Design of A Simulation S/W for Evaluation of Auto-Landing Algorithms

윤석준* · 김강수** · 안재준**

Yoon sug-joon* · Kim kang-soo** · Ahn jae-joon**

Abstract

A Simulation S/W is developed to evaluate performances of MLS (Microwave Landing System) and IBLS (Integrated Beacon Landing System) in precision auto-landing. For this study classical PID and optimal LQG controllers are developed as well as mathematical models of MLS and IBLS. Ship-landing condition is also considered by assuming sinusoidal movement of the ship in the pitch direction. The simulated aircraft is F-16 in the study of precision auto-landing. For the integrated simulation environment GUI windows are designed for input of parameter values necessary for simulation, such as vehicle performance and environmental data. For validation and verification of models various comparison graphs of simulation outputs are comprised in the GUI design as well as 3D visual simulation of vehicle dynamics.

Key Words : Aircraft, Simulation, Software, Auto-landing, Algorithms, Test Evaluation

1. Introduction

This study is to compare performances and stabilities of MLS (Microwave Landing System)[1]-[3] and IBLS (Integrated Beacon Landing System) [4]-[6] in precision auto-landing. For this study classical and optimal controllers are developed as well as mathematical models of MLS and IBLS. Ship-landing condition is also considered by assuming sinusoidal movement of the ship in the pitch direction. F-16 was chosen for the study of precision auto-landing, because reliable UAV data was not available at the time of this study. Mathematical models are realized in Matlab[7], Simulink and Sfunctions [8],[9] comprising C codes, which can be translated into C codes for real-time or fasttime simulation. For the integrated simulation environment GUI windows are designed for input of parameter values necessary for simulation, such as vehicle performance and environmental data. For validation and verification of models graphical expressions of various simulation results are also comprised in the GUI design as well as 3D visual simulation of vehicle dynamics. Thus the simulation S/W is composed of mathematical models, scheduler, GUI, and graph manager.

* Associate Professor, Dept. of Aerospace Engineering
** Research Assistant, Dept. of Aerospace Engineering

2. Development and Integrated Software Environment

The development efforts for the integrated simulation S/W are described in its four major components.

2.1 MATLAB/SIMULINK MODELS

Matlab and Simulink models express flight dynamics, guidance sensors, controllers, and atmospheric environments[12]. The flight dynamics of general airplanes, including F16[11], is modeled in S-function composed of C codes. Guidance sensor models of IBLS and MLS are expressed in Simulink blocks for easy access during real-time or non-real-time simulations. The Simulink models are controlled in realtime by the scheduler, using 'sim' command of Matlab. For the purpose, an input-port is added as shown in Figure 1, and the simulation begins right after initial values and a trigger signal are delivered to the Simulink models via the input-port. The simulation results are retrieved from the Simulink blocks to GUI and Graph Manger through 'To workspace' command as shown in Figure 2.

2.2 SCHEDULER

The scheduler takes a role of controlling various data and events occurring during the simulation process. The scheduler also enables the GUI component, which is written in Active-X controls of MS Windows[13], to communicate with the Simulink components by way of callback functions.

The scheduler includes 'actxcontrol' command, as in Figure 3, in order to transmit the events from GUI to the Matlab and Simulink components. Then this command returns handle values in order for Active-X controls to be operable on Matlab and Simulink. A programmer can access to Active-X controls by way of these handle values, and designates callback functions for events from Active-X controls by using 'actxcontrol' command. The callback functions are used to control Matlab and Simulink components in this S/W development.

In order to register Active-X controls in Matlab, 'Container' must be generated first by 'figure' command as in Figure 4. Then 'test.VBCtrl' is registered at Matlab workspace by using relevant handle values. During this process 'allevents' is registered together as a callback function of the event. This function is called each time its registered event occurs, and

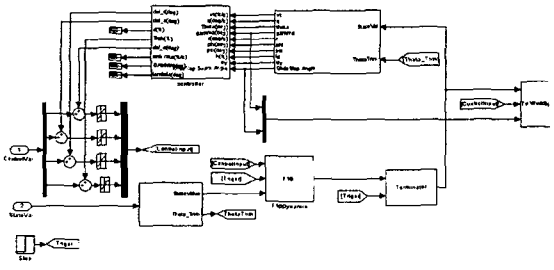

Figure 1. Control of Simulink Blocks in the UAV Simulation S/W

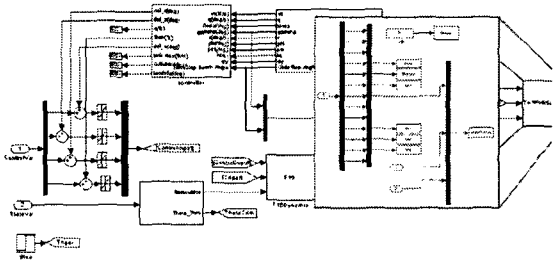

Figure 2. Simulink Blocks Showing Simulation Data Retrieval

relevant arguments are returned. The arguments consist of handle values, event names for calls, and so on. The scheduler processes relevant events using these arguments.

