Proposed ANSI National Standard

Flight Dynamics Model Exchange Standard

Draft 2010-04-23

Sponsored by

American Institute of Aeronautics and Astronautics

Approved XX Month 201X

Abstract

This is a standard for the interchange of simulation modeling data between facilities. The initial objective is to allow easy, straightforward exchanges of simulation model information and data between facilities. The standard applies to virtually any vehicle model (ground, air or space), but most directly applies to aircraft and missiles.

Contents

4

Foreword

This standard was sponsored and developed by the AIAA Modeling and Simulation Committee on Standards. Mr. Bruce Jackson of NASA Langley conceived Dynamic Aerospace Vehicle Exchange Markup Language (DAVE-ML). DAVE-ML is the embodiment of the standard in XML. The DAVE-ML reference document, including examples of its use, and the document type definition for the XML implementation are included in this standard (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	NASA Langley Research Center
Michael Madden	NASA Langley Research Center
Geoff Brian	Defence Science and Technology Organisation (DSTO)
Brent York	Indra Systems, Inc.
Michael Silvestro	Charles Stark Draper Laboratory, Inc.
Victoria Chung	NASA Langley Research Center
Bimal Aponso	NASA Ames Research Center
Jean Slane	Engineering Systems Inc.
Peter Grant	University of Toronto
William 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 201X.

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 Figure 1). This standard simulation interchange format is implemented in XML and is described fully in Annex B of this document.
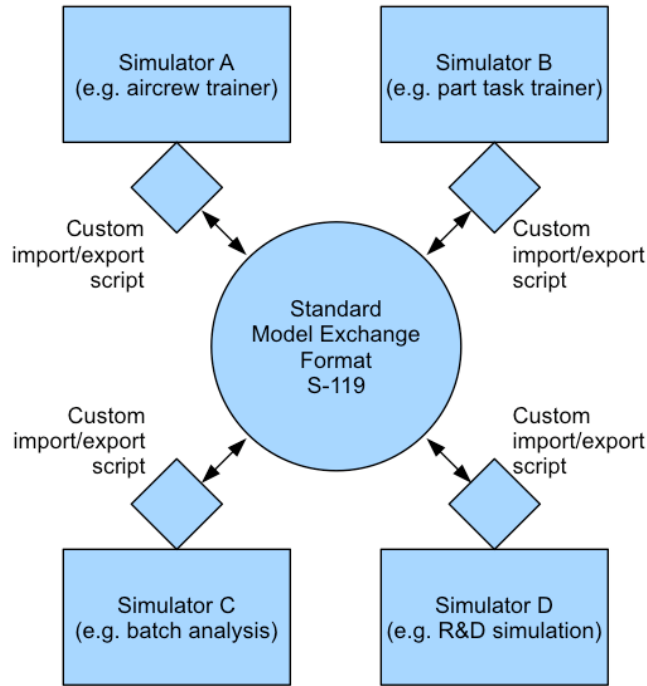
Figure 1 — Model exchange via a standardized format

The standard interchange format includes:

a) Standard variable name definitions — to facilitate the transfer of information by using these standard variables as a "common language." The interchange format can be used without using standard variable names. However, it will be more difficult because the exported model will have to include explicit definitions of all variables instead of just a subset unique to the particular model.

b) Standard function table definition — to allow easy transfer of non-linear function tables of arbitrary dimension.

c) Standard coordinate system and reference frame definitions — used by the variable names and function tables to clearly define the information being exchanged.

d) Standard static math equation representation — for definition of static equations forming part of aerodynamic, propulsive or other models.

A specialized grammar of XML provides a format for the exchange of this information, therefore each organization is required to design import/export tools that comply with 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 definitions 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 a detailed convention for representing simulation variables. The purpose of this is to unambiguously describe all variables within the model when it is exchanged between two simulation customers or facilities. The variable representation includes explicit specification of all coordinate systems, units, and sign conventions used. XML is used as the mechanism to facilitate automation of the exchange of the information. Based on the definitions in the standard, a list of recommended but non-obligatory simulation variable names is included in Annex A. This list of standard variable names should further simplify the exchange of information, but is not required for use of the standard.

The standard includes capabilities for a model to be self-validating and self-documenting, with the provenance of a model's components included within the model and transferred with it. Statistical descriptions of the quality of a model may also be included.

2 Tailoring

The requirements defined in this standard may be tailored to match the actual requirements of any particular program or project. Tailoring of requirements should 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

Not all aspects of this standard may be applicable to all models or simulation applications. 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. It is hoped that the team developing the new simulation environment would, if necessary, add to the list of standard variables and coordinate 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.

a) Apply the standard to the new development aspects of the project and all the function tables.

b) Assuming that most or all of the standard variable names and coordinate systems are applicable to the simulation, use them for the new code developed for the simulation.

c) 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 coordinate systems. However, in most cases the coordinate systems used in existing simulation environments will be covered in the standard coordinate system definitions herein (Section 5). Therefore the standard coordinate 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 application of the standard is critical. In this case, in each new software update, the original developers (or previous updaters) are probably no longer available, and the update is being performed by different personnel. Software developed using the standard should be easier for the new software team to understand. They are working with clear variable definitions with which they are familiar. The function table format is understood and the functions themselves are better documented. The coordinate system definitions are clear. Changes are recorded for the benefit of any future software update.

In simulations with a long expected life, use of the state, state derivative control and output conventions as part of the variable naming convention becomes critical as these variables form the core of the model

and the significant inputs. It is important that the personnel modifying the simulation are able to easily identify the states, state derivatives and controls.

2.2 Implementing the Standard in a non-flat or non-scalar namespace

The variable naming convention defined within the standard makes no assumption as to the hierarchy of data components, such as object-oriented model implementation or multi-dimensional storage of matrices and vectors. The standard can accommodate these implementations through the use of a period (.) inserted before the optional domain name (e.g. `aero.bodyForceCoefficient_X`) or through the use of an appropriate indexing mechanism for the chosen implementation language (e.g. `aerobodyForceCoefficient[0]` or `aerobodyForceCoefficient(1)`). However, it should not be expected that other members of the simulation community maintain implementation-specific conventions. Therefore, on export these variable constructs,should be converted to the flat, scalar namespace defined herein.

2.3 New and Reused Software Tailoring Guidance

The longer the expected life of the simulation, the more helpful the application of the standard becomes. The above tailoring guidelines may be categorized into two common situations; new and reused code.

New simulation code should

a) use coordinate system definitions (Section 5) that coincide with the definitions in the standard;

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

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

d) 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.

2.4 Creating New Variable Names and Coordinate Systems

The standard variable names and coordinate 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 values of the lift coefficient function. As a minimum, the independent variables used to define the function and their units, sign convention, and reference coordinate system must be defined. This need to precisely define variables is facilitated by having standard variable names and coordinate systems. Of course, new variable names, definitions, and other convenient coordinate systems may be used to exchange models between simulation facilities. However, in such cases, the exporters and importers must carefully define these variables and coordinate systems, or the exchanged model may not produce the desired results. Use of standard variable names and coordinate 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 coordinate systems or comments on existing standard variable names and coordinate 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*
IST-CR-90-50	*Distributed Interactive Simulation* (DIS Application Protocols, Version 2, March 1994)
www.w3.org/XML	*Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008-11-26*
www.w3.org/Math	*Mathematical Markup Language (MathML) Version 2.0 (Second Edition), 2003-10-21*
ISO 19111:2007	*Geographic information – Spatial referencing by coordinates*
ISO 1151-1:1988	*Flight dynamics – Concepts, quantities, and symbols – Part 1: Aircraft motion relative to the air*
ISO 1151-2:1985	*Flight dynamics – Concepts, quantities, and symbols – Part 2: Motions of the aircraft and the atmosphere relative to the Earth*
ISO 1151-3:1989	*Flight dynamics – Concepts, quantities, and symbols – Part 3: Derivatives of forces, moments and their coefficients*
ISO 1151-4:1994	*Flight dynamics – Concepts, quantities, and symbols – Part 4: Concepts, quantities and symbols used in the study of aircraft stability and control*
ISO 1151-5:1987	*Flight dynamics – Concepts, quantities, and symbols – Part 5: Quantities used in measurements*
ISO 1151-6:1982	*Terms and symbols for flight dynamics – Part 6: Aircraft geometry*
ISO 80000-1:2009	*Quantities and Units – General*

4 Vocabulary

4.1 Acronyms and Abbreviated Terms

AIAA	American Institute of Aeronautics and Astronautics
AND	aircraft nose down
ANR	aircraft nose right
ANSI	American National Standards Institute
CM	center of mass
DAVE-ML	Dynamic Aerospace Vehicle Exchange Markup Language
DIS	Distributed Interactive Simulation
DTD	Document Type Definition
FE	Flat Earth coordinate system
GE	Geocentric Earth fixed coordinate system
HLA	High Level Architecture
IC	initial condition
ISO	International Organization for Standardization
ISQ	International System of Quantities
MathML	Mathematical Markup Language
MRC	moment reference center
MSL	Mean sea level
RWD	right wing down
SI	Système Internationale d'Unites
URL	Uniform Resource Locator
WGS	World geodetic system
XML	eXtensible Markup Language

4.2 Terms and Definitions

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

Center of mass
This standard uses center of mass (CM) as the default location for several coordinate systems. The long-standing aeronautical tradition is to refer to this location as the center of gravity (CG). Center of gravity and center of mass are interchangeable for vehicle modeling on or above a large gravitational body in hydrostatic equilibrium like the Earth. However, the difference between CM and CG can become significant when a vehicle maneuvers near small, irregularly shaped gravitational bodies (e.g. asteroids).

8

Thus, center of mass is the more correct term for this important aerodynamic and kinematic reference point.

Coordinate system
A measurement system for locating points in space and attached to a reference frame (Stevens and Lewis, 2003). In this standard they will be orthogonal right-handed triads unless specifically noted.

Ground
Smooth surface of the earth at the nadir, not necessarily MSL.

Mean Sea Level
The zero elevation reference in the simulation, normally the geoid. Although many simulations treat MSL and the smooth surface of the earth as equivalent this is not always true. For example, the WGS84 model equates MSL with the geoid, not the smooth surface.

Presentation coordinate system
The specific coordinate system in which a variable or vector is presented (or expressed). For example, a specific vector may be presented (expressed) in the body axis coordinate system, the geocentric Earth (`ge`) coordinate system, or one of the alternatives presented in Section 5. The value of the vector's components (e.g. X, Y, & Z) differs depending on the presentation coordinate system. The presentation frame only determines how the vector is expressed as X, Y, and Z components, as the specific vector of an object is invariant with respect to any arbitrary coordinate frame (i.e. they are contravariant rank one tensors). In other words, the specific vector in two presentation frames differs only by a linear transformation (i.e. a rotation matrix). When one "presents" or "expresses" the vector in a presentation coordinate system, that presentation coordinate system is treated as if it is instantaneously fixed relative to the observer for the given time t (even if the presentation coordinate system is translating and rotating relative to the observer).

The presentation coordinate systems is identified in variable names by the initial coordinate system prefix.

Reference frame
Frames for short. A general definition for a reference frame is: three or more non-colinear points on a rigid body define a reference frame (Stevens and Lewis, 2003). Unlike a coordinate system, a reference frame has no fixed origin, it is in essence a rigid body wherein all points are fixed in position relative to each other. The location of a point or vector in a frame is expressed using a specified coordinate system. Any number of points or vectors may be expressed with any number of coordinate systems (with no relative motion) in the same frame. For example, the Earth is often a reference frame and may have the geocentric Earth (`ge`) coordinate system attached to the center, and any number of user defined topodetic coordinate systems (such as runways or launch sites) used to locate and orient fixed objects on the Earth. Note however an object moving on or above the surface of the Earth would be in a different reference frame.

Reference coordinate system
The coordinate system that defines the frame of interest of a rotational measurement such as attitude, angular rate or angular acceleration. Identified in variable names by the "`Of`" component. If not present, the default reference coordinate system is the body coordinate system.

Reference point
The point of interest of a translational measurement such as position, velocity or acceleration. Identified in variable names by an "`Of`" component. If not specified, the default reference point is the ownship's center of mass.

Relative coordinate system
A coordinate system that defines the origin of a measurement such as position, velocity or acceleration (translational or rotational). Identified in variable names by the "`Wrt`" component. If not specifed, the default relative coordinate system is the same as the presentation coordinate system for translation and

the locally-level coordinate system for rotations.

Observer coordinate system
A coordinate system from which motion of a point (a velocity, acceleration or higher derivative) is observed (or measured). In many cases this coordinate system is in the same reference frame as the relative coordinate system, however in the most general case, may exist in a different frame. The magnitude and direction of velocity (and higher derivatives) differ depending on the <u>observer</u> coordinate system due to the fact that the relative coordinate system may be in motion relative to the observer. Identified in variable names by the "`ObsFr`" component. If not specified, the observer coordinate system defaults to the same coordinate system as the relative coordinate system.

Velocity
The first derivative of position; in the general case, can be applied to either translational or rotational rate-of-change of position. This term normally applies to translational motion; the rotational equivalent is normally called "angular rate."

Inertial velocity
The special case of a velocity for which the relative reference and observer coordinate systems are an inertial frame.

Breakpoint
A value of an independent variable at which the value of its dependent variable is specified, 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
An output that is obtained by evaluation of a tabulated function or a MathML expression

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

Independent variable
The input(s) to a function table or a MathML expression

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

One-dimensional table
A table whose values are based upon only one independent variable

EXAMPLE $C_L(\alpha)$ may be represented by a one-dimensional table.

Two-dimensional table
A table whose values are based upon two independent variables

EXAMPLE $C_L(\alpha,\beta)$ may be represented by 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 (and state derivatives)
In the formulation of a non-linear simulation model shown as

$$\dot{x} = f_1\{x(t), u(t), w(t)\}$$
$$y = f_2\{x(t), u(t)\}$$

where

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

y represents a vector of the simulation outputs

w represents a vector of disturbances

Function Table
The numeral set of data points used to represent the value of an independent variable as a function of the value(s) of one or more independent variables

EXAMPLE $C_L(\alpha,\beta)$ may be represented by a function table.

Gridded Table
A multi-dimensional function table in which all independent variable breakpoints are constant across the function range, but not necessarily evenly spaced

NOTE 1 This is sometimes called an orthogonal table.

NOTE 2 All one-dimensional tables are gridded tables.

EXAMPLE   A gridded two-dimensional function

Gridded Function $f(x_1,x_2)$

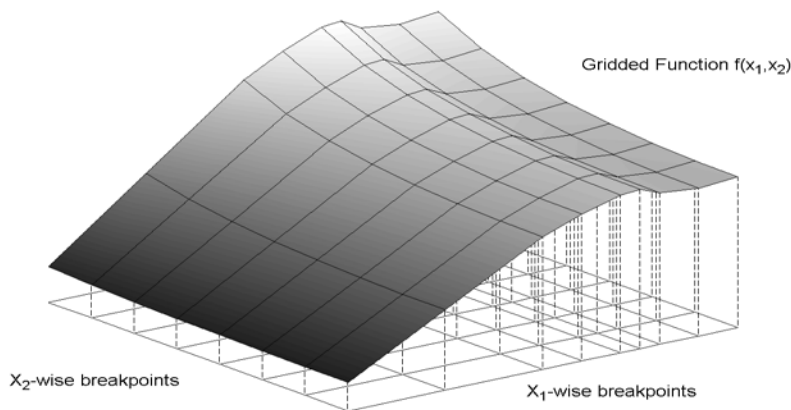X_2-wise breakpoints X_1-wise breakpoints

Ungridded Table
A multi-dimensional function table in which the independent variable breakpoints need not be constant across the function range

NOTE   This is sometimes called a non-orthogonal table.

EXAMPLE   An ungridded two-dimensional function

Ungridded function $f(x_1, x_2)$

X_2-wise regular spacing

X_1-wise irregular spacing
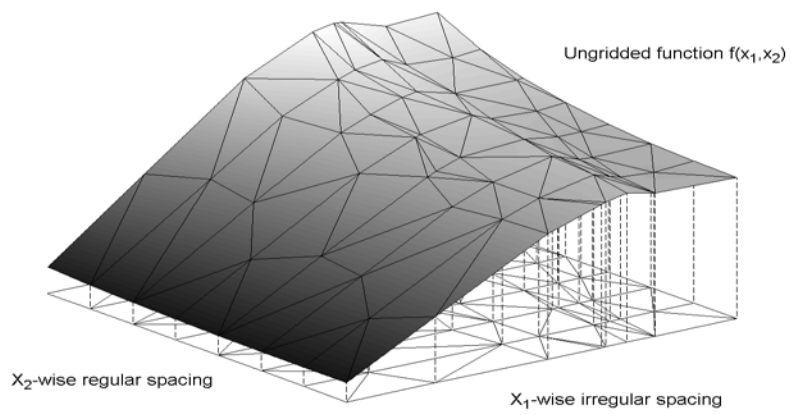
5   Standard Simulation Coordinate Systems

5.1   Background / Philosophy

Most of the coordinate 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.

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

5.1.1   Coordinate System Conventions

In general, ANSI/AIAA R-004-1992 is the normative reference for coordinate system definitions. These coordinate systems are discussed in Table 1. However, it is important to emphasize the correlation of the AIAA document and the *Distributed Interactive Simulation* (DIS) coordinate systems. The geocentric Earth fixed coordinate system and body coordinate system are both used in DIS and High Level Architecture (HLA) simulations.

5.1.1.1   Geocentric Earth Fixed Coordinate System

The Geocentric Earth Fixed Coordinate System (coordinate system 1.1.3 of Table 1) is identical to the DIS "Geocentric Cartesian Coordinate System" (also referred to as "World Coordinate System" in the DIS).

All variables referenced to this coordinate system use "`ge`" as part of their name for the Geocentric Earth Fixed Coordinate System. This coordinate system is also frequently called "Earth-centered, Earth-fixed."

5.1.1.2   Body Coordinate System

Another standard coordinate system is the Body Coordinate System (coordinate system number 1.1.7 in ANSI/AIAA R-004-1992). This is identical to the DIS "Entity Coordinates System." The body coordinate system is identified in the variable names by "`body`".

5.1.1.3   Additional Coordinate Systems

In addition to the coordinate systems defined in ANSI/AIAA R-004-1992, this standard has added the Moment Reference Center, Flat Earth, Locally Level, and Vehicle Reference (or Structural) coordinate systems. The Moment Reference Center coordinate system is a special case body coordinate system (number 1.1.7 in ANSI/AIAA R-004-1992, number 1.1.5 in ISO 1151-1:1998). The Flat Earth and Locally Level coordinate systems are respectively variants of the Normal Earth-fixed coordinate system (number 1.1.4 in ANSI/AIAA R-004-1992, number 1.1.2 in ISO 1151-1:1998) and the Vehicle-carried normal Earth coordinate system (number 1.1.6 in ANSI/AIAA R-004-1992, number 1.1.4 in ISO 1151-1:1998). Both these coordinate systems are normally used in conjunction with an assumption that the Earth forms an inertial reference frame. They are useful for simple simulations, and for creating vehicle model validation data. The Vehicle Reference coordinate system is a body coordinate system that may be used to locate vehicle components within the structure of the vehicle.

The moment reference center coordinate (MRC) system may be used to locate objects in the vehicle. Its axes are aligned with the body coordinate system, however, its origin is fixed at the moment reference center of the vehicle while the body coordinate system origin is at the center of mass (CM) and moves as the center of mass (CM) moves.

The moment reference center coordinate system is identified in the variable names by "`mrc`".

The flat Earth coordinate system is based on a fixed, non-rotating, flat Earth with no mapping to a round Earth coordinate system, and therefore, latitude and longitude are inappropriate (but can be scaled for

small maps). The purpose of this coordinate system is to allow, if desired, vehicle checkout simulation to be performed in this coordinate system. This simplifies the use of this standard by simulation facilities that do not normally use a round or oblate spheroidal, rotating Earth model.

The flat Earth coordinate system is identified in the variable names by "`fe`".

The locally level coordinate system is the reference coordinate system for angles and angular motion. It's origin is fixed at the vehicle center of mass.

The locally level coordinate system is identified in variable names by "`ll`".

The vehicle reference coordinate system is used to locate vehicle components. It is fixed to the vehicle structure and does not move. The specific definition differs for each vehicle. Sometimes the vehicle system may be the weight and balance reference system for the vehicle. The X origin is often in front of the vehicle, the Y origin in the centerline of the vehicle and the Z origin below the vehicle. The X-axis is often called the fuselage station and is often positive aft, the Y-axis is called butt line and is often positive to right, and the Z-axis is the waterline and is often positive up. However, these definitions may change with the vehicle and a manufacturing reference system may instead be used.

