Standards for the Exchange of Simulation Modeling Data

AIAA Modeling and Simulation Technical Committee

Preliminary Draft
Jan 2003

Table of Contents

1.0 INTRODUCTION

2.0 STANDARD SIMULATION VARIABLES

3.0 STANDARD SIMULATION AXIS SYSTEMS

4.0 STANDARD SIMULATION TABULAR DATA

5.0 STANDARD TIME HISTORY DATA FORMAT

6.0 DELETED

7.0 STANDARD SIMULATION MODEL VERIFICATION TESTS AND PROCEDURES

1.0 INTRODUCTION

1.1 Purpose of this Document

The purpose of this document is to present a standard for the interchange of simulation modeling data between facilities. As illustrated in Figure 1, the initial objective is to allow a person with a simulation of a certain type of vehicle or aircraft at facility A to transfer the simulation to facility B in an easy, straightforward manner. It is the objective of this document to form the foundation for more detailed and rigorous standards as discussed below.

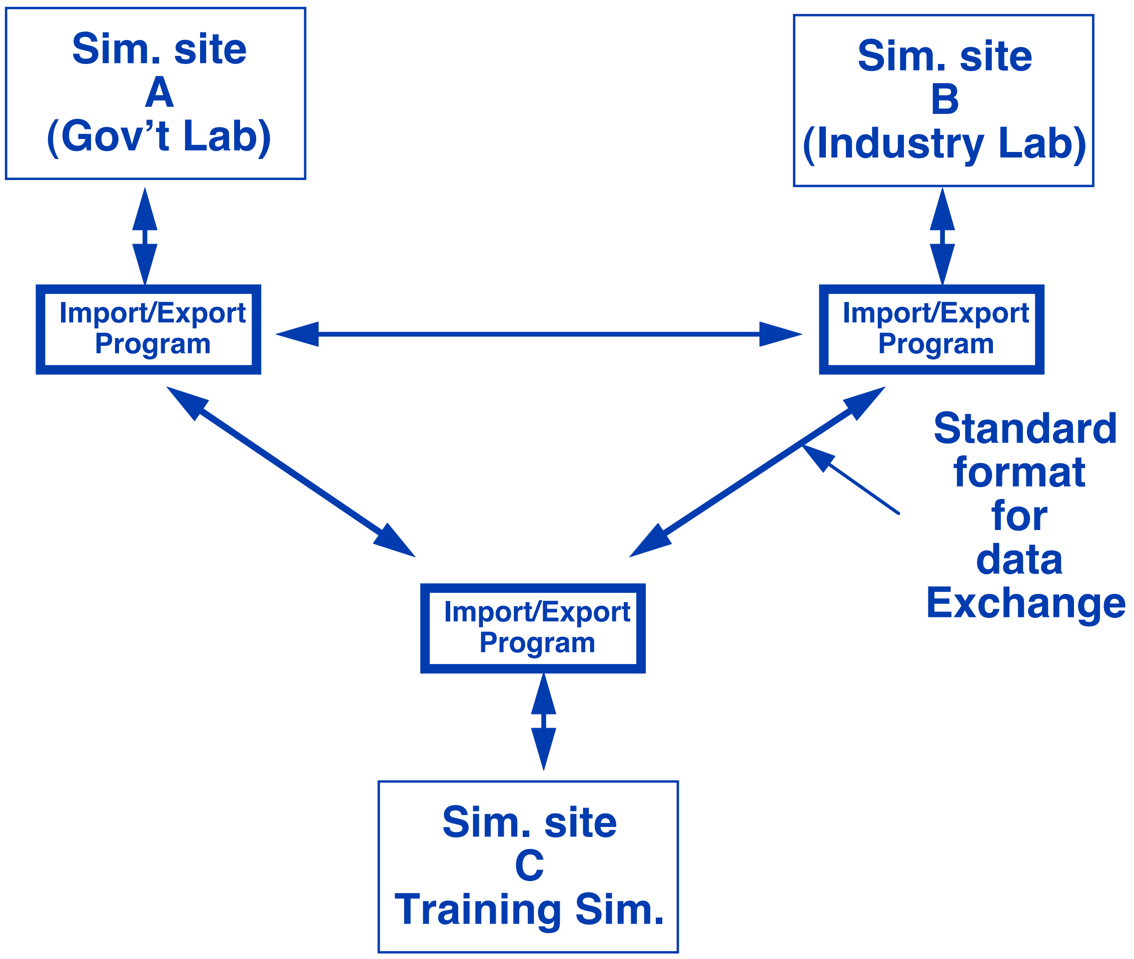


Figure 1. A Standard Format for Exchanging Simulation Modeling Data Between Simulation Facilities

Given a standard format for the exchange of simulation modeling data, the user community investment required to use the standard is minimal. Each facility internal standards and conventions may still be used. The only work required to commit to the standard is to write "translation" programs to import and export data from/to the standard. This one time investment would then allow any model to be exchanged easily and with good convidence that the “rehosting” of the model would require minimal effort. The user need not change anything inside his facility or software.

The first logical step in a simulation standard is to define and standardize the simulation modeling information sent between users. This forms the foundation for all other standards developed. They will all build upon this standard. It is also key to user acceptance of the standards because this standard is of immediate benefit to them, allowing them to more easily import simulations from other users. The logical development of simulation standrads is illustrated in Figure 2.

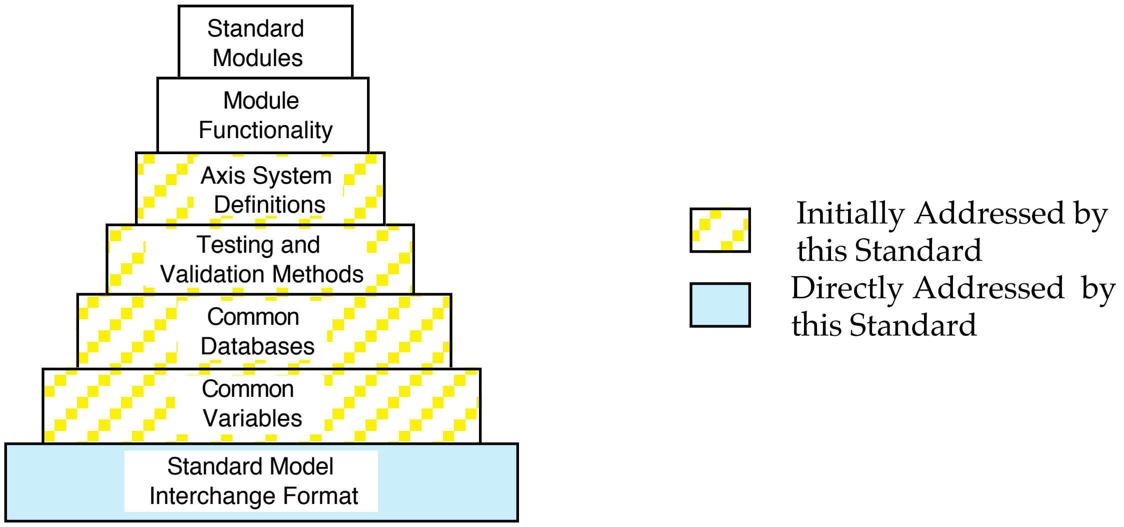

Standard Modules
Module Functionality
Axis System Definitions
Testing and Validation Methods
Common Databases
Common Variables
Standard Model Interchange Format
Initially Addressed by this Standard
Directly Addressed by this Standard

Figure 2. Standards Addressed by this Document

This level of standard is foundational in that, to exchange information between users at least the following must be clearly defined:

a) The variables or parameters being exchanged. This includes clear, unambiguous description, units, sign convention, axis system (if applicable), etc.;

b) Data format for the exchange. This includes parameters, time history data, and function table data; and

c) Method of verifying the information was exchanged correctly.

The above have clear relationships to and set the foundation for later levels of standardization including common variables, databases, testing, and axis systems. However, at this level of standard, they are easier to define because they are more limited in scope and, therefore, easier for the user to accept.

1.2 Participants in the Development of this Standard

1.3 Future Standards

Figure 2 in Section 1.1 above, shows the logical progression of standards in the subject area. It is the intent for this standard to form the foundation of those standards ultimately resulting in certain standard software modules in ANSI standard languages.

There is also the objective to submit this standard to ISO for consideration as an ISO standard.

1.4 If you want to Propose a Change

It is the philosophy of this group that standards should be flexible, and are not rigid. Instead, standards move and change with the needs of the user community. However, for a standard to be effective, change must be controlled and reasonable otherwise a standard is counterproductive. Proposed changes to this standard are solicited and encouraged. We suggest you submit any change, be it correction of an error or addition of a standard axis system or variable name to the following address:

AIAA Modeling and Simulation Technical Committee
AIAA Standards
372 L'Enfant Promenade S.W.
Washington, D.C. 20024-2518

1.5 How to Obtain Copies

Copies of this document may be obtained by calling 1-800-682-AIAA, or by writing to the following address:

AIAA Technical Publications
372 L'Enfant Promenade S.W.
Washington, D.C. 20024-2518

2.0 STANDARD SIMULATION VARIABLES

2.1 Background / Philosophy

Long term software maintenance of simulation software used to model the flight dynamics of an airplane is predicated upon identification of the states and controls in the simulation. The importance of this cannot be overstated. States and inputs (controls) are determined by the physics of the problem. Since the physics are immutable (although we have all seen software where they're not) the identification of what these variables are is crucial in software maintenance.

Again, according to physics, all outputs which we use in simulation are derived from states and inputs.

By definition, anything in a simulation we are interested in is an output. To create an output, for example indicated airspeed, we must be able to identify the states and inputs. Therefore, if we know the appropriate law of physics, we may correctly compute indicated airspeed. Too often in simulation modeling we forget these fundamental immutable concepts. We "fudge" things and create outputs from other outputs. Practically speaking, this is done because we can't determine what the states in the simulation are. That is, what is computed from what since it is an iterative process, it becomes very cloudy as to what variable is dependent upon what other variable.

This is why we must go back to the physics and realize everything is computed from states and inputs in the mathematical and physical sense.

Now the question becomes which state, that is, state at what time? Again, this becomes clear if we go back to the physics. Outputs at any time T are a function of the states and inputs at time T. Integration of the state derivatives at time T results in states at time T plus delta T. State derivatives at time T are functions of the states and inputs at time T. It is crucial that we do not mix variables at time T with variables at time T plus delta T.

Practically speaking, for simulation standards, what this means is that all integrations must be done in a centralized location in each simulation loop, otherwise we mix variables at time T with variables at time T plus delta T. The simulation industry has violated this mathematical principle for many years in the use of "in-line" difference equations for simulation filters and actuators, etc. This often works "OK" and "no harm is done". However, what is missed is that software maintenance becomes much more difficult when those states cannot be located because they are embedded—they are strung throughout the code. Therefore, the outputs cannot be properly created. Furthermore, when a modification comes to add a capability or to fix a bug, the key variables required to modify a simulation, again what would the states, state derivatives, and inputs, are impossible to find or inaccessible. Therefore, the fix made to the simulation is less than optimum and possibly creates more headaches down the road for the next fix, etc. , etc., ad infinitum.

Therefore the proposed AIAA Simulation Variable standard identifies states, state derivatives and inputs as part of the naming convention.

2.2 Variable Naming Convention

Proposed Convention

This paragraph will discuss the convention and philosophy used for naming of simulation variables. The purpose of this is so when other variables are added to the list they will follow the same general convention.

Both the C and Ada language conventions are supported. The only "oddity" is that in the C convention we us an underscore to separate the prefix and suffix from the body of the variable name. Conventionally, C does not use underscores in variable names at all.

General rules for naming variables:

> Variables shall have meaningful names. Mnemonics will not be used. Standard abbreviations are allowed.
>
> (C language) Distinct words in variable names shall be separated by capitalization of the separate words (example" BodyXAcceleration"). (Ada language) Distinct words in variable names shall be separated by underscores between the separate words (example" body_X_acceleration").
>
> Varaible names shall not exceed 60 characters in length. Brief, but complete names are most effective.
>
> The first letter of each word will be capitalized and the remaining lower case. This improves readability. Abbreviations are all capitals. Units should be all lower case. This reduces ambiguity.

Methodolgy for Creating New Names

The suggest method of creating the name is as follows.

> Eacn name up to six components. All components are not required to be used because in many cases they do not apply. these components are:

(prefix)_(variable source domain)_(axis or reference system)_(specific axis or reference)_(core name)_(units)

Prefix
The purpose of the prefix is to identify the variable in two ways. The prefix is used to identify differences between the own ship and threats or secondary

vehicles. The prefix is also used to identify the most important dynamic variables in the simulation, the states, the state derivatives, and the inputs.

The prefix is always separated from the body of the variable by an underscore.

Identification of States The states and state derivatives are those variables which make the simulation dynamic and are essentially the key variables in a real time flight simulation. Basically, anything that is integrated (mathematically) is a state derivative. The result of the integration is the state (integration of the state derivative results in the state). This is true for any integration in a simulation. Obviously then, if the user controls all the states, he controls the motion of the simulation. Also, these along with the inputs are the key variables for validation. All outputs are computed directly or indirectly from states and inputs.

Identification of the Vehicle The purpose of the prefix is also to indicate the threat or friendly simulations. For example, if the simulation also has a threat model there will be more than one position. There will be position for the own ship and a position for the threat. If there are n threats there will be n positions. The naming convention will allow us two acceptable manners of handling this. The variable names may be put in an array that each index of the array indicates a different aircraft. Alternatively, there can be a prefix T1 through Tn indicating Threat 1 through N followed by the standard variable name. If there are other friendly aircraft in this part of the simulation they will be prefixed by F1 through Fm. Examples are given below. In this case F0 would indicate own ship. Absence of an F or T prefix indicates ownship as the default.

