BSR/AIAA
S-119-200X

American National Standard

Flight Dynamics Model Exchange Standard

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. 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.

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

Foreword

This standard was sponsored and developed by the AIAA Modeling and Simulation Committee on Standards. Mr. Bruce Jackson of NASA Langley conceived DAVE-ML. DAVE-ML is the embodiment of the standard in XML. This document 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 **Modeling and Simulation CoS** were:

Bruce Hildreth, Chair	Science Applications International Corporation (SAIC)
Bruce Jackson, DAVE-ML Lead	NASA Langley Research Center
Geoff Brian	Defense Science Technical Organization (DSTO)
Brent York	Indra Systems, Inc.
Michael Silvestro	Charles Stark Draper Laboratory, Inc.
Barry Bryant	NASA Langley Research Center
Bimal Aponso	NASA Ames Research Center
Jean Slane	Engineering Systems Inc.
Peter Grant	University of Toronto
Bill Bezdek	Boeing Phantom Works
Jon Berndt	Jacobs

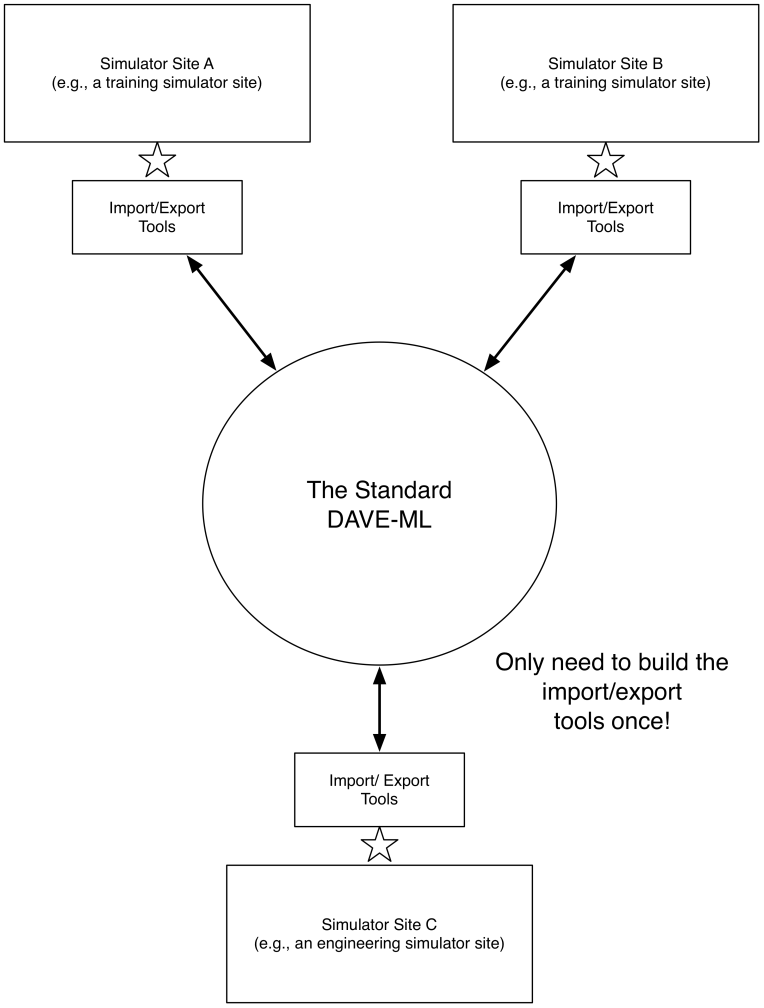
The above consensus body approved this document in Month 200X.

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

The AIAA Standards Procedures dictates that all approved Standards, Recommended Practices, and Guides are advisory only. Their use by anyone engaged in industry or trade is entirely voluntary. There is no agreement to adhere to any AIAA standards publication and no commitment to conform to or be guided by standards reports. In formulating, revising, and approving standards publications, the committees on standards will not consider patents that may apply to the subject matter. Prospective users of the publications are responsible for protecting themselves against liability for infringement of patents or copyright or both.

Introduction

The purpose of this standard is to clearly define the information and format required to exchange air vehicle simulation models between simulation facilities (see the figure below). 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”. The DAVE-ML standard can be used without using standard variable names, however it will 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, therefore each organization is required to design import/export tools which comply to the standard one time only.

Use of this standard will result in substantially reduced cost and time necessary to exchange aerospace simulations and model information. Test cases have indicated an order of magnitude reduction in effort to exchange simple models when utilizing this standard. Even greater benefits could 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 ®

MATLAB®

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. These tools can then be used to exchange models with any facility at minimal effort, rather than creating unique import/export tools for every exchange.

The standard includes definition of the information in order to clarify the data 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 systems is included. This list of standard variable names further simplifies the exchange of information, but is not required.

2 Tailoring

When viewed from the perspective of a specific program or project context, the requirements defined in this Standard may be tailored to match the actual requirements of the particular program or project. Tailoring of requirements shall be undertaken in consultation with the procuring authority where applicable.

NOTE Tailoring is a process by which individual requirements or specifications, standards, and related documents are evaluated and made applicable to a specific program or project by selection, and in some exceptional cases, modification and addition of requirements in the standards.

The following sections provide further guidance on specific tailoring situations.

2.1 Partial Use of the Standard

2.1.1 General

Each simulation created may not require the implementation of all aspects of this standard. The following guidelines are provided to encourage appropriate use of the standard in a number of example situations.

2.1.2 Creating a New Simulation Environment

This situation calls for use of the complete standard. In this situation it is hoped that the team developing this new simulation would add to the list of standard variables and axis systems.

2.1.3 Creating a New Simulation Model in an Existing Simulation Environment

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

In this case the following tailoring guidelines are applicable. 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 6). Therefore the standard axis systems can easily be referenced when documenting the simulation and interfaces between the new simulation components and those reused.

2.1.4 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 applicable.

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 as 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 becomes. The above tailoring guidelines may be categorized into two common situations; new and reused code.

New simulation code should

— use standard axis system definitions (Section 5) where they coincide with the definitions in the standard;

— use standard variable names (Section 6) to facilitate consistency and simplify documentation requirements;

— apply the convention for states, state derivatives and controls wherever possible; and

— use standard function tables (Section 7) for ALL function tables.

NOTE This facilitates consistency in the data, the documentation of the data, and collaboration with other organizations to improve or debug the data.

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

2.3 Creating New Variable Names and Axis Systems

The standard variable names and axis system definitions are included in 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 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 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.

This 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.

AIAA R-004-1992 *Atmospheric and Space Flight Vehicle Coordinate Systems*

4 Vocabulary

4.1 Acronyms and Abbreviated Terms

a/c	Aircraft
AIAA	American Institute of Aeronautics and Astronautics
ANSI	American National Standards Institute
cg	Center of gravity
DIS	Distributed Interactive Simulation
FE	Flat Earth axis system
GE	Geocentric Earth fixed axis system
SI	Système Internationale d'Unites
w.r.t	with respect to
XML	Extensible Markup Language

4.2 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

Confidence Interval
an estimate of the computed or perceived accuracy of the data

Dependent Variable
the output of a function table

EXAMPLE For $C_L(\alpha,\beta)$, C_L is the dependent variable, also called the output.

Independent Variable
the variable whose value determines the value of other variables

EXAMPLE For $C_L(\alpha,\beta)$, α and β are independent variables.

One Dimensional Table
a table containing only one independent variable

EXAMPLE $C_L(\alpha)$ is a one dimensional table.

Two Dimensional Table
a table containing two independent variables

EXAMPLE $C_L(\alpha,\beta)$ is a two dimensional table.

Static Equation
a mathematical statement where the output (left hand side) does not have direct dependence (right hand side) on a simulation state

Simulation States (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 in which the independent variable breakpoints do not change for different values of other independent variables

NOTE 1 This is sometimes called an orthogonal table.

NOTE 2 All one-dimensional tables are gridded tables.

Ungridded Table
a multi-dimensional function table in which the independent variable breakpoints change for different values of other independent variables

NOTE This is sometimes called a non-orthogonal table.

5 Standard Simulation Axis Systems

5.1 Background / Philosophy

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

Axis system standards are also reflected in the variable naming convention. When applicable, the axis system is included in the variable name (see Section 6).

5.1.1 Axis System Conventions

In general, ANSI/AIAA R-004-1992 should be referred to as the normative reference for axis system definitions. These axis systems are discussed in Table 1. However, it is important to emphasize the

correlation of the AIAA document and the *Distributed Interactive Simulation* (DIS) axis systems. The geocentric earth fixed axis system and body axis coordinate system axis system both are used in DIS and High Level Architecture (HLA) simulations.

5.1.1.1 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 as "World Coordinate System" in the DIS).

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 axis system is also frequently called "Earth Centered Earth Fixed".

5.1.1.2 Body Axis Coordinate System

Another standard axis system is the Body Axis System (axis system number 1.1.7 in ANSI/AIAA R-004-1992). This is identical to the DIS "Entity Coordinates System".

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

5.1.1.3 Additional Axis Systems

In addition to the axis systems defined in ANSI/AIAA R-004-1992, this standard has added the Flat Earth and Locally Level axis systems. These axis system is are defined only for convenience onfor use in simple simulations and for creating validation data.

The Flat Earth axis system 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".

The locally level axis system is also a simplified axis system convenient for simulation checkout and validation. The –Z axis passes through the vehicle CG. If a flat earth, the X axis is in the plane of the surface and oriented toward true North. The Y axis is also in the plane of the surface and completes the right hand triad (East). If not at flat earth, 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).

The locally level axis system is referred to in variable names as "LL".

5.1.1.4 Complete List of Axis Systems

The axis systems that are referenced are taken largely from paragraph 1.1 of ANSI/AIAA R-004-1992. The flat earth and locally level axis systems for atmospheric flight simulation approximation are added to that reference. Table 1 is the comprehensive list of axis systems that may be used.

The first column in Table 1 provides the abbreviation used for each axis system. The axis system may be referenced in a variable name. See Section 6 on the variable naming convention.

Table 1 — Standard axis systems

Reference Abbreviation for Variable Names	R-004-1992 Paragraph Number	Term	Definition	Symbol
EI (Earth centered inertial)	1.1.1	Geocentric inertial axis system (See Appendix D.2 of R-004 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 R-004)	$x_I y_I z_I$
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 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 to the north, and the y_G-axis completing the right-hand system. (See Appendix D.3 in R-004-1992)	$x_G y_G z_G$
	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 R-004-1992)	$x_o y_o z_o$ ($x_g y_g z_g$ is an acceptable alternative)
VO	1.1.5	Vehicle-carried orbit-defined axis system[a]	A system with the origin fixed in the vehicle, usually the center of mass, 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 R-004-1992)	$x_o y_o z_o$
VE	1.1.6	Vehicle-carried normal Earth axis system[a]	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.	$x_o y_o z_o$ ($x_g y_g z_g$ is an acceptable alternative)

Body	1.1.7	Body axis system[a]	A system fixed in the vehicle, with the origin, usually the center or mass, consisting of the following axes:	$x_By_Bz_B$
		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.[b] 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[a]	A system with the origin fixed in the vehicle, usually the center of mass, consistent of the following axes:	$x_wy_wz_w$
		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 $x_ay_az_a$.	z_w
SA (for stability axis system)	1.1.9	Intermediate axis system[a]	A system with the origin fixed in the vehicle, usually the center of mass, consisting of the following axes.	$x_sy_sz_s$

		x_e-axis	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.	x_s
		y_e-axis	An axis normal to the reference plane and positive to the right, coinciding with or parallel to the lateral axis (1.1.7).	y_s
		z_e-axis	An axis which coincides with or is parallel to the normal air-path axis so as to complete the orthogonal right-hand system.	z_s
FP	1.1.10	Flight-path axis system[a]	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[a] (USA practice: areoballistic axis system.)	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 (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.	$x_\Gamma y_\Gamma z_\Gamma$
FE		Flat Earth system (not from R-004-1992)	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 R-004-1992)	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).	$x_{LL} y_{LL\Gamma} z_{LL}$
[a] 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.

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

5.2 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.

5.3 References

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

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

6 Standard Simulation Variables

6.1 Background / Philosophy

6.1.1 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 information 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 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.

6.1.2 States and State Derivatives

Long-term 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 the identification of these variables is crucial in software maintenance.

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

By practice, anything in a simulation of interest is an output. To create an output, for example indicated airspeed, it is necessary to identify the states and inputs. Therefore, if the appropriate law of physics is known, the indicated airspeed may be correctly computed. Too often in simulation modeling these immutable fundamental concepts are forgotten. We “fudge” things and create states from outputs. Practically speaking, this is done because the states in the simulation cannot be determined. Since a simulation is an iterative process, it becomes unclear 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.

6.2 Variable Naming Convention

This clause will discuss the convention and philosophy used for naming simulation variables. This explanation is intended to ensure that new variables defined in the future are consistent with existing variables.

The mixed case variable name convention is used with one exception. The standard uses an underscore to separate the prefix and suffix from the body of the variable name. The standard could also be followed using underscores to separate the parts of the variable names.

The following general rules for naming variables shall be followed.

— Variables shall have meaningful names.

— Mnemonics shall not be used.

— Standard abbreviations are permitted.

— The first word in the variable name (not including the prefix, if any) shall start with a lower case. Distinct words thereafter in variable names shall be capitalized (for example, `angleofAttack_d`).

— Variable names shall not exceed 63 characters in length. Brief, but complete names are most effective.

— Abbreviations are generally all capitals.

6.3 Variable Name Creation Methodology

The suggested method of creating the name is as follows.

— Each name has up to eight components.

— All components are not required to be used because in many cases they do not apply.

— These components are:

1. (prefix)_

2. (variable domain)
3. (specific axis or reference)
4. (axis or reference system)
5. (core name)
6. Of (point on the vehicle)

   NOTE Generally only for positions, velocities and accelerations

7. WRT (reference point or frame)

   NOTE Generally only for positions, velocities and accelerations

8. _(units).

Very rarely, if ever, are all 8 components of a name used.

6.3.1 Prefix

The prefix is used to identify the most important dynamic variables in the simulation, the states and the state derivatives. (See Section 6.1.2)

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

6.3.1.1 Identification of States and State Derivatives

The states and state derivatives are those variables which make the simulation dynamic and are 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. 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.

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 identify them in the model implementation, making it easier to document and exchange the models.

Examples:

`s_XBodyVelocity_fs_1`	`s_` prefix indicates that this variable is a state
`sd_XBodyAcceleration_fs_2`	`sd_` prefix indicates that this variable is a state derivative

6.3.1.2 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.

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_avgAileronDeflection_d`	`c_` prefix indicates that this variable is a control
`c_pilotLongControlPos_r`	`c_` prefix indicates that this variable is a control

6.3.2 Variable Source Domain

This represents the domain in which the variable is calculated. In object oriented design, it could 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 include:

— Aero

— Engine or Thrust

— Controls

— Guidance

— Navigation

— GNC

— Wheel

— Landing Gear

— Hydraulic

— Electrical

— IO (for input/output)

— Motion

— CL or Control Loading

— Radar

— Weapons

— AIM9X (as an example, for the AIM-9X missile)

NOTE Users should add as many domains as needed to clearly identify the variable.

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

— `aeroXBodyForceCoefficient`

— `aeroXBodyForce_lbf`

— `thrustXBodyForce_lbf`

6.3.3 Specific Axis or Reference

This is the specific axis or reference used within the axis system (axis systems are defined in Section 5). If the axis system is included in the name, the specific axis or reference should also be included. For example

— (X, Y, Z), (N, E, D) or (U, V, W) 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.
`UBodyTurbulenceVelocity_fs_1`	where `Body` is the axis system and `U` is the specific axis in the body axis system, U indicating longitudinal translational motion.
`YGEVelocity_ms_1`	where `GE` is the axis system and `Y` is the specific axis, also indicating translational motion.
`ZRunway22VelocityOfLeftWheelWRTTD_fs_1`	where `Runway22` is the axis system (user defined) and `Z` is the specific axis, also indicating translational motion. `LeftWheel` is the point on the vehicle and `TD` (touchdown point) is the reference point.
`YBodyAccelOFPilotHead_ms_2`	where `Body` is the axis system and `Y` is the specific axis, also indicating translational motion. Design pilot head location is the point on the vehicle.

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) shall be used to avoid confusion.

Example 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)

6.3.4 Axis or Reference System

This is the axis or reference system to which the variable is referenced. Table 1 specifies the 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.

6.3.4.1 Conventions Used

Earth fixed frames and local reference frames by convention use X, Y, Z, Pitch, Roll, and Yaw for axis references. Local reference frame (FE for example) origin and orientation may be user defined. They are meant for runway, test range, target reference, navigational aid, etc. coordinate systems. Body fixed frames may use U, V, W, Pitch, Roll, and Yaw for axis references.

6.3.4.2 Variable Name Examples

The following variable names are provided as examples.

— `UBodyVelocity_fs_1` (or `XBodyVelocity_fs_1`, Body axis system)

— `s_XGEVelocity_fs_1` (in the case where the equations of motion are formulated such that the variable is a state, Geometric Earth axis system)

— `XGEVelocity_fs_1` (in the case where the equations of motion are formulated such that the variable is not a state)

— `UBodyVelocity_ms_1` (or `XBodyVelocity_ms_1`)

— `VBodyVelocity_fs_1` (or `YBodyVelocity_fs_1`)

— `S_XLLVelocity_fs_1` (Locally Level axis system)

— `S_XFEVelocity_fs_1` (Flat Earth axis system)

— `pitchBodyRate_rs_1` (or `YBodyAngularRate_rs_1`)

— `rollBodyRate_rs_1` (or `XBodyAngularRate_rs_1`)

— `yawBodyAccel_rs_2` (or `ZBodyAngularAccel_rs_1`)

Note that the standard encourages U, V, W, pitch, roll yaw for body frames in particular, since that is widely conventional. However, since the overall objective of the standard is to form a framework for clear communication between simulation facilities, the X, Y, Z convention is also acceptable. The appropriate core variable name shall be used to be clear whether the variable is a linear or angular variable.

6.3.5 Core Variable Name

This is the most specific (hence core) name for the variable. All variable names shall include this component of the name. Core variable name examples are as follows.

— `velocity`

— `rate`

— `accel`

— `forceCoefficient`

— `turbulenceVelocity`

— `angleOfAttack`

— `angleOfSideslip`

— `cosineOfAngleOfSideslip`

— `thrust`

— `torque`

— `aileronDeflection` (`aileron` could be considered a domain and `deflection` the core name)

The following variable names are provided as examples.

— `s_rollBodyRate_rs_1`

— `XBodyTurbulenceVelocity_fs_1`

— `ZGEVelocity_fs_1`

— `angleOfAttack_r`

— `angleOfSideslip_d`

— `cosineOfAngleOfSideslip`

— aileronDeflection_d

6.3.6 Reference Point or location on the vehicle

This component of the name is designed to clarify positions, velocities and accelerations and is normally omitted if the variable is not a position, velocity or acceleration. However, it may be used for any variable if desired. This component describes which point or object on the vehicle is being specified. "Of" is used to specify the point or object.

For those who prefer shorter variable names, the standard uses the convention that if the point or location on the vehicle is the center of mass (by convention, center of gravity, or CG) then the reference point may be omitted. However, use of "OfCG" is encouraged for clarity.

Reference points may be defined by the user and depend on the object the variable is describing.

Examples of reference points are as follows.

— OfCG (CG is the default, so "OfCG" may be omitted in any variable name)

— OfPilot

— OfIMU

— OfSensor

— OfMRC (for moment reference center)

— OfPilotEye (for the pilot eye point)

— OfRadAlt (for radar Altimeter)

— OfTerrain

The following variable names are provided as examples.

— UBodyVelocityWRTWind_fs_1 (OfCG understood)

— UBodyVelocityOfCGWRTWind_fs_1 (same meaning as above)

— UBodyVelocityWRTInertial_fs_1 (inertial velocity of the CG along the X body axis)

— heightOfCGWRTTerrain_f (CG may be omitted since it is the default)

— heightOfRadAltWRTTerrain_f (CG may be omitted since it is the default)

— heightOfTerrainWRTSurfaceReference_f

— XBodyPositionOfPilotEyeWRTCG_f

— longitudeRateOfIMUWRTWGS84_ds_1

— longitudeOfIMUWRTWGS84_d

— bodyAccelOfPilot_fs_2(3)

6.3.7 External Reference Frame or Reference Point on the Reference Frame

The external reference frame is generally used in conjunction with "reference point or location on the vehicle" above. It is primarily used in variables describing position, velocities and accelerations. This component defines the external reference frame which the motion is relative to. If the reference frame is rotating or the variable is describing angular motion, this component should define a specific point in the

reference frame. Stevens and Lewis (see Section 6.7) may be referred to for a more rigorous definition of "frames".

The standard uses the convention "WRT" to define the frame component of the variable name. For those who prefer shorter variable names, the inertial frame is default, and therefore while use of "WRTInertial" is encouraged it may be omitted. Some examples of reference frames are as follows.

— WRTInertial (WRTInertial is the default and may be omitted)

— WRTCG (this is commonly used to clarify definitions of positions)

— WRTMRC (moment reference center)

— WRTWGS84 (world geodetic system 84)

— WRTTD (ideal touchdown point)

— WRTImpact (the desired weapon impact point)

— WRTWind (the instantaneous wind velocity)

— WRTMeanSL

The following variable names are provided as examples.

— UBodyVelocityWRTWind_fs_1 (OfCG understood)

— UBodyVelocityOfCGWRTWind_fs_1 (same meaning as above)

— UBodyVelocityWRTInertial_fs_1 (inertial velocity of the CG along the X body axis. WRTInertial may be omitted since Inertial is the default reference frame)

— UBodyVelocity_fs_1 (inertial velocity of the CG along the X body axis, same meaning as above)

— bodyPositionOfPilotEyeWRTCG_f(3)

— longitudeOfIMUWRTWGS84_d

— longitudeOfCGWRTWGS84_d

— bodyPositionOfPilotEyeWRTCG_f(3)

— bodyPositionOfCGWRTMRC(3)

— ZRunway22VelocityOfLeftWheelWRTTD_fs_1

— heightOfRunwayWRTMeanSL_f

— UBodyVelocityWRTWind_fs_1

— totalVelocityWRTGround_fs_1

— GEVelocity_ms_1(3) (WRTInertial is omitted since inertial is the default)

6.3.8 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 self-documenting in this respect. Therefore, units shall be included in all variables except variables that are non-dimensional. Including units has the the added advantage of making this standard consistent and acceptable in countries utilizing the international system of units. For example, airspeed is just as acceptable as a standard both for the U.S. 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. 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_1lbf_1.

Further examples are as follows.

— `trueAirspeed_fs_1` for feet per second (f/s)

— `trueAirspeed_ms_1`for meters per second (m/s)

— `trueAirspeed_nmih_1` for knots (nautical miles per hour)

This standard defines what the variable name for airspeed is, the user defines the units being used. The suffix shall be separated from the body of the variable name by an underscore. The standard unit notations are given in Table 2, SI units and standard abbreviations are included.

Table 2 — Abbreviations used to designate units in standard variable names

Unit	Abbreviation
Time	
hour	h
second	s
minute	min
millisecond	ms
Length	
inch	inch
foot	f
meter	m
nautical mile	nmi
statute mile	smi
kilometer	km
centimeter	cm
millimeter	mm
Force	
pound force	lbf
Newton	N
kilogram force	kgf
Mass	
gram	g
kilogram	kg
pound mass	lbm
slug	slug

Unit	Abbreviation
Plane Angle	
degrees (angular)	d
radians	r
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	Jou
horsepower	Hp
Electrical	
volt direct current	vdc
volt alternating current	vac
ampere	A
cycles	cyc
watt	watt
henry	hy
farad	fd
ohm	ohm
Other	
candela (luminous intensity)	cd
mole (amt. of substance)	mol

6.4 Additional Discussion

Very rarely, if ever, are all 8 components of a name used. In the case of `s_rollBodyRate_rs_1` the following 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 “Of” and “WRT” were omitted because the variable is describing motion about (“Of”) the CG and it is relative to (“WRT”) the inertial frame of reference.

The intent is to provide clear communication when exchanging models, not to force the universal use of these variable names. s_rollBodyRate_rs_1 is intended to be a clear, brief, unambiguous name for the variable.

6.4.1 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

6.4.2 Discarded Conventions and Reasons

One convention considered was to have a prefix for simulation outputs as well as states and controls, but at the present time this has been discarded since the outputs required vary so widely, and there are typically an extremely large number of outputs. Practically speaking, every variable in the model including states, state derivatives and controls (inputs) could be considered an output.

Also considered was eliminating the suffix when the units were one of the standard set, but this concept was discarded since 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 should noticeably reduce the software maintenance effort after initial development when another software engineer is trying to understand the code to make bug fixes, offer enhancements, or reuse the code.

6.4.3 Relationship with Markup Grammar, DAVE-ML

At present, this variable naming convention is intended to be realized using DAVE-ML grammar of XML (see Section 7). In DAVE-ML, the state/state derivative designation and the units are identified in separate components from the variable names. Thus, including these in a variable name encoded in DAVE-ML would be redundant. .

The best practice is to strip these components (the prefix and suffix) from the variable name when encoding to DAVE-ML, and reinsert them into the variable name if code or model data is generated from the DAVE-ML. Following this convention will have two advantages.

1) Since the DAVE-ML grammar can be used with any variables, for those variables that do not conform to the naming convention and therefore do not have state/state derivative designation or units, DAVE-ML encourages the inclusion of this information which is critical to clear documentation of a model.

2) It allows XML processors to adopt the convention of automatically striping and adding the prefix and suffix to the variable names.

6.5 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 in Table 3 for illustrative purposes.

Interpretation of the standard variable name annex is best given by example. Table 3 presents the 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 – the short name is included to accommodate standard variable definitions in legacy compilers with name length restrictions

— One or more full names using the standard units convention — generally one full name with American convention units and one with SI units

NOTE Any suitable units may be used and no attempt is made to include all possible unites in Annex A.

— 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.

Table 3 — Standard variable name table example

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

6.6 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, indicated, or 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 for recommending clarification of existing names.

6.7 References

Stevens, Brian L., and Lewis, Frank L., **Aircraft Control and Simulation, Second Edition**. ISBN 978-0-471-37145-8, 2004, New York, J. Wiley and Sons, 2003, p. 3.

7 Standard Simulation Function Table Data Format and XML Implementation of the Standard: DAVE-ML

7.1 Purpose

This section explains 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 accessed 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.

7.2 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 simple 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.

7.3 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 numbers 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.

7.4 Standard Function Table Data — An Illustrative Example

Figure 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. This standard defines 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.

4) The valid ranges of the independent variables are different (ungridded table).

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

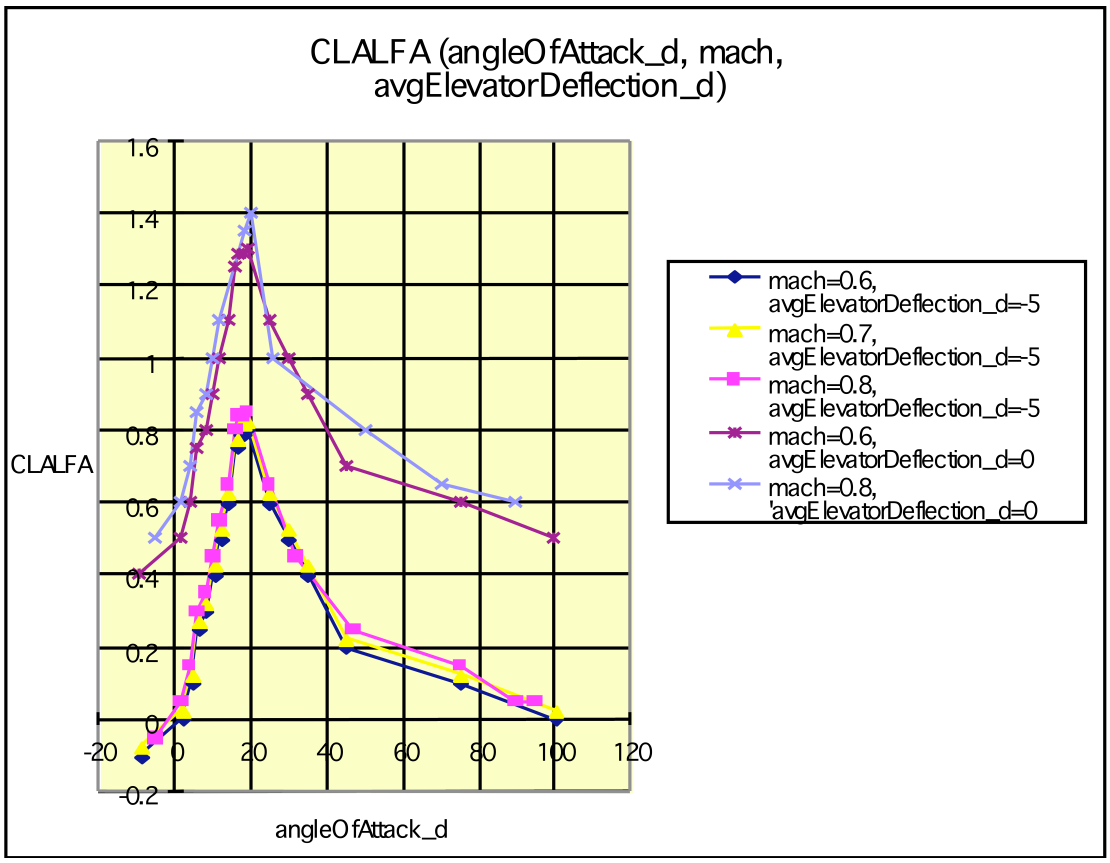


Figure 1 — An illustration of a 3 dimensional function table, CLALFA (`angleOfAttack_d`, `mach`, `avgElevatorDeflection_d`)

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 as follows.

