine grid is used to generate the third trajectory solution for horizontal, vertical and speed components, which includes the final route solution.

- The fourth iteration constructs a very fine vertical grid along the final route, and performs the final vertical profile and speed schedule optimization. This final vertical grid is shown in Figure 13 for the example flight.

The processes for generating the search grids and performing the trajectory optimizations are described in the following paragraphs.

Grid Construction -- A search grid based on a pre-defined route is automatically generated as follows:

1. The pre-defined route is divided into equal length segments to satisfy criteria based on the number of desired segments and limits on the maximum and minimum length of a segment. The first pre-defined route in the iterative process is a great circle on the earth's surface. Subsequent pre-defined routes are projections of trajectory solutions along the earth's surface, and are used to define the segments.
2. Preliminary grid points are generated along the surface of the earth as follows.
   - A radial vector is constructed from the center of the earth to the endpoint of each segment on the pre-defined route.
   - Each radial is rotated perpendicular to its segment to define an arc along the earth's surface.
   - Grid points are defined at equal distance along each arc along the earth's surface. The distance between these grid points defines the horizontal grid resolution for each iteration, and is dependent on user-specified parameters. The horizontal grid width depends on user-specified parameters, and varies depending on the location of the arc and the strategy for the given iteration. For example, iteration 2 is meant to be wide, and its maximum width might be set to 40% of the total flight length. Iteration 3 is tightly focused, so its width would be much narrower. Iterations 1 and 4 only do vertical optimization, and have zero width.

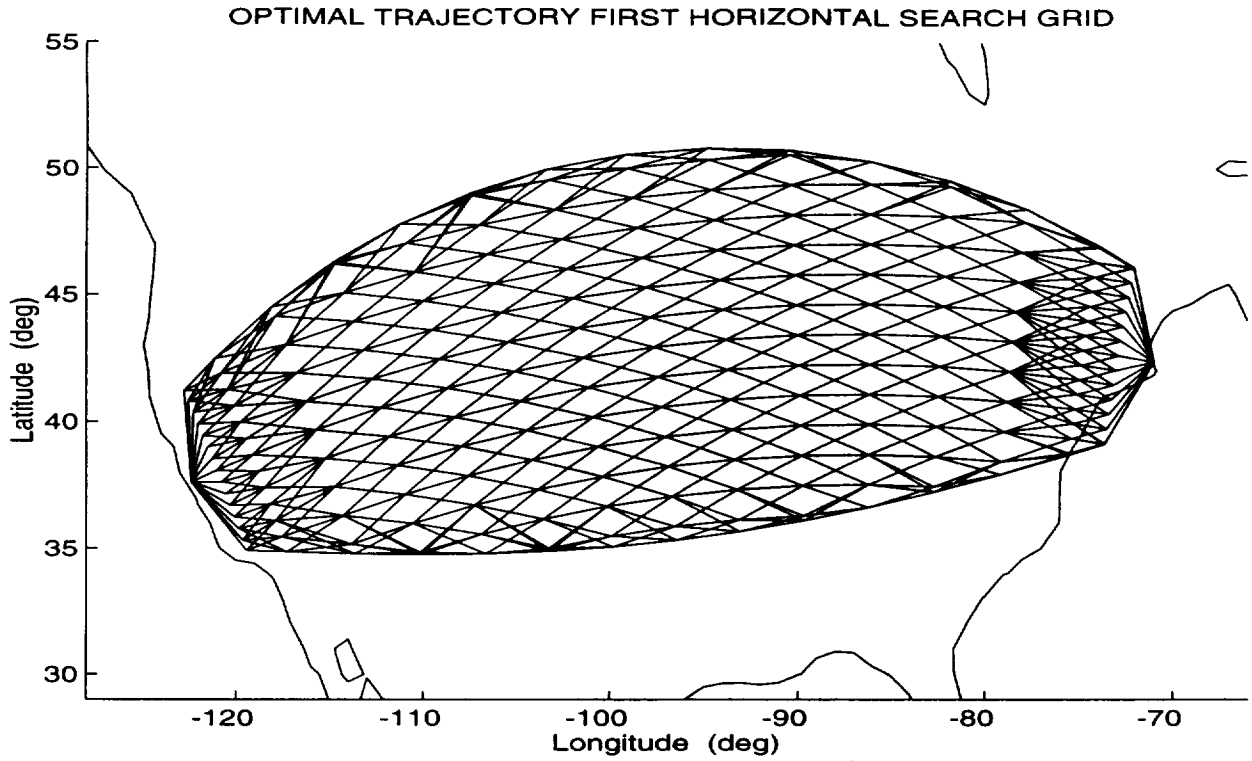

Figure 9. Wide iterative horizontal search grid for Boston to San Francisco flight

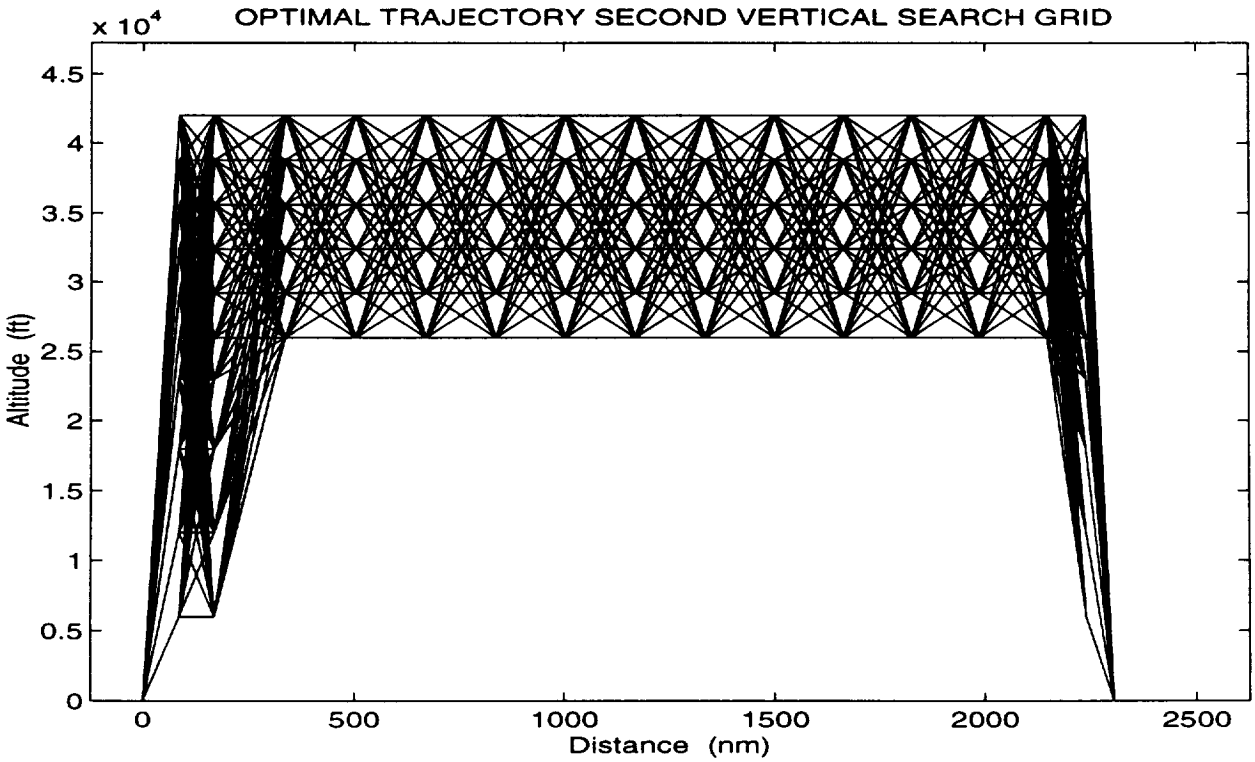

Figure 10. Wide iterative vertical search grid for Boston to San Francisco flight

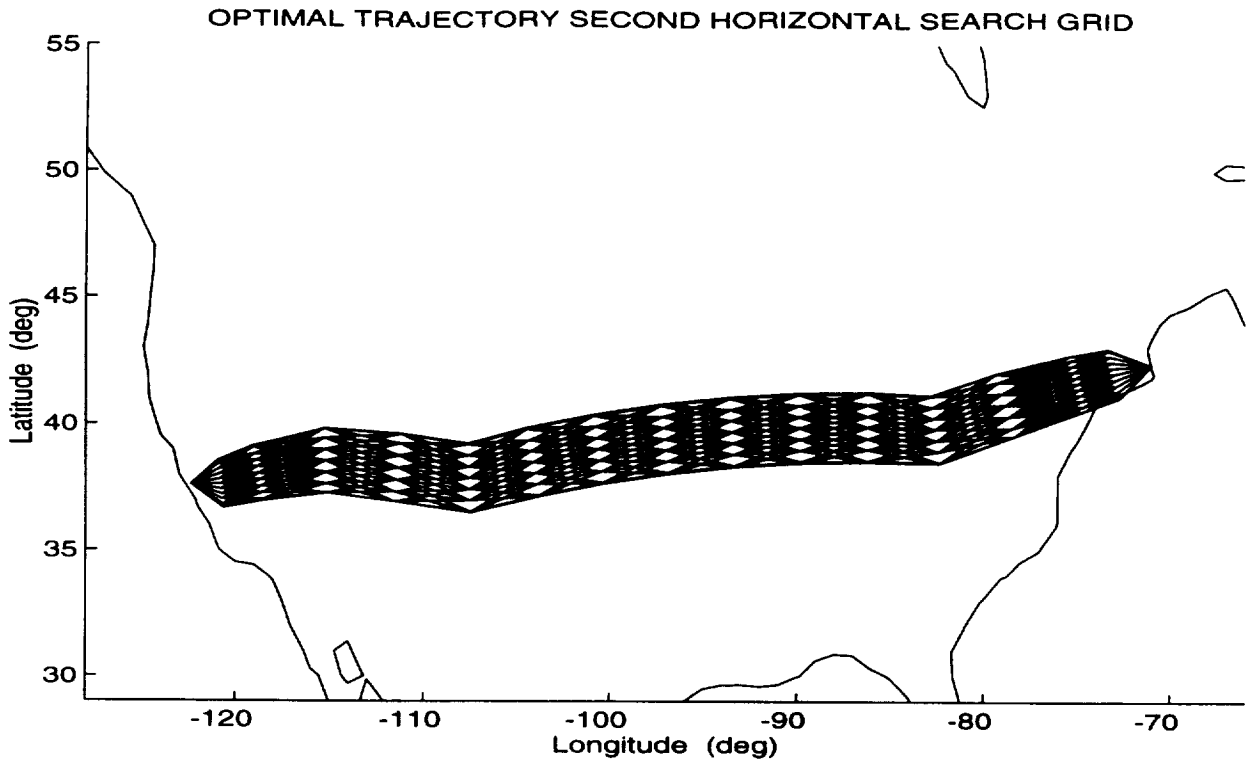

Figure 11. Finer iterative horizontal search grid for Boston to San Francisco flight

