BSR/ANSI/AIAA
S-XXX-200X

American National Standard

Flight Dynamics Model Exchange Standard, DAVE-ML

Sponsored by

American Institute of Aeronautics and Astronautics

Approved XX Month 200X

American National Standards Institute

Abstract

This is 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.

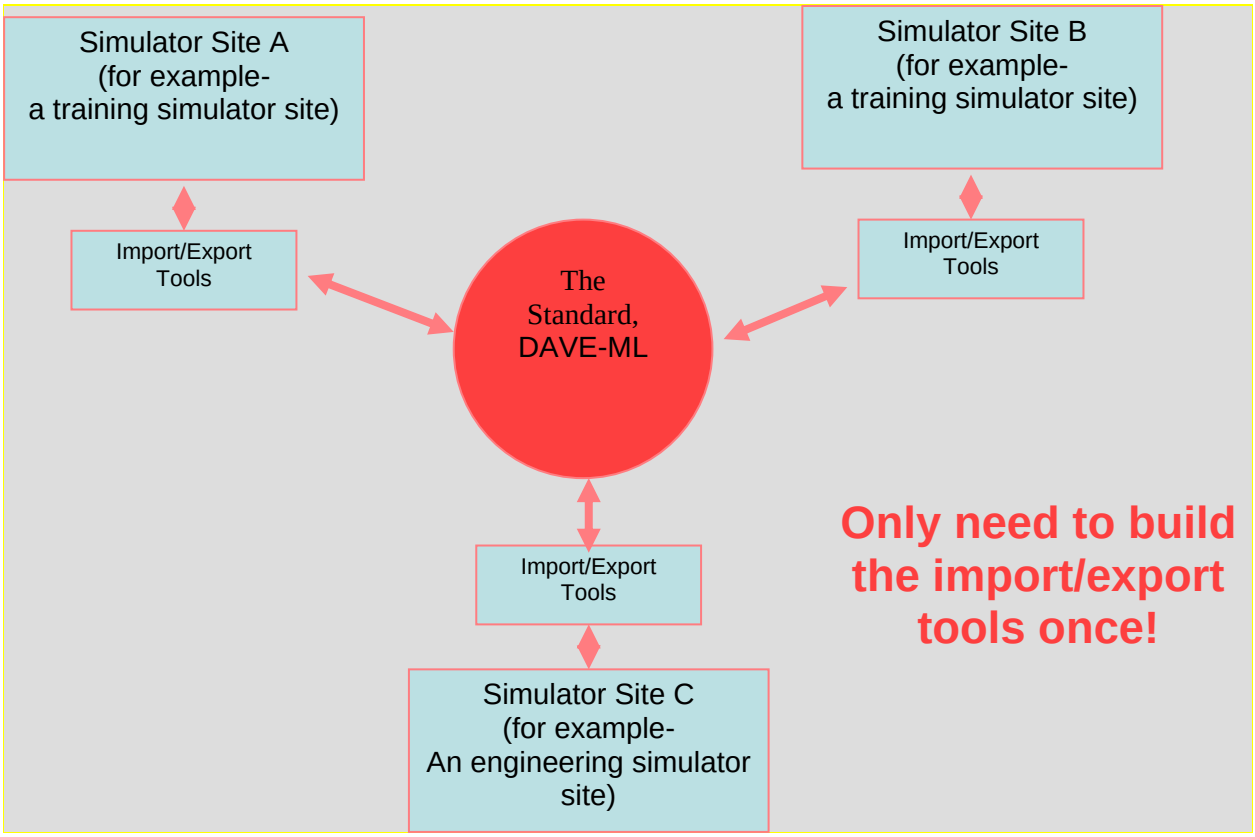

Figure 1. A Standard Format for Exchanging Simulation Modeling Data Between Simulation Facilities

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 DAVE-ML standard. This one time investment would then allow any model to be exchanged easily and with good confidence that the "rehosting" of the model would require minimal effort. Again, the user need not change anything inside his facility or software to use the model.

LIBRARY OF CONGRESS CATALOGING DATA WILL BE ADDED HERE BY AIAA STAFF

Published by
American Institute of Aeronautics and Astronautics
1801 Alexander Bell Drive, Reston, VA 20191

Printed in the United States of America

Contents

Foreward

This standard has been sponsored and developed by the AIAA Modelling and Simulation Technical Committee. Mr. Bruce Jackson of NASA Langley conceived DAVE-ML. DAVE-ML is the embodiment of the standard in XML. It is the data type descriptions for the XML implementation and includes examples of its use. (Annex B)

This implementation was then tested by trial exchange of simulation models between NASA Langley Research Center (Mr. Bruce Jackson), NASA Ames Research Center (Mr. Thomas Alderete and Mr. Bill Cleveland), and the Naval Air Systems Command (Mr. William McNamara and Mr. Brent York). Numerous improvements to the standard resulted from this “testing”.

At the time of approval, the members of the AIAA Committee on Standards were:

Bruce Hildreth	Chair Science Applications International Corporation (SAIC) Bruce.Hildreth@saic.com
Bruce Jackson	DAVE-ML Lead NASA Langley Research Center E.B.Jackson@larc.nasa.gov
Geoff Brian	Defense Science Technical Organization (DSTO) Geoff.Brian@dsto.defence.gov.au
Brent York	Indra Systems, Inc. byork@indra.cc
Michael Silvestro	Charles Stark Draper Laboratory, Inc. msilvestro@draper.com
Barry Bryant	NASA Langley Research Center r.b.bryant@nasa.gov
Bimal Aponso	NASA Ames Research Center baponso@mail.arc.nasa.gov
Jean Slane	Engineering Systems Inc. jhslane@esi-co.com
Peter Grant	University of Toronto prgrant@utias.utoronto.ca
Bill Bezdek	Boeing Phantom Works William.J.Bezdek@boeing.com
Jon Berndt	Jacobs Jon.Berndt@escg.jacobs.com

The above consensus body approved this document in Month 200X.

The AIAA Standards Executive Council (VP-Standards Name, Chairman) accepted the document for publication in Month 200X.

Introduction

The purpose of this standard is to clearly define the information required and format to exchange air vehicle simulation models between simulation facilities. This portion of the standard is implemented in XML and called DAVE-ML.

The Exchange Standard (DAVE-ML) Includes:

- Standard variable name definitions – the purpose of this is to facilitate the transfer of information by using these standard variables as a "common language". Of course, the DAVE-ML standard can be used without using standard variable names. It will just be more difficult because the person exporting the model will have to explicitly define all the variables instead of just a subset unique to the particular model.
- Standard function table definition - this allows easy transfer of non-linear function tables of n dimensions.
- Standard axis system definitions - this is used by the variable names and function tables to clearly define the information being exchanged.
- Standard static math equation representation - for definition of aero model (or other static models) equations. This is implemented using Math-ML.

XML provides a format for the exchange of this information, so each organization only has to make import/export tools to the standard one time, instead of each time they import from, or export to, a new or changed facility or simulation architecture.

The resultants of the use of this standard will be substantially reduced the cost and time to exchange aerospace simulations and model information. Test cases have shown use of the standard results in an order of magnitude reduction in effort to exchange simple models. Even greater benefits should be attained for large or complicated models.

Trademarks

The following commercial products that require trademark designation are mentioned in this document. This information is given for the convenience of users of this document and does not constitute an endorsement. Equivalent products may be used if they can be shown to lead to the same results.

