AIAA2001-4423

Applications of a Simulation Environment During Wind Tunnel Testing

Randy S. Hultberg
David R. Gingras
Jeffery W. Bell

Bihrle Applied Research Inc.
Hampton, VA

Modeling and Simulation Technologies Conference & Exhibit
In Association with the Canadian Aeronautics and Space Institute
6 - 9 Aug 2001
Montreal, Quebec, Canada

APPLICATIONS OF A SIMULATION ENVIRONMENT DURING WIND TUNNEL TESTING

RANDY S. HULTBERG[*]
DAVID R. GINGRAS[†]
JEFFERY W. BELL[‡]
Bihrle Applied Research Inc.
18 Research Drive
Hampton, VA 23666 USA
(757)766-2416 FAX (757)766-922

Abstract

There is a distinct parallel between software requirements for hardware in the loop simulation and dynamic wind tunnel testing. Timing, hardware interface, and usability are key factors in overall system quality. A reconfigurable PC-Based simulation environment running under the Windows family of operating systems met timing requirements needed for dynamic testing and was selected to perform data acquisition, processing, and display tasks in two different efforts. As expected, the application of the software was a success. The reconfigurable structure of the software allowed for the quick design and implementation of software for each test.

This paper provides a discussion of the software chosen for use during the wind tunnel testing with primary focus on simulation loop timing, and hardware interface. In addition, the paper presents details pertaining to the specific application of the software in the two wind-tunnel testing efforts.

Introduction

A typical wind-tunnel test to collect aerodynamic forces and moments consists of a test article, force and moment measurement device, model support (rig), signal conditioners, data acquisition and recording devices, and data monitor. Additional components of wind-tunnel test apparatus can be rig controllers, model actuators, and a variety of tunnel state monitors. As once would expect, the specific objectives of test dictate the complexity of the test setup. For example, dynamic wind tunnel testing requires the test article to be subject to a rotation and/or translated motion where position data must be synchronized with the acquired force and moment data. The resulting data must then be reduced and displayed for test engineers' inspection. The apparatus and data requirements for dynamic wind tunnel testing are not unlike those of real-time hardware in the loop simulations. Like real-time hardware in the loop simulations, overall system timing and communication are of the utmost importance. Once received the data must then be processed and provided to the pilot or test engineer. In hardware in the loop situation, the pilot responds to stimuli such as graphical displays or motion, whereas the test engineer and controllers respond to data reduced and provided in a timely manner.

Given the parallel between hardware in the loop simulation and dynamic wind tunnel testing, common tools should be relevant to each application. The remainder of this paper provides details pertaining to two such applications. A discussion of components and functionality of the selected simulation environment is presented in addition to information about the specific application of the environment in each of the tests.

Symbols

e_{ExF}	Simulation execution time error.
$T_{Simulation}$	Simulation clock time.
$\Delta T_{Simulation}$	Simulation frame duration.
T_{Start}	Reference simulation start time.
T_{True}	Reference clock time.

Simulation Environment

Overview

The simulation environment chosen for the wind tunnel test efforts was the Bihrle Applied Research Inc.

[*] Chief Test Engineer
† Senior Engineer, Senior Member
‡ Chief Software Engineer

(BAR) D-Six simulation environment. D-Six is a reconfigurable PC-Based simulation environment that runs in the Windows family of operating systems. Modular reconfigurability allows it to be extended to a number of applications (References 1 and 2).

D-Six uses a component-based architecture, such that the required core components are concerned primarily with loading and executing simulation models. Additional functionality is integrated by loading individual component libraries that interact with D-Six through one or more defined interfaces. Figure 1 contains a high-level diagram of the D-Six software structure and highlight key features in the environment.

The software consists of two major classes of components, project independent and project dependent. Project independent components are reusable from project to project and include code that maintains the core simulation loop and controls timing. Project independent components handle the graphical user interface, modular hardware interface, the simulation database manager, and the simulation function library. D-Six architecture allows for project independent reconfigurability by the implementation of additional user developed modules or plug-ins. Project dependent components of the environment are, as their name indicate, based on project specific information and consist of settings files and a project dynamically link library (DLL). Contained in this DLL, is specific project base code that is developed by the user. In flight model applications, user defined aerodynamics, propulsion, and control model equations are implemented. This project-based code also contains an interface to the project independent components mentioned above in order to take advantage of the various resources provided.

As mentioned in the introduction, all hardware in the loop applications, timing, hardware interface, and user interfaces very important contributors to a successful system. In the D-Six environment these duties are project independent and reusable. A detailed discussion of these tasks is provided in the sections below.

Timing

The topic of timing and simulation can be a confusing and controversial subject. The following discussion is provided to explain the timing of the D-Six simulation loop and present evidence that it meets the timing needs of the simulation and wind-tunnel testing tasks for which it is applied. For the purpose of this discussion, a distinction is made between the terms “true time”, “real-time” and “real-time system”. “True time” refers to a reference time used to track simulation timing. The use of the term “real time” in the context of this paper refers to timing of sufficient accuracy to solve the problem at hand. For the closed loop wind tunnel testing efforts, a 10-millisecond accuracy in desired. The term “real-time system” refers to a common definition found in Reference 3, provided below.

"A real-time system is one in which the correctness of the computations not only depends upon the logical correctness of the computation but also upon the time at which the result is produced. If the timing constraints of the system are not met, system failure is said to have occurred."

Based on these definitions, with the exception of Windows CE, the systems running any of the Windows family of operating systems cannot be considered “real-time systems.” However, this does not imply that software running under Windows operating systems cannot meet “real time” for a particular task.

With this in mind, since Windows is not a real-time system, there are no guarantees that a simulation time frame can be executed within a fixed period of time. While there are no guarantees Windows will not preempt an application process and unnecessarily delay it’s time frame execution, there is nothing inherent in the operating system that would cause this to happen. Timing errors are caused solely by software installed on the system. Therefore, it is possible to determine the total timing error for each time step using the Windows performance counter and use the information to detect errors greater than maximum values for a predefined task, thus providing a degree of determinism. In cases where these maximums are not met, it is generally possible to identify and correct the source of the error.

The D-Six simulation loop uses an error correction model to maintain relative real-time operation as shown in the simplified state diagram (Figure 2).

During the begin state, the initial true time is determined, and the simulation time is set to zero prior to calling the initialization routines for the simulation model. The wait state normally performs background operations as well as effective delay prediction to reduce system load, but for simplicity these operations are left out of the above state diagram. The simulation holds in a wait state until it is time to execute simulation frame. The execution state increments the current simulation time by a single frame length, then executes a single simulation step, upon completion it returns to the wait state.