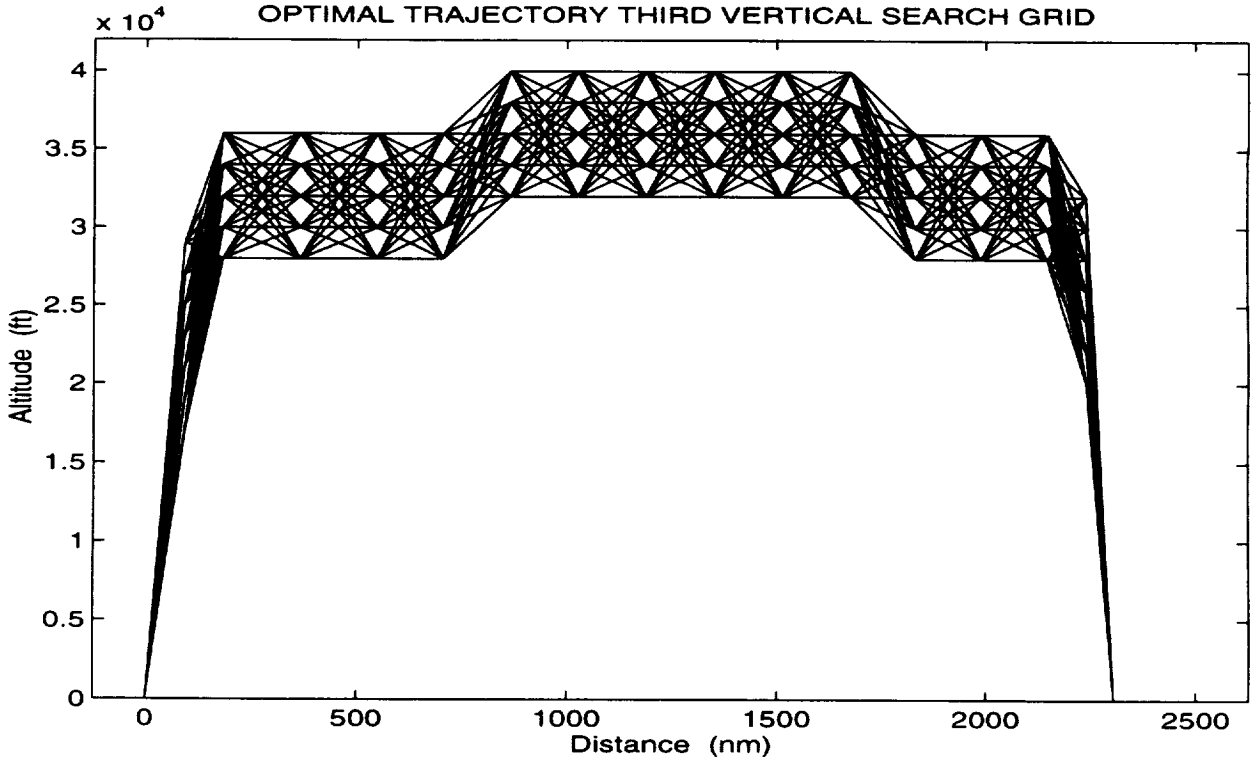

Figure 12. Finer iterative vertical search grid for Boston to San Francisco flight

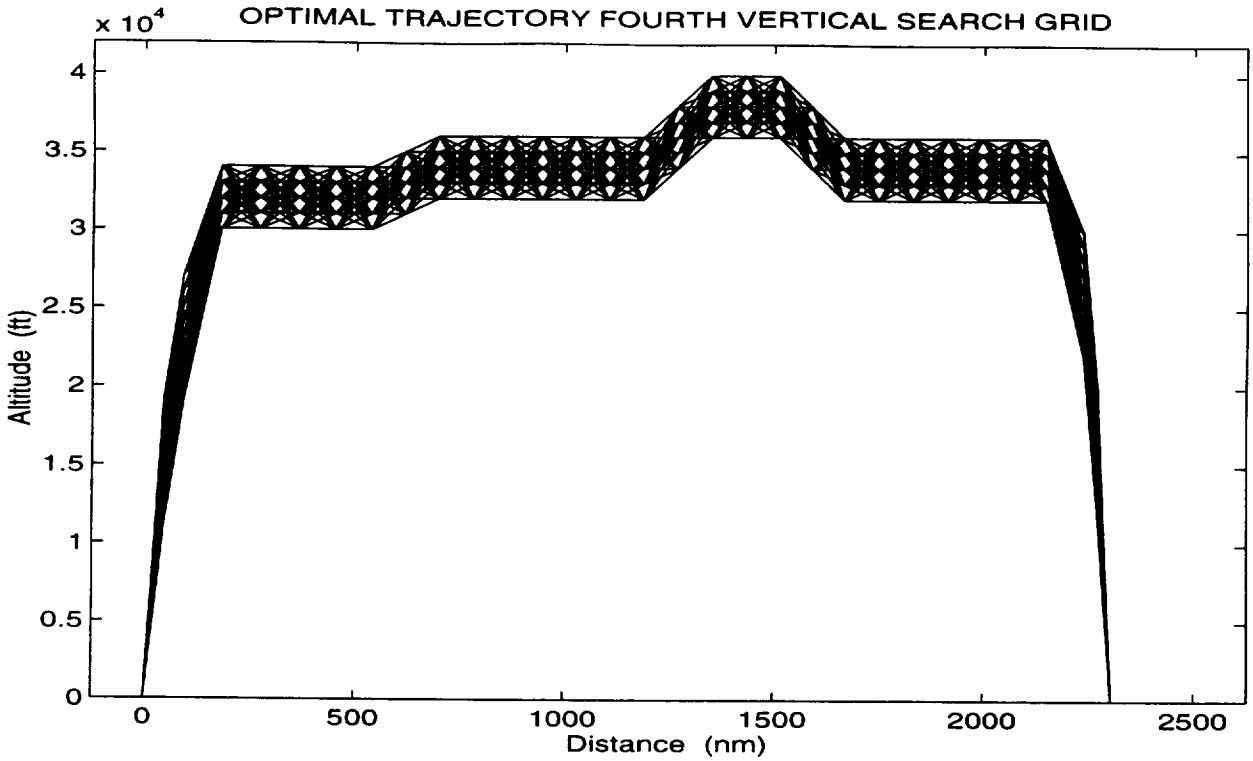

Figure 13. Final fine iterative vertical search grid for Boston to San Francisco flight

3. Airspace grid points are generated as follows.
   - A radial vector is constructed from the center of the earth through each grid point previously defined on the earth's surface.
   - New grid points are defined at equal distance along each radial in a specific range of airspace above the earth's surface. The distance between these grid points defines the vertical grid resolution for each iteration, and is dependent on user-specified parameters. The airspace range is defined by user-specified parameters. Except for the first iteration along the great circle route, the airspace covered along each radial is adjusted to follow the vertical profile of the of the previous iteration's trajectory solution. Those grid points generated by vectors through a common arc on the earth's surface represent a stage in the dynamic programming optimization process. A stage may be conceptualized as a vertical plane containing grid points at specific locations.
4. Two anchor grid points are defined at the origin and destination airports. These represent the first and last stages.
5. Pairwise connections in the grid are made between a grid point in one stage to a grid point in the next stage. These connections may be between two grid points at the same or at different altitudes, and represent level, climb or descent flight segments. The number and scope of connections are based user-specified and default parameters. For example, parameter values may limit the maximum horizontal divergence angle from a route, may limit the altitude range for climb

or descent, or may require one grid point to be horizontally connected to at least a minimum number of other grid points.

Optimization -- Once the grid for an iteration is generated, a dynamic programming "flooding" algorithm is used to search for the optimal minimum cost route through the grid. Cost is based on fuel burn or flight time calculations, or a combination of both. This optimal route can be found searching stage-by-stage forward or backward through the grid. For purposes of discussion, we assume we are searching forward in the grid, and therefore forward in time. The principles are the same searching backward. The concept is to flood outward from the departure grid point, traversing each possible connection only once. As each grid point is reached, only the connection with the lowest total cost to that point is recorded. That connection's cost and state information is also retained. The flooding continues until the destination grid point is reached, at which point the least cost connections are traced back through the grid to identify the entire optimal path.

The evaluation of an aircraft's flight along a connection between two grid points is handled as follows:

1. The approximate times at the current grid point and the next grid point are determined, and linear interpolation in both time and space is used to determine the meteorological conditions (wind vector and temperature) at those times at those grid points.
2. If the current iteration is evaluating a level cruise segment:
   - Linear interpolation is used again to determine the average wind vector and temperature over the course of this segment based on the meteorological conditions at the two grid points.
   - The airspeed which optimizes the cost function is determined by calculating the performance of the aircraft over this segment at various airspeeds during a cost minimization search. A constant true airspeed is assumed. The result is the minimum cost of using this level flight segment and the associated flight time.
3. If a climb or descent applies to the current iteration:
   - The segment is divided into a transition (climb or descent) and a level sub-segment. If the change in altitude is more than 2000 feet, then this maneuver is subdivided into multiple transition sub-segments, each of which does not exceed this limit.
   - A cost minimization loop is used to determine the optimal length of the level sub-segment by varying the length of this segment. This loop performs the meteorological interpolation and optimum speed search described above.
   - The cost minimization loop is used to determine the optimal speed, cost and flight time of the transition sub-section. A constant rate of climb or descent is assumed.

- The segment cost and flight time is calculated by summing the corresponding values of the transition and level sub-segments.

The flight planner software allows the user to identify airspace regions for which the flight is restricted from entering. The program labels the grid points and connections in the restricted airspaces as disallowed for flight planning. Figure 14 illustrates the iterative search grids constructed for selected restricted airspaces and the corresponding route solutions.

After a flight plan is finalized, a selected position on that flight plan may be used to initiate replanning of the flight for updated flight and weather conditions.

3.2 Aircraft Performance Model

The program currently uses a total energy model [10,11] of the aircraft to calculate fuel flow. This model was used because it provided data for several common turbojet transport aircraft.

The aircraft performance model currently used in the optimal flight plan generation kernel is a relatively simple model based on parametric modeling of various aircraft using a total energy model. The model uses parameters published for over 100 aircraft types by the Eurocontrol Experimental Centre in the Base of Aircraft Data Revision 2.4 [10], referred to hereinafter as BADA. Currently, parameters for seven aircraft types are included in our optimal flight planner: Boeing 737-300, 737-400, 747, 757, 767, and McDonnell Douglas DC-9 and DC-10.