C language Examples:

`S_BodyXVelocity_fps`	`s_ prefix indicates that this variable is a state`
`Sd_BodyXAcceleration_fps2`	`sd_ prefix indicates that this variable is a state derivative`
`F2_S_BodyXVelocity_fps`	`F2_ prefix indicates that this is the "s_Body_X_Velocity_f_p_s variable of the #2 friendly aircraft or vehicle.  Note that the s_ prefix indicates that this is also a state variable.`
`T1_TrueAirspeed_fps`	`T1_ prefix indicates that this is the "true_airspeed_f_p_s variable of the #1 threat aircraft or vehicle`

Ada language Examples:

s_body_X_velocity_f_p_s	s_ prefix indicates that this variable is a state
sd_body_X_acceleration_f_p_s2	sd_ prefix indicates that this variable is a state derivative
F2_s_body_X_velocity_f_p_s	F2_ prefix indicates that this is the "s_Body_X_Velocity_f_p_s variable of the #2 friendly aircraft or vehicle. Note that the s_ prefix indicates that this is also a state variable.
T1_true_airspeed_f_p_s	T1_ prefix indicates that this is the "true_airspeed_f_p_s variable of the #1 threat aircraft or vehicle

Variable Source Domain

This is the object domain in which the variable is used. In object oriented design, it would logically be the object. The parameter souce domain is normally not included if it (or the object) is the vehicle or aircraft being simulated (ex.).

Some domain examples:

Aero or Aerodynamic
Engine or Thrust
Controls
Wheel
Landing_Gear
Hydraulic
Electrical

Variable name examples:

C:
```
AeroBodyXForceCoefficient
AeroBodyXForce_lbf
ThrustBodyYForce_lbf
```

Ada:
```
aero_body_X_force_coefficient
aero_body_X_force_lbf
thrust_body_Y_force_lbf
```

Axis or Reference System

This is the axis or reference system to which the variable is referenced. The standard axis system abbreviations are used. If no axis system pertains to the variable of the core variable name needs no reference system to be unambiguous (ex. airspeed) then this part of the variable name may be omitted.

The following are some axis system examples which corrospond to the standard axis systems discussed below.

GEaxis	for the Geocentric Earth Fixed-Axis System.
Body	for the Body axis system
FEaxis	for the flat earth inertial axis system

Variable name examples:

```
s_Body_X_Velocity_f_p_s
GEaxis_Z_Velocity_f_p_s
```

<u>Specific Axis or Reference</u>

This is the specific axis or reference used within the axis system. If the axis or reference system component of the name is included in the name, then the specific axis or reference should also be included.

For example:
X, Y, or Z for linear motion
Pitch, Roll, or Yaw for angular motion

Variable name examples:

```
s_Body_Roll_Rate_rad_p_s
Body_X_Turbulence_Velocity_f_p_s
```

<u>Core Variable Name</u>

This is the most specific (hence core) name for the variable. All variable names must include this component of the name.

Core variable name examples:

```
Cosine_Of_Angle_Of_Sideslip
Rate
Turbulence_Velocity
Velocity
```

Variable name examples:

```
Cosine_Of_Angle_Of_Sideslip
s_Body_Roll_Rate_rad_p_s
Body_X_Turbulence_Velocity_f_p_s
GEaxis_Z_Velocity_f_p_s
```

<u>Suffix</u>
The suffix is used to describe the units of the variable.

The convention for the suffix is simple and is followed for all variables. This will allow the user, the programmer, and the reader of the code to check for homogeneity of the units and is obviously self-documenting in this respect.

Therefore, the units will be put on all variables except variables that are non-dimensional which will have no units. This also has the other significant advantage of making this standard consistent and acceptable in countries with the international system of units. For example, airspeed is just as acceptable as a standard both for the American system of units and the International system of units. In one case it's

airspeed_f_p_s	for feet per second (f/s)

and in the other case

airspeed_m_p_s	for meters per second (m/s)

The standard defines what true variable name for air speed is, the user defines what units he is using .

The suffix is always separated from the body of the variable name by an underscore.

The standard unit notations are given below. The seven *Système Internationale d'Unites* (SI) units and standard abbreviations are included.

time		
hours	h	
seconds	s	(SI Standard)
minutes	min	
milliseconds	ms	(milli prefix m)
length		
feet	f	
meter	m	(SI Standard)
nautical miles	nmi	
statute miles	smi	
kilometers	km	(kilo prefix k)
centimeters	cm	
millimeter	mm	(milli prefix m)
Force		
pound force	lbf	
Newton	N	
kilogram force	kgf	
Mass		
gram	g	
kilogram	kg	(SI Standard)
pound mass	lbm	
slug	slug	
Plane Angle		
degrees (angular)	deg	
radians	rad	
revolution	rev	

Temperature		
degrees Rankine	R	
degrees Centigrade	C	
degrees Kelvin	K	(SI Standard)
Power, energy, work, heat		
British thermal unit	btu	
erg	erg	
calorie	cal	
joule	jou	
horsepower	hp	
Electrical		
volt direct current	vdc	
volt alternating current	vac	
ampere	A	(SI Standard)
cycles	cyc	
watt	watt	
henry	hy	
farad	fd	
ohm	ohm	
Other		
candela (luminous intensity)	cd	(SI Standard)
mole (amt. of substance)	mol	(SI Standard)

<u>Discarded Conventions and Reasons</u>
We have considered having a prefix for simulation outputs as well as states but at the present time this has been discarded due to the fact that the outputs required vary so widely.

We considered eliminating the suffix when the units were one of the "standards" but eliminated this due to the fact that always having the units attached to the variable will help the programmer have consistent units when he is programming and reduce programming errors due to mixing of the units improperly.

Another naming convention that was considered and discarded was the use of prefixes and or suffixes or other naming structures to designate software organization. This was discarded based on the philosophical fact that variable names should not designate software organization. When the AIAA standard develops to the point in the future where there is standard software, architecture or structure, these variable names would be used in that software but will not be changed.

<u>Summary</u>

While It is strongly recommended that this naming convention be followed for all future variables, the real key to a "standard variable name" Is not the name, but the definition of the name. To exchange Information between two or more organizations, the most Important factor Is not whether a variable Is named "airspeed" or "as", but what Is the precise, unambiguous definition of the variable, Including units and axis system.

2.3 Standard Variable Names

see excel table

2.4 References

3.0 STANDARD SIMULATION AXIS SYSTEMS

3.1 Background / Philosophy

The axis system definitions discussed herein were taken from existing standards, the the *ANSI/AIAA Recommended Practice for Atmospheric and Space Flight Vehicle Coordinate Systems* (ANSI/AIAA R-004-1992) [Reference 3.1]and the *Distributed Interactive Simulation* (DIS Application Protocols, Version 2, IST-CR-90-50, March 1994) [Reference 3.2]. Reference 3.1 standard is based on ISO Standard 1151-1 of 1988 and 1151-3 of 1972.

Axis system standards also are reflected in the variable naming convention. When applicable, the axis system is included in the variable name.

3.2 Axis System Conventions

Proposed Convention

In general, ANSI/AIAA R-004-1992 [Reference 3.1] should be referred to as the definitive reference on axis system definitions. However, it is important to emphasize the correlation of reference [3.1] and [3.2] axis systems.

Geocentric Earth Fixed-Axis System

The Geocentric Earth Fixed-Axis System (Axis System 1.1.3 of Reference 3.1) is identical to the DIS "Geocentric Cartesian Coordinate System" (also referred to "World Coordinate System" of Reference 3.2).

It is a system with both the origin and axis fixed relative to, and rotating with, the earth. The origin is at the center of the earth, the x_G axis being the continuation of the line from the center of the earth through the intersection of the Greenwich Meridian and the Equator, the z_G axis being the mean spin axis of the earth, positive the north, and the y_G axis completing the right hand triad.

All variables in the simulation referenced to this axis system refer to the "GE" for the Geocentric Earth Fixed-Axis System.

This system is fixed to and rotating with the earth.

Body Axis Coordinate System

The next standard axis system is the Body Axis System (axis system number 1.1.7 in Reference 3.2). This is identical to the DIS "Entity Coordinates System" in Reference 3.2.

It is a system fixed in the vehicle with the origin at the center of mass consisting of the following axis orientation. The x axis is in the reference plane of the vehicle, or if the origin is outside that plane, is in that plane through the origin that is parallel to the reference plane, and positive forward. The reference plane is the plane of symmetry or clearly specified alternative. The y axis is normal to the reference plane and positive to the right. The z axis lies in or is parallel to the reference plane and completes the orthogonal right hand triad.

The body axis system is referred to in the variable names as "Body".

Flat Earth Axis System

The third axis system is defined only for convenience on creating validation data. It is a fixed, non-rotating, flat earth with no mapping to a round earth coordinate system, therefore, latitude and longitude are meaningless. The purpose of this coordinate system is to allow, if desired, vehicle checkout simulation to be performed in this axis system. This simplifies the use of this standard by the simulation facilities which do not normally use a round or oblate spheroid, rotating earth model.

The Flat Earth reference system is situated on the earth's surface directly under the cf of the vehicle at the initialization of the simulation. The x axis on the local frame points northwards and the y axis points eastward, with the z axis down. The x and y axis are parallel to the plane of the flat earth.

The flat earth axis systems is referred to in the variable names as "FE".

Discarded Conventions and Reasons

There are many other possible axis systems, several more are in referece 3.1. However other axis systems may be easily transformed to these and these axis systems will satisfy the needs of the vast majority of the users.

Summary
It is strongly recommended that this axis system convention, and those in Reference 3.1, be followed for all future equations of motion. If it is not a satisfactory system, the convention should be modified and all the defined axis systems modified according to the new convention.

3.3 References

3.1 *ANSI/AIAA Recommended Practice for Atmospheric and Space Flight Vehicle Coordinate Systems* (ANSI/AIAA R-004-1992).

3.2 *Distributed Interactive Simulation* (DIS Application Protocols, Version 2, IST-CR-90-50, March 1994).

4.0 STANDARD SIMULATION TIME HISTORY TABULAR DATA

4.1 Background / Philosophy

Probably the most basic standard of benefit to the simulation discipline is one that defines formats for the interchange of tabular and time history data. Tabular data is used almost universally for function generation of aerodynamic, atmospheric, and many other parameters. Time history data is used in the verification, validation, and qualification of flight simulators. The easy interchange of such data can greatly improve efficiency in the simulation community.

Most simulation developers and users have addressed this issue locally. And, in many communities, a family of tools has been built around existing local standards. Thus, the intent is not to obsolete these local standards, but rather to define a format for communication which will allow each site to develop a single format converter to and from their local format. It is hoped that a good standard will eventually be adopted for local use as well, but that is not required for the standard to succeed.

The approach taken in establishing this standard has been to examine standards already available. The search has turned up two potential standards: (1) the Hierarchical Data Format (or HDF) and (2) the Network Common Data Format (or netCDF).

NetCDF is an interface for scientific data access and a freely-distributed software library that provides implementation of the interface. It was developed at the Unidata Program Center in Boulder, Colorado. The interface, library, and format support the creation, access, and sharing of scientific data.

HDF is a multi-object file format for the transfer of graphical and numerical data between machines. Data models supported include raster images, color palettes, scientific data sets, text entry, binary tables. It was developed by The National Center for Supercomputing Applications (NCSA), located at the University of Illinois at Urbana-Champaign.

The netCDF library has been merged into HDF 3.3. HDF and netCDF files can now be read and written transparently.

4.2 Time History Data Format Conventions

<u>Proposed Convention</u>

The proposed convention is to utilize the Hierarchical Data Format (HDF). While continuing to evolve, HDF is maturing rapidly and is well documented, non-proprietary, and readily available on-line. Some salient features are:

1) HDF is a versatile file format. It supports six different data models. Each data model defines a specific type of data and provides a convenient interface for reading, writing, and organizing a unique set of data elements.
2) HDF is a self-describing format, allowing an application to interpret the structure and contents of a file without any outside information.
3) HDF is a flexible file format. With HDF, you can group sets of related objects together and then access them as a group or as individual objects. There are pre-defined sets for raster images and floating point multidimensional arrays.
4) User can also create their own grouping structures using an HDF feature called vgroups.
5) HDF is an extensible file format. It can easily accommodate new data models, regardless of whether they are added by the HDF development team or by HDF users.
6) HDF is a portable file format. HDF files can be shared across platforms. An hdf file created on one computer, say a Cray supercomputer, can be read on another system, say IBM PC, without modification.
7) HDF is available in the public domain.
8) Documentation, libraries, and utilities are available free on-line, or a CD-ROM can be ordered for a nominal charge.

Discarded Conventions and Reasons

Many conventions exist at various simulation sites. They were eliminated from consideration, not because they lack merit, but because they can't compete with the HDF in terms of maturity, availability, and documentation. Using an existing format saves resources and offers the benefit of compatibility with others in the scientific and engineering community. HDF is also more robust than most local conventions in that it will handle raster images and other data types of interest in the simulation discipline.

4.3 Proposed Time History Data Format

Time history data will be handled as a special case of tabular data (i.e. a function of one independent variable: time).

4.4 References

Fact sheets, answers to frequently asked questions, reference manuals, newsletters, source code and more are all available on-line at the following addresses:

netCDF information:
http://www.unidata.ucar.edu:80/packages/netcdf/

HDF information:
http://hdf.ncsa.uiuc.edu:8001/

Anonymous FTP for NetCDF software:
ftp.unidata.ucar.edu:pub/netcdf/

Anonymous FTP for HDF software:
ftp.ncsa.uiuc.edu/HDF/

4.5 Example

5.0 Standard Function Table Data Format

5.1 PURPOSE:

The purpose of this section is to explain the data requirements for which a Standard Function Table Format must be able to satisfy. This discusses theoretical information contained in the table and configuration management of the data in the table. As you will see the format of the table includes data for all these components.

