AIAA 2002–4599

Icing Encounter Flight Simulator with an Integrated Smart Icing System

Robert W. Deters, Glen A. Dimock, and Michael S. Selig
Department of Aeronautical and Astronautical Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

AIAA Modeling and Simulation Technologies Conference and Exhibit
August 5–8, 2002/Monterey, CA

Icing Encounter Flight Simulator with an Integrated Smart Icing System

Robert W. Deters,* Glen A. Dimock,† and Michael S. Selig‡
Department of Aeronautical and Astronautical Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

The Icing Encounter Flight Simulator is one part of the Smart Icing System project at the University of Illinois at Urbana-Champaign. The goal of the Smart Icing System project is to develop technology necessary to improve the safety of aircraft flying in icing conditions. The icing simulator is used as a platform to integrate different components of the Smart Icing System and to test the effectiveness of the components. To create an Icing Encounter Flight Simulator, functionality and Smart Icing System components were added to the FlightGear flight simulator, an open source flight simulator available on the internet. A reconfigurable aircraft model, an autopilot, and an icing model were some of the functionality added to FlightGear. Smart Icing System components integrated into the simulator include the neural-network-based icing characterization, envelope protection system, ice protection system, and an Ice Management System enhanced glass cockpit. To ensure a real-time simulation, computationally extensive processes have been distributed over several computers linked together by a local network. A tailplane stall scenario and a roll upset scenario have been designed to demonstrate the effectiveness of the Smart Icing System components on a nonlinear aerodynamics model of a DHC-6 Twin Otter aircraft in clean and iced conditions.

Introduction

The formation of ice on an aircraft in flight greatly affects its performance, stability, and control. This has led to several incidents and fatal accidents specifically tailplane stalls and roll upsets. Many times pilots are unaware of the ice accretion, and this adds a degree of danger. The Smart Icing System (SIS) project,[1,2] funded principally by NASA Glenn, was started to look into the dangers of ice accretion and how to counteract them. The goal of the project is to devise a system that will sense the presence of ice and its effects, notify the pilot, and if necessary take measures to secure the safety of the aircraft. One part of this project is the Icing Encounter Flight Simulator (IEFS). Its purpose is to bring together the different aspects of the project, such as a flight dynamics model, an autopilot,[3] an aircraft icing model,[4] an ice characterization routine,[5] an envelope protection system,[6] and human factors,[7] and perform virtual flight tests to demonstrate the effectiveness of the Smart Icing System.

Several different flight simulators were considered for the icing simulator, but the FlightGear flight simulator (FGFS)[8] was chosen for its open source modular code. The icing simulator was created by adding functionality to the FlightGear code and integrating the various SIS components. A reconfigurable aircraft model was added along with an icing model and autopilot. Other aspects of the SIS such as the ice weather model, the neural-network-based icing characterization, envelope protection system, ice protection system, and glass cockpit were integrated into the IEFS. Since several of the smart icing components are computationally extensive, parallel simulation was used to distribute the workload over several networked computers thereby ensuring real-time simulation. Testing of the IEFS is currently in progress by using two icing scenarios: tailplane stall and roll upset. These scenarios will demonstrate the effectiveness of the Smart Icing System.

FlightGear Flight Simulator Background

FlightGear flight simulator is an open-source multiplatform flight simulator. Since 1997 it has been led by Curtis Olson of the University of Minnesota.[9] It was started to provide a simulator in which anyone could develop and modify to any specifications. The FlightGear code is written in C/C++, and it employs OpenGL for its graphics. OpenGL is platform independent and is used to produce graphics using 2D and 3D objects and color bitmaps and images. Platforms currently compatible with FlightGear are Linux, Windows (NT and 95 families), BSD UNIX, SGI IRIX, Sun-OS, and Macintosh. Flight Gear is open source

*Graduate Research Assistant, Dept. of Aero. and Astro. Engineering, Student Member AIAA. rdeters@uiuc.edu
†Graduate Research Assistant, Dept. of Aero. and Astro. Engineering, Student Member AIAA. dimock@uiuc.edu
‡Associate Professor, Dept. of Aero. and Astro. Engineering, Senior Member AIAA. m-selig@uiuc.edu.
http://www.uiuc.edu/~m-selig

and follows the GNU General Public License (GPL), which allows anyone to change and redistribute the code.