The vehicle reference system is identified in variable names by "`vrs`".

5.2   Complete List of Coordinate Systems

The coordinate systems that are referenced are taken largely from paragraph 1.1 of ANSI/AIAA R-004-1992. The moment reference center, flat Earth and locally level coordinate systems for atmospheric flight simulation approximation are additional to that reference. A vehicle reference coordinate system is added for the purpose of locating systems and subsystems in the vehicle. Table 1 is the comprehensive list of coordinate systems that may be used under this standard.

The first column in Table 1 provides the abbreviation recommended for each coordinate system. The coordinate system may be referenced in a variable name by use of its abbreviation. See Section 6 on the variable naming convention.

Table 1 — Standard coordinate systems

Reference Abbreviation	R-004-1992 Paragraph Number	Term	Definition	Symbol
`ei` (for Earth centered inertial)	1.1.1	Geocentric inertial coordinate 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 coordinate system	A right-hand coordinate system, fixed relative to and rotating with the Earth, with the origin and axes directions chosen as appropriate.	$x_0 y_0 z_0$
`ge` (also called Earth-centered,	1.1.3	Geocentric Earth-fixed coordinate 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	$x_G y_G z_G$

14

Reference Abbreviation	R-004-1992 Paragraph Number	Term	Definition	Symbol
Earth-fixed)			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)	
	1.1.4	Normal Earth-fixed coordinate system	An Earth-fixed coordinate 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)	$x_O y_O z_O$
vo	1.1.5	Vehicle-carried orbit-defined coordinate system[a]	A system with the origin fixed in the vehicle, *(the default being 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 coordinate 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, *the default being* the center of mass.	$x_O y_O z_O$
body	1.1.7	Body coordinate system[a]	A system fixed in the vehicle, *with the default origin being the center of mass,* consisting of the following axes:	$x\,y\,z$
		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 CM forward towards the nose in the vertical plane of symmetry. It is also normally parallel to the waterline of the vehicle.	x
		Lateral axis	An axis normal to the reference plane and positive to the right of the x-axis (henceforth, positive to the right).	y
		Normal axis	An axis that lies in or parallel to the reference plane, whose positive direction is chosen to complete the orthogonal, right-hand system xyz.	z

Reference Abbreviation	R-004-1992 Paragraph Number	Term	Definition	Symbol
mrc (for moment reference center)	none	A body coordinate system (not in R-004) Longitudinal axis Lateral axis Normal axis	This is a body coordinate system. The default origin is not fixed at the center of mass, but at the moment reference center (mrc) and therefore does not move. It consists of the following axes: 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 MRC forward towards the nose in the vertical plane of symmetry. It is also normally parallel to the waterline of the vehicle. An axis normal to the reference plane and positive to the right. An axis that lies in or parallel to the reference plane, whose positive direction is chosen to complete the orthogonal, right-hand system xyz.	$x_M y_M z_M$
wind (for wind coordinate system)	1.1.8	Air-path system[a] x_a-axis; air-path axis y_a-axis; lateral air-path axis; cross-stream axis z_a-axis; normal air-path axis	A vehicle carried system with the origin fixed in the vehicle, *located at* the center of mass, consisting of the following axes: An axis in the direction of the vehicle velocity relative to the air An axis normal to the air-path axis and positive to the right of the plane formed by the x_a and z_a axes. An axis ▪ in the reference plane or, if the origin is outside that plane, parallel to the reference plane, and ▪ normal to the air-path axis. The positive direction of the z_a-axis is chosen so as to complete the orthogonal, right-hand system $x_a y_a z_a$.	$x_a y_a z_a$ x_a y_a z_a
sa (for stability axis system)	1.1.9	Intermediate coordinate system[a]	A system with the origin fixed in the vehicle, *located at* the center of mass, consisting of the following axes.	$x_s y_s z_s$

Reference Abbreviation	R-004-1992 Paragraph Number	Term	Definition	Symbol
		x_s-axis	The projection of the air-path x 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_s-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_s-axis	An axis that 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 coordinate system[a]	A system with the origin fixed in the vehicle (*at* 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 coordinate system[a] (USA practice: aeroballistic coordinate system.)	A system with the origin fixed in the vehicle, *at* the center of mass, in which the x_t-axis is coincident with the x-axis in the body coordinate system (1.1.7). The y_t-axis is perpendicular to the plane formed by the x_t-axis and the velocity vector, positive to the right. The z_t-axis is formed to complete the orthogonal, right-hand system.	$x_t y_t z_t$
`fe`	None	Flat Earth system (not in R-004)	The Flat Earth coordinate system origin is situated on the Earth's surface directly under the center of mass of the vehicle at the initialization of the simulation. The X_{FE}-axis points northwards and the Y_{FE}-axis points eastward, with the Z_{FE}-axis down. The X_{FE} and Y_{FE} axes are parallel to the plane of the flat Earth.	$X_{FE} Y_{FE} Z_{FE}$
`ll`	None	Locally Level coordinate system (not in R-004)	A vehicle related coordinate system (1.1.6) with the origin instantaneously at the ownship center of mass. The Z_{LL}-axis passes through the vehicle center of mass and points towards the nadir. The X_{LL}-axis is parallel to the smooth surface of the Earth and oriented toward true north in the geometric Earth model. The Y_{LL}-axis is	$X_{LL} Y_{LL} Z_{LL}$

Reference Abbreviation	R-004-1992 Paragraph Number	Term	Definition	Symbol
			parallel to the smooth surface of the Earth completing the right hand triad (East). The locally level coordinate system is valid for geometric or flat Earth models.	
vrs	None	Vehicle Reference system (not in R-004)	A vehicle fixed coordinate system used to locate items in the vehicle. It is often the weight and balance coordinate reference system for the vehicle, or the manufacturing coordinate reference system. The VRS may not be a right-handed coordinate system. X-axis is the longitudinal reference. It may be the Fuselage Station line, normally 0 being in front of the vehicle with the coordinate increasing aft. Y-axis is the lateral reference. It may be the Butt line perpendicular to the vertical symmetric plane of the vehicle and in the geometric center of the vehicle. Positive to the right facing forward (Starboard) Z-axis is the vertical reference. It may be the Waterline and its origin is normally under the vehicle, positive up.	

[a]*By default*, the origins of the coordinate systems selected from ANSI/AIAA-R-004-1992 1.1.5 through 1.1.11 coincide *and are at the center of mass*. If that is not the case, it is necessary to distinguish the different origins by appropriate suffixes and additional coordinate system references.

[b]The reference plane should be a plane of symmetry, or a clearly specified alternative. *This may be specified by the vehicle reference system (vrs).*

[c]Italics indicate clarifications to ANSI/AIAA-R-004-1992.

5.3 Summary

This coordinate system standard should be followed for all future equations of motion. It is necessary for unambiguous reference to coordinate systems in simulation variable names.

5.4 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)

ISO 1151-1:1988, *Flight Dynamics – Concepts, quantities and symbols – Part 1: Aircraft motion relative to the air, 15 April 1988.*

ISO 1151-3:1989, *Flight dynamics – Concepts, quantities, and symbols – Part 3: Derivatives of forces,*

moments and their coefficients, 01 April 1989.

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 coordinate system definitions are part of this standard to facilitate communication. They provide a "common language" for information exchange. For example, to unambiguously exchange a function representing a lift coefficient, the minimum information required to be transmitted includes the independent variables used to define the function (such as angle-of-attack, angle-of-sideslip, Mach number, Reynolds number and aircraft configuration), their units, their sign conventions, and their reference coordinate systems. Such an exchange will be facilitated by using standard variable names and coordinate systems.

If a model uses standard variable nomenclature the information defining the model data may be exchanged entirely by reference to this standard. Additionally, adherence to the variable naming convention included herein will allow the list of standard variables to grow as needed by the user community. Use of the convention to maintain consistent variable names will ease user workload and maximize the benefits to be obtained from this standard.

Positions, angles, velocities and angular velocities referred to in this standard are defined in accordance with ANSI/AIAA R-004-1992.

6.2 Variable Naming Convention

The purpose of the naming convention is to provide guidance for the creation of variable names consistent with the standard variable names (Annex A). This will allow expansion of Annex A over time, further expanding the set of names available to facilitate model exchanges.

Variable names are constructed from components that jointly serve to fully define the variable in its particular application. A combined mixed case and underscore variable name convention is used. In variable name components that consist of multiple words, the first letter of each word is capitalized (medial capitals). Where the simulation language in use allows it, and where the logic of the simulation requires it, an underscore may be replaced by a period (.) to indicate an object member, or by parentheses or brackets to indicate an array member.

The following general rules for naming all variables shall be followed.

a) Variables shall have meaningful names.

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

c) Names shall be constructed using US-ASCII 7-bit character encodings.

6.3 Variable Name Methodologies

There are three methods specified for defining variables consistent with this standard. Different methods are described for:

a) Position variables (linear or angular), arrays or structures

b) Motion variables (velocities, accelerations, or higher derivatives, both linear and angular), arrays or structures

c) All other variable names

The naming convention for position variables and for variables describing motion are different than other types of variables because of the general requirements to specify coordinate systems that uniquely define

position and velocity.

In additional, the following guidelines for capitalization when creating variable names are provided:

d) The first letter in the variable name is lower case. Similarly, the first letter in the prefix and the first component following the prefix are lower case.

e) The first letters in acronyms and abbreviations are capitalized.

f) Distinct components in variable names, after the first component, shall begin with a capital letter.

g) Units are not capitalized unless the unit abbreviation itself is.

6.3.1 The Physical Basis for the Position, Velocity, Acceleration (and Derivatives thereof) Naming Convention

To ground the discussion of variable names it is useful to refer to a standard dynamics text and to review the equation of Coriolis. Figure 2 below shows the derivation of the three-dimensional linear velocity of point p with respect to coordinate system **M** as observed from coordinate system **O**. Coordinate system **M** has translational and rotation motion relative to **O**; in the figure, ω depicts the angular rate of **M** with respect to **O**.
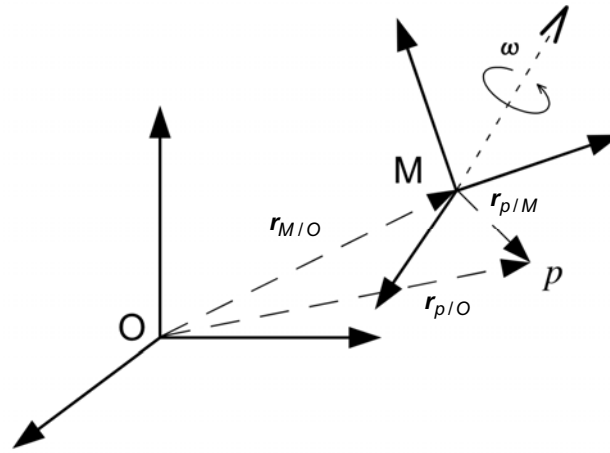
Figure 2 — Rotating reference frames and relative geometry

The velocity of p with respect to **M** as observed from **O** is the vector difference of the velocity of p with respect to **O** and the velocity of **M** with respect to **O** (both of which are observed from **O**):

$$^{O}\dot{r}_{p/M} = {}^{O}\dot{r}_{p/O} - {}^{O}\dot{r}_{M/O} \tag{1}$$

For this discussion, the vector notation used by Stevens and Lewis is used (see Figure 3 below) but other notations are equally valid. Using the equation of Coriolis (Stevens and Lewis equation 1.2-10), the velocity of p with respect to **O** observed from **O** is given by:

$$^{O}\dot{r}_{p/O} = {}^{O}\dot{r}_{M/O} + {}^{M}\dot{r}_{p/M} + \omega_{M/O} \times r_{p/M} \tag{2}$$

The left-hand side term and the first right hand side term in equation (2) are the velocity terms in the right-hand side of equation (1). Combining the two equations produces the following relationship:

$$^{O}\dot{r}_{p/M} = {}^{M}\dot{r}_{p/M} + \omega_{M/O} \times r_{p/M} \tag{3}$$

Equation (3) shows that the velocity of p with respect to **M** as observed from **O** does not equal the velocity of p with respect to **M** as observed from **M** in the general case. This illustrates the need to clearly specify, for derivatives of a linear position vector, the coordinate systems that define the point (this standard designates the point on the vehicle by "Of"), the origin from which the position is measured (this standard designates this by "Wrt"), and the frame in which the observation is made ("ObsFr"). An additional challenge is to achieve this specification using just the ASCII set of characters that are used to compose variable names.

Figure 3 shows how the notation used by Stevens and Lewis are mapped into the coordinate system components of the variable name schema. Within this standard, point p will be illustrative of the Of variable name component, coordinate system **M** will be illustrative of the Wrt variable name component and coordinate system **O** will be illustrative of the ObsFr variable name component.
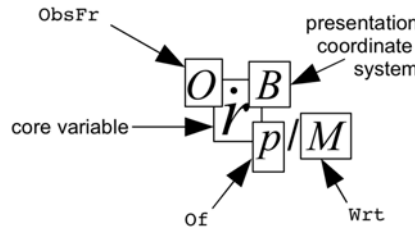
Figure 3 — Mapping Stevens and Lewis notation into variable name components

Another coordinate system that must be specified is that into which the three vector components are resolved in (this standard uses the term 'presentation coordinate system'). This is indicated in the Figure 3 as the right superscript B, which would indicate which coordinate system is used to resolve the vector into three scalar components.

In order to define position, velocity, acceleration or higher derivative variables (both translational and rotational), it is often necessary to specify each of these various coordinate systems. The kinematic requirements to clearly define these variables are presented below.

Positions: For positions (including rotational attitudes), the variable name must specify the origin from which the position is being measured. The name also must specify the coordinate system in which the vector is being resolved.

Take, for example, the core variable name

```
position
```

This name alone is meaningless. Therefore, it is necessary to describe what the position is representing:

```
positionOfPilotEye
```

This is still ambiguous as it necessary to specify both the point from which the pilot's eyepoint is being measured and into which coordinate system the position vector is being resolved.

```
positionOfPilotEyeWrtCm
```

This indicates what point is being located relative to what reference point, and therefore a relative position vector may be defined in 3-space. However, without knowing in which coordinate system the vector is being resolved, any number of coordinate systems could be used. Thus

```
bodyPositionOfPilotEyeWrtCm
```

This is a complete and unambiguous variable name representing a three-dimensional position. To define the name of any of the three body coordinate system axes, we need to append the name of the appropriate axis and units of measurement:

bodyPositionOfPilotEyeWrtCm_ft_X
This defines a scalar variable representing the X-body axis offset of the eyepoint from the center of mass, measured in feet.

Attitudes: Attitude (rotational position) is the orientation of one coordinate system relative to another; therefore, two coordinate systems are required (either explicit or implied) in the variable name to define an attitude. These axis systems are specified using the 'Of' and 'Wrt' components. The presentation coordinate system is not used because attitude (either as a quaternion, Euler angles, or rotation cosine matrix) is not a vector quantity resolved into X, Y, and Z components.

If the attitude is expressed as a set of Euler angles, the aeronautical convention of yaw-pitch-roll (3-2-1) rotation sequence is the default. To specify a different rotation sequence, the sequence should be appended to the Euler angle core name, e.g. eulerAngle313 for 3-1-3 rotations. To avoid confusion, for any rotation sequence other than 3-2-1, _First, _Second, _Third should be used for angle selectors in lieu of _Roll, _Pitch, _Yaw.

For example:

eulerAngleOfIssWrtEi_rad_Pitch
This variable represents the pitch attitude (rotation about the Y axis) of the user-defined International Space Station (Iss) coordinate system relative to the Earth-centered inertial (ei) coordinate system, measured in radians. This rotation uses the default yaw-pitch-roll (3-2-1) rotation convention.

eulerAngle313OfIssWrtOrion_rad_Second
This variable represents the second rotation angle of the International Space Station (Iss) coordinate system relative to another user-defined (Orion) spacecraft coordinate system measured in radians. This variable uses yaw-roll-yaw (3-1-3) rotation convention.

eulerAngleOfImuWrtBody_rad[_Roll|_Pitch|_Yaw]
This variable represents the three Euler angles of a user-defined inertial measurement unit (Imu) coordinate system relative to the body coordinate system.

eulerAngleOfImu1WrtImu2_rad[_Roll|_Pitch|_Yaw]
This variable represents the three Euler angles of a user-defined inertial measurement unit (Imu1) coordinate system relative to an Imu2 coordinate system, using the default yaw, pitch, roll (3-2-1) rotation convention.

Derivatives of position (translational or rotational): For variables representing derivatives of *translational* positions (velocities, accelerations and higher derivatives thereof) the observer coordinate system must be specified by the naming methodology. The observer coordinate system exists in the reference frame from which the movement (a velocity, acceleration or higher derivative) is observed (or measured). In many cases this is the same reference frame as the relative coordinate system (defined by the Wrt component) used to specify the position of the object being observed, but in the most general

case may be a different frame. The magnitude and direction of velocity (and higher derivatives) varies with selection of the observer's coordinate system since the relative coordinate system may be moving relative to the observer's coordinate system.

For *rotational* derivatives, it is unnecessary to specify an observer's coordinate system, but both coordinate frames that describe the rotational derivatives are necessary (both `Of` and `Wrt` components).

For example:

`eiAngularRateOfIssWrtOrion_rad_s_Y`
This variable represents the angular rate of the ISS coordinate system relative to the `Orion` spacecraft coordinate system. This is the angular velocity vector is resolved to the Earth-centered inertial (`ei`) coordinate system to measure the Y component in that system.

6.3.2   New Position Variables Naming Convention

The methodology for creating and defining a position variable (linear or angular) that is consistent with the requirements of this standard are as follows.

    a)   Each variable name may have up to nine components.

    b)   With the exception of the core name, all components are optional and should only be used if required by the application. Units must be specified unless the variable is non-dimensional and then the `_nd` units specification is encouraged.

The variable name components are listed immediately below. Descriptions of all the components follow in this section.

1.   *<variable domain>*

2.   *_<dynamic equation formulation prefix>_*

3.   *<presentation coordinate system>*

4.   *<core name>* — the only required component

5.   `Of`*<a point for positions or coordinate system for angles, normally on the vehicle>*

    If omitted in the variable name, `Of` defaults to the Cm for translational position and the body coordinate system for rotational position.

6.   `Wrt`*<"With respect to" a point or coordinate system>*

    If omitted in the variable name, `Wrt` defaults to the presentation coordinate system for linear positions and to the locally level coordinate system (`ll`) for angular position.

7.   `Ic` — initial condition designation

8.   *_<units>*

9.   *_<specific axis of the presentation coordinate system>*

Rarely are all 9 components of a name used.

For example:

```
universe_s_eiPositionOfCmWrtEarthIc_ft_X
```

- Axis
- Units
- Initial Condition Flag
- Relative ref. point/frame
- Reference point or frame
- Core name
- Coordinate System
- Dynamic component
- Domain

6.3.3 New Velocity, Acceleration or Higher Derivative Motion Variables Naming Convention

The methodology for creating and defining velocity and acceleration variables (or higher derivatives) consistent with the requirements of this standard are as follows:

a) Each variable name may have up to ten components.

b) With the exception of the core name, all components are optional and should only be used if required by the application. Units must be specified unless the variable is non-dimensional and then the _nd units specification is encouraged.
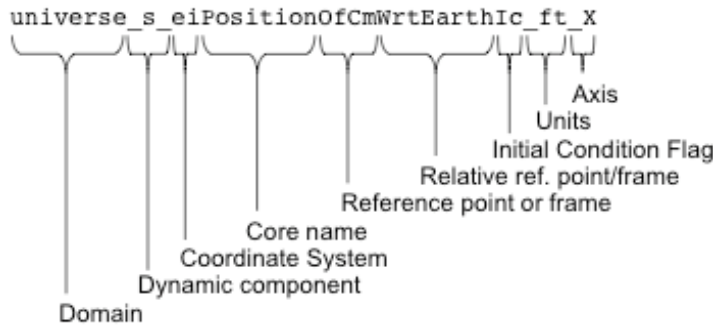
The variable name components are listed immediately below. Descriptions of all the components follow in this section.

1. *<variable domain>*

2. *_<dynamic equation formulation prefix>_*

3. *<presentation coordinate system>*

4. *<core name>* — the only required component

5. Of*<a point for translation or coordinate system for rotation, normally on the vehicle>*

   If omitted, Of defaults to the Cm for translation derivatives and the body coordinate system (body) for rotational derivatives.