This document also discusses conceptually how the data table should be looked up in an executable program. Although, it does not provide a program or function to perform table look-ups given data developed according to this standard.

5.2 DEFINITIONS:

Breakpoint - The value of the independent variable of a given dependent variable, or the X coordinate (or abscissa of) a one dimensional table, for example.

Confidence Interval - An estimate of the computed or perceived accuracy of the data.

Dependent Variable - The output of the function table is the dependent variable, in the example given in this document, CL Alpha is the dependent variable.

Independent Variable - The variable or variables of which the independent variable is a function. In the example, angle-of-attack, Mach number, and delta S are independent variables.

One Dimensional Table - A table where there is only one independent variable. For example, C_L(□).

Two Dimensional Table - A table where there are two independent variables. For example, C_L(□,□).

BACKGROUND:

Figure A presents a fairly standard three-dimensional set of data as is typical of aerodynamic data from flight test or from a wind tunnel. In the example given, lift coefficient is a function of angle-of-attack, Mach number, and a control position. Close examination of the data given will reveal the following characteristics:

1. The number of breakpoints of the independent variables varies for each independent variable. Not only are there a different number of angle-of-attack breakpoints, but also a different number of Mach number and control position breakpoints.

2. The values (breakpoints) of the independent variables are different.

3. The valid ranges of the independent variables are different.

4. The above three differences are not consistent for all the data. For example, in the example table the angle-of-attack, breakpoints for Mach = 0.6, and Mach = 0.7 for delta S = -5 are identical.

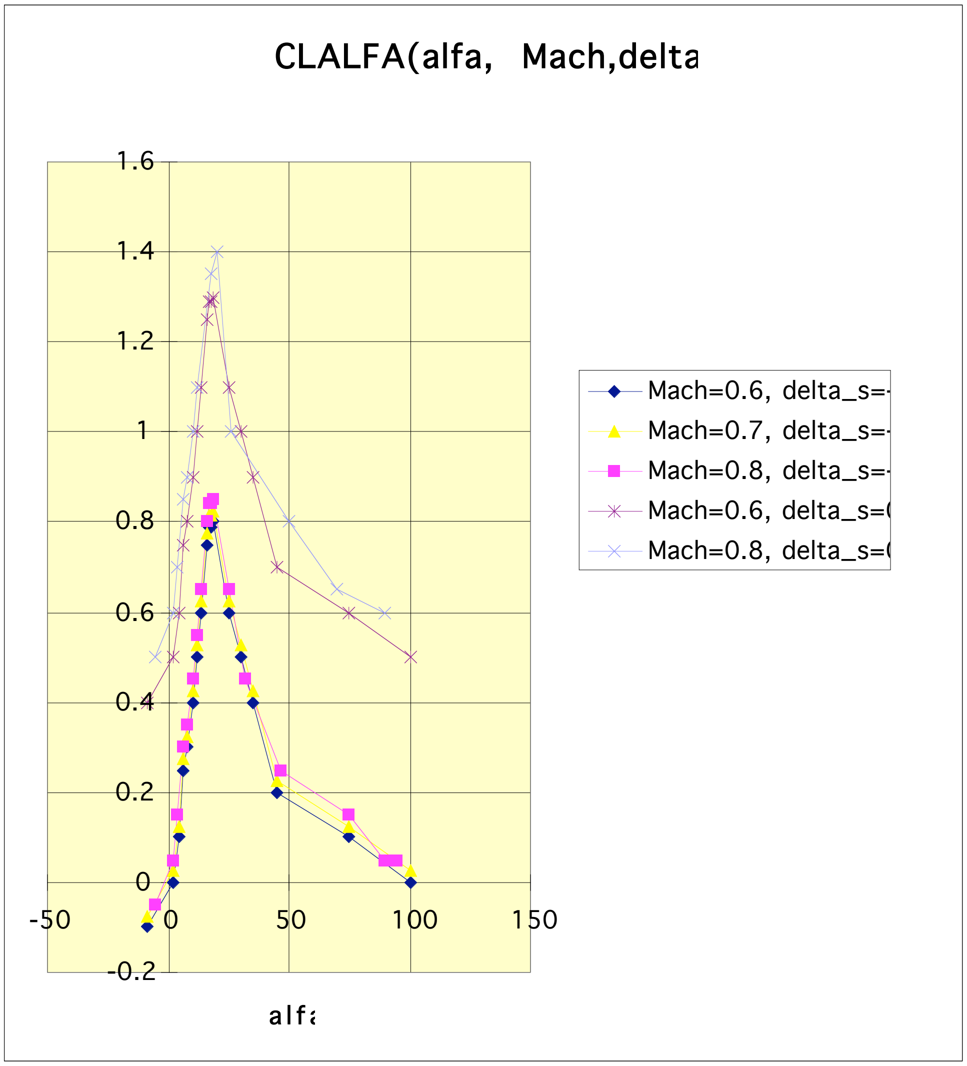


Figure A. An Illustration of a 3 Dimensional Function Table, CL_Alpha)alfa, Mach, Delta_s)

For the function data there is other information that is of significant importance to the user, without which the data is not very useful. In general this information is:

- where did the data come from?
- how is it defined?
- what are the units of the output (the dependent variable) and the independent variables?
- what is the sign convention of the independent and dependent variables? For example, is the control position positive trailing edge up or trailing edge down? Exactly which control surface is it?
- who created the table?
- how has it been modified and for what reason?

There are also different additional information about the table that is not presented in the plot:

- Most significantly is, how accurate is the data estimated to be? Or, mathematically what is the confidence interval of the data?
- By what method is the data intended to be interpolated? For example, linear interpolation or bi- cubic spline interpolation?
- By what method is the data intended to be extrapolated for data with different ranges?

The standard data format has data elements that contain all of the above information.

DESIGN OBJECTIVES OF THE STANDARD DATA TABLE

The design objectives of the Standard Data Table Format were first and foremost to make a data format that would meet all the requirements of real data, some of them shown above. This notably is the fact that the general case of the independent variables for a multi-dimensional table, the independent variables have different number of breakpoints, different breakpoints, and different valid ranges. An equally important design objective was to allow the table to contain information on where the data points come from (a reference) and a confidence interval for the data. Confidence Intervals can be used to mathematically combine two different estimates at the same parameter at the same point; therefore, are extremely valuable when attempting to update a data set (however the user must be careful as not all confidence intervals are equivalent, or even meaningful). Additionally, the table has to be easy to read by the computer and the human being, and be self-documenting as much as possible.

BUILD-UP OF A MULTI-DIMENSIONAL TABLE

Any multi-dimensional table is made up of several one dimensional tables. In the same manner the multi-dimensional table is a concatenation of single dimensional table formats. There is the exception when there is a multi-dimensional table where the breakpoints of the independent variables are identical and a unique table definition may used. However, this is a special case. The general case will be discussed below.

Figure blank shows the single dimension function of CL ALFA as a function of angle of attack. This is the simplest case, but still provides a good illustrative example to explain how the function tables are formed and used.

Functions are represented as text that includes a header, which is information about the table, and the tabular data itself. The header starts with the name of the dependent variable and describes the independent variable, and the axis system and sign convention on the independent variable. The independent variable additionally has specification of how the interpolation between the data points is done, and how the data is extrapolated if the independent variable is the definition of what CL ALFA is, including unit and sign conventions. Significantly there are two sections of documentation on the data itself, the personnel that created the data and contact information for them and the references uses to create the table originally. The reference letters are actually used on the function table for each data point to represent the origin of each individual data point. Function tables usually consisting of data from several different sources. References are used to track the source of every data point. Finally, there is a section for modification of the data table after it was originally made, again letters being used to reference the data source and of course who modified the tables and when.

As discussed above, the function table representation is in two major sections, header and data section. The header is simply information about the data.

The header itself, is conceptionally divided into three sections. Although there is no distinction of these sections or particular order required (with one exception) in the header information is key word identified in the key words can be put in any order. It is certain logically to put the key words in the following order. The first section of the header defines the independent and dependent variables and gives the user information about them. Most importantly, how the function table should be interpolated and extrapolated. The second section gives the references to the data sources used in the function table. This includes any modifications made and there is a correlation between the data sources and the data points in the table through reference. The third section of the header is logically the plot definition section. This is very important for initial validation of function data. The plot section allows specific definition of a plot of the data in order to keep these plots visual consistent. This allows the user to easily plot the data once it is in their computer system and create plot with the same "look" as everyone else, so you can visually quickly look at the data and do an initial evaluation that the data in your system is the same as the same that was supplied because the plots "look the same" It is also obviously a nice documentation feature for documenting the simulation itself.

As mentioned above, key words are used to put the data in the header. The keyword is omitted and the data is not part of the header. The only time the order of the information is important is to identify which variable, be it the dependent variable or which of the independent variables, the keyword is referring to. For example, the units keyword specifies what units a dependent or independent is in. When unit is in header, the convention is the units supplied the last variable referenced, whether it was independent variable or the dependent variable. So it is logically that the user should organize the header to discuss the dependent variable, its units, sign convention, method of extrapolation/interpolation etc., give a description of it (all these are keywords) and discuss an independent variable, using the same keywords to enter the information about the independent variable, then the next independent variable etc. The same logic flows with the plot definition, plot scaling, symbols and so forth are based for the independent variable or dependent variable axis whichever is specified by a keyword.

Close examination of the examples illustrates this showing keywords for dependent variable, a units keyword is repeated for each different variable. This way the user may include or not include whatever information they desire, in fact, no header is required for the function table to execute all information required for the function table is in the function data section.

Keyword	
Function	This describes the function as used in the simulation. This is the output of the function. It's the dependent variable with the independent variable(s) in parenthesis. Ex: Function =CL(AngleOfAttack, Mach,DeltaS)
IndependentVariable	This keyword defines what independent variable is under discussion. Any subsequent occurances of Description, AxisSystem, Units, SignConvention, etc. apply to the last mentioned IndependentVariable (or the Function if IndependentVarible has not been used). Ex: IndependentVariable=AngleOfAttack
Description	The description would be a text description of what the function or variable under discussion is. The variable under discussion would be defined by the occurrence of IndependentVariable or Function keyword. Ex: Description =True angle of attack
Units	This is the engineering units of the Function or IndependentVariable under discussion Ex: Units = deg
AxisSystem	This is the axis system of the Function or IndependentVariable under discussion Ex: AxisSystem = Body axes

SignConvention	By convention, this is the positive sign convention for the Function or IndependentVariable under discussion. For example, if is a control surface, the signe convention could be trailing edge down, which would indicate plus equals trailing edge down.
	IndependentVariable= DeltaS
	SignConvention = trailing edge down

Alias	This is another equivalent name for the variable under discussion. Alias is important that it allows the user of the function table to use the table as it was supplied and simply edit the alias to indicate to the users table lookup software what the independent and dependent variables are in their own simulation. For the example shown, *CL* is the function as supplied (defined by the supplier or in the standard), the alias is *basic_lift_coefficient*. This is meant so the user's (person importing the function) software then could define the output of the function *basic_lift_coefficient* and the user would not have to change the software inside their simulation to use this function.
Symbol	The mathematical symbol for the variable under discussion
Creation Information	This is text that by convention includes the name, phone number and address of all those involved in the table creation.

Date Created	Date this table was created.

Keyword	
Creation Reference Documents	
Reference	Reference keyword has a letter in the text following the letter associated with it. The letter is used in the function table to define which reference each data point is. The text following the letter is used as the readable definition of where that data came from. There may be as many references as required.
Modification nformation	
Reference	This is the reference used to modify the table. By convention, the text included to describe the modification reference should also put the name, address, POC of the person that did the modification and the date of the modification.
Plot Control	**Indicates that the keywords that follow pertain to plotting the function under consideration**
ndependent Variable	This keyword indicates the independent variable is the variable under consideration. Any plot specifications that follow this (before changing to a dependent variable) would apply to the independent variable.
MIN	Equals the minimal engineering units. The minimal scale plotted. Default is autoscale.
MAX	The maximum value on variable under discussion to be plotted Default is autoscale.
Grid	The engineering units between the major grids plotted. For example, 10 would be 10 for a variable in degrees would mean there is a major grid every 10 degrees of the scale. Default is autogrid.
Minor Grid	The number of engineering units between the plotting of minor grids on the scale on the variable under discussion.

	Default is autogrid.
ize	Size of the plot. Two arguments for the keyword size, the first is the X size of the plot (the width), the second is the y size of the plot (the height) in inches.
ize metric	This is the same as size except the units would be in centimeters
rientation	This specifies the orientation of the second independent variable axis for a 3D plot.
imension	This specifies whether a plot is a 2D or 3D view of the function (2D view is default)
tyle	This specifies different options for symbols and interpolation and extrapolation of the plot. Valid responses are points 1-points N, specifying different symbols only to be plotted with no lines or lines 0-lines N with specifying that symbols are plotted and connected by lines. Lines 0 that specify that no symbols were plotted, just lines.

The next sections show the actual data points and has labels for the data so it's readable by the human. This intended to be simple text files, again for simplicities for data transfer and to make it human readable. It should be noted that if certain data is not available (such as conference interval for a data point) it is simply omitted. Therefore, the actual format of the text will have to blanks and the alignment of the text indicates the data point that indicates that data that is grouped together. For example: in the angle of attack of 8 degrees, the confidence interval data are related to the 8 degree angle of attach and not the 6 degree angle of attack.

Multi dimensional tables is simple extension of the one dimensional table, it should be noted that by convention, interpolation will be done in the order that the independent variables occur for the table, first the interpolation will be done on angle attack, then by mach number and then finally stabilizer position.