— 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.

7.5 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.

7.6 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.

A simple one dimensional aero table is provided as an example, in this case pitching moment coefficient as a function of angle of attack, Table 4 and Figure 2.

Table 4 — 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

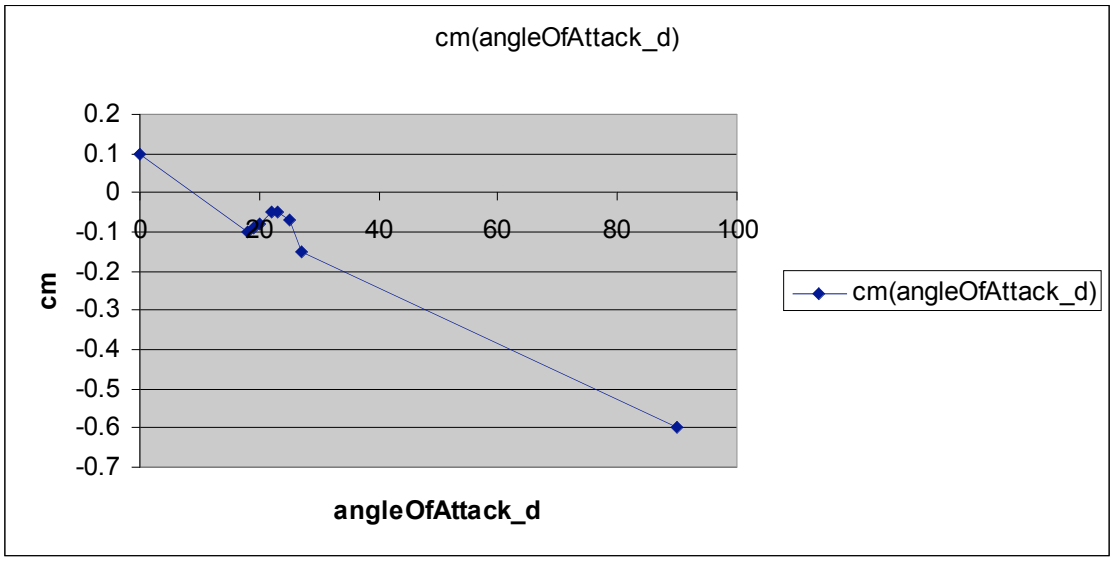


Figure 2 — The $C_m(\alpha)$ function — a simple one dimensional gridded function

The DAVE-ML implementation for this function could be as follows.

```
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>
      <varID>angleOfAttack_d</varID>
      <signalValue> 0.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.01</signalValue>
      <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 2">
    <checkInputs>
      <signal>
      <varID>angleOfAttack_d</varID>
      <signalValue> 5.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.04444</signalValue>
      <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 3">
    <checkInputs>
      <signal>
      <varID>angleOfAttack_d</varID>
      <signalValue>10.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
```

```
      <varID>CmAlfa</varID>
      <signalValue>-0.01111</signalValue>
      <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 4">
    <checkInputs>
      <signal>
      <varID>angleOfAttack_d</varID>
      <signalValue>15.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.06667</signalValue>
      <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 5">
    <checkInputs>
      <signal>
      <varID>angleOfAttack_d</varID>
      <signalValue>20.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.08</signalValue>
      <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 6">
    <checkInputs>
      <signal>
      <varID>angleOfAttack_d</varID>
      <signalValue>25.</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.07</signalValue>
      <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 7">
    <checkInputs>
      <signal>
      <varID>angleOfAttack_d</varID>
      <signalValue>50.</signalValue>
      </signal>
    </checkInputs>
```

```
      <checkOutputs>
        <signal>
        <varID>CmAlfa</varID>
        <signalValue>-0.31429</signalValue>
        <tol>0.00001</tol>
        </signal>
      </checkOutputs>
    </staticShot>
  </checkData>
</DAVEfunc>
```