One flight dynamics model (FDM) available in FlightGear is the NASA Langley Research Center flight simulator (LaRCsim).[10] The full six degrees of freedom (DOF) nonlinear equations of motion are modeled in LaRCsim, and it uses quaternions for coordinate transfer. Included with FGFS/LaRCsim are three hard-coded aircraft models: Navion, Cessna 172, and Piper Cherokee. Altitude hold and heading hold autopilots were added to FlightGear, but they are designed for the Cessna 172. Additional flight dynamics models, such as JSBsim and YAsim, have been added to FlightGear by developers. Currently FlightGear offers a wide variety of aircraft models ranging from general aircraft to military fighters.

In 1999, the University of Illinois at Urbana-Champaign (UIUC) Smart Icing Systems Research Group decided to adapt FlightGear for its own purposes. The SIS group developed a reconfigurable aircraft model to work with the LaRCsim FDM. Along with the reconfigurable aircraft model, an icing model was incorporated, and the ability to fly icing scenarios was added.

Code Organization

The FlightGear code was designed to be modular, meaning that different parts of the code are separate from each other. Being modular allows the different modules to be compiled independently and then linked together to form an executable. Figure 1 shows the basic layout of the FlightGear code. As shown, the main part of the program calls to the flight dynamics model, which in this case is LaRCsim, and LaRCsim calls to the UIUC model. The UIUC model calculates the forces and moments acting on the aircraft at an instant in time. LaRCsim takes the forces and moments and then calculates the new state of the aircraft.

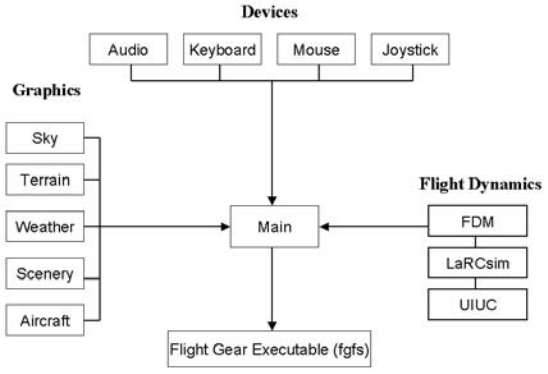

Fig. 1 FlightGear flight simulator code organization.

A basic diagram of the UIUC model is shown in Fig. 2. At time zero, aircraft specifications, such as geometry, mass, and stability and control derivatives, are read from an input file using `uiuc_menu()`. Aerodynamic forces and moments are calculated using `uiuc_force_moment()`. This function calls `uiuc_aerodeflections()` to determine the control surface deflections, and `uiuc_coefficients()` to calculate the aerodynamic forces and moments coefficients. If there is icing, `uiuc_ice()` is called to calculate the iced aerodynamic coefficients. Engine forces and moments are calculated using `uiuc_engine()`, and forces and moments from the landing gear are calculated using `uiuc_gear()`. If flight data is to be recorded, `uiuc_recorder()` is used.

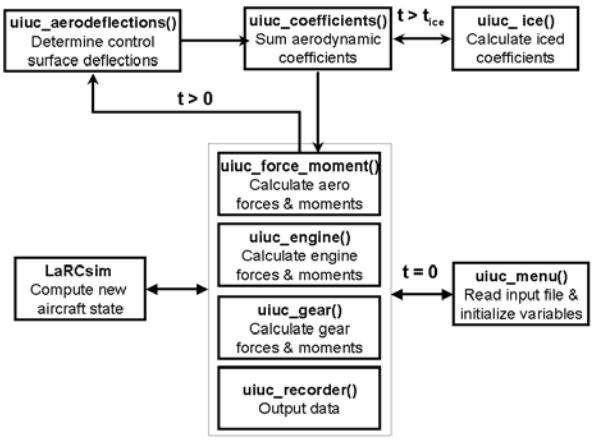

Fig. 2 UIUC code architecture.