Another note is that in the header information the format is basically no format, the only information readable by the computer is indicated by the equal signs. And these define the types of interpolation and extrapolations and the limit of the interpolation or extrapolations by each independent variable. Therefore the computer program will simply key on equal signs in the header to obtain information required to perform the table lookups.

The actual table information is again in rigorous text format and the order of the variables is required to be correct. No additional comments are part of this standard. In other words, the table format therefore is more rigorous at this point.

The "plot" section of the header is for convenience to the user. The function data is almost universal displayed to the user in graphical format of plots. Normally the plots should be viewed by the user through the functions that interpolate the data presented. Also, the function of the table may be used to compute the independent variable output for dependent variables out side the range of the data given. For example, in the single dimension table the valid range of angel tack is, in our example, plus 180 degrees to minus 180 degrees. Therefore it is good practice to plot the function table with that range of values to help the user analyzed the result of table for the full possible range of the inputs. This will reduce problems in simulations due to independent variable exceeding the range the validated range of the particular function table. This feature will help the user to determine what will happen if that occurs.

The plot section therefore defines a standard set of plot parameters for function tables for a one dimension table example it is going to plot CL ALFA from minus 180 degrees to plus 180 degrees in increments of _ a degree. The plot increments the intervals between the data point normally should be identical if the interpolations and extrapolates are both linear. The output plot show look just like the input plot. However, if the interpolations and the extrapolations are not linear, then the effect of the interpolation and extrapolation can be seen quite readily when given a high resolution of output data points. Again, the plot section in the header is simply user settable default for displaying the data as it will be looked up in the simulation.

The standard data format has data elements that contain all of the above information. It is easy to be read by the human and the computer and has two parts:

- A header which documents the table
- A data section which contains the function data.

The header is illustrated as an example in Figure 4. In general, it is self documenting but a few further remarks are warranted. It has three parts, the table definition, the references used when the data was created and personnel involved in the creation, and the modification history of the table. The table definition includes the methods by which the table is meant to be interpolated; if data is meant to be extrapolated when the independent variable arguments exceed the limits of the table; the method to use to extrapolate; and how far to extrapolate (the limits that are to be placed on the independent variables, if any).

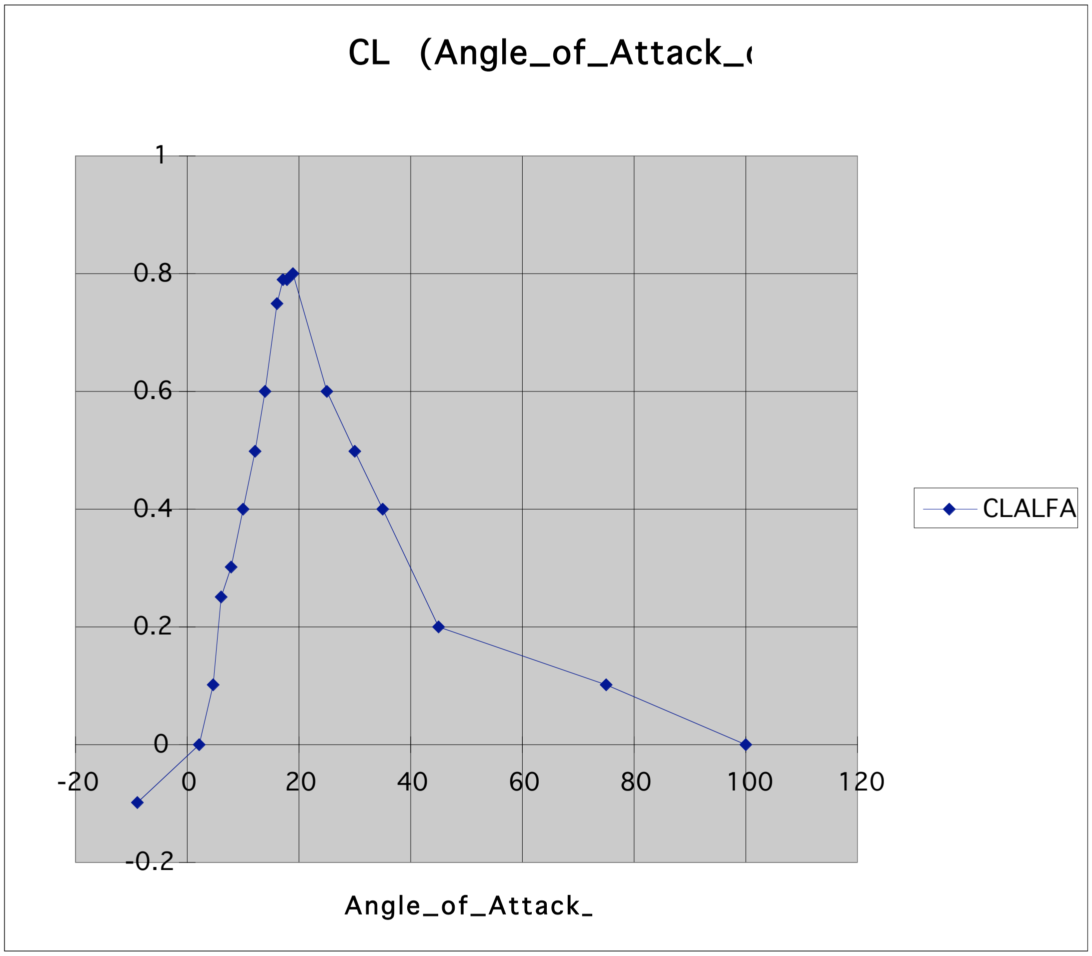
The data section of the table is shown in Figure 5. As in the header, the data section is intended to be self documenting and includes:

- The function data—the values of the dependent variable
- The breakpoint data—the values of the independent variables
- A reference for each data point
- A plus and minus 95% confidence for each data point.

The importance of the reference and confidence intervals for each data point cannot be overstated. The functional data is the heart of a model, it is what makes one non-linear model (airplane) different from another. The modeling and simulation industry in general has weak configuration management and tracking of this important information. This makes use of the reference critical.

The confidence interval (when known) is also important because it gives an intuitive and mathematical description of the estimated accuracy of the data. If the analyst has two estimates for the same data point, and these estimates have a confidence interval they can be mathematically combined and a new confidence interval established. The positive and negative values allow non-Gaussian (or non-normal) confidence intervals to be included. Entering only a positive value in the table indicates a Gaussian probability distribution for the data. Use of other probability distributions must be documented in the header.

Additional work that should be performed is to assure that these data formats are compliant with the DoD data dictionary system.



dependent_variable		CLALFA			
independent variables		Angle_of_Attack_deg wind_axis degrees nu	interpolation=cubic spline	extrapolate=	linear
Description (including Axis System)		lift coefficeint as a function of angle of attack			
Units		nd			
Sign Convention (+=)		up			
Symbol		CLo			
Creation Information:					
Created By:		Bruce Hildreth	Joe Smith	Joe Smith	
	Company	SAIC	SAIC	NAWCAD	
	Address	44417B Pecan Ct.			
		California, MD 20619			
	Phone	301-863-5077		301-342-7601	
	Fax	301-863-0299			
	E'mail	Bruce.Hildreth@cpmx.saic.com			
Date Created		12/5/1996			
Creation Reference Document(s)					
	Reference Letter	Reference Document(s) and Brief Description of the Data (include AD or Accession No. if available)			
	a	Anderson, L.C., Bunnell, J.W., "AV-8B Simulation Model Engineering Specification (Version 2.2)" Systems Control Tech. Report, Nov. 1985, Contract N00421-81-C-0289 DO 3 This is a complete model of the AV-8B derived from flight test data			
	b	"AV-8B Aerodynamics Data", MDC A1234, Rev. B, McDonnell Douglas Corp., 31 Oct. 1982, Contract N0019-79-C-0165 This updates reference a			
	c	Made up data simply to make sim. behave like an airplane at extremely hi angles of attack			

Modification Information:

Reference Letter	Reference Document(s) and Brief Description of the Data (include AD or Accession No. if available)	Modified By	Date of Mod
d	"AV-8B Aerodynamics Data", MDC A1234, Rev. D, McDonnell Douglas Corp., 15 May 1987 This updates reference a and b	Joe Smith	1/12/1998
e	"AV-8B NPE I", NATC Report 2345, Naval Air Test Center, Jan 1985 Systems Identification Estimates of CL at hi AOA from flight test data	Joe Smith	1/13/1998
f	This is my personal tweak of CL to match flight test 2-157	Jane Doe Her address Her phone Her email	4/13/1992

CLALFA(Angle_of_Attack_deg)

Angle_of_Attack_deg	- 9	2	4.5	6	8
CLALFA	-0.1	0	0.1	0.25	0.3
reference	a	d	e	e	e
confidence_interval_p					0.0020
confidence_interval_m					-0.0017
Angle_of_Attack_deg	10	12	14	16	17
CLALFA	0.4	0.5	0.6	0.75	0.79
reference	e	e	e	b	a
confidence_interval_p	0.0025	0.0022	0.0021		
confidence_interval_m	-0.0022				
Angle_of_Attack_deg	18	19	25	30	35
CLALFA	0.79	0.8	0.6	0.5	0.4
reference	b	a	a	a	b
confidence_interval_p					
confidence_interval_m					
0	45	75	100		
Angle_of_Attack_deg	0.2	0.1	0		
CLALFA	c	c	c		
reference					
confidence_interval_p					

CLALFA (Angle_of_Attack_deg, Mach_Number, delta_s)

delta_s	−5	Mach_Numbe	0.6	Angle_of_Attack_deg	−9	2	4.5	6	8	10	12	14	16	17	18	19	25	30
				CLALFA	−0.1	0	0.1	0.25	0.3	0.4	0.5	0.6	0.75	0.79	0.79	0.8	0.6	1
				reference	a	d	e	e	e	e	ee	b	a	b	a	a	a	
				confidence_interval_p		0	0			0.003		0						
				confidence_interval_m		−0				−0								
				Angle_of_Attack_deg	35	45	75	100										
				CLALFA	0.4	0.2	0.1	0										
				reference	b	c	c	c										
				confidence_interval_p														
				confidence_interval_m														
		Mach_Numbe	0.7	Angle_of_Attack_deg	−9	2	4.5	6	8	10	12	14	16	17	18	19	25	30
				CLALFA	−0.08	0.03	0.13	0.28	0.325	0.425	0.53	0.63	0.78	0.82	0.82	0.83	0.63	1
				reference	a	a	a	1	1f	d	d	a	a	a	a	a		
				confidence_interval_p		0.05	0.05	0.05										
				confidence_interval_m														
				Angle_of_Attack_deg	35	45	75	100										
				CLALFA	0.425	0.23	0.13	0.03										
				reference	b	c	c	c										
				confidence_interval_p														
				confidence_interval_m														

CLALFA (Angle_of_Attack_deg, Mach_Number, delta_s)

delta_s	−5	Mach_Numbe	0.8	Angle_of_Attack_deg	−5	2	4	6	8	10	12	14	16	17	18	19	25
				CLALFA	−0.05	0.05	0.15	0.3	0.35	0.45	0.55	0.65	0.8	0.84	0.84	0.85	0.65
				reference	a	d	a	a	a	aa	a	b	a	b	a	a	
				confidence_interval_p													
				confidence_interval_m													
				Angle_of_Attack_deg	32	47	75	90	95								
				CLALFA	0.45	0.25	0.15	0.05	0.05								
				reference	b	b	c	c	c								
				confidence_interval_p													
				confidence_interval_m													

CLALFA (Angle_of_Attack_deg, Mach_Number, delta_s)

delta_s	0	Mach_Numbe	0.61	Angle_of_Attack_deg	−9	2	4.5	6	8	10	12	14	1	17	18	19	25	30
				CLALFA	0.4	0.5	0.6	0.75	0.8	0.9	1	1.1	1.2	1.29	1.29	1.3	1.1	1
				reference	a	e	a	e	e	ee	e	b	a	b	a	a	a	
				confidence_interval_p		0.05		0.05	0.062	0.048	0.09	0.09						
				confidence_interval_m		−0		−0	−0.04	−0.03	−0.1	−0.1						
				Angle_of_Attack_deg	30	47	55	90										
				CLALFA	0.9	0.7	0.6	0.5										
				reference	b	b	c	c										
				confidence_interval_p														
				confidence_interval_m														

CLALFA (Angle_of_Attack_deg, Mach_Number, delta_s)

delta_s	0	Mach_Numbe	0.8	Angle_of_Attack_deg	−5	2	4	6	8	10	12	18	2	26	50	70	90
				CLALFA	0.5	0.6	0.7	0.85	0.9	1	1.1	1.35	1.	1	0.8	0.65	0.6
				reference	a	d	a	a	a	aa	a	b	a	b	c	c	
				confidence_interval_p													
				confidence_interval_m													

6.0 Standard Time History Data Format

1.0 Introduction

This document details a standard flight test data storage format recommended the AIAA Modeling and Simulation Data Standards for Air Vehicle Flight Dynamics Data. This standard is designed specifically to manage large amounts of simulation, flight test and analysis time histories.

2.0 General Definitions

2.1 Flight Record

The main division of flight data in the standard format is by flight records. A *flight record* is defined as a complete set of test data and related header information covering a portion of a flight. If test data are continuously recorded for the duration of a flight, they are provided as one record. If data recording devices were turned off and on during a test, the data are stored as several flight records each representing the time between turning the recording system on and off.

2.2 Time history Vector

The standard data stream stored is the time history vector. A *time history vector* is a single channel of recorded flight data along with its associated time tag. Conceptually, a time history vector is a N by 2 matrix with time points in the first column and data points in the second column. Each row contains the time and data value of a single measurement. Individual channels are stored in separate time history vectors. Since each channel is stored separately with its own time tags, there is no requirement for uniform sampling of data. Data should only be stored when sampled; there is no need to sample-and-hold or interpolate data to achieve uniform sampling. Also, in the case of a typical data drop out, neither the time nor the data value is included in the time history vector.