Simulink®

XML- Extensible Markup Language

Math-ML

1 Scope

This standard establishes the **definition of the information** and **format** used to exchange air vehicle simulations and validation data between disparate simulation facilities. This standard is not meant to require facilities to change their internal formats or standards. With the concept of an exchange standard, facilities are free to retain their well-known and trusted simulation hardware and software infrastructures. The model is exchanged through the standard, so each facility only needs to create import/export tools to the standard once. Then they can use these tools to exchange models with any facility at minimal effort, instead of making unique import/export tools for every facility that they exchange with.

The standard includes **definition of the information** in order to clarify the information exchanged. Such clarification includes axis systems referenced, units, and sign conventions used. XML is used as the mechanism to facilitate automation of the exchange of the information. Using the definitions in the standard, a **list of simulation variable names and axis system** is included. This list of standard variables names further simplifies the exchange of information, but is not required.

2 Tailoring

2.1 Partial Use of the Standard is Encouraged

It certainly does not make sense to implement all aspects of this standard for every simulation created, so the following guidelines are provided to encourage appropriate use in each situation.

2.1.1 Creating a new simulation environment: This situation clearly calls for use of the complete standard. In fact, in this situation it is hoped that the team developing this new simulation would add to the list of standard variables and axis systems.

2.1.2 Creating a new simulation model in an existing simulation environment. This is defined as creating a new system model (aircraft dynamic model for example) that will run in an existing simulation environment. The authors of this standard expect that this is the most commonly performed work that will see benefit by application of this standard.

In this case the tailoring guidelines are simple. Apply the standard to the new development aspects of the project and all the function tables. Assuming that most or all of the standard variable names and axis systems are applicable to the simulation, use them for the new code developed for the simulation. In the existing simulation environment that is being reused, for example the equations of motion, there is no need to rewrite the code to use the standard variable names or axis systems. However, in most cases the axis systems used in existing simulation environments will be covered in the standard axis system definitions herein (Section 5.2). Therefore the standard axis systems can easily be referenced to when documenting the simulation and interfaces between the new simulation components and those reused.

2.1.3 Creating or Updating a simulation with a long life expectancy: A pilot training simulator is an excellent example of this type of simulation. This simulation may only be updated every 3-10 years, so at first glance the standard may seem to be less important.

In fact the opposite is true. It is because of the infrequent maintenance that the standard is critical. In this case, in each new software update, the original developers (or last updaters) are probably gone, and the update is being done by new personnel. Therefore, software developed under the standard is much easier to understand by the new software team. They would be working with clear variable definitions that they are familiar with. The function table format is understood and the functions themselves better documented. Changes are recorded for the next software update team some years down the road. The axis system definitions are clear.

In simulations with long expected life, use of the state, state derivative and control conventions as part of the naming convention becomes critical. That is because these variables form the core of the model and control of it. It is critical that the personnel modifying the simulation are able to easily find the states, state derivatives and controls.

2.2 New and Reused Software-Tailoring Guidance

The longer the expected life of the simulation, the more important the use of the standard will be. The above tailoring guidelines may be categorized in to two common situations; new and reused code.

Reused simulation code- reference the standard only when convenient to document interfaces with new code.

New simulation code

- Use standard axis system definitions (Section 5.1) where they coincide with the definitions in the standard.
- Use standard variable names (Section 5.2) to facilitate consistency and simplify documentation requirements. Definitely apply the convention for states, state derivatives and controls wherever possible.
- Use standard function tables (Section 5.3) for ALL function tables. This facilitates consistency in the data, the documentation of the data, and collaboration with other organizations to improve or debug the data.

2.3 Creating New Variable Names and Axis Systems

The standard variable names and axis system definitions are part of the standard to facilitate communication. They provide a “common language” for the exchange. For example, it is not enough to exchange the lift coefficient function. As a minimum the independent variables used to define the function and their units, sign convention, and reference axis system must be defined. This is facilitated by having some standard variable names and axis systems. Of course, new variable names, definitions, and other convenient axis systems may be used to exchange models between simulation facilities. However, in such cases, the exporters and importers must be very careful to carefully define these variables and axes, otherwise, the exchanged model may not produce the desired results. Use of standard variable names and axis systems facilitates the exchange.

The standard includes a methodology for creating new standard variables. Its use is encouraged. Annex C provides the URL for submitting additional standard variable

names and axis systems or comments on existing standard variable names and axis systems.

3 Applicable Documents

The following documents contain provisions which, through reference in this text, constitute provisions of this standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

ANSI/AIAA Recommended Practice R-004-1992, Atmospheric and Space Flight Vehicle Coordinate Systems, 28 February 1992.

This is the normative reference for the coordinate systems used in this standard.

Tools which will help the community implement and use the standard are available at the URL referenced in Annex C.

Chapter 1 Vocabulary

Acronyms and Abbreviated Terms

AIAA	American Institute of Aeronautics and Astronautics
ANSI	American National Standards Institute

4 Terms and Definitions

For the purposes of this document, the following terms and definitions apply.

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 For example: $C_L(\alpha,\beta)$. C_L is the dependent variables. Also called the output.

Independent Variable - The variable or variables of which the independent variable is a function. For example: $C_L(\alpha,\beta)$. α and β are independent variables.

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

Two Dimensional Table - A table where there are two independent variables. For example, $C_L(\alpha,\beta)$.

Static Equation – An equation where the output (left hand side) does not have direct dependence (right hand side) on a simulation state.

Simulation States and State Derivatives – In the formulation of a simulation model shown as :

$\dot{x} = Ax + Bu$

$y = Cx + Du$

x represents a vector of the simulation states.

$\dot{x}$ represents a vector of the simulation state derivatives.

u represents a vector of the simulation controls (inputs)

Function Table – The numeral set of data points used to represent non-linear relationships between an independent variable based on (as a function of) one or more independent variables. Example $C_L(\alpha,\beta)$ is represented by a function table.

Gridded Table- A multi-dimensional function table where the independent variable breakpoints points do not change for different values of other independent variables. This is sometimes called an orthogonal table.

All one-dimensional tables are a “gridded table”.

Ungridded Table - A multi-dimensional function table where the independent variable breakpoints points do change for different values of other independent variables. This is sometimes called a non-orthogonal table.

5 CLAUSES

5.1 STANDARD SIMULATION AXIS SYSTEMS

5.1.1 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 5.1.4.1]and the *Distributed Interactive Simulation* (DIS Application Protocols, Version 2, IST-CR-90-50, March 1994) [Reference 5.1.4.2]. Reference 5.1.4.1 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 (see section 5.2 below).

5.1.2 Axis System Conventions

In general, ANSI/AIAA R-004-1992 [Reference 5.1.4.1] should be referred to as the normative reference on axis system definitions. These axis systems are discussed in Table 5.1-1 below.

However, in addition, it is important to emphasize the correlation of reference [5.1.4.1] and [5.1.4.2] axis systems. The geocentric earth fixed axis system and body axis coordinate system axis system.

Geocentric Earth Fixed-Axis System

The Geocentric Earth Fixed-Axis System (Axis System 1.1.3 of the table below) is identical to the DIS “Geocentric Cartesian Coordinate System” (also referred to “World Coordinate System” of Reference 5.1.4.2).

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.

Body Axis Coordinate System

The another standard axis system is the Body Axis System (axis system number 1.1.7 in reference 5.1.4.1). This is identical to the DIS “Entity Coordinates System” in Reference 5.1.4.2.