In this loop, the Windows performance counter determines timer resolution and is dependent on the processor and clock speed. For example, an Intel Pentium III-500MHz system has a performance counter frequency

(PCF) of approximately 3MHz, where an Intel Pentium II 233MHz system has a PCF of approximately 1.2MHz.

The simulation frame execution time error can be computed at the start of the execution state using equation (1).

$$e_{ExF} = T_{True} - T_{Start} - T_{Simulation} - \Delta T_{Simulation} \quad (1)$$

This value is a measure of the difference between a scheduled time for a simulation frame execution and the actual time it is executed.

A series of tests were performed to evaluate D-Six timing. A D-Six simulation was run under Windows 2000 in conjunction with a CPU stress tool included with the Windows Platform Software Development Kit (SDK). The test configuration is provided in Table I.

Table I. Timing test parameters.

Computer:	Intel Pentium II 500 MHz
Operating System:	Windows 2000 Professional
D-Six Configuration:	• Six-Degree of Freedom Flight model. Including nonlinear table look up. • 600 x 600 Pixel DirectX Graphics Window • A Single DirectX driven "real-time" strip chart. • A Single Microsoft Side Winder USB game stick.
Simulation Frame:	80Hz
Simulation Duration:	10 seconds

The accuracy of these tests is based on the assumption that the processor clock is accurate. This assumption is valid based on publicly available data (Reference 4).

Simulations were executed at four different levels of system load defined by the Window System load tool, low, medium, high, and critical. The results are provided in Figure 3. Table II contains a summary of test loads and maximum errors attained.

The test results show that under low system load, the D-Six environment is capable of reasonable frame scheduling within 3 ms of true time and meets timing requirements specified for this effort. Depending on task requirements, the environment may achieve sufficient timing under medium system loading, but would not be recommended for use under high or critical loads.

Hardware Interface

Interface with hardware and software devices is handled by a core component in D-Six named IOD (Input/Output Devices). Hardware devices are integrated with D-Six through IOD by loading a module that registers the hardware's capabilities (Figure 4). Software devices also register their capabilities with IOD, and can be hardware emulators, or can represent a user interface, for example, a graphic of a switchboard that responds to mouse clicks. IOD provides an efficient and consistent mechanism for passing information between different D-Six components in a generic, hardware independent manner.

Devices are registered with IOD by passing a description to a registration function. The description is essentially a flexible structure that declares channels of analog input and output, individual switches, and switch bit fields. IOD provides mapping between analog I/O channels and simulation variables with optional linear scaling. Channel scaling capabilities include setting a minimum and maximum, or setting an offset and multiplier. Switches are used as triggers for software events that are also registered in IOD. A software event can be any action carried out by a function call that receives a single flag that indicates if the event is being triggered on or off. Switches can be set to level or edge triggered and can be set active when zero on non-zero.

Table II. Timing test results

Load	$e_{ExecFrame}$ **[ms]**
Low: D-Six + no other apps	2.78
Medium: D-Six + 1 Normal Priority Thread	19.71
High: D-Six + 3 Normal Priority Threads	252.06
Critical: D-Six + 4 Threads Above Normal Priority	617.34

Because IOD performs all of the common data operations most hardware I/O require, writing a component that registers a device with IOD is generally trivial. Module wizards provide access to D-Six functionality in addition to MFC and DirectX components.

During execution, IOD polls each device before performing operation on the input and output data. Components with a device that requires polling use this execution time to communicate with the device. Components that do not require polling must wait until they are polled to read or write data to IOD. IOD is not thread safe and will only poll a device before the data is to be used, which is typically coincident with each simulation frame. However, users have the option of slowing this rate to compensate for slow hardware. Hardware that requires faster polling times may be handled by a separate thread that communicates with IOD only during polling calls on the main thread.

User Interface

D-Six contains a graphical user interface that allows users to configure a workspace that is best suited for the tasks at hand. The interface is a collection of tools that provide the user a number of options to access data before, during, or after simulation run and are listed in Table III.

Table III. D-Six user interface tools.

Tool	Description
DirectX Graphics	DirectX base 3-D graphics for out-the-window or external dynamic views.
Instrumentation Builder	Script controlled DirectX based instrumentation display package to configure custom displays, gauges, or dials from bitmaps for graphical display of information.
Sound Utility	DirectX based sound module that allows volume and frequency variations of .WAV files to be driven by simulation data and triggered as mulation events.
Strip Charting	Multi-window real-time plotting tool
Plot Utility	Post run data plotting utility.
Data Analysis and Editing	Reconfigurable post run data editor and analysis tool.
Project Overdrive	Environment functionality to test project algorithms by driving simulation with data that has been imported from previous simulations, flight, or previous experiments.
Script Engine	Script control of simulation tasks and data. The script engine also provides easy access to third party software for auto-report generation.

Wind-Tunnel Applications

As mentioned above, the D-Six environment was included in the design of two wind-tunnel experiments where the acquisition, processing, and feedback of data were critical. The two wind-tunnel experiments were associated with Small Business Innovative Research (SBIR) efforts for the US Air Force. The following sections provide details pertaining to each of the applications focusing on the integration of the simulation environment during the tests.

Formation Flight

As part of SBIR AF98-175 Phase II tasking, Bihrle Applied Research developed a wind tunnel testing capability to measure the effects of close formation flight on the aerodynamics of a trail vehicle in a two-ship formation (Figure 5). Because of the complex nature of aerodynamic interference from flying in formation and the effect of these interactions on control laws design, accurate physical modeling is important for the simulation for automated refueling or station keeping tasks (References 5 and 6).

As part of this research, two advanced configuration subscale wind tunnel models were tested in the Langley Full Scale Tunnel (LFST) in a number of relative formation positions. A majority of the data was collected with no control surfaces deflected and produced “clean” configuration formation effect data. Part of the objective in collecting the formation data was to assess control power requirements to keep formation as well as to determine potential performance benefits.

To further investigate the effect of formation on control power and trimmed performance, a model with actuated surfaces was placed in trail formation positions. The actuators were controlled by a real-time closed loop trimming control law during static formation points as well as dynamic vertical translations through formation positions.