Icing Encounter Flight Simulator Development

In creating an Icing Encounter Flight Simulator, new functionality had to added to FlightGear. The first step was the creation of a reconfigurable aircraft model that would use the flight dynamics model of LaRCsim. This reconfigurable aircraft model uses a keyword-based input file to describe the properties of the aircraft, such as geometry, mass, aerodynamic model, engine model, gear model, and initial conditions. Also included is the ability to set flags on what variables are to be saved to an output file for post-processing. There are over 280 keywords to define the aircraft properties and over 350 recordable variables. A sample of the available keywords are shown in Fig. 3. With the aircraft properties, the forces and moments acting on the aircraft are calculated and are sent to LaRCsim, as mentioned earlier, to determine the next aircraft state. The reconfigurable aircraft model allows one to model many different aircraft without having to hard-code aircraft properties.

Many additions to FlightGear are generic and can be used with any aircraft model; however, some additions have been written specifically for the aircraft used in

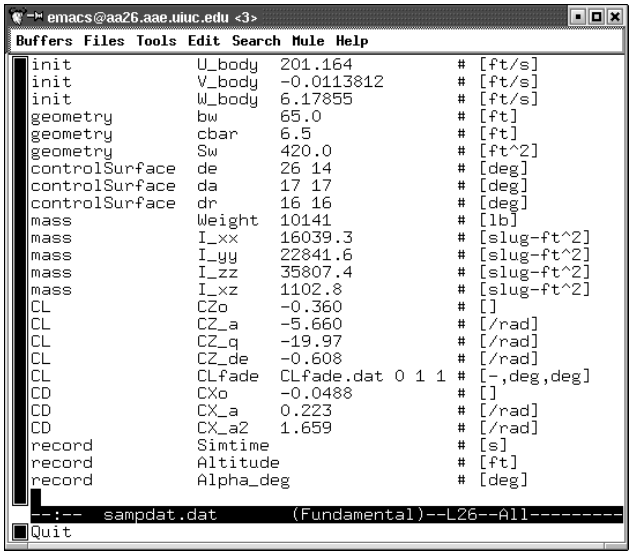

Fig. 3 Sample portion of a UIUC/FGFS keyword-based input file.

testing the IEFS. Since NASA Glenn has historically used the twin turboprop DHC-6 Twin Otter in its icing research, and because of the availability of icing data on the Twin Otter, this aircraft was chosen for the IEFS test aircraft.

Aerodynamics Model

Aerodynamic data can be specified using the reconfigurable aircraft model by two methods. Data can be provided in the form of linear stability and control derivatives or in lookup tables. Specifying aerodynamic data can be done by exclusively using linear stability and control derivatives, exclusively using lookup tables, or using a combination of the two. The advantage of the lookup tables is that nonlinear data can be provided. While using linear stability and control derivatives or lookup tables, aerodynamic data can be supplied in the wind-axis system or the body-axis system.

Two Twin Otter aerodynamic models are used with the IEFS. The first model uses linear stability and control derivatives based on the body-axis system calculated from published NASA Twin Otter flight results. The second model uses a series of three-dimensional lookup tables to model the nonlinear aerodynamics of the Twin Otter based on wind tunnel tests. There are six sets of lookup tables corresponding to the three aerodynamic force coefficients acting along the body-axis system (C_x, C_y, C_z) and the three aerodynamic moment coefficients (C_l, C_m, C_n). The first dimension of each lookup table is the angle of attack while the second is one of the following: sideslip angle, control surface deflection, roll rate, yaw rate, or pitch rate. The third dimension is the flap angle.

Autopilot

Pilots often find themselves flying in icing conditions with the autopilot on, and this has been a factor in several accidents. As stated before, the FlightGear autopilot was designed for the Cessna 172, so a new autopilot was designed for the Twin Otter. The new autopilot consists of a pitch attitude hold, an altitude hold, a roll attitude hold, and a heading hold.[3] With the new autopilot, flight scenarios dealing with icing when the autopilot is activated can be simulated.