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

Flat Earth Axis System

The flat earth 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 cg 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 system is referred to in the variable names as “FE”.

Complete List of Axis Systems

The axis systems that are referenced are taken largely from paragraph 1.1 of ANSI/AIAA R-004-1992, “Atmospheric and Space Flight Coordinate Systems” (reference 5.1.4.1). The flat earth axis system for atmospheric flight simulation approximation is added to that reference to make the complete set of axis system references. Table 5.1-1 below is the comprehensive list of axis systems that may be used.

The first column in the table below provides the abbreviation used for each axis system. The axis system may be referenced in a variable name. See the section below on the variable naming convention.

Table 5.1-1 Standard Axis Systems and the reference abbreviation used to define the axis system in a standard variable name.

Reference Abbreviation for Variable Names	Reference 5.1.4.1 Paragraph No.	Term	Definition	Symbol
EI (Earth centered inertial)	1.1.1	Geocentric Inertial axis system (See Appendix D.2 of 5.1.4.1 for a modification of this system used for launch vehicles.)	An inertial reference system of the FK5 mean equator and equinox of J2000.0 has the origin at the center of the Earth, the X_I-axis being the continuation of the line from the center of the Earth through the center of the Sun toward the vernal equinox, the Z_I-axis pointing in the direction of the mean equatorial plane's north pole, and the Y_I-axis completing the right-hand system. (See Figure 1A in Reference 5.1.4.1)	$X_I Y_I Z_I$
Not used, this forms a basis for other definitions	1.1.2	Earth-fixed axis system	A right-hand coordinate system, fixed relative to and rotating with the Earth, with the origin and axes directions chosen as appropriate.	$X_o y_o z_o$
GE (also called ECEF for earth centered earth fixed)	1.1.3	Geocentric Earth-fixed axis system	A system with both the origin and axes fixed relative to and rotating with the Earth (1.1.2). The origin is at the center of the Earth, the x_G-axis being 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 to the north, and the y_G-axis completing the right-hand system. (See Appendix D.3 in Reference 5.1.4.1)	$x_G y_G z_G$
	1.1.4	Normal Earth-fixed axis system	An Earth-fixed axis system (1.1.2) in which the z_o-axis is oriented according to the downward vertical passing through the origin (from the origin to the nadir). (See Figure 1C in Reference 5.1.4.1)	$x_o y_o z_o$ NOTE- However, $x_g y_g z_g$ is an acceptable alternative.
VO	1.1.5	Vehicle-carried orbit-	A system with the origin fixed in the vehicle, usually the center of mass,	$x_o y_o z_o$

		defined axis system[1]	in which the z_o-axis is directed from the spacecraft toward the nadir, the y_o-axis is normal to the orbit plane (positive to the right when looking in the direction of the spacecraft velocity), and the x_o-axis completes the right-hand system. (See Figure 1A in ANSI/AIAA R-004-1992)	
VE	1.1.6	Vehicle-carried normal Earth axis system[1]	A system in which each axis has the same direction as the corresponding normal Earth-fixed axis, with the origin fixed in the vehicle, usually the center of mass.	$x_o y_o z_o$ NOTE- However, $x_g y_g z_g$ is an acceptable alternative.
Body	1.1.7	Body axis system[1]	A system fixed in the vehicle, with the origin, usually the center or mass, consisting of the following axes:	$x_B y_B z_B$
		Longitudinal axis	An axis in the reference plane or, if the origin is outside that plane, in the plane through the origin, parallel to the reference plane, and positive forward.[2] In aircraft or missiles, this is normally from the CG forward towards the nose in the vertical plane of symmetry. It is also normally parallel to the waterline of the vehicle.	x_B
		Lateral axis	An axis normal to the reference plane and positive to the right of the x-axis (henceforth, positive to the right).	y_B
		Normal axis	An axis which lies in or parallel to the reference plane, whose positive direction is chosen to complete the orthogonal, right-hand system xyz.	z_B
Wind (for Wind axis system)	1.1.8	Air-path system[1)]	A system with the origin fixed in the vehicle, usually the center of mass, consistent of the following axes:	$x_w y_w z_w$
		x_a-axis; Air-path axis	An axis in the direction of the vehicle velocity relative to the air (1.5.1).	x_w
		y_a-axis; lateral air-path axis; cross-stream axis	An axis normal to the air-path axis and positive to the right.	y_w

		z_a-axis; normal air-path axis	An axis – in the reference plane or, if the origin is outside that plane, parallel to the reference plane, and – normal to the air-path axis. The positive direction of the z_a-axis is chosen so as to complete the orthogonal, right-hand system $x_a y_a z_a$.	z_w
SA (for Stability Axis system)	1.1.9	Intermediate axis system[1)] x_e-axis y_e-axis z_e-axis	A system with the origin fixed in the vehicle, usually the center of mass, consisting of the following axes. The projection of the air-path axis on the reference plane, or, if the origin is outside that lane, on the plane through the origin, parallel to the reference plane. An axis normal to the reference plane and positive to the right, coinciding with or parallel to the lateral axis (1.1.7). An axis which coincides with or is parallel to the normal air-path axis so as to complete the orthogonal right-hand system.	$x_s y_s z_s$ x_s y_s z_s
FP	1.1.10	Flight-path axis system[1)]	A system with the origin fixed in the vehicle (usually the center of mass) and in which the x_k-axis is in the direction of the flight-path velocity relative to the Earth. The y_k axis is normal to the plane of symmetry and positive to the right. The z_k axis completes the orthogonal right-hand system	$x_k y_k z_k$
AA	1.1.11	Total-angle-of-attack axis system[1)]	A system with the origin fixed in the vehicle, usually the center of mass, in which the x_Γ-axis is coincident with the x-axis in the body axis system	$x_\Gamma y_\Gamma z_\Gamma$

		(USA practice: areoballistic axis system.)	(1.1.7). The y_Γ axis is perpendicular to the plane formed by the x_Γ axis and the velocity vector, positive to the right. The z_Γ axis is formed to complete the orthogonal, right-hand system.	
FE		Flat earth system (not from reference 5.1.4.1)	The Flat Earth reference system is situated on the earth's surface directly under the cg 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.	$x_{FE}y_{FE\Gamma}z_{FE}$
LL		Locally Level axis system (not from reference 5.1.4.1)	A vehicle related axis system (1.1.6) with the origin on the smooth surface of the earth and moving with the vehicle. The –Z axis passes through the vehicle CG. The X axis is tangential to the smooth surface of the earth and oriented toward true north in the geometric earth model. The Y axis is tangential to the smooth surface of the earth completing the right hand triad (East).	$x_{LL}y_{LL\Gamma}z_{LL}$

[1)] Usually the origins of the axis systems defined in 1.1.5 through 1.1.11 coincide. If that is not the case, it is necessary to distinguish the different origins by appropriate suffixes.

[2)] The reference plan should be a plane of symmetry, or a clearly specified alternative.

5.1.3 Summary

This axis system standard should be followed for all future equations of motion. Additionally, it provides the naming convention to properly reference the definitions herein in simulation variable names.

5.1.4 References

5.1.4.1 ANSI/AIAA Recommended Practice R-004-1992, Atmospheric and Space Flight Vehicle Coordinate Systems, 28 February 1992.