Test Design

Active trimming portions of the wind-tunnel tests relied on the use of the D-Six simulation environment for data acquisition, monitoring, and processing. All data reduction, control-law implementation, control surface command, and data recording were handled by project dependent code loaded into D-Six. The function diagram in Figure 6, shows data flow for during the test. Data were acquired from the six-component strain gauge balance mounted in the trail vehicle using the IOD component of D-Six to interface with a National Instruments PCI Analog and Digital I/O board. This board was capable of 8 differential analog inputs that could be processed with a

16-bit analog to digital converter at a data acquisition rate of 20 kHz. Project code was run at a 50 Hz frame rate.

Once acquired, balance voltages were converted to aerodynamic coefficients. Pitch and roll coefficients were then used as feedback channels in a trim control law, Figure 7, which was a simple PI controller to maintain a reference pitching moment and rolling moment coefficient. Resulting control surface commands were sent to the digital output channels and fed to the surface actuators on the model.

In addition to trim capability, the controller was configured to follow prescribed deflection profiles, like sine waves and ramps built using a graphical interface. The controller was also capable of following deflection commands fed by a control stick interfaced through IOD.

User interface for the test consisted of several graphical real-time strip charts displaying control surface deflections and aerodynamics coefficients. Dynamic digital displays of a number of project parameters were also used.

Results

The use of the simulation environment and implementation of control laws were a success. Static data were collected for each point in a grid of formation positions with active trimming on and off. Dynamic tests were also performed where the actuated model translated vertically through the lead vehicle wake. Forces and moments were measured during these tests with trim active and not active. Figure 8 contains sample plots pitching moment and rolling moment coefficients from each of these cases.

Virtual Flight Test

In fulfillment of requirements for Phase I of SBIR AF00-297, Bihrle Applied Research was tasked to demonstrate concepts associated with Virtual Flight Testing (VFT)(Reference 7 and 8). VFT is a ground testing method where a full-scale aerial vehicle, such as a missile, is placed in a wind tunnel using minimal constraint and commanded to duplicate in-flight motions. The main objective of this test technique is to provide a control environment to test control strategy, hardware, and other key system components.

To investigate test methodology and model support geometry, a sub-scale wind-tunnel test was performed. The main objective was to excite vehicle motions in order to analyze test article response and predicting trajectories using data acquired during the test. The data acquisition, processing, simulation tasks were performed using the D-Six simulation environment.

Test Design

The main objective during the test design was to minimize set-up cost and maximize efficiency. Because of its accessibility and sufficient size, the Old Dominion University three- by-four-foot low speed wind tunnel was chosen for the test. A custom model support and measurement system was designed for the test that allowed the required freedom of motion in pitch while measuring translational forces. The measurement system consisted of twin strain gauge balances mounted in the main rig supports. Figure 9 contains a photo of the AIM-9 like article chosen for this test mounted on support apparatus. The model was designed and fabricated by Bihrle Applied Research and contained actuated pitch control fins, a potentiometer to measure attitude, and a piezo electric gyro to measure pitch rate.

As part of the test design, a simulation of the test article was developed and used to determine load constraints for the test and tune control laws. Using the missile DATCOM (Reference 9), a nonlinear six-degree-of-freedom aerodynamics model was generated then implemented incorporated into a VFT simulation, which contained models of all key components for the VFT test. These models included, propulsion, control, equations of motion, weight and balance, and a wind tunnel rig model.

Since the simulation modeled the test article motions and loads in the wind tunnel, it was easily adapted to model balance output and stimulate data reduction routines (Figure 10).

Once test software design was complete, the D-Six simulation environment was configured to provide data acquisition for force, attitude, and rate measurements, in addition to data processing and control of the missile motion while providing information to the engineers monitoring the test. Figure 11 contains a functional diagram of the integrated test arrangement. As the diagram indicates, large portions of code were identical to those in the simulation study. The control law implemented in D-Six project code transmitted control surface commands, reflecting user control or predefined motions, to the test article. Balance and sensor voltages were acquired through data acquisition hardware and provided to data reduction routines that computed pitch attitude, pitch rate, and aerodynamic coefficients. These aerodynamics coefficients converted to forces using user define simulation conditions and summed with propulsive forces from the test article simulation. These forces with aerodynamics moments computed by the simulation were used in the simulation equations of motion to predict trajectory.

The hardware configuration of D-Six was similar to its application during the formation wind-tunnel test. Additional signal conditioning was applied to the analog signals from the strain gauge balances and the canard and pitch attitude pots located in the test missile. It included amplification of the balance signals for optimum resolution during the analog to digital conversion as well as filtering of any high frequency noise in the signal. An accurate power supply was used for power to the strain gauge balance and position pots.

The user interface for the missile tests was configured to display aerodynamic coefficients and pitch rates in real time using graphical strip charts. A dynamic three-dimensional graphical display of the test article was driven with test data was used to monitor test motions remotely. Digital displays of project parameters were also used.

Results

During testing, data were collected from static and dynamic conditions. Static test points were set using the control surface to set the test article attitude. Dynamic tests consisted of a number of ramps, sine wave oscillations with prescribed amplitudes and frequencies, and a series of random motions. For each of the dynamic test cases, simulation initial conditions for weight and balance, velocity, altitude, and thrust used in the trajectory predictions. Figure 12 contains a screen shot of the user interface during one of the dynamics tests. Plots of predicted altitude, pitch angle, and surface deflection are plotted against time in the real-time plot tool.

Discussion

The efforts described above provide support for the use of Windows based "real-time" simulation environments in time-critical applications. Timing tests with a low-stress system load revealed that a simulation environment with DirectX based graphics will execute simulation frames well within three milliseconds of scheduled times. These tests support the applicability of the environment for applications such as hard-ware-in the loop simulation. Because of vast similarities between hardware in the loop simulation and interactive-dynamic wind tunnel testing, the simulation software's applicability was extended to the experimental tasks of data acquisition and test control in two different efforts.

The reconfigurable simulation environment's, D-Six, use in active trim testing in formation and virtual flight testing, allowed engineers to design and test reduction and processing code with greater flexibility, in less time, than with conventional methods. To facilitate test execution, engineers configured the D-Six user interface to include "real-time" parameter monitoring in the form of graphical strip charts and digital readouts. The interface for the VFT effort was further enhanced by with the use of three-dimensional graphics to display test article motions.

Overall, the application of the D-Six simulation environment in the unconventional hardware in the loop scenarios was a success, saving each project time and money in the design and execution of the respective tests.

References

1 Bell, J.W. and O'Rourke, M.J. Application of a Multipurpose Simulation Design, AIAA 97-3798.

2 Gingras, D. "AV-8B II+ Device S2F176 Flight Model Development Using A PC-Based Simulation Environment," AIAA 99-4191, August 1999.