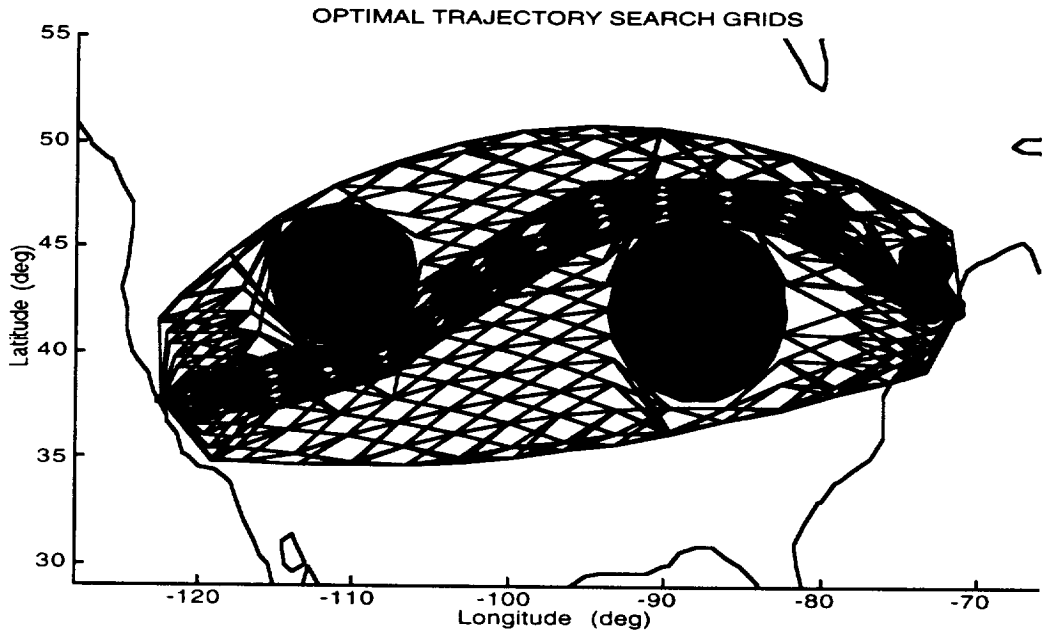

Figure 14. Iterative grids and route solutions for restricted airspaces

The current aircraft performance model is qualitatively accurate and useful for comparative studies of alternative flight plans. However, more accurate

performance modeling would be neces ary if this flight planner were to be used for generating ght plans for commercia ght operations. The most practical approach t ch detailed modeling w ld be to convert from the expression-based modeling t nique currently in use to a set of tabular data from which interpolations could be drawn for specific flight conditions. The impact of such a change on the run-time of the current flight planner would actually be to speed up its performance slightly, since table lookups and interpolations are less computationally expensive than the method currently in use. However, assembling detailed and accurate tabular data for a number of aircraft types is challenging, in that such data are usually generated at great cost by aircraft manufacturers and airlines, and closely held as highly valuable proprietary information. For this reason, the current approach was selected, which takes advantage of publicly available performance parameters for a large number of aircraft.[1]

Implementation -- Within the flight plan optimization program, the aircraft performance model is typically used simply to calculate the fuel flow rate given the aircraft type, weight, pressure altitude, temperature, airspeed and rate of climb. Under certain circumstances, the independent variables specified (e.g. airspeed and rate of climb or descent) may result in a specified boundary condition being exceeded (such as maximum or minimum thrust). For example, a given airspeed and rate of climb might require a thrust greater than the maximum allowed, in which case the calling routine will be informed that this flight condition is not possible. Maximum and minimum thrust levels are parametrically determined for each aircraft type and flight condition.

For some portions of the flight planning algorithm, it is necessary to determine the inverse problem of what true airspeed results from a given thrust and rate of climb for specified aircraft type, weight, pressure altitude and temperature. Because closed form solutions are quite difficult to determine for this problem, an iterative approach is used to determine numerically the airspeed, using the relationship for thrust as a function of airspeed and rate of climb given later.

After the aircraft performance model is used to determine the fuel flow rate for a given flight condition, a cost to fly the segment in question at the specified airspeed is assigned using the cost function

$$C = C_f W_f + C_t t \qquad (1)$$

where

C_f = cost of fuel per unit mass
C_t = cost of time per unit time
W_f = weight of fuel used to traverse segment
t = time spent to traverse segment

For each segment, whether level cruise, climb or descent, the optimal airspeed for the segment is determined iteratively to minimize the above cost function, taking into account the influence of wind for that segment. The minimum cost arrived at

[1] Performance parameters from BADA are available in compressed form via anonymous ftp at bada.eurocontrol.fr. The current revision is contained in bada/2.4/badaRevision2.4.tar.Z.

in this manner is assigned to the particular flight segment, and ultimately forms the basis for the dynamic programming search through the entire search grid to determine the optimal trajectory.

Values for C_t will be different for different aircraft types and dispatcher's objective for a particular flight. The value of C_t should include all costs of aircraft operation directly proportional to time in flight (including flight-time-dependent crew and maintenance costs). A convenient source for recent flight-time-dependent costs of operation is the industry periodical *Air Transport World*[12] which publishes data for different aircraft each month. The value of C_f is simply the cost of aviation fuel per unit weight, recent values of which are available from the same source. Special cases of interest for the cost coefficients include the minimum time trajectory, for which $C_f = 0$, and the minimum fuel trajectory, for which $C_t = 0$.

Modeling Process -- The following is a detailed description of the aircraft performance model currently used. A similar description of this modeling approach can be found in *User Manual for The Base of Aircraft DATA (BADA) Revision 2.4*.[11]

Required thrust is given by

$$T = \frac{mg}{V}\dot{h} + m\dot{V} + D \qquad (2)$$

where

T = thrust,
m = mass,
g = acceleration due to gravity,
V = true airspeed,
h = altitude,
D = drag.

Rate of climb is given by dh/dt in the above equation. Acceleration is given by dV/dt, the rate of change of true airspeed, which is set to zero for each segment of climb or descent.[2] The thrust calculated using this relationship is compared to the maximum and minimum allowable values described later, and if it falls outside these limits, the aircraft performance is limited appropriately.

Drag is represented by:

$$D = C_D \frac{\rho V^2 S}{2}, \qquad (3)$$

[2]In the current implementation, this calculation is carried out assuming nonaccelerating flight, i.e. with the assumption of constant airspeed. This assumption is reasonable for calculating cruise fuel flow, but breaks down when the aircraft is climbing or descending at constant calibrated airspeed (CAS) or Mach number. For these conditions, a correction factor can be applied to account for the rate of change of airspeed implicit in such a maneuver. However, in the current implementation, climbs and descents are broken up into short constant-airspeed segments, so this correction factor is assumed negligible. Airspeed changes are assumed to take place at the boundaries between segments, and there is no fuel cost explicitly assigned to them by the program. This approximation introduces a negligible change in overall fuel consumption for a typical long-distance flight.

where C_D is approximated by:

$$C_D = (C_{D0} + C_{D2}C_L^2)(1 + C_{M16}M^{16}). \quad (4)$$

Hence the left coefficient is approximated by:

$$C_L = \frac{2mg}{\rho V^2 S}. \quad (5)$$

Other coefficients and variables are:

ρ = density
S = reference wing area
C_{D0} = drag coefficient 0
C_{D2} = drag coefficient 2
C_{M16} = Mach 16 coefficient
M = Mach number

In practice, the density is calculated for a given geometric altitude or flight level from a standard atmospheric model with correction for nonstandard sea level pressure and nonstandard temperature. The coefficients S, C_{D0}, C_{D2} and C_{M16} are taken from the BADA tables for the aircraft being modeled.

Minimum and maximum thrust values are calculated by the following expressions. If the thrust calculated using the relationships above falls outside the range given, the flight planner calling routine is informed that the requested flight condition is not feasible.

Maximum Thrust:

$$(T_{max})_{ISA} = CT_{c,1}\,(1 - h/CT_{c,2} + CT_{c,3}\,h^2), \quad (6)$$

$$T_{max} = (T_{max})_{ISA}\,[1 - CT_{c,5}\,(\Delta T_{ISA})_{eff}] \quad (7)$$

where

$$(\Delta T_{ISA})_{eff} = \Delta T_{ISA} - CT_{c,4} \quad (8)$$

and this correction is limited by:

$$0.0 \le CT_{c,5}\,(\Delta T_{ISA})_{eff} \le 0.3 \quad (9)$$

and

$$CT_{c,5} \ge 0 \quad (10)$$

The coefficients $CT_{c,1}$, $CT_{c,2}$, $CT_{c,3}$, $CT_{c,4}$, and $CT_{c,5}$, are from the BADA tables for the aircraft being modeled. ΔT_{ISA} is the difference between the temperature and the standard day temperature.

Minimum Thrust:

For $h < h_{des}$,

$$(T_{des,low})_{nom} = C_{T_{des,low}} (m/m_{ref})^{0.5} (T_{max\ climb})_{nom} \quad (11)$$

For $h > h_{des}$,

$$(T_{des,high})_{nom} = C_{T_{des,high}} (m/m_{ref})^{0.5} (T_{max\ climb})_{nom} \quad (12)$$

and T_{des} is calculated as:

$$T_{des} = (T_{des})_{nom} (M/M_{des,ref})^2 \quad (13)$$

The coefficients h_{des}, $C_{T_{des,low}}$, $C_{T_{des,high}}$, $(T_{max\ climb})_{nom}$, m_{ref} and $M_{des,ref}$ are from the BADA tables for the aircraft being modeled.

With thrust determined using the relationships above, fuel flow rate is given by

$$W_f = \eta T, \quad (14)$$

where

$$\eta = C_{f1}\left(1 + \frac{V}{C_{f2}}\right), \quad (15)$$

and

η = specific fuel consumption
C_{f1} = fuel coefficient 1
C_{f2} = fuel coefficient 2

If the fuel flow calculated from the above relationship falls below a minimum given by the following expression, the fuel flow used is set to the minimum.

$$W_{f_{min}} = C_{f3} [1 + (h / C_{f4})], \quad (16)$$

where

C_{f3} = fuel coefficient 3
C_{f4} = fuel coefficient 4

The coefficients C_{f1}, C_{f2}, C_{f3} and C_{f4} are taken from the BADA tables for the aircraft in question.

In all, nineteen coefficients from the BADA tables are used as parametric inputs to the performance model for the aircraft in question. This level of model fidelity has proven adequate for qualitative assessments of the advantages of our flight plan optimization process.

The details of the flight plan kernel are found in Appendix A. A preliminary listing of data members and objects are presented on Appendix B.

4. Recommendations for Future Advanced AOC Workstation Design

The following recommended tasks and preliminary software data specifications are made to support NASA's development of an advanced AOC workstation simulation capability for laboratory experiments. The workstation would allow operational test and evaluation of alternative procedures for integrating dispatchers into the air traffic management (ATM) decision making process. A dispatcher would operate the workstation during real time experiments. These exercises would support NASA's development of AATT decision support tools, including enhancements for the Center-TRACON Automation System (CTAS) and the Surface Movement Advisor (SMA).

4.1 Tasks

The recommendations assume Seagull's flight planner is installed in one or more Unix workstations at NASA Ames simulation laboratories. The recommended tasks listed in the following are logical extensions to the current flight planner.