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

CLAUSE

5.2 STANDARD SIMULATION VARIABLES

5.2.1 Background / Philosophy

The Rationale for Having Standard Variable Name and Naming Conventions

The standard variable names and axis system definitions are part of the standard to **facilitate communication**. They provide a "common language" for the exchange. For example, it is not enough to exchange the lift coefficient function. As a **minimum** the independent variables used to define the function and their units, sign convention, and reference axis system must be defined. This is facilitated by having some standard variable names and axis systems.

Therefore, if you exchange models using the standard variables, you don't have to define a variable that is part of the standard, just refer to it in the standard. Additionally, the variable naming convention is presented to allow the list of standard variables to grow as needed by the user community. Hopefully the convention will keep some consistency in the variable names and make them easier for users to interpret.

States and State Derivatives

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 these variables is crucial in software maintenance.

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

By practice, 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 immutable fundamental concepts. We "fudge" things and create states from outputs. Practically speaking, this is done because we can't determine what the states in the simulation are. Since a simulation 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 remember 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.

The identification of states and state derivatives is simply for the purpose of encouraging good mathematical fundamentals and to facilitate software maintenance. Therefore this AIAA Simulation Variable standard identifies states and state derivatives as part of the naming convention.

Identification of controls (also called inputs), while a good idea, is very difficult because so many variables are controls and the controls change with the mode of operation of the simulation. As a consequence, identification of controls is optional but should be strongly considered for inclusion in the development of new dynamic simulation models.

5.2.2 Variable Naming 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.

The C language convention is supported. The only "oddity" is that 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. The standard could easily be used following the Ada naming convention by using underscores to separate the parts of the variable names.

General rules for naming variables:

> Variables shall have meaningful names. Mnemonics will not be used. Standard abbreviations are allowed.

The first word in the variable name (not including the prefix, if any) starts with a lower case. Distinct words thereafter in variable names shall be separated by capitalization of the separate words (C example" XbodyAccel" [Ada example would be " X_body_accel"]).

Varaible names shall not exceed 60 characters in length. Brief, but complete names are most effective.

Abbreviations are generally all capitals.

5.2.3 Methodolgy for Creating New Names

The suggested method of creating the name is as follows.

Each name has 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))(Specific axis or reference) (Axis or reference system)(Core name)_(units)

As will be seen, very rarely, if ever, are all 6 components of a name used.

5.2.3.1 Prefix

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

The prefix is always separated from the body of the variable by an underscore or as a separate component of a structure.

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 controls (inputs) are the key variables for validation. All outputs are computed directly or indirectly from states and controls.

Clearly, the formulation of the equations of motion and the model itself determines what variables are states. **This naming convention is not meant to standardize on any variable as a state, just for the simulation engineer to explicitly name them,** making it easier to document and exchange the models.

Examples:

```
s_bodyXVelocity_fs_1                s_ prefix indicates that this
                                    variable is a state

sd_bodyXAcceleration_fs_2           sd_ prefix indicates that this
                                    variable is a state derivative
```

Identification of Controls (optional) The controls are those variables which provide the pilot/crew or the simulation operator's inputs to the simulation. As with the states and state derivatives, the controls are the key variables for validation. All outputs are computed directly or indirectly from states and controls.

Again, the formulation of the equations of motion and the model itself determines what variables are controls. **This naming convention is not meant to standardize on any variable as a control, just for the simulation engineer to explicitly name them,** making it easier to document and exchange the models.

Examples:

```
c_aileronPosition_d                 c_ prefix indicates that this
                                    variable is a control

c_longControlPos_r                  c_ prefix indicates that this
                                    variable is a control
```

5.2.3.2 Variable Source Domain

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

Some domain examples:

Aero or Aerodynamic
Engine or Thrust
Controls
Wheel
Landing Gear
Hydraulic
Electrical
IO (for input/output)
Motion
CL or ControlLoading
Radar
Weapons
AIM9X (for AIM-9X missile)
.
.
.

Users may (should) add as many domains as needed to clearly identify the variable.

Variable name examples using “aero” and “thrust”:

```
aeroXBodyForceCoefficient
aeroXBodyForce_lbf
thrustXBodyForce_lbf
```

5.2.3.3 Specific Axis or Reference

This is the specific axis or reference used within the axis system (axis systems are defined in Section 5.1 above). If the axis system is included in the name, then the specific axis or reference should also be included.

For example:

(X, Y, Z), (N, E, D) or (U, V, W) is for linear/translational motion
(Pitch, Roll, Yaw) or (P, Q, R) for angular motion

Variable name examples:

`s_rollBodyRate_rs_1`	where `Body` is the axis system and `roll` is the specific axis in the body axis system. Roll indicating angular motion. NOTE: in this example `rollBodyRate` is designated as a state.
`XbodyTurbulenceVelocity_fs_1`	where `Body` is the axis system and `X` is the specific axis in the body axis system. X indicating translational motion.
`EGEVelocity_ms_1`	where `GE` is the axis system and `E` (East) is the specific axis, also indicating translational motion.
`Zrunway22Velocity_fs_1`	where `Runway22` is the axis system (user defined) and `Z` is the specific axis, also indicating translational motion.
`YbodyPilotAccel_ms_2`	where `Body` is the axis system and `Y` is the specific axis, also indicating translational motion.

Examples as a vector:

s_bodyAngularRate_rs_1(3)
Where element 1 would be about the X axis (pitch), element 2 would be about the Y axis (roll) and element 3 would be about the Z axis (yaw).

Alternatively

The specific axis or reference can logically be a vector or an array. When vectors are used, a right handed triad in order (x, y, z) must be used to avoid confusion.

5.2.3.4 Axis or Reference System

This is the axis or reference system to which the variable is referenced. Table 5.1 above specifies that standard axis system abbreviations that should be used. If no axis system pertains to the variable or 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 correspond to the standard axis systems discussed in section 4 above.

Conventions Used

Earth fixed frames and Local reference frames

By convention these axis systems use X, Y, Z, Pitch, Roll, and Yaw for axis references.

Local reference frames (LL for example) origin and orientations may be user defined. They are meant for runway, test range, target reference, navigational aid, etc. coordinate systems.

Body fixed frames use U, V, W, Pitch, Roll, Yaw for axis references.

Variable name examples

`UbodyVelocity_fs_1`	
`s_XGEVelocity_fs_1`	in the case where the equations of motion are formulated such that the variable IS a state
`XGEVelocity_fs_1`	in the case where the equations of motion are formulated such that the variable IS NOT a state
`UbodyVelocity_ms_1`	
`VbodyVelocity_fs_1`	
`s_XLLVelocity_fs_1`	
`s_XFEVelocity_fs_1`	
`pitchBodyRate_rs_1`	
`rollBodyRate_rs_1`	
`rollBodyAccel_rs_2`	

5.2.3.5 Core Variable Name

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:

```
velocity
rate
accel
forceCoefficient
turbulenceVelocity
angleOfAttack
angleOfSideslip
cosineOfAngleOfSideslip
thrust
torque
aileronDeflection        (aileron could be considered a domain and
                          deflection the core name)
```

Variable name examples:

```
s_rollBodyRate_rs_1
XbodyTurbulenceVelocity_fs_1
ZGEVelocity_fs_1
angleOfAttack
angleOfSideslip
cosineOfAngleOfSideslip
aileronDeflection_d
```

5.2.3.6 Suffix- Units

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 therefore 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.