3. Gillies,D, (gillies@ee.ubc.ca), FAQ newsgroup, comp.realtime, www.faqs.org/faqs/realtime-computing/faq/, July 2001.

4 Microsoft Developer's Network Library, April 2001.

5 Blake, W.B and Gingras, D.R. "Comparision of Predicted and Measure formation Flight Interference Effects," AIAA 2001-4136, August 2001.

6 Gingras D.R., Player J.L., and Blake W.B., "Static and Dynamic Wind Tunnel Testing of Air Vehicles in Close Proximity," AIAA-2001-413, August 2001.

7 Marquart, E. J. and Lawrence, F.C. "Virtual Flight Testing (VFT) at the Arnold Engineering Development Center," Presented at ITEA 1999 Conference, Tullahoma, TN, October 1999.

8 Gebert, Glen, Kelly, Joy, and Lopez, Juan; "Virtual Flight Test (VFT) Modeling and Assessment"; TEAS Reference Number: 9800723-90U; 30 September 1998.

9 Air Force Research Laboratory, Wright Patterson AFB, AFRL-VA-WP-TR-1998-3009.

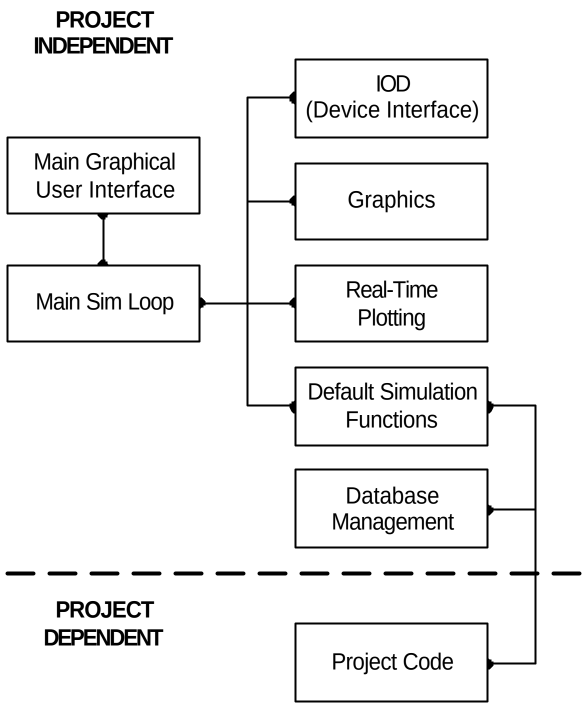


Figure 1. D-Six software structure.

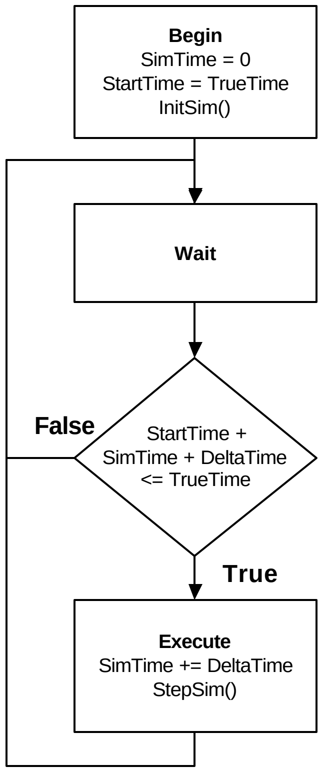


Figure 2. Simulation loop state diagram.

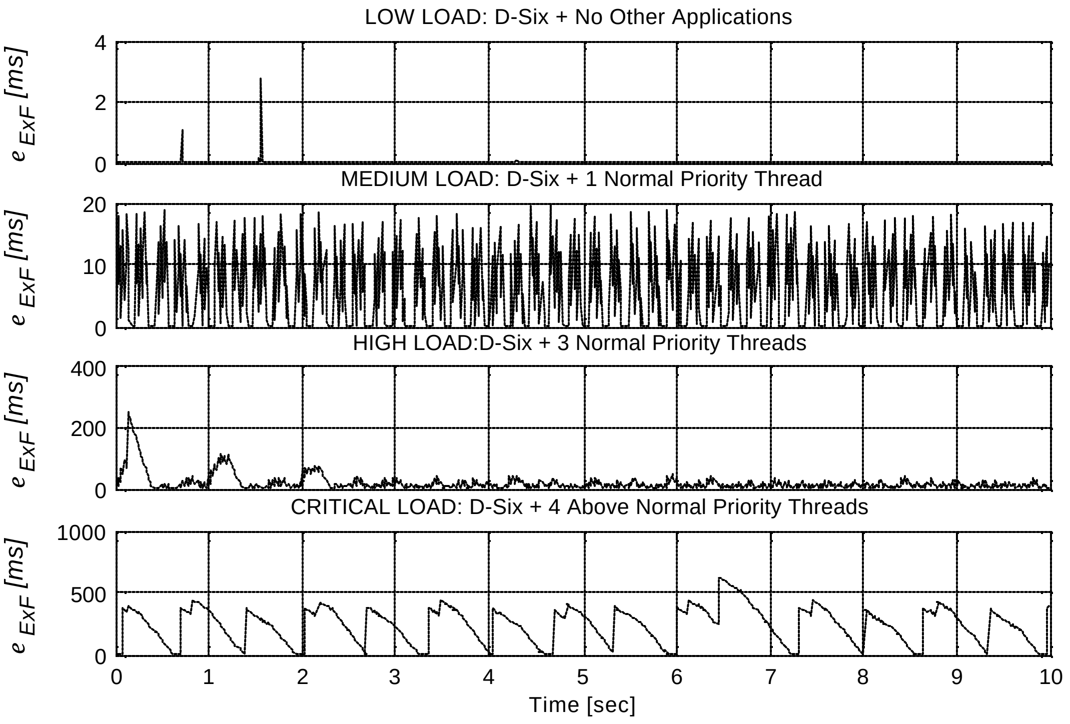


Figure 3. Simulation frame execution scheduling test results.