1. Expand the current flight planning software to encompass the following:
   a. For specific aircraft types, add fuel burn tables and models where fuel burn is given as a function of airspeed, altitude, aircraft weight, and outside air temperature. Include other constraint data such as maximum takeoff weight, speed and altitude boundaries.
   b. Add ability to store and use incoming GRIB code wind and temperature in a weather data base.
   c. Add the Jeppesen or equivalent navigation data base providing standard instrument departures (SIDs) and arrival routes (STARs), runways, navigation aid locations, restricted airspace, oceanic track entry/exit points, international flight information region (FIR) boundaries, etc.
   d. Add a National Route Program (NRP)-compliant flight planning capability that accounts for SID and STAR requirements and airspace constraints.
   e. Add example OAG schedules for selected city pairs and a few major airlines.
   f. Modify algorithms and code as appropriate to improve flight planning fidelity and robustness.
2. Build a dispatcher front end - graphical user interface (GUI) - to enable efficient flight planning and replanning for an arbitrary pairing of cities, airspace points, or both. Include ability to designate alternates, re-dispatch points, contingency fuel load, multiple sequential flight segments, etc. Also add output so that resulting flight plans are in correct International Civil Aviation Authority (ICAO) form for international flights and FAA form for domestic travel. This includes electronic forms to be sent to both the FAA for

flight filing and to the flight management system (FMS)/pilot for in flight use.

3. Expand the GUI to include an aeronautical chart/map display with input from simulated flights to represent a flight following interface. This includes the option for tags showing aircraft location, velocity, fuel remaining, altitude, expected time of arrival (ETA), etc. Include ability to pan, zoom and reconfigure the region shown on the monitor. Include ability to overlay wind vectors or convective weather cell locations at various flight levels. Include ability to show a single airline's fleet or a composite of all flying aircraft similar to the Aircraft Situation Display (ASD). Include ability to show airport surface map with taxing aircraft and SMA status.
4. Add ability to show dispatcher status of CTAS information such as Traffic Management Advisor (TMA) schedule regarding approaching aircraft within the airline's fleet. Also include ability to show planned departure schedules, planned overflights, and other airline fleet, military, or general aviation aircraft as airspace constraints.
5. Extend the CTAS interface to allow the dispatcher to manipulate the TMA schedule for preferred arrival or departure scheduling.
6. Provide the dispatcher with a group flight planner and assessment capability that is sensitive to traffic loadings at fixes and corner posts, bank integrity requirements and rush (e.g., east-west push) characteristics.
7. Integrate the AOC workstation into the NASA simulation network to enable real time experiments with dispatchers interacting with pilots and ATM - both air traffic control (ATC) and traffic flow management (TFM) - for free flight, CTAS, and SMA testing.
8. Work with NASA and airline personnel to plan meaningful experiments to evolve free flight.
9. Conduct special studies to evaluate the potential benefits of selected enhancements, including top-of-climb (TOC) replanning.

With respect to the TOC replanning special study of Recommendation 8, at least one dispatcher [3] has pointed-out that more efficient flight plans might be produced by an advanced AOC system than most current systems if more accurate departure information is used. The actual payload is not known precisely until terminal gate departure, which is after flight plans are filed. The delay encountered and route or trajectory used during taxi-out and climb are subject to perturbation due to traffic, meteorological and related factors. The resulting travel time, distance and fuel burn during taxi-out and climb cannot be predicted precisely during pre-departure flight planning. The actual location of TOC and the actual weight of the aircraft at TOC may vary significantly from those that were planned. Hence, the optimum trajectory beyond the actual TOC could differ from that of the flight plan.

The amount of the cost savings possible by replanning at TOC is a topic for further research. Most current flight planning systems are not designed to implement

replanning easily at airspace points that are not part of the flight plan, and they would not be suitable for supporting an extensive study of TOC replanning. Replanning is a feature of an advanced AOC workstation and of our prototype flight planner kernel. We recommend using the flight planner kernel and airline historic data describing actual flight plans to evaluate the potential benefits of TOC replanning. Our existing free flight trajectory data base is a candidate source of actual flight plans. The kernel would generate optimum trajectories from actual TOCs using appropriate meteorological and aircraft performance data. These trajectories would be compared with the actual flight plans to assess the worth of implementing TOC replanning.

4.2 Preliminary Software Data Specifications

The advanced AOC workstation would provide the dispatcher with various output features, including:

- Graphical display of traffic using an aircraft situation display (ASD) feed;
- Separate graphical display of weather;
- Text command window for flight plan generation;
- Text window for weather reports and forecasts;
- Text window of airline special reports and data lists;
- Graphical overlay of flight plan route on ASD display with time/fuel burn;
- Graphical weather overlay on ASD display; and
- Paper copies of flight assignments; printed mail.

Input features would include:

- Point-and-drag computer-generated flight plan route;
- Flight replanning from arbitrary location to arbitrary destination; and
- Telephone/radio switchboard.

Voice and data link communications facilities and hard copy documents would also be furnished. It is recommended that the system design be expanded to include these features.

The data items identified in Appendix B represent the initial step in defining a software specification that supports the advanced AOC workstation features. Appendix B lists objects and their data members, and represents one approach for organizing the software. Data members are identified functionally by name, but their data type (e.g., integer, float, char) are to be determined (TBD). The intention is to identify a broad range of data items as a basis for subsequently developing more detailed software specifications.

In the context of a C++ program definition, the more detailed specifications would modify and reorganize the class objects and data members, identify data types and

operating functions, identify class privileges and hierarchies, including inheritance, container and friend relationships, and describe libraries, utilities, and interaction mechanisms. It is recommended that this approach be followed in extending the design of the AOC workstation.

Appendix A -- Flight Planner Kernel Operation

Various user interfaces with computer system are provided to operate the program, set up data, and control output data.

Running the Kernel

The program is started by calling the program name with no arguments.

The first activity visible to the user is a one-time read-in of a weather data set for an 18 hour portion of time.

After the weather data has been read in, the program enters a flight planning loop. For each flight plan, the user is asked for a number of parameters governing that flight plan, and then the corresponding flight plan is generated. Summary information concerning the flight plan is displayed on the user's display, and data files for the current flight plan are placed in the "Results" directory. To end the program, enter "quit" as the starting or ending airport.

Each of the questions has a default value (often indicated in parenthesis) that will be used if the user simply types the "return" key. If at any point you wish to start over filling in parameters for the flight plan, type the "tab" key and then enter return.

In any of the questions that ask for a location there are several forms that the answer may take. The "nav.dat" file contains definitions for names, and these names may be used to refer to the corresponding location given in that file. For example, "LAX" will give you the location of Los Angeles International. Another way to specify a location is to enter the word "point." You will then be prompted with

Please enter Name Latitude Longitude Altitude:

The latitude and longitude are in decimal degrees, and the altitude is in feet. For example, to specify 10000 feet above Logan airport in Boston you could enter

BostonSky 42.3575 -70.9896 10000

After a point has been specified in this way, the name may be used anytime later in the program run just as any pre-defined entry in nav.dat is.

A final method for specifying a location is to use the "waypoint" command. This may be used anytime after the first flight plan has been done. This will cause a list of waypoints to be displayed from the last flight plan and you will be prompted to choose one.

The following is a list of the questions asked of the user to prepare the flight plan.

Please enter starting airport:

Please enter ending airport:

Both of these questions should be answered with a location specification as previously discussed. If the second question is answered with the keyword "extra," you will be prompted for target waypoints to add to the middle of the initial route.

These waypoints are not considered mandatory and are only used for generating the starting route. The final flight plan need not pass through these waypoints.

After these questions have been answered the program will print out the names and positions of the endpoints of the flight plan. Any waypoints produced for a finalized flight plan may subsequently be identified as the starting airport and thereby used as the starting point to replan the flight.

The next question, or series of questions, allows restricted airspaces to be specified for the flight plan. The restricted airspaces are simply modeled as circles in this version of the program, so they are described by a center and a radius.

Please enter restricted airspace center (none):

If a location is specified for the center of airspace then you will be prompted with

Please enter airspace radius (200 (nm)):

to indicated the desired radius. You may specify as many airspaces as you like for a given flight plan.

The next questions concern the plane type to be used for the flight plan.

Plane type is: B757 - Boeing 757

Would you like to change the plane type (y/n)?

If you want to use a plane other than the B757, enter "y" and you will be given a list of plane types that you can choose from. The next area of information and questions concern the cost function used for the flight. A cost model which is a linear cost of time and fuel is used. The current parameters for each of these is listed. For a B757 the current values are given as

Cost per kg of fuel: 0.237

Cost per hour of time: 540

Would you like to change the cost function values (y/n)?

If you enter "y" to this last question you will be prompted for new values for these two parameters. The final area of standard questions concerns the loading of the plane for the flight plan.

Reserve Fuel (kg): 3000

Payload Mass (kg): 14000

Would you like to change the reserve fuel or payload mass value (y/n)?

Answering "y" to the question will provide prompts for new values. Note that the use of the fuel value will change depending on the direction of search for the grid. The default direction is for backward search through the grid. The default direction is changeable as described in the expert section which starts in the next paragraph. For backward search the program provides whatever starting fuel is necessary to arrive at the destination with exactly the specified amount of reserve fuel. For forward searches the program initially loads the plane with the maximum amount

of fuel allowed given the payload loading, and so the reserve fuel specified is essentially ignored.

Would you like to set expert level grid parameters (y/n)?

At this point the user is given the choice of either ending the standard dialogue of flight plan parameters or continuing on with questions that might normally be reserved for expert program users. If you enter "n," then the program will start calculating the flight plan and the results will be displayed. If you enter "y," then the following additional questions will come up.

Grid direction (b/f):

You may choose either "b" for backward or "f" for forward. Depending on this choice your next question will be one of the following two questions.

Arrival time (m/d/y-hr:min (6/30/1996:5:50)):

or

Departure time (m/d/y-hr:min (6/29/1996:12:10)):

The arrival time is asked for in backward search, and the departure time is asked for in forward search. Note that the time must be given with the exact punctuation given in the two examples and with no spaces. The time specified must allow the entire flight plan to fall completely within the time the weather data are available for. In the current case, this means the entire flight plan must occur during the 18 hour interval between 6/29/1996, 12:00, and 6/30/1996, 6:00.

Starting iteration step for step maneuvers (4):

Climbs and descents from one grid point to the next are done in one of two ways. The "steady" approach is to do a straight geometric climb from one grid point to the next with a constant rate of climb or descent for the entire segment. The "step" approach uses either a step climb at the start of the segment or a step descent at the end of the segment -- in both cases the rest of the segment is performed with level flight. If a step maneuver is used, the program not only optimizes over the best possible airspeeds to use for the separate level and climb/descent portions of the flight, it also optimizes over the rate of climb or descent (or equivalently the length of level flight). Since step maneuvers are more computationally expensive and since steady maneuvers provide adequate accuracy for route optimization, normally step maneuvers are only used on the last iteration of the search. If you specify an iteration larger than 4, then no step maneuvers will be used at all in this flight plan.

Minimum climb rate (fpm (300)):

This specifies the minimum acceptable climb rate. All step climbs will have at least this climb rate. During iterations where steady climbs are used, a steady climb will only be used if a step climb with this minimum rate is possible.

Maximum descent angle (degrees (7)):

This is included to keep descents in line with normal commercial flying practices.