Simulator Batch Mode

In order to effectively use the simulator as a testing platform, functionality had to be added so the simulator could run in a batch mode. The purpose of the batch mode is to be able to start the simulator at a given initial condition and to be able to control the aircraft through prescribed control inputs. The initial condition of the aircraft is supplied through the keyword-based input file by supplying the orientation and velocity. The orientation is given by the three Euler angles while the velocity is given by the three body-axis velocities and the three angular rates. The position of the aircraft can be specified by the altitude, longitude, and latitude provided as command line options built into FlightGear. Time histories of the control surface and throttle inputs are provided for the simulator as data files. During a batch mode simulation, the simulator uses these time histories instead of the inputs given by the pilot through the joystick, keyboard, or mouse. Linear interpolation is used with the time histories so that a deflection is not needed at every time step. Using the initial conditions and the control surface and throttle time histories, the simulator can run different flight scenarios without any pilot input.

SIS Additions

The main purpose of the Smart Icing System additions to the simulator is to develop and test an Ice Management System (IMS) that is designed to sense ice accretion, characterize its effect on the aircraft, protect the aircraft, and notify the pilot of the icing conditions. The first SIS addition to the simulator was an icing model. Its function is to simulate the degradation of the aircraft aerodynamics as ice accumulates. The main processes of the IMS are divided among different SIS components. Sensing ice accretion is performed by a neural-network-based icing characterization routine. Characterizing the effects of the ice accretion is also done by the neural network as well as by an envelope protection system (EPS) that can detect unsafe maneuvers. Protection of the aircraft is done by an ice protection system (IPS) along with the EPS. A glass cockpit enhanced with IMS features is used to notify the pilot of icing conditions and unsafe maneuvers.

Icing Model

Two icing models can be used in the IEFS depending on whether a linear or nonlinear aerodynamic model is being used. For a linear aerodynamic model, the icing model uses an icing severity factor η_{ice} and icing constants $k'_{C_{(A)}}$ to modify the aerodynamic coefficients. The resulting iced coefficient is calculated by

$$C_{(A)_{\text{iced}}} = (1 + \eta_{\text{ice}} k'_{C_{(A)}}) C_{(A)}$$

where $C_{(A)}$ is any aerodynamic coefficient.[4] To model different icing cases, such as wing ice, tail ice, and full aircraft icing, different icing constants are used.

For the nonlinear case, the icing model is specific to the Twin Otter nonlinear aerodynamic model. In this case, the icing model uses the same basic equation to modify the aerodynamic coefficients, but $k'_{C_{(A)}}$ is not constant. For the current nonlinear icing model, $k'_{C_{(D)}}$ and $k'_{C_{(L)}}$ are functions of the angle of attack, and $k'_{C_{(m)}}$ is a function of the angle of attack and the elevator deflection.

In either icing model, η_{ice} can be found either through data provided in the aircraft input file, or by the ice weather model. Using the input file method, ice accretion is modeled by a ramp function using the final value of η_{ice} and the amount of time it takes to reach the final value. The ice weather model simulates how ice would accumulate on the Twin Otter as it passes through a cloud. First the weather model determines the liquid water content (LWC), median volumetric diameter (MVD), and the air temperature. It then passes these values along with the angle of attack and airspeed to an ice accretion function to calculate η_{ice}.

Ice Management System

The core of the ice management system is the neural-network-based icing characterization. This process uses estimates of the flight dynamic parameters and expected values to calculate an estimate of the ice severity.[5] This ice severity estimate is then used by the other components of the IMS and is given to pilot. For details on how the neural network was integrated in the simulator, see Sehgal et al.[11] Figure 4 shows the block diagram of the neural-network-based icing characterization. The current icing characterization[5] is too computationally intensive to estimate η_{ice} in real time. Nevertheless, the neural-network-based icing characterization will still be referred to since its inclusion into the simulator is still planned in the future. For those SIS components that depend on the η_{ice} estimate, the real η_{ice} value used by the icing model is provided instead.