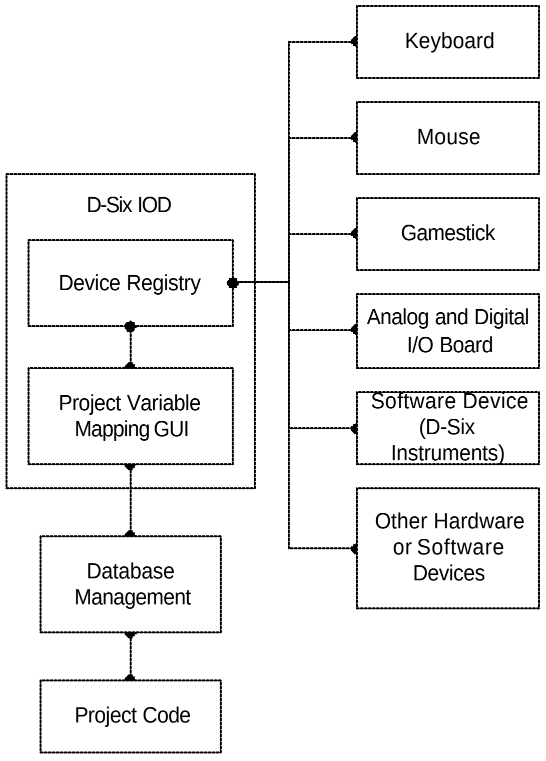

Keyboard
Mouse
Gamestick
Analog and Digital I/O Board
Software Device (D-Six Instruments)
Other Hardware or Software Devices
D-Six IOD
Device Registry
Project Variable Mapping GUI
Database Management
Project Code

Figure 4. The D-Six reconfigurable device interface.

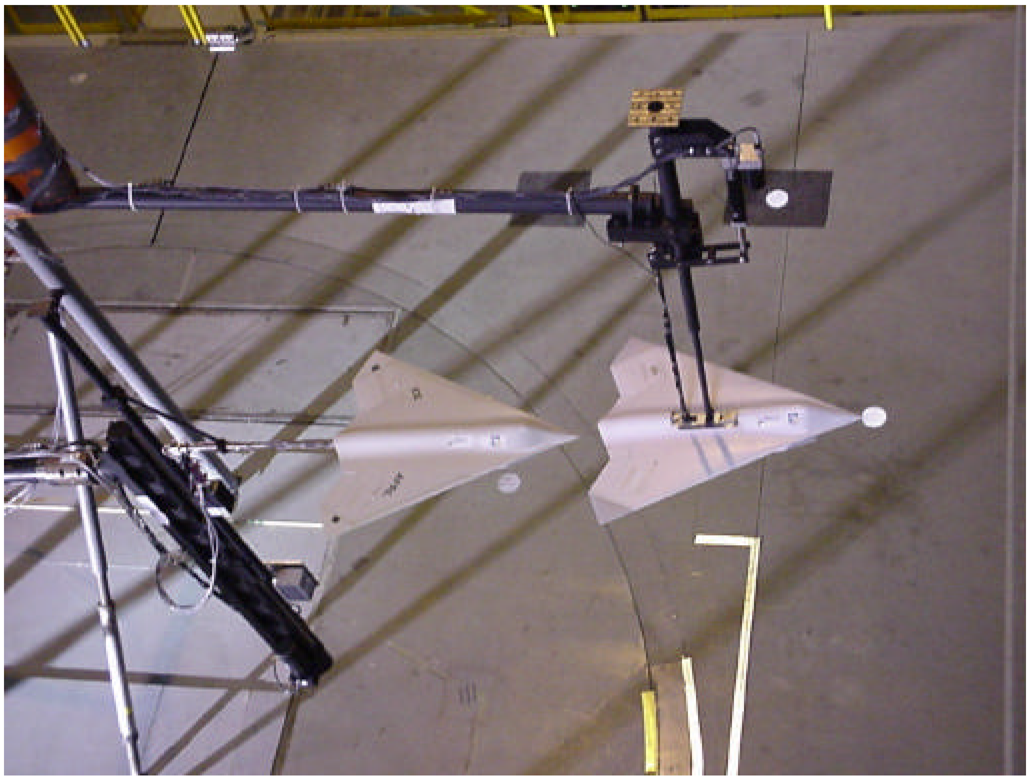


Figure 5. Test articles during two-ship formation-flight testing in the Langley Full Scale Tunnel