An analogy for to standard for exponential expression is used for specifying units. A standard expression for feet cubed per second squared (for example) would be:

f^3s^{-2}

By eliminating the superscript we have:

f3s-2

However, a compiler would interpret this as subtracting 2 from f3s. Therefore instead of using the negative sign for exponents, we replace it with the underscore. Thus:

Feet cubed per second squared can be represented as:

f3s_2

So feet per second is fs_1 and feet per second squared is fs_2. Every term in the denominator has an exponent. For example:

(r/s^2)/(f*lbf) would be expressed as rs_2f_llbf_1.

Some more examples:

trueAirspeed_fs_1	for feet per second (f/s)
and in the other case	
trueAirspeed_ms_1	for meters per second (m/s)
or	
trueAirspeed_nmih_1	for knots (nautical miles per hour)

The standard defines what the variable name for airspeed is, the user defines what units they are 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.

Table 5.2-1 Abbreviations used to designate units in standard variable names

time		
hours	h	
seconds	s	(SI Standard)
minutes	min	
milliseconds	ms	(milli prefix m)
length		
inches	inch	
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)	d	
radians	r	
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)

5.2.4 Some Additional Discussion

Very rarely, if ever, are all 6 components of a name used. In the case of:

```
s_rollBodyRate_rs_1
```

5 components were used:

(prefix [s] indicating that in this formulation of the equations of motion this variable is a state),
(specific axis or reference [Roll]),
(axis or reference system [body]),
(core name [Rate]), and

(units suffix [rs_1]).

In this case “variable source domain” was omitted because “`s_rollBodyRate_rs_1”` is a variable defined by the laws of physics and there cannot be a body rate from aerodynamics and a body rate from the moments produced by the engine. If however, the user wanted to have a multi-body simulation, logically the “variable source domain” could be used to discriminate between different elements of the body, or, perhaps more logically, an array or structure would be used to define different elements in a multi-body or flexible structure problem.

The *whole point here is to help provide clear communication when exchanging models, not force variable names*. “s_rollBodyRate_rs_1” is intended to be a clear, brief unambiguous name for the variable.

5.2.4.1 Initial Condition Convention

A helpful convention that may be used is adding IC to the end of any variable name, but before the units to designate that the variable is an initial condition specification. This can be added to virtually any variable, conceptually creating a constant.
For example:

```
s_rollBodyRateIC_rs_1
grossWeightIC_kg
```

5.2.4.2 Discarded Conventions and Reasons

We have considered having a prefix for simulation outputs as well as states and controls, but at the present time this has been discarded due to the fact that the outputs required vary so widely, and there are typically an extremely large number of outputs. Practically speaking, every variable not a state, state derivative or control could be considered an output.

We considered eliminating the suffix when the units were one of the “standards”, but discarded this concept due to the fact that always having the units attached to the variable will help the programmer/engineer have consistent units when they are programming and reduce programming errors due to mixing of the units improperly. It also noticeably reduces the software maintenance effort when years after initial development, another software engineer is trying to unravel the code and make bug fixes or enhancements.

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. If the AIAA standard develops to the point in the future where there is standard software, architecture or structure offered, these variable names could be used in that software but will not be changed.

5.2.5 Standard Variable Name Table Example

Using the conventions discussed above, a set of standard variable names has been created. These are presented in Annex A. An excerpt of Annex A is given below for illustrative purposes.

Interpretation of the standard variable name annex is best given by example. In the table below is standard variable defining the Roll Euler Angle, its axis system and positive sign convention (+ = RWD, or right wing down). Four name examples are provided:

- The short name, PHI
- One of more full names using the standard units convention**. Generally,** one full name with American convention units and one with SI units. Any suitable units may be used.
- A description of the variable. When applicable the description should include the axis system in which the variable is defined.
- The POSITIVE sign convention of the variable
- Minimum and Maximum values of the variable, normally only specified for angles

In addition this example also illustrates the pitch and yaw Euler angles.

Since roll, pitch and yaw may also conveniently be expressed as a vector, the shaded area is the standard definition of the Euler angle vector. Again, eulerAngle_r(3) would be the standard vector using radians as the units and is fully compliant with the standard.

The standard allows use of any of the standard set of units.

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Min Value	Max Value
		Vehicle Positions and Angles				
$\underline{\varepsilon}$	EUL(3)	`eulerAngle_d(3)` `eulerAngle_r(3)`	Vector of the roll, pitch, and yaw Euler angles comprised of the elements defined below. LL (locally level) frame.			
Φ	PHI	`rollEulerAngle_d` `rollEulerAngle_r`	Roll Euler Angle, LL frame.	RWD	-180,-π	180, π
Θ	THET	`pitchEulerAngle_d` `pitchEulerAngle_r`	Pitch Euler Angle, LL frame	ANU	-90, -π/2	90, π/2

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Min Value	Max Value
ψ	PSI	`yawEulerAngle_d` `yawEulerAngle_r`	Yaw Euler Angle, LL frame	ANR	-180,-π	180, π

5.2.6 Summary

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 (true airspeed, indicated airspeed, calibrated airspeed?, etc.), including units and axis system.

Using the "standard variable name" simply provides a common language and set of definitions within which to facilitate transfer of the model.

The simulation community is encouraged to propose additional standard variable names. Annex C describes the web site used to support this standard. There is an appropriate URL or email address for submitting additional names or recommend clarification of existing names.

5.2.7 References

None

CLAUSE

5.3 STANDARD SIMULATION FUNCTION TABLE DATA FORMAT AND XML IMPLEMENTATION OF THE STANDARD: DAVE-ML

5.3.1 Purpose

This section will explain the data requirements which a Standard Function Table Format must be able to satisfy. It includes the content of the information contained in the table and configuration management of the data in the table. As you will see, the definition of the table format includes data for all these components.

This document also discusses conceptually how the data table should be looked up in an executable program.

The standard is implemented in XML as specified by DAVE-ML, Annex B. Annex C provides links to example programs for loading and looking up data in the XML standard.

Philosophy

Probably the most immediate benefit of the standard to the simulation discipline is one that defines formats for the interchange of tabular data. Tabular data is used almost universally for non-linear function generation of aerodynamic, engine, atmospheric, and many other model parameters. The easy interchange of such data can greatly improve efficiency in the simulation community.

Most simulation developers and users have addressed this issue locally. In many simulation communities, a family of tools has been built around existing local function table standards. Thus, the intent of this standard 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. **This is an exchange standard.** It is hoped that this standard will eventually be adopted for local use as well, but that is not required for the standard to succeed.

5.3.2 Design Objective

The design objectives of the Standard Data Table Format were first and foremost to make a data format that would include all the information about real multi-dimensional data, not just the data values. This notably is the fact that, in 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 (provenance, via reference), and a confidence interval for the data. Confidence Intervals can be used for Monte Carlo simulations and to mathematically combine two different estimates of the same parameter at the same point. Therefore, confidence statistics 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.

5.3.3 Standard Function Table Data-An Illustrative Example

Figure 5.3-1 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. More generally stated, a function output (dependent variable), CLALFA is dependent on three inputs (independent variables), angleOfAttack_d, mach, and avgElevatorDeflection_d.

Close examination of the example 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 (angleOfAttack_d) breakpoints, but also a different number of Mach number (mach) and control position (avgElevatorDeflection_d) breakpoints. We have defined this as an “Ungridded Table”. (A “Gridded Table” is one where the number of breakpoints of a specific independent variable are the same for each of the other independent variables. For example, there are the same number of Mach breakpoints for each angle of attack breakpoint.)