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>
```

7.7 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 authoritative 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.

8 Future Work

The AIAA Modeling and Simulation Technical Committee plans to continue its 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. The following sections describe the two tasks of primary interest.

8.1 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.

8.2 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.

9 Conclusion

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 to facilitate effective communication. 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.

Annex A Standard Variable Names (Normative)

A.1 General

The table in this annex is meant to contain simulation variables that are independent of the particular vehicle type being simulated. These variables are tailored towards aircraft simulation. Visit http://DaveML.nasa.gov to suggest additional variables or changes to the existing list

A.2 Table Explanation

Interpretation of the standard variable name table is best given by example. In general the table has 7 columns. These are described below using the `rollEulerAngle` as an example:

1) The symbol for that variable, Φ

2) The short name, PHI

3) One of more full names using the standard units conventions — generally, one full name with American convention units and one with SI units.

   NOTE Any suitable units may be used. In the example for `rollEulerAngle` both the `_d` for degrees and the `_r` for radians are given. The "Full Variable Name" column does not necessarily provide all acceptable units for each variable.

4) A description of the variable, if applicable should always specify the axis system.

5) The POSITIVE sign convention of the variable — RWD indicates that positive `rollEulerAngle` is right wing down

6) Minimum value, normally only specified for angles

7) Maximum values of the variable, normally only specified for angles

This example also illustrates the pitch and yaw Euler angles.

Some variables may be used to represent variables referenced to more than one axis system. In this case the axis system is specified as xx and any axis system reference (refer to the body of this standard) may be substituted for the xx. For example, `Yxx`Velocity_fs_1 may represent:

— `YEIVelocity_fs_1` for the EI axis system - Earth centered Inertial (also know as geocentric inertial) axis system

— `YECEFVelocity_fs_1` for the ECEF axis system - Earth centered Earth Fixed (also known as Geocentric Earth [GE] axis system, `YGEVelocity_fs_1` is the same as `YECEFVelocity_fs_1`)

— `YVOVelocity_fs_1` for the VO axis system - Vehicle carried, Orbit defined axis system

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.

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

The variable name table below does not specify which variables are states, state derivatives, inputs or initial conditions. These specifications may be added to any appropriate variable. See the body of this standard.

A.3 Standard Variable Name Tables

Table A.1 — Vehicle Positions and Angles

Symbol	Short Name	Full Variable Name	Description	Sign Convention
$\underline{\varepsilon}$	EUL	`eulerAngle_d[3]` `eulerAngle_r[3]`	Vector of the roll, pitch, and yaw Euler angles frame.	
Φ	PHI	`rollEulerAngle_d` `rollEulerAngle_r`	Roll Euler Angle, LL frame.	RWD
θ	THET	`pitchEulerAngle_d` `pitchEulerAngle_r`	Pitch Euler Angle, LL frame	ANU
ψ	PSI	`yawEulerAngle_d` `yawEulerAngle_r`	Yaw Euler Angle, LL frame	ANR
$\sin \Phi$	SPHI	`rollEulerAngleSine`	Sine Of Euler Roll Angle	RWD
$\cos \Phi$	CPHI	`rollEulerAngleCosine`	Cosine Of Euler Roll Angle	RWD
$\sin \theta$	STHT	`pitchEulerAngleSine`	Sine Of Euler Pitch Angle	ANU
$\cos \theta$	CTHT	`pitchEulerAngleCosine`	Cosine Of Euler Pitch Angle	ANU
$\sin \psi$	SPSI	`yawEulerAngleSine`	Sine Of Euler Yaw Angle	ANR
$\cos \psi$	CPSI	`yawEulerAngleCosine`	Cosine Of Euler Yaw Angle	ANR

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Intial Value	Min Value	Max Value
$\underline{T}_{FE/B}$		FEToBodyT[3,3]	The FE to Body transformation matrix composed of the elements defined below				
$T_{FE/B}(1,1)$	T11	FEToBodyT11	CTHT*CPSI (FE To B) axis transformation element				
$T_{FE/B}(2,1)$	T21	FEToBodyT21	SPHI*STHT*CPSI - CPHI*SPSI (FE To B) axis transformation element				
$T_{FE/B}(3,1)$	T31	FEToBodyT31	CPHI*STHT*CPSI + SPHI*SPSI (FE to B) axis transformation element				
$T_{FE/B}(1,2)$	T12	FEToBodyT12	CTHT*SPSI (FE to B) axis transformation element				
$T_{FE/B}(2,2)$	T22	FEToBodyT22	SPHI*STHT*SPSI + CPHI*CPSI (FE to B) axis transformation element				
$T_{FE/B}(3,2)$	T32	FEToBodyT32	CPHI*STHT*SPSI - SPHI*CPSI (FE to B) axis transformation element				
$T_{FE/B}(1,3)$	T13	FEToBodyT13	-STHT (FE to B) axis transformation element				

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Intial Value	Min Value	Max Value
$T_{FE/B}(2,3)$	T23	FEToBodyT23	SPHI*CTHT (FE to B) axis transformation element				
$T_{FE/B}(3,3)$	T33	FEToBodyT33	CPHI*CTHT (FE to B) axis transformation element				
γ_V	GAMV	flightPathAngle_r flightPathAngle_d	Flight Path Angle Above Horizon	ANU		-p/2 -90	p/2 90
γ_H	GAMH	flightPathAzimuth_r flightPathAzimuth_d	Flight Path Angle In Horizon Plane, from North	CWFN		-p -180	p 180
h	ALT	altitudeMSL_f altitudeMSL_m	Geometric altitude of vehicle altimeter above Mean Sea Level	UP			
	XLON	longitudeWRTzzz_r longitudeWRTzzz_d	Longitude of Vehicle CG with respect to the zzz reference frame.	WEST			
	XLAT	latitudeWRTzzz_r latitudeWRTzzz_d	Latitude of Vehicle CG with respect to the zzz reference frame.	NORTH			
	XLONIMU	longitudeOfIMUWRTzzz_r longitudeOfIMUWRTzzz_d	Longitude of Vehicle IMU with respect to the zzz reference frame.	NORTH			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Intial Value	Min Value	Max Value
	XLATIMU	latitudeOfIMUWRTzzz_r latitudeOfIMUWRTzzz_d	Latitude of Vehicle IMU with respect to the zzz reference frame.	NORTH			
EXAMPLE							
		longitudeOfIMUWRTWGS84_d latitudeOfIMUWRTWGS84_d	Longitude and latitude of the vehicle IMU in the World Grid System 1984 reference frame				
	HGT_RWY	runwayHeightAboveSL_ft runwayHeightAboveSL_m	Height Of Runway w.r.t. mean Sea Level	Above			
		General Definition xxPositionOfyyyWRTzzz_f[3] xxPositionOfyyyWRTzzz_m[3] For Example: xxPosition_f[3] is the same as xxPositionOfCG_f[3]	General Definition Vector of positions of yyy with respect to zzz (a user defined reference point or frame) in the xx axis system. The lengths of xx, yyy, zzz are not restricted to 2 and 3 characters respectively. The axis system, xx, must always be defined. If the yyy is not defined the definition defaults to the vehicle cg. If the zzz is not defined the reference point defaults to the origin of the axis system. Comprised of the three components as defined below.				
	XCG	XxxPositionOfyyyWRTzzz_f XxxPositionOfyyyWRTzzz_m or XxxPosition_f	X position of yyy with respect to zzz (a user defined reference point) in the xx axis system. Defaults to th CG and origin of the axis system.	(yyy –zzz)			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Intial Value	Min Value	Max Value
	YCG	YxxPositionOfyyyWRTzzz_f YxxPositionOfyyyWRTzzz_m or YxxPosition_f	Y position of yyy with respect to zzz (a user defined reference point) in the xx axis system. Defaults to th CG and origin of the axis system.	(yyy –zzz)			
	ZCG	ZxxPositionOfyyyWRTzzz_f ZxxPositionOfyyyWRTzzz_m or ZXxxPosition_f	Z position of yyy with respect to zzz (a user defined reference point) in the xx axis system. Defaults to the CG and origin of the axis system.	(yyy –zzz)			
		General Definition xxPositionOfMRCWRTzzz_f[3] xxPositionOfMRCWRTzzz_m[3] Example xxPositionOfMRC_f[3]	General Definition Vector of positions of the moment reference center (MRC) with respect to zzz (a user defined reference point) in the xx axis system. The lengths of xx, yyy, zzz are not restricted to 2 and 3 characters respectively. The moment reference center is sometimes more convenient to locate a vehicle since the moment reference center is fixed in the vehicle, but the CG moves. zzz may be defaulted to the origin of the axis system. Comprised of the three components as defined below.				
	XREF	XxxPositionOfMRCWRTzzz_f XxxPositionOfMRCWRTzzz_m	X position of the moment reference center (MRC) with respect to zzz in the xx axis system.	XxxPositionOfMRCWRTzzz_f XxxPositionOfMRCWRTzzz_m			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Intial Value	Min Value	Max Value
	YREF	YxxPositionOfMRCWRTzzz_f YxxPositionOfMRCWRTzzz_m	Y position of the moment reference center (MRC) with respect to zzz in the xx axis system.	YxxPositionOfMRCWRTzzz_f YxxPositionOfMRCWRTzzz_m			
	ZREF	ZxxPositionOfMRCWRTzzz_f ZxxPositionOfMRCWRTzzz_m	Z position of the moment reference center (MRC) with respect to zzz in the xx axis system.	ZxxPositionOfMRCWRTzzz_f ZxxPositionOfMRCWRTzzz_m			
		bodyPositionOfPilotEyeWRTCG_f[3] bodyPositionOfPilotEyeWRTCG_f[3]	Vector of positions of the pilot's eye with respect to the CG in the body axis system. Comprised of the three components as defined below.				
	XPLT2CG	XBodyPositionOfPilotEyeWRTCG_f XBodyPositionOfPilotEyeWRTCG_f	X position of pilot eye point w.r.t. CG, in the body axis system	Eye FWD of CG			
	YPLT2CG	YBodyPositionOfPilotEyeWRTCG_f YBodyPositionOfPilotEyeWRTCG_f	Y position of pilot eye point w.r.t. CG, in the body axis system	Eye Right of the CG			
	ZPLT2CG	ZbodyPositionOfPilotEyeWRTCG_f ZbodyPositionOfPilotEyeWRTCG_f	Z position of pilot eye point w.r.t. CG, in the body axis system	Eye below CG			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Intial Value	Min Value	Max Value
EXAMPLE							

		`Runway22Position_f[3]`
		indicates position of the CG with respect to the origin of the Runway22 axis system
		`Runway22PositionOfFwdLeftMainWheelWRTTD_f[3]`
		indicates position of the forward left main wheel with respect to the touchdown point in the Runway 22 axis system
		NOTE All are user defined
			Vector of positions of the vehicle CG relative to the Runway 22 (a user defined axis system) touchdown reference point. Comprised of the three components as defined below.				

	XCGTD	`XRunway22PositionOfCGWRTTD _f` `XRunway22PositionOfCGWRTTDD_m`	CG X-position w.r.t. Runway touchdown point in the specified (Runway22) axis system.	CG Down the runway from the reference point			
	YCGTD	`YRunway22PositionOfCGWRTTD _f` `YRunway22PositionOfCGWRTTD_m`	CG Y-position w.r.t. Runway touchdown point in the specified (Runway22) axis system.	CG to the right of the reference point			
	ZCGTD	`ZRunway22PositionOfCGWRTTD _f` `ZRunway22PositionOfCGWRTTD _m`	CG Z-position w.r.t. Runway touchdown point in the specified (Runway22) axis system(this variable is normally negative)	CG below the TD point			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Intial Value	Min Value	Max Value
	RE	smoothEarthRadius_f smoothEarthRadius_m	Radius of Earth (center to smooth surface which is mean sea level), round earth model or oblate spheroid under the aircraft.				
	RALT	heightOfCGWRTTerrain_f heightOfCGWRTTerrain _m	Height of the aircraft CG above the terrain	NSG			
	HTERRAIN	heightOfTerrainWRTSurfaceReference_f heightOfTerrainWRTSurfaceReference_m	Height of the terrain under the A/C CG. It is the terrain height above the smooth surface of of the earth, regardless whether a flat, round or oblate spheroid model is used.				

Table A.2 — Vehicle velocities and angular rates

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
$\underline{\omega}_B$	OMB	bodyAngularRate_rs_1[3] bodyAngularRate_ds_1[3]	Vector of body axis angular rates comprised of the three components as defined below. Motion is always wrt the inertial frame unless otherwise specified.				
p_B	PB	rollBodyRate_rs_1 rollBodyRate_ds_1	Vehicle roll velocity, body axis system	RWD			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
q_B	QB	pitchBodyRate_rs_1 pitchBodyRate_ds_1	Vehicle pitch velocity, body axis system	ANU			
r_B	RB	yawBodyRate_rs_1 yawBodyRate_ds_1	Vehicle yaw velocity, body axis system	ANR			
$\underline{\dot{\varepsilon}}$	EULD	eulerAngleRate_ds_1[3] eulerAngleRate_rs_1[3]	Vector of the roll, pitch, and yaw Euler angle rates defined below. LL (locally level) axis system				
$\dot{\phi}$	PHID	rollEulerAngleRate_rs_1	Euler roll rate, LL axis system	RWD			
$\dot{\theta}$	THETD	pitchEulerAngleRate_rs_1	Euler pitch rate, LL axis system	ANU			
$\dot{\psi}$	PSID	yawEulerAngleRate_rs_1	Euler yaw rate, LL axis system	ANR			
		General Definition XxxVelocityOfyyyWRTzzz_fs_1 XxxVelocityOfyyyWRTzzz_ms_1 YxxVelocityOfyyyWRTzzz_fs_1 YxxVelocityOfyyyWRTzzz_ms_1 ZxxVelocityOfyyyWRTzzz_fs_1 ZxxVelocityOfyyyWRTzzz_ms_1	General expression for velocities along the X, Y and Z axes of the xx coordinate system. yyy indicates the reference point on the vehicle and the Ofyyy may be omitted if it is the CG. zzz represents the frame that the vehicle is moving with repect to and the WRTzzz may be omitted if it is the inertial frame. So XFEVelocity_fs_1 is the inertial velocity of the vehicle CG along the X axis of the FE coordinate system and is the short version of XFEVelocityOfCGWRTInetial_fs_1.				
$\underline{V}_B$	VELB	bodyVelocityWRTWind_fs_1[3] bodyVelocityWRTWind_ms_1[3] can also be expressed as: bodyVelocityOfCGWRTWind_fs_1[3]	Vector of body axis velocities of the cg with respect to the instantaneous wind comprised of the three components as defined below.				
u_B	*UB*	UBodyVelocityWRTWind_fs_1 UBodyVelocityWRTWind_ms_1	X-velocity Body axis system.	*FWD*			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
v_B	VB	`VBodyVelocityWRTWind_fs_1` `VBodyVelocityWRTWind_ms_1`	Y-velocity Body axis system	RT			
w_B	WB	`WBodyVelocityWRTWind_fs_1` `WBodyVelocityWRTWind_ms_1`	Z-velocity Body axis system	DWN			
$\underline{V}_{FE}$	VELFE	`FEVelocity_fs_1(3)` `FEVelocity_ms_1(3)`	Vector of Flat Earth (FE) axis translational velocities comprised of the three components as defined below.				
V_N	VNFE	`NfeVelocity_fs_1` `NfeVelocity_ms_1`	Northward Velocity Over Flat Earth (FE) axis system [flat, non-rotating earth]	NORTH			
v_E	VEFE	`EfeVelocity_fs_1` `EfeVelocity_ms_1`	Eastward Velocity Over Flat Earth (FE) axis system [flat, non-rotating earth]	EAST			
V_D	VDFE	`DfeVelocity_fs_1` `DfeVelocity_ms_1`	Downward Velocity Toward Earth Ctr,.(FE) axis system [flat, non-rotating earth]	DOWN			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
$\underline{V}_{GE}$	VELxx	xxVelocity_fs_1(3) xxVelocity_ms_1(3)	Vector of aircraft cg translational velocities wrt the specified (xx) axis system comprised of the three components as defined below.				
V_N	VNxx	NxxVelocity_fs_1 NxxVelocity_ms_1	Northward Velocity Over specified (xx)Earth Fixed Axis System	NORTH			
V_E	VExx	ExxVelocity_fs_1 ExxVelocity_ms_1	Eastward Velocity Over specified (xx)Earth Fixed Axis System	EAST			
V_D	VDxx	DxxVelocity_fs_1 DxxVelocity_ms_1	Downward Velocity Over specified (xx)Earth Fixed Axis System	DOWN			
EXAMPLES							
		EGEVelocity_fs_1	Eastward (Y axis) velocity over the earth in the geocentric earth (GE) axis system in ft/sec	East			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
		NEFVelocity_kms_1	Northward (X axis) velocity over the earth in the earth centered earth fixed (EF) axis system in kilometers/sec	North			
		UBodyVelocity_fs_1	X axis velocity in the Body axis system in ft/sec	Forward			
		ZRunway22Velocity_fs_1	Z axis velocity in the user defined “runway22” coordinate system in f/s	Down			
$V_{T_{XX}}$	VTxx	xxTotalVelocity_fs_1 xxTotalVelocity_ms_1	Total Velocity where xx is the reference frame as defined in the body of this standard.	Forward			
$V_{G_{XX}}$	VGxx	xxGroundSpeed_fs_1 xxGroundSpeed_ms_1	Vehicle velocity relative to the ground, where xx is the reference frame as defined in the body of this standard.	Forward			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
M_N	XMACH	mach	Mach number of the aircraft	Forward			
$V_{RW_{XX}}$	VELRWxx	xxVelocityRelativeToWind_fs_1(3) xxVelocityRelativeToWind_ms_1(3)	Vector of fixed xx axis translational velocities wrt the specified (xx) axis system comprised of the three components as defined below.				
V_{NRW}	VNRWxx	xxVelocityXRelativeToWind_fs_1 xxVelocityXRelativeToWind_ms_1	North Relative Velocity Vn-vnw in the xx frame.	NORTH			
V_{ERW}	VERWxx	xxVelocitxxRelativeToWind_fs_1 xxVelocitxxRelativeToWind_ms_1	East Relative Velocity Ve-vew in the xx frame.	EAST			
V_{DRW}	VDRWxx	xxVelocityZRelativeToWind_fs_1 xxVelocityZRelativeToWind_ms_1	Down Relative Velocity Vd-vdw in the xx frame.	DOWN			
$\dot{h}_{XX}$	ALTDxx	xxAltitudeRate_fs_1 xxAltitudeRate_ms_1	Altitude time rate of change in xx frame.	DOWN			
	XLOND	xxLongitudeRate_rs_1 xxLongitudeRate_ds_1	Longitude Rate Of Change in xxfrane.	WEST			
	XLATD	xxlatitudeRate_rs_1 xxlatitudeRate_ds_1	Latitude Rate Of Change in xx frame.	NORTH			
p_s	PS	rollVFRate_rs_1 rollVFRate_ds_1	Roll about the X axis in the VF reference frame, also know as stability axis roll rate.	RWD			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
r_s	RS	`yawVFRate _rs_1` `yawVFRate_ds_1`	Yaw about the Z axis in the VF reference frame, also known as the Stability Axis yaw rate	ANR			

Table A.3 — Vehicle linear and angular accelerations

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
$\underline{\dot{\omega}}_B$	OMBD	`bodyAngularAccel_rs_2(3)` `bodyAngularAccel_ds_2(3)`	Vector of body axis angular accelerations comprised of the three components as defined below.				
$\dot{p}_B$	PBD	`rollBodyAccel_rs_2` `rollBodyAccel_ds_2`	Aircraft Roll Acceleration, Body frame	RWD			
$\dot{q}_B$	QBD	`pitchBodyAccel_rs_2` `pitchBodyAccel_ds_2`	Aircraft Pitch Accel, Body frame	ANU			
$\dot{r}_B$	RBD	`yawBodyAccel_rs_2` `yawBodyAccel_ds_2`	Aircraft Yaw Acceleration, Body frame	ANR			
		`bodyAccel_fs_2(3)` `bodyAccel_ms_2(3)`	Vector of accelerations of the cg of the a/c wrt the interital frame in the body axis system. Therefore does not include the gravity vector. Comprised of the three components as defined below.				

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
u_B	UBD or UBD	UbodyAccel_fs_2 UbodyAccel_ms_2	Lonngitudinal acceleration (along the X-body axis)	FWD			
v_B	VBD or VBD	VBodyaccel_fs_2 VBodyaccel_ms_2	Right Sideward Acceleration, Body axis	RT			
w_B	WBD or WBD	WBodyaccel_fs_2 WBodyaccel_ms_2	Downward Acceleration, Body axis	DOWN			
$\dot{V}_{T_{XX}}$	VTDxx	xxTotalAccel_fs_2 xxTotalAccel_ms_2	Rate of change of inertial velocity, where xx is the reference frame as defined in the body of this standard.	Forward			
		xxAccel_fs_2 xxAccel_ms_2	Vector of aircraft cg translational wrt the specified (xx) axis system comprised of the three components as defined below.				
V_N	VND	NxxAccel_fs_2 NxxAccel_ms_2	North Acceleration Over Earth	NORTH			
V_E	VED	ExxAccel_fs_2 ExxAccel_ms_2	East Acceleration Over Earth	EAST			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
V_D	VDD	DxxZAccel_fs_2 DxxAccel_ms_2	Down Acceleration Toward Earth surface or center	DOWN			
		bodyCgAccelSensed_fs_2(3) bodyCgAccelSensed_ms_2(3)	Vector of accelerations sensed at the cg (including the effects of the gravity vector) in the body axis system. Comprised of the three components as defined below.				
	AX	XBodyCgAccelSensed_fs_2 XBodyCgAccelSensed_ms_2	X Acceleration Of A/c C.g. (body axis) Includes the gravity vector.	FWD			
	AY	YBodyCgAccelSensed_fs_2 YBodyCgAccelSensed_ms_2	Y Acceleration Of A/c C.g. (body axis) Includes the gravity vector.	RT			
	AZ	ZBodyCgAccelSensed_fs_2 ZBodyCgAccelSensed_ms_2	Z Acceleration Of A/c C.g. (body axis) Includes the gravity vector.	DOWN			
		bodyPilotAccel_fs_2(3) bodyPilotAccel_ms_2 (3)	Vector of accelerations at the pilot reference point, in the body axis system, comprised of the three components as defined below.				

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
	AXP	`XBodyPilotAccel_fs_2` `XBodyPilotAccel_ms_2`	X Acceleration Of Pilot reference point (body axis)	FWD			
	AYP	`YBodyPilotAccel_fs_2` `YBodyPilotAccel_ms_2`	Y Acceleration Of Pilot reference point(body axis)	RT			
	AZP	`ZBodyPilotAccel_fs_2` `ZBodyPilotAccel_ms_2`	Z Acceleration Of Pilot reference point(body axis)	DOWN			
	G	`localGravity_fs_2` `localGravity_fs_2`	Acceleration Due To Gravity (at the vehicle altitude)	DOWN			

Table A.4 — Vehicle air data

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
α	ALFA	`angleOfAttack_d` `angleOfAttack_r`	Angle Of Attack, Body axis	ANU	From aircraft trim	$-\pi$,-180	$+\pi$, +180
β	BETA	`angleOfSideslip_d` `angleOfSideslip_r`	Sideslip Angle, Body axis	ANL	From aircraft trim	$-\pi$,-180	$+\pi$, +180
$\dot{\alpha}$	ALFD	`angleOfAttackRate_rs_1`	Angle Of Attack Rate, Body axis	ANU	From aircraft trim		

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
$\dot{\beta}$	BETD	angleOfSideslipRate_rs_1	Sideslip Angle Rate	ANL	From aircraft trim		
$\sin\alpha$	SALPH	sineAngleOfAttack	Sine Of Angle Of Attack	ANU		-1.0	1.0
$\cos\alpha$	CALPH	cosineAngleOfAttack	Cosine Of Angle Of Attack	ANU		-1.0	1.0
$\sin\beta$	SBETA	sineAngleOfSideslip	Sine Of Sideslip Angle	ANL		-1.0	1.0
$\cos\beta$	CBETA	cosineAngleOfSideslip	Cosine Of Sideslip Angle	ANL		-1.0	1.0
V_{CAL}	VCAL	calibratedAirspeed_nmih_1	Calibrated Air Speed, knots	FWD			
V_{EQ}	VEQ	equivalentAirspeed_nmih_1	Equivalent Air Speed	FWD			
V_{IND}	VCAL	indicatedAirspeed_nmih_1	Calibrated Air Speed,	FWD			
V_{RW}	VRW	trueAirspeed_fs_1 trueAirspeed_ms_1 trueAirspeed_ nmih_1	Vehicle Velocity relative to the local wind (true airspeed)	FWD			
$\overline{q}$	QBAR	dynamicPressure_lbff_2 dynamicPressure_Nm_2	Dynamic Pressure	NSC			
$\overline{q}_c$	QBARC	impactPressure_lbff_2 impactPressure_Nm_2	Inpact Pressure	NSC			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
ρ	RHO	airDensity_lbmf_3 airDensity_kgpm_3	Air Density, At Altitude of the aircraft	NSC			
	DENALT	densityAltitude_f densityAltitude_f	Density altitude				
a	SOUND	speedOfSound_fs_2 speedOfSound_ms_2	Velocity Of Sound At Altitude of the aircraft	NSC			
T_{TOT_R}	TR	totalTempRatio_C totalTempRatio_K	Total Temperature Ratio	NSC			
P_{TOT_R}	PR	totalPressureRatio_C totalPressureRatio_K	Total Pressure Ratio	NSC			
T_{AMB}	TAMB	ambientTemperature_C ambientTemperature_K	Ambient Temperature at altitude	NSC			
P_{AMB}	PAMB	ambientPressure_lbff_2 ambientPressure_Nm_2	Ambient Pressure at altitude	NSC			
P_{AMB_R}	PAMBR	ambientPressureRatio	Ratio Of ambient pressure at altitude to sea level ambient pressure	NSC			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
T_{AMB_R}	TAMBR	ambientTemperatureRatio	Ratio Of ambient temperature at altitude to sea level ambient temp.	NSC			
T_{TOT}	TTOT	totalTemp_C totalTemp_K	Total Temperature at altitude	NSC			
P_{TOT}	PTOT	totalPressure_lbff_2 totalPressure_Nm_2	Total Pressure at altitude	NSC			
	TAMB_R	ambientTemperatureAtAlt_K ambientTemperatureAtAlt_R ambientTemperatureAtAlt_C	Ambient temperature, at the altitude of the CG				
	TTOT_R	totalTemperatureAtAlt_K totalTemperatureAtAlt_R totalTemperatureAtAlt_C	Total temperature at the altitude of the CG				
	ALT_SET	InstrumentAltimeterSetting_inchMercury	Cockpit Altimeter setting (Kohlsman window)	29.92 is standard day			
	P_ALT	PressureAltitude_f PressureAltitude_m	Pressure altitude at the CG				
	RHO_SL	seaLevelAirDensity_lbfpf3	Air density at sea level				

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value

	TAMB_SL	`seaLevelAmbientTemp_K`
		`seaLevelAmbientTemp_R`
		`seaLevelAmbientTemp_C`
			Ambient temperature at mean sea level				

	PAMB_SL	`seaLevelAmbientPressure_lbff2` `seaLevelAmbientPressure_Nm2`	Ambient pressure at sea level				

Table A.5 — Atmospheric disturbances and turbulence

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
	WIND_SPEED	`steadyStateWindVelocity_fs_1` `steadyStateWindVelocity_ms_1`	Total velocity of steady wind				
	WIND_DIRECTION	`steadyStateWindDirection_d`	Steady wind heading (blowing FROM true North)	Wind blowing from			
$\underline{V}_{B_{Turb}}$	VELBT	`bodyTurbulenceVelocity_fs_1(3)` `bodyTurbulenceVelocity_ms_1(3)`	Vector of body axis translational turbulence velocities comprised of the three components as defined below.				
$u_{B_{Turb}}$	UBTURB	`UbodyTurbulenceVelocity_fs_1` `UbodyTurbulenceVelocity_ms_1`	X-velocity Turb. Component, Body axis	FWD			
$v_{B_{Turb}}$	VBTURB	`VbodyTurbulenceVelocity_fs_1` `VbodyTurbulenceVelocity_ms_1`	Y-velocity Turb. Component, Bodyaxis	RT			
$w_{B_{Turb}}$	WBTURB	`WbodyTurbulenceVelocity_fs_1` `WbodyTurbulenceVelocity_ms_1`	Z-velocity Turb. Component, Body axis	DWN			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
$\underline{V}_{W_{XX}}$	VWxx	xxWindVelocity_fs_1(3) xxWindVelocity_ms_1(3)	Vector of fixed xx frame wind velocities velocities wrt the specified (xx) axis system comprised of the three components as defined below.				
W_N	VNWxx	XxxWindVelocity_fs_1 XxxWindVelocity_ms_1	North component of wind velocity in xx frame	To the North			
W_E	VEWxx	YxxWindVelocity_fs_1 YxxWindVelocity_ms_1	East component Of wind velocity in xx frame.	To the East			
W_D	VDWxx	ZxxWindVelocity_fs_1 ZxxWindVelocity_ms_1	Down Component Of Wind Velocity in xx frame.	To Downward			
$W_{T_{XX}}$	VTWxx	xxTotalwindVelocity_fs_1 xxTotalwindVelocity_ms_1	Total Wind Velocity, in xx frame.	NSC			
		netWindVel_fs_1 (3) netWindVel_ms_1 (3)	Vector of the net wind velocities impinging on the aircraft. Comprised of the three components as defined below.				
	VTWN	netWindVelFromNorth_fs_1 netWindVelFromNorth_ms_1	Net wind velocity from North. Net wind is the steady state winds plus any turbulences and shears.	From the North			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
	VTWE	`netWindVelFromEast_fs_1` `netWindVelFromEast_ms_1`	Net wind velocity from East. Net wind is the steady state winds plus any turbulences and shears.	From the East			
	VTWD	`netWindVelFromBelow_fs_1` `netWindVelFromBelow_ms_1`	Net wind velocity from below. Net wind is the steady state winds plus any turbulences and shears.	From below			
		`turbulence_fs_1 (3)` `turbulence_ms_1 (3)`	Vector of the wind turbulence velocities impinging on the aircraft. Comprised of the three components as defined below.				
	VNTURB	`turbulenceFromNorth_fs_1` `turbulenceFromNorth_ms_1`	North component of turbulence	From the North			
	VETURB	`turbulenceFromEast_fs_1` `turbulenceFromEast_ms_1`	East component of turbulence	From the East			
	VDTURB	`turbulenceFromBelow_fs_1` `turbulenceFromBelow_ms_1`	Vertical component of turbulence	From below			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
		bodyAngularTurbulence_ds_1 (3) bodyAngularTurbulence_rs_1 (3)	Vector of angular turbulence velocities comprised of the three components as defined below. Body frame.				
	PTURB	rollBodyTurbulenceRate_ds_1 rollBodyTurbulenceRate_rs_1	Body axis roll turbulence	The turbulence would move the aircraft right wing down			
	QTURB	pitchBodyTurbulenceRate_ds_1 pitchBodyTurbulenceRate_rs_1	Body axis pitch turbulence	The turbulence would move the aircraft nose up			
	RTURB	yawBodyTurbulenceRate_ds_1 yawBodyTurbulenceRate_rs_1	Body axis yaw turbulence	The turbulence would move the aircraft nose right			

Table A.6 — Vehicle physical characteristics

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
I		bodyMomentOfInertia_slugf2 (3,3) bodyMomentOfInertia_kgm2 (3,3)	Matrix of the total moments of inertia of the aircraft. This is wrt the CG and includes everything in or attached to the aircraft (stores, passengers, crew, fuel, etc.). It is comprised of the components below. I_{XX} $-I_{XY}$ $-I_{ZX}$ $-I_{XY}$ I_{YY} $-I_{YZ}$ $-I_{ZX}$ $-I_{YZ}$ I_{ZZ}				

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
I_{xx}	XIXX	bodyXXMomentOfInertia_slugf2 bodyXXMomentOfInertia_kgm2	Vehicle Roll Moment Of Inertia about Cg, body frame	NSC			
I_{xx}	XIYY	bodyYYMomentOfInertia_slugf2 bodyYYMomentOfInertia_kgm2	Vehicle Pitch Moment Of Inertia about Cg, body frame	NSC			
I_{zz}	XIZZ	bodyZZMomentOfInertia_slugf2 bodyZZMomentOfInertia_kgm2	Vehicle Yaw Moment Of Inertia about Cg, body frame	NSC			
I_{xz}	XIZX	bodyZXProductOfInertia_slugf2 bodyZXProductOfInertia_kgm2	Vehicle ZX Cross Product Of Inertia about Cg, body frame	NSC			
I_{xy}	XIXY	bodyXYProductOfInertia_slugf2 bodyXYProductOfInertia_kgm2	Vehicle XYy Cross Product Of Inertia about Cg, body frame	NSC			
I_{yz}	XIYZ	bodyYZProductOfInertia_slugf2 bodyZProductOfInertia_kgm2	Vehicle YZ Cross Product Of Inertia about Cg, body frame	NSC			
		BodyCGPosition_f (3) BodyCGPosition_m (3)	Vector of the CG position of the aircraft in the body axis system. Comprised of the three components as defined below.				
	XCGREF	XBodyCGPosition_f XBodyCGPosition_m	C.g. Position w.r.t. L.e. Of the mean aerodynamic chord	CG AFT of LEMAC			
	YCGREF	YBodyCGPosition_f YBodyCGPosition_m	C.g. Position w.r.t. the centerline of the aircraft	CG Right of the a/c centerline			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
	ZCGREF	`ZBodyCGPosition_f` `ZBodyCGPosition_m`	C.g. Position w.r.t. the waterline reference of the aircraft (usually WL 0, see `ZBodyWaterline_`)	CG below the a/c waterline reference			
		`BodyAeroMomentArm_ft` `BodyAeroMomentArm _m`	Vector of the distance from the Moment Reference center to the CG position of the aircraft in the body axis system. Comprised of the three components as defined below.				
ΔX_{cg}	DXCG	`XBodyAeroMomentArm_ft` `XBodyAeroMomentArm _m`	Cg Displacement From the aerodynamic force and moment reference center, + is CG fwd of the Moment Reference Center (MRC). The MRC is the reference point that the aero model forces and moments act upon the aircraft.	FWD			
ΔY_{cg}	DYCG	`YBodyAeroMomentArm _ft` `YBodyAeroMomentArm _m`	Cg Displacement From the aerodynamic force and moment reference center, + is CG to the right of the ARC	RT			
ΔZ_{cg}	DZCG	`ZBodyAeroMomentArm _ft` `ZBodyAeroMomentArm _m`	Cg Displacement From the aerodynamic force and moment reference center, + is CG below the the ARC	DWN			
		`BodyMRCPosition_f (3)` `BodyMRCPosition_m (3)`	Vector of the location of the moment reference center (MRC) of the aircraft in the body axis system. Comprised of the three components as defined below.				

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
	XMRC	`XBodyMRCPosition_f` `XBodyMRCPosition_m`	X MRC Position w.r.t. L.e. Of the mean aerodynamic chord	MRC AFT of LEMAC			
	YMRC	`YBodyMRCPosition_f` `YBodyMRCPosition_m`	Y MRC Position w.r.t. the centerline of the aircraft	MRC Right of the a/c centerline			
	ZMRC	`ZBodyMRCPosition_f` `ZBodyMRCPosition_m`	Z MRC Position w.r.t. the waterline reference of the aircraft (usually WL 0, see `ZBodyWaterlinePosition_`)	MRC below the a/c waterline reference			
	ZWL	`ZBodyWaterlinePosition_f` `ZBodyWaterlinePosition_m`	The waterline (vertical) reference position on the a/c body. This is a constant used to locate the vertical cg and MRC position to the aircraft. Waterline reference position is normally 0 but does not have to be.	NSC			
M	XMASS	`totalMass_slug` `totalMass_kg`	Total Mass Of Vehicle (including Fuel, crew, cargo, stores, passengers, etc.)	NSC			
W	WEIGHT	`grossWeight_lbf` `grossWeight_N`	Aircraft Gross Weight (mass*gravity), including all fuel, occupants, stores, etc.	NSC			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
A	AREA	`referenceWingArea_f2` `referenceWingArea_m2`	Reference Wing Area	NSC			
b	SPAN	`referenceWingSpan_f` `referenceWingSpan_m`	Reference Wing Span	NSC			
c	CHORD	`referenceWingChord_f` `referenceWingChord_m`	Mean Aerodynamic Chord (reference wing chord)	NSC			
		`engineMomentOfInertia_slugf2` `engineMomentOfInertia_kgm2`	Matrix of the moments of inertia of the Rotating engine, for an engine with the propeller, includes the propeller and drive train. This is w.r.t. the rotational axis of the engine. For multi-engine vehicles is for one engine. It is comprised of the components below. I_{EXX} $-I_{EXY}$ $-I_{EZX}$ $-I_{EXY}$ I_{EYY} $-I_{EYZ}$ $-I_{EZX}$ $-I_{EYZ}$ I_{EZZ}				
I_{Exx}	IEXX	`engineXXMomentOfInertia_slugf2` `engineXXMomentOfInertia_kgm2`	Moment of inertia about the X-axis Of Rotating Eng, for an engine with the propeller, includes the propeller This is w.r.t. the rotational axis of the engine				

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
I_{EYY}	IEYY	`engineYYMomentOfInertia_slugf2` `engineYYMomentOfInertia_kgm2`	Moment of inertia about the Y-axis Of Rotating Eng, for an engine with the propeller, includes the propeller This is w.r.t. the rotational axis of the engine				
I_{EZZ}	IEZZ	`engineZZMomentOfInertia_slugf2` `engineZZMomentOfInertia_kgm2`	Moment of inertia about the Z-axis Of Rotating Eng, for an engine with the propeller, includes the propeller This is w.r.t. the rotational axis of the engine				
I_{EXZ}	IEXZ	`engineXZProductOfInertia_slugf2` `engineXZProductOfInertia_kgm2`	Product of inertia about the XZ-axis Of Rotating Eng, for an engine with the propeller, includes the propeller This is w.r.t. the rotational axis of the engine				

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value

I_{EXY}	IEXY	`engineXYProductOfInertia_slugf2`
		
		`engineXYProductOfInertia_kgm2`
		
		`(engine_xy_product_of_inertia_slugf2)`
			Product of inertia about the XY-axis Of Rotating Eng, for an engine with the propeller, includes the propeller
			
			This is w.r.t. the rotational axis of the engine				

I_{EYZ}	IEYZ	`engineYZProductOfInertia_slugf2`
		
		`engineYZProductOfInertia_kgm2`
		
		`(engine_yz_product_of_inertia_slugf2)`
			Product of inertia about the YZ-axis Of Rotating Eng, for an engine with the propeller, includes the propeller
			
			This is w.r.t. the rotational axis of the engine				

		`fuelInTank_lbm(number of fuel tanks)`
		
		`fuelInTank_kg(number of fuel tanks)`
			Vector of fuel weight by tank. Each aircraft tank is normally numbered and the vector should be ordered according to fuel tank number. In the absence of tank numbering the convention of port to starboard, upper to lower, then front to rear should be used.				

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value

		`fuelTankCentroid_f(number of fuel tanks,3)`
		
		`fuelTankCentroid_m(number of fuel tanks,3)`
			Matrix used to locate the centoids of the fuel tanks. Each aircraft tank is normally numbered and the matrix should be ordered according to fuel tank number. The second component is the x, y and z moment arms from the moment reference center to the tank centroid in the body axis. In the absence of tank numbering the convention of port to starboard, upper to lower, then front to rear should be used.	Tank centroid behind, right, and below the moment reference center.			

Table A.7 — Vehicle control position

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value

		`pilotLongControlPos_d`
		
		`pilotLongControlPos_r`
			Longitudinal control position of the pilot.	AFT			

		`pilotLatControlPos_d`
		
		`pilotLongControlPos_r`
			Lateral control position of the pilot.	RT			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
		pilotPedalControlPos_d pilotPedalControlPos_r	Net Directional control position of the pilot. Normally, Right pedal – left pedal.	Pedal in or clockwise twist of a sidestick			
		pilotRightPedalControlPos_d pilotRightPedalControlPos_r	Right Directional control position of the pilot.	Pedal in.			
		pilotLeftPedalControlPos_d pilotLeftPedalControlPos_r	Left Directional control position of the pilot.	Pedal in.			
		pilotCollectiveControlPos_d pilotCollectiveControlPos_r	Pilot collective control position.	UP			
		pilotAvgThrottleControlPos_d pilotAvgThrottleControlPos_r	Average pilot throttle control position.	FWD			
		pilotThrottleControlPos_d (number of engines) pilotThrottleControlPos_r (number of engines)	Individual pilot throttle control positions. Order is outboard port (left) to outboard starboard.	FWD			
		copilotLongControlPos_d copilotLongControlPos_r	Longitundal control position of the copilot.	AFT			
		copilotLatControlPos_d copilotLongControlPos_r	Lateral control position of the copilot.	RT			
		copilotPedalControlPos_d copilotPedalControlPos_r	Net Directional control position of the copilot. Nornally, Right pedal – left pedal.	Pedal in or clockwise twist of a sidestick.			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
		copilotRightPedalControlPos_d copilotRightPedalControlPos_r	Right Directional control position of the copilot.	Pedal in.			
		copilotLeftPedalControlPos_d copilotLeftPedalControlPos_r	Left Directional control position of the copilot.	Pedal in.			
		copilotCollectiveControlPos_d copilotCollectiveControlPos_r	Copilot collective control position.	UP			
		copilotAvgThrottleControlPos_d copilotAvgThrottleControlPos_r	Average copilot throttle control position.	FWD			
		copilotThrottleControlPos_d (number of engines) copilotThrottleControlPos_r (number of engines)	Individual copilot throttle control positions. Order is outboard port (left) to outboard starboard.	FWD			
		avgThrottleControlPos_d avgThrottleControlPos_r	Average pilot and copilot throttle control position.	FWD			
		throttleControlPos_d (number of engines) throttleControlPos_r (number of engines)	Individual throttle control position (pilot and copilot average). Order is outboard port (left) to outboard starboard.	FWD			
		avgPropControlPos_d avgPropControlPos_r	Average pilot and copilot propeller blade pitch control position.	FWD			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
		propControlPos_d (number of engines) propControlPos_r (number of engines)	Individual propeller blade pitch control position. Order is outboard port (left) to outboard starboard.	FWD			
		trailingEdgeFlapDeflection (number of leading edge flap control surfaces)	Vector of trailing edge flap positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	LED			
		avgTrailingEdgeFlapDeflection_d	Trailing edge flap deflection. Average for all trailing edge flap surfaces.	TED			
		differentialTrailingEdgeFlapDeflection_d	Measure of roll control due to trailing edge flap deflection differences in vehicles with multiple control surfaces, usually (left deflections-right deflections)	RWD control			
		leadingEdgeFlapDeflection (number of leading edge flap control surfaces)	Vector of leadng edge flap positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	LED			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
		avgLeadingEdgeFlapDeflection_d	Leading edge flap/slat deflection. Average for all deflected leading edge flap/slat surfaces.	LED			
		differentialLeadingEdgeFlapDeflection_d	Measure of roll control due to leading edge flap deflection differences in vehicles with multiple control surfaces, usually (left deflections-right deflections)	RWD control			
		spoilerDeflection (number of spoiler control surfaces)	Vector of spoiler control positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	TEU			
		avgSpoilerDeflection_d	Spoiler deflection. Average for all deflected spoilers	TEU			
		differentialSpoilerDeflection_d	Measure of roll control due to spoiler deflection differences in vehicles with multiple control surfaces, usually (right deflections-left deflections)	RWD control			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
		aileronDeflection (number of aileron control surfaces)	Vector of aileron control positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	TEU			
		avgAileronDeflection	Differential aileron deflection, right-left	Right aileron TEU			
		rudderDeflection_d (number of rudder control surfaces)	Vector of rudder control positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	TEL			
		avgRudderDeflection_d	Average rudder deflection	TEL			
		differentialRudderDeflection_d	Measure of yaw control due to rudder deflection differences in vehicles with multiple control surfaces, usually (right deflections-left deflections)				
		rudderTabDeflection_d (number of rudder tab control surfaces)	Vector of rudder tab control positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	TEL			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
		avgRudderTabDeflection_d	Average rudder tab deflection	TEL			
		differentialRudderTabDeflection_d	Measure of yaw control due to rudder tab deflection differences in vehicles with multiple control surfaces, usually (right deflections-left deflections)				
		elevatorDeflection_d (number of elevator control surfaces)	Vector of elevator (or stabilizer/stabilator) control positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	TEU			
		avgElevatorDeflection_d	Average elevator (or stabilizer/stabilator) deflection	TEU			
		differentialElevatorDeflection_d	Measure of roll control due to elevator (or stabilizer/stabilator) deflection differences in vehicles with multiple control surfaces, usually (right deflections-left deflections)	Right control TEU			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
		elevatorTabDeflection_d (number of elevator tab control surfaces)	Vector of elevator (or stabilizer/stabilator) tab control positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	TEU			
		avgElevatorTabDeflection_d	Average elevator (or stabilizer/stabilator) tab deflection	TEU			
		differentialElevatorTabDeflection_d	Measure of roll control due to elevator (or stabilizer/stabilator) tab deflection differences in vehicles with multiple control surfaces, usually (right deflections-left deflections)	Right control TEU			
		canardDeflection_d (number of canard control surfaces)	Vector of canard control positions, one for each surface. Order is outboard port (left) to outboard starboard.	TED			
		avgCanardDeflection_d	Average canard deflection	TED			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
		differentialCanardDeflection_d	Measure of roll control due to canard deflection differences in vehicles with multiple control surfaces, usually (right deflections-left deflections)	Right control TED			
		canardTabDeflection_d (number of canard tab control surfaces)	Vector of canard tab control positions, one for each surface. Order is outboard port (left) to outboard starboard.	TED			
		avgCanardTabDeflection_d	Average canard tab deflection	TED			
		differentialCanardTabDeflection_d	Measure of roll control due to canard tab deflection differences in vehicles with multiple control surfaces, usually (right deflections-left deflections)	Right control TED			
		speedbrakeDeflection_d	Speedbrake deflection	Extended			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
		`landingGearPosition (number of landing gear struts)`	Vector of landing gear positions, one for each strut. Order is outboard port (left) to outboard starboard.	0= up and locked 1= full extension with no weight on wheels			
		`landingGearWeightOnWheels_lbf (number of landing gear struts)` `landingGearWeightOnWheels_kg (number of landing gear struts)`	Vector of landing gear weight on wheels, one for each strut. Order is outboard port (left) to outboard starboard.				
		`landingGearWheelSpeed_rs_1 (number of landing gear struts, number of trucks, number of wheels per truck)`	Array of landing gear wheel speeds by strut, one for each strut. Order of struts is outboard port (left) strut, to outboard starboard. Order of trucks is front to rear. Order of wheels on each truck is port to starboard.				

Table A.8 — Vehicle aerodynamic characteristics

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
C_L	CL	`totalCoefficientOfLift`	Coefficient Of Lift, Total, includes effects of stores	UP			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
C_D	CD	totalCoefficientOfDrag	Coefficient Of Drag, Total, includes effects of stores	AFT			
		aeroBodyForceCoefficient(3)	Vector of total aerodynamic force coefficients in the body axis system, comprised of the three components as defined below.				
C_X	CX	aeroXBodyForceCoefficient	X-body Force Coefficient due to aerodynamic loads, includes stores (Body axis)	FWD			
C_Y	CY	aeroYBodyForceCoefficient	Y-body Force Coefficient due to aerodynamic loads, includes stores (Body axis)	RT			
C_Z	CZ	aeroZBodyForceCoefficient	Z-body Force Coefficient due to aerodynamic loads, includes stores (Body axis)	DOWN			
		aeroBodyForce_lbf (3) aeroBodyForce_N (3)	Vector of total aerodynamic forces in the body axis system, including stores. Comprised of the three components as defined below.				
F_{AX}	FAX	aeroXBodyForce_lbf aeroXBodyForce_N	Total X-body Force due to aerodynamic loads, includes stores (Body axis)	FWD			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
F_{AY}	FAY	aeroYBodyForce_lbf aeroYBodyForce_N	Total Y-body Force due to aerodynamic loads, includes stores (Body axis)	RT			
F_{AZ}	FAZ	aeroZBodyForce_lbf aeroZBodyForce_N	Total Z-body Force due to aerodynamic loads, includes stores (Body axis)	DOWN			
		thrustBodyForce_lbf (3) thrustBodyForce_N (3)	Vector of total net propulsion system forces in the body axis system (includes installion losses, inlet efficieny and propeller efficiency). Comprised of the three components as defined below.				
F_{EX}	FEX	thrustXBodyForce_lbf thrustXBodyForce_N	Total net engine thrust Force, X-body axis	FWD			
F_{EY}	FEY	thrustYBodyForce_lbf thrustYBodyForce_N	Total net engine thrust Force , Y-body axis	RT			
F_{EZ}	FEZ	thrustZBodyForce_lbf thrustZBodyForce_N	Total net engine thrust Force, Z-body axis	DOWN			
		gearBodyForce_lbf (3) gearBodyForce_N (3)	Vector of total landing gear ground reaction forces in the body axis system. Does NOT include aerodynamic forces on the landing gear which are included in aeroBodyForce defined above. Comprised of the three components as defined below.				
F_{GX}	FGX	gearXBodyForce_lbf gearXBodyForce_N	Total landing gear ground reaction force, X-body axis	FWD			
F_{GY}	FGY	gearYBodyForce_lbf gearYBodyForce_N	Total landing gear ground reaction force, Y-body axis	RT			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
F_{GZ}	FGZ	`gearZBodyForce_lbf` `gearZBodyForce_N`	Total landing gear ground reaction force, Z-body axis	DOWN			
		`totalBodyForce_lbf (3)` `totalBodyForce_N (3)`	Vector of total forces in the body axis system. Includes all forces exerted upon the aircraft. Comprised of the three components as defined below.				
F_{xTOT}	FX	`totalXBodyForce_lbf` `totalXBodyForce_N`	Total Forces On a/c, X-body axis	FWD			
F_{yTOT}	FY	`totalYBodyForce_lbf` `totalYBodyForce_N`	Total Forces On a/c, Y-body axis	RT			
F_{zTOT}	FZ	`totalZBodyForce_lbf` `totalZBodyForce_N`	Total Forces On a/c, Z-body axis	DOWN			
		`aeroBodyMomentCoefficient (3)`	Vector of total aerodynamic moment coefficients in the body axis system, including stores. Comprised of the three components as defined below.				
C_l	CLL	`aeroRollBodyMomentCoefficient`	Total Aerodynamic Rolling Moment Coefficient including stores. Moment about the X-body axis	RWD			
C_m	CLM	`aeroPitchBodyMomentCoefficient`	Total Aerodynamic Pitching Moment Coefficient, including stores. Moment about the Y-body axis	ANU			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
C_n	CLN	aeroYawBodyMomentCoefficient	Total Aerodynamic yawing Moment Coefficient, including stores. Moment about the Z-body axis	ANR			
		aeroBodyMoment_flbf (3) aeroBodyMoment_Nm (3)	Vector of total aerodynamic moments in the body axis system, including stores. Referenced to the moment reference center. Comprised of the three components as defined below.				
L_A	TAL	aeroRollBodyMoment_flbf aeroRollBodyMoment_Nm	Total Aerodynamic Rolling moment (including attached stores), about the X-body axis	RWD			
M_A	TAM	aeroPitchBodyMoment_flbf aeroPitchBodyMoment_Nm	Total Aerodynamic pitching moment (including attached stores), about the Y-body axis	ANU			
N_A	TAN	aeroYawBodyMoment_flbf aeroYawBodyMoment_Nm	Total Aerodynamic yawing moment (including attached stores), about the Z-body axis	ANR			
		thrustBodyMoment_flbf (3) thrustBodyMoment_Nm (3)	Vector of total net propulsion system moments in the body axis system (includes installation losses, inlet efficiency and propeller efficiency). Referenced to the moment reference center. Comprised of the three components as defined below.				
L_E	TEL	thrustRollBodyMoment_flbf thrustRollBodyMoment_Nm	Total Engine Rolling Moment, about the X-body axis	RWD			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value

M_E	TEM	`thrustPitchBodyMoment_flbf`
		`thrustPitchBodyMoment_Nm`
		`(thrust_body_pitch_moment_flbf)`
			Total Engine pitching Moment, about the Y-body axis	ANU			

N_E	TEN	`thrustYawBodyMoment_flbf` `thrustYawBodyMoment_Nm`	Total Engine yawing Moment, about the X-body axis	ANR			
		`landingGearBodyMoment_flbf (3)` `landingGearBodyMoment_Nm (3)`	Vector of total landing gear ground reaction moments in the body axis system. Referenced to the moment reference center. Does NOT include aerodynamic moments on the landing gear which are included in `aeroBodyMoment` defined above. Comprised of the three components as defined below.				
L_G	TGL	`landingGearRollBodyMoment_flbf` `landingGearRollBodyMoment_Nm`	Total Landing Gear Rolling Moment, about the X-body axis	RWD			
M_G	TGM	`landingGearPitchBodyMoment_flbf` `landingGearPitchBodyMoment_Nm`	Total Landing gear Pitch Moment, about the Y-body axis	ANU			
N_G	TGN	`landingGearYawBodyMoment_flbf` `landingGearYawBodyMoment_Nm`	Total Landing Gear Yawing Moment, about the Z-body axis	ANR			
		`totalBodyMoment_flbf (3)` `totalBodyMoment_Nm (3)`	Vector of total moments in the body axis system. Referenced to the moment reference center. Includes all moments exerted upon the aircraft. Comprised of the three components as defined below.				
L_{TOT}	TTL	`totalRollBodyMoment_flbf` `totalRollBodyMoment_Nm`	Total Rolling Moment, about the X-body axis	RWD			
M_{TOT}	TTM	`totalPitchBodyMoment_flbf` `totalPitchBodyMoment_Nm`	Total Pitching Moment, about the Y-body axis	ANU			

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value
N_{TOT}	TTN	`totalYawBodyMoment_flbf` `totalYawBodyMoment_Nms`	Total Yawing Moment, about the Z-body axis	ANR			

Table A.9 — Simulation control parameters

Symbol	Short Name	Full Variable Name	Description	Sign Convention	Initial Value	Min Value	Max Value

	TIME	`simTime_s`
		`simTime_s`
		`(sim_time_s)`
			Time Since Start Of Operate Mode	NSC			

		`deltaTime_s` (number of different integration step sizes)	Vector of Integration step sizes				

Annex B Dynamics Aerospace Vehicle Exchange Markup Language (DAVE-ML) Reference (Normative)

Dynamic Aerospace Vehicle Exchange Markup Language (DAVE-ML) Reference

Version 2.0 (Release Candidate 3)

$Revision: 382 $

AIAA Modeling and Simulation Technical Committee
[http://www.aiaa.org/portal/index.cfm?GetComm=79&tc=tc]

Bruce Jackson, NASA Langley Research Center <bruce.jackson@nasa.gov>

Abstract

This is a draft version of the eventual reference manual for DAVE-ML syntax and markup. DAVE-ML syntax is specified by the `DAVEfunc.dtd` Document Type Definition file; the version number above refers to the version of the `DAVEfunc.dtd`.

DAVE-ML is an open standard, being developed by an informal team of members of the American Institute of Aeronautics and Astronautics (AIAA). Contact the editor above for more information or comments regarding further refinement of DAVE-ML.

Table of Contents

B-1. Changes to this document

A list of changes during the course of development of the DAVE-ML Document Type Definition is given in this section.

B-1.1. Changes since version 2.0RC2

- In the DTD, the default DAVE-ML namespace definition was added to the top-level <DAVEfunc> element to observe a best-practice for XML DTDs, at the suggestion of Dan Newman.
- In the DTD and reference manual, changed the <bpVals> elements to allow whitespace separation of values in addition to comma separation (which has always been the case for the <dataTable> element). Ditto for <dependentVarPts> and< independentVarPts>. This may cause a problem for some existing parsers.
- No other changes have been made to the DAVE-ML grammar since 2.0RC2, but this version of the DTD and reference manual include clean-up of a number of inconsistencies (found by Dennis Linse and Trey Arthur) between the reference manual and element descriptions, as noted below.
- More DTD clean-up to removed redundant 'optional' specifiers on elements and attributes; made the 'internal identifier' descriptions consistent throughout the DTD and reference manual; tweak to the style of the reference pages in the reference manual, fixed some poor grammar in the DTD, replaced parentheses with brackets in the BNF syntax in the reference manual for comments to avoid confusion, and reformatted the reference pages in the manual for readibility, and added references to W3C on-line documentation where necessary. Thanks to Dennis Linse for shaming me into fixing these long-standing inconsistencies and distractions.
- Put editorial comments in BNF syntaxes in braces to avoid confusion with parentheses, which are used to indicate a choice; thanks, Dennis, for the catch.
- In the DTD, a sentence was added to the description of the deprecated <fileCreationDate> element and a similar description was added to the deprecated <functionCreationDate> element mentioning that they have been deprecated in version 2.0, at the near-simultaneous suggestion of both Trey Arthur and Dan Newman.
- Added a paragraph about the alias attribute of the <variableDef> element; it's a valid attribute but was missing an explanation. Thanks to Trey Arthur for catching this long-standing omission.
- Changed the definition of atan2 so that the input arguments are not limited to ±1 and are not referred to as 'sine' and 'cosine' to better match ANSI C. Thanks to Dan Newman for catching this inconsistency.
- Corrected typos in <fileHeader>, <variableDef>, <griddedTableDef>, <ungriddedTableDef>, and <function> element schematic overview syntaxes in sections 6.2.1-6.2.6, thanks to Dennis Linse.
- Added a sentence about the <tol> subelement being an absolute difference to remove ambiguity over its interpretation.
- In the reference manual syntax layout, moved the <provenance> and <provenanceRef> subelements out of <checkData> and into <staticShot> subelements since checkData, a singleton placeholder, doesn't warrant a description, but each staticShot does. provenance and provenanceRef are still in the DTD for checkData but only for backwards

compatibility; their use in a checkData element is deprecated and may be removed in a future version. Thanks to Dennis Linse for prompting this needed change.

- Added a note to <documentRef> docID attribute that it is deprecated. Thanks to Dennis Linse for catching this omission.
- Removed the redundant "optional" in the description of <reference> element's xlink:href attribute. Added the constant value for attributes xmlns:xlink and xlink:type. Expanded the somewhat terse description of this element. Thanks to Dennis Linse for this correction and other helpful suggestions.

B-1.2. Changes since version 2.0RC1

- Tweaked examples and syntaxes to match the 2.0 RC 2 DTD. Cleaned up a couple figures; incorporated several new reference citations. Added interpolation paragraph.
- Deprecated <signalID> used for internal signals in checkcase data in favor of the more consistent <varID>, (which meant the introduction of a formal <varID> element) and made the specification of <signalUnits> subelement mandatory for input and output signals for consistency. Thanks to Dan Newman for helping solidify the thinking about this.
- Changed examples in this text to use updated AIAA variable names to match revised (but unpublished) draft standard of September 2008. Changed many 'examples' to 'excerpts' to emphasize the missing portions of a valid DAVE-ML file. Corrected units in checkcase examples to match proposed AIAA standard.
- Removed the symmetric attribute of the <uniformPDF> subelement; this was redundant as symmetric distribution is implied with a single <bounds> subelement, and asymmetric is implied by two bounds subelements. Kudos to Dan Newman and Dennis Linse for catching the inconsistent examples and for suggesting this convention.
- Corrected the DTD by reversing the definition of the floor and ceiling values of the interpolate attribute of the <independentVarPts> element; also corrected the correlated uncertainty example; thanks to Dan Newman for catching both of these in 2.0 RC1.
- Corrected the DTD so that only one <checkData> element is allowed (but it can have multiple, different, <staticShot> test conditions). Thanks to Dan and Dennis for reporting this inconsistency between the reference manual and the DTD.
- Added a <description> sub-element to <staticShot> element in response to a suggestion by Dennis Linse; added to example listings.
- Added a section about namespaces; removed the hard link in DTD that incorrectly set namespace for the <calculation> element.
- Added new multi-purpose <creationDate> element to replace the single-purpose <fileCreationDate> and <functionCreationDate> elements, at the suggestion of Dennis Linse of SAIC. <fileCreationDate> and <functionCreationDate> are now deprecated.
- Corrected descriptions of <ungriddedTableDef> and <griddedTableDef> to reflect the possible use of an internal <function> element; previously the descriptions implied that these elements were only specified external to functions and thus the utID and gtID attributes, respectively, were required. Thanks to Dennis Linse for the correction.
- Depicted <provenanceRef> as an option to the <provenance> subelement for <griddedTableDef>, <ungriddedTableDef>, and <function> elements in the

narrative part of this manual (it was described correctly in the reference section). Also added both <provenance> and <provenanceRef> as optional subelements of the <variableDef> and <checkData> elements. Thanks to Dennis Linse for the correction.

- Added '[Deprecated]' in the description of <griddedTable> , <ungriddedTable> , and <confidenceBound> elements for consistency; these were previously deprecated but not marked clearly in each element's 'purpose' section. Thanks to Dennis Linse for the suggestion.
- Updated the acknowledgment paragraph of the DTD; significantly reformatted the .pdf version of this document and added section numbers to all versions.

B-1.3. Changes since version 1.9b3

- Added ceiling and floor enumeration selections to interpolate attribute of <independentVarPts> and <independentVarRef> elements at the suggestion of Geoff Brian, Giovanni Cignoni, Randy Brumbaugh, and Daniel M. Newman.
- Added five uncertainty examples.
- Cleaned up all FIXME and BUG notes.
- Corrected and expanded the labels on the DAVE-ML excerpt figure.

B-1.4. Changes since version 1.9b2

- Corrected link to [Jackson04] paper.
- Added discrete enumeration selection to interpolate attribute of <independentVarPts> and <independentVarRef> elements at the suggestion of Geoff Brian, DSTO.
- Added a section and a <variableDef> example on extending the MathML-2 function set with atan2.
- Removed all xns attributes from examples.
- Amplified that it is a good practice to provide <variableDef>s in sorted sequence.

B-1.5. Changes since version 1.8b1

- Added a quadraticSpline enumerated value to the interpolate attributes of the <independentVarPts> and <independentVarRef> elements in response to a request from Geoff Brian of DSTO; fixed typo in cubicSpline attribute string. Added reference to Wikipedia article on spline interpolation [http://en.wikipedia.org/wiki/Spline_interpolation].
- Added a classification attribute to the <reference> element; added a date attribute to the <modificationRecord> element, per suggestions by Geoff Brian of DSTO.
- Added two-D and three-D ungridded table examples and figures; corrected typo on ungridded table definition syntax (thanks to Dr. Peter Grant of U. Toronto's UTIAS and Geoff Brian of Australia's DSTO).
- Reintroduced <!ENTITY> to include MathML2 DTD (complete) in the body of this DTD. This entity definition quietly went away in version 1.6 due to a misunderstanding of proper way to include external DTDs; it is reintroduced to assist validating parsers.
- Added a <description> sub-element to the <provenance> element, so the provenance entry can contain more information about change justification documents; made

<provenance> or <provenanceRef> acceptable sub-elements to <variableDef> and <checkData> elements after a request from Geoff Brian of DSTO.

B-1.6. Changes since version 1.7b1

- Renamed docID attribute to refID of the <modificationRecord> so the attribute name is consistent; docID attribute is deprecated but remains for compatibility with older documents.
- Added <correlatesWith> and <correlation> sub-elements of <uncertainty> element to allow for multiple-dimensioned linear correlation of uncertainty of selected functions and variables.
- Added a new element, <contactInfo> , to replace the single <address> element. This format supports multiple ways to indicate the means of contacting the author of a document or reference. <address> is deprecated but is retained for backwards compatibility. This element also replaces the email and xns attributes of <author> .
- Fixed a typographical error in <ungriddedTableRef> element: incorrect gtID attributed corrected to utID.
- Allowed multiple <author> elements wherever one was allowed before.
- Added a new tag, <isStdAIAA/> , to indicate a variableDef refers to one of the standard AIAA variables.
- Removed <[un]griddedTable[Ref|Def]> sub-elements of the <confidenceBound> element since this leads to circular logic.
- Changed SYSTEM ID to reflect new daveml.nasa.gov domain availability.
- Removed e-mail URLs to protect privacy of individual contributors.
- Added a new attribute, interpolate, to the <independentVarPts> element to indicate whether the table interpolation should be linear or cubic spline in the given dimension [modified to include quadratic in version 1.9].
- Added a new tag, <isState/> , to indicate a variableDef refers to a state variable in the model.
- Added a new tag, <isStateDeriv/> , to indicate a <variableDef> refers to a state derivative variable in the model.

B-1.7. Changes since version 1.6b1

Added <checkData> and associated elements. Added <description> sub-element to <reference> element.

B-1.8. Changes since version 1.5b3

Added an <uncertainty> element. Emphasized MathML content markup over presentation markup. Several grammatical and typographical errors fixed; added figure 1. Added ISO 8601 (Dates and Times) reference.

B-1.9. Changes since version 1.5b2

- Added Bill Cleveland (NASA Ames' SimLab) and Brent York (NAVAIR's Manned Flight Simulator) to the acknowledgments section, to thank them for their pioneering initial trials of DAVE-ML.

- Added <provenanceRef> element and changed all parents of <provenance> elements to be able to use a <provenanceRef> reference instead (these were <function>, <griddedTableDef> and <ungriddedTableDef>) to eliminate duplicate <provenance> elements.

- Realization dawned that there was little difference between <griddedTable> and <griddedTableDef>s but the latter was more flexible (ditto <ungriddedTable> and <ungriddedTableDef>s). By making the gtID and utID attributes "implied" instead of "required," we can use the Def versions in both referenced-table and embedded-table <function>s. Thus the original <griddedTable> and <ungriddedTable> elements have been marked as "Deprecated." They are still supported in this DTD for backwards compatibility but should be avoided in future use; the easiest way to modify older DAVE-ML models would be to rename all <griddedTable>s as <griddedTableDef>s.

B-1.10. Changes since version 1.5b

- Fixed typographical errors pointed out by Bill Cleveland.

- Added <fileVersion> element to <fileHeader> element, so each version of a particular DAVEfunc model can be uniquely identified. Format of the version identifier is undefined.

- Added an email attribute to the <author> element. The eXtensible Name Service (xns [http://www.xns.org]) standard doesn't appear to be catching on as rapidly as hoped, so a static e-mail link will have to do for now, at least until the replacement XRI technology is more widely adopted.

- Added a mandatory varID attribute to both <independentVarPts> and <dependentVarPts> so these can be associated with an input and output signal name (<variableDef>), respectively.

- Added an optional <extraDocRef> element to the <modificationRecord> element so more than one document can be associated with each modification event; if only one document needs to be referenced, use of the optional refID in the <modificationRecord> itself will suffice.

B-2. Introduction

This document describes the format for DAVE-ML model definition files. DAVE-ML is a proposed standard format for the interchange of aerospace vehicle flight dynamic models. The intent of DAVE-ML is to significantly expedite the process of re-hosting a simulation model from one facility to another, as well as an improved method to promulgate changes to a particular model between various facilities.

DAVE-ML is based on the eXtensible Markup Language (XML), a World-Wide Web Consortium (W3C) standard. More information on XML is available here [http://www.w3.org/XML/].

Many benefits may be derived from application of XML in general, and DAVE-ML in particular, to the exchange of aerospace vehicle data:

- Human-readable text description of the model,
- Unambiguous machine-readable model description, suitable for conversion into programming language or direct import into object-oriented data structures at run-time,
- The same source file can be used for computer-aided design and real-time piloted simulation,
- Based on open, non-proprietary, standards that are language- and facility-independent,
- Statistical properties, such as confidence bounds and uncertainty ranges, can be embedded, suitable for Monte Carlo or other statistical analysis of the model,
- Compliant with AIAA draft simulation data standards,
- Self-contained, complete, archivable data package, including references to reports, wind-tunnel tests, author contact information, data provenance, and
- Self-documenting and easily convertible to on-line and hard-copy documentation,

A more complete discussion on the benefits and design of DAVE-ML can be found at the DAVE-ML web site: *http://daveml.nasa.gov* [http://daveml.nasa.gov]

B-3. Purpose

DAVE-ML is intended to encode (for transmission and long-term archive) an entire flight vehicle dynamic simulation data package, as is traditionally done in initial delivery and updates to engineering development, flight training, and accident investigation simulations. It is intended to provide a programming-language-independent representation of the aerodynamic, mass/inertia, landing gear, propulsion, and guidance, navigation and control laws for a particular vehicle.

Traditionally, flight simulation data packages are often a combination of paper documents and data files on magnetic or optical media. This collection of information is very much vendor-specific and is often incomplete or inconsistent. Many times, the preparing facility makes incorrect assumptions about how the receiving facility's simulation environment is structured. As a result, the re-hosting of the dynamic flight model by the receiving facility can take weeks or longer as the receiving facility staff learns the contents and arrangement of the data package, the model structure, the various data formats, variable names/units/sign conventions and then spends additional time running check cases (if any were included in the transmittal) and tracking down inevitable differences in results.

There are obvious benefits if this tedious, manual process could be mostly automated. Often, when a pair of facilities has exchanged one model, the transmission of another model is much faster since the receiving facility will probably have devised some scripts and processes to convert the data (both model and check-case data).

The purpose of DAVE-ML is to define a common exchange format for these flight dynamic models. The advantage gained is that any simulation facility or laboratory, after having written a DAVE-ML import and/or export script, could automatically receive and/or transmit such packages (and updates to those packages) rapidly with other DAVE-ML-compliant facilities.

To accomplish this goal, the DAVE-ML project is starting with the bulkiest part of the most aircraft simulation packages: the aerodynamic model. This early version of DAVE-ML can be used to transport a complete aerodynamics model, including descriptions of the aerodynamic build-up equations and the data tables, as well as include references to the documentation about the aerodynamic model and check-case data. This format also lends itself to any static subsystem model (i.e. one that contains no state vector) such as the mass & inertia model, or a weapons load-out model, or perhaps a navigational database. The only requirement is that model outputs can be unambiguously defined in terms of inputs, with no past history (state) information required.

B-4. Background

The idea of a universally-understood flight dynamics data package has been discussed for at least two decades within the American Institute of Aeronautics and Astronautics (AIAA) technical committees. There have been proposals in the past to standardize on FORTRAN as well as proprietary, vendor-specified modeling packages (including graphical ones). The National Aerospace Plane (NASP) program, under the guidance of Larry Schilling of NASA Dryden, developed a combination Web- and secure FTP-based system for exchanging NASP subsystem models, as well as a naming convention for variables, file names, and other simulation components. Some simulation standards have subsequently been proposed by the AIAA and are under active consideration at this writing.

B-4.1. Existing standards

The AIAA has published a Recommended Practice concerning sign conventions, axes systems, and symbolic notation for flight vehicle models [AIAA92].

The AIAA Modeling and Simulation Technical Committee has prepared a draft standard for the exchange of simulation modeling data. This included a methodology for accomplishing the gradual standardization of simulation model components, a mechanism for standardizing variable names within math models, and proposed HDF as the data format. [AIAA01], [AIAA03]

B-4.2. DAVE-ML proposal

In a 2002 AIAA paper, Jackson and Hildreth proposed using XML to exchange flight dynamic models [Jackson02]. This paper gave outlines for how such a standard could be accomplished, and provided a business justification for pursuing such a goal.

This proposal included several key aspects from the draft standard, including allowing use of the AIAA variable name convention, data table schema, and including traceability for each data point back to a referenced document or change order.

In a subsequent paper, Jackson, Hildreth, York and Cleveland [Jackson04] reported on results of a demonstration of using DAVE-ML to transmit two aerodynamic models between simulation facilities, showing the feasibility of the idea.

B-4.3. Recent applications

Several successful applications of DAVE-ML have been reported. These include the adoption of DAVE-ML by the Australian Defense Science and Technology Organization (DSTO) for threat models [Brian05] and the U.S. Navy for their Next Generation Threat System [Hildreth08]. Import tools to allow the direct use of DAVE-ML models (without recompilation) in real-time piloted simulations have been reported by NASA Langley Research Center [Hill07] and at NASA Ames Research Center. Some interest has been generated within NASA's Orion Project as well [Acevedo07]. Other applications include TSONT, a trajectory optimization tool ([Durak06]) and aircraft engine simulations ([Lin04]).

DAVE-ML format for models has also been supported by the GeneSim [http://genesim.sourceforge.net] project, which is providing open-source utility programs that realize a DAVE-ML model in object-oriented source code such as C++, Java and C#.

B-5. Supporting technologies

DAVE-ML relies on MathML, version 2.0, as a means to describe mathematical relationships. MathML is an XML grammar for describing mathematics as a basis for machine to machine communication. It is used in DAVE-ML to describe relationships between variables and function tables and may also be used for providing high-quality typeset documentation from the DAVE-ML source files. More information is available at the MathML home web page, found at http://www.w3.org/Math/.

MathML provides a fairly complete set of mathematical functions, including trigonometric, exponential and switching functions. One function that is available in most programming languages and computer-aided design tools, but is missing from MathML-2, is the two-argument arctangent function which provides a continuous angle calculation by comparing the sine and cosine components of a two-dimensional coordinate set. DAVE-ML provides a means to extend MathML-2 for a small predefined set of functions (currently only the 'atan2' function is supported). Thus, a DAVE-ML compliant processing tool should recognize this extension (which is accomplished using the MathML-2 `<csymbol>` element). See the variable definition element section for a discussion and an example of inserting an extension to MathML-2, the `atan2` function, into a DAVE-ML `<calculation>` element.

B-6. Major Elements

At present, only one major element of DAVE-ML has been defined: the function definition element, or `DAVEfunc`. `DAVEfunc` is used to describe static models such as aerodynamic and inertia/mass models, where an internal state is not included. Static check-cases can also be provided for verification of proper implementation.

Other major elements are envisioned to describe dynamic portions of the vehicle model (such as propulsion, landing gear, control systems, etc.) and dynamic check case (time history) data. Ultimately DAVE-ML should be capable of describing a complete flight dynamics model with sufficient data to validate the proper implementation thereof.

B-6.1. The `DAVEfunc` major element

The `DAVEfunc` element contains both data tables and equations for a particular static model. A `DAVEfunc` element is broken into six components: a file header, variable definitions, breakpoint definitions, table definitions, function definitions and optional check-cases. This decomposition reflects common practice in engineering development flight simulation models in which the aerodynamic database is usually captured in multi-dimensional, linearly interpolated function tables. The input to these tables are usually state variables of the simulation (such as Mach number or angle-of-attack). The outputs from these interpolated tables are combined to represent forces and moments acting on the vehicle due to aerodynamics.

It is possible, using DAVEfunc and MathML elements, to completely define an aerodynamic model without use of function tables (by mathematical combinations of input variables, such as a polynomial model) but this is not yet common in the American flight simulation industry.

A `fileHeader` element is included to give background and reference data for the represented model.

Variables, or more properly *signals*, are used to route inputs, calculations and outputs through the subsystem model. Each variable is defined with a `variableDef` element. Variables can be thought of as parameters in a computer program, or signal paths on a block diagram. They can be inputs to the subsystem model, constant values, outputs of the model, and/or the results of intermediate calculations. Variables must be defined for each input and output for any function elements as well as any input or output of the subsystem represented. MathML [http://www.w3.org/Math] *content* markup can be used to define constant, intermediate, or output variables as mathematical combination of constant values, function table outputs, and other variables. MathML *presentation* markup can also be used to define the symbol to use in documentation for each defined variable. Variables also represent the current value of a function (the `dependentVariableDef` in a `function` definition) so the output of functions can be used as inputs to other variables or functions.

Breakpoint definitions, captured in `breakpointDef` elements, consist of a list of monotonically-increasing floating-point values separated by commas. These sets are referenced by "gridded" function table definitions and may be referenced by more than one `function` definition.

Function table definitions, described by `griddedTableDef` and `ungriddedTableDef` elements, generally contain the bulk of data points in an aero model, and typically represent a smooth hyper-surface representing the value of some aerodynamic non-dimensional coefficient as a function of one or more vehicle states (typically Mach number, angle of attack, control surface deflection, and/or angular body rates). These function tables can be either "gridded," meaning the function has a value at every intersection of each dimension's breakpoint, or "ungridded," meaning each data point has a specified coordinate location in n-space. The same table can be reused in several functions, such as a left- and right-aileron moment contribution.

Function definitions (described by `function` elements) connect breakpoint sets and data tables to define how an output signal (or dependent variable) should vary with one or more input signals (or independent variables). The valid ranges of input signal magnitudes, along with extrapolation requirements for out-of-range inputs, can be defined. There is no limit to the number of independent variables, or function dimensionality, of the function.

Check case data (described by a single `checkData` element) can be included to provide information to automatically verify the proper implementation of the model by the recipient. Multiple check cases can (and should) be specified using multiple `<staticShot` test case definitions, as well as optional internal signal values within the model to assist in debugging an instantiation of the model by the recipient.

Figure B-1 contains excerpts from an example model, showing five of the six major parts of a DAVE-ML file.

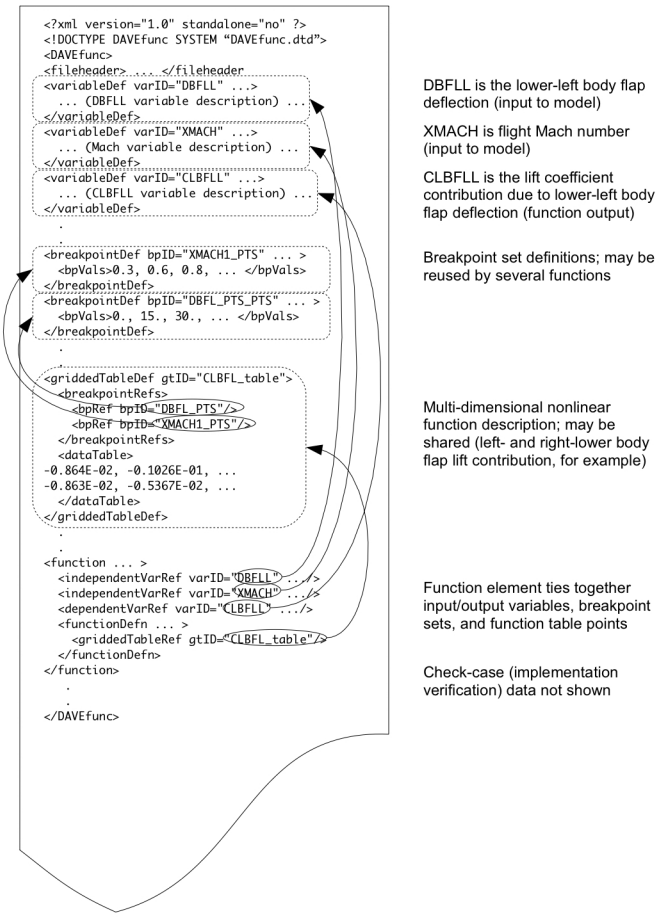


Figure B-1. Excerpts from an example DAVE-ML file

A simpler version of a `function` is available in which the dependent variable breakpoint values and dependent output values are specified directly inside the `function` body. This may be preferred for models that do not reuse function or breakpoint data.

A third form of `function` is to give the gridded table values or ungridded table values inside the `function` body, but refer to externally defined breakpoint sets. This allows reuse of the breakpoint sets by other functions, but keeps the table data private.

B-6.2. Schematic overview of `DAVEfunc`

Shown below are schematic overviews of the various elements currently available in `DAVEfunc`. Each element is described in detail in Section B-8 [53] later in this document. The following key is used to describe the elements and associated attributes.