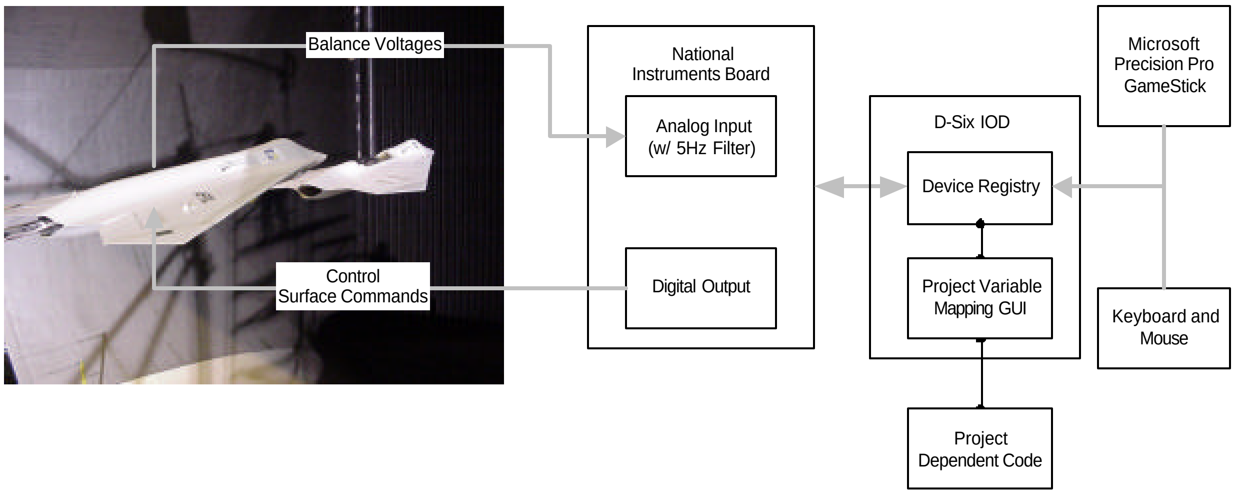


Figure 6. Hardware configuration for the active trimming wind-tunnel experiment

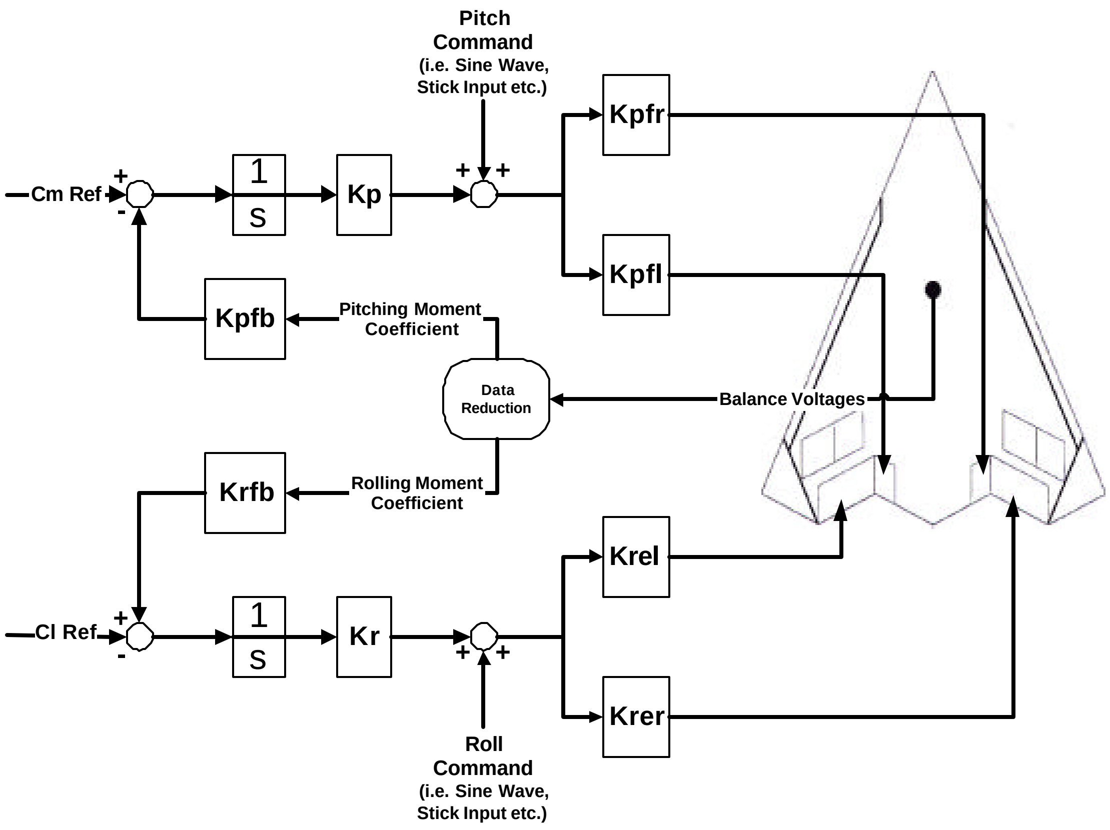

Pitch Command (i.e. Sine Wave, Stick Input etc.)
Cm Ref
+
-
1
S
Kp
+
+
Kpfr
Kpfl
Kpfb
Pitching Moment Coefficient
Data Reduction
Balance Voltages
Krfb
Rolling Moment Coefficient
Cl Ref
Kr
Krel
Krer
Roll Command (i.e. Sine Wave, Stick Input etc.)

Figure 7. Functional diagram of the active trimming control law implemented during testing.

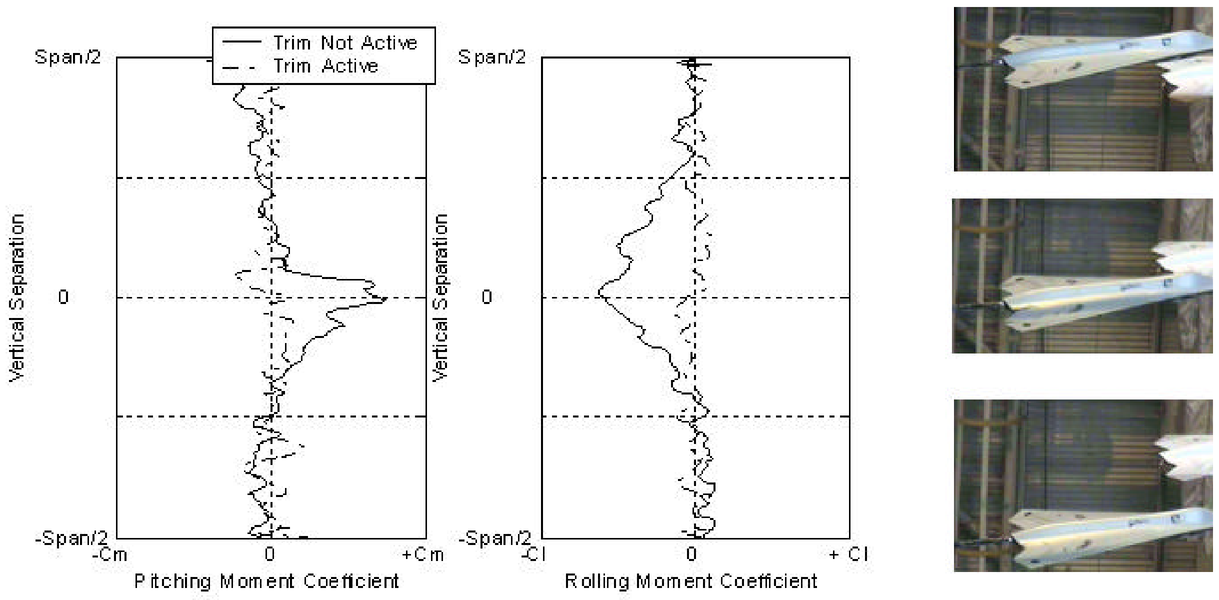


Figure 8. Sample results from active trim control experiments.