The envelope protection system uses the ice severity to determine whether the aircraft is in a safe flight regime. Conventional envelope protection schemes use predetermined limits on parameters such as angle of attack and bank angle, but this is not always effective in icing situations where the aircraft performance, stability, and control are continuously changing with the ice severity.[6] The EPS for the IEFS needs limits calculated in real time. Maximum and minimum pitch angles, roll angles, airspeeds, and throttle settings along with maximum angle of attacks and flap deflections are given to the pilot to ensure that the aircraft stays in a safe flight envelope. The current EPS is only version one and it calculates the maximum angle of attack by estimating the stall angle of attack. The maximum pitch angle, the minimum throttle setting, and the minimum airspeed are then calculated by using the maximum angle of attack. Later versions of the EPS will take the current state of the aircraft and estimate the aircraft state in a future time to determine if the aircraft is still in a safe flight envelope. In conjunction with the EPS, work is being done to modify the autopilot to keep the aircraft within the new icing flight envelope.

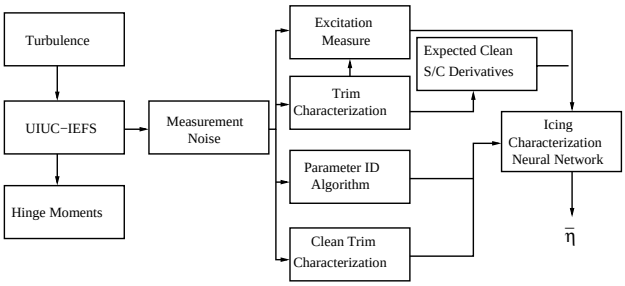

Fig. 4 Neural-network-based icing characterization architecture integrated with the IEFS. The output from the various modules is fed as input to the neural-network block where η_{ice} estimate is generated.

With icing information calculated and displayed, the pilot has the opportunity to manually turn on the aircraft ice protection system or have the smart icing system turn it on automatically. The IPS developed for the simulator is modeled after an ice boot, and when activated, the icing severity factor is reduced. Currently the simulator contains a separate ice boot system for the left and right wings and the horizontal tail. Work on additional protection systems is in progress and could possible include a stick shaker to inform the pilot when an unsafe aircraft state is being approached.

An important goal during the development of the IMS was to be able to present the information provided by the IMS to the pilot in a manner which will aid in the pilot's decision making process.[2,7] Using the research gathered by the human factors group on pilots' information requirements during icing conditions,[12] a glass cockpit was designed to inform pilots on the status of the IMS (Fig. 5). The glass cockpit was developed from baseline software written in OpenGL by Fuesz.[13] Icing information is provided to the pilot in a variety of ways. The ice severity esti-

mate is presented to the pilot as an η_{ice} trend chart and on a bird's eye view of the aircraft showing the ice location (Fig. 6). Limits provided by the EPS are displayed on the electronic attitude director indicator (EADI), throttle indicator, airspeed indicator, angle of attack indicator, and flap indicator. When the aircraft reaches or passes the indicated limits, the EPS also informs the pilot of a recommended procedure such as pitch down or increase throttle (Fig. 7). The glass cockpit also informs the pilot of the status of the ice boots (Fig. 6). When new information is being provided to the pilot, an ambient strip will appear to alert and guide the pilot's attention.

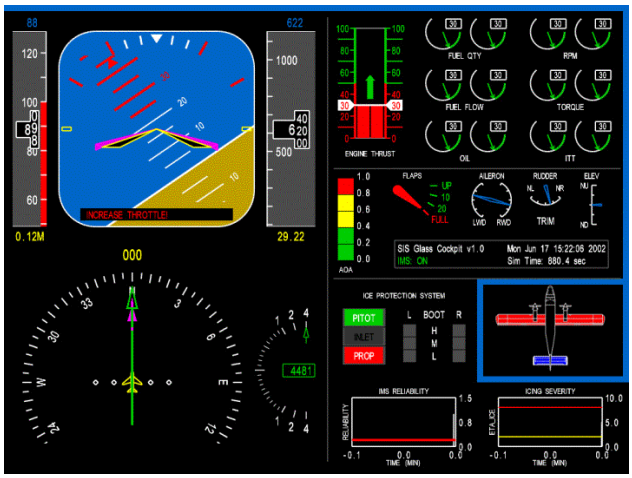

Fig. 5 Screen grab of the glass cockpit display and IMS interface.