3.0 Standard Format

3.1 General Format

Each flight data record consists of one information or header file accompanied by the proper number of time history vector files. Figure 3.1 shows the general layout of each flight record. Each flight record is stored in a separate directory. The information and time history vector files associated with the record are placed in the corresponding directory.

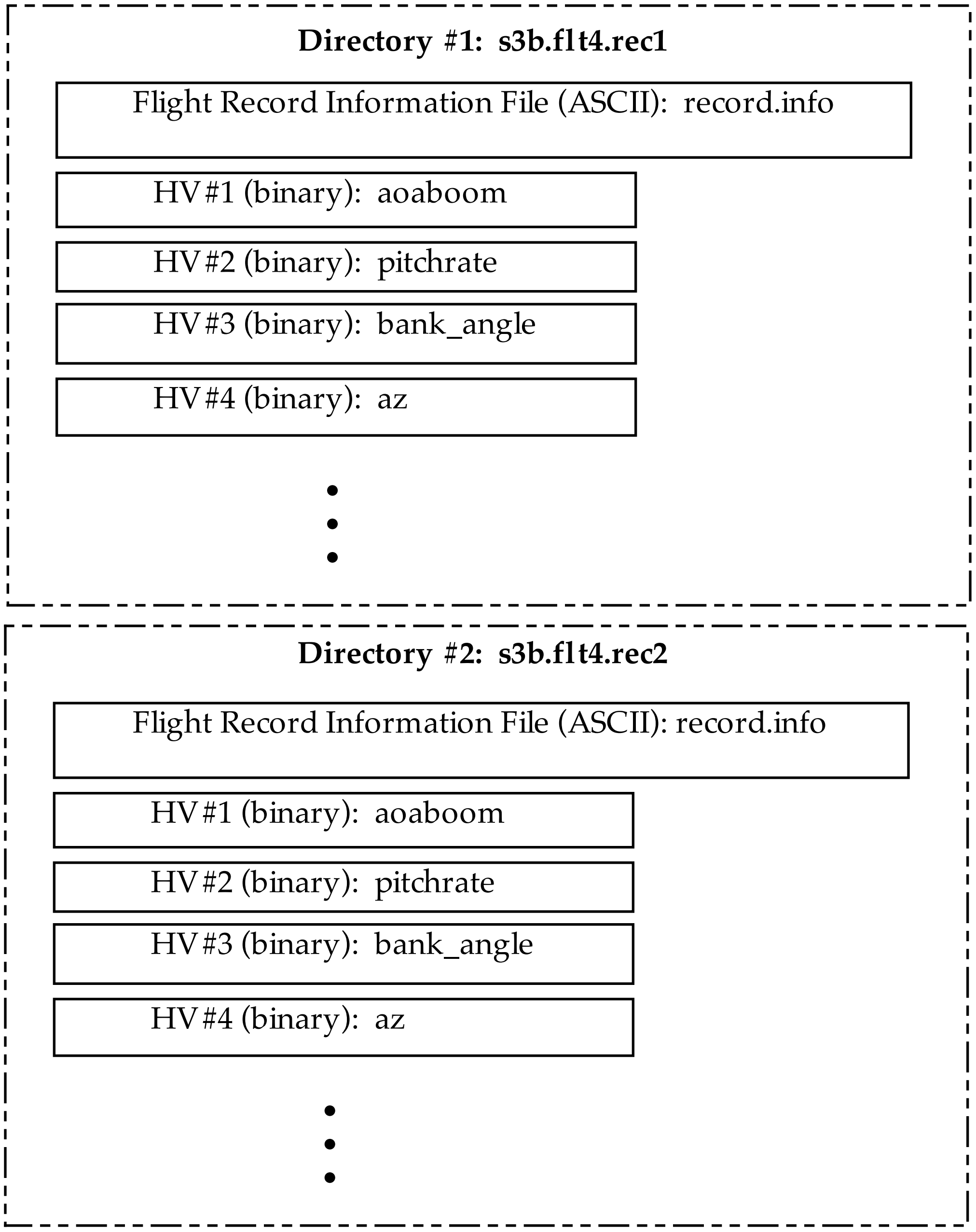

Directory #1: s3b.flt4.rec1
Flight Record Information File (ASCII): record.info
HV#1 (binary): aoaboom
HV#2 (binary): pitchrate
HV#3 (binary): bank_angle
HV#4 (binary): az
Directory #2: s3b.flt4.rec2
Flight Record Information File (ASCII): record.info
HV#1 (binary): aoaboom
HV#2 (binary): pitchrate
HV#3 (binary): bank_angle
HV#4 (binary): az

Figure 3.1 The standard record directory and file format

3.1.1 Directory Structure Naming Convention

The flight record directory follow the naming convention

```
    [project code].[flight identifier].[record identifier]
```

For example, record number 3 from the 4th flight of the S-3B flight dynamics test program may have the directory name:

```
./s3b.flt4.rec3
```

3.2 Flight Record Information File Format (Header File)

The flight record information file (often called the header file) associated with each flight record is a text file. The flight record information file has the following properties:

File Type:	Formatted (ASCII)
File Access:	Sequential
Record Length:	Variable
File Length:	1 record
File Name:	record.info
File Quantity:	1 per record

The flight record information file contains a single information record with multiple fields describing the particular flight and the associated channels of data. An example of a flight record information file is provided to illuminate the fields and items that make up the file.

3.2.1 Information Record

The information record contains 18 field types. Seventeen of these fields occur once in each record. The eighteenth field occurs once for each time history vector file.

Each field in the information record begins with a fixed key to identify the information provided. The key is separated from the information with a "==" symbol. The data up until the next line starting with a new key are associated with the previous key. New line characters not followed by information record keys are retained with no special interpretation. New line characters do no indicate the end of a field unless followed by a new key or the end of file. Blank lines are ignored. When the desired (but not required) information is not available, the space provided for it is left blank.

The 18 field types of the information record are shown in Table 3.2.1-1. Eight of the fields contain required items that must be filled in for a correct information record. For those fields that do not have required items, the key must be included, but the information item may be left blank. Table 3.2.1-1 shows the default ordering of the fields.

Table 3.2.1-1 Information Record Keys

Key	Brief Description	Required Information
AIRCRAFT TYPE	Aircraft type	Required
BUREAU NUMBER	Aircraft bureau number	Required
REFERENCE TIME	Reference time	Required
FLIGHT NUMBER	Flight number	Required

DESCRIPTION	Description of flight	
TAKEOFF TIME	Takeoff time	
CREW	Crew members	
RECORD START TIME	Record start time	Required
RECORD STOP TIME	Record stop time	Required
COMMENT	Misc. comments	
REFERENCE WEIGHT	Reference weight	
FUEL LOADING	Fuel loading	
A/C LOAD CONFIG	Aircraft loading configuration	
FIELD CONDITIONS	Field conditions	
FORECAST WINDS ALOFT	Forecast of winds aloft	
TESTED CONFIG	Tested aircraft configurations	
NUMBER OF HV	Number of time history vectors	Required
HV	Time history vector specification	Required

Elements noted as required must contain valid fields. The remaining records may be left empty.

The following is an example of the information record in the Flight Record Information File. The new-line character marking then end of a line is indicated by '\n'.

Example 3.2.1-1. Flight Record Information File

```
AIRCRAFT TYPE == S-3B\n
BUREAU NUMBER == N175356\n
REFERENCE TIME == TUE DEC 12 10:24:15 1995\n
FLIGHT NUMBER == #23A\n
DESCRIPTION == High angle-of-attack longitudinal tests\n
               and PID test block at 20K feet\n
TAKEOFF TIME == TUE DEC 12 10:39:10 1995\n
CREW == J. Doe and M. Smith\n
RECORD START TIME == 38350.370450\n
RECORD STOP TIME == 42783.54660\n
COMMENT == Heavy turbulence encountered throughout.\n
           Angle of attack exceeded 50 degrees.\n
           Left PLA signal is noisy.\n
REFERENCE WEIGHT == 36805.9 zero fuel weight.\n
FUEL LOADING == 1855 lbs left wing tank\n
                1849 lbs right wing tank\n
                1200 lbs left aux tank\n
                1200 lbs right aux tank\n
A/C LOAD CONFIG ==   nominal cg x-location 23.8 %mac\n
                     nominal cg y-location buttline +5\n
                     nominal  cg  z-location  waterline
+169\n
                     star wars inst. package installed.\n
                     Aero-1D tanks installed.\n
FIELD CONDITIONS ==  temperature 15 deg c\n
                     dew point 5 deg c\n
                     baro 29.99\n
                     7000 scat. 25000 overcast\n
                     surface wind 230 at 15 Knots\n
FORECAST WINDS ALOFT ==   at Pax River\n
                          3000 ft.  260 at 18 Kn\n
                          6000 ft.  300 at 21 Kn\n
                          9000 ft.  310 at 25 Kn\n
                          12000 ft. 300 at 25 Kn\n
                          15000 ft. 310 at 29 Kn\n
                          18000 ft. 320 at 33 Kn\n
TESTED CONFIG ==     PID test block 5 at 15000 ft\n
                     approach flaps\n
                     landing gear down\n
NUMBER OF HV == 93\n
HV == aoaboom     S  R  rad         +ANU     Noseboom is
bent\n
HV == pitchrate  S R rad/sec   +ANU    from AHRS\n
               .
               .
               .
```

3.2.1.1 Aircraft Type Field

The aircraft type field contains the aircraft type key, separator, and a single aircraft type item. The aircraft type item is up to 256 characters long. The field is terminated by a new line character followed by the next key. The aircraft type item contains a description of the aircraft type including any special features of the test aircraft. This item should contain the standard aircraft type specification including the hyphen.

In the above example, the aircraft is an S-3B.

3.2.1.2 Bureau Number Field

The bureau number field contains the bureau number key, separator, and a single bureau number item. The bureau number item is up to 256 characters long. The item is terminated a new line character followed by the next key. The bureau number item contains the bureau number of the test aircraft.

In Example 3.2.1-1, the aircraft bureau number is N175356.

3.2.1.3 Reference Time Field

The reference time field contains the reference time key, separator, and a single time and date of flight item. This item is exactly 24 characters long. The item is terminated by a new-line character followed by the next key. The reference time item contains the date and time of a reference event. The reference event is typically the time associated with turning on the recording device or resetting the recording device. Other reference events may be engine start or takeoff times. The reference time is provided as a three letter day-of-the-week code, a three letter month code, a two digit day-of-the-month number, a time of day item specifying hours, minutes, and seconds separated by ":", and a four digit year specifier. The elements are separated by single blank spaces. The time-of-day is specified in UTC (Coordinated Universal Time, or Zulu time) in 24 hour format.

In Example 3.2.1-1, the recording devices were turned on at 10:24:15.34569 on Tuesday, December 12, 1995, so the reference time field is

```
Dec 12 10:24:15 1995
```

Notice that the reference time filed has a resolution of 1 second. The record start and stop time fields contain higher resolution time measurements.

3.2.1.4 Flight Number Field

The flight number field contains the flight number key, separator, and a single flight number item. The flight number item is up to 256 characters long. The item is terminated by a new-line character followed by the next key. The flight number item contains the identification of this test flight within the flight test plan.

In Example 3.2.1-1, the flight number is Flight #23A of the test program.

3.2.1.5 Description Field

The description field contains the description key, separator, and a single description item. The description item is up to 256 characters long. The item is terminated by a new-line character followed by the next key. The description item contains a brief description of important features of the test flight conducted. Table 3.2.1.5-1 lists suggested terms and phrases for the description item. A uniform use of standard descriptions will make future searches more beneficial. Additional information regarding the exact configuration and test should also be included.

Table 3.2.1.5-1 Suggested Flight and Maneuver Descriptions

Flight Types	Maneuver Types
Power Takeoff	Wind-up Turn
Cruise	Symmetric Turn
High Angle of Attack	Steady-Heading Sideslip
Power Approach	Bank-to-Bank Roll
Ground Effect	Pull Up/Push Over
Low/High Altitude	Stall
Engine Out	Symmetric Throttle Transient
	Level Acceleration/Deceleration
	Configuration Change
	Static/Dynamic Stability
	Short Period Mode Test
	Phugoid Mode Test
	Dutch Roll Mode Test
	Roll Mode Test
	Spiral Mode Test

In Example 3.2.1-1, the description is "High angle-of-attack"

3.2.1.6 Takeoff Time Field

The takeoff time field contains the takeoff time key, separator, and a single takeoff time item. The takeoff time item is exactly 24 characters long. The item is terminated by a new-line character followed by the next key. The takeoff time item is provided in the same format as the reference time item. It contains a three letter day-of-the-week code, a three letter month code, a two digit day-of-the-month number, a time of day item specifying hours, minutes, and seconds separated by ":", and a four digit year specifier.

In Example 3.2.1-1, the takeoff time is Tuesday, December 12, 1995 at 10:39:10 UTC.

3.2.1.7 Crew Field

The crew field contains the crew key, separator, and a single crew item. The crew item is up to 256 characters long. The item is terminated by a new-line character followed by the next key. The crew item identifies the flight crew conducting the test flight.

In Example 3.2.1-1, the flight crew consists of J. Doe and M. Smith.

3.2.1.8 Start Time Field

The start time field contains the start time key, separator, and a single start time item. The start time item is up to 256 characters. The item is terminated by a new-line character followed by the next key. The start time item contains an ASCII representation of a floating point number. The floating point value is read in as a double-precision number indicating the exact start time for the flight record in seconds referenced to 00:00:00 UTC.

In Example 3.2.1-1, the start time is 38350.370450 seconds total time.