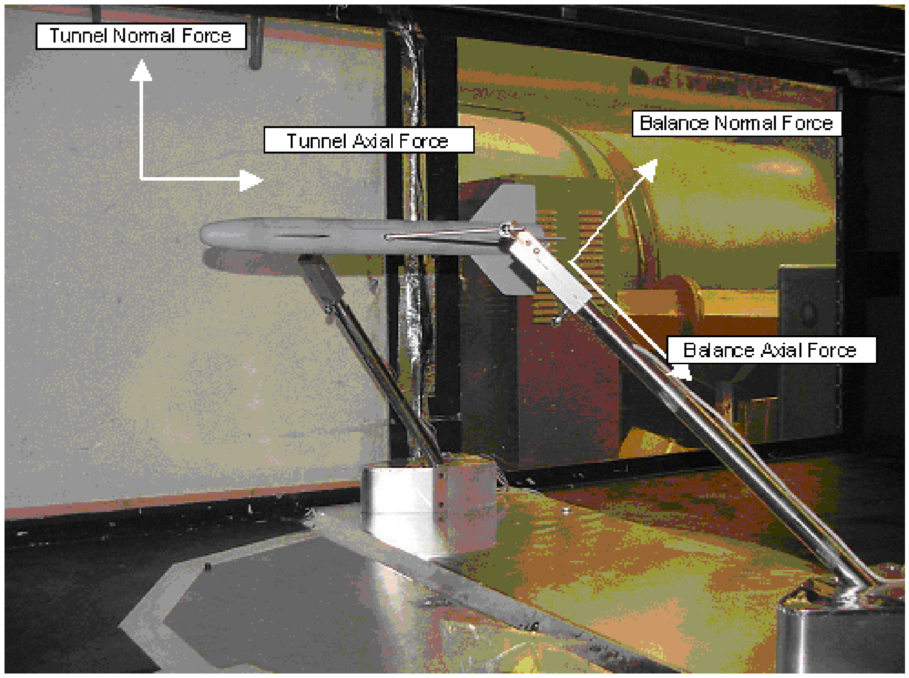


Figure 9. Test article mounted with the VFT custom rig in ODU 4 x 3 tunnel.

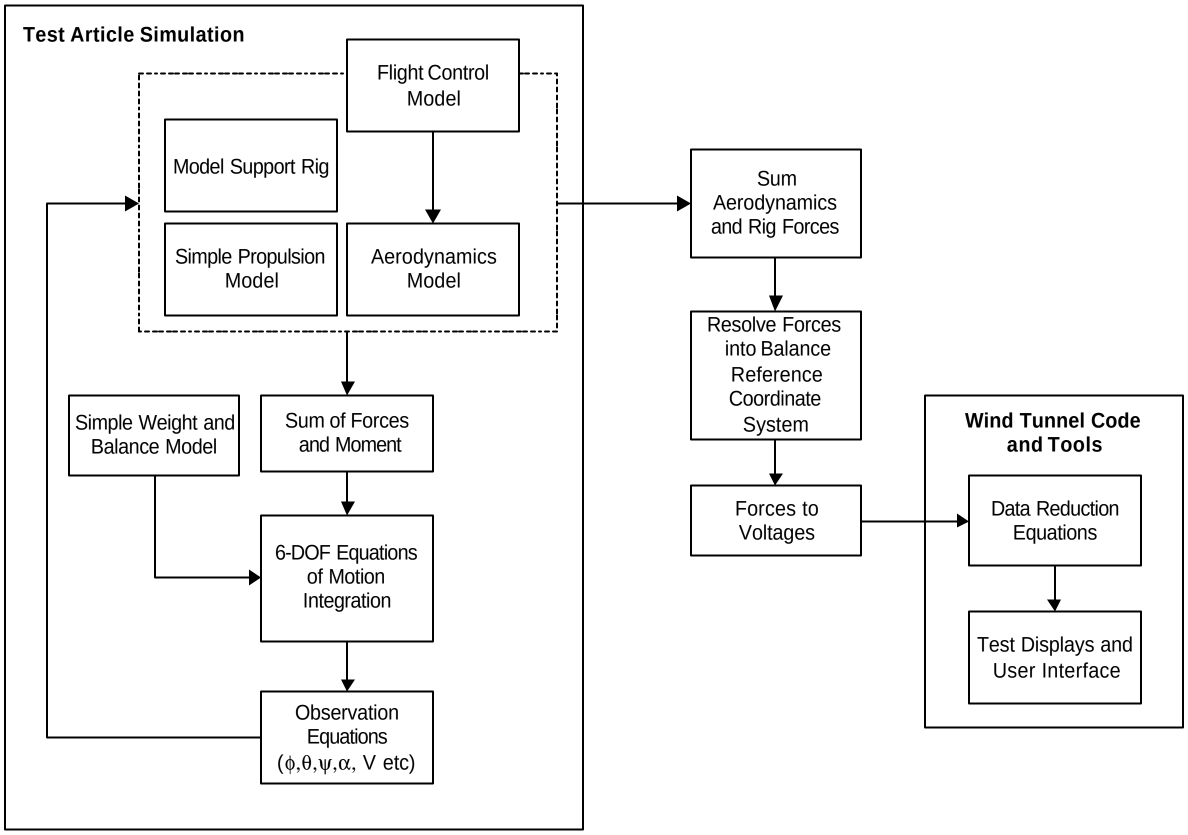

Test Article Simulation
Flight Control Model
Model Support Rig
Simple Propulsion Model
Aerodynamics Model
Simple Weight and Balance Model
Sum of Forces and Moment
6-DOF Equations of Motion Integration
Observation Equations (φ,θ,ψ,α, V etc)
Sum Aerodynamics and Rig Forces
Resolve Forces into Balance Reference Coordinate System
Forces to Voltages
Wind Tunnel Code and Tools
Data Reduction Equations
Test Displays and User Interface

Figure 10. Simulation structure for VFT test design.