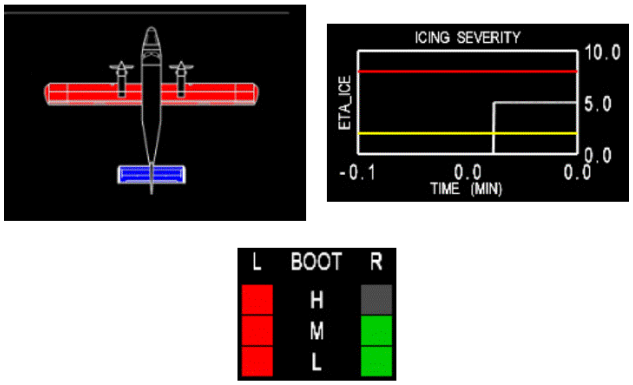

Fig. 6 Example of the η_{ice} trend chart, η_{ice} location indicator, and the ice boot indicator. The η_{ice} location indicator shows heavy icing on the wing (red) and light icing on the tail (blue). The ice boot indicator shows a failure in the left-wing boot (red) and that the right-wing boot is active (first two levels green).

Parallel Simulation

The Icing Encounter Flight Simulator contains several processor-intensive modules, which include the

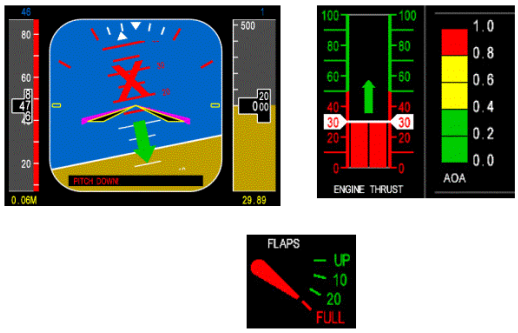

Fig. 7 Envelope protection system alerts showing a pitch down (red X, green arrow down) and increase throttle command (green arrow up).

flight dynamics model, an out-the-window display, the glass cockpit display, and the neural-network-based icing characterization. While a modern Pentium-based PC is capable of executing one or two of these modules concurrently, a complete simulation requires the use of multiple processors for real-time, parallel execution of all modules. To this end, the simulation runs across multiple PCs, each handling one or more modules. Note that the icing characterization module still cannot be run in real time on a dedicated processor, but this limitation is expected to be overcome as faster processors are released. The distributed approach to computationally demanding tasks has long been used in commercial flight simulators and myriad other systems where discrete-time events may be processed independently.

The IEFS distributed model combines a client-server architecture with some peer-to-peer features for increased performance. A central server maintains the master simulation state and communicates with each of the clients using the TCP/IP protocol in a networked environment. The clients, in turn, host the simulator modules, with each module running in a separate process. One machine may host several modules, although processor-intensive modules are run on dedicated PCs for practical real-time execution. A top-level IEFS communication protocol is derived from the sockets API, providing library functions for simple I/O in each client.[14]

During one simulation time step, each module receives a relevant subset of the current simulation state via the IEFS protocol, performs module-specific state updates, and transmits the new state subset to the server. To alleiate some issues regarding the order in which state variables are read and updated, state variables can be broadcast simultaneously to all clients in peer-to-peer fashion when necessary, instead of passing through the server for distribution. Most variables,

however, are not as time-sensitive and are only used by a few clients, making them better candidates for the client-server method.[15]

The current IEFS setup uses four Linux and Windows-based PCs to run the simulation (Fig. 8). One Linux-based machine acts as the central server that maintains the simulation state and distributes information among the clients. Two Windows-based machines run the visuals: one displays an out-the-window display via Microsoft Flight Simulator 2002, while the other machine displays the IEFS glass cockpits. The fourth PC is Linux-based and runs two processes: the UIUC version of FlightGear and the SIS additions. The UIUC version of FlightGear includes the reconfigurable aircraft model and the icing model, while the SIS additions run as a single process to include the icing weather model, the EPS, and the IPS. Note that the icing characterization module is not yet included in the simulation due to processor limitations but is shown in Figure 8 as a concept.