```
Key:
    elementname : mandatory_attributes, [optional_attributes]
            mandatory_single_sub-element
            optional_single_sub-element?                    {editorial comment}
            ( choice_1_sub-element | choice_2_sub-element )
            zero_or_more_sub-elements*
            one_or_more_sub-elements+
            (character data) implies UNICODE text information
```

The `DAVEfunc` element has six possible sub-elements:

```
    DAVEfunc :
        fileHeader
        variableDef+
        breakpointDef*
        griddedTableDef*
        ungriddedTableDef*
        function*
        checkData?
```

DAVEfunc sub-elements:

`fileHeader` — This mandatory element contains information about the origin and development of this model.

`variableDef` — Each DAVEfunc model must contain at least one signal path (such as a constant output value). Each input, output or internal signal used by the model must be specified in a separate `variableDef` .

A signal can have only a single origin (an input block, a calculation, or a function output) but can be used (referenced) more than once as an input to one or more functions, signal calculations, and/or as a model output.

The `variableDef` s should appear in calculation order; that is, a `variableDef` should not appear before the definitions of variables upon which it is dependent. This is good practice since doing so avoids a circular reference. If a variable depends upon the output (`dependentVar`) of a `function` it can be assumed that dependence has been met, since function definitions appear later in the `DAVEfunc` element.

`breakpointDef` — A `DAVEfunc` model can contain zero, one or more breakpoint set definitions. These definitions can be shared among several gridded function tables. Breakpoint definitions can appear in any order.

`griddedTableDef`	A `DAVEfunc` model can contain zero, one, or more gridded nonlinear function table definitions. Each table must be used by at least one but can be used by more than one `function` definition if desired for efficiency. Alternatively, some or all `function` s in a model can specify their tables internally with an embedded `griddedTableDef` element. A gridded function table contains dependent values, or data points, corresponding to the value of a function at the intersection of one or more breakpoint sets (one for each dimension of the table). The independent values (coordinates, or breakpoint sets) are not stored within the gridded table definition but are referenced by the parent function. This allows a function table to be supported by more than one set of breakpoint values (such as left and right aileron deflections).
`ungriddedTableDef`	A DAVEfunc model can contain zero, one, or more ungridded nonlinear function table definitions. Unlike a rectangularly-gridded table, an ungridded table specifies data points as individual sets of independent and dependent values. Each table must be used by at least one but can be used by more than one function definition if necessary for efficiency. Or all functions can retain their tables internally with a ungriddedTable element without sharing the table values with other functions. Ungridded table values are specified as a single (unsorted) list of independent variable (input) values and associated dependent variable (output) values. While the list is not sorted, the order of the independent variable values is important and must match the order given in the using function. Thus, functions that share an ungridded table must have the same ordering of independent variables. The method of interpolating the ungridded data is not specified.
`function`	A `function` ties together breakpoint sets (for gridded-table nonlinear functions), function values (either internally or by reference to table definitions), and the input- and output-variable signal definitions, as shown in figure B-1. Functions also include provenance, or background history, of the function data such as wind tunnel test or other source information.
`checkData`	This optional element contains information allowing the model to be automatically verified after implementation by the receiving party.

An example of each of these sub-elements is given below. Complete descriptions of each element in detail is found in Section B-8 [53].

B-6.2.1. The file header element

The `fileHeader` element contains information about the source of the data contained within the `DAVEfunc` major element, including the author, creation date, description, reference information, and a modification history.