Note:

> The start and stop time fields are provided as total time in seconds including the available fractions of second. For example, if the recording devices were turned on at 10:24:15.34569 on Tuesday, December 12, 1995, the data are
>
> | | |
> |---|---|
> | Reference time field: | Dec 12 10:24:15 1995 |
> | Record start time field: | 37455.34569 |
>
> The first time tag for any time history vector is usually 0.0, but may be some other positive value indicating that the channel was not recorded at the exact instant that overall recording was started.

3.2.1.9 Stop Time Field

The stop time field contains the stop time key, separator, and a single stop time item. The stop time item is up to 256 characters. The item is terminated by a new-line character followed by the next key. The stop time item contains an ASCII representation of a floating point number. The floating point value is read in as a double-precision number indicating the last available time point in seconds referenced to 00:00:00 UTC. . See the note in section 3.2.1.9 for additional details.

In Example 3.2.1-1, the stop time is 42783.54660 seconds total time

3.2.1.10 Comment Field

The comment field contains the comment key, separator, and a single comment item. The comment item is up to 4,096 characters long. The comment item is terminated by a new-line character followed by the next key. The comment item contains a text description of additional features of the flight and the specific record.

In Example 3.2.1-1, the comments are "Heavy turbulence encountered throughout," followed by "Angle of attack exceeded 50 degrees," and "Left PLA signal is noisy."

3.2.1.11 Reference Weight Field

The reference weight field contains the reference weight key, separator, a weight item, and a description item for a total of up to 256 characters. The item is terminated by a new-line character followed by the next key. The reference weight item contains an ASCII representation of a floating point number followed by descriptive text. The floating point value is read in as a double-precision number indicating the reference aircraft weight as specified in the aircraft weight and center of gravity worksheets. Several types of reference weights may be used. Typical reference weights are aircraft basic empty weight, basic operational empty weight, or zero-fuel weight. The specific type of the reference weight is described following the reference weight value. The weight is provided in pounds.

3.2.1.12 Fuel Loading Field

The fuel loading field contains the fuel loading key, separator, and a single text item that holds a description of the fuel loading. The fuel loading item is up to 1,024 characters long. The item is terminated by a new-line character followed by the next key. This item contains text descriptions and ASCII representation of floating point numbers that specify the nominal fuel quantity in each tank and associated information such as tank location as required. Unless otherwise specified in the associated text, the fuel quantities are provided in pounds of fuel.

3.2.1.13 Aircraft Load Configuration Field

The aircraft load configuration field contains the aircraft load key, separator, and a single aircraft load configuration item. The aircraft weight configuration item is up to 1,024 characters long. The item is terminated by a new-line character followed by the next key. This item contains text descriptions and ASCII representation of floating point numbers that specify the nominal weight distribution, internal and external stores status, and other related information. Unless otherwise specified in the associated text, all weights are provided in pounds.

3.2.1.14 Airfield Conditions Field

The airfield condition field contains the airfield condition key, separator, and a single airfield condition item. The airfield condition item is up to 1,024 characters long. The item is terminated by a new-line character followed by the next key. This item contains text descriptions and ASCII representation of floating point numbers that specify the airfield atmospheric conditions and other related information. The desired atmospheric conditions include surface temperature, dew point, barometric pressure, cloud cover type and altitude, and other pertinent information usually available from the control tower. If the test data involves more than one airfield, the above information should be provided for both locations. The measurement units are provided for each specified measurement.

3.2.1.15 Winds Field

The wind field contains the winds key, separator, and a single winds aloft item. The winds aloft item is up to 1,024 characters long. The item is terminated by a new-line character followed by the next key. The winds aloft item contains text descriptions and ASCII representation of floating point numbers which specify the surface and forecast winds aloft conditions and other related information. The winds are specified using the magnitude and direction notation. The wind magnitude is provided in Knots, and the wind direction is given in degrees from the true North indicating where the wind is coming from. If wind measurements are provided in other units or conventions, additional explanation must be included.

3.2.1.16 Test Configuration Field

The test configuration field contains the test configuration key, separator, and a single test configuration item. The test configuration item is up to 1,024 characters long. The item is terminated by a new-line character followed by the next key. The test configuration item contains text descriptions and ASCII representation of floating point numbers that specify the aircraft and various subsystem configuration, as well as other related information. The applicable information includes the nominal position of lift and control surfaces such as flaps, slats, spoilers, control tabs, flight control system configuration, such as "yaw damper active" or "longitudinal augmentation off," landing gear status, and so on.

3.2.1.17 Number of Time history Vectors Field

The number of time history vectors field contains the number of time history vectors key, separator, and a single number of time history vectors item. The number of time history vectors item is up to 256 characters. The item is terminated by a new-line character followed by the next key. The item contains an ASCII representation of an integer. The valid range is from 0 to UINT_MAX (as defined in ANSI C and POSIX.1). The integer is the number of time history vectors in the current flight record. The programs that input this data can expect to find this number of time history specification fields in the current information record and this number of time history vector files in the current directory.

In Example 3.2.1-1, there are 97 time history vectors in the current flight record.

3.2.1.18 Time history Vector Specification Field

Each time history vector specification field contains the following six items in order:

name item (required)
precision item (required)
ordering item (required)
units item (required)
sign convention item (required)
comment item

Items noted as required must contain valid entries. The remaining items may be left blank. The items are separated by at least one blank space. The item is terminated by a new-line character followed by the next key.

The time history vector specification field is repeated the number of times indicated in the number of time history vector field (3.2.1.17). There is no required ordering of the fields.

In Example 3.2.1-1, an angle of attack and a pitch rate time history vectors are defined.

3.2.1.18.1 Name Item

The name item is up to 20 characters long. The item is terminated by a blank space. The name item contains the name of a data channel in each flight record.

In Example 3.2.1-1, the channel names are "aoaboom" and "pitchrate."

3.2.1.18.2 Precision Item

The precision item is used to specify the 'unit length' of the binary test data. The precision item consists of a single character which can be assigned a "S" for single precision format (32 bits/ 4 bytes) or a "D" for double precision format (64 bits/ 8 bytes) data. The item is terminated by a blank space.

In Example 3.2.1-1, the time history data are double precision binary numbers.

3.2.1.18.3 Ordering Item

The ordering item is used to specify the sequential ordering of the binary test data. The ordering item consists of a single character which can be assigned a "R" for row-wise ordering or a "C" for column-wise ordering. The item is terminated by a blank space. Row-wise ordering means that sequential read statements retrieve data in the order (time1, data1, time2, data2, ..., timeN, dataN). Column-wise ordering means that sequential read statements retrieve data in the order (time1, ..., timeN, data1, ..., dataN). For the example set

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.2 & 1.5 \\ 0.3 & 1.7 \\ 0.5 & 2.0 \end{bmatrix}$$

row-wise ordering would be (0.0, 1.0, 0.2, 1.5, 0.3, 1.7, 0.5, 2.0), while column-wise ordering would be (0.0, 0.2, 0.3, 0.5, 1.0, 1.5, 1.7, 2.0).

In Example 3.2.1-1, the time history data is stored in row-wise ordering.

3.2.1.18.4 Units Item

The units item is up to 20 characters long. The item is terminated by a blank space. The units item contains a text description of the unit of measure used by the data channel. When possible, the units item is filled with the abbreviations from Table 3.2.1.18.4-1. If necessary, other, non-standard, units may be used.

Table 3.2.1.18.4-1 Abbreviations for Units of Measure

Time	
hours	h
min	min
seconds	s
Length	
feet	ft
meter	m
nautical miles	nmi
statute miles	smi
kilometers	km
Force	
pound force	lbf
Newton	N
kilogram force	kgf
Mass	
gram	g
kilogram	kg
pound mass	lbm
slug	slug
Plane Angle	
degrees (angular)	deg
radians	rad
revolution	rev
Temperature	
degrees Rankine	R
degrees Centigrade	C
degrees Kelvin	K
Power, energy, work, heat	
British thermal unit	Btu
erg	erg
calorie	cal
joule	J
horsepower	hp
Electrical	
volt direct current	VDC
volt alternating current	VAC
ampere	A
ohm	ohm

In Example 3.2.1-1, the boom angle of attack is in radians and the AHRS pitch rate is given in radians per seconds.

3.2.1.18.5 Sign Convention Item

The sign convention item is up to 20 characters. The item is terminated by a blank space. The sign convention item contains a text description of the sign convention used by the data channel. When possible, the sign convention item is filled with the abbreviations from Table 3.2.18.5-1. If necessary, other, non-standard, sign conventions may be used.

Table 3.2.1.18.5-1 Abbreviations for Sign Conventions

Aircraft Nose Up	+ANU
Aircraft Nose Down	+AND
Aircraft Nose Right	+ANR
Aircraft Nose Left	+ANL
Right Wing Down	+RWD
Left Wing Down	+LWD
Up	+UP
Down	+DOWN
Left	+LEFT
Right	+RIGHT
Forward	+FWD
Aft	+AFT
Clockwise	+CW
Counter-clockwise	+CCW
North	+N
South	+S
East	+E
West	+W
Trailing Edge Up	+TEU
Trailing Edge Down	+TED
Trailing Edge Left	+TEL
Trailing Edge Right	+TER

In Example 3.2.1-1, the sign convention is given as positive for aircraft nose up.

3.2.1.18.6 Comment Item

The comment item is up to 256 characters long. The item is terminated by a new-line character followed by the next key. The comment item contains a text description of unique features of the channel.

3.3 Time history Data File Format

The time history data file contains the time and data values for a single channel. Each channel of data is stored in a separate time history data file as indicated in Fig. 3.2.1-2. The contents of the time history data file are described by the time history vector information field (Paragraph 3.2.18).

FileType: Binary (ANSI/IEEE Std 754-1985) (with optional compression)

File Access:	Sequential
Record Length:	Variable
File Length:	1 record
File Name:	from the Name item in record.info file (3.2.18.1)
File Quantity:	1 per time history vector

A binary file is created by storing data to a file in ANSI/IEEE Std 754-1985 binary format for single- or double-precision numbers.

There is one time history data record per file. The data are stored as entirely single-precision or entirely double-precision values as specified in the precision item of the record information file (3.2.1.18.2).

For single-precision values, the record length in bytes (before compression) is

2 □ (Number of data points) □ (4 bytes/point)

For double-precision values, the record length in bytes (before compression) is

2 □ (Number of data points) □ (8 bytes/point)

Optionally, the final binary file may be compressed using the gzip (GNU zip) loss-less compression utility (licensed under the GNU General Public License) to save space. If the data file is compressed, an extension of ".gz" is added to its specified name. gzip compression typically offers approximately 60-70% reduction in file size for the binary data stored in a time history vector.

3.3.1 Time history Data Record

The time history data record contains a continuous stream of 8 bit binary floating point values. The values may be stored as either 4-byte single-precision values or 8-byte double-precision values as specified in the precision item of the information record (Paragraph 3.2.1.18.2). The individual values are stored using bit 0 as the least significant bit and byte 0 as the most significant byte. Thus, a 32 bit (4 byte) single precision number (data or time value n) is stored as (starting from file offset N)

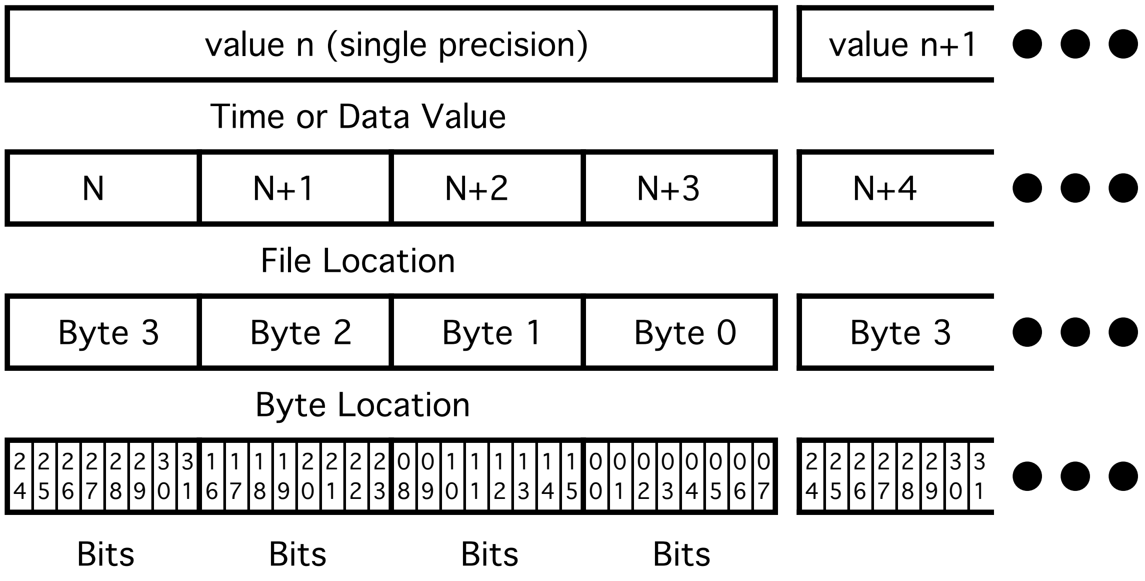


Figure 3.3.1-1 Floating Point File Storage Layout

The number of single-precision or double-precision numbers stored in each record is determined by the specified precision in the information file and the file length.