The scheduler becomes active each time an event occurs, and a relevant workspace is generated. Since a relevant workspace dies when the scheduler becomes inactive, an additional effort has to be considered in order to reserve necessary data. Among feasible options a method is chosen in this study, which is to store necessary data at the root area of 'figure' in the form of 'structure'. As in Figure 4, 'UserData' is retrieved from the root of 'figure' by using 'get' function, and is stored at the root area again when the scheduler becomes inactive. An advantage of this method is that any data can be processed conveniently at a sufficiently high speed.

```
p=cd;
f=figure('pos', [10 155 725 540]);
set(f, 'MenuBar', 'none');
set(f, 'Resize', 'off');
set(f, 'NumberTi tle', 'off');
set(f, 'Name', 'Flight Control Simulation');
% ",";...
h=actxcontrol('test. VBctrl', [00725540].f,...
{...
'Initail,' 'allevens';...
'Trim','allevents';...
                    ?
                    ?
```

Figure 3. An Example of Active-X Controls Used in the Scheduler

```
function allevents(varargin)
h=varargin{1{;
f=h.FigureHandle;
MM=get (0, 'UserData');
addpath([h.currentpath'/mls/main__gui
₩]);
switch varargin{4}
case 'Initial'
                    ?
                    ?
                    ?
case 'QuickSave'
cd(h.CurrentPath);
```

2.3 GUI

GUI of the UAV simulation S/W is realized in the form of an Active-X control, using Visual Basic. Visual Basic provides a convenient development environment in designing GUI, and allows quick configuration of events and attributes. GUI is composed of three major components. The first component is the GUI window for setting parameter values necessary for simulation. Through this GUI component various initial conditions for test and evaluation can be set. This module allows store and retrieval of simulation results in the form of common computer files. The second GUI component comprises option-setting functions for viewing simulation results in plots or graphs. The GUI provides buttons, which are linked to 'plot' functions of Matlab for automatic generation of various performance graphs. A user of the simulation S/W is supposed to set options in this GUI window for viewing necessary information. The third component of GUI is to visualize

flight trajectory in 3D graphics. The 3D graphic animation is realized by the graphic simulation technology used in debriefing processes of flight training devices. The 3D graphic environment is coded in C++ and Open GL, and encapsulated in an Active-X control, which is integrated in the total GUI environment using Visual Basic. This GUI component allows visual understanding of attitudes and positions of a simulated aircraft, varying in time. The speed of the image replay can be controlled by a sliding bar in the GUI component.

2.4 GRAPH MANAGER

Simulation results are provided in the form of numerical data. The graph manager makes the analysis more intuitive by transforming the data into graphical forms. The graph manager developed for the purpose takes advantage of Matlab' s plot functions. This graph manager is coded in Matlab functions, and called by the scheduler.

3. Examples and Test Evaluation

Some of the simulation results are plotted in Figure 8-10. The legends in the figures represent the auto-landing conditions in Table 1. Figures show that IBLS and MLS satisfy the requirements of CAT III precision[10] auto-