2. The values (breakpoints) of the independent variables are different. Again, an “Ungridded Table”.

3. The valid ranges of the independent variables are different. (“Ungridded Table”)

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

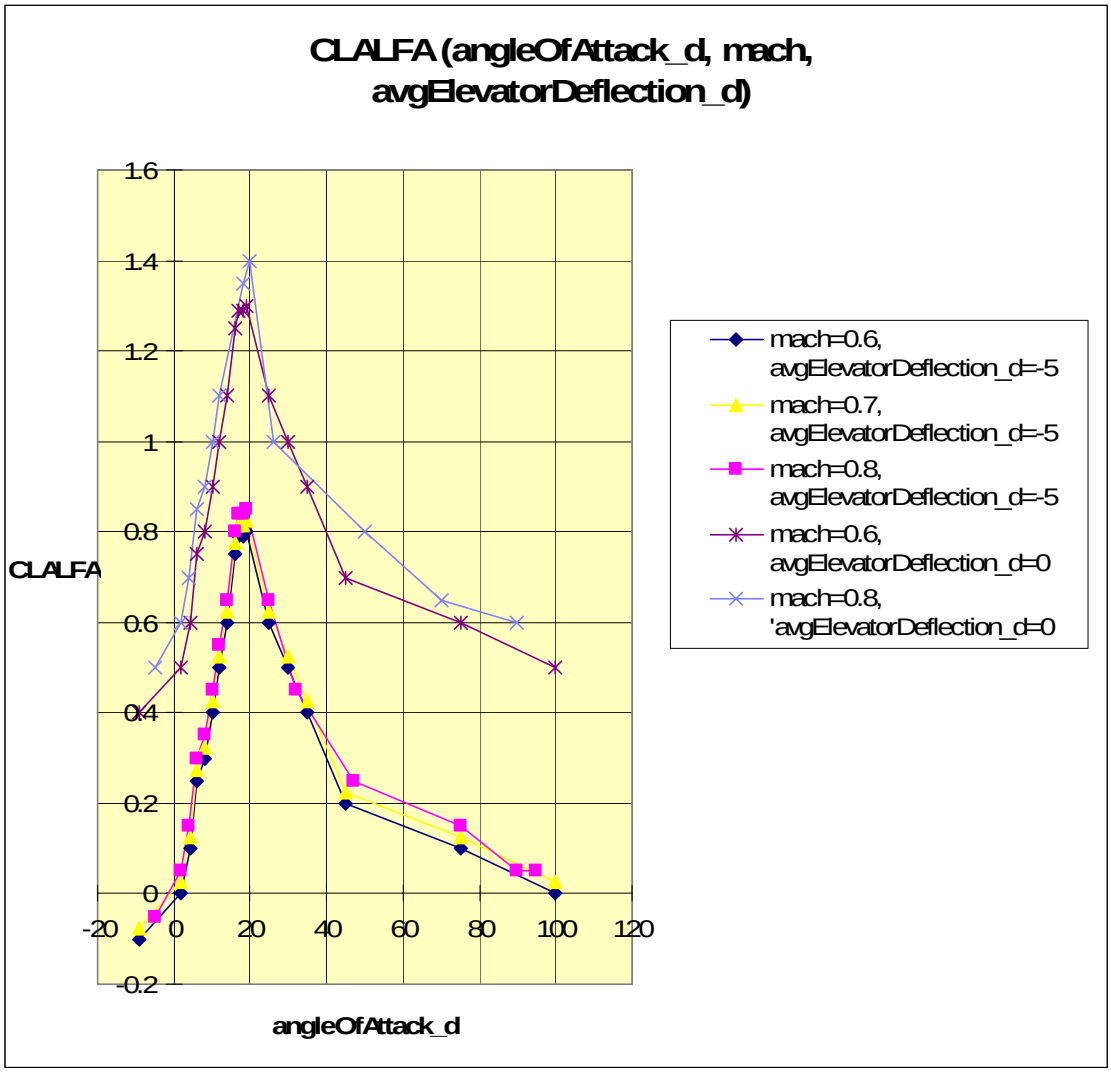

Figure 5.3 -1 An Illustration of a 3 Dimensional Function Table, CLALFA (angleOfAttack_d, mach, avgElevatorDeflection_d)

For 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? For example what report?
- how is it defined? For example, is this at a specific altitude? What configuration is it for?
- what are the engineering 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? Not where the data came from, but what person decided that this was the correct data for this table?
- how has it been modified and for what reason?

- 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. It has been implemented in XML as seven Major Elements and is discussed in detail in Annex B. An introduction and overview will be provided here.

Additionally, DAVE-ML also includes the ability to automate static checks of the function data to allow spot checking of the function after it has been exchanged.

5.3.4 DAVE-ML Major Elements (reference Annex B)

These major elements are provided in the same order as they must be in the XML files. In general, most attributes and sub-elements are optional. In fact, only the *fileHeader* and *variableDef* major elements are required.

The logical flow of information is such that the lower major elements refer upward to information previously defined, in general, so that information (breakpoints, data points, provenance, etc.) that is re-used in more than one function does not need to be repeated.

1. *fileHeader*- the fileHeader contains the file provenance (who created the file and how to contact that person or team), all references and overall description about all the functions in this particular file. The provenance of each particular function refers to the fileHeader.
2. *variableDef* – defines the signals used (variables) to generate the functions, at a minimum, the independent variables (inputs) and the dependent variables (outputs). Additionally, it includes the definition of any intermediate variables used to generate the functions, and defines any calculations that are to be performed (defined as MathML).
3. *breakpointDef* - here, all the breakpoints, or independent variable data points, for gridded tables are defined. One set of breakpoints may be used by many functions. This section does not apply to ungridded tables. They contain their breakpoints within the *ungriddedTableDef* major element. There may be a provenance for the breakpoints, which again may refer to the *fileHeader*.
4. *griddedTableDef* - contains the data points of the function. These data points use the breakpoints defined in the breakpointDef major element. The provenance of each set of data points may be explicitly defined here, and may refer to documents defined in the *fileHeader*.
5. *ungriddedTableDef* - contains the breakpoints and the data points of the ungridded tables. These are specified as sets of breakpoints and data points together and do not refer to the *breakPointDef* major element. As in *griddedTableDef*, the provenance of each set of data points may be explicitly defined here, and may refer to documents defined in the *fileHeader*.
6. *function* – combines the breakpoints with the data points, and defines which independent variables are used as inputs to the functions. This element also includes definition of how

the function should be interpolated and extrapolated, and is the definitive element to include provenance on the particular function (where did the data for this function come, who decided this set of data points would be used for this function, etc.). The nonlinear function definition is complete at this point.

7. *checkData* - contains a set of static check cases to verify the functions. It includes an optional tolerance on the outputs. If the checkData element is used, it must include check cases for all outputs in the file (it cannot check some functions and not others).

Annex B contains a detailed description and examples of the data element definitions of the DAVE-ML function table standard. Appendix A of Annex B provides detailed XML element references and descriptions.

5.3.5 A Simple DAVE-ML Example

The easiest way to understand the standard is through an example. Annex B contains many more examples of the DAVE-ML implementation of the standard.

Lets take a simple one dimensional aero table, in this case pitching moment coefficient as a function of angle of attack, Table 5.3-1 and Figure 5.3-2 below.