The sequential ordering of time and data values within the time history data record is specified by the ordering item in the time history vector specification field (3.2.1.18.3). For row-wise ordering, the data are stored as

```
(time1, data1, time2, data2, ..., timeN, dataN)
```

For column-wise ordering, the data are stored as

```
(time1, ..., timeN, data1, ..., dataN)
```

This can be viewed as either reading across rows (row-wise) or down columns (column-wise) of the matrix containing the time history data in the following format:

$$\begin{bmatrix} \text{time}_1 & \text{data}_1 \\ \text{time}_2 & \text{data}_2 \\ \vdots & \vdots \\ \text{time}_N & \text{data}_N \end{bmatrix}$$

7.0 STANDARD SIMULATION MODEL VERIFICATION TESTS AND PROCEDURES

7.1 Background / Philosophy

This portion of the standard is to verify that the simulation model, when transferred from location A to location B, is implemented correctly. We will verify to mean that the software model runs in the manner it was intended to. No overt bugs exist that cause the model computations to incorrectly compute response.

This is not intended to validate that the particular model flies like the airplane. Only that it was installed properly at your own facility. Obviously, verification is the first step of validation.

Therefore, the philosophy used for verification in this document is to define a minimum set of specific standard maneuvers that would be easy to reproduce at each facility. That when the output from one facility is compared to the output from another, it would be easy to determine whether the receiving facility has installed the model correctly. Further test validation consists of the following information:

- Initialization information
- Control inputs
- Data output

The standard verification procedure simply becomes warning a set of specified maneuvers in the receiving facility simulator and comparing the outputs sent by the sending facility with the specified outputs at the receiving facility. Those familiar with simulation will realize that the most technical portion of performing verification such as this is the fact that different computers have different math functions and different numerical accuracy such that after ten seconds of integration over a simulation run a model installed exactly the same on two different computer architectures will have small errors in the least significant portion of the ________ point number. Therefore, it is very difficult to know if these errors are due to a small bug, or simply due to architectural differences in the computer system.

These differences after a simulation run are further aggravated by the fact that almost every simulation architecture initializes its model differently. Oftentimes a optimized ________ control surfaces to put the simulation model into approximate equilibrium. Flight differences in initial condition when integrated over time, may mean a significant difference in the end-state of the vehicle.

Therefore, the initialization of the model becomes very important to the verification of the model since it introduces the largest discrepancy between the simulation run at the sending site and the same model run at the receiving site. Initialization algorithms are not intended to be part of the model. Specification of the initialization state *is* intended to be part of the model and part of the standard.

7.2 Conventions Used

Proposed Convention

This paragraph will discuss the convention and philosophy used for selecting the tests and procedures to be performed. The purpose of this is so if other tests and procedures are added to the standard they will be consistent with these tests and procedures.

Discarded Tests and Procedures and Reasons

Summary
It is strongly recommended that this convention be followed for all tests and procedures. If it is not a satisfactory convention , the convention should be modified and all the defined tests and procedures. modified according to the new convention.

7.3 Proposed Tests and Procedures

___________________________ discuss ____________________ there are three major portions of this standard:

1. Specification of the initialization
2. Running the model as a function of time with a specified input maneuver
3. Comparison of model data to verify correct operation.

The initialization will be discussed in detail below because of its importance in the verification process. If the simulation model is not initialized precisely, then small differences in initialization integrated over time make a large difference in the response of the vehicle.

Running the simulation model with specified input is a fairly simple task. For all verification maneuvers the control inputs are purposely kept simple so they can be exactly duplicated at different facilities.

The final task for each maneuver is comparing the results to the baseline results. There is no known standard for estimating the accuracy between two computers.

7.4 References

Instructions from BH-
Use Section 3.2 and 3.3. in their entirety from the Omeed Simulation Specification version 0.4 (Omeed has it and just paste it there). After that insert the next item which would be the section titled "Maneuvers to be Simulated".

3.2 Initialization and Trimming

The objective of model initialization is to provide initial condition values for all the simulation states so that model propagation can proceed accurately as compared with flight measured data. Since some differences between the model and actual aircraft are expected, for some applications, it is necessary to provide several "degrees of freedom" to correctly initialize the model response.

3.2.1 Setting Primary Trajectory Parameters

For all cases, the states that directly affect the total energy of the trajectory being simulated are set and fixed. These states are the inertial altitude, total inertial and air-relative velocities, ground track (horizontal flight path angle), and vertical flight path angle. If a round rotating or an oblate spheroid rotating earth model is used, the aircraft longitude and latitude shall also be set. In addition all the related parameters shall be set to their measured values. The related parameters include, but are not limited to the aircraft mass and inertial properties, atmospheric conditions, relative winds, and so on. The following command lines may be used:

```
simproc ic variable vt         ic_value % total inertial
velocity
simproc ic variable gamv       ic_value % flight path angle
simproc ic variable gamh       ic_value % ground track
simproc ic variable xcgloc     ic_value % aircraft x-
location
simproc ic variable ycgloc     ic_value % aircraft y-
location
simproc ic variable zcgloc     ic_value % aircraft z-
location
simproc ic variable xmass      ic_value % aircraft mass
simproc ic variable xixx       ic_value % aircraft inertia
Ixx
simproc ic variable xiyy       ic_value % aircraft inertia
Iyy
simproc ic variable xizz       ic_value % aircraft inertia
Izz
```

```
simproc ic variable xixy       ic_value  % aircraft inertia Ixy
simproc ic variable xixz       ic_value  % aircraft inertia Ixz
simproc ic variable xiyz       ic_value  % aircraft inertia Iyz
simproc ic variable vnw        ic_value  % north wind velocity
simproc ic variable vew        ic_value  % east wind velocity
simproc ic variable vdw        ic_value  % down wind velocity
simproc ic variable pamb       ic_value  % ambient pressure
simproc ic variable tamb       ic_value  % ambient temperature
simproc ic variable rho        ic_value  % ambient density
simproc ic init
```

The flight dynamics states and state derivatives are also set to their measured values. This ensures that the model propagation starts at the proper state. The following flight dynamic states are initialized to their flight measured values.

```
simproc ic variable pb         ic_value  % aircraft roll rate
simproc ic variable qb         ic_value  % aircraft pitch rate
simproc ic variable rb         ic_value  % aircraft yaw rate
simproc ic variable phi        ic_value  % aircraft bank angle
```

3.2.2 Parameters Adjusted During Initialization

Since the model may differ from the aircraft in its aerodynamics and propulsion system, it is necessary to provide some degrees of freedom in state initialization to account for such differences. For example, the difference in generated lift (or total weight) between the aircraft and the simulation can be accounted for by adjusting the initial angle of attack, or equivalently, the pitch angle. Biasing the flight control surface positions from those measured in flight can correct for moment differences. For this discussion, we are including the propulsion system input as a control parameter as well. The following states and control positions are freed to allow for such an adjustment.

```
simproc trim control theta     min_value max_value % pitch angle
simproc trim control psi       min_value max_value % heading
simproc trim control control   min_value max_value % control pos.
```

It should be noted that pitch and yaw states used here effectively control the aircraft angle of attack and angle of sideslip. Experience has shown that this arrangement provides the best results for model propagation. For certain applications, it is possible to freeze the heading state and vary the bank angle as necessary to achieve similar results with the sideslip angle. For conventional aircraft applications, the controls will include the throttle position(s), one longitudinal control surface, one lateral control surface and one directional control surface.

If the aircraft is equipped with more than one independent control surface in each axis, the user must select one of the available surfaces to perform the model initialization with. For example, if the aircraft is equipped with a movable horizontal stabilizer and an elevator, and if they are not geard together, the user has to select one of them. When multiple control surfaces are actuated simultaneously using a known control law or are gear together, we can treat them as a single equivalent control.

In the case of conventional rotorcraft, the main rotor and tail rotor controls including the collective should be biased as needed. If the helicopter is equipped with a control system to washout the angle between the rotor and the fuselage, it may be necessary to vary fuselage pitch and bank angle as well.

3.2.3 Initialization Measures of Performance

The optimization algorithm attempts to minimize a set of residuals by changing various trim control variables. For typical rigid-body airframe configuration, such an initialization is acheived by driving the three linear and three angular accelerations close to some nominal conditions. In addition, in some cases, one or more state derivatices in the propulsion system model may also be required to complete the trimming process. The primary state derivatives for a six-degree-of-freedorm rigid-body dynamic are provided in the following equations.

$$\begin{bmatrix} \dot{V}_{north} \\ \dot{V}_{east} \\ \dot{V}_{down} \end{bmatrix} = \frac{1}{mass_{A/C}} \begin{bmatrix} F_{north} \\ F_{east} \\ F_{down} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

$$\begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} = [IN] \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} h_{x_{rotor}} \\ h_{y_{rotor}} \\ h_{z_{rotor}} \end{bmatrix}$$

where the inertial matrix IN is defined as;

$$[IN] = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & Izz \end{bmatrix}$$

The angular acceleration terms are written as

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = [IN]^{-1} \bullet \left(\begin{bmatrix}T_l\\ T_m\\ Tn\end{bmatrix} - \begin{bmatrix}0 & -r & q\\ r & 0 & -p\\ -q & p & 0\end{bmatrix} \begin{bmatrix}h_x\\ h_y\\ h_z\end{bmatrix} \right)$$

In many applications it is more convenient to specify body-axis linear acceleration instead of earth referenced accelerations used in the standard equations of motion. It is therefore necessary for the equations of motion module to compute body axis linear acceleration as well. The relation between earth referenced and body axis linear accelerations are given in the following equations. The matrix T_{LL2B} performs the necessary rotation between the locally level earth axis system and the body fixed axis system.

$$\begin{bmatrix}\dot{u}_b\\ \dot{v}_b\\ \dot{w}_b\end{bmatrix} = \left[T_{LL2B}\right] \begin{bmatrix}\dot{V}_{north} - \dot{V}_{wind_{north}}\\ \dot{V}_{east} - \dot{V}_{wind_{east}}\\ \dot{V}_{down} - \dot{V}_{wind_{down}}\end{bmatrix} + \begin{bmatrix}0 & r & -q\\ -r & 0 & p\\ q & -p & 0\end{bmatrix} \begin{bmatrix}u_b\\ v_b\\ w_b\end{bmatrix}$$

Methods of deriving the appropriate initial values for the above linear and angular accelerations are provided in section 3.2.6 of this document. Examples of residual specification is provided below.

```
simproc trim resid  ubd       nom_value      %  x-velocity
dot
simproc trim resid  vbd       nom_value      %  y- velocity
dot
simproc trim resid  wbd       nom_value      %  z- velocity
dot
simproc trim resid  pbd       nom_value      % roll-accel
simproc trim resid  qbd       nom_value      % pitch-accel
simproc trim resid  rbd       nom_value      % yaw-accel
```

The process of optimal initialization described above is essentially the same as solving for N unknowns in a system of N equations. The six residuals shown above are usually sufficient to solve a traditional rigid-body aircraft configuration. Adjusting the elevator, ailerons, and rudder primarily affects the pitch, roll, and yaw accelerations. Adjustments in the pitch angle and heading effectively varies the angle of attack and sideslip which in turn affect y-axis and z-axis acceleration among others. Finally, throttle or power lever adjustments mostly impact the x-acceleration. In practice, each control can influecnce all the residuals, and therefore, an optimizer is used to quickly solve this system of simultaneous equations. A more theoretical explanation for this process is provided in the following paragraphs.

3.2.4 Overview of Initialization Process

Consider the general form of the model which is propagated in the simulation process.

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{t}), \quad \mathbf{x}(t_0) = \mathbf{x}_{\varnothing}$$
$$\mathbf{y} = h(\mathbf{x}, \mathbf{u})$$

where

$$\mathbf{x} = [\, x_1 \; x_2 \; . \; . \; . \; . \; . \; x_N \,]^T$$
$$\mathbf{u} = [\, u_1 \; u_2 \; . \; . \; . \; . \; . \; u_M \,]^T$$

In order to define a unique inital state, $x_{\varnothing}$ we have to assume that there is almost no control activity for a short time, Δt, prior to the initial time. Next, the state vector is divided into two sections. The first set of states represent those with slow eigenvalues which means they do not reach steady-state condition within Δt. The second set of states have fast roots, and we can assume that state derivatives associated with those states are zero at the initial time.

$$\mathbf{u}(t_0 - \Delta t \rightarrow t_0) \approx \text{constant}$$
$$\mathbf{x} = [\, \mathbf{x}_I \; \mathbf{x}_{II} \,]^T$$

$$\dot{\mathbf{x}}_{I_{\varnothing}} = \dot{\mathbf{x}}_{\text{nominal}} = \text{user specified nominal values}$$
$$\dot{\mathbf{x}}_{II_{\varnothing}} = 0.0$$

and the initial conditions can be found by solving the unknowns in the following equation.

$$\begin{bmatrix} \dot{\mathbf{x}}_{\text{nominal}} \\ 0.0 \end{bmatrix} = f(\mathbf{x}_{\varnothing}, \mathbf{u}_{\varnothing})$$

Note that for an aircraft with static stability, as $\Delta t \rightarrow \infty$ the nominal value for all the state derivatives approach zero. This is a classical steady-state trim condition. In general, however, we assume that the aircraft has non-zero rates and some non-zero acceleration which can be purely attributed to dynamic response and not the instantaneous motion of the controls. The critical task is to determine an appropriate Δt which provides a small and measurable number of non-zero state derivatives for a given simulation model.

Aside from rigid-body dynamic states, the major contributors to the simulation state vector are the flight controls and propulsion system model. It is usually not difficult to locate steady-state conditions for the propulsion system. The time constant for the propulsion states is typically in the order of several seconds and the propulsion controls such as the throttle angles are not usually in transition. The flight control system, on the other hand, may contain many states with time constants similar to that of the aircraft dynamics. A common method to simplify the initialization problem is to use the control surface positions as inputs to the flight dynamics portion of the model. The optimization algorith will be used to determine a set of ideal surface positions which provide the required state derivatives. A separate optimization is then performed to determine the nominal initial inputs and states of the control system such that the desired surface positions are output.