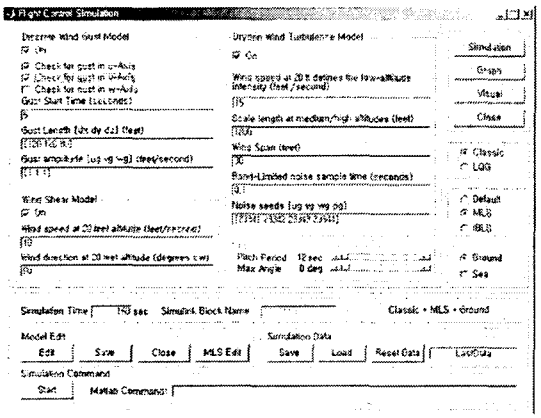

Figure 5. Simulation Environment Setting at GUI

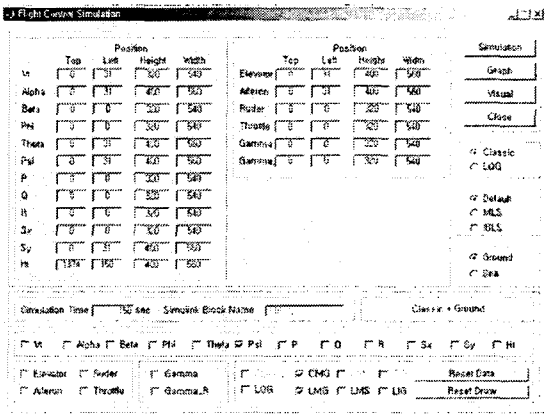

Figure 6. Flight Parameter Setting at GUI

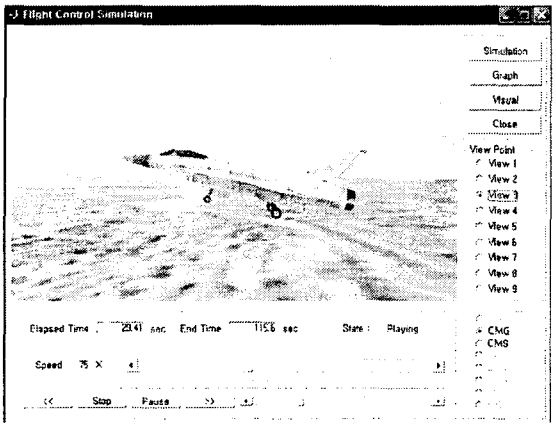

Figure 7. A Snap Shot of 3D Visual Simulation

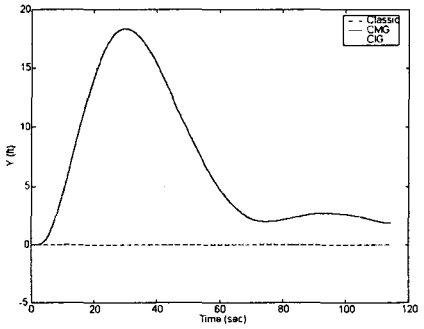

Figure 8. Comparison of Deviations in the Lateral Direction during Ground Landing with No Disturbance

landing when they are combined with either classical or optimal controllers.

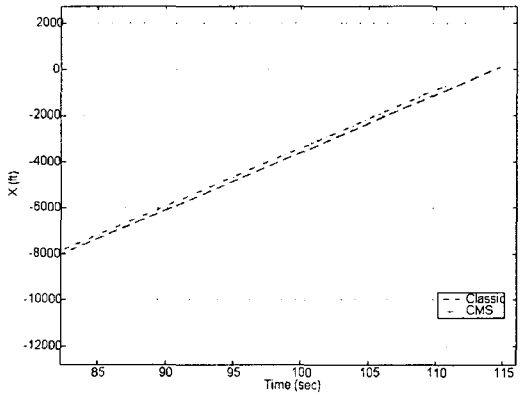

Figure 9. Comparison of Deviations in the Longitudinal Direction during Ship Landing with Gusty Wind

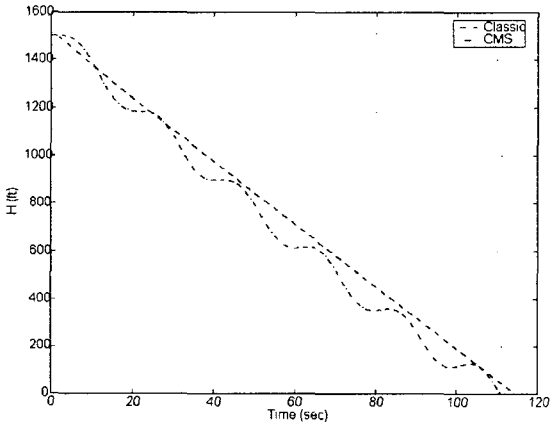

Figure 10. Comparison of Altitudes during Ship Landing with Gusty Wind

Symbols	Auto-Landing Condotions
Classic	PID Control + Ground
CMG	Classical + MLS + Ground
CIG	Classical + IBLS + Ground
CMS	Classical + MLS + Sea
LQG	LQG + Ground
LIG	LQG + IBLS + Ground
LMS	LQG + MLS + Sea

4. Conclusions

A Simulation S/W was developed for test and evaluation of precision auto-landing algorithms. The S/W is based on various computer languages such as Matlab, Simulink, C++, Visual Basic, and Open GL. In order to integrate subsystems coded in these different languages from each other, Active-X controls and callback functions were effectively used.

Reference

[1] Clark E. Cohen, Boris Pervan, Real-Time Cycle Ambiguity Resolution using a Pseudolite for Precision Landing of Aircraft With GPS, 1993. 3.30

[2] Clark E. Cohen, Boris Pervan, Real-Time Flight Test Evaluation of the GPS Marker Beacon Concept for Category III Kinematics GPS Precision Landing, ION GPS-93, 1993. 9.22

[3] Clark E. Cohen, Boris Pervan, Precision Landing Tests with Improved Integrity Beacon Pseudolites, ION GPS-95, 1995.9

[4] R. J. Kelly and E. F. C. LaBerge, "MLS: A Total System Approach," IEEE AES Mag., pp.27~40, May, 1990

[5] M. H. Carpentier, Principles of Modern Radar System, Artech House, London, UK, 1988.

[6] D. K. Barton, Modern Radar System Analysis, Artech House, London, UK, 1988.

[7] Using Matlab, Version 5, Mathworks Inc., 1999.

[8] Using Simulink, Version 3.0, Mathworks Inc., 1999.

[9] Real-time Workshop User's Manual, Version 3.0, Mathworks Inc., 1999.

[10] MIL-F-8785C, "U. S. Dept. of Defense Military Specification: Flying Qualities of Piloted Airplanes", Nov.5.1980.

[11] Brian L. Stevens, Frank L. Lewis, "Aircraft Control and Simulation", 1992, John Willy & Sons.

[12] Robert C. Nelson, "Flight Stability and Automatic Control", 1998, McGraw-hill.

[13] Microsoft MSDN Library, October, 2002.