```
fileHeader : [name]
    author+ : name, org, [email]
        contactInfo* : [contactInfoType, contactLocation]
            {text describing contact information for author}
    creationDate : date
    fileVersion?
        {version identification, character data}
    description?
        {description of model, character data}
    reference* : refID, author, title, date, [classification, accession, href]
        description? :
            {written information about reference, character data}
    modificationRecord* : modID, date, [refID]
        author+ : name, org, [email]
            contactInfo* : [contactInfoType, contactLocation]
                {text describing contact information for author}
        description?
                    {description of modification, character data}
        extraDocRef* : refID
    provenance* : [provID]
            author : name, org, [email]
            contactInfo* : [contactInfoType, contactLocation]
                {text describing contact information for author}
            creationDate : date {in YYYY-MM-DD format}
            documentRef* : [docID,] refID
            modificationRef* : modID
            description?
```

`fileHeader` sub-elements:

`author`	Name, organization, optional email address and other contact information for each author.
`creationDate`	Creation date of this file. See Section B-6.5.4 [51] later in this document for the recommended format for encoding dates.
`fileVersion`	A string that indicates the version of the document. No convention is specified for the format, but best practices would include an automated revision number from a configuration control system.
`description`	An optional but recommended text description: what does this DAVE-ML file represent?
`reference`	An optional list of one or more references with a document-unique ID (must begin with alpha character), author, title, date, and optional accession and URL of the reference. Can include a description of the reference.
`modificationRecord`	An optional list of one or more modifications with optional reference pointers, as well as author information and descriptions for each modification record. These modifications are referred to by individual function tables and/or data points, using the AIAA modification letter

	convention. If more than one document is associated with the modification, multiple sub-element extraDocRefs may be used in place of the `modificationRecord` 's refID attribute.
`provenance`	An optional list of one or more provenance elements allows the author to describe the source and history of the data within this model. Since the model may be constructed from several sources, more than one provenance may be provided, one for each source of data.

Example B-1. An file excerpt with an example of a `fileHeader` element

```
<!--                          =================                          --> ❶
<!-- ========================   FILE HEADER   ======================== -->
<!--                          =================                          -->


  <fileHeader> ❷
    <author name="Bruce Jackson" org="NASA Langley Research Center">
      <contactInfo contactInfoType='address' contactLocation='professional'>
         MS 308 NASA, Hampton, VA 23681
      </contactInfo>
      <contactInfo contactInfoType='email' contactLocation='professional'>
        Bruce.Jackson@nospam.nasa.gov
      </contactInfo>
    </author>
    <creationDate date="2003-03-18"/> ❸

    <fileVersion>$Revision: 1.24 $</fileVersion> ❹

    <description>
     Version 2.0 aero model for HL-20 lifting body, as described in
     TM-107580. This aero model was used for HL-20 approach and
     landing studies at NASA Langley Research Center during 1989-1995
     and for a follow-on study at NASA Johnson Space Center in 1994
     and NASA Ames Research Center in 2001. This DAVE-ML version
     created 2003 by Bruce Jackson to demonstrate DAVE-ML.
    </description>

    <reference refID="REF01" ❺
        author="Jackson, E. Bruce; Cruz, Christopher I. & and Ragsdale, W. A."
        title="Real-Time Simulation Model of the HL-20 Lifting Body"
        accession="NASA TM-107580"
        date="1992-07-01"
    />

    <reference refID="REF02"
        author="Cleveland, William B. <nospam@mail.arc.nasa.gov>"
        title="Possible Typo in HL20_aero.xml"
        accession="email"
        date="2003-08-19"
    />

    <modificationRecord modID="A" refID="REF02"> ❻
      <author name="Bruce Jackson" org="NASA Langley Research Center">
        <contactInfo contactInfoType='address' contactLocation='professional'>
           MS 308 NASA, Hampton, VA 23681
        </contactInfo>
      </author>
      <description>
        Revision 1.24: Fixed typo in CLRUD0 function description which
        gave dependent signal name as "CLRUD1." Bill Cleveland of NASA
        Ames caught this in his xml2ftp script. Also made use of 1.5b2
        fileHeader fields and changed date formats to comply with
        convention.
      </description>
    </modificationRecord>

  </fileHeader>
```

❶ Use of comments makes these big files more readable by humans.

❷ Start of `fileHeader` element.

❸ See the note regarding date format convention below.

❹ In this example, the revision number is automatically inserted by CVS or RCS, an automated versioning system.

❺ All documents referenced by notation throughout the file should be described here, in `reference` elements.

❻ All modifications made to the contents of this file should be given here for easy reference in separate `modificationRecord` elements.

B-6.2.2. The variable definition element

The `variableDef` element is used to define each constant, parameter, or variable used within or generated by the defined subsystem model. It contains attributes including the variable name (used for documentation), an internal `varID` identifier (used for automatic code generation), the units of measure of the variable, and optional axis system, sign convention, alias, and symbol declarations. Optional sub-elements include a written text description and a mathematical description, in MathML-2 content markup, of the calculations needed to derive the variable from other variables or function table outputs. An optional sub-element, `isOutput`, serves to indicate an intermediate calculation that should be brought out to the rest of the simulation. Another optional sub-element, `isStdAIAA`, indicates the variable name is defined in the AIAA simulation standards document. A final sub-element, `uncertainty`, captures the statistical properties of a (normally constant) parameter.

There must be a single `variableDef` for each and every input, output or intermediate constant or variable within the DAVEfunc model.

```
variableDef : name, varID, units, [axisSystem, sign, alias, symbol, initialValue]
    description? :
            {description character data}
    EITHER
        provenanceRef? : provID
    OR
        provenance? : [provID]
            author+ : name, org, [email]
                contactInfo* : [contactInfoType, contactLocation]
                    {text describing contact information}
            creationDate : date {in YYYY-MM-DD format}
            documentRef* : [docID,] refID
            modificationRef* : modID
            description?
    calculation? :
        math {defined in MathML2.0 DTD}
    isOutput?
    isState?
    isStateDeriv?
    isStdAIAA?
    uncertainty? : effect
        (normalPDF : numsigmas) | (uniformPDF : bounds+)
```

`variableDef` attributes:

`name`	A UNICODE name for the variable (may be the same string as the `varID`).
`varID`	An XML-legal name that is unique within the file.
`units`	The units-of-measure for the signal, using the AIAA standard units convention.
`axisSystem`	An optional indicator of the axis system (body, inertial, etc.) in which the signal is measured. See Section B-6.5 [51] below for recommended practice for nomenclature.
`sign`	An optional indicator of which direction is considered positive (+RWD, +UP, etc.). See the section on Section B-6.5 [51] below for recommended practice for abbreviations.
`alias`	An optional, facility-specific variable name, perhaps used in the equations of motion or control system model, that does

	not conform to the AIAA standard for variable names. Use of this attribute is discouraged for portability reasons.
`symbol`	A UNICODE Greek symbol for the signal [to be superseded with more formal MathML or TeX element in a later release].
`initialValue`	An optional initial value for the parameter. This is normally specified for constant parameters only.

`variableDef` sub-elements:

`description`	An optional text description of the variable.
`provenance`	The optional provenance element allows the author to describe the source and history of the data within this `variableDef`. Alternatively, a `<provenanceRef>` reference can be made to a previously defined `provenance`.
`calculation`	An optional container for the MathML content markup that describes how this variable is calculated from other variables or function table outputs. This element contains a single `math` element which is defined in the MathML-2 markup language [http://www.w3.org/Math].
`isOutput`	This optional element, if present, signifies that this variable needs to be passed as an output. How this is accomplished is up to the implementer. Unless specified by this element, a variable is considered an output only if it is the result of a calculation or function AND is not used elsewhere in the `DAVEfunc` .
`isStdAIAA`	This optional element, if present, signifies that this variable is one of the standard AIAA simulation variable names that are defined in the (draft) AIAA Simulation Standard Variable Names [AIAA01]. Such identification should make it easier for the importing process to connect this variable (probably an input or output of the model) to the appropriate variable to/from the user's simulation framework.
`isState`	This optional element, if present, signifies that this variable serves as a state of the model.
`isStateDeriv`	This optional element, if present, signifies that this variable serves as a state derivative of the model.
`uncertainty`	This optional element, if present, describes the uncertainty of this parameter. See the section on Statistics below for more information about this element. Note that the `uncertainty` sub-element makes sense only for constant parameters (e.g., those with no `calculation` element but with an `initialValue` specified.

Example B-2. Two excerpts with examples of `variableDef` elements defining input signals

In the excerpts below, two input variables are defined: `XMACH` and `DBFLL`. These two variables are inputs to a table lookup function shown in example B-11 [37] below.

```
<!--                    ==========================                    -->
<!-- ====================   VARIABLE DEFINITIONS   ==================== -->
<!--                    ==========================                    -->


     <!-- ================== -->
     <!--  Input variables   -->
     <!-- ================== -->

  <variableDef name="MachNumber"❶ varID="XMACH"❷ units="" symbol="M">
    <description>
        Mach number (dimensionless)
    </description>
    <isStdAIAA/>❸
  </variableDef>

  <variableDef name="dbfll" varID="DBFLL" units="d"❹ sign="+TED"❺
               symbol="&#x3B4;bfll"❻>
    <description>
        Lower left body flap deflection, deg, +TED (so deflections are
        always zero or positive).
    </description>
  </variableDef>
```

❶ The `name` attribute is intended for humans to read, perhaps as the signal name in a wiring diagram. Note that "MachNumber" is one of the standard AIAA simulation parameter name.

❷ The `varID` attribute is intended for the processing application to read. This must be an XML-valid identifier and must be unique within this model.

❸ The optional `isStdAIAA` sub-element indicates this signal is one of the predefined standard variables that most simulation facilities define in their equations of motion code. The `name` attribute should correspond to the standard AIAA parameter name from [AIAA01] or subsequent standards document

❹ The optional `units` attribute describes the units of measure of the variable. See Section B-6.5.6 [52] below for a recommended list of units-of-measure abbreviations.

❺ The optional `sign` attribute describes the sign convention that applies to this variable. In this case, the lower-left body-flap is positive with trailing-edge-down deflection. See Section B-6.5.5 [51] below for a recommended list of sign abbreviations.

❻ The optional `symbol` attribute allows a UNICODE character string that might be used for this variable in a symbols listing.

Example B-3. A simple local variable

This DAVE-ML excerpt defines CRBFLLO which is the "independent variable" output from the table lookup function shown in example B-11 [37] below.

```
      <!-- ================== -->
      <!--  Local variables   -->
      <!-- ================== -->

<!-- PRELIMINARY BUILDUP EQUATIONS -->

<!--  LOWER LEFT BODY FLAP CONTRIBUTIONS -->

<!--    table output signal   -->
  <variableDef name="Cldbfll_0" varID="CRBFLL0" units="">
    <description>
        Output of CRBFLL0 function; rolling moment contribution of
        lower left body flap deflection due to alpha^0 (constant
        term).
    </description>
  </variableDef>
```

Example B-4. A more complete excerpt using a `calculation` element

Here the local variable `CLBFLL` is defined as a calculated quantity, based on several other input or local variables (not shown). Note the `description` element is used to describe the equation, in FORTRAN-ish human-readable text. The `calculation` element describes this same equation in MathML-2 content markup syntax; this portion should be used by parsing applications to create either source code, documentation, or run-time calculation structures.

```
<!--    lower left body flap lift buildup -->
  <variableDef name="CLdbfll" varID="CLBFLL" units="">
    <description>
        Lift contribution of lower left body flap deflection
        CLdbfll = CLdbfll_0 + alpha*(CLdbfll_1 + alpha*(CLdbfll_2
                            + alpha*CLdbfll_3)) ❶
    </description>
    <calculation>  ❷
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>  ❸
        <plus/>
          <ci>CLBFLL0</ci>
          <apply>
            <times/>
            <ci>ALP</ci>
            <apply>
              <plus/>
              <ci>CLBFLL1</ci>
              <apply>
                <times/>
                <ci>ALP</ci>
                <apply>
                  <plus/>
                  <ci>CLBFLL2</ci>
                  <apply> ❹
                    <times/>
                    <ci>ALP</ci>
                    <ci>CLBFLL3</ci>
                  </apply> <!--             a*c3   -->  ❺
                </apply> <!--        (c2 + a*c3)  -->
              </apply> <!--        a*(c2 + a*c3)  -->
            </apply> <!--    (c1 + a*(c2 + a*c3)) -->
          </apply> <!--    a*(c1 + a*(c2 + a*c3)) -->
        </apply> <!-- c0 + a*(c1 + a*(c2 + a*c3)) -->
      </math>
    </calculation>
  </variableDef>
```

❶ This FORTRAN-ish equation, located in the `description` element, is provided in this example for the benefit of human readers; it should not parsed by the processing application.

❷ A `calculation` element always embeds a MathML-2 `math` element; note the definition of the MathML-2 namespace.

❸ Each `apply` tag pair surrounds a math operation (in this example, a `plus`) operator) and the arguments to that operation (in this case, a variable `CLBFLL` defined elsewhere is added to the results of the nested `apply` operation).

❹ Inner-most apply multiplies variables `ALP` and `CLBFLL3`.

❺ The comments here are useful for humans to understand how the equation is being built up; the processing application ignores all comments.

Example B-5. An output variable based on another `calculation` element

This excerpt is an example of how an output variable (`CL`) might be defined from previously calculated local variables (in this case, `CL0`, `CLBFL`, etc.).

```
    <!-- ================== -->
    <!--  Output variables  -->
    <!-- ================== -->

<variableDef name="CL" varID="CL" units="" sign="+UP" symbol="CL">
  <description>
      Coefficient of lift
      CL = CL0 + CLBFUL + CLBFUR + CLBFLL + CLBFLR +
                 CLWFL + CLWFR + CLRUD + CLGE + CLLG
  </description>
  <calculation>
    <math>
      <apply> ❶
        <plus/>
        <ci>CL0</ci>
        <ci>CLBFUL</ci>
        <ci>CLBFUR</ci>
        <ci>CLBFLL</ci>
        <ci>CLBFLR</ci>
        <ci>CLWFL</ci>
        <ci>CLWFR</ci>
        <ci>CLRUD</ci>
        <ci>CLGE</ci>
        <ci>CLLG</ci>
      </apply>
    </math>
  </calculation>
  <isOutput/> ❷
</variableDef>
```

❶ Here `<apply>` simply sums the value of these variables, referenced by their `varID`s.

❷ The `isOutput` element signifies to the processing application that this variable should be made visible to models external to this DAVEfunc.

Example B-6. An intermediate variable with a `calculation` element that uses a DAVE-ML extension (atan2) to the standard MathML function set

In this excerpt, we demonstrate a means to encode a non-standard MathML-2 math function, atan2. The atan2 function is used often in C, C++, Java and other modeling languages and has been added to the DAVE-ML standard by use of the MathML `csymbol` element, specifically provided to allow extension of MathML for cases such as this.

```
    <!-- ================== -->
    <!--    ATAN2 example   --> ❶
    <!-- ================== -->

<variableDef name="Wind vector roll angle" varID="PHI" units="r">
  <description>
    This encodes the equation PHI = atan2( tan(BETA), sin(ALPHA) ) where atan2
    is the two-argument arc tangent function from the ANSI C standard math
    library.
  </description>
  <calculation>
    <math>
      <apply>
        <csymbol definitionURL="http://daveml.nasa.gov/function_spaces.html#atan2"
                 encoding="text">  ❷
          atan2
        </csymbol>
        <apply>
          <tan/>
          <ci>BETA</ci>   ❸
        </apply>
        <apply>
          <sin/>
          <ci>ALPHA</ci>  ❹
        </apply>
      </apply>
    </math>
  </calculation>

</variableDef>
```

❶ This excerpt shows how to calculate wind roll angle, phi, from angle of attack and angle of sideslip; it comes from the Apollo aero data book [NAA64].

❷ The `csymbol` element is provided by MathML-2 as a means to extend the function set of MathML. Only a limited set of extensions given in this Standard are supported but others may be added to the standard in later versions. Note the specific URL that uniquely identifies this function; it is also the address of the documentation of the interpretation of the atan2 function.

❸ BETA is the `varID` of a previously defined variable.

❹ ALPHA is the `varID` of a previously defined variable.

B-6.2.3. The breakpoint set definition element

The breakpoint set definition element, `breakpointDef` , is used to define a list of comma-separated values that define the coordinate values along one axis of a gridded linear function value table. It contains a mandatory `bpID` attribute, an optional `name` and units-of-measure attributes, an optional text `description` element and the comma-separated list of floating-point values in the `bpVals` element. This list must be monotonically increasing in value.

```
breakpointDef* : bpID, [name, units]
    description? :
    bpVals :
        {character data of comma-separated breakpoints}
```

`breakpointDef` attributes:

`bpID`	An XML-legal name that is unique within the file.
`name`	A UNICODE name for the set (may be the same string as `bpID`).
`units`	The units-of-measure for the breakpoint values. See Section B-6.5.6 [52] below.

`breakpointDef` sub-elements:

`description`	An optional text description of the breakpoint set.
`bpVals`	A comma-separated, monotonically-increasing list of floating-point values.

Example B-7. Two excerpts with examples of `breakpointDef` elements

Two breakpoint sets are defined which are used in the `function` element given below (example B-11 [37]). Breakpoint sets `XMACH1_PTS` and `DBFL_PTS` contain values for Mach and lower body flap deflection, respectively, which are used to look up function values in several gridded function tables; one example is given below in example B-8 [29].

```
<!--                          ====================                          -->
<!-- ========================   BREAKPOINT SETS   ======================== -->
<!--                          ====================                          -->


  <breakpointDef name="Mach" bpID="XMACH1_PTS" units="">  ❶
    <description>
      Mach number breakpoints for all aero data tables
    </description>
    <bpVals>
        0.3, 0.6, 0.8, 0.9, 0.95, 1.1, 1.2, 1.6, 2.0, 2.5, 3.0, 3.5, 4.0   ❷
    </bpVals>
  </breakpointDef>

  <breakpointDef name="Lower body flap" bpID="DBFL_PTS" units="d">
    <description>Lower body flap deflections breakpoints for tables</description>
    <bpVals>0., 15., 30., 45., 60.</bpVals>
  </breakpointDef>
```

❶ This `breakpointDef` element describes a Mach breakpoint set uniquely identified as `XMACH1_PTS` with no associated units of measure.

❷ The breakpoint values are given as a comma-separated list and must be in monotonically increasing numerical order.

B-6.2.4. The gridded table definition element

The `griddedTableDef` element defines a multi-dimensional table of values corresponding with the value of an arbitrary function at the intersection of a set of specified independent inputs. The coordinates along each dimension are defined in separate `breakpointDef` elements that are referenced within this element by `bpRef` s, one for each dimension.

The data contained within the data table definition are a comma-separated set of floating-point values. This list of values represents a multi-dimensional array whose size is inferred from the length of each breakpoint vector. For example, a two-dimensional table that is a function of an eight-element Mach breakpoint set and a ten-element angle-of-attack breakpoint set is expected to contain 80 comma-separated values.

By convention, the `breakpointRefs` are listed in order such that the last breakpoint set varies most rapidly in the associated data table listing. See Section B-6.5.1 [51] below.

An optional `uncertainty` element may be provided that represents the statistical variation in the values presented. See the section on Statistics below for more information about this element.

```
griddedTableDef : [name, gtID, units]
    description?
      {description of table in character data}
    EITHER
        provenanceRef? : provID
    OR
        provenance? : [provID]
            author+ : name, org, [email]
                contactInfo* : [contactInfoType, contactLocation]
                    {text describing contact information}
            creationDate : date {in YYYY-MM-DD format}
            documentRef* : [docID,] refID
            modificationRef* : modID
            description?
    breakpointRefs :
        bpRef+
    uncertainty? : effect
        (normalPDF : numSigmas | uniformPDF)
    dataTable
      {character data of comma-separated table values}
```

`griddedTableDef` attributes:

`gtID`	An XML-legal name that is unique within the file.
`name`	A UNICODE name for the table (may be the same string as `gtID`).
`units`	The units-of-measure for the table's output signal. See Section B-6.5.6 [52] below.

`griddedTableDef` sub-elements:

`description`	The optional description element allows the author to describe the data contained within this `griddedTable`.
`provenance`	The optional provenance element allows the author to describe the source and history of the data within this `griddedTable`. Alternatively, a `<provenanceRef>` reference can be made to a previously defined `provenance`.
`breakpointRefs`	The mandatory `breakpointRefs` element contains separate `bpRef` elements, each pointing to a separately-defined `breakpointDef` . Thus, the independent coordinates associated with this function table are defined elsewhere and only a reference is given here. The order of appearance of the `bpRef` s is important; see the text above.
`uncertainty`	This optional element, if present, describes the uncertainty of this parameter. See the section on Statistics below for more information about this element.
`dataTable`	The numeric values of the function at the function vertices specified by the breakpoint sets are contained within this element, in a single comma-separated list. Parsing this list

and storing it in the appropriate array representation is up to the implementor. By convention, the last breakpoint value increases most rapidly.

Example B-8. An excerpt showing an example of a `griddedTableDef` element

This non-linear function table is used by a subsequent `function` in example B-11 [37] to specify an output value based on two inputs values - body flap deflection and Mach number. This table is defined outside of a `function` element because this particular function table is used by two functions - one for the left lower body flap and one for the right lower body flap; thus, their actual independent (input) variable values might be different.

```
        <!-- ================================== -->     ❶
        <!-- Lower Body Flap Tables (definitions) -->
        <!-- ================================== -->

  <griddedTableDef name="CLBFL0" gtID="CLBFL0_table">     ❷
    <description>      ❸
        Lower body flap contribution to lift coefficient,
        polynomial constant term
    </description>
    <provenance>        ❹
      <author name="Bruce Jackson" org="NASA Langley Research Center"
 email="e.b.jackson@larc.nasa.gov"/>
      <creationDate date="2003-01-31"/>
      <documentRef docID="REF01"/>
    </provenance>
    <breakpointRefs> ❺
      <bpRef bpID="DBFL_PTS"/>
      <bpRef bpID="XMACH1_PTS"/>
    </breakpointRefs>
    <dataTable> <!-- last breakpoint (XMACH) changes most rapidly --> ❻
<!-- CLBFL0 POINTS  -->
<!-- DBFL =        0.0       -->
 0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 ,
 0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 ,
 0.00000E+00 , 0.00000E+00 , 0.00000E+00 ,
<!-- DBFL =       15.0       --> ❼
-0.86429E-02 ,-0.10256E-01 ,-0.11189E-01 ,-0.12121E-01 ,-0.13520E-01 ,
-0.86299E-02 ,-0.53679E-02 , 0.76757E-02 , 0.11300E-01 , 0.62992E-02 ,
 0.51902E-02 , 0.38813E-02 , 0.37366E-02 ,
<!-- DBFL =       30.0       -->
 0.22251E-01 , 0.26405E-01 , 0.28805E-01 , 0.31206E-01 , 0.34806E-01 ,
 0.31321E-01 , 0.28996E-01 , 0.19698E-01 , 0.18808E-01 , 0.12755E-01 ,
 0.10804E-01 , 0.98493E-02 , 0.83719E-02 ,
<!-- DBFL =       45.0       -->
   .
   . [other points in table]
   .
    </dataTable>
  </griddedTableDef>
```

❶ Comments are a good idea for human readers

❷ `name` is used for documentation purposes; `gtID` is intended for automatic wiring (autocode) tools.

❸ Descriptions are a good idea whenever possible - Here we explain the contents of the function represented by the data points.

❹ `provenance` is the story of the origin of the data.

❺ These `bpRef`s are in the same order as the table is wrapped (see text above) and must be reflected in the referencing function in the same order. In this excerpt, the referencing function must list the `independentVarRefs` such that the signal that represents delta body flap (`DBFL`) must precede the reference to the signal that represents Mach number (`XMACH`).

❻ The points listed within the `dataTable` element are given as if the last `bpRef` varies most rapidly. See the discussion above.

❼ Embedded comments are a good idea.

B-6.2.5. The ungridded table definition element

The `ungriddedTableDef` element defines a set of non-orthogonal data points, along with their independent values (coordinates), corresponding with the dependent value of an arbitrary function.

A 'non-orthogonal' data set, as opposed to a gridded or 'orthogonal' data set, means that the independent values are not laid out in an orthogonal grid. This form must be used if the dependent coordinates in any table dimension cannot be expressed by a single monotonically-increasing vector.

See the excerpts below for two instances of ungridded data.

An optional `uncertainty` element may be provided that represents the statistical variation in the values presented. See the section on Statistics below for more information about this element.

```
ungriddedTableDef* : [utID, name, units]
    description? :
        {description character data}
    EITHER
        provenanceRef? : provID
    OR
        provenance? : [provID]
            author+ : name, org, [email]
                contactInfo* : [contactInfoType, contactLocation]
                    {text describing contact information}
            creationDate : date {in YYYY-MM-DD format}
            documentRef* : [docID,] refID
            modificationRef* : modID
            description?
    uncertainty? : effect
        (normalPDF : numsigmas) | (uniformPDF : bounds+)
    dataPoint+ :
```

`ungriddedTableDef` attributes:

`utID`	A mandatory XML-legal name that is unique within the file
`name`	An optional UNICODE name for the table (may be the same string as `utID`).
`units`	Optional units-of-measure for the table's output signal.

`ungriddedTableDef` sub-elements:

`description`	The optional description element allows the author to describe the data contained within this `ungriddedTable`.
`provenance`	The optional provenance element allows the author to describe the source and history of the data within this `ungriddedTable`. Alternatively, a `<provenanceRef>` reference can be made to a previously defined provenance.
`uncertainty`	This optional element, if present, describes the uncertainty of this parameter. See the section on Statistics below for more information about this element.
`dataPoint`	One or more sets of coordinate and output numeric values of the function at various locations within it's input space.

This element includes one coordinate for each function input variable. Parsing this information into a usable interpolative function is up to the implementor. By convention, the coordinates are listed in the same order that they appear in the using function.

Example B-9. An excerpt showing an `ungriddedTableDef` element, encoding the data depicted in here.

This two-dimensional function table is an example provided by Dr. Peter Grant of the University of Toronto. Such a table definition would be used in a subsequent `function` to describe how an output variable would be defined based on two independent input variables. The function table doesn't indicate which input and output variables are represented; this information is supplied by the `function` element later so that a single function table can be reused by multiple functions.

```
<ungriddedTableDef name="CLBASIC as function of flap angle and angle of
       attack" utID="CLBAlfaFlap_Table" units="">
  <description>
    CL basic as a function of flap angle and angle of attack. Note the alpha
    used in this table is with respect to the wing design plane (in degrees).
    Flap is in degrees as well.
  </description>

  <provenance>
    <author name="Peter Grant" org="UTIAS"/>  ❶
    <creationDate date="2006-11-01"/>
    <documentRef refID="PRG1" />
  </provenance>

  <!--For ungridded tables you provide a list of dataPoints--> ❷
       <dataPoint> 1.0 -5.00  -0.44 <!-- flap, alfawdp, CLB--></dataPoint>   ❸
       <dataPoint> 1.0  10.00  0.95 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 1.0  12.00  1.12 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 1.0  14.00  1.26 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 1.0  15.00  1.32 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 1.0  17.00  1.41 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 5.0 -5.00  -0.55 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 5.0  0.00  -0.03 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 5.0  5.00   0.50 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 5.0  10.00  1.02 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 5.0  12.00  1.23 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 5.0  14.00  1.44 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 5.0  16.00  1.63 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 5.0  17.00  1.70 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 5.0  18.00  1.75 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint modID='A'> 10.0 -5.00 -0.40 <!-- flap, alfawdp, CLB--></dataPoint> ❹
       <dataPoint> 10.0 14.00  1.57 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 10.0 15.00  1.66 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 10.0 16.00  1.75 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 10.0 17.00  1.80 <!-- flap, alfawdp, CLB--></dataPoint>
       <dataPoint> 10.0 18.00  1.84 <!-- flap, alfawdp, CLB--></dataPoint>

</ungriddedTableDef>
```

❶ Example courtesy of Dr. Peter Grant, U. Toronto

❷ Comments are a good idea for human readers

❸ For a two-dimensional table such as this one, data points give two columns of independent breakpoint values and third column of function value at those breakpoints.

❹ The `modID` attribute implies this point was edited during modification 'A' of this model, as described in the file header information.

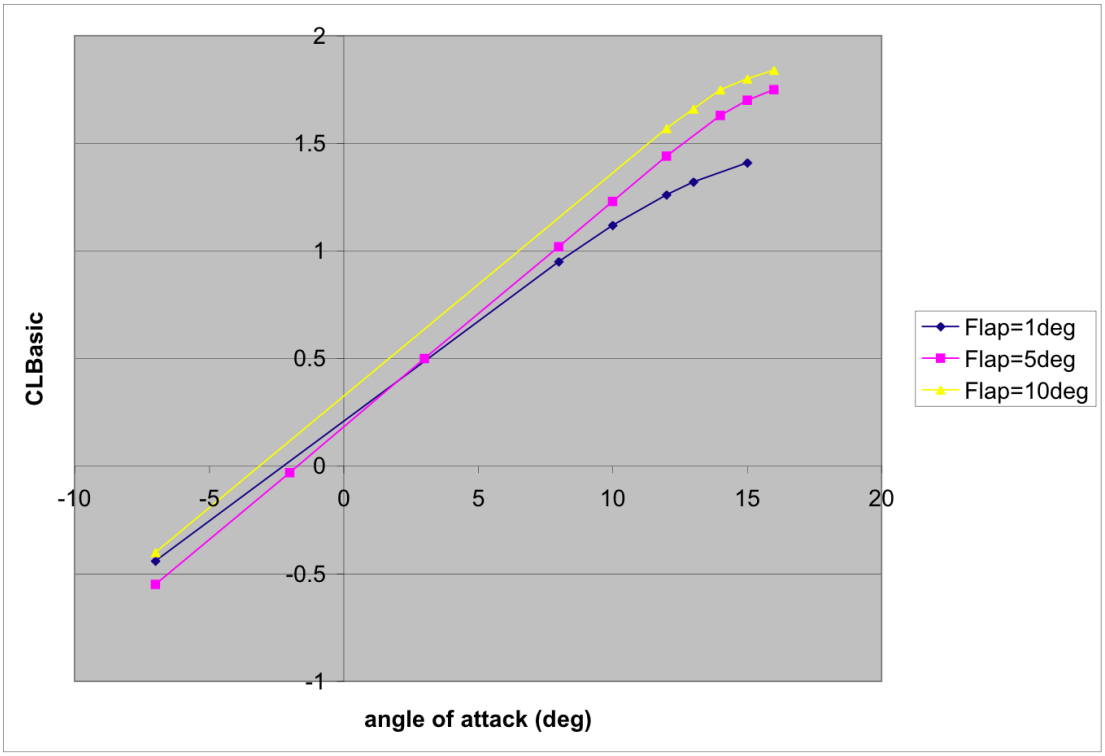


Figure B-2. The two-dimensional lift function given in example B-9 [31]

Example B-10. An excerpt from a sample aero model giving an example of a three-dimensional `ungriddedTableDef` element, encoding the data shown in figure B-3 [34].

In this example, the dependent coordinates all vary in each dimension.