An essential part of this process is to identify and reduce or disable the dynamic states of the system. Figure 3.2-1 shows the general schematic of the dynamic model discussed above. If a given state falls under the first category discussed above, with slow and dominant eigenvalues, it has to be directly included in the trimming process. This means that the state and its derivative should be passed outside the simulation code and provisions shall be made to bypass the integration of the state and override its value with the solution found by the trim optimizer. Figure 3.2-2 provides the general configuration of the dynamic model during trimming.

Figure 3.2-1 Dynamic Model Configuration

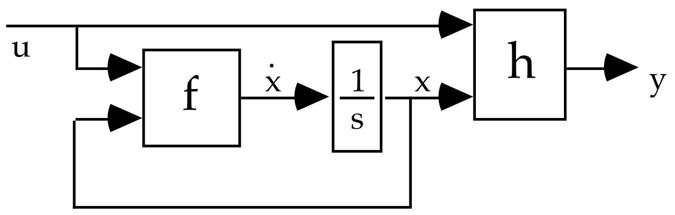
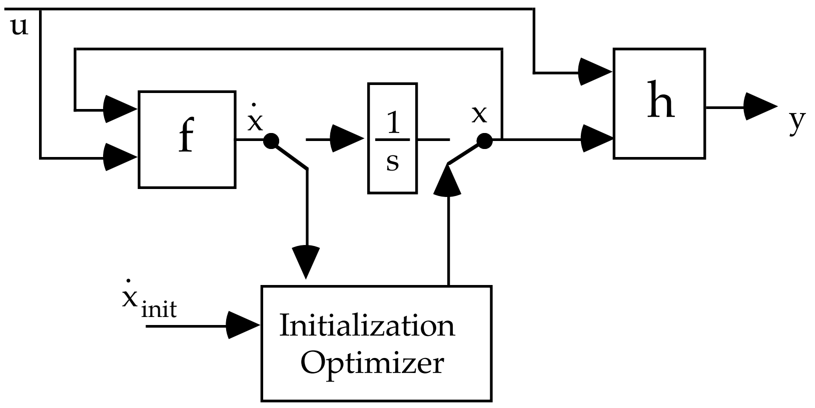


Figure 3.2-2 Dynamic Model Configuration During Trimming Operation

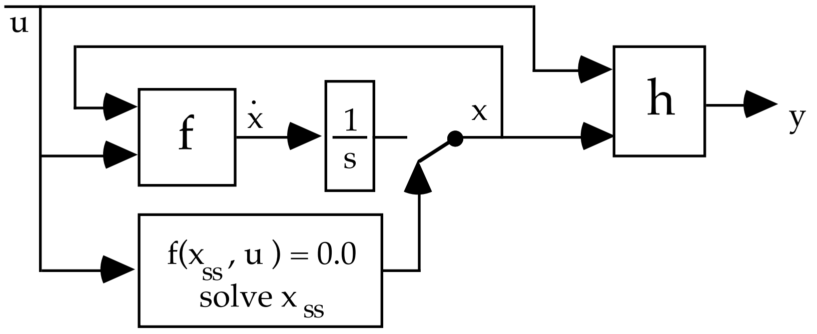


Those dynamic state with known steady-state solution and with fast enough eigenvalues can be removed from the trim process. This is accomplished by replacing the state integration with a code to compute the steady-state value of the state during trim process. It should be noted that the `IMODE` flag is set to -1 during the initialization process. Figure 3.2-3 provides an example of this implementation. This type of dynamic states are usually encountered when initializing propulsion or flight control system models. For example, a simple first order lag filter shall be modified so that during initialization, its state is set to its steady state value which is related to its input. As an example, consider a transfer function, $\frac{y}{u} = \frac{K}{\tau s + 1}$, the code required to represent this dynamics state and initialize it can be written as,

```
if (imode == -1)
     x = u;
else
     x = x + (dt/tau)(u - x);
y = K * x
```

Similarly, a first order high-pass filter can be initialized by setting the value of its state variable to zero. Carefull attention shall be made to represent all the dynamic states of a system by the appropriate method as shown in fugures 3.2-2 and 3.2-3.

Figure 3.2-3 Initialization of Dynamic Model With Known Steady-State Solution



3.2.5 Setup of Equations of Motion for Initialization

Although a standard set of equations govern the motion of a six-degree-of-freedom rigid-body flight dynamics motion, calculation of the initial condition vector $\mathbf{x}_{\o}$ requires several different modes to best fit the typical requirements of model initialization. For example, it is more convenient to specify Euler angle rates of change instead of body axis angular rates when placing the aircraft in a coordinated turn initial condition. It is therefore necessary to provide a translator to compute the initial value of the states from the user specified alternate flight condition parameters.

For practical reasons it is assumed that the baseline equations of motion adhere strictly to the standards set by ANSI/AIAA R-004-1992, "Recommended Practice: Atmospheric and Space Flight Vehicle Coordinate Systems." The states of the system are defined in table 3.2.5-1. The initial condition translator converts the user specified initial flight conditions to be used by this set of state equations.

Table 3.2.5-1 Flight Dynamics States

Symbol	Description
p_b	body axis roll rate
q_b	body axis pitch rate
r_b	body axis yaw rate
e_0	quaternion
e_1	quaternion
e_2	quaternion
e_3	quaternion
V_N	inertial velocity along north
V_E	inertial velocity along east
V_D	inertial down velocity
X	x-axis position of aircraft
Y	y-axis position of aircraft
Z	z-axis position of aircraft

<u>Angular Rate Specification</u>

Body axis angular rates are usually recorded during flight test programs. For initialization of the simulation model, the recorded angular rates at the desired initial time must be deposited in the simulation. When initializing to a flight measured state, it is important to deposit all three measured angular rates in the model.

When measurements of the angular rates are not available, it is more convenient to specify the desired Euler rates instead. For example, the model may be placed in a coordinated turn by specifying zero $\dot{\phi}$ and $\dot{\theta}$ and a constant rate of turn $\dot{\psi}$. In the case the Euler angle rates are specified instead of body axis angular rates, the initial condition translator performs the necessary calculations to provide the angular rates.

$$p = \dot{\phi} - \dot{\psi}\sin\theta + \Delta p$$
$$q = \dot{\theta}\cos\phi + \dot{\psi}\sin\phi\cos\theta + \Delta q$$
$$r = \dot{\psi}\cos\theta\cos\phi - \dot{\theta}\sin\phi + \Delta r$$

where Δp, Δq, and Δr terms specify instantaneous perturbation values such as turbulence terms. These terms are typically set to zero for normal operation.

<u>Velocity Specification</u>

The flight dynamics model of the aircraft uses inertial velocities in North, East, and Down directions as primary states. For convenience during initialization, total air-relative velocity, Mach number, or related parameters may be specified. The translator will perform the following calculations based on the information provided by the user.

If the aircraft Mach number is specified, the total air-relative velocity will be calculates as:

$$V_{air\square relative} = c \cdot M_{user\square specified}$$

where c is the local speed of sound. If equivalent airspeed in Knots is specified, the total air-relative velocity (true airspeed) is computed using the following relation.

$$V_{air\square relative} = \frac{1.689 \ V_{equivalent}}{\sqrt{\square}}$$

where □ is the density ratio at altitude and the factor 1.689 is a conversion constant from Knots to feet per seconds. In either case, the total inertial velocity of the aircraft is calculated by solving the following relations.

$$\begin{aligned} V^2_{air\square relative} &= V^2_{inertial} + V^2_{wind_{total}} \ \square \ 2V_{inertial} \, ^* \\ &\quad \left(\square V_{wind_{north}} \cos\square\cos\square \ \square \ V_{wind_{east}} \cos\square\sin\square + V_{wind_{down}} \sin\square \right) \\ V^2_{wind_{total}} &= V^2_{wind_{north}} + V^2_{wind_{east}} + V^2_{wind_{down}} \end{aligned}$$

The north, east, and down components of the total inertial velocity are calculated using the vertical flight path angle, □, and ground track, □.

$$\begin{aligned} V_{north} &= V_{inertial} \cos\square\cos\square \\ V_{east} &= V_{inertial} \cos\square\sin\square \\ V_{down} &= \square V_{inertial} \sin\square \end{aligned}$$

<u>Specifying Local Average Winds</u>

The standard simulation equations of motion require instantaneous wind components in north, east, and down directions. The details of the wind calculations is specific to various simulation models and are excluded here. For initialization purposes, the average wind vector must be defined. Typically, the horizontal wind vector is specified as a magnitude and direction from true north. The wind magnitude is specified in Knots. The direction is specified in degrees, with true north as 360°. It should be noted that the wind direction specifies where the wind is coming from. The vertical wind, if any, is specified separately. The translator performs the following calculations.

$\square$ = wind direction from true north

$$\begin{aligned} V_{wind_{north}} &= V_{wind_{total}} \cos\square \\ V_{wind_{east}} &= V_{wind_{total}} \sin\square \end{aligned}$$

Specifying Atmospheric Parameters

The simulation model shall include an atmospheric model capable of generating standard and non-standard conditions for use by the propulsion and aerodynamic models. The atmospheric conditions may be expressed in different ways depending on the available data. When initializing the model to a flight condition measured during flight test program, it is most convenient to input the local (at flight altitude) atmospheric parameters. For this case, the user shall specify the pressure altitude, local temperature and a representative sea-level barometric pressure. The translator will compute the following atmospheric calculations.

inputs:

h_p	Pressure altitude in feet
T	Local ambient temperature in degrees Kelvin
P_{SL}	Sea-level barometric pressure in in-Hg

output equations:

$$P = 2116.2 \times \left(1.0 - 6.87563E-6 \times h_p\right)^{5.2561}$$

$$\delta = \frac{P}{P_{SL}}$$

$$\theta = \frac{T}{T_{SL}} = \delta^{0.1902551321}$$

$$T_{SL} = \frac{T}{\theta} - 273.15 \qquad \text{in degrees celcius}$$

$$\rho = \frac{P}{1716.0 \times 1.8 \times T} \qquad \text{in slugs / ft}^3$$

$$c = \sqrt{1.4 \times 1716.0 \times 1.8 \times T} \qquad \text{in ft / sec}$$

Assuming a temperature lapse rate of -0.001981213 degrees Kelvin per foot of altitude, the corrected altitude is calculates as

$$h_{corr} = \frac{T_{SL}(^\circ K) - T}{0.001981213}$$

3.2.6 Derivation of Initial States from Flight Test Data

The discussion in this section centers around methods of determining the proper initial conditions from recorded flight test data. In particular, the signals that are subject to higher relative signal to noise ratio like the acceleration measurements will be addressed. It Is assumed that the flight test data under consideration has already been calibrated and the kinematics signals have been corrected using a kinematics consistency checking and navigation calibration tool such as NAVIDNT in IDEAS.

Such a calibration will ensure that some critical signals such as the angular rates and linear accelerations are free of scale factor or bias errors. We further assume that the calibrated signal contains a noise component with a mean of zero and a known or measurable standard deviation. If the noise magnitude is relatively high, it may be necessary to filter the calibrated data channel at the appropriate frequency. The initial state value is then extracted from the filtered data at the correct initial time. Special attention shall be made to the phase distortion characteristics of the filter being used. It is recommended that one of IDEAS filtering schemes such as RFILTER be used for this process.

<u>Angular Accelerations</u>

Measurements of angular accelerations are usually not available. It is therefore necessary to numerically differentiate the angular rate signals, a process which is very sensitive to measurement noise. A smoothing and differentiating filter like RFILTER in IDEAS data pre-processing tool box can perform this task. However, the user may have to experiment with the selected cutoff frequency and other parameters for best results.

3.3 Model Propagation Utilities

3.3.1 Maneuver Generator

The maneuver generator utility provides a means of exciting the model inputs during a simulation propagation. This utility is designed to be simple and very flexible in operation. It does, however, require the user to be intimately familiar with the operation of the simulation model.

The maneuver generator is composed of *n* separate channels, where *n* is defined by the user. Each generator channel has the general form shown in figure 3.3.1-1.

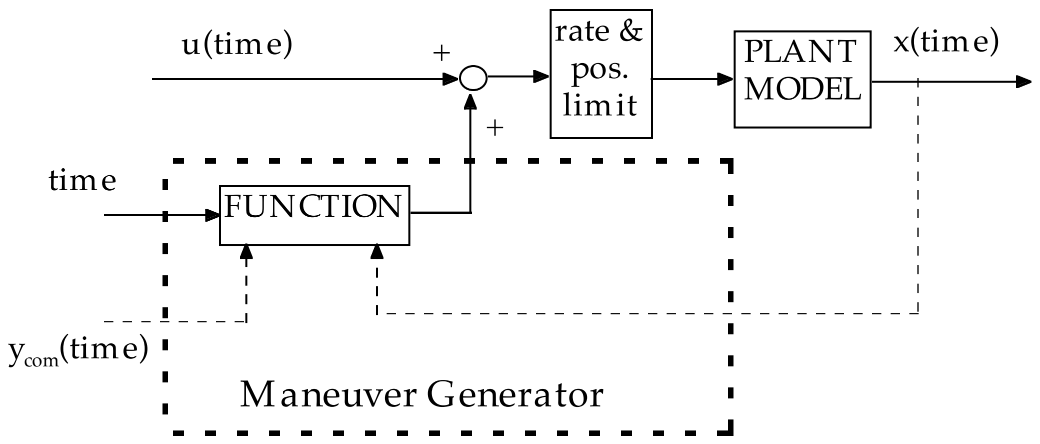


Figure 3.3.1-1 General Configuration of Maneuver Generator

the function block shown in figure 3.3.1-1 can be one of the functions described in section 2.3.2 of this document. The maneuver generator architecture is designed to accomodate future implementation of generic autopilots requiring state feedback from the model to track a desired commanded parameter.

3.4 Validation Maneuvers