Vertical change cost percentage (0.7):

To keep flights from becoming too much of a roller coaster ride, a penalty is assigned to doing a climb immediately followed by a descent, or doing a descent immediately followed by a climb. The cost assigned is equal to this percentage of the last segment cost.

Iteration 4 Vertical Increment (1000 (ft)):

The last search iteration is entirely a vertical profile optimization. This increment specifies the vertical spacing of grid points for iteration 4.

Stage distance scaling (1.0):

This parameter may be used to set the horizontal spacing of grid points. The total width of each stage is always the same. For example, setting the scale factor to 2.0 will double the number of grid points along each grid stage.

Iteration 2 connection angle (30 (deg)):

Iteration 3 connection angle (20 (deg)):

These angles refer to the maximum allowed angle that will be connected automatically. The angle is measured by calculating the angle between this potential segment and the corresponding flight segment of the best known route used to generate this grid.

Maximum segment length (nm (350)):

Minimum segment length (nm (25)):

Target segment count (13):

These three parameters effect the segment length used between grid stages. The maximum and minimum values are rough upper and lower bounds which the program attempts to not go beyond. The target segment count is the desired number of segments to be used. The segment count is used unless using it would cause the segment length to exceed one of the two limits. For iteration 4 the segment lengths are cut in half unless this brings the length below the minimum limit.

Use extra printing (t/f):

This parameter turns on or off the user display of the individual segment flight plan information.

Most of the flight plan information displayed to the user is self explanatory. If "extra printing" is left on, then information from each of the four grid search iterations is printed out after it is calculated, with summary information always available at the end.

A segment may be flown using more than one maneuver. A maneuver is a portion of the flight flown at a constant true airspeed and with a constant rate of climb. The maneuver information is given on lines that start with "M" followed by the maneuver index. The information describing a maneuver is of the form

M1: A nm, B kg/m, C%, D kg, E fpm, F(G)(H) kts, I -> J

where "A nm" is the ground distance covered, "B kg/m" is the fuel flow rate, "C%" is the percent of maximum thrust, "D kg" is the average mass of the plane, "E fpm" is the climb or descent rate, F is the true airspeed in knots, G is the airspeed in mach units, H is the ground speed in knots, I is the starting altitude in feet, and J is the ending altitude in feet.

Creating a Weather Data Set

The raw weather data we have been using has been from the publicly accessible internet site "nic.fb4.noaa.gov." On this site the directory "/pub/avn" contains two subdirectories "avn.00z" and "avn.12z" which contain aviation related weather information updated each day at 0 Zulu and 1200 Zulu, respectively. In these directories are many data files. The files of interest have names of the form gblav.TxxZ.PGrbFxx, where TxxZ is either T00Z or T12Z and Fxx ranges from F00 to F72 by 3's. These files are in GRIB format (GRIdded Binary) and use National Environmental Modeling Center (NEMC) grid #3 layout -- the data is given in a global 1 degree x 1 degree (lat/long) format. There are quite a few data fields available in these files. The data fields we use are the temperature ("TMP"), geopotential altitude ("HGT"), north wind component ("VGRD"), and east wind component ("UGRD"). This data is given at the 16 mandatory pressure levels ranging from 1000mb to 10mb. Each file represents a weather forecast offset from the given base time -- for example, gblav.T12Z.PGrbF09 is the 2100 Zulu forecast given in the 1200 Zulu data set. So, the first step is to transfer the GRIB files gblav.TxxZ.PGrbF00 through gblav.TxxZ.PGrbF18 to your site.

The NEMC has provided a program called "wgrib" for decoding and handling the data in a GRIB file. We have successfully compiled and run this program on our Sun and NT boxes. The awk script "grib.awk" can be used to filter the GRIB file to reduce it to just the data of interest.

The following is the contents of the file "grib.script" which is a Unix shell script for performing this procedure:

```
wgrib.sun -s $1 | awk -f grib.awk | wgrib.sun -GRIB $1 -i
```

(note that wgrib.sun is the Sun executable of the wgrib program)

When grib.script is called with a filename of one of the NEMC raw data files then a file named "dump" is produced, which is a new GRIB file containing winds, temperatures, and altitudes for just the pressure levels of interest.

The executable "WxPreprocess" is then run on each of the resulting "dump" files to produce a weather data file in the form used by the flight planning program. This program is called with one argument, the name of the dump file. The output file is automatically named using the time of the weather forecast. This output file should be placed in the "wx.dat" subdirectory.

Since the data files used by the flight planning program are in binary form, and since they are read into the program with essentially no processing, they are dependent on the byte ordering of binary data for the machine they are produced on as well as the

byte ordering of the machine the flight planning program is run on. If the data files need to be moved between a little endian and a big endian machine, the executable "wx_swap" should be used for the conversion of all of the weather data files before running the flight planner. The program wx_swap is called with two arguments -- the first is the name of the input file to be converted, and the second is the name to be used for the output file.

Flight Planner Graphical Display Utilities Description

The current optimal flight plan demonstrator produces output files at run time that can be post processed to produce graphical displays of the search grids used, the horizontal and vertical profiles of the selected 3D optimal trajectory, and any restricted airspace zones enforced by the user.

Scripts have been written to enable graphical depiction of these results using Matlab (a commercial software package produced by The MathWorks, Natick, MA). In order to use these scripts, the user must have a copy of Matlab running on a computer on which the flight planner output files are available, and it is most convenient if this is the same computer that the flight planner itself is being run on.

The following descriptions of the Matlab scripts are intended as a brief reference guide to the available display utilities. A future version of the flight planner would likely have its own built-in graphical user interface, which would include many functions not provided by the current Matlab display scripts.

Preparing to Display Results

Prior to using the graphical display scripts, copies of the scripts should be placed in the "Results" directory under the directory in which the flight planner executable resides. Alternatively, if Matlab is not available on the same computer on which the flight planner will be run, the scripts should be placed in the same directory as result files generated by the flight planner.

Finally, Matlab should be run from within the same directory, and the command

load topo

should be issued. This command loads a Matlab-resident database of Earth topography to enable display of background coastline maps.

To ensure that Matlab's current working directory is the same directory within which the display scripts reside, issue the command

help vert34

If Matlab cannot find the file and display the help lines for it, then it will be necessary to change Matlab's working directory before proceeding. If other methods for doing this fail, as a last resort one can open the M-file "vert34.m" using "Open M-file" under the "File" menu in Matlab; then select the "Save and Execute" option.

Matlab display scripts

The three types of graphical displays generated by the current Matlab scripts are listed below with the scripts that generate them.

1. Interim and selected routes overlaid on wind vector field, plus vertical profile of selected trajectory

s.m	No restricted airspace shown
sas.m	Restricted airspace shown

2. Interim and selected routes overlaid on last two search grids

g.m	No restricted airspace shown
gas.m	Restricted airspace shown

3. Vertical profile of search grid

vert1.m	Exaggerated vertical profile of search grid 1
vert2.m	Exaggerated vertical profile of search grid 2
vert3.m	Exaggerated vertical profile of search grid 3
vert4.m	Exaggerated vertical profile of search grid 4
vert34.m	Exaggerated vertical profile of search grids 3 and 4

In order to execute any of these display scripts, simply type the name of the script within the Matlab window after the instructions above have been followed, leaving off the ".m" extension. For example, to show a display of the interim and final horizontal routes selected by the flight plan optimizer on a background of wind vectors, along with an exaggerated vertical profile of the flight trajectory, one would simply type

s <return>

at the Matlab prompt.

The graphical results shown by these scripts are useful for engineering evaluation of the performance of the optimal flight planner. They are not intended to exemplify the type of display that would be most useful to an airline dispatcher.

Software Structure

The flight planner software is modularly structured using C++ object oriented programming. The software consists of database and processing files, developed from a set of computer program class objects.

Files Inventory

The data processing files are organized as follows:

1. Executable File -- named "fp.exe," "flight_plan" or otherwise, as appropriate

2. Navigation Data File -- named "nav.dat." This file should be in the same directory as the executable. This contains a list of place names that can be used

for location references in the main program. The format for each line in the file is:

<place name> <latitude> <longitude> <altitude> <anything else ignored>

The place name cannot contain any spaces or tabs, but is otherwise unrestricted. The latitude and longitude should be given in a decimal number of degrees. The altitude should be given in a decimal number of feet. Anything after the altitude is simply ignored.

3. Weather Data Directory -- named "wx.dat." This directory should be a subdirectory of the one containing the executable. The data files in this directory should have names of the form WX.<year>.<2 digit month>.<2 digit day>.<4 digit hour>.F<hour offset>, for example "WX.1996.06.29.1200.F09."

4. Results Directory -- named "Results" to receive various output files produced by the flight planning program. This directory should be a subdirectory of the one containing the executable. The files in this directory are overwritten during the production of each new flight plan during the running of the program. The files that are produced by the program and installed in this directory are as follows:

4a. Airspace.dat -- This file contains a geometric description of any restricted airspaces specified by the user for the current flight plan. If none are given the file will be empty. For each restricted airspace there is a header followed by data. The first line of the header consists of two numbers -- the first is the number of lines in the header and the second is the number of lines of data. Currently the rest of the header should be ignored. The data lines each consists of a pair of latitudes and longitudes. When connected up these locations trace out the boundary of the given restricted airspace.

4b. Link <n>.dat (for n = 1, 2, 3, and 4) -- These files list the links used in the horizontal version of the grid for iteration n. Each line in a file describes one link. There are four numbers in each line -- the first pair of numbers are the latitude and longitude of the start of the link, and the second pair of numbers are the latitude and longitude of the end of the link.

4c. N ode <n>.dat (for n = 1, 2, 3, and 4) -- These files list the nodes used in the horizontal version of the grid for iteration n. Each line in a file describes one node. Each line contains the latitude and longitude for a given node. Note that the first and last nodes are each given once at the beginning of the file. After those two listings all of the nodes (including the first and last nodes) are given.

4d. Vert <n>.dat (for n = 1, 2, 3, and 4) -- These files list the altitudes used in the spatial grid for iteration n. Each line gives one altitude used by a stage in the grid. A line contains two numbers -- the first is the geometric distance of a representative node in the stage from the start of the grid, and the second is the altitude in feet that is used to generate the grid for that stage.

4e. Route.dat -- This file contains information on the best routes found by each iteration of the program. Each line in the file contains 11 numbers that a waypoint along the route. The first number is the iteration number, the second is the waypoint index, the third is the waypoint latitude, the fourth is the waypoint longitude, the fifth is the geometric altitude in feet, the sixth is the pressure altitude in 100's of feet of one end of this flight segment, the seventh is the time in seconds taken so far in the flight plan at the start of this segment, the eighth is the mass in kilograms of the fuel used so far at the start of this segment, the ninth is the true air speed in knots, the tenth is the mass in kilograms of the plane sometime during this segment, and the eleventh is the direct geometric distance in nautical miles from the start of the flight.

4f. Wind.dat & Wind.<date>.<altitude> -- The current pressure altitudes (in feet) used for these output files are 30106, 34052, 38730, and 44739. The data in each of these files give the wind information for the given pressure altitude for the given date. Each line in such a file consists of four numbers. The first is the latitude and the second is the longitude. The third and fourth numbers are the east and north wind components respectively given in meters per second.