```
    <!--====================================-->  ❶
    <!--  Three-D Table Definition Example -->
    <!--====================================-->

    <ungriddedTableDef name="yawMomentCoefficientTable1" units=""
utID="yawMomentCoefficientTable1">
              <!--  alpha,     beta,       delta -->    ❷
      <dataPoint> -1.8330592 -5.3490387 -4.7258599 -0.00350641</dataPoint>
      <dataPoint> -1.9302179 -4.9698462 0.2798654  -0.0120538</dataPoint>
      <dataPoint> -2.1213095 -5.0383145 5.2146443  -0.0207089</dataPoint>
      <dataPoint> 0.2522004  -4.9587161 -5.2312860 -0.000882368</dataPoint>
      <dataPoint> 0.3368831  -5.0797159 -0.3370540 -0.0111846</dataPoint>
      <dataPoint> 0.2987289  -4.9691198 5.2868938  -0.0208758</dataPoint>
      <dataPoint> 1.8858257  -5.2077654 -4.7313074 -0.00219842</dataPoint>
      <dataPoint> 1.8031083  -4.7072954 0.0541231  -0.0111726</dataPoint>
      <dataPoint> 1.7773659  -5.0317988 5.1507477  -0.0208074</dataPoint>
      <dataPoint> 3.8104785  -5.2982162 -4.7152852 -0.00225906</dataPoint>
      <dataPoint> 4.2631596  -5.1695257 -0.1343410 -0.0116563</dataPoint>
      <dataPoint> 4.0470946  -5.2541017 5.0686926  -0.0215506</dataPoint>
      <dataPoint> -1.8882611 0.2422452  -5.1959304 0.0113462</dataPoint>
      <dataPoint> -2.1796091 0.0542085  0.2454711  -0.000253915</dataPoint>
      <dataPoint> -2.2699103 -0.3146657 4.8638859  -0.00875431</dataPoint>
      <dataPoint> 0.0148579  0.1095599  -4.9639500 0.0105144</dataPoint>
      <dataPoint> -0.1214591 -0.0047960 0.2788827  -0.000487753</dataPoint>
      <dataPoint> 0.0610233  0.2029588  5.0831767  -0.00816086</dataPoint>
      <dataPoint> 1.7593356  -0.0149007 -5.0494446 0.0106762</dataPoint>
      <dataPoint> 1.9717048  -0.0870861 0.0763833  -0.000332616</dataPoint>
      <dataPoint> 2.0228263  -0.2962294 5.1777078  -0.0093807</dataPoint>
      <dataPoint> 4.0567507  -0.2948622 -5.1002243 0.010196</dataPoint>
      <dataPoint> 3.6534822  0.2163747  0.1369900  0.000312733</dataPoint>
      <dataPoint> 3.6848003  0.0884533  4.8214805  -0.00809437</dataPoint>
      <dataPoint> -2.3347682 5.2288720  -4.7193014 0.02453</dataPoint>
      <dataPoint> -2.3060350 4.9652745  0.2324610  0.0133447</dataPoint>
      <dataPoint> -1.8675176 5.0754646  5.1169942  0.00556052</dataPoint>
      <dataPoint> 0.0004379  5.1220145  -5.2734993 0.0250468</dataPoint>
      <dataPoint> -0.1977035 4.7462188  0.0664495  0.0124083</dataPoint>
      <dataPoint> -0.1467742 5.0470092  5.1806131  0.00475277</dataPoint>
      <dataPoint> 1.6599338  4.9352809  -5.1210532 0.0242646</dataPoint>
      <dataPoint> 2.2719825  4.8865093  0.0315210  0.0125658</dataPoint>
      <dataPoint> 2.0406858  5.3253471  5.2880688  0.00491779</dataPoint>
      <dataPoint> 4.0179983  5.0826426  -4.9597629 0.0243518</dataPoint>
      <dataPoint> 4.2863811  4.8806558  -0.2877697 0.0128886</dataPoint>
      <dataPoint> 3.9289361  5.2246849  4.9758705  0.00471241</dataPoint>
      <dataPoint> -2.2809763 9.9844584  -4.8800790 0.0386951</dataPoint>
      <dataPoint> -2.0733070 9.9204337  0.0241722  0.027546</dataPoint>
      <dataPoint> -1.7624546 9.9153493  5.1985794  0.0188357</dataPoint>
      <dataPoint> 0.2279962  9.8962508  -4.7811258 0.0375762</dataPoint>
      <dataPoint> -0.2800363 10.3004593 0.1413907  0.028144</dataPoint>
      <dataPoint> 0.0828562  9.9008011  5.2962722  0.0179398</dataPoint>
      <dataPoint> 1.8262230  10.0939436 -4.6710211 0.037712</dataPoint>
      <dataPoint> 1.7762123  10.1556398 -0.1307093 0.0278079</dataPoint>
      <dataPoint> 2.2258599  9.8009720  4.6721747  0.018244</dataPoint>
      <dataPoint> 3.7892651  9.8017197  -4.8026383 0.0368199</dataPoint>
      <dataPoint> 4.0150716  9.6815531  -0.0630955 0.0252014</dataPoint>
      <dataPoint> 4.1677953  9.8754433  5.1776223  0.0164312</dataPoint>
    </ungriddedTableDef>
```

❶ Example courtesy of Mr. Geoff Brian, DSTO

❷ Columns are labelled with an XML comment for human readers; association of each input (alpha and beta) and the single output (delta) with the function inputs and output happens in the context of the referring function definition(s) which gives independent variables and the dependent (output) variable.

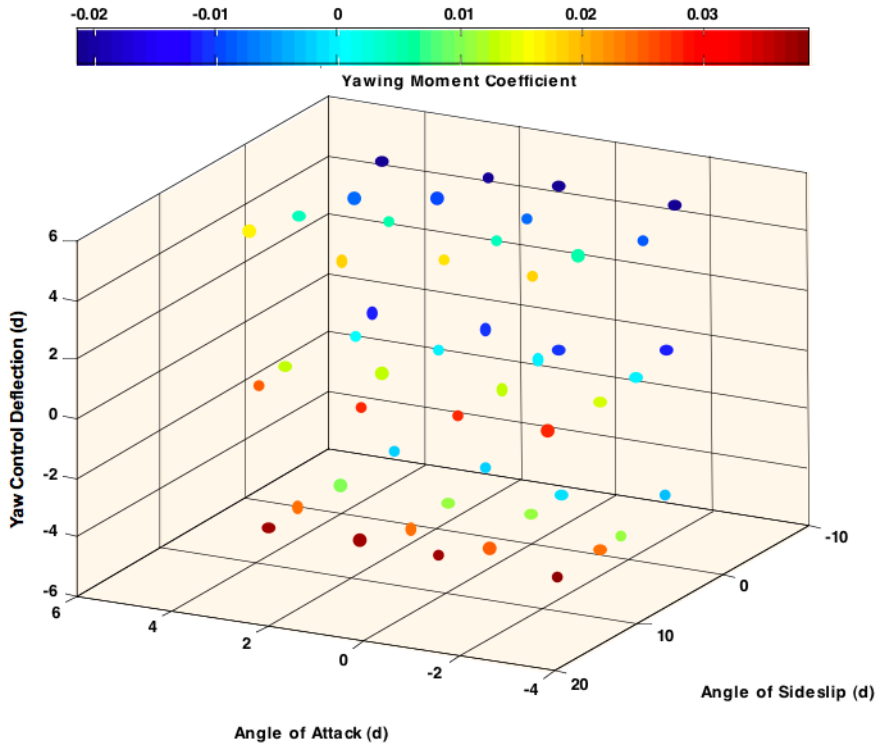


Figure B-3. The three-dimensional function given in the previous example

B-6.2.6. The function definition element

The `function` element connects breakpoint sets (for gridded tables), independent variables, and data tables to their respective output variable.

```
function* : name
    description? :
    EITHER
        provenanceRef? : provID
    OR
        provenance? : [provID]
            author+ : name, org, [email]
                contactInfo* : [contactInfoType, contactLocation]
                    {text describing contact information}
            creationDate : date {in YYYY-MM-DD format}
            documentRef* : [docID,] refID
            modificationRef* : modID
            description?
    EITHER
    {
      independentVarPts+ : varID, [name, units, sign, extrapolate, interpolate]
          {input values as character data}
      dependentVarPts : varID, [name, units, sign]
          {output values as character data}
    }
    OR
    {
      independentVarRef+ : varID, [min, max, extrapolate, interpolate]
      dependentVarRef : varID
      functionDefn : [name]
          CHOICE OF
          {
```

```
            griddedTableRef : gtID
        OR
            griddedTableDef : [name, gtID, units]
    description?
                {text description of table}
              (provenance | provenanceRef)?   [as described earlier]
              breakpointRefs
                  bpRef+ : bpID
              uncertainty? : effect
                  (normalPDF : numsigmas) | (uniformPDF : bounds+)
              dataTable
                  {gridded data table as character data}
        OR
            ungriddedTableRef : utID
        OR
            ungriddedTableDef : [name, utID, units]
    description?
                {text description of table}
              (provenance | provenanceRef)?   (as described earlier)
              uncertainty? : effect
                  (normalPDF : numsigmas) | (uniformPDF : bounds+)
              dataPoint+
                {coordinate/value sets as character data}
        }
  }
```

`function` attributes:

`name` — A UNICODE name for the function.

`function` sub-elements:

`description` — The optional description element allows the author to describe the data contained within this `function`.

`provenance` — The optional provenance element allows the author to describe the source and history of the data within this `function`. Alternatively, a `<provenanceRef>` reference can be made to a previously defined provenance.

`independentVarPts` — If the author chooses, [he|she] can express a linearly-interpolated functions by specifying the independent (breakpoint) values sets as one or more `independentVarPts` which are comma-separated, monotonically increasing floating-point coordinate values corresponding to the `dependentVarPts` given next. In the case of multiple dimensions, more than one `independentVarPts` must be specified, one for each dimension. The mandatory `varID` attribute is used to connect each `independentVarPts` with an input variable.

An optional 'interpolate' attribute specifies the preference for using linear, quadratic, or cubic relaxed splines for calculating dependent values when the independent arguments are in between specified values. When not specified, the expectation would be a linear spline interpolation between points. The performance of interpolation of various orders is left up to the processing application. More information on relaxed spline interpolation may be found in [wiki01].

`dependentVarPts` — This element goes along with the previous element to specify a function table. Only one `dependentVarPts` may be

	specified. If the function is multi-dimensional, the convention is the last breakpoint dimension changes most rapidly in this comma-separated list of floating-point output values. The mandatory `varID` attribute is used to connect this table's output to an output variable.
`independentVarRef`	One or more of these elements refer to separately-defined `variableDef` s. For multi-dimensional tables, the order of specification is important and must match the order in which breakpoints are specified or the order of coordinates in ungridded table coordinate/value sets. An optional 'interpolate' attribute specifies the preference for using discrete, linear, quadratic, or cubic splines for calculating dependent values when the independent arguments are in between specified values. When not specified, the expectation would be a linear spline interpolation between points. The performance of interpolation of various orders is left up to the processing application. See the section below on interpolation. More information on quadratic and cubic spline interpolation may be found in [wiki01].
`dependentVarRef`	One `dependentVarRef` must be specified to connect the output of this function to a particular `variableDef`.
`functionDefn`	This element identifies either a separately-specified data table definition or specifies a private table, either gridded or ungridded.
`griddedTableRef`	If not defining a simple function table, the author may use this element to point to a separately-specified `griddedTableDef` element.
`griddedTable`	As an alternative to reutilization of a previously defined table, this element may be used to define a private output gridded table. See the writeup on `griddedTableDef` for more information. [Deprecated: use of this element is discouraged and will not be supported in future DAVE-ML versions. Use a `griddedTableDef` instead.]
`ungriddedTableRef`	If not using a simple function table, the author may use this element to point to separately-specified `ungriddedTableDef` element.
`ungriddedTable`	As an alternative to reuse of a previously defined table, this element may be used to define a private output ungridded table. See the writeup on `ungriddedTableDef` for more information. [Deprecated: use of this element is discouraged and will not be supported in future DAVE-ML versions. Use an `griddedTableDef` instead.]

Example B-11. An excerpt giving the example of a function which refers to a previously defined `griddedTableDef`

This example ties the input variables `DBFLL` and `XMACH` into output variable `CLBFLLO` through a function called `CLBFLO_fn`, which is represented by the linear interpolation of the gridded table previously defined by the `CLBFL0_table` `griddedTableDef` (see the griddedTableDef example above).

```
    <!-- ============================== -->
    <!-- Lower left body flap functions -->
    <!-- ============================== -->

<function name="CLBFLL0">
  <description>
    Lower left body flap lookup function for lift, polynomial constant term.
  </description>
  <independentVarRef varID="DBFLL"  min="0.0" max="60." extrapolate="neither"/> ❶
  <independentVarRef varID="XMACH" min="0.3" max="4.0" extrapolate="neither"/>
  <dependentVarRef varID="CLBFLL0"/> ❷
  <functionDefn name="CLBFL0_fn">
    <griddedTableRef gtID="CLBFL0_table"/> ❸
  </functionDefn>
</function>
```

❶ The independent variables must be given in the order of least-rapidly-changing to most-rapidly-changing values in the previously defined function table. The processing application needs to pay attention to the `extrapolate` attribute, which specifies how to treat a variable whose value exceeds the stated limits on input.

❷ The dependent variable (XML name `CLBFLL0`) is the output variable for this function. `CLBFLL0` must have been declared previously with a `variableDef` element.

❸ This is a reference to the previously declared `griddedTableDef`.

Example B-12. A `function` that has an internal table

In this example, the function `CLRUD0` returns, in the variable `CLRUD0`, the value of function `CLRUD0_fn` represented by gridded table `CLRUD0_table`. The inputs to the function are `abs_rud` and `XMACH` which are used to normalize breakpoint sets `DRUD_PTS` and `XMACH1_PTS` respectively. The input variables are limited between 0.0 to 15.0 and 0.3 to 4.0, respectively.

In this case, the use of `CLRUD0` string for both the function `name` attribute and as the `varID` for the dependent (output) variable reference do not interfere (although they are confusing); `namess` are not in the XML namespace. The `name` attribute is only used for documentation (such as a label for a box representing this function).

```
      <!-- ================ -->
      <!-- Rudder functions -->
      <!-- ================ -->

<!-- The rudder functions are only used once, so their table
     definitions are internal to the function definition.
--> ❶

  <function name="CLRUD0">
    <description>
        Rudder contribution to lift coefficient,
        polynomial multiplier for constant term.
    </description>
    <provenance> ❷
      <author name="Bruce Jackson" org="NASA Langley Research Center"
 email="e.b.jackson@larc.nasa.gov"/>
      <creationDate date="2003-01-31"/>
      <documentRef docID="REF01"/>
    </provenance>
    <independentVarRef varID="abs_rud"  min="0.0" max="15." extrapolate="neither"/>
    <independentVarRef varID="XMACH" min="0.3" max="4.0" extrapolate="neither"/>
    <dependentVarRef varID="CLRUD0"/>

    <functionDefn name="CLRUD0_fn">
      <griddedTableDef name="CLRUD0_table"> ❸
        <breakpointRefs>
          <bpRef bpID="DRUD_PTS"/>
          <bpRef bpID="XMACH1_PTS"/>
        </breakpointRefs>
        <dataTable> <!-- last breakpoint changes most rapidly -->
<!-- CLRUD0  POINTS  -->
<!-- RUD =    0.0    -->
 0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 ,
 0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 ,
 0.00000E+00 , 0.00000E+00 , 0.00000E+00 ,
<!-- RUD =   15.0    -->
-0.13646E-01 , 0.26486E-01 , 0.16977E-01 ,-0.16891E-01 , 0.10682E-01 ,
 0.75071E-02 , 0.53891E-02 ,-0.30802E-02 ,-0.59013E-02 ,-0.95733E-02 ,
 0.00000E+00 , 0.00000E+00 , 0.00000E+00 ,
<!-- RUD =   30.0    -->
-0.12709E-02 , 0.52971E-01 , 0.33953E-01 ,-0.33782E-01 , 0.21364E-01 ,
 0.15014E-01 , 0.10778E-01 ,-0.61604E-02 ,-0.11803E-01 ,-0.19147E-01 ,
 0.00000E+00 , 0.00000E+00 , 0.00000E+00
        </dataTable>
      </griddedTable>
    </functionDefn>
  </function>
```

❶ This comment helps humans understand the reason for an embedded table.
❷ The `provenance` element is required by the AIAA standard.
❸ This example has an embedded gridded table.

Example B-13. A simple one-dimensional `function`

At the other end of the spectrum, a simple N-d nonlinear function can be defined, with no reuse, as follows:

```
<function name="CL">
  <independentVarPts varID="alpdeg"> ❶
   -4.0, 0., 4.0, 8.0, 12.0, 16.0
  </independentVarPts>
  <dependentVarPts varID="cl"> ❷
    0.0, 0.2, 0.4, 0.8, 1.0, 1.2
  </dependentVarPts>
</function>
```

❶ Breakpoints in angle-of-attack are listed here.
❷ Values of `cl` are given, corresponding to the angle-of-attack breakpoints given previously.

B-6.2.7. The checkData element

The `checkData` element contains one or more input/output vector pairs (and optionally a dump of internal values) for the encoded model to assist in verification and debugging of the implementation.

```
    checkData
 staticShot* : name, [refID]
     description?
       {text description of the static test case}
         (provenance | provenanceRef)?   {as defined previously}
     checkInputs
  signal+
      signalName
      signalUnits
      signalValue
     internalValues?
  signal+
      EITHER
   signalName
   signalUnits
      OR
   varID
      signalValue
     checkOutputs
  signal+
      signalName
      signalUnits
      signalValue
      tol
```

`checkData` sub-elements:

`staticShot`	One or more check-case data sets, which each contain mandatory subelements `<checkInputs>` and `<checkOutputs>` vectors (with required match tolerances), optional `<provenance>` , `<provenanceRef>` , `<description>` and `<internalValues>` subelements.

Example B-14. Check-case data set excerpt

A DAVE-ML file excerpt specifying a check-case data set example for a simple model

```
<checkData>
  <staticShot name="Nominal" refID="NOTE1"> ❶
    <description>An example static check of a simple DAVE-ML model</description>
    <checkInputs>  ❷
      <signal>  ❸
        <signalName>trueAirspeed</signalName>
        <signalUnits>f/s</signalUnits>
        <signalValue> 300.000</signalValue>
      </signal>
      <signal>
        <signalName>angleOfAttack</signalName>
        <signalUnits>d</signalUnits>
        <signalValue>   5.000</signalValue>
      </signal>
       .
       .       (similar input signals omitted)
       .
      <signal>
        <signalName>delta elevator</signalName>
        <signalUnits>d</signalUnits>
        <signalValue>   0.000</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs> ❹
      <signal>  ❺
        <signalName>CX</signalName>
        <signalUnits/>   ❻
        <signalValue>-0.00400000000000</signalValue>
        <tol>0.000001</tol>
      </signal>
       .
       .       (similar output signals omitted)
       .
    </checkOutputs>
  </staticShot>
  <staticShot name="Positive pitch rate"> ❼
    <checkInputs>
       .
       .       (similar input and output signal information omitted)
       .
    </checkOutputs>
  </staticShot>
  <staticShot name="Positive elevator">
    <checkInputs>
       .
       .       (similar input and output signal information omitted)
       .
    </checkOutputs>
  </staticShot>
</checkData>
```

❶ This first check case refers to a note given in the file header; this is useful to document the source of the check case values.

❷ The checkInputs element defines the input variable values, by variable name, as well as units (so they can be verified).

❸ Multiple variable values are given, constituting the input "vector."

❹ The checkOutputs element defines output variable values that should result from the specified input values.

❺ Note the included tolerance value, within which the output values must match the check-case data values.

❼ Multiple check cases may be specified; this one differs from the previous check-case due to an increase in the pitching rate input.

❻ Empty `signalUnits` implies a non-dimensional signal.

Example B-15. A second `checkData` example with internal values given

```
 <checkData>
   <staticShot name="Skewed inputs">
     <description>
       Another example static check; this one includes all the internal, intermediate
calculations
       to assist in debugging the implementation.
     </description>
     <checkInputs>
       <signal>
         <signalName>trueAirspeed</signalName>
         <signalUnits>fs_1</signalUnits>\
         <signalValue> 300.000</signalValue>
       </signal>
       <signal>
         <signalName>angleOfAttack</signalName>
         <signalUnits>d</signalUnits>
         <signalValue>  16.200</signalValue>
       </signal>
        .
        .       (similar input values omitted)
        .
       <signal>
         <signalName>XBodyPositionOfCG</signalName>
         <signalUnits>fracMAC</signalUnits>
         <signalValue>   0.123</signalValue>
       </signal>
     </checkInputs>
     <internalValues> ❶
       <signal>
         <varID>vt</varID>
         <signalValue>300.0</signalValue>
       </signal>
       <signal>
         <varID>alpha</varID>
         <signalValue>16.2</signalValue>
       </signal>
        .
        .       (similar internal values omitted)
        .
     </internalValues>
     <checkOutputs>
       <signal>
         <signalName>aeroXBodyForceCoefficient</signalName>
         <signalValue> 0.04794994533333</signalValue>
         <signalUnits/>
         <tol>0.000001</tol>
       </signal>
       <signal>
         <signalName>aeroZBodyForceCoefficient</signalName>
         <signalValue>-0.72934852554344</signalValue>
         <signalUnits/>
         <tol>0.000001</tol>
       </signal>
       <signal>
         <signalName>aeroPitchBodyMomentCoefficient</signalName>
         <signalValue>-0.10638585796503</signalValue>
         <signalUnits/>
         <tol>0.000001</tol>
       </signal>
     </checkOutputs>
   </staticShot>
 </checkData>
```

❶ A dump of all model-defined variable values is included in this example to aide in debugging the implementation by the recipient.

B-6.3. Function interpolation/extrapolation

It is possible to specify the method of interpolation to be used for non-linear function tables by use of the `interpolation` attribute of the `<independentVarPts>` and `<independentVarRef>` elements. This attribute, combined with the 'extrapolate' flag, provide several different ways of realizing the intermediate values of the function when not at one of the specified intersections of independent values.

Implementation of the specific interpolation algorithm is left up to the application, however, the following implementation notes are suggested:

- An infinite set of quadratic interpolations are possible; it is suggested to use the one that minimizes either the deviation from a linear interpolation or the slope error at any edge.
- For cubic interpolation, the natural cubic spline (which has a second derivative of zero at each end) implementation is suggested when the 'extrapolate' attribute is 'none'. When the 'extrapolate' attribute is 'both', a clamped cubic spline that matches the extrapolation slopes is suggested.
- For the discrete interpolation values (discrete, ceiling, or floor) the value of the 'extrapolate' attribute is meaningless.

The figures below give nine different examples for a one-dimensional table whose independent values are [1, 3, 4, 6, 7.5] with dependent values of [2, 6, 5, 7, 1.5].

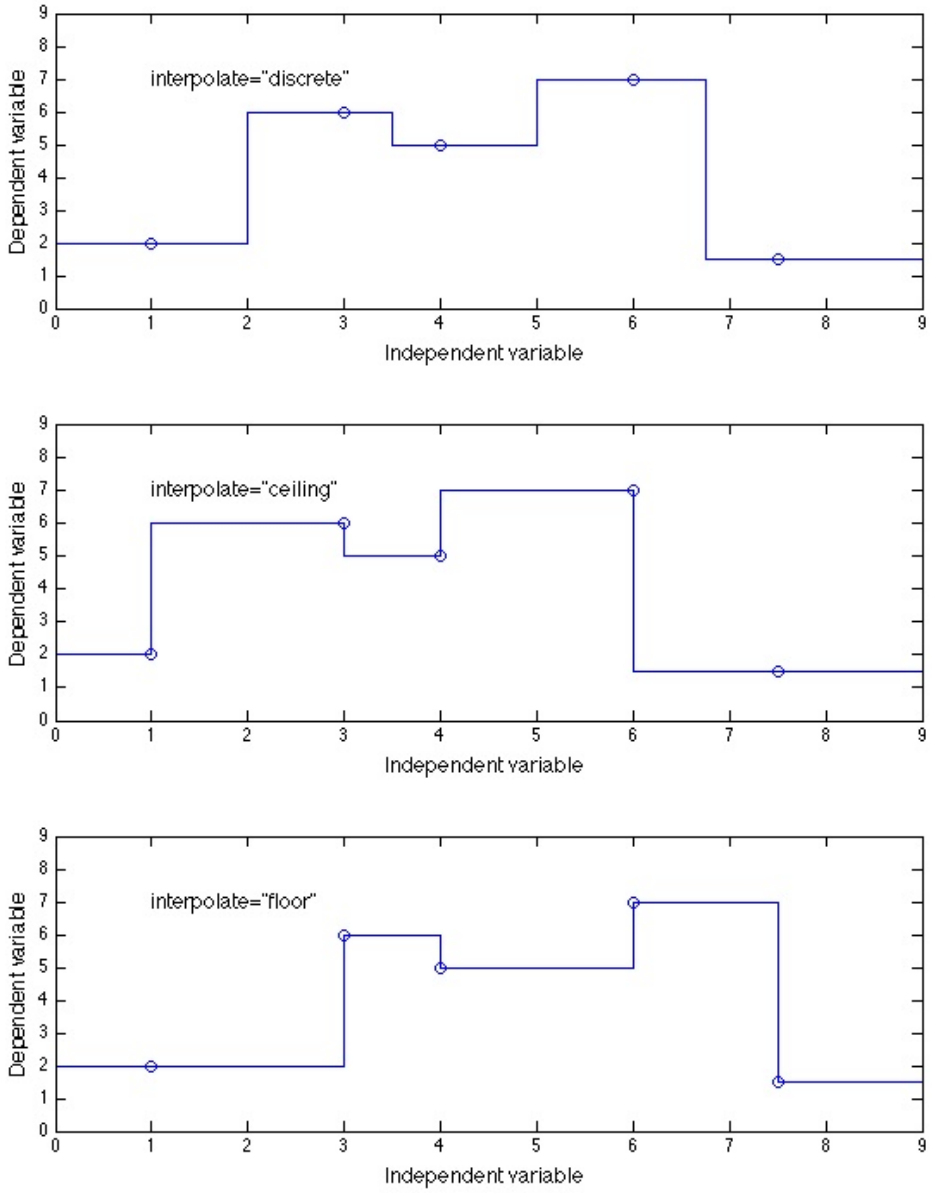


Figure B-4. Example of the three discrete enumeration values of `interpolate` attribute of the `<independentVarPts>` and `<independentVarRef>` elements for a one-dimensional function table.

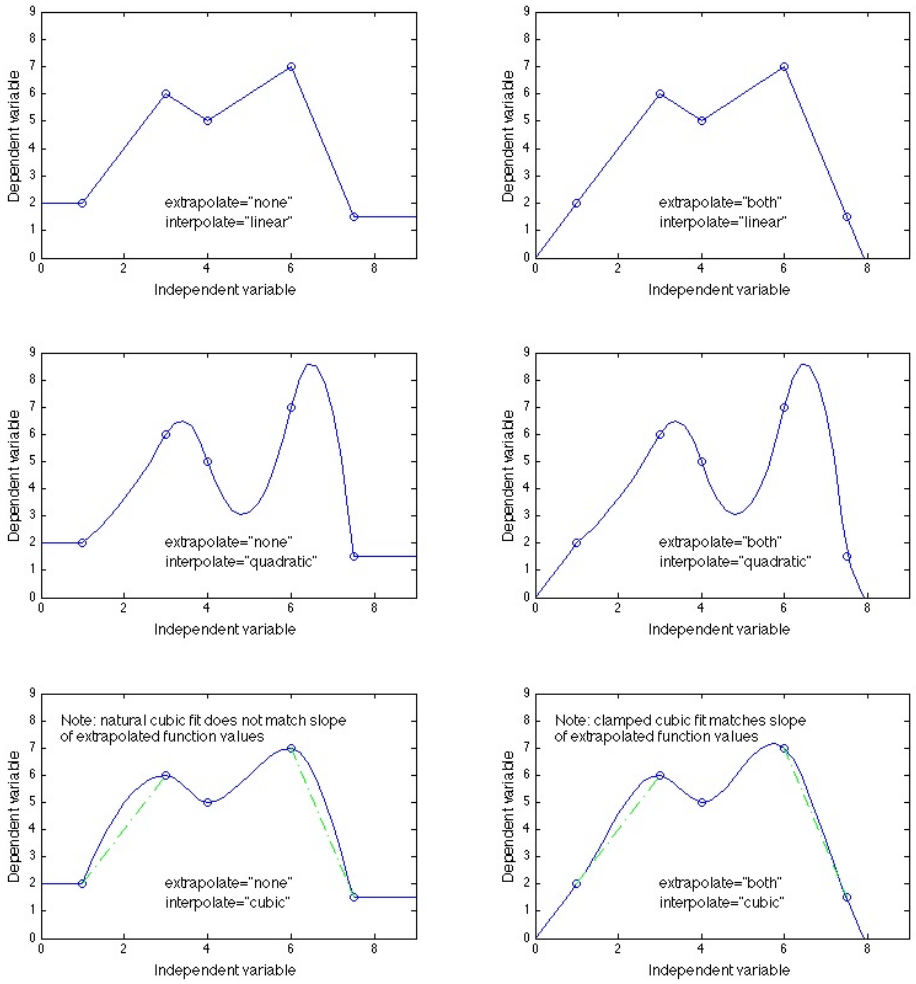


Figure B-5. Examples of the three higher-order interpolation methods showing the effect of the `interpolate` attribute of the `<independentVarPts>` and `<independentVarRef>` elements for a one-dimensional function table.

B-6.4. Statistical information encoding

Statistical measures of variation of certain parameters and functions can be embedded in a DAVE-ML model. This information is captured in a `uncertainty` element, which can be referenced by `variableDef`, `griddedTableDef` and `ungriddedTableDef` elements.

Uncertainty in the value of a parameter or function is given in one of two ways, depending on the appropriate probability distribution function (PDF): as a Gaussian or normal distribution (bell curve) or as a uniform (evenly spread) distribution. One of these distributions is selected by including either a `normalPDF` or a `uniformPDF` element within the `uncertainty` element.

Linear correlation between the randomness of two or more variables or functions can be specified. Although the correlation between parameters do not have a dependency direction (that

is, the statistical uncertainty of either parameter is specified in terms of the other one so the calculation order doesn't matter) correlation is customarily specified as a dependency of one random variable on the value of another random variable. `correlatesWith` identifies variables or functions whose uncertainty 'depends' on the current value of this variable or parameter; the `correlation` sub-element specifies the correlation coefficient and identifies the (previously calculated) random variable or function on which the correlation depends.

These correlation sub-elements only apply to normal (Gaussian) probability distribution functions.

Each of these distribution description elements contain additional information, as described below.