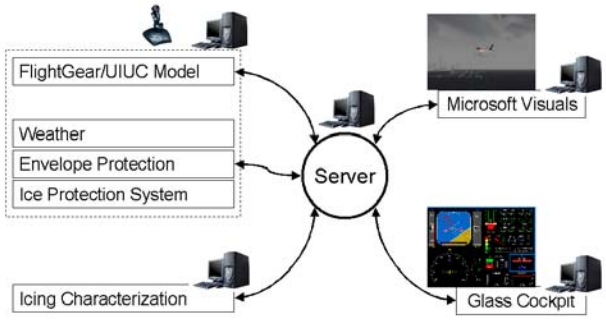

Fig. 8 IEFS client-server layout with five computers: server, outside visuals using Microsoft Flight Simulator, glass cockpit, UIUC-FlightGear with SIS components, and icing characterization (not currently used)

Icing Scenarios

Two fictional icing encounters are being developed to demonstrate the ice management system (IMS) capabilities and benefits. The first involves airframe ice leading to tailplane stall of a generic turboprop commuter aircraft. A map view of the flight is shown in Fig. 8. In this fictional account, a myriad of conceivable factors put the pilots in the following situation on approach: (a) flight in an aircraft with a history of tailplane stalls related to airframe icing (b) erratic winds at destination, leading to a runway change and an unfamiliar, over-mountain approach (c) use of 45-deg flaps during approach phase, due to steep approach (d) high gross weight and forward CG due to full passenger and cargo load, placing aircraft near edge of certified flight envelope (e) low airspeed on final approach, due to spacing requirements for departing traffic (f) a distracting element, hydraulic pump failure. Tailplane stall was the result.

In this scenario, leading factors contributing to the failure can be categorized according to the three lines of defense: avoidance, the ice protection system, or the pilots. With respect to avoidance, the descent occurred in known icing conditions with an inoperative de-ice boot. The forecast did not include icing type or severity, and the PIREPs did not indicate significant icing over the destination. As for the IPS, there was tail de-ice boot failure, the anti-ice was ineffective after significant ice accumulation, tail ice severity was not indicated by nose-mounted visual probe, and icing detection procedures were not adequately defined in flight manual. Finally, the pilots were busy with last-minute approach changes, new charts, checklists, and hydraulic pump failure; hence, ice management was not the highest priority.

The demonstration will include the above scenario as well as one with the extra layer of protection provided by an onboard IMS. With the IMS, there are two possible interventions. First, icing effects characterization would alert pilots to tail icing severity during approach, overcoming the inadequate IPS and allowing for timely corrective action. Second, the envelope protection features would prevent pilots from placing aircraft in steep descent configuration, eliminating high tail down-force and resulting tailplane stall. Both interventions would have prevented a crash.

The second scenario, currently under development by the SIS group, involves the roll upset of a generic commuter turboprop. In this scenario, the fictional aircraft is in approach configuration with large droplet icing conditions. The autopilot is on during known icing conditions, and the crew is unaware of icing severity. Ice accumulation takes place behind de-ice boots leading to three roll excursions (stall/spin sequences) and eventual recovery. With the IMS, there are two possible interventions. Icing effects characterization would alert the pilots to icing severity during holding pattern, overcoming complicating factors, thereby allowing for timely corrective action (altitude change) and full activation of the IPS. Envelope protection would allow pilots to successfully recover the aircraft before the onset of stall/spin.

Summary

The Icing Encounter Flight Simulator was developed by adding functionality to and integrating Smart Icing System components into the FlightGear flight simulator. The necessity of simulating a multitude of different aircraft properties led to the development of a keyword-base reconfigurable aircraft model. A keyword-based input file allows the user to define an aircraft model through its geometry, mass, aerody-

namic model, engine model, and gear model. Both linear and nonlinear aerodynamics can be modeled by specifying stability and control derivatives for the linear case or by supplying aerodynamic information in the form of lookup tables for the nonlinear case. For testing purposes, a linear model and a nonlinear model using three-dimensional lookup tables are used as the aerodynamic models for a Twin Otter.

For icing simulations, two icing models have been added to FlightGear to model the degradation of the Twin Otter aircraft performance during icing. Both icing models modify the aerodynamic coefficients based on ice severity. One model is designed to work with the linear Twin Otter aerodynamic case while the other is designed to work with the nonlinear aerodynamic case. To model the ice severity factor η_{ice} used by the icing model, either a user-defined ramp function or the icing weather model can be used.