6. Wrt*<"With respect to" a point, frame or coordinate system>*

   The Wrt component (relative coordinate system) may be omitted; if so, the relative coordinate system defaults to the presentation coordinate system for translational variables and the locally-level coordinate (ll) system for rotational variables.

7. ObsFr*<"Observed From" coordinate system, only used for translational motion>*

   If "ObsFr" is not specified, it defaults to the same coordinate system specified by the relative coordinate system (the Wrt component). If the relative coordinate system is not present, ObsFr defaults to the presentation coordinate system.

8. Ic — initial condition designation

9. *_<units>*

10. *_<specific axis of the presentation coordinate system>*

Rarely are all 10 components of a name used.

For example:

```
sensor_s_geVelocityOfIssWrtOrionObsFrGroundStationIc_ft_s_X
```

Core
name

Ref.
pt./frame

Observer Frame

Presentation
coordinate system

Relative pt./frame

Axis

Units

Dynamic component

Initial Condition Flag

Domain

6.3.4   New General Variables Naming Convention

The methodology for defining variables other than positions and derivatives thereof that is consistent with the requirements of this standard are as follows.
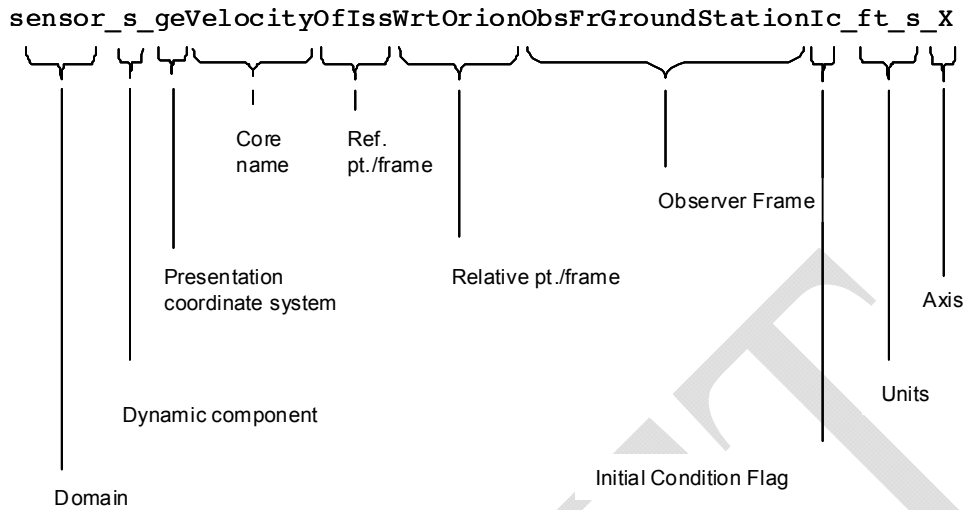
    a) Each variable name may have up to seven components.

    b) With the exception of the core name, all components are optional and should only be used if required by the application. Units must be specified unless the variable is non-dimensional and then the _nd units specification is encouraged.

The variable name components are listed immediately below. Descriptions of all the components follow in this section.

    1. *<variable domain>*

    2. *_<dynamic equation formulation prefix>_*

    3. *<presentation coordinate system>*

    4. *<core name>* — the only required component

    5. Ic — initial condition designation

    6. *_<units>*

    7. *_<specific axis of the presentation coordinate system>*

For example:
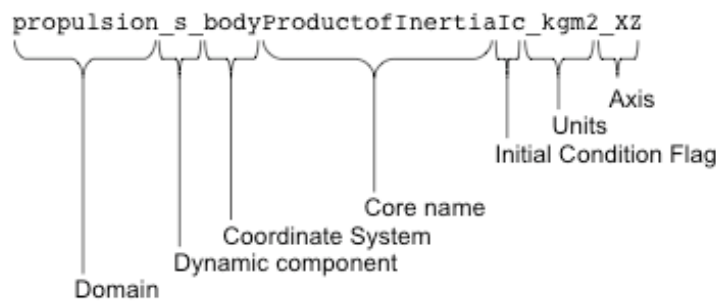
6.3.5   Adapting the Naming Convention to Hierarchical and Nested Data Representations

The naming methodology provides all the essential information about a variable in its name. This convention is concise for flat data representations (e.g. single-datum variables, arrays, common blocks). However, the convention can lead to repetition of information if applied to the members of hierarchical and nested data structures (e.g. classes, structures, records) since structure organization might parallel one or more of the variable name components. For example, a developer could create a structure to hold all of the variables in an aerodynamics model. The developer could then declare an instance of the structure with the name "aero;" the structure name would repeat information in the variable source domain component of its member variables.

An example of a "flat" variable name is:

        `aero_bodyForceCoefficient_X` where "aero" is the variable domain
Prepending the name of the structure to the standard variable name could result in an expression like the one below:

        `aero.aero_bodyForceCoefficient_X`
Such repetition can lead to unnecessarily long expressions. To avoid such repetition, developers may use the following guidelines to adapt the naming convention to hierarchical and nested data structures. First, when a level of a structure represents an organization of data that is equivalent to a variable name component, the developer should use the naming rules for that component to name instances of that structure level. Second, if a component of the variable name appears at a higher level or lower level, the developer should not include that component in the variable names at the current level. Using these guidelines, the example above can be changed to:

        `aero.bodyForceCoefficient_X`
If the developer made a further change to represent a vector as a structure and replaced the X, Y, and Z variables for the body force coefficient with that structure, the above expression would change to:

        `aero.bodyForceCoefficient.X`
To meet the intent of the guidelines, it is not necessary that the structure levels address the variable name components in the same order that the convention specifies for variable names. The intent of the naming convention is to unambiguously identify the information represented by a variable; it is not intended to shape data design. Thus, the name components can appear in a different order for a hierarchical or nested data expression. For example:

        `bodyForce_lbf.aero.X`
In this example, the data design is such that all the external forces on a vehicle (expressed in body coordinates and in units of pound-force) are collected in a structure whose instance is named "bodyForce" and whose members represent each generator of force as an instance of a structure representing a vector. The variable source domain appears after the core name and units due to the chosen data design. Even so, this data expression unambiguously identifies the variable as effectively as the equivalent scalar variable name, `aero_bodyForce_lbf_X`.

27

6.4 Components Used to Create Variable Names

6.4.1 Variable Domain Component

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.

In some cases the domain name component only provides background information when exchanging models. For example, in one simulation architecture the domain for `ambientPressure_N_m2` might be "`environment`" and another architecture the domain might be "`atmosphere`". The core component of the name is the key. For example:

> `environment_ambientPressure_N_m2` in one simulation architecture is identical to `atmosphere_ambientPressure_N_m2` in an another simulation architecture

However in some cases the domain component is critical. For example:

> `aero_bodyForce_lbf_X` and `thrust_bodyForce_lbf_X` are two different variables, both are body axis forces but one comes from the propulsion system model and one from the aerodynamic model. It is this type of variable where domain must be included.

Some domain examples are presented in Table 2. The domain names presented here are not part of any standard; instead they are presented here as examples.

Table 2 — Examples of domain names

`aero`	aerodynamic models
`airLaunchedWeapon`	modeling of munitions launched into the air that have their own dynamics; includes Missile as a sub-domain
`cautionAndWarning`	caution and warning simulation
`cockpit`	input/output from/to cockpit instruments and controls
`controlLaw`	simulation of a control algorithm
`controlLoading`	models of the control system feel
`controlSurface`	simulation of an aerodynamic control effector
`controlSystem`	collective model of control laws and control effectors on a vehicle
`electrical`	models of the electrical system
`engine` (or `thrust` or `propulsion`)	thrust generation models
`environment`	atmospheric models (ambient properties, wind, clouds, etc.)
`failureSystem`	failure modeling and fault injection
`fltDirect`	flight director models
`fuelSystem`	fuel system models
`gun`	model of vehicle mounted guns
`hydraulics`	hydraulic system models

landingGear	landing gear models
massProperties	tree-based modeling of vehicle mass and moments of inertia
missile	missile models. Domain could be more specific, for example `missileAim9x`.
motion	motion system models and algorithms
navigationDatabase	mapping of waypoints, airports, runways, legs, procedures, and navigation transmitters (all of which are subdomains)
navigationReceiver	modeling of signal-based navigation sensors
navigationTransmitter	modeling of navigation signal generators (e.g. radios, GPS)
parachute	parachute models
propulsion	models the collection of thrust generators (engines) on a vehicle
radar	models the radar system. Domain could be more specifc, for example `radarApg79`)
relGeom	relative state (position, velocity, and acceleration) of each vehicle to each other vehicle
sensor	models of sensors
sensorSystem	modeling the collection of sensors on a vehicle
sim	Domain encompassing control of the simulation, configuration of a simulation run, control of mathematical techniques such as integration type, etc.
vehicle	modeling of the vehicle as a cooperating system of other domain models
weaponSystem	collective model of guns and air-launched weapons on a vehicle
wheel	landing gear wheel models
world	world model (shape, dynamics, and reference time(s) plus navigation database and environment domains)
universe	domain encompassing world, vehicle, and relative geometry domains

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

Variable name examples using "aero" and "thrust" include:

   a) `aero_bodyForce_lbf_X`

   b) `thrust_bodyForce_lbf_X`

   c) `aero_bodyForceCoefficient_X`

d) `thrust.bodyForceCoefficient_X` — this is an example of `thrust` as a structure.

e) `thrust.bodyForceCoefficient(X)`

6.4.2 Dynamic Equation Formulation Prefix Component

The dynamic equation formulation prefix is used to identify the most important dynamic variables in the simulation, the states (x) and their derivatives ($\dot{x}$), inputs (u), outputs (y), and disturbances (w) as presented in the equation below. These variables characterize the resultant dynamic response of a vehicle as shown in the equations below. In addition to these variables, the standard allows the prefix to separately designate simulation control variables (c). Simulation control variables are used to modify the behavior of the model during simulation and are not part of the vehicle model, while inputs (u) are variables that represent the inputs to the vehicle model which may include pilot control positions. Finally, in simulation or analysis where noise and environmental disturbances are modeled, the disturbances (w) are the final component in the simulation of the total system dynamics.

$$\dot{x} = f_1\{x(t), u(t), w(t)\}$$
$$y = f_2\{x(t), u(t)\}$$

The prefix shall be separated from the body of the variable by an underscore (_) and from the domain name by an underscore (or a period if preceded by a member of a structure or class). The leading underscore is not permitted if a domain name is not present.

6.4.2.1 Identification of Simulation Model States and State Derivatives

The states (x) and state derivatives (dx) are those variables that make the simulation dynamic and are the key variables in a flight simulation model. Basically, any variable that is mathematically integrated is a state derivative. The result of integration of a state derivative over a period of time is a change in the value of the corresponding state over that time. This is true for any integration in a simulation. If the user controls the changes in all the states, they control the trajectory of the simulated model. The time histories of the states and inputs are the key variables required for validation. All outputs are computed directly or indirectly from states and inputs.

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, but allows the simulation engineer to explicitly identify states in the model implementation, making it easier to document and exchange the models.

Examples:

`x_bodyVelocityWrtEi_ft_s_X`   `x_` prefix indicates that this variable is a state

`dx_bodyAccelerationWrtEi_ft_s2_X`   `dx_` prefix indicates that this variable is a state derivative

6.4.2.2 Identification of Simulation Model Inputs

The simulation model inputs (u) are those variables that provide the pilot or autopilot inputs to the vehicle model. These are also called controls in many references. As with the states and state derivatives, the model inputs are key variables for validation. All model outputs are computed directly or indirectly from model states and inputs.

The formulation of the model itself determines which variables are inputs. This naming convention is not meant to standardize on any variable as an input, but allows the simulation engineer to explicitly identify

them, making it easier to document models, exchange them, and verify them.

Examples:

`u_controlSurfacePos_deg_avgAileron`	`u_` prefix indicates that this is a model input
`controlLaw_u_controlSurfacePos_deg_avgAileron`	same variable name with domain prefix
`controlLaw.u_controlSurfacePos_deg_avgAileron`	same variable name in a hierarchical architecture
`u_pilotControlPos_rad_long`	Another example of the longitudinal pilot input

6.4.2.3   Identification of Simulation Model Disturbances

The disturbances (w) are those variables that provide environmental disturbances or system noise to the simulation models.

Disturbances may be inserted into the vehicle model, environment, or equations of motion, depending upon implementation schemes. This naming convention is not meant to standardize on any variable as a disturbance, but allows the simulation engineer to explicitly identify disturbances, making it easier to document models, exchange them, and verify them.

Examples:

`w_bodyAngularRateTurbulenceWrtGe_deg_s_Yaw`	`w_` prefix indicates that this variable is a disturbance
`environment_w_bodyAngularRateTurbulenceWrtGe_deg_s_Yaw`	same variable name with domain (environment) added
`environment.w_bodyAngularRateTurbulenceWrtGe_deg_s_Yaw`	same variable name in a hierarchal architecture

6.4.2.4   Identification of Simulation Model Outputs

The simulation model outputs (y) are those variables that are the outputs of the physics of the simulation models as formulated by the state equations. This is meant to assist in the specification of the state equations, mainly to help simplify model exchanges between simulations used for analysis and those used for real-time man in the loop or hardware in the loop simulations.

Example:

`y_leftHorizonalActuatorRamPosition_deg`	`y_` prefix indicates that this variable is a model output.

6.4.2.5 Identification of Simulation Controls

The simulation controls (c) are those variables that provide the simulation operator control of the simulation [not to be confused with simulation model inputs (u)]. The simulation controls should not affect any vehicle states, state derivatives or outputs.

The software and hardware architecture of the simulation determines what variables are simulation controls. This naming convention is not meant to standardize on any variable as an control, but allows the simulation engineer to explicitly identify simulation controls. Clear definition of simulation controls makes validation of a simulation much easier after a model is exchanged.

Examples:

`c_simDuration_s`	`c_` prefix indicates that this variable is a simulation control
`c_deltaTime_s`	another example
`sim_c_deltaTime_s`	same variable name with domain added
`sim.c_deltaTime_s`	same variable name in a hierarchal architecture

6.4.3 Presentation Coordinate System Component

This is the coordinate or reference system to which the variable is referenced or in which it is measured (it is indicated by the "B" in Figure 3). Table 1 specifies the standard coordinate system abbreviations that should be used. If no coordinate system pertains to the variable or the core variable name needs no reference system to be unambiguous (e.g. Airspeed), this part of the variable name may be omitted.

6.4.3.1 Conventions Used

Earth fixed and local coordinate systems by convention use X, Y, Z, for both translation and (X, Y, Z), (Pitch, Roll, and Yaw), or (First, Second, Third) for rotation axis indices. The origin and attitude of local coordinate systems (flat Earth for example) may be user defined (such as N, E, D). Local coordinate systems are meant for runway, test range, target reference, navigational aids, etc.

6.4.3.2 Variable Name Examples

The following variable names are provided as examples.

`x_bodyAngularRate_rad_s_Roll`
 	
`x_bodyAngularRate_rad_s.Roll`
 	
`x_bodyAngularRate_rad_s_X`
	These are all equivalent variable names where `body` is the coordinate system and `roll` is the specific axis in the body axis system, roll indicating angular motion about the longitudinal axis relative to the locally-level frame (`Wrt` defaults to locally-level for rotational variables).
	
	NOTE In this example the variable is designated as a state.

`x_bodyVelocityWrtEi_m_s_X`
 	
`x_bodyVelocityWrtEi_m_s.X`
	These represent translational inertial velocity in the body coordinate system along the longitudinal axis (the translational variable analogous to the rotational variable above).
	
	These are all equivalent variable names.

`geVelocity_m_s_Y`	This represents a translational velocity with `ge` specified as the coordinate system and `Y` as the specific axis. This non-inertial velocity of the CM is relative to and measured in the geocentric earth-fixed (ge) coordinate system.

`bodyTurbulenceVelocityWrtGe_ft_s_Z`
 	
`bodyTurbulenceVelocityWrtGe_ft_s [3]`
 	
`bodyTurbulenceVelocityWrtGe_ft_s.Z`
	These are all equivalent variable names where `body` is the coordinate system and Z is the specific axis in the body coordinate system, `z` indicating vertical translational motion. Also illustrated as a vector and structure.

`runway22VelocityOfLeftWheelWrtTd_ft_s_Z`	Here `runway22` is the coordinate 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.
`bodyAccelOfPilotEyeWrtEi_m_s2_Y`	Here `body` is the coordinate system and `Y` is the specific axis, also indicating translational motion. Design pilot eyepoint is the point on the vehicle.
`x_eiVelocity_ft_s_X`	This is a case where the equations of motion are formulated such that the variable is a state, resolved in the earth centered inertial (ei) coordinate system
`eiVelocity_ft_s_X`	This is a case where the equations of motion are formulated such that the variable is not a state
`x_llVelocity_ft_s_X`	Locally Level coordinate system
`x_feVelocity_ft_s_X`	Flat Earth coordinate system
`bodyAngularRate_rad_s_Pitch`	
`bodyAngularRate_rad_s_Roll`	
`bodyAngularAccel_rad_s2_Yaw`	

Note that the standard allows (X, Y, Z) or (Roll, Pitch, Yaw) or (First, Second, Third) as selectors for rotational positions and derivatives, 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 use of X, Y, Z selectors is acceptable. The appropriate core variable name shall be used to indicate whether the variable is a translational or rotational variable.

6.4.4   Core Variable Name Component

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`   convention for velocities

- `angularRate`   convention for angular rates

33

- accel   convention for translational accelerations
- angularAccel   convention for angular accelerations
- pilotControlPos   conventions for pilot controls
- pilotControlRate
- pilotControlAccel
- pilotControlForce
- copilotControlPos   conventions for pilot controls
- copilotControlRate
- copilotControlAccel
- copilotControlForce
- controlSurfacePos   conventions for control surfaces
- **controlSurfaceRate**
- **controlSurfaceAccel**
- controlSurfaceHingeMoment
- liftCoefficient
- dragCoefficient
- forceCoefficient
- turbulenceVelocity
- angleOfAttack
- angleOfSideslip
- cosineOfAngleOfSideslip
- thrust
- torque

The following extended variable names are provided as examples.

- x_bodyAngularRate_rad_s_Roll
- bodyTurbulenceVelocityWrtGe_ft_s_X
- geVelocity_ft_s_Z
- geVelocity_m_s_Z
- pilotControlPos_deg_long
- pilotControlPos_deg_lat
- pilotControlRate_deg_s_pedal
- pilotControlAccel_deg_s2_long

- copilotControlPos_deg_long

- copilotControlPos_deg_lat

- copilotControlRate_deg_s_long

- copilotControlAccel_deg_s2_long

- controlSurfacePos_deg_elevator[number of surfaces]

- **controlSurfaceRate**_deg_s_rudder[number of surfaces]

- **controlSurfaceAccel**_deg_s2_aileron[number of surfaces]

- controlSurfaceHingeMoment_ftlbf_canard[number of surfaces]

- angleOfAttack_rad

- angleOfSideslip_deg

- cosineOfAngleOfSideslip

- controlSurfacePos_deg_aileron

- totalPressure_N_m2

- ambientPressure_N_m2

- totalLiftCoefficient

- aeroBodyForceCoefficient

- aeroBodyForce_lbf

- aeroBodyForce_N

- thrustBodyForce_N

6.4.5   Reference Point or Coordinate System ("Of")

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 that is being specified. "Of" is used to specify the point or object (this is point p in Figure 2).

For those who prefer shorter variable names, the standard adopts the convention that if the point or location on the vehicle is the center of mass for translational motion variables, then the reference point may be omitted. For rotational motion, the default reference coordinate system is the body axis coordinate system.

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

Examples of reference points are as follows.

- OfCm   the center of mass is the default point, so "OfCm" is normally omitted in any variable name.

- OfImu   or OfImu1, OfImu2, OfImuLtn200, etc.

- OfSensor   or OfRadar, OfFlir, OfRadarApg67, etc.

- OfMrc   for moment reference center

- `OfPilotEye`   for the pilot eye point

- `OfRadAlt`   for radar altimeter

- `OfTerrain`   a normal Earth-fixed coordinate system with origin where the vehicle nadir intersects the terrain

The following variable names are provided as examples.

`bodyPositionOfImuWrtCm_m[3]`	This three-element vector locates the Imu relative to the CM in the body axis coordinate system. Note that [3] indicates (for this example) that the referenced variable is a three-element vector.