Modules and Classes

The code objects and their file locations are organized as follows:

1. Airspaces -- The source code for this is contained in the files: Region.cpp, Region.h, RestrictedAirspace.cpp, and RestrictedAirspace.h.

1a. Regions

- BoundarySegment

This class inherits from the Arc geometry library class. In addition to the normal Arc information, it also has member information for the normal of the plane which goes through the center of the earth and the two Arc endpoints. This plane can then be used as part of a quick test to see whether a point on the small circle that contains the Arc is actually on the Arc.

- BoundingCircle

This class represents a spherical small or great circle. Its principle methods are PositionIntersectsBoundingCircle and ArcIntersectsBoundingCircle. BoundingCircle objects are used to provide simple, approximate boundaries for regions, and these two methods provide quick tests for deciding whether further investigation is needed when looking at whether a Position or Arc intersects a region.

- ConvexRegion

ConvexRegions are areas on a sphere that are describable as having a boundary made up of a convex collection of BoundarySegments. It has

associated with it a BoundingCircle used for quick estimates of its size. Its principle methods are PositionIntersectsConvexRegion and ArcIntersectsConvexRegion, which are used for determining whether a position or arc intersects the region.

- Region

Regions are made up of a collection of disjoint ConvexRegions. It has associated with it a BoundingCircle used for quick estimates of its size. Its principle methods are PositionIntersectsRegion and ArcIntersectsRegion, which are used for determining whether a position or arc intersects the region.

1b. Restricted Airspaces

- RestrictedAirspace

This class inherits from the Region class. Its principle methods are PositionIntersectsAirspace and ArcIntersectsAirspace. These methods are for determining when grid points or flight paths go through a restricted airspace.

- RestrictedAirspaceList

This class provides simple methods for handling lists of restricted airspaces.

2. FlightPlan -- The source code for this is contained in the files: FlightPlan.cpp and FlightPlan.h.

- Schedule

This class provides simple schedule data. Currently included are the starting and ending location, departure and arrival time, and the plane type.

- RouteRecord

This class is designed to hold the summary data for the best route found during each iteration of a flight plan. It contains the node and link count for the associated search grid, the total ground distance, the total flight time, the total fuel consumed, the total cost of the flight, and the elapsed computation time to do all of the calculations for this iteration.

- UserGridParameters

This is a simple class designed to communicate user parameter selections to the rest of the program.

- FlightPlan

This class brings together all of the information needed to create, analyze and generate a flight plan. The data associated with this class is: a Schedule, PlaneData for the plane to be used, WeatherService, the NavAidData database, the starting route (usually a great circle), an output file for route data, RouteRecords for each iteration of the flight plan search, and a

RestrictedAirspace list. This class has a number of significant methods associated with it:

CreatePlan -- This gathers together the data needed to create a new flight plan. The vast majority of this effort is receiving flight plan parameters from the user and filling in associated information once the parameters are selected.

CalculateRoute -- This is the main calculation loop of the program. It controls the four iterations of the best route search. It starts out with a best known flight route which is the great circle route. For each iteration it first calculates the grid parameters to use by calling Set_Grid_Rules, it then subdivides the best known route using Subdivide_Ground_Flight_Route, uses the best known route to make a horizontal grid with Make_Horizontal_Grid_From_Flight_Route, makes a spatial grid from the horizontal grid using Make_Spatial_Grid_From_Horizontal_Grid, finds the best route in the spatial grid using FindBestFlightRoute, and finally displays and records the iteration information using PrintFlightPlanSummary.

Set_Grid_Rules -- This determines and sets all of the grid spacing, width, and height rules and parameters for generating horizontal and spatial grids.

FindBestFlightRoute - This performs a flooding search throughout the spatial grid for the optimal flight route in that grid. This floods out from one end of the grid one stage at a time. For a given node in a stage it looks at all nodes in the previous stage that are connected to it. For all those nodes it calculates the total cost of arriving at the current node by flying through the node in the previous stage. It then compares all of these costs and records the single, lowest cost node in the previous stage. When it finishes this flooding out through the grid, it then finds the best route through the grid by starting at the last endpoint node and backtracking along best connections through the graph to the other end of the grid.

PrintFlightPlanSummary -- This displays and records summary information about each flight plan iteration after the calculation is done. If user display printing is enabled, then flight plan information is displayed to the user for each maneuver for each waypoint in the route. It also puts a subset of the information about each waypoint into the "Route.dat" data file. It records basic iteration summary information into a data structure for later display by the Report method at the end of the entire flight plan calculation. Finally, it records altitude information about the spatial grid into the "Vert_<n>.dat" file.

Report -- This displays to the user the final summary information after all of the iterations have finished and the flight plan work is complete. The summary information is brief information concerning the results of each iteration. If the flight plan exceeds loading limits for the plane an error

message will describe the problem and display the values of related parameters.

3. Grids -- The source code for this is contained in the files: DirectVEGraph.cpp, DirectVEGraph.h, fp_grid.cpp, fp_grid.h, and GridRules.h.

3a. Graph --The graphs used here are directed graphs with explicit vertices and edges as objects in their own right (in contrast, say, to having the edges implicitly recorded by having a list of connected vertices).

- Vertex<VertexData, EdgeData>

These objects maintain a list of the edges coming in and a list of the edges going out from this vertex. In addition, an object of type VertexData is part of the object, and there are simple setting and getting methods available for accessing that data.

- Edge<VertexData, EdgeData>

These objects contain a reference to the vertices at their starting and ending endpoints. In addition, an object of type EdgeData is part of the object, and there are simple setting and getting methods available for accessing that data.

- DirectVEGraph<VertexData, EdgeData>

An object in this class contains a set of Vertex's and a set of Edges. There are basic methods for adding and removing a Vertex or an Edge from the sets. Since these sets tend to have thousands of elements, certain allowances have been made for the sake of efficiency for our particular application. In particular, these sets are maintained as simple vectors and no duplication checking is done when a Vertex or Edge is added.

3b. Grid

- GridPosition

This is just for recording the stage in a Grid a node is in.

- AltRestrictions

This is designed to record the kind of altitude restrictions that can be associated with a link or node in order to generate reasonable links or nodes in a spatial grid. There are three kinds of these restrictions used. One kind is a simple explicit list of altitudes. Another kind contains a starting and ending altitude and the altitude increment to use to generate altitudes in between. The final kind produces relative altitudes -- it produces a fixed range of altitudes about a given central altitude, with the additional restriction that all of the produced altitudes must stay between a stated minimum and maximum.

- NodeData

Objects in this class contain all of the information needed to be stored at nodes in a grid in order to do a flooding search through the grid. Member

data consists of Position, geometric altitude, pressure altitude, GridPosition, WeatherColumn, approx. time arriving at the node, approx. plane mass, altitude restrictions, PlaneState, best grid link found by the flooding search, reference vertical node, and linked list of vertical nodes.

• LinkData

Objects in this class contain all of the information needed to be stored at links in a grid in order to do a flooding search through the grid. Member data consists of altitude restrictions, geometric length of link, and the list of maneuvers used to fly the link.

• FPNode

This is essentially just another name for the class Vertex<NodeData,LinkData>.

• FPLink

This is essentially just another name for the class Edge<NodeData,LinkData>.

• FPGrid

This class inherits from DirectVEGraph<NodeData,LinkData>. Member data in this class records the start and end nodes as well as the sets that form each of the stages of the grid.

• FPFlightRoute

This is an FPGrid that is assumed to be just a single linked list of nodes. Each node has at most one link coming into it and at most one link going away from it. Usually the stage information is not set.

• GridWidthRule

This class is used to describe the width rules used to generate horizontal grids. Member data includes the grid width at the endpoints and in the middle of the grid, how flat to curve the width from the middle to the endpoints, and whether to use a fixed separation distance or node count when filling out the nodes in each stage.

• GridHeightRule

This class is used to describe the altitude rules used to generate the vertical nodes in a spatial grid. The member data describes a maximum and minimum altitude, as well as the altitude increment for filling out the nodes in between. Additionally, there is a connection limit which sets the maximum altitude change allowed when creating a link between two nodes.

• GridConnectionRule

This class is used for describing which links may be created between stages in a horizontal grid. The member data describes the maximum deviation angle allowed for making a connection, and whether that angle is measured

relative to the direction of the destination node or relative to the flight route used to generate the horizontal grid.

- Miscellaneous functions include:

  Subdivide_Ground_Flight_Route -- This takes a given flight route and subdivides it as needed to meet the given requirement concerning the maximum allowed length of a segment. This also includes special rules for the takeoff and landing portions of the flight so that not too many low altitude nodes will be generated near the ends of the grid.

  Make_Horizontal_Grid_From_Flight_Route -- Given a FPFlightRoute object this creates a horizontal grid centered about the flight route. This uses the given grid rules to decide on stage widths and the spacing of nodes along each stage. The static function Add_Stage_Nodes is used for creating the nodes for each stage. This also uses grid rules for determining which horizontal nodes should be connected with links. If restricted airspaces are present, then nodes and links intersecting those airspaces will not be included in the graph. An altitude restriction is attached to each node and link for when this horizontal grid is used to generate a spatial grid. This function also records the altitudes used for each stage so that this vertical information can be given to the user in the Vert_<n>.dat files.

  Make_Spatial_Grid_From_Horizontal_Grid -- This makes a spatial grid from a given horizontal grid and associated height rules. Nodes and links in the spatial grid are generated according to the altitude restrictions and height rules above each node and link in the horizontal grid.

  Make_Matlab_Grid_File -- This produces the data files Node_<n>.dat and Link_<n>.dat that describe the nodes and links in the horizontal grids for each iteration.

4. *Navigation aids* -- The source code for this is contained in the files: NavData.cpp and NavData.h.

- Location

The member data for this class describes the Position, name, and type (airport, navigation aid, etc.).

- NavAidData

This class provides an interface to a database of Locations. It has two simple methods for adding a location and retrieving a location by name.

5. *Plane* - The source code for this is contained in the files: Plane.cpp, Plane.h, PlanePerfTest.cpp, and fp_grid.cpp.

- Maneuver

This class contains all of the information needed for a plane to perform a maneuver and to record the results. The data members are: the

climb/descent type (e.g. takeoff, steady, step), the velocity type - e.g. constant TAS (true airspeed), constant mach, - average temperature, average mass, rate of climb/descent, TAS, mach, wind speed along flight path, wind speed perpendicular to flight path, starting position, ending position, starting altitude, ending altitude, fuel flow rate for the maneuver, total cost of the maneuver, and the percentage of maximum thrust used.

- ManeuverCons

This class is used to create a linked list of all maneuver objects destroyed. Maneuvers are created and destroyed in such large numbers that it has its own special resource handling.

- PlaneState

Objects in this class record the state of a plane at a given moment in time. The data members are Position, pressure altitude, TAS (true airspeed), mach, ground speed, rate of climb/descent, total fuel used, fuel available, total mass of the plane, total time taken, and the total cost of the flight so far.