Table 5.3-1. A Simple Function

angleOfAttack_d	0	18	19	20	22	23	25	27	90
cm(angleOfAttack_d)	0.1	-0.1	-0.09	-0.08	-0.05	-0.05	-0.07	-0.15	-0.6

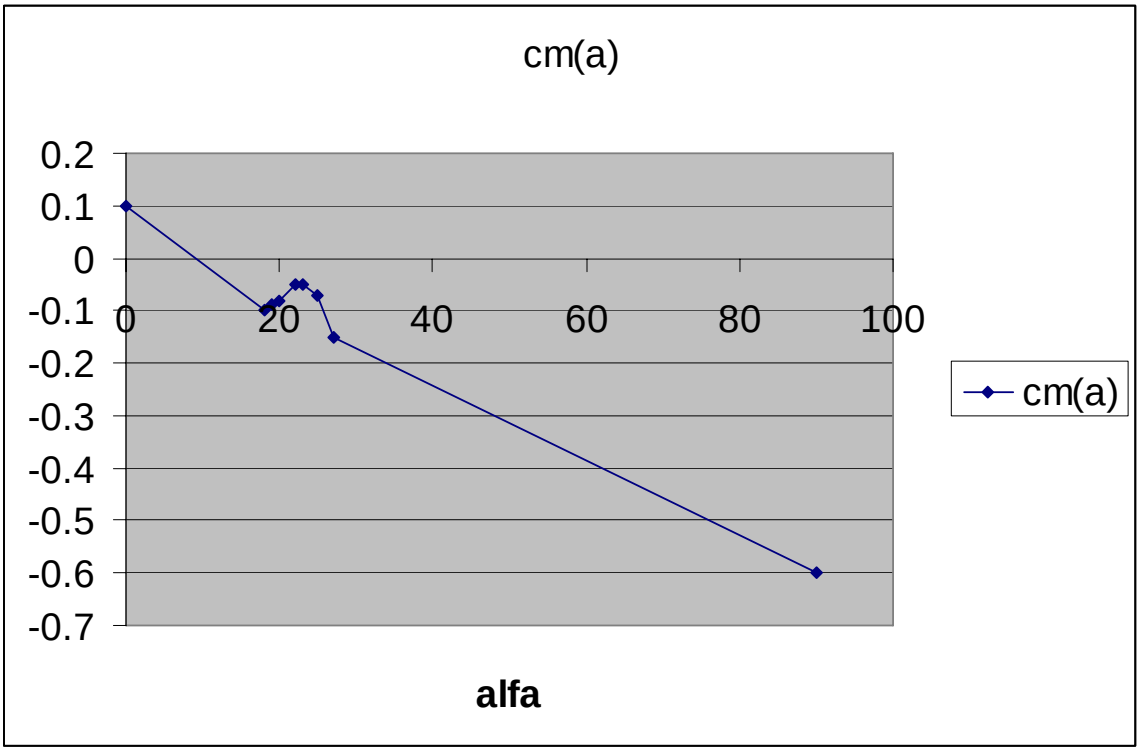

Figure 5.3-2. The $C_m(\alpha)$ Function- A simple one dimensional gridded function.

The DAVE-ML for this function could be:

```
CmaExample.dml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE DAVEfunc PUBLIC "-//NASA//DTD for Flight Dynamic Models - Functions
2.0//EN" "DAVEfunc.dtd">
<DAVEfunc>

<!--             ==========================                 -->
<!--=============  File Header Components  ===============  -->
<!--             ==========================                 -->
<fileHeader>

  <!--  This is an example of the file header components of the
          derivative of Cm as a function of angle of attack.  It must
        remembered that all fileheader components of all functions
        in the file must be grouped together into one file header
        area.

          Also note that there is not much information in this header,
          Mainly because it is mean to be a simple example.  In
          reality, probably the most important information is the
          author, the reference and the modification record, because
          these data describe where the data came from and if it has
          been changed (and how).  See annex B for more complete
          examples.
 -->

  <author name="Bruce Hildreth" org="SAIC" email="bruce.hildreth@saic.com"/>
  <fileCreationDate date="2006-03-18"/>
  <description>
    This is made up data to use as an example of a simple gridded function.
  </description>
  <reference refID="BLHRpt1" author="Joe Smith"
           title="A Generic Aircraft Simulation Model (does not really
exist)"
           accession="ISBN 1-2345-678-9" date="2004-01-01"/>

 <!-- no modifications so far, so we don't need a modificationRecord yet -->

</fileHeader>

<!--             ==================================                 -->
<!--=============  Variable Definition Components  ===============  -->
<!--             ==================================                 -->


 <!-- Input variable -->

 <variableDef name="Angle of attack" varID="angleOfAttack_d" units="deg" >
   <isStdAIAA/> <!-- Indicates that this variable is a standard
                 variable, which is why the author omitted
                 description and sign convention
                 and any other info. (it certainly could
                 be included here)                              -->
```

```
</variableDef>

<!-- Output (function value) -->

<variableDef name="Pitching moment coefficient due to angle of attack"
           varID="CmAlfa" units="nondimensional" sign="+ANU">
  <description>
    The derivative of total pitching moment with respect to
    angle of attack.
 </description>
</variableDef>

<!--             ============================                 -->
<!--=============  Breakpoint Definition Set  ===============  -->
<!--             ============================                 -->

<breakpointDef bpID="angleOfAttack_d_bp1">

  <!--
    Note that the bpID can be any name for the breakpoints.  The
    author here chose to use a name related to the independent
    variable that is expected to be used to look up the function.  In
    fact, if this set of breakpoints were shared by many functions
    and different independent variables would be used to look up the
    function, then the bpID of "angleOfAttack_d_BP1" would be
    misleading and a more generic name like "AOA" would probably be
    better.
  -->

  <description>
    Angle of attack breakpoint set for CmAlfa, CdAlfa, and ClAlfa
  </description>

  <bpVals>    <!--  Always comma separated values  -->
    0, 18, 19, 20, 22, 23, 25, 27, 90
  </bpVals>

</breakpointDef>

<!--             ============================                 -->
<!--=============  Gridded Table Definition   ===============  -->
<!--             ============================                 -->

<griddedTableDef gtID="CmAlfa_Table1">
  <description>
    The derivate of Cm wrt fuselage AOA in degrees
  </description>

  <provenance>
    <author name="Jake Smith" org="AlCorp"/>
    <functionCreationDate date="2006-12-31"/>
    <documentRef refID="BLHRpt1" />  <!--  This points back to the Header,
                                 which provides the information
                                 about BLHRpt1. -->
  </provenance>
```

```
    <breakpointRefs>
      <bpRef bpID="angleOfAttack_d_bp1" />
    </breakpointRefs>

    <uncertainty effect="percentage">
      <normalPDF numSigmas="3">
        <bounds>12</bounds>
      </normalPDF>
      <!--  This means that the 3 sigma confidence is +-12% on the Data. -->
    </uncertainty>

    <dataTable>          <!--  Always comma separated values -->
      0.1,-0.1,-0.09, -.08, -0.05, -0.05, -0.07, -0.15, -0.6
    </dataTable>

  </griddedTableDef>

  <!--             ========================                -->
  <!--=============  Function Definition   =============== -->
  <!--             ========================                -->

  <!-- The function definition ties together input and output variables
       to table definitions. This allows a level of abstraction such
       that the table, with it's breakpoint definitions, can be reused
       by several functions (such as left and right aileron or multiple
       thruster effect tables).
  -->

  <function name="Cm_alpha_func">
    <description>
      Variation of pitching moment coefficient with angle of attack (example)
    </description>
    <independentVarRef varID="angleOfAttack_d"/>
    <dependentVarRef varID="CmAlfa"/>
    <functionDefn>
      <griddedTableRef gtID="CmAlfa_Table1"/>
    </functionDefn>
  </function>

  <!--             ======================                -->
  <!--=============  Check Data Cases   =============== -->
  <!--             ======================                -->

  <!-- Checkcase data provides automatic verification of the model by
       specifying the tolerance in output values for a given set of
       input values. One 'staticShot' is required per input/output
       mapping; in this case for a single input, single output model,
       we have a single input signal and a single output signal in each
       test point.
  -->

  <checkData>
    <staticShot name="case 1">
      <checkInputs>
        <signal>
        <signalID>angleOfAttack_d</signalID>
```

```
      <signalValue> 0.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
      <signalID>CmAlfa</signalID>
      <signalValue>0.01</signalValue>
      <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 2">
    <checkInputs>
      <signal>
      <signalID>angleOfAttack_d</signalID>

      <signalValue> 5.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
      <signalID>CmAlfa</signalID>
      <signalValue>0.04444</signalValue>
      <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 3">
    <checkInputs>
      <signal>
      <signalID>angleOfAttack_d</signalID>

      <signalValue>10.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
      <signalID>CmAlfa</signalID>
      <signalValue>-0.01111</signalValue>
      <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 4">
    <checkInputs>
      <signal>
      <signalID>angleOfAttack_d</signalID>

      <signalValue>15.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
      <signalID>CmAlfa</signalID>
      <signalValue>-0.06667</signalValue>
      <tol>0.00001</tol>
```

```
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 5">
    <checkInputs>
      <signal>
      <signalID>angleOfAttack_d</signalID>

      <signalValue>20.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
      <signalID>CmAlfa</signalID>
      <signalValue>-0.08</signalValue>
      <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 6">
    <checkInputs>
      <signal>
      <signalID>angleOfAttack_d</signalID>

      <signalValue>25.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
      <signalID>CmAlfa</signalID>
      <signalValue>-0.07</signalValue>
      <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 7">
    <checkInputs>
      <signal>
      <signalID>angleOfAttack_d</signalID>

      <signalValue>50.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
      <signalID>CmAlfa</signalID>
      <signalValue>-0.31429</signalValue>
      <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
 </checkData>
</DAVEfunc>
```