`bodyPositionWrtImu_m[3]`
 	
`bodyPositionOfCmWrtImu_m[3]`
	Both of these vector names refer to the same quantity; it is the opposite of the vector above (they locate the CM relative to the Imu). In the first name "`OfCm`" is omitted since it is default.

`eulerAngleOfImuWrtBody_rad[3]`	This is the angular equivalent of the first variable above. The 3-2-1 rotation convention is implied.
`x_bodyAngularRateWrtEi_rad_s[3]`	Here element 1 would be about the X-axis (roll), element 2 would be about the Y-axis (pitch) and element 3 would be about the Z-axis (yaw). These are inertial rates since they are measured with respect to the Earth inertial (Ei) coordinate system.
`bodyVelocityWrtAir_ft_s_X`	`OfCm` is implied
`bodyVelocityOfCmWrtAir_ft_s_X`	Same meaning as above
`bodyVelocityWrtEi_ft_s_X`	Inertial velocity of the CM along the X-body axis
`bodyVelocityWrtEi_m_s_X`	Inertial velocity of the CM along the X-body axis in SI units
`heightOfCmWrtTerrain_ft`	`OfCm` may be omitted since it is the default
`heightOfRadAltWrtTerrain_ft`	
`heightOfTerrainWrtWgs84_ft`	Height of nadir intersection with terrain above the reference ellipsoid
`bodyPositionOfPilotEyeWrtCm_ft_X`	
`geLongitudeRateOfImu_deg_s`	
`longitudeOfImuWrtGe_deg` `geLongitudeOfImu_deg`	These are the same scalar quantity.
`bodyAccelOfPilotWrtEi_ft_2[3]`	Inertial acceleration vector in the body axis

6.4.6   Component Indicating Relative Reference Point or Relative Reference Coordinate System

The relative reference component is generally used in conjunction with the "reference point or location on the vehicle" component described in section 6.4.5. It is primarily used in variables describing position,

velocities and accelerations. This component defines the reference that the motion or position is relative to. This component, preceded by "Wrt" (with respect to) in the variable name, is the equivalent of coordinate system **M** in Figure 2, as noted in Figure 3.

For position variables, Wrt refers to the reference point for linear positions. For angular positions, Wrt refers to the relative coordinate system. For derivatives of position (velocities, accelerations, etc.) Wrt is used to define relative motion of two objects.

If "Wrt" is omitted then the default points or relative coordinate systems are:

the presentation coordinate system for linear positions and translational motion. For example,

bodyPositionOfImu_m[3]. Here WrtBody is implied.

the locally level coordinate system specified for rotational variables. For example:

eulerAngleOfImu_rad[3]. Here "WrtLl" is implied.

Note: Since for translational motion the "Wrt" defaults to the presentation coordinate system the variable:

bodyVelocity_f_s_X

while a valid variable, has little usefulness because, fully enumerated is:

bodyVelocityOfCmWrtBodyObsFrBody_f_s_X

Body velocity of the Cm with respect to the body is virtually always near zero. It would only represent the movement of the Cm within the body, due to cargo shift, fuel burn, etc. It would not represent velocity of the Cm with respect to a coordinate system outside the aircraft.

Examples of reference points are as follows:

- WrtCm   this is commonly used to clarify definitions of positions within the vehicle

- WrtEi   identifies a variable that is referenced to inertial space

- WrtMrc   moment reference center

- WrtTgt   aim point

- WrtImpact   the desired weapon impact point

- WrtAir   the local atmosphere, used to define air-relative (or wind-relative) motion

- WrtMeanSL   mean sea level

- WrtGe   the geocentric Earth fixed coordinate system

- WrtGround   a normal Earth-fixed coordinate system with origin where the vehicle nadir intersects the smooth surface of the Earth

- WrtTerrain   a normal Earth-fixed coordinate system with origin where the vehicle nadir intersects the terrain

The following linear and angular position variable names are provided as examples:

bodyPositionOfImuWrtCm_m[3]	This vector locates the Imu relative to the CM in the body axis coordinate system.
	WrtCm may be omitted since CM is the default reference point for linear position measurements.

`eulerAngleOfImu_rad[3]` `eulerAngleOfImuWrtLl_rad[3]`	This is the angular equivalent of the first variable above. In the first variable "`WrtLl`" is omitted since the locally-level coordinate system is the default relative coordinate system for angular positions.
`eulerAngleOfImuWrtBody_rad[3]`	This is a vector representing the alignment of the IMU with respect to the aircraft body.
`eulerAngleOfImu1WrtImu2_rad[3]`	This is a vector representing the attitude of Imu1 relative to Imu1.
`bodyPositionOfPilotEye_ft [3]`	`WrtBody` is implied since the Wrt defaults to the presentation coordinate system. Since the Body coordinate system origin is at the Cm, this variable represents the position of the Pilot design eye with respect to the Cm (+ = eye fwd: right: below Cm)
`geLongitudeOfImu_deg`	`WrtGe` is implied (since ge is the presentation coordinate system)
`geLongitude_deg`	`OfCm` is implied when not given.
`bodyPositionOfCmWrtMrc_ft[3]`	
`heightOfRunwayWrtMeanSL_ft`	

6.4.7   Component Indicating Observer's Coordinate System (Vehicle translational motion variables only)

For variables representing derivatives of linear positions (velocities, accelerations and higher derivatives thereof), the observer's coordinate system (indicated by `ObsFr` for "Observed From")) must be specified. The observer's coordinate system is in that reference frame from which the movement (a velocity, acceleration or higher derivative) is observed or measured. In many cases this is the same reference frame as the relative coordinate system (given by `Wrt`) but in the most general case it may be a different frame. The magnitude and direction of velocity (and higher translational motion derivatives) differ depending on the motion of the observer's coordinate system. This is the coordinate system shown as the upper left superscript in the Stevens and Lewis convention shown in Figure 3 and is illustrated by coordinate system **O** in Figure 2.

It is conventional to omit the observer's coordinate system when it is the same as the reference (`Wrt`) coordinate system. As noted in Section 6.4.6, when the `Wrt` coordinate system is omitted it defaults to the presentation coordinate system for translational variables, so when both the `Wrt` (reference) and the observer's (`ObsFr`) coordinate systems are omitted, the default observer's coordinate system is the presentation coordinate system.

It is neither necessary nor appropriate to specify the observer's coordinate system for rotational motion variables as rotations are invariant with the location of the observer.

Some examples are:

`geVelocity_ft_s_X`
Velocity of the ownship center-of-mass in the geocentric Earth coordinate system (velocity along the GE X-axis). Note that the reference point (`Of`), relative coordinate system (`Wrt`) and observer's coordinate system (`ObsFr`) default to the velocity of the center of mass with respect to and

observed from the geocentric Earth-fixed coordinate system since they are omitted.	

`geVelocityOfCmWrtGeObsFrGe_ft_s_X`

This is the same variable as the previous one with the reference point (`OfCm`), relative coordinate system (`WrtGe`) and observer coordinate system (`ObsFrGe`) all explicitly specified (they are not required since they are the defaults).

`bodyAccelWrtEi_ft_s2_X`

Acceleration of the vehicle center of mass relative to and observed from the Earth-fixed inertial (EI) coordinate system and presented in the body axis coordinate system (acceleration along the body X-axis). Naming convention components that can be implied are omitted.

`bodyAccelOfCmWrtEiObsFrEi_ft_s2_X`

This is the same variable as the previous one with the reference point, relative coordinate system and observer coordinate system all explicitly specified.

`bodyVelocityWrtAir_ft_s_X`

This is the air-relative velocity of the CM expressed in body coordinates; the `ObsFr` component is omitted to indicate that the observer's coordinate system is in the same frame as the steady state air mass reference frame (Air).

`bodyVelocityOfCmWrtAirObsFrAir_ft_s_X`

Same as the variable above, fully expressed

`bodyPositionOfPilotEyeWrtCm_ft_X`

The position of the pilot's eyepoint relative to the vehicle center-of-mass along the body X-axis. Note that the sign convention is clear: since the X-axis origin is at the center of mass, and is positive forward, the pilot's eyepoint position is positive when forward of the center of mass.

`bodyPositionOfPilotEyeWrtMrc_ft_X`

The position of the pilot's eyepoint relative to the moment reference center along the body X-axis. Note that the sign convention is clear: since the body X-axis origin is at the center of mass, and is positive forward, the pilot's eyepoint position is positive when forward of the MRC.

`bodyAccelOfPilotWrtEi_ft_s2_[3]`

This represents the inertial acceleration of the pilot, resolved into the vehicle's body axes. The `ObsFr` component is omitted, implying the motion is observed from the `Wrt` coordinate system (here, Earth Inertial).

`bodyAccelOfPilotWrtEiObsFrEi_ft_s2_[3]`	Same as above. `ObsFr` explicitly stated.
`bodyVelocityWrtEi_ft_s_X`	Inertial velocity of the CM along the X-body axis.
`bodyVelocityOfCmWrtEiObsFrEi_ft_s_X`	Same meaning as the previous variable, fully expressed

`runway22VelocityOfLeftWheelWrtTdObsFrTd_ft_s_Z`

This is the velocity of the wheel relative to the TD point observed from the TD coordinate system. The user must explicitly define the TD coordinate system, but logically it is in an Earth-fixed reference frame, probably with the origin at the desired touchdown point and aligned with the

runway.
`runway22VelocityOfLeftWheelWrtTd_ft_s_Z`
This is the same variable as above, since omitting `ObsFr` implies it is the `Wrt` coordinate system.
`velocityWrtGround_ft_s`
This scalar variable is commonly known as groundspeed.

6.4.8 Component Indicating Initial Variables

A convention proposed by this standard is adding "`Ic`" to the end of any variable name, before any units, to designate that the variable is an initial condition specification. This can be added to virtually any variable without an underscore separator, conceptually creating a constant, for example:

1. `x_bodyVelocityWrtEiIc_rad_s_X`

2. `grossWeightIc_N`

6.4.9 Units Suffix

The suffix is used to describe the units of the variable. The convention for the suffix is simple and is followed for all variables. When exchanging simulation models, the units of all variables must be specified and this is the mechanism to do so. This will also 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. If required for clarity, "`nd`" may be used in the units suffix to indicate a non-dimensional variable. Including units has the added advantage of making this standard consistent and acceptable in countries utilizing the international system of units. For example, airspeed is equally acceptable as a standard both for the U.S. system of units and the International system of units.

The standard uses an underscore (_) to separate the numerator from the denominator an analogy to exponential notation for the specification of units. For example, the unit expression for cubic feet per second squared (for example) would be ft^3s^{-2}. Eliminating the superscripts leaves `ft3s-2`. Separating the numerator from the denominator results in `ft3_s2`, since the negative sign in the denominator exponential term is dropped.

With few exceptions, only base units are supported; it is not allowed to have, for example, milliseconds (ms). Here the proper use would be to express that variable in seconds.

Further examples are as follows:

1. `trueAirspeed_ft_s` for feet per second (ft/s)

2. `trueAirspeed_m_s` for meters per second (m/s)

3. `trueAirspeed_kt` for knots (nautical miles per hour)

4. `bodyAccelWrtEi_ft_s2_X` for feet per second squared (ft/s^2)

The suffix shall be separated from the body of the variable name by an underscore. The standard unit notations are given in Table 3; SI units and standard abbreviations are included where available.

Table 3 — Abbreviation for units-of-measure in standard variable names

Time	
second	s^a
minute	min^e
hour	h^e
Length	
inch	inch
foot	ft
meter	m^a
nautical mile	nmi
statute mile	smi
kilometer	km
centimeter	cm
millimeter	mm
astronomical unit	ua^e

volt, alternating current	Vac^d
current (ampere)	A^a
frequency (hertz)	Hz^b
inductance (henry)	H^b
capacitance (farad)	F^b
charge (coulomb)	C^b
conductance (siemens)	S^b
resistance (ohm)	ohm^d

Other	
pressure, stress (pascal)	Pa^b
standard gravitational acceleration unit	g
luminous intensity (candela)	cd^a
luminous flux (lumen)	lm^b
illuminance (lux)	lx^b
amount of substance (mole)	mol^a
magnetic flux density (tesla)	T^b
magnetic flux (weber)	Wb^b
radioactive activity (becquerel)	Bq^b
absorbed dose (gray)	Gy^b
dose equivalent (sievert)	Sv^b
nautical mile per hour	kt
non-dimensional	nd

Force	
pound force	lbf
Newton	N^b
kilogram force	kgf
Mass	
kilogram	kg^a
pound mass	lbm
slug	slug
Solid Angle	
steradian	sr^b
Plane Angle	
degree	deg
radian	rad^b
revolution	rev

Temperature	
degrees Rankine	dgR
degrees Celsius	dgC^d
degrees Fahrenheit	dgF
Kelvin	dgK^c

Power, energy, work, heat	
energy (British thermal unit)	btu
energy (erg)	erg
energy (calorie)	Cal
energy (joule)	J^b
power (horsepower)	Hp
power (watt)	W^b
Electrical units	
potential (volt)	V^b
volt, direct current	Vdc^d

Notes:
> a. SI base unit (reference ISO 80000-1:2009, §6.5.2)
> b. SI derived unit (reference ISO 80000-1:2009, §6.5.3)
> c. SI base unit with modified abbreviation
> d. SI derived unit with modified abbreviation
> e. ISO recognized non-SI unit (reference ISO 80000-1:2009, §6.5.6)

6.4.10 Units-agnostic models

Some models have identical formulations whether the system of measurement is the SI system or the U.S. customary system. The units of the model's outputs depend only on the units of the values that are provided as inputs. These "units-agnostic" models can be reused in simulations with few or no conversions performed on inputs or outputs by the host simulation. To allow the exchange of units-agnostic models, the standard provides a set of abbreviations for generic units that represent the base units that differ between SI and U.S. customary system, namely length, mass, and temperature:

Unit	Abbreviation
length	L^{a}
mass	M^{a}
temperature	dgT^{b}

Notes:
> a. ISQ base unit. (reference ISO 80000-1:2009, §3.7)
> b. ISQ base unit with modified abbreviation

The two systems of measurement use the same unit for time (second). The U.S. customary system does not define units for the base quantities of electric current, luminous intensity, and amount of substance; the SI units (see ISO 80000-1:2009) fill this omission. Therefore, generic units are not required for time, electric current, luminous intensity or amount of substance; the SI unit for these quantities is used for variable names in unit agnostic models.

Examples of variable names using generic units:

- `bodyForce_ML_s2_Z`

- `bodyVelocityWrtGe_L_s_X`

- `thermalConductivity_ML_s3dgT`

Note that the last variable, thermal conductivity, would normally be published in SI units of W/(m-°K) [equal to kg-m/(s^3-K)] with a conversion factor to the typical English units of BTU/(hr-ft-°F), neither of which equals the English unit substitution in the variable name of slug-ft/(s^3-°R). So, a host simulation based on U.S. customary units would need to convert the published value to slug-ft/(s^3-°R) before passing the value to the model. (A unit conversion is not required in a host simulation based on SI units.) If this model were originally developed for that host simulation, the model developer would likely have placed the necessary conversions from BTU and hours (or from the published SI value) within the model; however, by formulating the model as unit agnostic, responsibility for conversion has been moved to the host simulation. The thermal conductivity example illustrates how a unit agnostic model may still require conversion of inputs or outputs by the host simulation.

6.4.11 Component Indicating Specific Axis, Coordinate Component or Reference

The last component is the specific axis, coordinate component or reference used within the coordinate system (coordinate systems are defined in Section 5). It may also indicate elements of vectors and arrays. It is separated from the units by an underscore (_). As can be seen in the examples, this component is appended last to keep the naming convention consistent for variables that are scalars or vectors. If the coordinate system is included in the name, the specific axis or reference should also be included.

Standard axes selector sets are:

    a) (X, Y, Z) for linear/translational motion,

    b) (X, Y, Z), (Roll, Pitch, Yaw) or (First, Second, Third) for angular motion.

When the specific axis or reference can logically be a vector or an array, the vector or array component may be convenient for a specific implementation. When coordinate system vectors are used, a right-handed triad in order (X, Y, Z) shall be used to avoid confusion. Due to differences between 0- and 1-based array indexing in various implementation languages, use of numerical indices is discouraged.

In the following examples, `Z` would be defined as a constant of either 2 or 3 depending on the implementation language array indexing convention.

Variable name examples:

`x_bodyAngularRate_rad_s_Roll`
 	
`x_bodyAngularRate_rad_s[Roll]`
 	
`x_bodyAngularRate_rad_s.Roll`
	Here `body` is the coordinate system and roll is the specific axis in the body coordinate system, `roll` indicating angular motion. Examples show alternate scalar and vector implementations and implementation as a structure.
	
	NOTE In this example the variable is designated as a state.

`bodyTurbulenceVelocityWrtGe_ft_s_Z` (standard)
 	
`bodyTurbulenceVelocityWrtGe_ft_s[Z]`
 	
`bodyTurbulenceVelocityWrtGe_ft_s.Z`
	Here `body` is the coordinate system and `z` is the specific axis in the body coordinate system, `z` indicating vertical translational motion.

`geVelocity_m_s_Y`	Here `ge` is the coordinate system and `Y` is the specific axis, also indicating translational motion.
`runway22VelocityOfLeftWheelWrtTd_ft_s_Z`	where `runway22` is the coordinate 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.
`bodyAccelOfPilotEyeWrtEi_m_s2_Y`	Here `body` is the coordinate system and `Y` is the specific axis, also indicating translational motion. Design pilot eyepoint location is the point on the vehicle.
`bodyProductOfInertia_slugf2_YZ`	Here, `_YZ` selects the element in the second row and third column of the inertial matrix.

6.5 Additional Discussion

Very rarely, if ever, are all 10 components of a name used. In the case of

    `x_bodyAngularRate_rad_s_Roll`

the following five components were used:

    1. prefix (`x_`) indicating that in this formulation of the equations of motion this variable is a state,

2. coordinate or reference system (`body`),

3. core name (`AngularRate`),

4. units suffix (`rad_s`), for radians per second

5. specific axis or reference (`Roll`).

In this case "variable source domain" was omitted because `x_bodyAngularRate_rad_s_Roll` is a single quantity defined by the laws of physics; there should not be separate body rates associated with aerodynamics and a propulsion system. 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`, `Wrt`, and `ObsFr` were omitted because the variable is describing motion about (`Of`) the CM and relative to the locally-level coordinate system. Recall that `Wrt` defaults to the locally-level frame for rotational motion and rotational motion variables do not allow specification of an observer's (`ObsFr`) coordinate system.

An `Ic` flag is not present, indicating that this variable does not specify an initial condition.

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

6.5.1   Discarded Conventions and Reasons

One convention considered eliminated the units suffix when the units were from a standard set, but this concept was discarded since always having the units associated with the variable name should help the developer maintain consistent units in the simulation and to reduce programming errors due to improper mixing of units. Consistent application of units in variable names should also reduce the software maintenance effort when a subsequent developer is trying to understand the code to make bug fixes, implement enhancements, or reuse the code.

6.5.2   Relationship with Markup Grammar, DAVE-ML

At present, this variable naming convention is targeted for use with the DAVE-ML XML grammar for model exchange (see Section 7). In DAVE-ML, the dynamic equation formulation prefix and the units suffix are stored as separate components (attributes or child elements) of the variable definition. Thus, including these in a variable name encoded in DAVE-ML would be redundant and a potential source of conflicting information.

The recommended practice is therefore 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 has two advantages.

1) The DAVE-ML grammar does not enforce naming rules; 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 to assist with the clear documentation of a model.

2) The convention allows XML processors to adopt the practice of automatically stripping and adding the prefix and suffix to the variable names, reducing the possibility of human error during translation.

6.6   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 4 for illustrative purposes.

Interpretation of the standard variable name annex is best given by example. Table 4 presents the standard variable defining the roll Euler angle, its axis system and positive sign convention (positive is RWD, or right wing down). Four name examples are provided. The table includes:

1) The symbol for that variable, Φ

2) The short name, `PHI` - the short name is included to accommodate standard variable definitions for legacy compilers with significant name length restrictions

3) One or more full names using the standard units conventions — generally, one full name with American convention units and one with SI units. Refer to section 6 for a list of the standard units and their abbreviations.

NOTE: While the variable naming convention described in Section 6 encourages the use of the <variable domain> component, this Annex does not include variable domains as part of the normative standard names. This is because the variable domain is normally dependent upon the simulation architecture, and as such is immaterial to the exchange of a simulation model unless the exchange is between facilities with similar architectures.