- ACPerformance

This class records all of the parameters associated with a given plane type that govern its flight performance. All of the parameters needed for the BADA performance model are recorded here.

- CostFunction

This encapsulates the cost function associated with a given plane and flight plan. The two data members are the cost of fuel (in dollars per kilogram) and the cost of time (in dollars per hour). The member function ComputeCost will compute the cost for a given amount of time and fuel.

- PlaneType

The data members for this class are the name (e.g. B737), a simple string description of the type (e.g. Boeing 737), and a copy of the ACPerformance data.

- PlaneTypeList

This class provides a list of PlaneTypes together with simple access methods by name or by index in the list.

- ClimbLimitData

This class is used by the ClimbMassLimitTable class for its table entries. It records the atmospheric density and the mass.

- ClimbMassLimitTable

For each flight plan a minimum rate of climb is specified. For that minimum rate of climb and the given airplane, a table is constructed which has entries for the altitudes of interest in this plane's range. For each altitude an object of type ClimbLimitData is recorded. This object records the air density of this

altitude and the maximum mass which this plane can lift at this altitude at the required rate of climb.

- PlaneData

This brings together all of the information associated with a given plane in a flight plan. Its data members are the ACPerformance, PlaneState, CostFunction, PlaneType, and ClimbMassLimitTable. This class also has a number of member functions which calculate the plane's performance when performing maneuvers:

DetermineSegmentManeuvers
PerformSteadyAscent
PerformSteadyDescent
PerformStepAscent
PerformStepDescent
OptimizeStepAscent
OptimizeStepDescent

Depending on the climb/descent type specified, DetermineSegmentManeuvers calls one of the Perform<steady/step><Ascent/Descent> methods to perform that kind of maneuver. These methods take a given flight segment between two geometric positions and break it up into maneuvers of their type. Methods performing step ascents or descents break the segment into a variable length level portion and a remaining non-level steady portion -- an optimization method then searches for the optimal length of the level portion. Steady climbs and descents are broken up into pieces that have no more than 2,000 feet in altitude change. Ultimately all segments of any climb/descent type are broken down into portions of steady maneuvers, and these maneuvers are performed by the PerformSingleManeuver method.

PerformSingleManeuver
OptimizeCostFunctionOverTAS
CalculateFuelFlow
TASFromThrust
GetMaxThrust
GetMinDescentThrust

PerformSingleManeuver first calculates weather information including pressure, density, wind vectors, and temperature. PerformSingleManeuver then uses OptimizeCostFunctionOverTAS in order to find the optimal TAS for the given maneuver. Both PerformSingleManeuver and OptimizeCostFunctionOverTAS use CalculateFuelFlow to determine the fuel flow rate for a given maneuver done at a given airspeed. CalculateFuelFlow is based on the BADA model for doing its performance calculations.

- Miscellaneous functions i ude:

Check_Fuel_Flow
Check_Fuel_Flow_For_Plane

These are test functions for determining if the fuel flow calculations for a given plane match the results publicized in the BADA material.

6. *Weather* -- The source code for this is contained in the files: StandardDay.cpp, StandardDay.h, Weather.cpp, Weather.h, WeatherService.cpp, WeatherService.h, and WeatherUnits.h.

6a. Weather Units -- These classes are structured similarly to the Units library classes discussed in the library classes section. These two classes are for handling the atmospheric rate of change of temperature versus altitude. Associated specific unit classes associated with these are FtPerRankine, RankinePerFt, MPerKelvin, and KelvinPerM.

- DistPerTemp
- TempPerDist

6b. Physical Laws and StandardDay -- These classes are designed to handle scientific manipulations involving atmospheric calculations.

- StandardDay

This is based on the ICAO standard day model. The class has a single data member which is the geopotential altitude. It has methods for accessing the temperature, pressure, density, speed of sound, and geometric altitude for a given geopotential altitude. Additionally, the geopotential altitude may be set in terms of pressure or geometric altitude.

- Miscellaneous Functions

Compute_Speed_Of_Sound

This computes the speed of sound in air given the air temperature.

Gas_Law_For_Dry_Air

This function calculates the relationship that holds in dry air between pressure, temperature, and density. If the function is given two of these quantities for arguments it will return the value of the remaining quantity.

GeopotentialToGeometric
GeometricToGeopotential

These functions convert between geopotential and geometric altitude. Geopotential altitude is the standard altitude used in standard day calculations. Geopotential altitude is constructed so that gravitational acceleration remains constant with altitude.

Check_Standard_Day

This is a test function that produces a table of data that can be used to verify that the standard day model is producing nominal results.

6c. Weather Data and Service

- WeatherData

This class is used for recording the aviation weather information at a given point. It has data members for the wind velocity vector (3 dimensional), temperature, and geometric altitude.

- WeatherColumn

This class represents a vertical column of aviation weather data above a given point on the earth. Its data member is a vector of WeatherData objects that represent the WeatherData information at pressure altitudes from 0 to 48,000 feet in 2,000 foot increments.

- RawWeatherData

This class is used for recording the NOAA raw aviation weather data at a given point. It has data member data for the geopotential altitude in meters, the temperature in tenths of a kelvin, the east wind component in tenths of a meter per second, and the north wind component in tenths of a meter per second.

- RawWorldWeatherData

This class holds an entire NOAA world aviation weather forecast for a given moment in time. Its data is represented in a 360 by 181 grid which is, not surprisingly, on a 1 degree longitude by 1 degree latitude world grid. For a given grid point on the earth the RawWeatherData is given at standard pressure altitudes. The class has data members for the time of the forecast, a vector of the millibar levels used for the weather data, and a single weather vector for the world's weather data. The single vector representation is used in order to make it as efficient as possible to read in the large weather data directly for its data files without any manipulations.

- WeatherService

This is the main organizing class of the classes related to world weather data. It holds the world weather data for the various times available. It has a method for calculating, using linear interpolation in space and time, a WeatherColumn for any given world position and time within the forecast range.

7. Library Classes

7a. Units - Each of these classes have associated specific classes which are generally too numerous to list. For example, associated with the general Force class are specific classes for Newtons, Dynes, Poundals, OunceForce, and PoundForce. The philosophy of these classes is to do all possible work in terms of the general class so that all knowledge of the specific units involved can be ignored. When data is initially read in, or when it is finally written out, it is of course in terms of some specific units class such as Newtons. However, no calculations can be performed with the specific classes, so they are always immediately converted to the more general units class, in this case Force. All of these classes store their value as a single double, so no extra storage is required. In addition to these classes there are numerous

overloadings of "*" and "/" to ease the handling of the interactions between the general classes. For example, Pressure * Area will produce a Force, and Force / Acceleration will produce a Mass.

- Acceleration
- Angle
- Area
- Density
- Distance
- Energy
- Force
- Mass
- MassPerTime
- Power
- Pressure
- Temperature
- TimeUnits
- VA
- Velocity
- Volume
- VV
- Weight

7b. Geometry -- The position classes are modeled in style after the units classes, but they are of course somewhat more involved. The Position class is the general class and represents the location of a point relative to the earth. PositionLLA is a specific class that represents the location of a point as a geodetic longitude, latitude, and altitude. PositionXYZ is another specific class that represents the location of a point in terms of a rectangular, cartesian coordinate system.

- PositionLLA
- PositionXYZ
- Position
- Arc

The Arc class represents an arc as having a rotation center (rotation around this vector can generate the arc), a starting position, an ending position, and a rotation angle (the angle of rotation needed around the rotation center vector to sweep out the entire arc).

7c. LinearAlgebra -- These classes are light weight classes for doing basic linear algebra manipulations. They are light weight in the sense that they require no more storage space and no more execution time than equivalent hand-coded C code would. The classes include overloading of basic mathematical operations (e.g. "*") to make the object manipulations as natural as possible.

- Vector
- SquareMatrix

7d. Time -- These classes are for doing time keeping. UTCTime is for handling UTC time information. Time is maintained with nanosecond accuracy and requires only 64 bits of storage. The period of time which can be handled is +/- 2^31 seconds (about 68 years) from the given epoch. The epoch may be set to any appropriate time. The TimePeriod class is designed to be consistent with the UTCTime class. For example, the difference between two UTCTime objects is a TimePeriod, and a UTCTime incremented by a TimePeriod is the expected UTCTime.

- TimePeriod
- UTCTime

Appendix B -- Preliminary Objects and Data Members

Address
- datatype_tbd orginization_name
- datatype_tbd contact_name
- datatype_tbd telephone_number
- datatype_tbd alternate_telephone_number
- datatype_tbd fax_number
- datatype_tbd email_address
- datatype_tbd street
- datatype_tbd city
- datatype_tbd state
- datatype_tbd zip_code
- datatype_tbd country

Geographic_Location
- Latitude_Longitude latitude_longitude
- datatype_tbd elevation
- datatype_tbd magnetic_variance_east_or_west

Time
- datatype_tbd hours
- datatype_tbd minutes
- datatype_tbd seconds

Time_Zone
- Time standard_time_offset_from_UTC
- Time daylight_time_start_date
- Time daylight_time_end_date

Fuel_Specification
- datatype_tbd fuel_cost_rate
- datatype_tbd fuel_density

Closure_Schedule
- datatype_tbd start_date
- datatype_tbd end_date
- datatype_tbd start_time
- datatype_tbd end_time