```
uncertainty : effect=['additive'|'multiplicative'|'percentage'|'absolute']
  EITHER
    normalPDF : numSigmas=['1', '2', '3', ...]
        bounds :
        (correlatesWith* : varID |
         correlation* : varID, corrCoef )
  OR
    uniformPDF
        bounds [, bounds]
```

`uncertainty` attributes:

`effect`	Indicates, by choice of four enumerated values, how the uncertainty is modeled: as an additive, multiplicative, or percentage variation about the nominal value, or an specific number (absolute).

`uncertainty` sub-elements:

`normalPDF`	If present, the uncertainty in the parameter value has a probability distribution that is Gaussian (bell-shaped). A single parameter representing the additive (± some value), percentage (± some %) of variation from the nominal value in terms of 1, 2, 3, or more standard deviations (sigmas) is specified. Note here multiplicative and absolute bounds don't make much sense.
`uniformPDF`	If present, the uncertainty in the parameter or function value has a uniform likelihood of taking on any value between symmetric or asymmetric boundaries, which are specified in terms of additive (either ±x or +x/-y), multiplicative, percentage, or absolute variations. If absolute, the specified range of values must bracket the nominal value. For this element, the `bounds` sub-element may contain one or two values in which case the boundaries are symmetric or asymmetric.

Example B-16. A variable with absolute uncertainty bounds

This example shows how to specify that a constant parameter that can take on a specified range of values with uniform probability distribution. The nominal value of the minimum drag coefficient is specified to be 0.005, but when performing parametric variations, it is allowed to take on values between 0.001 and 0.01.

```
<DAVEfunc>
    <fileHeader>
      .
      .
      .
    </fileHeader>
    <variableDef name="CD zero" varID="CDo" units="" initialValue="0.005">     ❶
        <description>
            Minimum coefficient of drag with an
            asymmetric uniform uncertainty band
        </description>
        <isOutput/>
        <uncertainty effect="absolute">    ❷
            <uniformPDF>
                <bounds>0.001</bounds>
                <bounds>0.010</bounds>
            </uniformPDF>
        </uncertainty>
    </variableDef>
 </DAVEfunc>
```

❶ We declare the parameter `CDo` as having a nominal value of 0.005.

❷ When parametric variations are applied, the value of `CDo` can vary uniformly between 0.001 and 0.010.

Example B-17. 10% uncertainty applied to output variable with uniform distribution

This example shows how to specify that a variable has a 10% uniformly distributed uncertainty band. In this example, the output variable comes from a non-linear one-dimensional function, but the uncertainty is applied downstream of the table.

```
<DAVEfunc>
    <fileHeader>
       .
       .
       .
    </fileHeader>
    <variableDef name="angleOfAttack" varID="Alpha_deg" units="d">
        <isStdAIAA/>
    </variableDef>
    <variableDef name="Cm_u" varID="Cm_u" units="">
        <description>
            Coefficient of pitching moment with 10 percent
            symmetric uniform uncertainty band
        </description>
        <isOutput/>
        <uncertainty effect="percentage">    ❶
            <uniformPDF>
                <bounds>10.0</bounds>
            </uniformPDF>
        </uncertainty>
    </variableDef>
    <breakpointDef bpID="ALP">
        <bpVals>0, 5, 10, 15, 20, 25, 30, 35</bpVals>
    </breakpointDef>
    <function name="Nominal Cm">
        <description>
            Nominal pitching moment values prior to application
            of uncertainty
        </description>
        <independentVarRef varID="Alpha_deg"/>
        <dependentVarRef varID="Cm_u"/>
        <functionDefn>   ❷
            <griddedTableDef>
                <breakpointRefs>
                    <bpRef bpID="ALP"/>
                </breakpointRefs>
                <dataTable>
5.2, 4.3, 3.1, 1.8, 0.3, 0.1, 0.0, -0.1
                </dataTable>
            </griddedTableDef>
        </functionDefn>
    </function>
</DAVEfunc>
```

❶ We declare the output variable `Cm_u` as having the uncertainty of ±10% uniform distribution.

❷ This function gives the nominal values of `Cm_u` as a one-dimensional function of angle of attack (alpha).

Example B-18. Asymmetric additive uncertainty applied to output variable with uniform distribution

This example shows how to specify that a variable has an asymmetric, uniformly distributed, additive uncertainty band. In this example, the output variable comes from a non-linear one-dimensional function, but the uncertainty is applied downstream of the table.

```
<DAVEfunc>
    <fileHeader>
      .
      .
      .
    </fileHeader>
    <variableDef name="angleOfAttack" varID="Alpha_deg" units="d">
        <isStdAIAA/>
    </variableDef>
    <variableDef name="Cm_u" varID="Cm_u" units="">
        <description>
            Coefficient of pitching moment with an
            asymmetric uniform uncertainty band
        </description>
        <isOutput/>
        <uncertainty effect="additive">  ❶
            <uniformPDF>
                <bounds>-0.50</bounds>
                <bounds>0.00</bounds>
            </uniformPDF>
        </uncertainty>
    </variableDef>
    <breakpointDef bpID="ALP">
        <bpVals>0, 5, 10, 15, 20, 25, 30, 35</bpVals>
    </breakpointDef>
    <function name="Nominal Cm">
        <description>
            Nominal pitching moment values prior to application
            of uncertainty
        </description>
        <independentVarRef varID="Alpha_deg"/>
        <dependentVarRef varID="Cm_u"/>   ❷
        <functionDefn>
            <griddedTableDef>
                <breakpointRefs>
                    <bpRef bpID="ALP"/>
                </breakpointRefs>
                <dataTable>
5.2, 4.3, 3.1, 1.8, 0.3, 0.1, 0.0, -0.1
                </dataTable>
            </griddedTableDef>
        </functionDefn>
    </function>
</DAVEfunc>
```

❶ We declare the output variable `Cm_u` varies by as much as -0.5 to +0.0 about the nominal value. This delta value is in the same units as the nominal value, i.e. it is not a multiplier or percentage, but an additive delta to the nominal value which comes from the one-dimensional `Cm_u` function table description.

❷ This function gives the nominal values of `Cm_u` as a one-dimensional function of angle of attack (alpha).

Example B-19. A one dimensional table, point-by-point, Gaussian

In this example, a Gaussian (normal) distribution function is applied to *each* point in a one-dimensional function table, with the 3-sigma value expressed as a multiplier of the nominal value.

```
<DAVEfunc>
    <fileHeader>
       .
       .
       .
    </fileHeader>
    <variableDef name="angleOfAttack" varID="Alpha_deg" units="d">
        <isStdAIAA/>
    </variableDef>
    <variableDef name="Cm_u" varID="Cm_u" units="">
        <description>
            Coefficient of pitching moment with 10 percent
            symmetric uniform uncertainty band
        </description>
        <isOutput/>
    </variableDef>
    <breakpointDef bpID="ALP">
        <bpVals>0, 5, 10, 15, 20, 25, 30, 35</bpVals>
    </breakpointDef>
  <function name="Uncertain Cm">
    <independentVarRef varID="Alpha_deg"/>
    <dependentVarRef varID="Cm_u"/>
    <functionDefn>
      <griddedTableDef>
        <breakpointRefs>
          <bpRef bpID="ALP"/>
        </breakpointRefs>
        <uncertainty effect="multiplicative">  ❶
          <normalPDF numSigmas="3">  ❷
            <bounds>
              <dataTable>  ❸
                0.10, 0.08, 0.06, 0.05, 0.05, 0.06, 0.07, 0.12
              </dataTable>
            </bounds>
          </normalPDF>
        </uncertainty>
        <dataTable>   ❹
          5.2, 4.3, 3.1, 1.8, 0.3, 0.1, 0.0, -0.1
        </dataTable>
      </griddedTableDef>
    </functionDefn>
  </function>
</DAVEfunc>
```

❶ This declares the statistical uncertainty bounds of the `Cm_u` dependent variable will be expressed as a multiplication of the nominal value.

❷ This declares that the probability distribution is a normal distribution and the bounds represent 3-sigma (99.7%) confidence bounds.

❸ This table defines ±3-sigma bounds of the `Cm_u` function as a table. The table must have the same dimensions and independent variable arguments as the nominal function; it is in effect an overlay to the nominal function table, but the values represent the bounds as multiples of the nominal function value.

❹ This table defines the nominal values of the function; these values will be used if the random variable associated with the uncertainty of this function is zero or undefined by the application.

Example B-20. Two nonlinear functions with correlated uncertainty

In this example, uncertainty in pitching-moment coefficient varies in direct correlation with lift coefficient uncertainty.

```
<DAVEfunc>
    <fileHeader> . . . </fileHeader>
    <variableDef name="angleOfAttack" varID="Alpha_deg" units="d">
        <isStdAIAA/>
    </variableDef>
    <variableDef name="CL_u" varID="CL_u" units="">
        <description> Coefficient of lift with a symmetric Gaussian uncertainty
            of 20%; correlates with Cm uncertainty. </description>
        <uncertainty effect="multiplicative">    ❶
            <normalPDF numSigmas="3">
                <bounds>0.20</bounds>
                <correlatesWith varID="Cm_u"/>   ❷
            </normalPDF>
        </uncertainty>
    </variableDef>
    <variableDef name="Cm_u" varID="Cm_u" units="">
        <description> Coefficient of pitching moment with a symmetric Gaussian uncertainty
            distribution of 30%; correlates directly with lift uncertainty. </description>
        <isOutput/>
        <uncertainty effect="percentage">        ❸
            <normalPDF numSigmas="3">
                <bounds>30</bounds>
                <correlation varID="CL_u" corrCoef="1.0"/>  ❹
            </normalPDF>
        </uncertainty>
    </variableDef>
    <breakpointDef bpID="ALP">
        <bpVals>0, 5, 10, 15, 20, 25, 30, 35</bpVals>
    </breakpointDef>
    <function name="Nominal CL">
        <description> Nominal lift coefficient values prior to uncertainty </description>
        <independentVarRef varID="Alpha_deg"/>
        <dependentVarRef varID="CL_u"/>
        <functionDefn>
            <griddedTableDef>
                <breakpointRefs><bpRef bpID="ALP"/></breakpointRefs>
                <dataTable> 0.0, 0.1, 0.2, 0.3, 0.4, 0.45, 0.5, 0.45 </dataTable>
            </griddedTableDef>
        </functionDefn>
    </function>
    <function name="Nominal Cm">
        <description> Nominal pitching moment values prior to uncertainty </description>
        <independentVarRef varID="Alpha_deg"/>
        <dependentVarRef varID="Cm_u"/>
        <functionDefn>
            <griddedTableDef>
                <breakpointRefs><bpRef bpID="ALP"/></breakpointRefs>
                <dataTable> 5.2, 4.3, 3.1, 1.8, 0.3, 0.1, 0.0, -0.1 </dataTable>
            </griddedTableDef>
        </functionDefn>
    </function>
</DAVEfunc>
```

❶ Lift coefficient has a nominal value which varies with angle of attack according to a non-linear one-dimensional table defined later. When performing parametric variations, `CL_u` can take on a multiplicative variation of up to 20% (3-sigma) with a Gaussian distribution.

❷ This element indicates that the variation of pitching moment `Cm_u` correlates with the variation of lift coefficient.

❸ Pitching-moment coefficient has a nominal value which varies as a function of angle of attack, according to a non-linear one-dimensional table defined later. When performing parametric variations, `Cm_u` can take on a 30% variation (3-sigma) with a Gaussian distribution.

❹ This element indicates that the variation of pitching moment correlates directly with the variation in lift coefficient.

B-6.5. Additional DAVE-ML conventions

To facilitate the interpretation of DAVE-ML packages, the following conventions are proposed. Failure to follow any of these should be noted prominently in the data files and any cover documentation.

B-6.5.1. Ordering of points

In listing data points in multi-dimensional table definitions, the sequence should be that the last dimension changes most rapidly. In other words, a table that is a function of f(a,b,c) should list the value for f(1,1,1), then f(1,1,2), etc. This may be different than, say, a FORTRAN DATA, Matlab® script or C++ initialization statement; the responsibility for mapping into the 'solution space' is left up to the interpreter.

B-6.5.2. Locus of action of moments

It is recommended that all force and moments be considered to act around a defined reference point, given in aircraft coordinates. It is further recommended that all subsystem models (aerodynamic, propulsive, alighting gear) provide total forces and moments about this reference point and leave the transfer of moments to the center of mass to the equations of motion.

B-6.5.3. Decomposition of flight dynamic subsystems

It is recommended that a vehicle's flight dynamic reactions be modeled, at least at the highest level, as aerodynamic, propulsive, and landing/arresting/launch gear models. This is common practice in most aircraft simulation environments familiar to the authors.

B-6.5.4. Date format in DAVE-ML

The recommended way of representing dates in DAVE-ML documentation, especially date attribute and creation date elements, is numerically in the order YYYY-MM-DD. Thus, July 15, 2003 is given as 2003-07-15. This is to conform to ISO-8601 regarding date and time formats.

B-6.5.5. Common sign convention notation

The following list of sign convention notation is recommended for adoption. Note the sign convention for most quantities is already fixed by the AIAA Recommended Practice [AIAA92], so this is actually a list of abbreviations for typical sign conventions:

Common DAVE-ML sign convention notation

Acronym: +AFT
Meaning: Positive aft
Acronym: +ANR
Meaning: Positive aircraft nose right
Acronym: +ANU
Meaning: Positive aircraft nose up
Acronym: +CWFN
Meaning: Positive clockwise from north
Acronym: +DN
Meaning: Positive down
Acronym: +E
Meaning: Positive eastward
Acronym: +FWD
Meaning: Positive forward
Acronym: +LFT

Meaning: Positive left
Acronym: +N
Meaning: Positive northward
Acronym: +OUT
Meaning: Positive outward
Acronym: +POS
Meaning: Always positive
Acronym: +RCL
Meaning: Positive right of centerline
Acronym: +RT
Meaning: Positive right
Acronym: +RWD
Meaning: Positive right wing down
Acronym: +TED
Meaning: Positive trailing edge down
Acronym: +TEL
Meaning: Positive trailing edge left
Acronym: +THR
Meaning: Positive beyond threshold
Acronym: +UP
Meaning: Positive up

B-6.5.6. Units of measure abbreviation

Each variable definition includes a mandatory `<units>` attribute. This attribute gives the units-of-measure for the signal represented by the variable, and either 'ND' (for No Dimensions) or blank if the signal is a dimensionless quantity or flag. In addition, the use of 'fract' to indicate a fraction (0-1) or 'pct' to indicate a percentage (0-100, or more) is encouraged.

Informally, this attribute can take on any reasonable abbreviation for a set of units that might be understandable by the intended audience, in any set of units (English or ISO). For greater re-usability, it is recommended that the set of measurements listed in the AIAA Standard for Flight Simulation Models, of which this document is a part, be used. The Standard suggests how to encode superscripted powers of units (fs_1 for ft/sec, for example).

B-6.5.7. Internal identifiers

Identifiers are used throughout DAVE-ML to connect signals, functions, modification records and reference documents with each other, e.g. `<utID>`, `<gtID>`, `<bpID>`, `<refID>`, etc. These identifiers are character strings that must comply with the official XML specification for Names [W3C-XML] and thus must start with a character or underbar, may not start with "xml" or "XML", and may not contain colons, among other restrictions. See the XML specification for more information regarding valid XML Names.

B-6.5.8. XML Namespaces

The XML standard allows for namespace domains, such that element names (tag names) belong to either the empty (null) namespace or in a namespace that belongs to a particular XML grammar. Namespaces are identified by a uniform resource identifier (URI) that is fashioned, similar to a URL, to be guaranteed to be unique.

The DAVE-ML namespace should be defined in the top-level element as follows:

```
<DAVEfunc xmlns="http://daveml.nasa.gov/2008/DAVEML">
```

This will allow DAVE-ML models to be embedded in other XML documents without confusion. Note that this is not a URL; http queries at that address may not lead to any useful information.

The `<reference>` element can include two elements that belong to the World-Wide Web Consortium (W3C)'s XLINK protocol [W3C-XLINK]; they are defined with an `xlink:` prefix which actually refers to the namespace uniquely defined with the `http://www.w3c.org/1999/xlink` URI. If external links to documents will be included in a DAVE-ML document, the top-level element (currently `<reference>` *must* include a namespace declaration (which looks like, but technically is not, an attribute):

```
<reference xmlns:xlink="http://www.w3.org/1999/xlink">
```

Similarly, the MathML elements are normally defined in a MathML namespace, so any calculation defined using MathML notation should be conducted inside the MathML namespace:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
```

B-6.6. Planned major elements

Additional major elements may be defined to support the goal of rapid exchange of simulation models, including

- Subsystem models, to support hierarchical decomposition and problem abstraction.
- State variables, both discrete and continuous, to support dynamic models.
- Dynamic (time-history) data file format, to allow for validation check cases for dynamic models

B-7. Further information

Further information, background, the latest DTD and example models of some aircraft data packages can be found at the DAVE-ML web site: http://daveml.nasa.gov

B-8. Element references and descriptions

B-8.1. Element descriptions

address

address — Street address or other contact information of an author [Deprecated.]

Content model

```
address :
     (#PCDATA)
```

Attributes

NONE

Possible parents

`author`

Allowable children

NONE

Future plans for this element

This element has been subsumed by the contactInfo element below.

author

author — Gives name and contact information for originating party of the associated data

Content model

```
author :  name,  org,  [xns],  [email]
     (address* | contactInfo*)
```

Attributes

`name` - the name of the author or last modifier of the associated element's data

`org` - the author's organization

`xns` (optional) - the eXtensible Name Service identifier for the author [Deprecated]

`email` (optional) - the e-mail address for the primary author [Deprecated]

Description

author includes alternate means of identifying author using XNS or normal e-mail/address. The address subelement is to be replaced with the more complete contactInfo subelement.

Possible parents

`fileHeader`
`modificationRecord`
`provenance`

Allowable children

`address`
`contactInfo`

Future plans for this element

Both the xns and email attributes are deprecated and will be removed. XNS was a proposed internet technology (eXtensible Name Service) to reduce spam that didn't catch on. It is replaced with the 'iname' subelement as a single means to identify an individual or corporation in lieu of typical (and quickly dated) e-mail, phone, or address information. The address element itself is deprecated and should be replaced with the contactInfo element

bounds

bounds — Describes limits or standard deviations of statistical uncertainties

Content model

```
bounds :
     (dataTable | variableDef | variableRef)*
```

Attributes

NONE

Description

This element contains some description of the statistical limits to the values the citing parameter element might take on. This can be in the form of a scalar value, a[n] [un]griddedTableRef reference to an existing table definition, or a private [un]griddedTableDef, or a private table. In the case of formal table references or definitions, these tables define their own dependency, independent of the underlying random variable (whose nominal value is probably specified in a parent table definition). In the more common instance, this element will either be a scalar constant value or a simple table, whose dimensions must match the parent nominal function table and whose independent variables are identical to the nominal table. It is also possible that this limit be determined by an independent variable.

Possible parents

```
normalPDF
uniformPDF
```

Allowable children

```
dataTable
variableDef
variableRef
```

bpRef

bpRef — Reference to a breakpoint list

Content model

```
bpRef :  bpID
     EMPTY
```

Attributes

`bpID` - the internal identifier for a breakpoint set definition

Description

The bpRef element provides references to breakpoint lists so breakpoints can be defined separately from, and reused by, several data tables.

Possible parents

`breakpointRefs`

Allowable children

NONE

bpVals

bpVals — String of whitespace- or comma-separated values of breakpoints

Content model

```
bpVals :
     (#PCDATA)
```

Attributes

NONE

Description

bpVals is a set of breakpoints; that is, a set of independent variable values associated with one dimension of a gridded table of data. An example would be the Mach or angle-of-attack values that define the coordinates of each data point in a two-dimensional coefficient value table.

Possible parents

`breakpointDef`

Allowable children

NONE

breakpointDef

breakpointDef — Defines breakpoint sets to be used in model

Content model

```
breakpointDef :  [name],  bpID,  [units]
     (description?, bpVals)
```

Attributes

`name` (optional) - the name of the breakpoint set

`bpID` - the internal identifier for the breakpoint set

`units` (optional) - the units of measure for the breakpoint set

Description

A breakpointDef is where gridded table breakpoints are given. Since these are separate from function data they may be reused.

Possible parents

`DAVEfunc`

Allowable children

`description`
`bpVals`

breakpointRefs

breakpointRefs — Reference to a breakpoint definition

Content model

```
breakpointRefs :
     bpRef+
```

Attributes

NONE

Description

The breakpointRefs elements tie the independent variable names for the function to specific breakpoint values defined earlier.

Possible parents

`griddedTableDef`
`griddedTable`

Allowable children

`bpRef`

calculation

calculation — Used to delimit a MathML v2 calculation

Content model

```
calculation :
     math [11]
```

Attributes

NONE

Description

The calculation element is MathML 2 content markup describing how the signal is calculated.

Possible parents

`variableDef`

Allowable children

`math` [11]

checkData

checkData — Gives verification data for encoded model

Content model

```
checkData :
     (
          (provenance? | provenanceRef?)
, staticShot*)
```

Attributes

NONE

Description

This top-level element is the placeholder for verification data of various forms for the encoded model. It will include static check cases, trim shots, and dynamic check case information. The provenance sub-element is now deprecated and has been moved to individual staticShots; it is allowed here for backwards compatibiliy.

Possible parents

DAVEfunc

Allowable children

provenance
provenanceRef
staticShot

checkInputs

checkInputs — Lists input values for check case

Content model

```
checkInputs :
     signal+
```

Attributes

NONE

Description

Specifies the contents of the input vector for the given check case.

Possible parents

`staticShot`

Allowable children

`signal`

checkOutputs

checkOutputs — Lists output values for check case

Content model

```
checkOutputs :
     signal+
```

Attributes

NONE

Description

Specifies the contents of the output vector for the given check case.

Possible parents

`staticShot`

Allowable children

`signal`

confidenceBound

confidenceBound — Defines the confidence in a function [Deprecated]

Content model

```
confidenceBound :  value
     EMPTY
```

Attributes

`value` - percent confidence (like 95%) in the function

Description

The confidenceBound element is used to declare the confidence interval associated with the data table. This is a placeholder and will be removed in a future version of DAVE-ML.

Possible parents

`griddedTable`
`ungriddedTable`

Allowable children

NONE

Future plans for this element

Deprecated. Used only in deprecated [un]griddedTable elements. Use uncertainty element instead.

contactInfo

contactInfo — Provides multiple contact information associated with an author or agency

Content model

```
contactInfo :  [contactInfoType],  [contactLocation]
     (#PCDATA)
```

Attributes

`contactInfoType` (optional) - Indicates type of information being conveyed (enumerated)

- `address`
- `phone`
- `fax`
- `email`
- `iname`
- `web`

`contactLocation` (optional) - Indicates which location is identified. Default is professional. (enumerated)

- `professional`
- `personal`
- `mobile`

Description

Used to provide contact information about an author. Use contactInfoType to differentiate what information is being conveyed, and contactLocation to denote location of the address.

Possible parents

`author`

Allowable children

NONE

correlatesWith

correlatesWith — Identifies other functions or variables whose uncertainty correlates with our random value

Content model

```
correlatesWith :  varID
     EMPTY
```

Attributes

`varID` - Identifies the variable or function output that will depend on this function or variable's randomness

Description

When present, this element indicates the parent function or variable is correlated with the referenced other function or variable in a linear sense. This alerts the application that the random number used to calculate this function or variable's immediate value will be used to calculate another function of variable's value.

Possible parents

`normalPDF`

Allowable children

NONE

correlation

correlation — Indicates the linear correlation of this function or variable's randomness with a previously computed random variable

Content model

```
correlation :  varID,  corrCoef
     EMPTY
```

Attributes

`varID` - Identifies the variable or function output that helps determine the value of this random variable or function.

`corrCoef` - Indicates the amount of correlation between this variable and the referenced variable. The value should be between -1 and 1; 0 indicates no correlation, 1 indicates perfect correlation and -1 indicates inverse (negative) correlation.

Description

When present, this element indicates the parent function or variable is correlated with the referenced other function or variable in a linear sense, and gives the correlation coefficient for determining this function's random value based upon the correlating function(s) random value.

Possible parents

`normalPDF`

Allowable children

NONE

creationDate

creationDate — Gives date of creation of entity

Content model

```
creationDate :  date
     EMPTY
```

Attributes

`date` - The date of creation of the entity, in ISO 8601 (YYYY-MM-DD) format

Description

creationDate is simply a string with a date in it. We follow ISO 8601 and use dates like "2004-01-02" to refer to January 2, 2004.

Possible parents

`fileHeader`
`provenance`

Allowable children

NONE

dataPoint

dataPoint — Defines each point of an ungridded table

Content model

```
dataPoint :  [modID]
     (#PCDATA)
```

Attributes

`modID` (optional) - the internal identifier for a modification record

Description

The dataPoint element is used by ungridded tables to list the values of independent variables that are associated with each value of dependent variable. For example: <dataPoint> 0.1, -4.0, 0.2 <!- Mach, alpha, CL -> </dataPoint> <dataPoint> 0.1, 0.0, 0.6 <!- Mach, alpha CL -> </dataPoint> Each data point may have associated with it a modification tag to document the genesis of that particular point. No requirement on ordering of independent variables is implied. Since this is a ungridded table, the intepreting application is required to handle what may be unsorted data.

Possible parents

`ungriddedTableDef`
`ungriddedTable`

Allowable children

NONE

dataTable

dataTable — Lists the values of a table of function or uncertainty data

Content model

```
dataTable :
     (#PCDATA)
```

Attributes

NONE

Description

The dataTable element is used by gridded tables where the indep. variable values are implied by breakpoint sets. Thus, the data embedded between the dataTable element tags is expected to be sorted ASCII values of the gridded table, wherein the last independent variable listed in the function header varies most rapidly. Values are comma or whitespace separated. The table data point values are specified as comma- or whitespace-separated values in conventional floating-point notation (0.93638E-06) in a single long sequence as if the table had been unraveled with the last-specified dimension changing most rapidly. Line breaks are to be ignored. Comments may be embedded in the table to promote [human] readability, with appropriate escaping characters. A dataTable element can also be used in an uncertainty element to provide duplicate uncertainty bound values.