NOTE: Any suitable units may be used. In the example for `eulerAngle_Roll` both the `_deg` for degrees and the `_rad` 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 coordinate system. Refer to section 5 for a description of the standard coordinate systems.

5) The POSITIVE sign convention of the variable — RWD indicates that positive `eulerAngle_Roll` is right wing down. (See section A.2.1 for a list of sign convention acronyms)

6) Minimum value, normally only specified for angles

7) 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 an array, the first variable name in Table 4 is the standard definition of the Euler angle array. Again, `eulerAngle_rad(3)` would be the standard array using radians as the units and is fully compliant with the standard.

Euler angles are used in virtually any air vehicle simulation. While normally the coordinate system would be included in the name, it was not included due to the universal definition of Euler angles. A more rigorous name would be `llEulerAngleOfBodyWrtLl_deg[3]` which expresses all the defaults (the variable is the Euler angles of the body with respect to the locally level [ll] coordinate system and is presented in [or measured in] the locally level coordinate system). Aircraft simulations typically use Euler angles defined via the 3-2-1 angle rotation sequence (yaw, pitch, roll); other rotation sequences may be used but should be explicitly identified as in `EulerAngle313,` for example.

The standard allows use of any of the standard set of units (degrees or radians in this case).

Table 4 — Standard variable name table excerpt

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Min Value	Max Value
Vehicle Positions and Angles						
ε	EUL(3)	`eulerAngle_deg(3)` `eulerAngle_rad(3)`	Array of the ownship roll, pitch, and yaw Euler angles comprised of the elements defined below. LL (locally level) coordinate system.			

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

Φ	PHI	`eulerAngle_deg_Roll`
		
		`eulerAngle_rad_Roll`
			Roll Euler Angle, LL coordinate system.	RWD	-180,-π	180, π

θ	THET	`eulerAngle_deg_Pitch`
		
		`eulerAngle_rad_Pitch`
			Pitch Euler Angle, LL coordinate system	ANU	-90, -π/2	90, π/2

ψ	PSI	`eulerAngle_deg_Yaw`
		
		`eulerAngle_rad_Yaw`
			Yaw Euler Angle, LL coordinate system	ANR	-180,-π	180, π

6.7 Summary

While it is recommended that this naming convention be adopted for defining 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 "`VRW`," but that there exists a precise, unambiguous definition of the variable (true, indicated, or calibrated airspeed, etc.), including units and coordinate 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.

6.8 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.

International Organization for Standardization, *Quantities and units - General*, ISO 80000-1, First edition, 2009-11-15

United States of America Standards Institute: *USA Standard Code for Information Interchange,* USAS X3.4-1967, 1967-07-07

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

7.1 Purpose

This section explains the requirements that a standard simulation data format must be able to satisfy. It includes the content of defined functions and configuration management of the content. The definition of the DAVE-ML format includes data for these components.

This document also discusses conceptually how a data table should be accessed in an executable program.

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

7.2 Philosophy

Probably the greatest benefit of the standard to the simulation discipline is the definition of formats for the interchange of tabular data. Tabular data is used widely for non-linear function representation of aerodynamic, engine, atmospheric, and many other model parameters. The simplified interchange of such data should 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 formats. The intent of this standard is not to replace these local standards, but rather to define a format for communication that will allow each site to develop a single format converter to and from their local format. The DAVE-ML data representation is proposed as an exchange standard.

7.3 Design Objective

The first design objective of the standard data table format was to include all relevant information about real multi-dimensional functions, not just the data values. In the general case of a multi-dimensional table, the independent variables have different numbers of breakpoints, different breakpoints, and different valid ranges, which are all relevant to consistent evaluation of the function.

An equally important design objective was to allow the table to contain information on the data source (provenance, via reference), and a confidence interval for the data. Uses of confidence intervals within a model include direct computation of output confidence levels, estimation of output confidence intervals through Monte Carlo simulations, and mathematically combining different estimates of the same parameter at the same input values. Therefore, confidence statistics should be included when updating an existing or creating a new data set. DAVE-ML allows different types of confidence intervals, not all of which can be meaningfully combined.

The data format must also be easily read by computer or human, and be as self-documenting as possible.

7.4 Standard Function Table Data — An Illustrative Example

Figure 4 presents a fairly standard three-dimensional set of aerodynamic data typical of flight test or wind tunnel results. In the example, lift coefficient is a function of angle-of-attack, Mach number, and a control surface position. More generally stated, the function output (dependent variable) CLALFA is dependent on three inputs (independent variables): `angleOfAttack_deg`, `mach`, & `controlSurfacePos_deg_avgElevator`.

The example illustrates the following characteristics.

1) The number of breakpoints may be different for each independent variable. Data is presented for a different number of angle-of-attack (`angleOfAttack_deg`) points at each combination of Mach number (`mach`) and control position (`controlSurfacePos_deg_avgElevator`). For the first combination of Mach number and control position (`mach` = 0.6, `controlSurfacePos_deg_avgElevator` = 5) there are 17 angle-of-attack points. For the last combination of Mach number and control position (`mach` = 0.8, `controlSurfacePos_deg_avgElevator` = 0) there are 12 angle-of-attack points. There are also different numbers of Mach number points for each control position. The standard requires this to be represented as an ungridded table.

   In contrast, a gridded table would require an function value be defined for every combination of a fixed set of Mach, angle-of-attack and control position breakpoint values.

2) At some breakpoints, the values of the other independent variables are different. Again, this is a characteristic of an ungridded table.

3) The valid ranges of the independent variables are different, another ungridded table characteristic.

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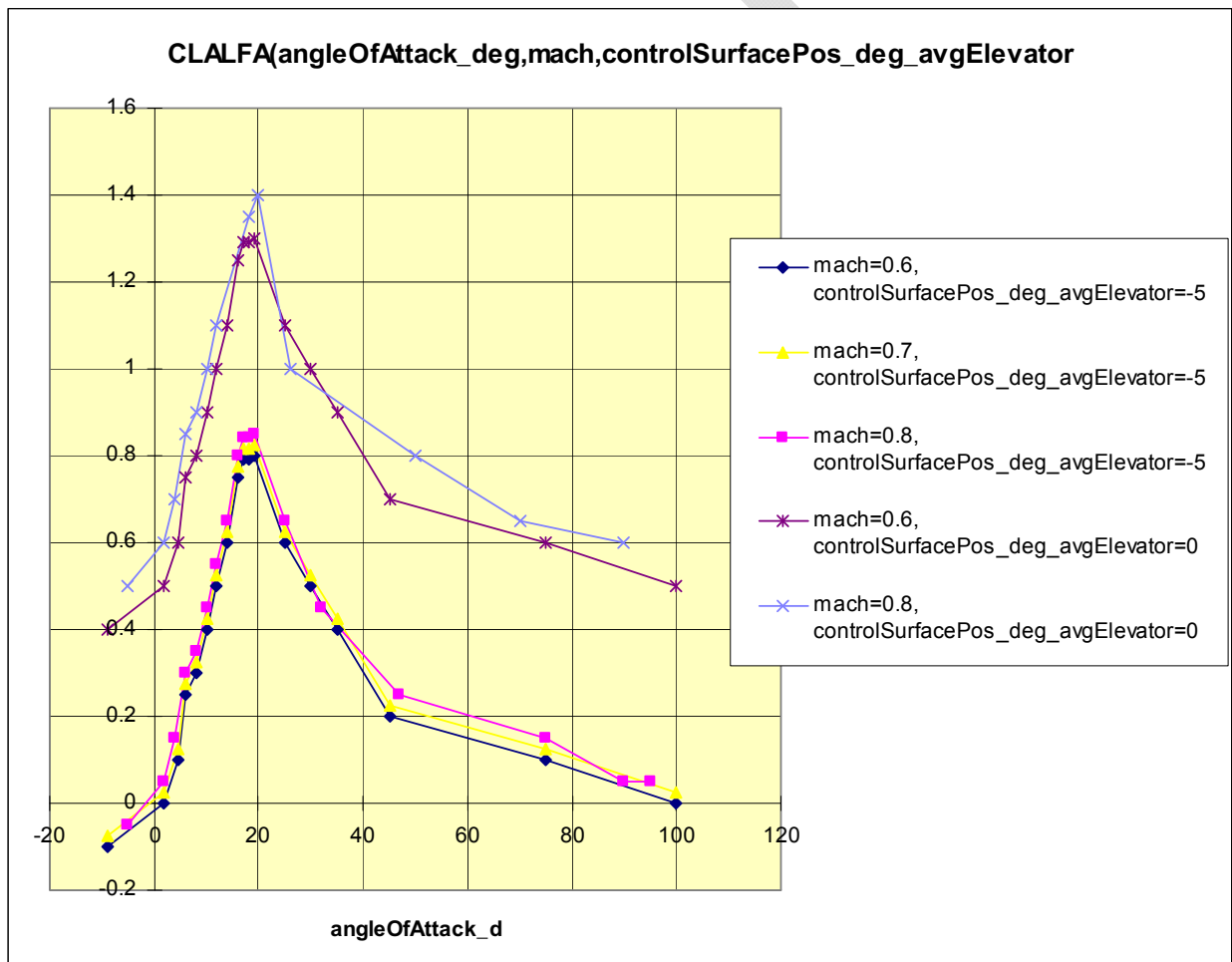

48

Figure 4 — An illustration of a three-dimensional function table

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

a) Where did the data come from? For example what wind tunnel test or computational model?

b) How is it defined? For example, is this at a specific altitude? What is the vehicle configuration ?

c) What are the engineering units of the output (the dependent variable) and the independent variables?

d) 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?

e) Who created the table? Not where the data came from, but what person decided that this was the correct data for this table?

f) How has it been modified and for what reason?

g) How accurate is the data estimated to be? Or, mathematically what is the confidence interval of the data?

h) By what method is the data intended to be interpolated? For example, linear interpolation or cubic spline interpolation?

i) By what method is the data intended to be extrapolated when the independent variable values are outside the specified range for the breakpoint data?

The DAVE-ML grammar has data elements that contain all of the above information. It also includes the ability to automate static checks of the function data to allow spot checking of the function after it has been exchanged. It is discussed in detail in the DAVE-ML reference document contained in Annex B. An introduction and overview of DAVE-ML's seven major elements is provided here.

7.5 DAVE-ML Major Elements (Annex B)

The major elements of DAVE-ML are listed below in the order required by the DAVE-ML DTD. The root element of a DAVE-ML model file, *DAVEfunc*, can have several sub-elements and attributes; most attributes and sub-elements are optional. The only sub-elements a DAVE-ML file must contain are the *fileHeader* and at least one *variableDef* .

Information (breakpoints, data points, provenance, etc.) that is used by more than one major element should be defined once and then referenced in any subsequent use.

The sub-elements must appear in the following order (as required by the DTD):

fileHeader — states the source and purpose of the file. It must include the author's contact details and the file creation date, and may include a description, reference information, and modification history.

variableDef — each *variableDef* defines one of the constants or signals (variables) used in the DAVE-ML model, whether input, output or internal. The definition includes all the attributes and sub-elements required to fully characterize the variable of interest, including a MathML definition if the variable's value is equation-based. Standard variables as defined in Section 6 and Annex A are encouraged here.

breakpointDef — defines breakpoint sets to be used in the model. The breakpoints are the coordinate values along one axis of a gridded linear function value table. One *breakpointDef* may be reused by several functions.

griddedTableDef — defines an orthogonally-gridded multi-dimensional table of the values of a

function at the intersection of a set of specified independent inputs (breakpoints). The coordinates along each dimension are defined in separate *breakpointDef* elements.

ungriddedTableDef — defines a table of non-orthogonal values of a function, each with the values of their independent coordinates.

function — defines a function by connecting independent variables, breakpoints and data tables to their output value.

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

Annex B contains the latest version of the DAVE-ML reference document, including a detailed description and examples of the data element definitions of the DAVE-ML standard. Section 8 of Annex B provides detailed XML element references and descriptions.

7.6   Simple DAVE-ML Examples

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 relationship example, giving pitching moment coefficient as a function of angle of attack, is shown in Table 5 and Figure 5.

Table 5 — A simple one-dimensional function table

angleOfAttack deg	0	18	19	20	22	23	25	27	90
cm(angleOfAttack deg)	0.1	-0.1	-0.09	-0.08	-0.05	-0.05	-0.07	-0.15	-0.6
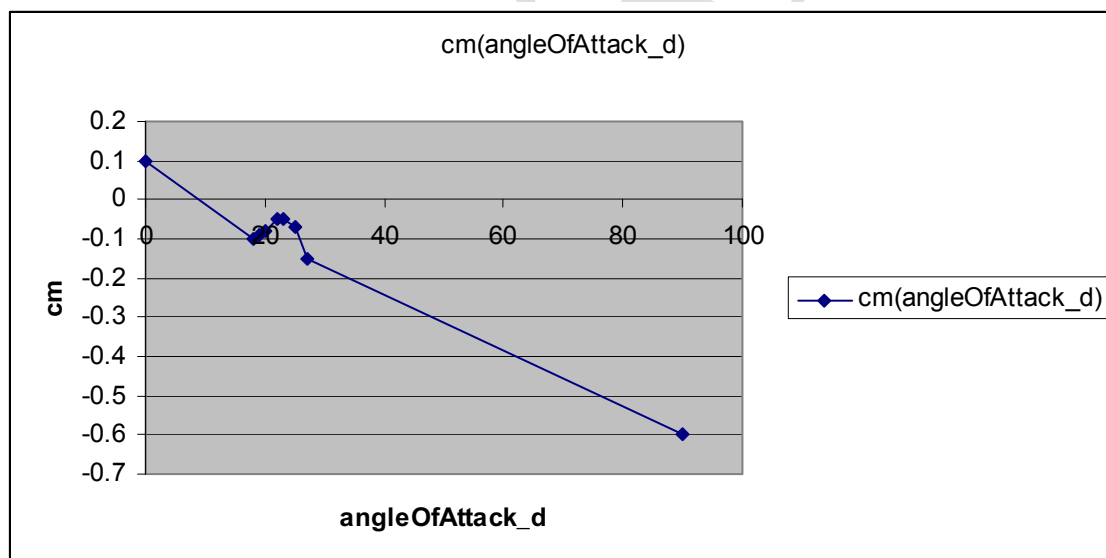
Figure 5 — A simple one-dimensional gridded function

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

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE DAVEfunc PUBLIC "-//AIAA//DTD for Flight Dynamic Models - Functions
2.0//EN" "DAVEfunc.dtd">
<DAVEfunc xmlns="http://daveml.org/2010/DAVEML">


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

50

```
<!--                    =========================                 -->
<fileHeader>

  <!-- This is an example of the file header components of the
        derivative of Cm as a function of angle of attack. It must
        define all documents that are later referenced by any function.

        Note that there is not much information in this header,
        because it is meant 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="JFTI" email="bhildreth@jfti.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" 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_bp1">

   <!--
```

```
      Note that the bpID can be any valid XML string to uniquely identify
      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_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 derivative 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_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 input and output variables
       with table definitions. This allows a level of abstraction such
       that the table, with its breakpoint definitions, can be reused
```

52

```
  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"/>
 <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</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</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</varID>
   <signalValue>10.</signalValue>
```

```xml
   </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</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</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</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</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 excessively long for a function with only 9 data points, most of its content involves self-documentation and checking. Therefore, as well as the function's data it includes the data's units, coordinate systems, uncertainty descriptions and provenance. It also includes many instructional comments, and verification data for multiple simulation conditions. 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 nominal data can be represented as shown. It is also possible to completely remove all whitespace between elements for more compactness, but this greatly affects readability by humans.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE DAVEfunc PUBLIC "-//AIAA//DTD for Flight Dynamic Models - Functions
2.0//EN" "DAVEfunc.dtd">
<DAVEfunc xmlns="http://daveml.org/2010/DAVEML">
    <fileHeader>
        <author name="Bruce Hildreth" org="SAIC"/>
        <fileCreationDate date="2006-03-18"/>
    </fileHeader>
    <variableDef name="Angle of attack" varID="angleOfAttack"
units="deg"/>
    <variableDef name="CMalpha" varID="CmAlfa" units=""/>
    <breakpointDef bpID="angleOfAttack_bp1">
        <bpVals> 0, 18, 19, 20, 22, 23, 25, 27, 90 </bpVals>
    </breakpointDef>
    <griddedTableDef gtID="CmAlfa_Table1">
        <breakpointRefs>
            <bpRef bpID="angleOfAttack_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"/>
        <dependentVarRef varID="CmAlfa"/>
        <functionDefn>
            <griddedTableRef gtID="CmAlfa_Table1"/>
        </functionDefn>
    </function>
</DAVEfunc>
```

The other principal means of model representation in DAVE-ML is through Math-ML elements, which are used to specify calculations.

For example:

  totalThrust_N= engine1Thrust_N + engine2Thrust_N + engine3Thrust_N

This equation, which is part of the model being exchanged, may be encoded in DAVE-ML and exchanged as data as shown below.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

```
<!DOCTYPE DAVEfunc PUBLIC "-//AIAA//DTD for Flight Dynamic Models - Functions
2.0//EN" "DAVEfunc.dtd">
<DAVEfunc xmlns="http://daveml.org/2010/DAVEML">
 <fileHeader>
  <author name="Dan Newman" org="Quantitative Aeronautics"/>
  <creationDate date="2009-11-10"/>
  <description>
   Simple MathML example : total thrust is the sum of three inputs.
  </description>
 </fileHeader>
<variableDef name="Engine #1 thrust" varID="engine1Thrust" units="N"/>
<variableDef name="Engine #2 thrust" varID="engine2Thrust" units="N"/>
<variableDef name="Engine #3 thrust" varID="engine3Thrust" units="N"/>
<variableDef name="Total thrust" varID="totalThrust" units="N" >
 <calculation>
     <math>
      <apply>
       <plus/>
          <ci>engine1Thrust</ci>
          <ci>engine2Thrust</ci>
          <ci>engine3Thrust</ci>
       </apply>
      </math>
  </calculation>
 </isOutput>
 </variableDef>
</DAVEfunc>
```

7.7   Summary

The DAVE-ML implementation of the standard enables nearly effortless transfer of simulation aerodynamics models between simulation facilities or architectures. Inclusion of Math-ML elements allows the formulation of algebraic equations, such as aerodynamic, propulsion, inertial, landing gear or control system models, to be included as data in the model. DAVE-ML is also suitable for use or transfer of tabular functions and algebraic equations for any type of data, not just simulation models.

While the above paragraphs provide an overview of the concepts implemented in DAVE-ML, Annex B is the normative authority for this standard. It includes much more detail and examples on how to easily build a DAVE-ML compliant simulation. Annex C provides a reference to the DAVE-ML web site, which includes tools that facilitate use of DAVE-ML based models in many applications.

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://www.daveml.org 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 Committee's 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 coordinate systems provide a common language to facilitate effective communication. The standard is also valuable for documenting a model, since the names and coordinate system definitions are clearly documented for the user.

A model can be DAVE-ML compliant without using any standard names or coordinate 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. `http://www.daveml.org` contains information on how to be part of this effort and/or submit change or improvement recommendations.

10   Standard Variable Names (Normative)

(Now a separate .doc for ease of editing)

Standard Variable Names (Normative)

A.1   General

The standard variable naming convention is described in detail in Section 6. The table in this annex contains a set of standard simulation variables that are independent of the particular vehicle type being simulated. Use of these standard variables provides a "standard language" which will facilitate the communication of the information required to exchange simulation models. These variables are tailored towards aircraft simulation and to a lesser extent, spacecraft. Visit `http://DaveML.org` 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 `eulerAngle_Roll` as an example:

1) The symbol for that variable, Φ

2) The short name, PHI

3) One or more full names using the standard units conventions — generally, one full name with American convention units and one with SI units. Refer to section 6 for a list of the standard units and their abbreviations.

NOTE: While the variable naming convention described in Section 6 encourages the use of the <variable domain> component, this Annex does not include variable domains as part of the normative standard names. This is because the variable domain is normally dependent upon the simulation architecture, and as such is immaterial to the exchange of a simulation model unless the exchange is between facilities with similar architectures.

NOTE: Any suitable units may be used. In the example for `eulerAngle_Roll` both the `_deg` for degrees and the `_rad` 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 coordinate system. Refer to section 5 for a description of the standard coordinate systems.

5) The POSITIVE sign convention of the variable — RWD indicates that positive `eulerAngle_Roll` is right wing down. (See section A.2.1 for a list of sign convention acronyms)

6) Minimum value, normally only specified for angles

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

Note:

This example also illustrates the pitch and yaw Euler angles.