Runway_System

```
    datatype_tbd    number_of_runways
    Runway runway_data <array/list>
Runway
    datatype_tbd    identifier
    datatype_tbd    length
    datatype_tbd    landing_aid_data <array/list>
    datatype_tbd    aircraft_type_restrictions_data <array/list>
    datatype_tbd    aircraft _weigt_restriction
    Closure_Schedule closure
Airport
    datatype_tbd    name
    datatype_tbd    airport_identifier
    datatype_tbd    icao_airport_identifier
    datatype_tbd    fir_uir_identifier
    datatype_tbd    atc_en_route_center_identifier
    datatype_tbd    atc_terminal_center_identifier
    datatype_tbd    atc_tower_identifier
    datatype_tbd    city
    datatype_tbd    state
    datatype_tbd    country
    Geographic_Location  geographic_location
    Runway_System        runway_system
    Time_Zone            time_zone
    Fuel_Specification   fuel_on_site_specification
    Closure_Schedule     curfew
    Closure_Schedule     closure
    Closure_Schedule     atc_tower_closure
    Address              airport_manager
Leg_Identifier
    Airport         origin
    Airport         destination
    Time            departure_time
    Time            arrival_time
Aircraft_Identifier
    datatype_tbd    aircraft_registration_number
    datatype_tbd    aircraft_type
```

```
Flight_Schedule
    datatype_tbd            flight_number
    datatype_tbd            date
    Leg_Identifier          scheduled_leg_id <pointers/references>
    Aircraft_Identifier     scheduled_aircraft_id <pointers/references>
Flight_Trip_Schedule        //flight sequence for one flight_number
    datatype_tbd            flight_number
    datatype_tbd            number_of_flights
    Flight_Schedule         flight_schedule_sequence <pointers/references>
Aircraft_Trip_Schedule      // flight sequence for one aircraft_registration_number
    datatype_tbd            aircraft_registration_number
    datatype_tbd            number_of_filghts
    Flight_Schedule         flight_schedule_sequence <pointers/references>
Daily_Fleet_Flight_Trip_Schedule
    datatype_tbd            day_date
    datatype_tbd            number_of_flight_trips
    Flight_Trip_Schedule    flight_trip_schedule <pointers/references>
Daily_Fleet_Aircraft_Trip_Schedule
    datatype_tbd            day_date
    datatype_tbd            number_of_aircraft_trips
    Aircraft_Trip_Schedule  aircraft_trip_schedule <pointers/references>
Monthly_Fleet_Flight_Trip_Schedule
    datatype_tbd                    month_date
    datatype_tbd                    number_of_days
    Daily_Fleet_Flight_Trip_Schedule daily_fleet_flight_trip_schedule<pntrs/refs>
Monthly_Fleet_Aircraft_Trip_Schedule
    datatype_tbd                    month_date
    datatype_tbd                    number_of_days
    Daily_Fleet_Aircraft_Trip_Schedule  daily_fleet_aircraft_trip_schedule<p/r>
Flight_Identifier
    datatype_tbd    flight_number
    datatype_tbd    date
    datatype_tbd    origin
    datatype_tbd    departure_time
```

```
Speed
    datatype_tbd    mach
    datatype_tbd    indicated
    datatype_tbd    true
    datatype_tbd    ground
Wind
    datatype_tbd    speed
    datatype_tbd    direction
Weather
    Wind            wind
    datatype_tbd    temperature
    datatype_tbd    sky_cover
    datatype_tbd    visibility
    datatype_tbd    percipitation
    datatype_tbd    turbulance
    datatype_tbd    icing
    datatype_tbd    tropopause_height
Latitude_Longitude
    datatype_tbd        latitude
    datatype_tbd        longitude
Place_Bearing_Distance
    datatype_tbd        fix_name
    Latitude_Longitude  latitude_longitude
    datatype_tbd        bearing
    datatype_tbd        distance
Cursor_Coordinates
    datatype_tbd        x
    datatype_tbd        y
Position_Type               //enum
    Fix_Name_Type           = 0
    Nav_Aid_Type            = 1
    Airport_Type            = 2
    Latitude_Longitude_Type = 3
    Place_Bearing_Distance_Type = 4
    Cursor_Coordinates      = 5
```

```
Position_Data                  //union
    datatype_tbd               fix_name
    datatype_tbd               nav_aid
    datatype_tbd               airport
    Latitude_Longitude         latitude_longitude
    Place_Bearing_Distance     place_bearing_distance
    Cursor_Coordinates         cursor_coordinates

Position
    Position_Type      position_type
    Position_Data      position_data

Position_Type      position_type Crossing_Report
    Position           reporting_point
    Time               time_at_reporting_point
    datatype_tbd       altitude_at_reporting_point

Fuel_Report
    Crossing_Report    crossing_report
    datatype_tbd       fuel_remaining

Position_Report
    Crossing_Report    crossing_report
    Time               expected_time_at_next_reporting_point
    Position           next_reporting_point
    Position           next_succeding_reporting_point

Position_Weather_Report
    Crossing_Report    crossing_report
    Weather            weather_report

Departure_Station_Event_Projection
    datatype_tbd       expected_time_of_gate_departure
    datatype_tbd       expected_time_of_takeoff
    datatype_tbd       expected_payload_at_gate_departure
    datatype_tbd       expected_gross_takeoff_weight
    datatype_tbd       expected_departure_gate_id
    datatype_tbd       expected_departure_runway
```

Arrival_Station_Event_Projection

datatype_tbd	expected_time_of_touchdown
datatype_tbd	expected_time_of_gate_arrival
datatype_tbd	expected_fuel_at_gate_arrival
datatype_tbd	expected_arrival_gate_id
datatype_tbd	expected_arrival_runway

Departure_Station_Event_Report

datatype_tbd	station_id	
datatype_tbd	scheduled_time_of_gate_departure	
datatype_tbd	scheduled_time_of_takeoff	
Departure_Station_Event_Projection	current_dep_station_event_projection	
datatype_tbd	time_of_gate_departure	//out
datatype_tbd	time_of_takeoff	//off
datatype_tbd	planned_fuel_at_gate_departure	
datatype_tbd	fuel_at_gate_departure	
datatype_tbd	payload_at_gate_departure	
datatype_tbd	gross_takeoff_weight	
datatype_tbd	departure_gate_id	
datatype_tbd	departure_runway	

Arrival_Station_Event_Report

datatype_tbd	station_id	
datatype_tbd	scheduled_time_of_touchdown	
datatype_tbd	scheduled_time_of_gate_arrival	
Arrival_Station_Event_Projection	current_arr_station_event_projection	
datatype_tbd	time_of_touchdown	//on
datatype_tbd	time_of_gate_arrival	//in
datatype_tbd	planned_fuel_at_gate_arrival	
datatype_tbd	fuel_at_gate_arrival	
datatype_tbd	arrival_gate_id	
datatype_tbd	arrival_runway	

Crew_Person_Identifier

datatype_tbd	name
datatype_tbd	crew_position
datatype_tbd	equpment_experience
datatype_tbd	position_experience
datatype_tbd	training_experience
datatype_tbd	operational_qualifications

```
Flight_Status
    datatype_tbd            flight_state      //eg: sched, planned,active,cancel,end
    datatype_tbd            flight_status     //eg:nomal,impaired,emergency
    datatype_tbd            last_event_reported //eg:skd,pln,etd,eto,out,off,cnl,pre

Flight_Record
    Flight_Identifier                   flight_id
    Aircraft_Identifier                 aircraft_id
    datatype_tbd                        leg_number
    datatype_tbd                        flight_plan_number
    Flight_Status                       current_flight_staus
    Departure_Station_Event_Report      departure_station_event_report <ptr/ref>
    Arrival_Station_Event_Report        arrival_station_event_report <ptr/ref>
    datatype_tbd                        arrival_alternate_airport
    Crew_Person_Identifier              flight_deck_crew_roster <ptrs/refs>
    Crew_Person_Identifier              cabin_crew_roster <pointers/references>
    Position_Report                     position_report_history <array/list>
    Fuel_Report                         fuel_report_history <array/list>
    Position_Weather_Report             position_weather_report_history<ptrs/refs>

Flight_Trip_Record      //flight record sequence for one flight_number
    datatype_tbd        flight_number
    datatype_tbd        number_of_flights
    Flight_Record       current_flight_record <ptr/ref>
    Flight_Record       flight_record_sequence <ptrs/refs>

Aircraft_Trip_Record // flt record sequence for one aircraft_registration_number
    datatype_tbd        aircraft_registration_number
    datatype_tbd        number_of_filghts
    Flight_Record       current_flight_record <ptr/ref>
    Flight_Record       flight_record_sequence <ptrs/refs>

Flight_Plan_Identifier
    Flight_Identifier   flight_id
    datatype_tbd        flight_plan_number
```

Flight_Point_Status

Position	name	
Position	position	//eg: latitude_longitude
Time	time	//eg: sced, actual, latest arrival time
datatype_tbd	altitude	
datatype_tbd	weight	//eg: actual, max takeoff or landing weight

Flight_Planning_Parameters

Flight_Plan_Identifier	flight_plan_id	
Flight_Schedule	current_flight_schedule	
Flight_Point_Status	flight_start_point_status	// eg: replan
Flight_Point_Status	flight_end_point_status	//eg: divert
datatype_tbd	arrival_alternate_airport_1	
datatype_tbd	arrival_alternate_airport_2	
datatype_tbd	takeoff_alternate_airport	
datatype_tbd	optimization_mode	//eg: min time,fuel,cost;on-time
datatype_tbd	payload	
datatype_tbd	holding_time_at_destination	
datatype_tbd	additional_fuel	
datatype_tbd	additional_enroute_burn_fuel	
datatype_tbd	additional_taxi_burn_fuel	
datatype_tbd	cruise_speed_regime	
datatype_tbd	minimum_equipmemt_list	
datatype_tbd	restrictions	

Flight_Segment

Flight_Point_Status	start_point_status
Flight_Point_Status	end_point_status
datatype_tbd	true_air_speed
datatype_tbd	ground_speed
datatype_tbd	heading
Wind	average_wind_velocity_direction
datatype_tbd	average_headwind_component
datatype_tbd	forecast_tropopause_height
datatype_tbd	turbulance_index
datatype_tbd	teperature_deviation_from_standard

Flight_Plan_Data	
Flight_Plan_Identifier	flight_plan_id
Time	time_of_generation
datatype_tbd	date_of_generation
datatype_tbd	originator
Flight_Planning_Parameters	flight_plan_generation_parameters <ptr/ref>
Flight_Segment	flight_segment_sequence <array/list>
datatype_tbd	alternate_airport

References

1. RTCA, Incorporated, "Final Report of RTCA Task Force 3, Free Flight Implementation," RTCA, Inc., Washington , DC 20036 (October 1995)
2. RTCA, Incorporated, "Free Flight Action Plan," RTCA, Inc., Washington , DC 20036 (August 1996)
3. Beatty, R., "Dispatcher Workload and Automation," draft manuscript (September 1991) and direct communication (1996-1997)
4. Sorensen, J. A., "Design and Analysis of Advanced Flight Planning Concepts," NASA CR 4063, Seagull Technology, Inc., Los Gatos, CA 95014 (March 1987).
5. Airline Dispatchers Federation, Seagull Technology, Inc., "Airline Operational Control Overview," FAA-Boeing Commercial Airplane Group Cooperative Research and Development Agreement 93-CRDA-0034 (December 1996)
6. RTCA, Incorporated, "Minimum Aviation Performance Standard (MASP) for Air Traffic Management (ATM) - Aeronautical Operational Control (AOC) Ground-Ground Information Exchange," RTCA Paper No. 440-96/SC169-273, RTCA Special Committee (SC) 169, RTCA, Inc., Washington , DC 20036 (November 1996)
7. Dorsky, S., "AOC Simulator User's Guide," FAA Contract DTFA01-93-Y-01016, Seagull Technology, Inc., Los Gatos, CA 95014 (January 1995).
8. Couluris, G. J., "Oceanic System Improvement Study (OASIS), Volume I," Final Report No. FAA-EM-81-17, FAA Contract DOT-FA79WA-4265, SRI International, Menlo Park, CA 94025 (September 1981)
9. Aho, A., Hopcroft, J. and Ullman J., "The Design and Analysis of Computer Algorithms," pp. 207-209, Addison-Wesley Publishing Company, ISBN 0-201-00029-6 (1974)
10. EUROCONTROL Agency, "Aircraft Performance Summary Tables for the Base of Aircraft Data (BADA)," EUROCONTROL Experimental Centre, Revision 2.4, EEC Note No. 7/96 (February 1996)
11. EUROCONTROL Agency, "User Manual for the Base of Aircraft Data (BADA)," Revision 2.4, EUROCONTROL Experimental Centre, EEC Note No. 5/96 (February 1996)
12. "Air Transport World," Penton Publishing, Washington, DC 20036 (published monthly)