5.3.6 Shorter version

While the above seems incredibly long for a function with only 9 data points, keep in mind it also includes many instructional comments and optional, but very important information, such as units and where the data came from (provenance). Also, a very large complex function would only be expanded by the additional data points. The definitions and provenance information included with the function would probably not change much.

In the minimum, the same data can be represented as shown.

shorter_cma_example.dml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?> <!DOCTYPE DAVEfunc
PUBLIC "-//NASA//DTD for Flight Dynamic Models - Functions 2.0//EN"
"DAVEfunc.dtd"> <DAVEfunc>
     <fileHeader>
         <author name="Bruce Hildreth" org="SAIC"/>
         <fileCreationDate date="2006-03-18"/>
     </fileHeader>
     <variableDef name="Angle of attack" varID="angleOfAttack_d"
units=""/>
     <variableDef name="CMalpha" varID="CmAlfa" units=""/>
     <breakpointDef bpID="angleOfAttack_d_bp1">
         <bpVals> 0, 18, 19, 20, 22, 23, 25, 27, 90 </bpVals>
     </breakpointDef>
     <griddedTableDef gtID="CmAlfa_Table1">
         <breakpointRefs>
             <bpRef bpID="angleOfAttack_d_bp1"/>
         </breakpointRefs>
         <dataTable> 0.1,-0.1,-0.09, -.08, -0.05, -0.05, -0.07, -0.15, -0.6
</dataTable>
     </griddedTableDef>
     <function name="Cm_alpha_func">
         <independentVarRef varID="angleOfAttack_d"/>
         <dependentVarRef varID="CmAlfa"/>
         <functionDefn>
             <griddedTableRef gtID="CmAlfa_Table1"/>
         </functionDefn>
     </function>
</DAVEfunc>
```

5.3.7 Summary

The DAVE-ML embodiment of the standard truly enables nearly effortless transfer of simulation aerodynamics models between simulation facilities or architectures. The addition of the Math-ML allows the formulation of algebraic equations, aero or engine model coefficient buildup equations, for example, to be included as data in the model. DAVE-ML is also suitable for use of transfer of tabular functions and supporting algebraic equations for any type of data, not just simulation models.

While the above paragraphs explained the concepts implemented in DAVE-ML, Annex B is the authoritive normal for this standard. It provides much more detail and examples on how to

easily build a DAVE-ML compliant simulation. Annex C provides reference to the DAVE-ML web site that includes tools to facilitate using DAVE-ML based models in you particular simulation.

5.4 Future Work

The AIAA modeling and simulation Technical Committee plans to continue our efforts in facilitation of the exchange of simulations and models throughout the user community. Comments and suggestions on this expansion are welcomed on the simulation standards discussion group. Visit http://daveml.nasa.gov for submittal information.

Two tasks are of primary interest:

5.4.1 Time history information

The immediate task that is being pursued is the transfer of validation data between facilities. This is for the purpose of sending time response validation data when a model is exchanged.

The approach being taken is to adopt a flight test data standard. This has the advantage of using an existing standard and facilitating the use of flight test data to validate a simulation. Lockheed Martin has an existing internal standard that they have released for use by the community. It is implemented in hierarchal data format (HDF) and has been adopted by the JSF community and other programs. It is the Modeling and Simulation Technical Committees intent to adopt this for the transfer of simulation validation data. Some work will be required to define the data elements that are required for the validation of a simulation. This is expected to be a subset of the data elements that comprise flight test data.

5.4.2 Dynamic element specification

The addition of the specification of dynamics (e.g. continuous and discrete states) is being considered to expand the scope of the standard. This expansion would allow more of the domain of a flight vehicle model (flight controls as a good example) to be exchanged in a non-proprietary, facility-neutral way.

5.5 Final Summary

This is a standard for the purpose of facilitating the exchange of simulation models between users. This purpose cannot be emphasized enough. It is not meant to enforce any standard simulation architecture. DAVE-ML provides the mechanism for exchange of the modeling data and equations; the standard variables and axis systems provide a common language for the users to use to communicate. The standard is also valuable for documenting a model, since the names and axis system definitions are clearly documented for the user.

A model can be DAVE-ML compliant without using any standard names or axis systems, but the exchange of such a model between users will be more difficult, since clear definitions will have to be exchanged also.

It is the earnest desire of the authors of this standard that the user community will employ the current standard for aerodynamic models, continue to suggest improvements to the standard, and

develop tools to enhance the standard. Visit http://daveml.nasa.gov for information on how to be part of this effort and/or submit change or improvement recommendations.

ANNEXES

A.1 Normative Annex A: Standard Variable Names

A.1 Normative Annex B: Dynamics Aerospace Vehicle Exchange Markup Language (DAVE-ML) Reference, Version 2.0RC1

A.3 Informative Annex C: DAVE-ML Web Site

A.3 Informative Annex C: DAVE-ML Web Sites

The "official" DAVE-ML site is:

http://daveml.nasa.gov/

This link contains all DAV-ML documentation and links and information on DAVE-ML tools and applications.

Additional information is available at:

http://www.aiaa.org/