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

`eiVelocity_ft_s_Y` for the velocity along the Y axis of the ei coordinate system - Earth centered Inertial (also known as geocentric inertial) coordinate system

`geVelocity_ft_s_Y` for the velocity along the Y axis of the geocentric Earth (ge) coordinate system. Also referred to as the Earth Centered Earth Fixed (ECEF coordinate system).

`voVelocity_ft_s_Y` for the vo coordinate system (Vehicle carried, Orbit defined coordinate system)

Roll, pitch and yaw may also conveniently be expressed as an array. Again, `eulerAngle_rad[3]`[1] would be the standard array using radians as the units and is fully compliant with the standard.

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

ε	EUL[3]	`eulerAngle_deg[3]`
		
		 `eulerAngle_rad[3]`
			Array of the roll, pitch, and yaw Euler angles comprised of the elements defined below. LL (locally level) coordinate system.			

Φ	PHI	`eulerAngle_deg_Roll`
		
		 `eulerAngle_rad_Roll`
			Roll Euler Angle, LL coordinate system.	RWD	-180,-π	180, π

θ	THET	`eulerAngle_deg_Pitch`
		
		 `eulerAngle_rad_Pitch`
			Pitch Euler Angle, LL coordinate system	ANU	-90, -π/2	90, π/2

Ψ	PSI	`eulerAngle_deg_Yaw`
		
		 `EulerAngle_rad_Yaw`
			Yaw Euler Angle, LL coordinate system	ANR	-180,-π	180, π

The variable name table below does not specify which variables are states, state derivatives, inputs, disturbances, simulation controls or initial conditions. These specifications may be added to any appropriate variable. Refer to Section 4.2 in the body of this standard for use of dynamic equation prefixes and section 6.4.8 for initial condition specification.

---

[1] A number or pair of numbers between square brackets represents the number of elements in an array or matrix respectively.

A.2.1   Annex A Acronyms

Positive Sign Convention Acrpnyms

The following acronyms may appear as values for the "positive sign convention" of variables defined in the variable name tables.

+X	in the positive direction of the presentation coordinate system's X-axis
+Y	in the positive direction of the presentation coordinate system's Y-axis.
+Z	in the positive direction of the presentation coordinate system's Z-axis
ABV	positive above
AFT	positive aft
AH	positive above horizon
ANR	positive aircraft nose right
ANU	positive aircraft nose up
BH	positive below horizon
BLO	positive below
CCFN	positive counterclockwise from north
COMP	positive compressed
CWFN	positive clockwise from north
DEC	decrease
DN or Down	positive down
E or East	positive East
FWD	positive forward
INC	positive increase
LED	positive leading edge down
LT or Left	positive left
MAC	percent mean aerodynamic chord
N or North	positive North
NSC	no sign convention (variable is always positive)
OUT	positive outward
POS	always positive
RT or Right	positive right
RTCL	positive right of centerline
RWD	positive right wing down

TED	positive trailing edge down
TEL	positive trailing edge left
TER	positive trailing edge right
TEU	positive trailing edge up
UP	positive up
WOW	weight on wheels

Control Surface and Position Acronyms

LEF	leading edge flap
TEF	trailing edge flap

A.3   Standard Variable Name Tables

Table A.1 — Vehicle Positions and Angles

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Intial Value	Min Value	Max Value
ε	EUL	`eulerAngle_deg[3]` `eulerAngle_rad[3]`	Array of the roll, pitch, and yaw Euler angles defined below. LL (locally level) coordinate system.				
Φ	PHI	`eulerAngle_deg_Roll` `eulerAngle_rad_Roll`	Roll Euler Angle, LL coordinate system.	RWD		-180, -π	180, π
θ	THET	`eulerAngle_deg_Pitch` `eulerAngle_rad_Pitch`	Pitch Euler Angle, LL coordinate system	ANU		-90, -π/2	90, π/2
ψ	PSI	`eulerAngle_deg_Yaw` `eulerAngle_rad_Yaw`	Yaw Euler Angle, LL coordinate system	ANR		-180, -π	180, π
sin Φ	SPHI	`sinEulerAngle_Roll`	Sine Of Euler Roll Angle	RWD		-1.0	1.0
cos Φ	CPHI	`cosEulerAngle_Roll`	Cosine Of Euler Roll Angle	RWD		-1.0	1.0
sin θ	STHT	`sinEulerAngle_Pitch`	Sine Of Euler Pitch Angle	ANU		-1.0	1.0
cos θ	CTHT	`cosEulerAngle_Pitch`	Cosine Of Euler Pitch Angle	ANU		0.0	1.0
sin ψ	SPSI	`sinEulerAngle_Yaw`	Sine Of Euler Yaw Angle	ANR		-1.0	1.0
cos ψ	CPSI	`cosEulerAngle_Yaw`	Cosine Of Euler Yaw Angle	ANR		-1.0	1.0
$\mathbf{T}_{FE/B}$	T	`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) coordinate transformation element				
$T_{FE/B}[2,1]$	T21	`feToBodyT21`	SPHI*STHT*CPSI - CPHI*SPSI (FE To B) coordinate transformation element				

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Intial Value	Min Value	Max Value
$T_{FE/B}[3,1]$	T31	`feToBodyT31`	CPHI*STHT*CPSI + SPHI*SPSI (FE to B) coordinate transformation element				
$T_{FE/B}[1,2]$	T12	`feToBodyT12`	CTHT*SPSI (FE to B) coordinate transformation element				
$T_{FE/B}[2,2]$	T22	`feToBodyT22`	SPHI*STHT*SPSI + CPHI*CPSI (FE to B) coordinate transformation element				
$T_{FE/B}[3,2]$	T32	`feToBodyT32`	CPHI*STHT*SPSI - SPHI*CPSI (FE to B) coordinate transformation element				
$T_{FE/B}[1,3]$	T13	`feToBodyT13`	-STHT (FE to B) coordinate transformation element				
$T_{FE/B}[2,3]$	T23	`feToBodyT23`	SPHI*CTHT (FE to B) coordinate transformation element				
$T_{FE/B}[3,3]$	T33	`feToBodyT33`	CPHI*CTHT (FE to B) coordinate transformation element				
γ	GAMV	`flightPathAngle_rad` `flightPathAngle_deg`	Flight Path Angle Above Horizon	ANU		$-\pi/2$ -90	$\pi/2$ 90
χ	GAMH	`flightPathAzimuth_rad` `flightPathAzimuth_deg`	Flight Path Angle In Horizon Plane, from North	CWFN		$-\pi$ -180	π 180
h	ALT	`altitudeMsl_ft` `altitudeMsl_m`	Geometric altitude of vehicle altimeter above Mean Sea Level	UP			

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

λ	XLON	`geLongitude_rad`
		`geLongitude_deg`
		Note The coordinate system may be deleted if Ge coordinate system, resulting in
		`longitude_rad`
		`longitude_deg`
			Longitude of Vehicle Cm with respect to the ge (geocentric Earth) coordinate system.	WEST			

φ	XLAT	`geLatitude_rad`
		`geLatitude_deg`
		Note The coordinate system may be deleted if Ge coordinate system, resulting in
		`latitude_rad`
		`latitude_deg`
			Geodetic Latitude of Vehicle Cm with respect to the ge (geocentric Earth) coordinate system.	NORTH			

	XLONIMU	`geLongitudeOfImu_rad`
		`geLongitudeOfImu_deg`
		
		Note The coordinate system may be deleted if Ge coordinate system, resulting in
		`longitudeOfImu_rad`
		`longitudeOfImu_deg`
			Longitude of Vehicle Imu with respect to the ge coordinate system. This variable does not include any Imu sensor errors.	West			

	XLATIMU	`gelatitudeOfImu_rad`
		`gelatitudeOfImu_deg`
		
		Note The coordinate system may be deleted if Ge coordinate system, resulting in
		`latitudeOfImu_rad`
		`latitudeOfImu_deg`
			Geodetic Latitude of Vehicle Imu with respect to the Gecoordinate system. This variable does not include any Imu sensor errors.	NORTH			

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

		`geSensedLongitudeOfImu_rad`
		`eSensedLongitudeOfImu_deg`
		
		Note The coordinate system may be deleted if Ge coordinate system, resulting in
		`sensedLongitudeOfImu_rad`
		`sensedLongitudeOfImu_deg`
			Longitude of Vehicle Imu with respect to the `Ge` coordinate system. This variable includes any Imu sensor errors.	West			

		`geSensedLatitudeOfImuWrtZzz_rad`
		`geSensedLatitudeOfImuWrtZzz_deg`
		
		Note The coordinate system may be deleted if Ge coordinate system, resulting in
		`sensedLatitudeOfImu_rad`
		`sensedLatitudeOfImu_deg`
			Geodetic Latitude of Vehicle Imu with respect to the `Ge` coordinate system. This variable includes any Imu sensor errors.	NORTH			

	HGT_RWY	`runwayHeightWrtMsl_ft` `runwayHeightWrtMsl_m`	Height Of Runway w.r.t. mean Sea Level	Above			

		General Definition
		`xxPositionOfYyyWrtZzz_ft[3]`
		`xxPositionOfYyyWrtZzz_m[3]`
		For Example:
		`xxPosition_ft[3]`
		is the same as
		`xxPositionOfCmWrtxx_ft[3]`
			General Definition for Linear and Angular Positions: Refer to section 6 of this standard for complete guidance on the naming of variables.
			Vector of positions of `Yyy` with respect to `Zzz` (a user defined reference point or coordinate system origin) in the xx coordinate system. The lengths of `xx`, `Yyy`, `Zzz` are not restricted to 2 and 3 characters respectively.
			The coordinate system xx must always be defined. If the `OfYyy` is not defined the definition defaults to the vehicle Cm for linear positions and the body coordinate system for angular positions. If the `WrtZzz` is not defined the reference point defaults to the origin of the `xx` coordinate system for linear position and the locally level (Ll) coordinate system for angular position.
			
			Comprised of the three components as defined below.				

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

		`xxPositionOfYyyWrtZzz_ft_X`
		`xxPositionOfYyyWrtZzz_m_X`
		
		or
		
		`xxPosition_ft_X`
			X position of `Yyy` with respect to `Zzz` (a user defined reference point) in the `xx` coordinate system.
			
			Defaults to the Cm and origin of the `xx` coordinate system.
				(`Yyy` –`Zzz`)			

		`xxPositionOfYyyWrtZzz_ft_Y`
		`xxPositionOfYyyWrtZzz_m_Y`
		
		or
		
		`xxPosition_ft_Y`
			Y position of `Yyy` with respect to `Zzz` (a user defined reference point) in the `xx` coordinate system.
			
			Defaults to th Cm and origin of the `xx` coordinate system.
				(`Yyy` –`Zzz`)			

		`xxPositionOfYyyWrtZzz_ft_Z`
		`xxPositionOfYyyWrtZzz_m_Z`
		or
		`xxPosition_ft_Z`
			Z position of `Yyy` with respect to `Zzz` (a user defined reference point) in the `xx` coordinate system.
			
			Defaults to the CG and origin of the `xx` coordinate system.
				(`Yyy` –`Zzz`)			

		`fePosition_ft [3]`	Vector of positions of the CM in the flat Earth coordinate system. Comprised of the three components as defined below.				
	XCG	`fePosition_ft_X`	X or North position of the CM in the flat Earth coordinate system				
	YCG	`fePosition_ft_Y`	Y or East position of the CM in the flat Earth coordinate system				
	ZCG	`fePosition_ft_Z`	Z or Down position of the CM in the flat Earth coordinate system	minus equals above the ground			

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Intial Value	Min Value	Max Value
		`xxPositionOfMrcWrtZzz_ft_X` `xxPositionOfMrcWrtZzz_m_X`	X position of the moment reference center (`Mrc`) with respect to `Zzz` in the `xx` coordinate system.				
		`xxPositionOfMrcWrtZzz_ft_Y` `xxPositionOfMrcWrtZzz_m_Y`	Y position of the moment reference center (`Mrc`) with respect to `Zzz` in the `xx` coordinate system.				
		`xxPositionOfMrcWrtZzz_ft_Z` `xxPositionOfMrcWrtZzz_m_Z`	Z position of the moment reference center (`Mrc`) with respect to `Zzz` in the `xx` coordinate system.				
	REF [3]	`bodyPositionOfMrc_ft [3]`	Vector of positions of the aerodynamic moment reference center in the body coordinate system. Since the origin of the body coordinate system is the Cm, this vector is the positions of the Mrc with respect to the Cm Comprised of the three components as defined below.				
	XREF	`bodyPositionOfMrc_ft_X`	X position of the Mrc in the body coordinate system	Mrc in front of the origin (Cm)			
	YREF	`bodyPositionOfMrc_ft_Y`	Y position of the Mrc in the body coordinate system	Mrc out the right wing with respect to the origin (Cm)			
	ZREF	`bodyPositionOfMrc_ft_Z`	Z position of the Mrc in the body coordinate system	Mrc below the origin (Cm)			
	PLT2CG[3]	`bodyPositionOfPilotEyeWrtCm_f[3]` `bodyPositionOfPilotEyeWrtCm_ft[3]`	Vector of positions of the pilot's eye with respect to the Cm in the body coordinate system. Comprised of the three components as defined below.				
	XPLT2CG	`bodyPositionOfPilotEyeWrtCm_ft_X` `bodyPositionOfPilotEyeWrtCm_ft_X`	X position of pilot eye point w.r.t. Cm, in the body coordinate system	Eye FWD of Cm			
	YPLT2CG	`bodyPositionOfPilotEyeWrtCm_ft_Y` `bodyPositionOfPilotEyeWrtCm_ft_Y`	Y position of pilot eye point w.r.t. Cm, in the body coordinate system	Eye Right of the Cm			

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Intial Value	Min Value	Max Value
	ZPLT2CG	`bodyPositionOfPilotEyeWrtCm_ft_Z` `bodyPositionOfPilotEyeWrtCm_ft_Z`	Z position of pilot eye point w.r.t. Cm, in the body coordinate system	Eye below Cm			
EXAMPLES							

		`runway22Position_ft[3]`
		indicates position of the Cm (default) with respect to the origin of the Runway22 coordinate system (default), in the runway22 coordinate system. A more complete and clear name would be:
		`runway22PositionOfCmWrtRunway22_ft[3]`
		
		`runway22PositionOfFwdLeftMainWheelWrtTd_ft[3]`
		indicates position of the forward left main wheel with respect to the touchdown point in the Runway 22 coordinate system
		NOTE All coordinate systems are user defined
			Vector of positions of the vehicle Cm relative to the Runway 22 (a user defined coordinate system) touchdown reference point. Comprised of the three components as defined below.				

	XCGTD	`runway22PositionOfCmWrtTd_ft_X` `runway22PositionOfCmWrtTd_m_X`	Cm X-position w.r.t. runway touchdown point in the specified (Runway22) coordinate system.	Cm Down the runway from the reference point			
	YCGTD	`runway22PositionOfCmWrtTd_ft_Y` `runway22PositionOfCmWrtTd_m_Y`	Cm Y-position w.r.t. runway touchdown point in the specified (Runway22) coordinate system.	Cm to the right of the reference point			
	ZCGTD	`runway22PositionOfCmWrtTd_ft_Z` `runway22PositionOfCmWrtTd_m_Z`	Cm Z-position w.r.t. runway touchdown point in the specified (Runway22) coordinate system. (This variable is normally negative.)	Cm below the TD point			
	RE	`smoothEarthRadius_ft` `smoothEarthRadius_m`	Geocentric radius of Earth (center to smooth surface), round Earth model or oblate spheroid at the nadir of the aircraft.				

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

	RALT	`heightOfCmWrtTerrain_ft`
		`heightOfCmWrtTerrain_m`
		`or`
		`heightWrtTerrain_ft`
		`heightWrtTerrain_m`
			Height of the aircraft Cm above the terrain	NSC			

	HTERRAIN	`heightOfTerrainWrtGround_ft` `heightOfTerrainWrtGround_m`	Height of the terrain at the nadir of the aircraft Cm. It is the terrain height above the smooth surface of the Earth, regardless of 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	Positive Sign Convention	Initial Value	Min Value	Max Value
$V_{T_{XX}}$	VTxx	`totalSpeedWrtXx_ft_s` `totalSpeedWrtXx_m_s`	Total Velocity with respect to and observed from `Xx` where `Xx` is the coordinate system as defined in the body of this standard.	NSC			
V_K	VG	`groundSpeed_ft_s` `groundSpeed_m_s`	Vehicle speed along the ground .	NSC			
M_N	XMACH	`mach`	Mach number of the aircraft	NSC			

$\dot{h}_{XX}$	ALTDxx	`xxAltitudeRate_ft_s`
		`xxAltitudeRate_m_s`
		default coordinate system is locally level and -Z axis
			Altitude time rate of change in `xx` coordinate system.	DN			

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

	XLOND	`xxLongitudeRate_rad_s`
		
		`xxLongitudeRate_deg_s`
		
		Default is Ge coordinate system
			Longitude Rate Of Change in `xx` coordinate system.	WEST			

	XLATD	`xxlatitudeRate_rad_s`
		
		`xxlatitudeRate_deg_s`
		
		Default is Ge coordinate system
			Geodetic Latitude Rate Of Change in `xx` coordinate system.	NORTH			

		General Definition-Translational Velocities
		`xxVelocityOfYyyWrtZzzObsFrWww_ft_s [3]`
		`xxVelocityOfYyyWrtZzzObsFrWww_m_s [3]`
			General Definition for Translational Motion: Refer to section 6 of this standard for complete guidance on the naming of variables.
			General expression for vector of velocities presented (or measured) in the xx coordinate system. Yyy indicates the reference point on the vehicle and the `OfYyy` may be omitted if it is the Cm. Zzz represents the coordinate system that the vehicle is moving with respect to and `WrtZzz` may be omitted if it is the xx coordinate system. `ObsFrWww` represents the coordinate system from which the vehicle motion is observed. The `ObsFrWww` may be omitted if it is the Zzz coordinate system (see section 6.3.3 for more detail)
			
			Therefore `eiVelocity_ft_s_X` is the velocity of the vehicle Cm along the X axis of the `ei` coordinate system, measured with respect to the ei coordinate system and observed from the ei coordinate system and is the default expression for `eiVelocityOfCmWrtEiObsFrEi_ft_s_X`.				

		General Definition-Angular Velocities
		`xxAnggularRateOfYyyWrtZzz_rad_s_[3]`
		`xxAnggularRateOfYyyWrtZzz_deg_s_[3]`
			General Definition for Rotational Motion: Refer to section 6 of this standard for complete guidance on the naming of variables.
			General expression for angular velocities presented in the `xx` coordinate system. `Yyy` indicates the reference coordinate system on the vehicle and the `OfYyy` may be omitted if it is the body coordinate system. `Zzz` represents the coordinate system that the vehicle is moving with respect to and from which the vehicle motion is observed. The `WrtZzz` may be omitted if it is the `locally level (Ll)` coordinate system. (see section 6.3.3 for more detail)
			
			Therefore `eiAngularRate_rad_s_Roll` is the angular rate of the body axis of the vehicle with respect to the locally level coordinate system and presented (or measured) in the Earth centered inertial (ei) coordinate system and is the default expression for `eiAngularRateOfBodyWrtLl_rad_s_Roll`				

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
Ω	OM	`bodyAngularRate_rad_s[3]` `bodyAngularRate_deg_s[3]`	Vector of body coordinate angular rates of the ownship body coordinate system with respect to the locally level (LI) coordinate system. Comprised of the three components as defined below. Angular motion is always with respect to the LI coordinate system unless otherwise specified.				
p	P	`bodyAngularRate_rad_s_Roll` `bodyAngularRate_deg_s_Roll`	Vehicle roll velocity, body coordinate system	RWD			
q	Q	`bodyAngularRate_rad_s_Pitch` `bodyAngularRate_deg_s_Pitch`	Vehicle pitch velocity, body coordinate system	ANU			
r	R	`bodyAngularRate_rad_s_Yaw` `bodyAngularRate_deg_s_Yaw`	Vehicle yaw velocity, body coordinate system	ANR			
Ω_B	OMB	`bodyAngularRateWrtEi_rad_s[3]` `bodyAngularRateWrtEi_deg_s[3]`	Vector of body coordinate angular rates with respect to the Earth centered inertial (Ei) coordinate system. Comprised of the three components as defined below.				
p_B	PB	`bodyAngularRateWrtEi_rad_s_Roll` `bodyAngularRateWrtEi_deg_s_Roll`	Vehicle roll velocity, body coordinate system	RWD			
q_B	QB	`bodyAngularRateWrtEi_rad_s_Pitch` `bodyAngularRateWrtEi_deg_s_Pitch`	Vehicle pitch velocity, body coordinate system	ANU			
r_B	RB	`bodyAngularRateWrtEi_rad_s_Yaw` `bodyAngularRateWrtEi_deg_s_Yaw`	Vehicle yaw velocity, body coordinate system	ANR			
p_{vf}	PS	`saAngularRateOfBodyWrtEi_rad_s_Roll` `saAngularRateOfBodyWrtEi_deg_s_Roll`	Body coordinate system Roll rate about the X axis in the `sa` coordinate system, also know as stability axis roll rate.	RWD			
r_{vf}	RS	`saAngularRateOfBodyWrtEi _rad_s_yaw` `saAngularRateOfBodyWrtEi_deg_s_yaw`	Body coordinate system Yaw rate about the Z axis in the `sa` coordinate system, also known as the Stability Axis yaw rate	ANR			

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
$\dot{\underline{\varepsilon}}$	EULD	`eulerAngleRate_deg_s[3]` `eulerAngleRate_rad_s[3]`	Array of the roll, pitch, and yaw Euler angle rates defined below. LL (locally level) coordinate system				
$\dot{\phi}$	PHID	`eulerAngleRate_rad_s_Roll`	Euler roll rate, LL coordinate system	RWD			
$\dot{\theta}$	THETD	`eulerAngleRate_rad_s_Pitch`	Euler pitch rate, LL coordinate system	ANU			
$\dot{\psi}$	PSID	`eulerAngleRate_rad_s_Yaw`	Euler yaw rate, LL coordinate system	ANR			

$\underline{V}$	VEL [3]	`bodyVelocityWrtAir_ft_s[3]`
		 `bodyVelocityWrtAir_m_s[3]`
		 can also be expressed as:
		 `bodyVelocityOfCmWrtAir_ft_s[3]`
			Vector of body coordinate velocities of the Cm with respect to the instantaneous wind comprised of the three components as defined below.
			