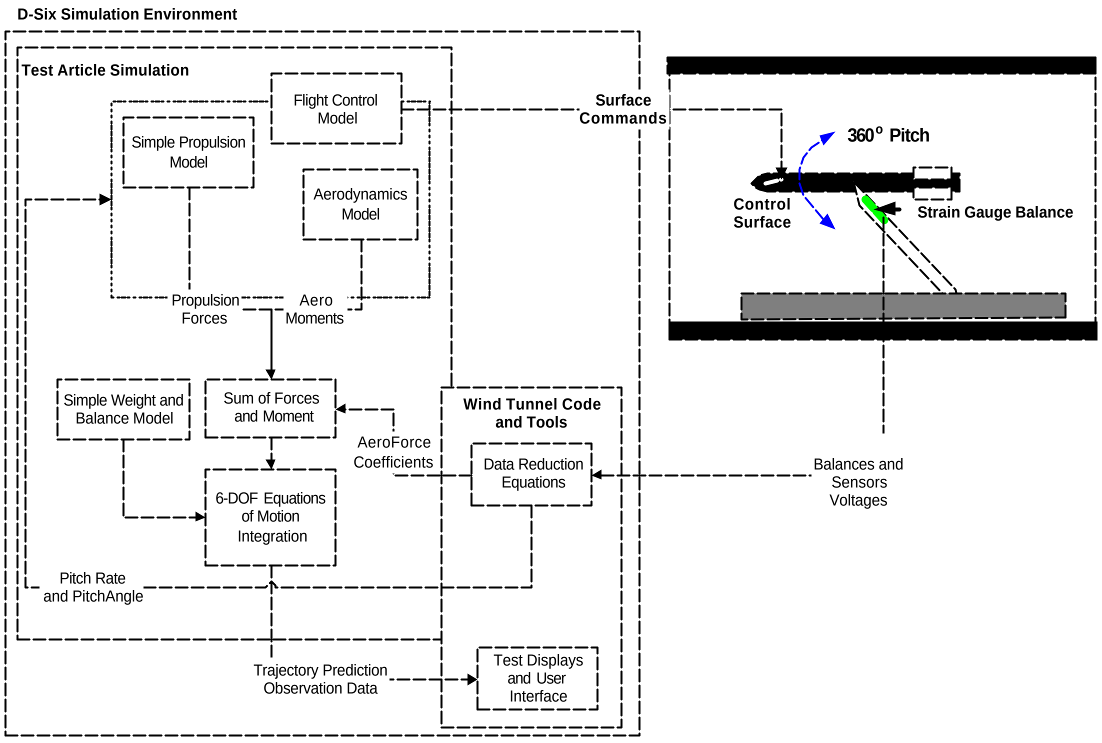
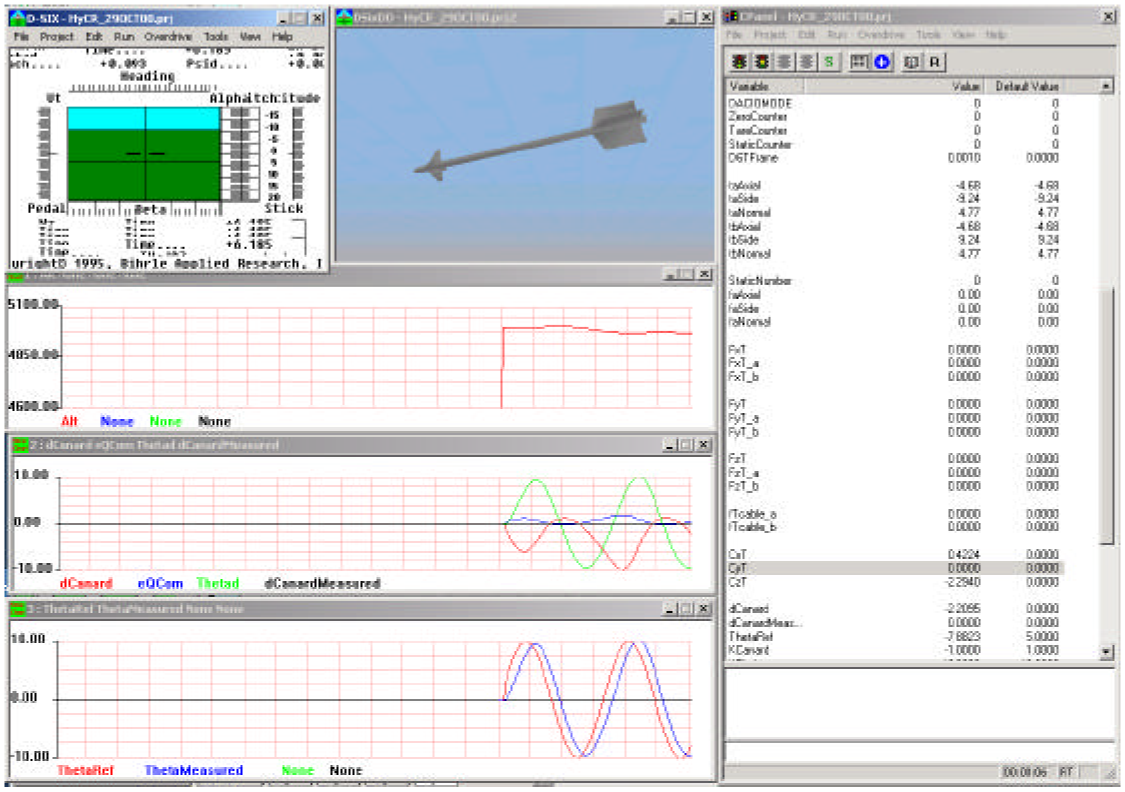

D-Six Simulation Environment
Test Article Simulation
Flight Control Model
Simple Propulsion Model
Aerodynamics Model
Propulsion Forces
Aero Moments
Simple Weight and Balance Model
Sum of Forces and Moment
AeroForce Coefficients
6-DOF Equations of Motion Integration
Pitch Rate and PitchAngle
Trajectory Prediction Observation Data
Wind Tunnel Code and Tools
Data Reduction Equations
Test Displays and User Interface
Surface Commands
360° Pitch
Control Surface
Strain Gauge Balance
Balances and Sensors Voltages

Figure 11. D-Six test article simulation and data reduction integration during VFT.


D-SIX - HyCR_29OCT00.prj
File Project Edit Run Overdrive Tools View Help
Heading
Alphaltcltitude
Pedal
Beta
Stick
Copyright© 1995, Bihrle Applied Research, I
5100.00
4850.00
4600.00
Alt None None None
2 : dCanard eQCom Thetad dCanardMeasured
10.00
0.00
-10.00
dCanard eQCom Thetad dCanardMeasured
3 : ThetaRef ThetaMeasured None None
10.00
0.00
-10.00
ThetaRef ThetaMeasured None None
CPanel - HyCR_29OCT00.prj
00:00:06 RT

Variable	Value	Default Value
DAQMODE	0	0
ZeroCounter	0	0
TareCounter	0	0
StaticCounter	0	0
DGTFlame	0.0010	0.0000
faAxial	-4.68	-4.68
faSide	-9.24	-9.24
faNormal	4.77	4.77
fbAxial	-4.68	-4.68
fbSide	9.24	9.24
fbNormal	4.77	4.77
StaticNumber	0	0
fsAxial	0.00	0.00
fsSide	0.00	0.00
fsNormal	0.00	0.00
FxT	0.0000	0.0000
FxT_a	0.0000	0.0000
FxT_b	0.0000	0.0000
FyT	0.0000	0.0000
FyT_a	0.0000	0.0000
FyT_b	0.0000	0.0000
FzT	0.0000	0.0000
FzT_a	0.0000	0.0000
FzT_b	0.0000	0.0000
fTcable_a	0.0000	0.0000
fTcable_b	0.0000	0.0000
CxT	0.4224	0.0000
CyT	0.0000	0.0000
CzT	-2.2940	0.0000
dCanard	-2.2095	0.0000
dCanardMeas...	0.0000	0.0000
ThetaRef	-7.8823	5.0000
KCanard	-1.0000	1.0000

Figure 12. Screen shot from the D-Six user interface during the VFT effort.