The main component of the Smart Icing System is the Ice Management System which is composed of a neural-network-based icing characterization, envelope protection system, ice protection system, and IMS enhanced glass cockpit. The presence of ice is detected by the neural-network-based icing characterization and an ice severity estimate is calculated. The current icing characterization is too computationally intensive to run in real time, so the ice severity from the icing model is used instead of an estimate. Based on the ice severity, the EPS can modify the aircraft flight envelope to ensure the safety of the aircraft, and the IPS can activate the ice boots to reduce the ice severity. During the icing conditions, the pilot is constantly informed of the icing conditions and any necessary safety precautions through the glass cockpit.

To ensure real-time simulation, parallel simulation has been used to divide computationally extensive processes such as the external visuals, glass cockpit, and the flight dynamics model, among several computers on a local network. Testing the effectiveness of the SIS components is being done through two scenarios, tailplane stall and roll upset. These scenarios are a major concern while flying during icing conditions.

Acknowledgments

This research has been sponsored by NASA Glenn Research Center, and its support is gratefully acknowledged. The authors wish to acknowledge Bipin Sehgal and Jeff Scott (both former UIUC AAE graduate students) for their valuable contributions to the IEFS code. Also, the FlightGear development group is thanked for all their help, support, and suggestions.

References

[1]Bragg, M.B., Perkins, W.R., Sarter, N.B., Basar, T., Voulgaris, P.G., Gurbacki, H.M., Melody, J.W., and McCray, S.A., "An Interdisciplinary Approach to Inflight Aircraft Icing Safety," AIAA 98-0095, Reno, NV, Jan. 1998.

[2]Bragg, M.B., Perkins, W.R., Basar, B., Voulgaris, P.G., Selig, M.S., Melody, J.W., and Sarter N.B., "Smart Icing Systems for Aircraft Icing Safety," AIAA 2002-0813, Reno, NV, Jan. 2002.

[3]Sharma, V. and Voulgaris, P., "Effects of Ice Accretion on Aircraft Autopilot Stability and Performance," AIAA 2002-0815, Reno, NV, Jan. 2002.

[4]Bragg, M.B., Hutchison, T., Merret, J., Oltman, R., and Pokhariyal, D., "Effects of Ice Accretion on Aircraft Flight Dynamics," AIAA 2000-0360, Reno, NV, Jan. 2000.

[5]Melody, J.W., Pokhariyal, D., Merret, J., Basar, T., Perkins, W.R., and Bragg, M.B., "Sensor Integration for Inflight Icing Characterization using Neural Networks," AIAA 2001-0542, Reno, NV, Jan. 2001.

[6]Merret, J., Hossain, K., and Bragg, M.B., "Envelope Protection and Atmospheric Disturbances in Icing Encounters," AIAA 2002-0814, Reno, NV, Jan. 2002.

[7]Sarter, N.B. and Schroeder, B.O., "Supporting Decision-Making and Action Selection Under Time Pressure and Uncertainty: The Case of Inflight Icing," *Human Factors*, in press 2001.

[8]FlightGear website, http://www.flightgear.org, 1997–present.

[9]Olson, C.L., University of Minnesota, http://www.menet.umn.edu/~curt, 2002.

[10]Jackson, B.E., *Manual for a Workstation-Based Generic Flight Simulation Program (LaRCsim) Version 1.4*, NASA TM 110164, April 1995.

[11]Sehgal, B., Deters, R.W., Selig, M.S., "Icing Encounter Flight Simulator," AIAA 2002-0817, Reno, NV, Jan. 2002.

[12]McGuirl, J.M. and Sarter, N.B., "Inflight Icing: A Survey of regional Carrier Pilots," Unpublished CSEL Technical Report prepared for NASA Glenn Research Center, Columbus, OH, The Ohio State University, 2001.

[13]Fuesz B., Private Communications, 1996–2002.

[14]Stevens, R.W., *Unix Network Programming Vol. 1*, Prentice-Hall, Inc., 1998.

[15]Fujimoto, R.M., *Parallel and Distributed Simulation Systems*, John Wiley & Sons, Inc., 2000.