			 This is the conventional body coordinate system airspeed vector.				

u	U	`bodyVelocityWrtAir_ft_s_X` `bodyVelocityWrtAir_m_s_X`	X-velocity Body coordinate system.	FWD			
v	V	`bodyVelocityWrtAir_ft_s_Y` `bodyVelocityWrtAir_m_s_Y`	Y-velocity Body coordinate system	RT			
w	W	`bodyVelocityWrtAir_ft_s_Z` `bodyVelocityWrtAir_m_s_Z`	Z-velocity Body coordinate system	DN			

$\underline{V}_B$	VELB[3]	`bodyVelocityWrtEi_ft_s[3]`
		 `bodyVelocityWrtEi_m_s[3]`
		 can also be expressed as:
		 `bodyVelocityOfCmWrtEi_ft_s[3]`
			Vector of body coordinate inertial velocities of the Cm comprised of the three components as defined below.				

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
u_B	UB	`bodyVelocityWrtEi_ft_s_X` `bodyVelocityWrtEi_m_s_X`	X-velocity Body coordinate system.	FWD			
v_B	VB	`bodyVelocityWrtEi_ft_s_Y` `bodyVelocityWrtEi_m_s_Y`	Y-velocity Body coordinate system	RT			
w_B	WB	`bodyVelocityWrtEi_ft_s_Z` `bodyVelocityWrtEi_m_s_Z`	Z-velocity Body coordinate system	DN			

$\underline{V}_E$	VELE[3]	`bodyVelocityWrtGe_ft_s[3]`
		`bodyVelocityWrtGe_m_s[3]`
		can also be expressed as:
		`bodyVelocityOfCmWrtGe_ft_s[3]`
			Vector of body coordinate Cm velocities with respect to the geocentric Earth (Ge) coordinate system.
			
			Comprised of the three components as defined below.				

u_E	UE	`bodyVelocityWrtGe_ft_s_X` `bodyVelocityWrtGe_m_s_X`	X-velocity Body coordinate system.	FWD			
v_E	VE	`bodyVelocityWrtGe_ft_s_Y` `bodyVelocityWrtGe_m_s_Y`	Y-velocity Body coordinate system	RT			
w_E	WE	`bodyVelocityWrtGe_ft_s_Z` `bodyVelocityWrtGe_m_s_Z`	Z-velocity Body coordinate system	DN			
$\underline{V}_{FE}$	VELFE	`feVelocity_ft_s[3]` `feVelocity_m_s[3]`	Vector of Flat Earth (FE) coordinate translational velocities of the Cm comprised of the three components as defined below.				
V_N	VNFE	`feVelocity_ft_s_X` `feVelocity_m_s_X`	Northward Velocity Over Flat Earth (FE) coordinate system [flat, non-rotating Earth]	NORTH			

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

V_E	VEFE	`feVelocity_ft_s_Y`
		
		`feVelocity_m_s_Y`
			Eastward Velocity Over Flat Earth (FE) coordinate system [flat, non-rotating Earth]	EAST			

V_D	VDFE	`feVelocity_ft_s_Z`
		
		`feVelocity_m_s_Z`
			Downward Velocity Toward Earth Ctr,.(FE) coordinate system [flat, non-rotating Earth]	DN			

$\underline{V}_E$		`geVelocity_ft_s [3]`
		
		`geVelocity_m_s [3]`
			Vector of Geocentric Earth (Ge) coordinate translational velocities of the Cm comprised of the three components as defined below.				

V_{X_E}	VXGE	`geVelocity_ft_s_X`
		
		`geVelocity_m_s_X`
			X axis velocity over the Earth in the geocentric Earth (GE) coordinate system in ft/sec	+X			

V_{Y_E}	VYGE	`geVelocity_ft_s_Y`
		
		`geVelocity_m_s_Y`
			Y axis velocity over the Earth in the geocentric Earth (GE) coordinate system in ft/sec	+Y			

V_{Z_E}	VZGE	`geVelocity_ft_s_Z`
		
		`geVelocity_m_s_Z`
			Y axis velocity over the Earth in the geocentric Earth (GE) coordinate system in ft/sec	+Z			

OTHER EXAMPLES							
$V_{X_{ge}}$	VXGE	`geVelocity_km_s_X`	X axis velocity of the vehicle Cm is the geocentric (ge) coordinate system in kilometers/sec	+X			
		`runway22Velocity_ft_s_Z`	Z axis velocity of the Cm in the user defined "runway22" coordinate system in ft/s	DN			

Table A.3 — Vehicle linear and angular accelerations

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

		General Definition-Translational Accelerations
		`xxAccelOfYyyWrtZzzObsFrWww_ft_s2 [3]`
		`xxAccelOfYyyWrtZzzObsFrWww_m_s2 [3]`
			General Definition for Translational Motion: Refer to section 6 of this standard for complete guidance on the naming of variables.
			General expression for vector of accelerations presented (or measured) in the xx coordinate system. Yyy indicates the reference point on the vehicle and the `OfYyy` may be omitted if it is the Cm. Zzz represents the coordinate system that the vehicle is moving with respect to and `WrtZzz` may be omitted if it is the xx coordinate system. `ObsFrWww` represents the coordinate system from which the vehicle motion is observed. The `ObsFrWww` may be omitted if it is the Zzz coordinate system (see section 6.3.3 for more detail)
			Therefore `eiVelocity_ft_s2_X` is the acceleration of the vehicle Cm along the X axis of the `ei` coordinate system, measured with respect to the ei coordinate system and observed from the ei coordinate system and is the default expression for `eiVelocityOfCmWrtEiObsFrEi_ft_s2_X`.				

		General Definition-Angular Accelerations
		`xxAnggularAccelOfYyyWrtZzz_rad_s2_[3]`
		`xxAnggularAccelOfYyyWrtZzz_deg_s2_[3]`
			General Definition for Rotational Motion: Refer to section 6 of this standard for complete guidance on the naming of variables.
			General expression for angular accelerations presented in the `xx` coordinate system. `Yyy` indicates the reference coordinate system on the vehicle and the `OfYyy` may be omitted if it is the body coordinate system. `Zzz` represents the coordinate system that the vehicle is moving with respect to and from which the vehicle motion is observed. The `WrtZzz` may be omitted if it is the `locally level (Ll)` coordinate system. (see section 6.3.3 for more detail)
			Therefore `eiAngularAccel_rad_s2_Roll` is the angular acceleration of the body axis of the vehicle with respect to the locally level coordinate system and presented (or measured) in the Earth centered inertial (ei) coordinate system and is the default expression for `eiAngularAccelOfBodyWrtLl_rad_s2_Roll`				

$\underline{\dot{\omega}}$	OMBD	`bodyAngularAccel_rad_s2[3]` `bodyAngularAccel_deg_s2[3]`	Vector of body coordinate angular accelerations of the ownship body axis coordinate system with respect to the Locally level (Ll) coordinate system (Note: WrtLl is the default and is therefore omitted), comprised of the three body axis components as defined below.				
$\dot{p}$	PBD	`bodyAngularAccel_rad_s2_Roll` `bodyAngularAccel_deg_s2_Roll`	Aircraft Roll Acceleration, Body coordinate system	RWD			

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

$\dot{q}$	QBD	`bodyAngularAccel_rad_s2_Pitch`
		
		`bodyAngularAccel_deg_s2_Pitch`
			Aircraft Pitch Acceleration, Body coordinate system	ANU			

$\dot{r}$	RBD	`bodyAngularAccel_rad_s2_Yaw`
		
		`bodyAngularAccel_deg_s2_Yaw`
			Aircraft Yaw Acceleration, Body coordinate system	ANR			

$\underline{\dot{\omega}}_B$	OMBDE	`bodyAngularAccelWrtEi_rad_s2[3]`
		
		`bodyAngularAccelWrtEi_deg_s2[3]`
			Vector of body coordinate angular accelerations with respect to the Earth-centered inertial coordinate system, comprised of the three body axis components as defined below.				

$\dot{p}_B$	PBDE	`bodyAngularAccelWrtEi_rad_s2_Roll`
		
		`bodyAngularAccelWrtEi_deg_s2_Roll`
			Aircraft Roll Acceleration, Body coordinate system	RWD			

$\dot{q}_B$	QBDE	`bodyAngularAccelWrtEi_rad_s2_Pitch`
		
		`bodyAngularAccelErtEi_deg_s2_Pitch`
			Aircraft Pitch Acceleration, Body coordinate system	ANU			

$\dot{r}_B$	RBDE	`bodyAngularAccelWrtEi_rad_s2_Yaw`
		
		`bodyAngularAccelWrtEi_deg_s2_Yaw`
			Aircraft Yaw Acceleration, Body coordinate system	ANR			

$\underline{\dot{V}}_B$		`bodyAccelWrtEi_ft_s2[3]`
		
		`bodyAccelWrtEi_m_s2[3]`
			Vector of accelerations of the Cm of the aircraft with respect to and observed from the `Ei` coordinate system (Earth-centered inertial) in the body coordinate system. This variable includes the gravitation vector. Comprised of the three components as defined below.				

$\dot{u}_B$	UBD	`bodyAccelWrtEi_ft_s2_X`
		
		`bodyAccelWrtEi_m_s2_X`
			Longitudinal acceleration (along the X-body coordinate)	FWD			

$\dot{v}_B$	VBD	`bodyAccelWrtEi_ft_s2_Y`
		
		`bodyAccelWrtEi_m_s2_Y`
			Right Sideward Acceleration, Body coordinate	RT			

$\dot{w}_B$	WBD	`bodyAccelWrtEi_ft_s2_Z`
		
		`bodyAccelWrtEi_m_s2_Z`
			Downward Acceleration, Body coordinate	DNDN			

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

$\underline{\dot{V}}$		`bodyAccelWrtAir_ft_s2[3]`
		
		`bodyAccelWrtAir_m_s2[3]`
			Vector of body coordinate accelerations of the Cm with respect to the mean air mass and comprised of the three components as defined below. This variable includes the gravitation vector. Comprised of the three components as defined below.				

$\dot{u}$		`bodyAccelWrtAir_ft_s2_X`
		
		`bodyAccelWrtAir_m_s2_X`
			Longitudinal acceleration (along the X-body coordinate)	FWD			

$\dot{v}$		`bodyAccelWrtAir_ft_s2_Y`
		
		`bodyAccelWrtAir_m_s2_Y`
			Right Sideward Acceleration, along the Y Body coordinate	RT			

$\dot{w}$		`bodyAccelWrtAir_ft_s2_Z`
		
		`bodyAccelWrtAir_m_s2_Z`
			Downward Acceleration, along the Z Body coordinate	DNDN			

$\underline{\dot{V}}_E$		`bodyAccelWrtGe_ft_s2[3]`
		
		`bodyAccelWrtGe_m_s2[3]`
			Vector of body coordinate Cm velocities with respect to the geocentric Earth (Ge) coordinate system.
			
			Comprised of the three components as defined below.comprised of the three components as defined below. This variable includes the gravitation vector. Comprised of the three components as defined below.				

$\dot{u}_E$		`bodyAccelWrtGe_ft_s2_X`
		
		`bodyAccelWrtGe_m_s2_X`
			Longitudinal acceleration (along the X-body coordinate)	FWD			

$\dot{v}_E$		`bodyAccelWrtGe_ft_s2_Y`
		
		`bodyAccelWrtGe_m_s2_Y`
			Right Sideward Acceleration, along the Y Body coordinate	RT			

$\dot{w}_E$		`bodyAccelWrtGe_ft_s2_Z`
		
		`bodyAccelWrtGe_m_s2_Z`
			Downward Acceleration, along the Z Body coordinate	DN			

		`totalAccelWrtEi_ft_s2`	Magnitude of the inertial acceleration vector	NSC			

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

	VTDxx	`totalVelocityRateWrtXx_ft_s2`
		
		`totalVelocityRateWrtXx_m_s2`
			Rate of change of speed with respect to and observed from Xx, where `xx` is the coordinate system as defined in the body of this standard.
			
			Total velocity (speed) rate of change is not the same as total acceleration (magnitude of the acceleration vector). For example, a velocity vector that only undergoes a direction change shows zero change in speed (magnitude) but still shows a positive total acceleration to its direction change.
				INC			

		`xxAccel_ft_s2 [3]`
		
		`xxAccel_m_s2 [3]`
			General form for the vector of aircraft Cm translational acceleration with respect to and observed from the specified (`xx`) coordinate system comprised of the three components as defined below.				

		`xxAccel_ft_s2_X`
		
		`xxAccel_m_s2_X`
			X-axis acceleration in xx coordinate system	+X			

		`xxAccel_ft_s2_Y`
		
		`xxAccel_m_s2_Y`
			Y-axis acceleration in xx coordinate system	+Y			

		`xxAccel_ft_s2_Z`
		
		`xxAccel_m_s2_Z`
			Z-axis acceleration in xx coordinate system	+Z			

		`bodyAccelOfImuWrtEi_ft_s2[3]`
		
		`bodyAccelOfImuWrtEi_m_s2[3]`
			Vector of true inertial accelerations at the inertial measurement unit (Imu) including the effects of the gravitation vector. This variable assumes a perfect Imu.
			
			 Comprised of the three body coordinate system components as defined below.				

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

	AX	`bodyAccelOfImuWrtEi_ft_s2_X`
		
		`bodyAccelOfImuWrtEi_m_s2_X`
			X Acceleration Of aircraft Cm (body coordinate)
			
			Includes the gravitation vector.
				FWD			

	AY	`bodyAccelOfImuWrtEi_ft_s2_Y`
		
		`bodyAccelOfImuWrtEi_m_s2_Y`
			Y Acceleration Of aircraft Cm (body coordinate)
			
			Includes the gravitation vector.
				RT			

	AZ	`bodyAccelOfImuWrtEi_ft_s2_Z`
		
		`bodyAccelOfImuWrtEi_m_s2_Z`
			Z Acceleration Of aircraft Cm (body coordinate)
			
			Includes the gravitation vector.
				DN			

		`bodySensedAccelOfImuWrtEi_ft_s2[3]`
		
		`bodySensedAccelOfImuWrtEi_m_s2[3]`
			Vector of sensed inertial accelerations at the inertial measurement unit (Imu) including the effects of the gravitation vector. This variable includes any senor scale, bias and noise..
			
			Comprised of the three body coordinate system components as defined below.				

		`bodySensedAccelOfImuWrtEi_ft_s2_X`
		
		`bodySensedAccelOfImuWrtEi_m_s2_X`
			X Acceleration Of aircraft Cm (body coordinate)
			
			Includes the gravitation vector.
				FWD			

		`bodySensedAccelOfImuWrtEi_ft_s2_Y`
		
		`bodySensedAccelOfImuWrtEi_m_s2_Y`
			Y Acceleration Of aircraft Cm (body coordinate)
			
			Includes the gravitation vector.
				RT			

		`bodySensedAccelOfImuWrtEi_ft_s2_Z`
		
		`bodySensedAccelOfImuWrtEi_m_s2_Z`
			Z Acceleration Of aircraft Cm (body coordinate)
			
			Includes the gravitation vector.
				DN			

		`bodyAccelOfPilotWrtEi_ft_s2[3]`
		
		`bodyAccelOfPilotWrtEi_m_s2 [3]`
			Vector of inertial accelerations at the pilot reference point, in the body coordinate system, comprised of the three components as defined below.				

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

	AXP	`bodyAccelOfPilotWrtEi_ft_s2_X`
		
		`bodyAccelOfPilotWrtEi_m_s2_X`
			X Acceleration Of Pilot reference point (body coordinate)	FWD			

	AYP	`bodyAccelOfPilotWrtEift_s2_Y`
		
		`bodyAccelOfPilotWrtEi_m_s2_Y`
			Y Acceleration Of Pilot reference point(body coordinate)	RT			

	AZP	`bodyAccelOfPilotWrtEi_ft_s2_Z`
		
		`bodyAccelOfPilotWrtEi_m_s2_Z`
			Z Acceleration Of Pilot reference point(body coordinate)	DN			

	GVBODY [3]	`bodyLocalGravitation_ft_s2 [3]`
		
		`bodyLocalGravitation_m_s2 [3]`
			Local gravitation vector in the body coordinate system.				

	GVBODYX	`bodyLocalGravitation_ft_s2_X` `bodyLocalGravitation_ft_s2_X`	X axis component	FWD			
	GVBODYY	`bodyLocalGravitation_ft_s2_Y` `bodyLocalGravitation_ft_s2_Y`	Y axis component	RT			
	GVBODYZ	`bodyLocalGravitation_ft_s2_Z` `bodyLocalGravitation_ft_s2_Z`	Z axis component	DN			

	GVGE [3]	`geLocalGravitation_ft_s2 [3]`
		
		`geLocalGravitation_m_s2 [3]`
			Local gravitation vector in the ge (geocentric Earth) coordinate system.				

	GVGEX	`geLocalGravitation_ft_s2_X` `geLocalGravitation_ft_s2_X`	X axis component	FWD			
	GVGEY	`geLocalGravitation_ft_s2_Y` `geLocalGravitation_ft_s2_Y`	Y axis component	RT			

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
	GVGEZ	`geLocalGravitation_ft_s2_Z` `geLocalGravitation_ft_s2_Z`	Z axis component	DN			

	GVEI[3]	`eiLocalGravitation_ft_s2 [3]`
		
		`eiLocalGravitation_m_s2 [3]`
			Local gravitation vector in the ei (Earth centered inertial) coordinate system.				