Possible parents

`griddedTableDef`
`griddedTable`
`bounds`

Allowable children

NONE

DAVEfunc

DAVEfunc — Root level element

Content model

```
DAVEfunc :  xmlns

 (fileHeader, variableDef+, breakpointDef*, griddedTableDef*, ungriddedTableDef*, function*, checkData?)
```

Attributes

`xmlns` - This attribute specifies that the default namespace for unprefixed elements is this DTD.

Description

Root element is DAVEfunc, composed of a file header element followed by 1 or more variable definitions and 0 or more break point definitions, gridded or ungridded table definitions, and function elements.

Possible parents

NONE - ROOT ELEMENT

Allowable children

```
fileHeader
variableDef
breakpointDef
griddedTableDef
ungriddedTableDef
function
checkData
```

dependentVarPts

dependentVarPts — Defines output breakpoint values

Content model

```
dependentVarPts :  varID,  [name],  [units],  [sign]
     (#PCDATA)
```

Attributes

`varID` - the internal identifier of the output signal this table should drive

`name` (optional) - the name of the function's dependent variable output signal

`units` (optional) - the units of measure for the dependent variable

`sign` (optional) - the sign convention for the dependent variable

Description

A dependentVarPts element is a simple comma- or whitespace-delimited list of function values and contains a mandatory varID as well as optional name, units, and sign convention attributes. Data points are arranged as single vector with last-specified breakpoint values changing most frequently. Note that the number of dependent values must equal the product of the number of independent values for this simple, gridded, realization. This element is used for simple functions that don't share breakpoint or table values with other functions.

Possible parents

`function`

Allowable children

NONE

dependentVarRef

dependentVarRef — Identifies the signal to be associated with the output of a function

Content model

```
dependentVarRef :  varID
     EMPTY
```

Attributes

`varID` - the internal identifier for the output signal

Description

A dependentVarRef ties the output of a function to a signal name defined previously in a variable definition.

Possible parents

`function`

Allowable children

NONE

description

description — Verbal description of an entity

Content model

```
description :
     (#PCDATA)
```

Attributes

NONE

Description

The description element is a free-form text block describing something.

Possible parents

```
fileHeader
variableDef
breakpointDef
griddedTableDef
ungriddedTableDef
function
reference
modificationRecord
provenance
staticShot
```

Allowable children

NONE

documentRef

documentRef — Reference to an external document

Content model

```
documentRef :  [docID],  refID
     EMPTY
```

Attributes

`docID` (optional) - the internal identifier of a reference definition element [Deprecated]

`refID` - the internal identifier of a reference definition element

Possible parents

`provenance`

Allowable children

NONE

Future plans for this element

The 'docID' attribute is deprecated; it has been renamed 'refID' to match its use in the 'reference' element. This attribute will be removed in a future version of DAVE-ML.

extraDocRef

extraDocRef — Allows multiple documents to be associated with a single modification event

Content model

```
extraDocRef :  refID
     EMPTY
```

Attributes

`refID` - The internal identifier of a reference definition element

Description

A single modification event may have more than one documented reference. This element can be used in place of the refID attribute in a modificationRecord to record more than one refIDs, pointing to the referenced document.

Possible parents

`modificationRecord`

Allowable children

NONE

fileCreationDate

fileCreationDate — Gives date of creation of entity [Deprecated]

Content model

```
fileCreationDate :  date
     EMPTY
```

Attributes

`date` - The date of the file, in ISO 8601 (YYYY-MM-DD) format

Description

fileCreationDate is simply a string with a date in it. We follow ISO 8601 and use dates like "2004-01-02" to refer to January 2, 2004. It's use is now deprecated in favor of the more simple creationDate.

Possible parents

`fileHeader`

Allowable children

NONE

Future plans for this element

The fileCreationDate and functionCreationDate have been replaced with a new creationDate multipurpose element.

fileHeader

fileHeader — States source and purpose of file

Content model

```
fileHeader :  [name]
     (author+,
          (creationDate | fileCreationDate)
, fileVersion?, description?, reference*, modificationRecord*, provenance*)
```

Attributes

`name` (optional) - the name of the file

Description

The header element requires at least one author, a creation date and a version indicator; optional content are description, references and mod records.

Possible parents

`DAVEfunc`

Allowable children

`author`
`creationDate`
`fileCreationDate`
`fileVersion`
`description`
`reference`
`modificationRecord`
`provenance`

fileVersion

fileVersion — Indicates the version of the document

Content model

```
fileVersion :
     (#PCDATA)
```

Attributes

NONE

Description

This is a string describing, in some arbitrary text, the version identifier for this function description.

Possible parents

`fileHeader`

Allowable children

NONE

function

function — Defines a function by combining independent variables, breakpoints, and tables

Content model

```
function :  name
     (description?,
          (provenance? | provenanceRef?)
,
          (
               (independentVarPts+, dependentVarPts)
 |
               (independentVarRef+, dependentVarRef, functionDefn)
)
)
```

Attributes

`name` - the name of this function

Description

Each function has optional description, optional provenance, and either a simple input/output values or references to more complete (possible multiple) input, output, and function data elements.

Possible parents

`DAVEfunc`

Allowable children

`description`
`provenance`
`provenanceRef`
`independentVarPts`
`dependentVarPts`
`independentVarRef`
`dependentVarRef`
`functionDefn`

functionCreationDate

functionCreationDate — Date of creation of a function table [Deprecated]

Content model

```
functionCreationDate :  date
     EMPTY
```

Attributes

`date` - the creation date of the function, in ISO 8601 (YYYY-MM-DD) format

Description

functionCreationDate is simply a string with a date in it. We follow ISO 8601 and use dates like "2004-01-02" to refer to January 2, 2004. It's use is now deprecated in favor of the more simple creationDate.

Possible parents

`provenance`

Allowable children

NONE

Future plans for this element

The fileCreationDate and functionCreationDate have been replaced with a new creationDate multipurpose element.

functionDefn

functionDefn — Defines a function by associating a table with other information

Content model

```
functionDefn :  [name]
     (griddedTableRef | griddedTableDef | griddedTable | ungriddedTableRef | ungriddedTableDef
 | ungriddedTable)
```

Attributes

`name` (optional) - the name of this function definition

Description

A functionDefn defines how function is represented in one of two possible ways: gridded (implies breakpoints), or ungridded (with explicit independent values for each point).

Possible parents

`function`

Allowable children

`griddedTableRef`
`griddedTableDef`
`griddedTable`
`ungriddedTableRef`
`ungriddedTableDef`
`ungriddedTable`

griddedTable

griddedTable — Definition of a gridded table; associates breakpoint data with table data [Deprecated]

Content model

```
griddedTable :  [name]
     (breakpointRefs, confidenceBound?, dataTable)
```

Attributes

`name` (optional) - the name of the gridded table being defined

Possible parents

`functionDefn`

Allowable children

`breakpointRefs`
`confidenceBound`
`dataTable`

Future plans for this element

Deprecated. Use griddedTableDef instead.

griddedTableDef

griddedTableDef — Defines an orthogonally-gridded table of data points

Content model

```
griddedTableDef :  [name],  [gtID],  [units]
     (description?,
          (provenance? | provenanceRef?)
, breakpointRefs, uncertainty?, dataTable)
```

Attributes

`name`	(optional) - the name of the gridded table
`gtID`	(optional) - an internal identifier for the table
`units`	(optional) - units of measure for the table values

Description

A griddedTableDef contains points arranged in an orthogonal (but multi-dimensional) array, where the independent variables are defined by separate breakpoint vectors. This table definition may be specified separately from the actual function declaration; if so, it requires an gtID identifier attribute so that it may be used by multiple functions.

Possible parents

`DAVEfunc`
`functionDefn`

Allowable children

`description`
`provenance`
`provenanceRef`
`breakpointRefs`
`uncertainty`
`dataTable`

griddedTableRef

griddedTableRef — Reference to a gridded table definition

Content model

```
griddedTableRef :  gtID
     EMPTY
```

Attributes

`gtID` - the internal identifier of a gridded table definition

Possible parents

`functionDefn`

Allowable children

NONE

independentVarPts

independentVarPts — Simple definition of independent breakpoints

Content model

```
independentVarPts :  varID,  [name],  [units],  [sign],  [extrapolate],  [interpolate]
     (#PCDATA)
```

Attributes

`varID` - the internal identifier of the input signal corresponding to this independent variable

`name` (optional) - the name of the function's independent variable input signal

`units` (optional) - the units of measure for the independent variable

`sign` (optional) - the sign convention for the independent variable

`extrapolate` (optional) - extrapolation flags for IV out-of-bounds (default is neither) (enumerated)

- `neither`
- `min`
- `max`
- `both`

`interpolate` (optional) - Interpolation flags for independent variable (default is linear) (enumerated)

- `discrete`
- `floor`
- `ceiling`
- `linear`
- `quadraticSpline`
- `cubicSpline`

Description

An independentVarPts element is a simple whitespace- or comma-separated list of breakpoints and contains a mandatory varID identifier as well as optional name, units, and sign convention attributes. An optional extrapolate attribute describes how to extrapolate the output value when the input value exceeds specified values (default is 'neither,' meaning the value of the table is held constant at the nearest defined value). An optional interpolate attribute indicates how to perform the interpolation within the table (supporting discrete, linear, cubic or quadratic splines). There are three different discrete options: 'discrete' means nearest-neighbor, with an exact mid-point value being rounded in the positive direction; 'ceiling' means the function takes on the value associated with the next (numerically) higher independent breakpoint as soon as original value is exceeded, and 'floor' means the function holds the value associated with each breakpoint until

the next (numerically) higher breakpoint value is reached by the independent argument. The default interpolation attribute value is 'linear.' This element is used for simple functions that don't share breakpoint or table values with other functions.

Possible parents

`function`

Allowable children

NONE

independentVarRef

independentVarRef — References a predefined signal as an input to a function

Content model

```
independentVarRef :  varID,  [min],  [max],  [extrapolate],  [interpolate]
     EMPTY
```

Attributes

`varID` - the internal identifier for the input signal

`min` (optional) - the allowable lower limit for the input signal

`max` (optional) - the allowable upper limit for the input signal

`extrapolate` (optional) - extrapolation flags for IV out-of-bounds (enumerated)

- `neither`
- `min`
- `max`
- `both`

`interpolate` (optional) - Interpolation flags for independent variable (enumerated)

- `discrete`
- `floor`
- `ceiling`
- `linear`
- `quadraticSpline`
- `cubicSpline`

Description

An independentVarRef more fully describes the input mapping of the function by pointing to a separate breakpoint definition element. An optional extrapolate attribute describes how to extrapolate the output value when the input value exceeds specified values (default is 'neither,' meaning the value of the table is held constant at the nearest defined value). An optional interpolate attribute indicates how to perform the interpolation within the table (supporting discrete, linear, cubic or quadratic splines). There are three different discrete options: 'discrete' means nearest-neighbor, with an exact mid-point value being rounded in the positive direction; 'floor' means the function takes on the value associated with the next (numerically) higher independent breakpoint as soon as original value is exceeded, and 'ceiling' means the function holds the value associated with each breakpoint until the next (numerically) higher breakpoint value is reached by the independent argument. The default interpolation attribute value is 'linear.' This element allows reuse of common breakpoint values for many tables, but with possible differences in interpolation or extrapolation for each use.

Possible parents

`function`

Allowable children

NONE

internalValues

internalValues — A dump of internal model values for debugging checkcases

Content model

```
internalValues :
     signal+
```

Attributes

NONE

Description

Provides a set of all internal variable values to assist in debugging recalcitrant implementations of DAVE-ML import tools.

Possible parents

`staticShot`

Allowable children

`signal`

isOutput

isOutput — Flag to identify non-obvious output signals from model

Content model

```
isOutput :
     EMPTY
```

Attributes

NONE

Description

The presence of the isOutput element indicates that this variable should be forced to be an output, even if it is used internally as an input elsewhere. Otherwise, the processing program should assume a signal defined with no calculation is an input; a signal defined with a calculation but not used elsewhere is an output; and a signal defined as a calculation and used subsequently in the model is an internal signal.

Possible parents

`variableDef`

Allowable children

NONE

isState

isState — Flag to identify a state variable within a dynamic model

Content model

```
isState :
     EMPTY
```

Attributes

NONE

Description

The presence of an isState element indicates that this variable is one of possibly multiple state variables in a dynamic model; this tells the processing entity that this is the output of an integrator (for continous models) or a discretely updated state (for discrete models).

Possible parents

`variableDef`

Allowable children

NONE

isStateDeriv

isStateDeriv — Flag to identify a state derivative within a dynamic model

Content model

```
isStateDeriv :
     EMPTY
```

Attributes

NONE

Description

The presences of an isStateDeriv element indicates that this variable is one of possibly several state derivative variables in a dynamic model; this tells the processing entity that this is the output of an integrator (for continous models only).

Possible parents

`variableDef`

Allowable children

NONE

isStdAIAA

isStdAIAA — Flag to identify standard AIAA simulation variable

Content model

```
isStdAIAA :
     EMPTY
```

Attributes

NONE

Description

The presence of an isStdAIAA element indicates this variable is one of the [draft] standard AIAA variable names which should be recognizable exterior to this module, e.g. AngleOfAttack_deg. This flag should assist importing tools determine when an input or output should match a facility-provided signal name without requiring further information.

Possible parents

`variableDef`

Allowable children

NONE

modificationRecord

modificationRecord — To associate a reference single letter with a modification event

Content model

```
modificationRecord :  modID,  date,  [refID]
     (author+, description?, extraDocRef*)
```

Attributes

`modID` - a single letter used to identify all modified data associated with this modification record

`date` - the date of the modification, in ISO 8601 (YYYY-MM-DD) format

`refID` (optional) - a reference to a predefined document that describes the reason for this modification

Description

A modificationRecord associates a single letter (such as modification "A") with modification author(s), address, and any optional external reference documents, in keeping with the AIAA draft standard.

Possible parents

`fileHeader`

Allowable children

`author`
`description`
`extraDocRef`

modificationRef

modificationRef — Reference to associated modification information

Content model

```
modificationRef :  modID
     EMPTY
```

Attributes

`modID` - the internal identifier of a modification definition

Possible parents

`provenance`

Allowable children

NONE

normalPDF

normalPDF — Defines a normal (Gaussian) probability density function

Content model

```
normalPDF :  numSigmas
     (bounds, correlatesWith*, correlation*)
```

Attributes

`numSigmas` - Indicates how many standard deviations is represented by the uncertainty values given later. Integer value > 0.

Description

In a normally distributed random variable, a symmetrical distribution of given standard deviation is assumed about the nominal value (which is given elsewhere in the parent element). The correlatesWith subelement references other functions or variables that have a linear correlation to the current parameter or function. The correlation subelement specifies the correlation coefficient and references the other function or variable whose random value helps determine the value of this parameter.

Possible parents

`uncertainty`

Allowable children

`bounds`
`correlatesWith`
`correlation`

provenance

provenance — Describes origin or history of the associated information

Content model

```
provenance :  [provID]
     (author+,
          (creationDate | functionCreationDate)
, documentRef*, modificationRef*, description?)
```

Attributes

`provID` (optional) - This attribute allows the provenance information to be cited elsewhere.

Description

The provenance element describes the history or source of the model data and includes author, date, and zero or more references to documents and modification records.

Possible parents

`fileHeader`
`variableDef`
`griddedTableDef`
`ungriddedTableDef`
`function`
`checkData`
`staticShot`

Allowable children

`author`
`creationDate`
`functionCreationDate`
`documentRef`
`modificationRef`
`description`

provenanceRef

provenanceRef — References a previously defined data provenance description

Content model

```
provenanceRef :  provID
     EMPTY
```

Attributes

`provID` - the internal identifier for the previously defined provenance

Description

When the provenance of a set of several data is identical, the first provenance element may be given a provID and referenced by later data elements as a space-saving measure.

Possible parents

`variableDef`
`griddedTableDef`
`ungriddedTableDef`
`function`
`checkData`
`staticShot`

Allowable children

NONE

reference

reference — Describes an external document

Content model

```
reference :  xmlns:xlink,  xlink:type,  refID,  author,  title,  [classification],
 [accession],  date,  [xlink:href]
     description?
```

Attributes

`xmlns:xlink`	- The value of this attribute must be "http://www.w3.org/1999/xlink". If omitted, it should be provided by the parser since it is specified in the DTD.
`xlink:type`	- The value of this attribute must be "simple". If omitted, it should be provided by the parser since it is specified in the DTD.
`refID`	- an internal identifier for this reference definition
`author`	- the name of the author of the reference
`title`	- the title of the referenced document
`classification`	(optional) - the security classification of the document
`accession`	(optional) - the accession number (ISBN or organization report number) of the document
`date`	- the date of the document, in ISO 8601 (YYYY-MM-DD) format
`xlink:href`	(optional) - a URL to an on-line copy of the referenced document

Description

This element gives identifying (citation) information to an external, possibly on-line, reference document, including a user-specified author, title, classification, accession number, date and URL.

Possible parents

`fileHeader`

Allowable children

`description`

signal

signal — Documents an internal DAVE-ML signal (value, units, etc.)

Content model

```
signal :
     (
          (
               (signalName, signalUnits)
 |
               (
                    (varID | signalID)
)
)
, signalValue, tol?)
```

Attributes

NONE

Description

This element is used to document the name, ID, value, tolerance, and units of measure for checkcases. When used with checkInputs or checkOutputs, the signalName subelement must be present (since check cases are viewed from "outside" the model); when used in an internalValues element, the varID subelement should be used to identify the signal by its model-unique internal reference. When used in a checkOutputs vector, the tol element must be present. Tolerance is specified as a maximum absolute difference between the expected and actual value. The signalID subelement is now deprecated in favor of the more consistent varID.

Possible parents

checkInputs
internalValues
checkOutputs

Allowable children

signalName
signalUnits
varID
signalID
signalValue
tol

signalID

signalID — Gives the internal identifier of a varDef [Deprecated]

Content model

```
signalID :
     (#PCDATA)
```

Attributes

NONE

Description

Used to specify the input or output varID. Now deprecated; reuse of varID is best practice.

Possible parents

`signal`

Allowable children

NONE

Future plans for this element

The signalID element has been deprecated with 2.0 in favor of the more consistent varID element and will be unsupported in a future release of this specification.

signalName

signalName — Gives the external name of an input or output signal

Content model

```
signalName :
     (#PCDATA)
```

Attributes

NONE

Description

Used inside a checkCase element to specify the input or output variable name

Possible parents

`signal`

Allowable children

NONE

signalUnits

signalUnits — Gives the unit-of-measure of an input or output variable

Content model

```
signalUnits :
     (#PCDATA)
```

Attributes

NONE

Description

Used inside a checkCase element to specify the units-of-measure for an input or output variable, for verification of proper implementation of a model.

Possible parents

`signal`

Allowable children

NONE

signalValue

signalValue — Gives the value of a checkcase signal/variable

Content model

```
signalValue :
     (#PCDATA)
```

Attributes

NONE

Description

Used to give the current value of an internal signal or input/output variable, for verification of proper implementation of a model.

Possible parents

`signal`

Allowable children

NONE

staticShot

staticShot — Used to check the validity of the model once instantiated by the receiving facility or tool.

Content model

```
staticShot :  name,  [refID]
     (description?,
          (provenance? | provenanceRef?)
, checkInputs, internalValues?, checkOutputs)
```

Attributes

`name`

`refID` (optional) - points to a reference given in the file header

Description

Contains a description of the inputs and outputs, and possibly internal values, of a DAVE-ML model in a particular instant of time.

Possible parents

`checkData`

Allowable children

`description`
`provenance`
`provenanceRef`
`checkInputs`
`internalValues`
`checkOutputs`

tol

tol — Specifies the tolerance of value matching for model verification

Content model

```
tol :
     (#PCDATA)
```

Attributes

NONE

Description

This element specifies the allowable tolerance of error in an output value such that the model can be considered verified. It is assumed all uncertainty is removed in performing the model calculations. Tolerance is specified as a maximum absolute difference between the expected and actual value.

Possible parents

`signal`

Allowable children

NONE

uncertainty

uncertainty — Describes statistical uncertainty bounds and any correlations for a parameter or function table.

Content model

```
uncertainty :  effect
     (normalPDF | uniformPDF)
```

Attributes

`effect` - Indicates how uncertainty bounds are interpreted (enumerated)

- `additive`
- `multiplicative`
- `percentage`
- `absolute`

Description

The uncertainty element is used in function and parameter definitions to describe statistical variance in the possible value of that function or parameter value. Only Gaussian (normal) or uniform distributions of continuous random variable distribution functions are supported.

Possible parents

```
variableDef
griddedTableDef
ungriddedTableDef
```

Allowable children

```
normalPDF
uniformPDF
```

ungriddedTable

ungriddedTable — Definition of an ungridded set of function data [Deprecated]

Content model

```
ungriddedTable :  [name]
     (confidenceBound?, dataPoint+)
```

Attributes

`name` (optional) - the name of the ungridded table being defined

Possible parents

`functionDefn`

Allowable children

`confidenceBound`
`dataPoint`

Future plans for this element

Deprecated. Use ungriddedTableDef instead.

ungriddedTableDef

ungriddedTableDef — Defines a table of data, each with independent coordinates

Content model

```
ungriddedTableDef :  [name],  [utID],  [units]
     (description?,
          (provenance? | provenanceRef?)
, uncertainty?, dataPoint+)
```

Attributes

`name` (optional) - the name of the ungridded table

`utID` (optional) - an internal identifier name for the gridded table

`units` (optional) - the units of measure for the table values

Description

An ungriddedTableDef contains points that are not in an orthogonal grid pattern; thus, the independent variable coordinates are specified for each dependent variable value. This table definition may be specified separately from the actual function declaration; if so, it requires an internal utID identifier attribute so that it may be used by multiple functions.

Possible parents

`DAVEfunc`
`functionDefn`

Allowable children

`description`
`provenance`
`provenanceRef`
`uncertainty`
`dataPoint`

ungriddedTableRef

ungriddedTableRef — Reference to an ungridded table

Content model

```
ungriddedTableRef :  utID
     EMPTY
```

Attributes

`utID` - the internal identifier of a ungridded table definition

Possible parents

`functionDefn`

Allowable children

NONE

uniformPDF

uniformPDF — Defines a uniform (constant) probability density function

Content model

```
uniformPDF :
     bounds+
```

Attributes

NONE

Description

In a uniformly distributed random variable, the value of the parameter has equal likelihood of assuming any value within the (possibly asymmetric, implied by specifying two) bounds, which must bracket the nominal value (which is given elsewhere in the parent element).

Possible parents

`uncertainty`

Allowable children

`bounds`

variableDef

variableDef — Defines signals used in DAVE-ML model

Content model

```
variableDef :  name,  varID,  units,  [axisSystem],  [sign],  [alias],  [symbol],
 [initialValue]
     (description?,
          (provenance? | provenanceRef?)
, calculation?, isOutput?, isState?, isStateDeriv?, isStdAIAA?, uncertainty?)
```

Attributes

`name`	- the name of the signal being defined
`varID`	- an internal identifier for the signal
`units`	- the units of the signal
`axisSystem`	(optional) - the axis in which the signal is measured
`sign`	(optional) - the sign convention for the signal, if any
`alias`	(optional) - possible alias name (facility specific) for the signal
`symbol`	(optional) - UNICODE symbol for the signal
`initialValue`	(optional) - an initial and possibly constant numeric value for the signal

Description

variableDef elements provide wiring information - that is, they identify the input and output signals used by these function blocks. They also provide MathML content markup to indicate any calculation required to arrive at the value of the variable, using other variables as inputs. The variable definition can include statistical information regarding the uncertainty of the values which it might take on, when measured after any calculation is performed. Information about the reason for inclusion or change to this element can be included in an optional provenance subelement.

Possible parents

`DAVEfunc`
`bounds`

Allowable children

`description`
`provenance`
`provenanceRef`
`calculation`
`isOutput`
`isState`
`isStateDeriv`
`isStdAIAA`
`uncertainty`

variableRef

variableRef — Reference to a variable definition

Content model

```
variableRef :  varID
     EMPTY
```

Attributes

`varID` - the internal identifier of a previous variable definition

Possible parents

`bounds`

Allowable children

NONE

varID

varID — Gives the internal identifier of a varDef

Content model

```
varID :
     (#PCDATA)
```

Attributes

NONE

Description

Used to specify the input or output varID. Replaces earlier signalID element.

Possible parents

`signal`

Allowable children

NONE

References

[Acevedo07] : Acevedo, Amanda; Arnold, Jason; Othon, William; and Berndt, Jon : *ANTARES: Spacecraft Simulation for Multiple User Communities and Facilities* . AIAA 2007-6888, presented at the AIAA Modeling and Simulation Technologies Conference, 20 August 2007, Hilton Head, South Carolina.

[AIAA92] : American Institute of Aeronautics and Astronautics : *American National Standard: Recommended Practice for Atmospheric and Space Flight Vehicle Coordinate Systems*. ANSI/AIAA R-004-1992

[AIAA01] : AIAA Flight Simulation Technical Committee : " Standard Simulation Variable Names [http://daveml.nasa.gov/AIAA_stds/SimParNames_May_2007_v2.pdf] ", Draft, May 2007

[AIAA03] : AIAA Modeling and Simulation Technical Committee : " Standards for the Exchange of Simulation Modeling Data [http://daveml.nasa.gov/AIAA_stds/SimDataExchange_Jan2003.pdf] ", Preliminary Draft, Jan 2003

[Brian05] : Brian, Geoff; Young, Michael; Newman, Daniel; Curtin, Robert; and Keating, Hilary : *The Quest for a Unified Aircraft Dataset Format [http://www.siaa.asn.au/get/2411855949.pdf]* . Presented at the Simulation Industry Association of Australia, 2005.

[Durak06] : Durak, Umut; Oguztuzun, Halit; and Ider, S. Kemal : *An Ontology for Trajectory Simulation* . Proceedings of the 2006 Winter Simulation Conference, 2006.

[Hildreth08] : Hildreth, Bruce; Linse, Dennis J.; and Dicolo, John : *Pseudo Six Degree of Freedom (6DOF) Models for Higher Fidelity Constructive Simulations* . AIAA 2008-6857, presented at the AIAA Modeling and Simulation Technologies Conference, 18 August 2008, Honolulu, Hawaii.

[Hill07] : Hill, Melissa A.; and Jackson, E. Bruce : *The DaveMLTranslator: An Interface for DAVE-ML Aerodynamic Models [http://http://daveml/papers/2007_AIAA_DaveMLTranslator_paper.pdf]* . AIAA 2007-6890, presented at the AIAA Modeling and Simulation Technologies Conference, 20 August 2007, Hilton Head, South Carolina.

[ISO8601] : International Organization for Standards : " Data elements and interchange formats - Information interchange - Representation of dates and times [http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=40874&ICS1=1&ICS2=140&ICS3=30] " ISO 8601:2000, 2000

[Jackson04] : Jackson, E. Bruce; Hildreth, Bruce L.; York, Brent W.; and Cleveland, William : *Evaluation of a Candidate Flight Dynamics Model Simulation Standard Exchange Format [http://dscb.larc.nasa.gov/DCBStaff/ebj/Papers/aiaa-04-5038-DAVEdemo1.pdf]* . AIAA 2004-5038, presented at the AIAA Modeling and Simulation Technologies Conference, 17 August 2004, Providence, Rhode Island.

[Jackson02] : Jackson, E. Bruce; and Hildreth, Bruce L. : *Flight Dynamic Model Exchange using XML [http://techreports.larc.nasa.gov/ltrs/PDF/2002/aiaa/NASA-aiaa-2002-4482.pdf]* . AIAA 2002-4482, presented at the AIAA Modeling and Simulation Technology Conference, 5 August 2002, Monterey, California.

[Lin04] : Lin, Risheng; and Afjeh, Abdollah A. : " Development of XML Databinding Integration for Web-Enabled Aircraft Engine Simulation ". *J. Computing and Information Science and Engineering* 4 3 American Society of Mechanical Engineers September 2004

[NAA64] : North American Aviation : *Aerodynamic Data Manual for Project Apollo* SID 64-174C, 1964

[W3C-XLINK] : World Wide Web Consortium (W3C) : “ W3C Recommendation: XML Linking Language (XLink) Version 1.0 [http://www.w3.org/TR/xlink/] ” 2001-06-27

[W3C-XML] : World Wide Web Consortium (W3C) : “ W3C Recommendation: Extensible Markup Language (XML) 1.0 [http://www.w3.org/TR/xml/] ” 2008-11-26

[wiki01] : Combined wisdom of the Internet : “ http://wikipedia.org/wiki/Spline_interpolation: "Spline Interpolation" [http://en.wikipedia.org/wiki/Spline_interpolation] ” Cited 2006

The editors would like to acknowledge the contributions, encouragement and helpful suggestions from the following individuals: Dennis Linse (formerly SAIC, now Vuelo Software Analysis), Jon Berndt (Jacobs Sverdrup), Daniel M. Newman (formerly Ball Aerospace, now Quantitative Aeronautics), Brent York (Indra), Bill Cleveland (NASA Ames), Geoff Brian (Australia's DSTO), J. Dana McMinn (NASA Langley), Peter Grant (UTIAS), Giovanni A. Cignoni (University of Pisa), Hilary Keating (Fortburn Pty. Ltd.), Trey Arthur (NASA Langley Research Center), Riley Rainey (SDS International), Jeremy Furtek (Delphi Research) and Randy Brumbaugh (Indigo Innovations).

Annex C DAVE-ML Website (Informative)

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/