	GVEIX	`eiLocalGravitation_ft_s2_X` `eiLocalGravitation_ft_s2_X`	X axis component	FWD			
	GVEIY	`eiLocalGravitation_ft_s2_Y` `eiLocalGravitation_ft_s2_Y`	Y axis component	RT			
	GVEIZ	`eiLocalGravitation_ft_s2_Z` `eiLocalGravitation_ft_s2_Z`	Z axis component	DN			

	G	`localGravity_ft_s2`
		
		`localGravity_m_s2`
			Acceleration Due To Gravity (at the vehicle altitude), in the LI coordinate system.	DN			

Table A.4 — Vehicle air data

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

α	ALFA	`angleOfAttack_deg`
		
		`angleOfAttack_rad`
			Angle Of Attack, Body coordinate	ANU		$-\pi$,- 180	$+\pi$, +180

β	BETA	`angleOfSideslip_deg`
		
		`angleOfSideslip_rad`
			Sideslip Angle, Body coordinate	ANL		$-\pi$,- 180	$+\pi$, +180

$\dot{\alpha}$	ALFD	`angleOfAttackRate_rad_s`	Angle Of Attack Rate, Body coordinate	ANU			

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
$\dot{\beta}$	BETD	`angleOfSideslipRate_rad_s`	Sideslip Angle Rate	ANL			
$\sin\alpha$	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_nmi_h`	Calibrated Air Speed, knots	FWD			
V_{EQ}	VEQ	`equivalentAirspeed_nmi_h`	Equivalent Air Speed	FWD			
V_{IND}	VCAL	`indicatedAirspeed_nmi_h`	Calibrated Air Speed,	FWD			

V	VRW	`trueAirspeed_ft_s`
		`trueAirspeed_m_s`
		`trueAirspeed_nmi_h`
			Vehicle Velocity relative to the local wind (true airspeed)	FWD			

$\overline{q}$	QBAR	`dynamicPressure_lbf_ft2` `dynamicPressure_N_m2`	Dynamic Pressure	NSC			
$\overline{q}_c$	QBARC	`impactPressure_lbf_ft2` `impactPressure_N_m2`	Impact Pressure	NSC			
ρ	RHO	`airDensity_lbm_ft3` `airDensity_kg_m3`	Air Density, At Altitude of the aircraft	NSC			
	DENALT	`densityAltitude_ft` `densityAltitude_m`	Density altitude				

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

a	SOUND	`speedOfSound_ft_s`
		
		 `speedOfSound_m_s`
			Velocity Of Sound At Altitude of the aircraft	NSC			

T_{TOTR}	TR	`totalTempRatio` `totalTempRatio`	Total Temperature Ratio	NSC			
P_{TOTR}	PR	`totalPressureRatio` `totalPressureRatio`	Total Pressure Ratio	NSC			

T_{AMB}	TAMB	`ambientTemperature_dgC`
		
		 `ambientTemperature_dgK`
			Ambient Temperature at altitude	NSC			

P_{AMB}	PAMB	`ambientPressure_lbf_ft2`
		
		 `ambientPressure_N_m2`
			Ambient Pressure at altitude	NSC			

P_{AMBR}	PAMBR	`ambientPressureRatio`	Ratio Of ambient pressure at altitude to sea level ambient pressure	NSC			
T_{AMBR}	TAMBR	`ambientTemperatureRatio`	Ratio Of ambient temperature at altitude to sea level ambient temp.	NSC			

t_T	TTOT	`totalTemp_dgC`
		
		 `totalTemp_dgK`
			Total Temperature at altitude	NSC			

p_T	PTOT	`totalPressure_lbf_ft2`
		
		 `totalPressure_N_m2`
			Total Pressure at altitude	NSC			

	TAMB	`ambientTemperatureAtAlt_dgK`
		
		 `ambientTemperatureAtAlt_dgR`
		
		 `ambientTemperatureAtAlt_dgC`
			Ambient temperature, at the altitude of the Cm				

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

	TTOT	`totalTemperatureAtAlt_dgK`
		
		`totalTemperatureAtAlt_dgR`
		
		`totalTemperatureAtAlt_dgC`
			Total temperature at the altitude of the Cm				

	ALT_SET	`instrumentAltimeterSetting_inchMercury`	Cockpit Altimeter setting (Kohlsman window)	29.92 is standard day			

	P_ALT	`pressureAltitude_ft`
		
		`pressureAltitude_m`
			Pressure altitude at the Cm				

	RHO_SL	`seaLevelAirDensity_lbm_ft3`	Air density at sea level				

	TAMB_SL	`seaLevelAmbientTemp_dgK`
		
		`seaLevelAmbientTemp_dgC`
			Ambient temperature at mean sea level				

	PAMB_SL	`seaLevelAmbientPressure_lbf_ft2`
		
		`seaLevelAmbientPressure_N_m2`
			Ambient pressure at sea level				

Table A.5 — Atmospheric disturbances and turbulence

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

	WIND_SPEED	`meanAirMassSpeed_ft_s`
		
		`meanAirMassSpeed_m_s`
			Total velocity of mean wind				

	WIND_DIRECTION	`meanAirMassDirection_deg`	Mean wind direction	Wind blowing FROM North			

V_w	VTWS	`totalWindSpeed_ft_s`
		
		`totalWindSpeed_m_s`
			Total Wind Speed (mean wind plus turbulence)	NSC			

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

$\underline{V}_{WT_{ll}}$		`llTurbulenceVelocityWrtGe ft s[3]`
		 `llTurbulenceVelocityWrtGe_m_s[3]`
			Vector of locally level coordinate wind turbulence velocities with respect to the geocentric earth (Ge) fixed coordinate system.
			
			 Comprised of the three components as defined below.
			
			 Note: This vector is designed to be transformed from the GRAM 07 small wind pertubation velocities. GRAM 07 outputs the wind pertubation vector:
			
			 `usmall` (+ East)
			
			 `vsmall` (+ North)
			
			 `wsmall` (+ UP)
			
			 Therefore:
			
			 `llturbulenceVelocityWrtGe_m_s_X = vsmall`
			
			 `llturbulenceVelocityWrtGe_m_s_Y = usmall`
			
			 `llturbulenceVelocityWrtGe_m_s_Z = -wsmall`				

$N_{W_{T_{ll}}}$	NSMALL	`llTurbulenceVelocityWrtGe ft s X` `llTurbulenceVelocityWrtGe_m_s_X`	X-velocity Turb. Component	to the North			
$E_{W_{T_{ll}}}$	ESMALL	`llTurbulenceVelocityWrtGe ft s Y` `llTurbulenceVelocityWrtGe_m_s_Y`	Y-velocity Turb. Component	to the East			
$D_{W_{T_{ll}}}$	DSMALL	`llTurbulenceVelocityWrtGe ft s Z` `llTurbulenceVelocityWrtGe_m_s_Z`	Z-velocity Turb. Component	to downward			
$\underline{V}_{Turb}$	VELBWT	`bodyTurbulenceVelocityWrtGe ft s[3]` `bodyTurbulenceVelocityWrtGe_m_s[3]`	Vector of body coordinate translational turbulence velocities comprised of the three components as defined below.				
U_{Turb}	UBTURB	`bodyTurbulenceVelocityWrtGe_ft_s X` `bodyTurbulenceVelocityWrtGe_m_s_X`	X-velocity Turb. Component, Body coordinate	FWD (tailwind)			
V_{Turb}	VBTURB	`bodyTurbulenceVelocityWrtGe ft s Y` `bodyTurbulenceVelocityWrtGe_m_s_Y`	Y-velocity Turb. Component, Body coordinate	RT			

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
W_{Turb}	WBTURB	`bodyTurbulenceVelocityWrtGe_ft_s_Z` `bodyTurbulenceVelocityWrtGe_m_s_Z`	Z-velocity Turb. Component, Body coordinate	DN			

$\underline{V}_{W_{M_{ll}}}$	VELMW	`llVelocityOfMeanAirMassWrtGe_ft_s[3]`
		`llVelocityOfMeanAirMassWrtGe_m_s[3]`
			Vector of locally level coordinate mean wind translational velocities comprised of the three components as defined below.
			
			Note: This vector is designed to be transformed from the GRAM 07 wind model velocities. GRAM 07 outputs the mean wind velocity vector:
			
			   `umean` (+ East)
			
			   `vmean` (+ North)
			
			   `wmean` (+ UP)
			
			Therefore:
			
			   `llVelocityOfMeanAirMassWrtGe_m_s_X = vmean`
			
			   `llVelocityOfMeanAirMassWrtGe_m_s_Y = umean`
			
			   `llVelocityOfMeanAirMassWrtGe_m_s_Z = -wmean`				

$N_{W_{M_{ll}}}$	NMEAN	`llVelocityOfMeanAirMassWrtGe_ft_s_X` `llVelocityOfMeanAirMassWrtGe_m_s_X`	X-velocity Component, locally level coordinate system	to the North			
$E_{W_{M_{ll}}}$	EMEAN	`llVelocityOfMeanAirMassWrtGe_ft_s_Y` `llVelocityOfMeanAirMassWrtGe_m_s_Y`	Y-velocity Component, locally level coordinate system	to the East			
$D_{W_{M_{ll}}}$	DMEAN	`llVelocityOfMeanAirMassWrtGe_ft_s_Z` `llVelocityOfMeanAirMassWrtGe_m_s_Z`	Z-velocity Component, locally level coordinate system	to downward			

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
$\underline{V}_{W_{M_B}}$	VELMWB	`bodyVelocityOfMeanAirMassWrtGe_ft_s[3]` `bodyVelocityOfMeanAirMassWrtGe_m_s[3]`	Vector of body coordinate system mean wind translational wind velocities comprised of the three components as defined below. Note: This vector is designed to be the GRAM 07 wind model mean velocities transformed into the body coordinate system.				
$u_{W_{M_B}}$	UMEANB	`bodyVelocityOfMeanAirMassWrtGe_ft_s_X` `bodyVelocityOfMeanAirMassWrtGe_m_s_X`	X-velocity Component, body coordinate system	FWD (tailwind)			
$v_{W_{M_B}}$	VMEANB	`bodyVelocityOfMeanAirMassWrtGe_ft_s_Y` `bodyVelocityOfMeanAirMassWrtGe_m_s_Y`	Y-velocity Component, body coordinate system	To the RT			
$w_{W_{M_B}}$	WMEANB	`bodyVelocityOfMeanAirMassWrtGe_ft_s_Z` `bodyVelocityOfMeanAirMassWrtGe_m_s_Z`	Z-velocity Component, body coordinate system	DN			
$\underline{V}_{Tw}$	VTW	`bodyWindVelocityWrtGe_ft_s[3]` `bodyWindVelocityWrtGe_m_s[3]`	Vector of the body coordinate system of net wind velocities impinging on the aircraft at the Cm. This would include mean air mass motion and any disturbances such as turbulence and shear. Comprised of the three components as defined below. Note: Winds are always with respect to and observed from an Earth-fixed reference frame. This vector is with respect to the geocentric Earth (Ge)fixed coordinate system.				
U_{Tw}	UBTWX	`bodyWindVelocityWrtGe_ft_s_X` `bodyWindVelocityWrtGe_m_s_X`	Net wind velocity impinging on the vehicle in the X body axis. Net wind is the mean air mass plus any turbulence, shears, or other wind disturbances.	FWD (tailwind)			

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
V$_{Tw}$	VBTWY	`bodyWindVelocityWrtGe_ft_s_Y` `bodyWindVelocityWrtGe_m_s_Y`	Net wind velocity impinging on the vehicle in the Y body axis Net wind is the mean air mass plus any turbulence, shears, or other wind disturbances.	RT			
W$_{Tw}$	VBTWZ	`bodyWindVelocityWrtGe_ft_s_Z` `bodyWindVelocityWrtGe_m_s_Z`	Net wind velocity impinging on the vehicle in the Z body axis Net wind is the mean air mass plus any turbulence, shears, or other wind disturbances.	DN			
		`bodyAngularRateTurbulenceWrtLl_deg_s [3]` `bodyAngularRateTurbulenceWrtLl_rad_s [3]`	Vector of angular turbulence velocities comprised of the three body coordinate system components as defined below. Note: Turbulence angular rate is always with respect to and observed from the locally level (LI) reference frame.				
	PTURB	`bodyAngularRateTurbulenceWrtLl_deg_s_Roll` `bodyAngularRateTurbulenceWrtLl_rad_s_Roll`	Body coordinate roll turbulence	The turbulence would move the aircraft right wing down			
	QTURB	`bodyAngularRateTurbulenceWrtLl_deg_s_Pitch` `bodyAngularRateTurbulenceWrtLl_rad_s_Pitch`	Body coordinate pitch turbulence	The turbulence would move the aircraft nose up			
	RTURB	`bodyAngularRateTurbulenceWrtLl_deg_s_Yaw` `bodyAngularRateTurbulenceWrtLl_rad_s_Yaw`	Body coordinate yaw turbulence	The turbulence would move the aircraft nose right			

Table A.6 — Vehicle physical characteristics

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

I		`bodyMomentOfInertia_slugft2 [3,3]`
		
		`bodyMomentOfInertia_kgm2 [3,3]`
			Matrix of the total moments of inertia of the aircraft. This is with respect to the Cm 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}				

I_x	XIXX	`bodyMomentOfInertia_slugft2_X`
		
		`bodyMomentOfInertia_kgm2_X`
			Vehicle Roll Moment Of Inertia about Cm, body coordinate system	NSC			

I_y	XIYY	`bodyMomentOfInertia_slugft2_Y`
		
		`bodyMomentOfInertia_kgm2_Y`
			Vehicle Pitch Moment Of Inertia about Cm, body coordinate system	NSC			

I_z	XIZZ	`bodyMomentOfInertia_slugft2_Z`
		
		`bodyMomentOfInertia_kgm2_Z`
			Vehicle Yaw Moment Of Inertia about Cm, body coordinate system	NSC			

I_{xz}	XIZX	`bodyProductOfInertia_slugft2_ZX`
		
		`bodyProductOfInertia_kgm2_ZX`
			Vehicle ZX Cross Product Of Inertia about Cm, body coordinate system	NSC			

I_{xy}	XIXY	`bodyProductOfInertia_slugft2_XY`
		
		`bodyProductOfInertia_kgm2_XY`
			Vehicle XYy Cross Product Of Inertia about Cm, body coordinate system	NSC			

I_{yz}	XIYZ	`bodyProductOfInertia_slugft2_YZ`
		
		`bodyProductOfInertia_kgm2_YZ`
			Vehicle YZ Cross Product Of Inertia about Cm, body coordinate system	NSC			

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

	MrcPos	`vrsPositionOfMrc_ft[3]`
		
		`vrsPositionOfMrc_m[3]`
			Vector of the location of the moment reference center (Mrc) of the aircraft in the vehicle reference system (vrs). Comprised of the three components as defined below. This vector is used to define the fixed physical location of the moment reference center in the vehicle. Note that the vrs definition is specific to each vehicle.				

	XMrc	`vrsPositionOfMrc_ft`
		
		`vrsPositionOfMrc_m`
			X Mrc Position				

	YMrc	`vrsPositionOfMrc_ft`
		
		`vrsPositionOfMrc_m`
			Y Mrc Position				

	ZMrc	`vrsPositionOfMrc_ft`
		
		`vrsPositionOfMrc_m`
			Z Mrc position				

Δ_{cg}	DCG	`bodyPositionOfCmWrtMrc_ft [3]`	Vector of the location of the Cm with respect to the moment reference center (Mrc) of the aircraft in the body coordinate system. Comprised of the three components as defined below. This vector is used to define the location of the Cm which moves Wrt to the fixed moment reference center in the vehicle. Note that `vrsPositionOfMrc` locates the Mrc in the vehicle.				

ΔX_{cg}	DXCG	`bodyPositionOfCmWrtMrc_ft_X`
		
		`bodyPositionOfCmWrtMrc_m_X`
			X Cm position	Cm forward of the Mrc			

ΔY_{cg}	DYCG	`bodyPositionOfCmWrtMrc_ft_Y`
		
		`bodyPositionOfCmWrtMrc_m_Y`
			Y Cm position	Cm to the right (out the right wing) of the Mrc			

ΔZ_{cg}	DZCG	`bodyPositionOfCmWrtMrc_ft_Z`
		
		`bodyPositionOfCmWrtMrc_m_Z`
			Z Cm position	Cm below the Mrc			

m	XMASS	`totalMass_slug`
		
		`totalMass_kg`
			Total Mass Of Vehicle (including Fuel, crew, cargo, stores, passengers, etc.)	NSC			

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

W	WEIGHT	`grossWeight_lbf`
		
		`grossWeight_N`
			Aircraft Gross Weight (mass*gravity), including all fuel, occupants, stores, etc.	NSC			

S	AREA	`referenceWingArea_ft2`
		
		`referenceWingArea_m2`
			Reference Wing Area	NSC			

b	SPAN	`referenceWingSpan_ft`
		
		`referenceWingSpan_m`
			Reference Wing Span	NSC			

c	CHORD	`referenceWingChord_ft`
		
		`referenceWingChord_m`
			Mean Aerodynamic Chord (reference wing chord)	NSC			

		`engineMomentOfInertia_slugft2[3,3]`
		
		`engineMomentOfInertia_kgm2[3,3]`
			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.
			
			$I_{EXX} \quad -I_{EXY} \quad -I_{EZX}$
			
			$-I_{EXY} \quad I_{EYY} \quad -I_{EYZ}$
			
			$-I_{EZX} \quad -I_{EYZ} \quad I_{EZZ}$				

I_{Ex}	IEXX	`engineMomentOfInertia_slugft2_X`
		
		`engineMomentOfInertia_kgm2_X`
			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	Positive Sign Convention	Initial Value	Min Value	Max Value

I_{Ey}	IEYY	`engineMomentOfInertia_slugft2_Y`
		
		`engineMomentOfInertia_kgm2_Y`
			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_{Ez}	IEZZ	`engineMomentOfInertia_slugft2_Z`
		
		`engineMomentOfInertia_kgm2_Z`
			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	`engineProductOfInertia_slugft2_XZ`
		
		`engineProductOfInertia_kgm2_XZ`
			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				

I_{Exy}	IEXY	`engineProductOfInertia_slugft2_XY`
		
		`engineProductOfInertia_kgm2_XY`
			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				

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

I_{Eyz}	IEYZ	`engineProductOfInertia_slugft2_YZ`
		
		`engineProductOfInertia_kgm2_YZ`
			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.				

		`fuelTankCentroid_ft[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 structural coordinate system. 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	Positive Sign Convention	Initial Value	Min Value	Max Value

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
		Standard naming convention for crew control inputs "Pos" indicates a position input "Rate" indicates the derivative of the position input "Accel" indicates the derivative of the rate "Force" indicates the force input	Control positions for a vehicle are often uniquely defined for that vehicle, therefore the: Description Positive Sign Convention Initial Value Min Value Max Value definitions may all change based on the particular vehicle. **The table below shall be used as default descriptions and values when they are not otherwise explicitly defined for the vehicle.**				
D_{δ_m}		pilotControlPos_deg_long pilotControlPos_rad_long	Longitudinal control position of the pilot.	AFT 0= vertical when vehicle body is level (pitch and roll attitude = 0)			
		pilotControlRate_deg_s_long pilotControlRate_rad_s_long	Rate of the pilot Longitudinal control movement.	Moving aft			
		pilotControlAccel_deg_s2_long pilotControlAccel_rad_s2_long	Acceleration of the pilot Longitudinal control movement.	Accelerating aft			
		pilotControlForce_lbf_long pilotControlForce_N_long	Longitudinal control force of the pilot.	Aft force			
		The convention and examples above apply to all pilot and copilot controls defined below					

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

D_{δ_l}		`pilotControlPos_deg_lat`
		
		 `pilotControlPos_rad_lat`
			Lateral control position of the pilot.	LEFT
				
				 0= vertical when vehicle body is level (pitch and roll attitude = 0)			

D_{δ_n}		`pilotControlPos_deg_avgPedal`
		
		 `pilotControlPos_rad_avgPedal`
			Net Directional control position of the pilot. Normally, left pedal + right pedal.	Left Pedal in or counter clockwise twist of a sidestick			

		`pilotControlPos_deg_rtPedal`
		
		 `pilotControlPos_rad_rtPedal`
			Right Directional control position of the pilot.	NEGATIVE for Pedal in
				
				 0= pedal full aft
							0.0

		`pilotControlPos_deg_ltPedal`
		
		 `pilotControlPos_rad_ltPedal`
			Left Directional control position of the pilot.	Pedal in.
				
				 0= pedal full aft
						0.0	

D_{δ_z}		`pilotControlPos_deg_collective`
		
		 `pilotControlPos_rad_collective`
			Pilot collective control position.	UP
				
				 0 = full down
						0.0	

		`pilotControlPos_deg_avgThrottle`
		
		 `pilotControlPos_rad_avgThrottle`
			Average pilot throttle control position.	FWD
				
				 0= full aft w/o thrust reversing. Thrust reversing is negative			

		`pilotControlPos_deg_throttle [number of engines]`
		
		 `pilotControlPos_rad_throttle [number of engines]`
			Individual pilot throttle control positions. Order is outboard port (left) to outboard starboard.	FWD
				
				 0= full aft w/o thrust reversing. Thrust reversing is negative			

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

		`copilotControlPos_deg_long`
		
		`copilotControlPos_rad_long`
			Longitundal control position of the copilot.	AFT
				
				0= vertical when vehicle body is level (pitch and roll attitude = 0)			

		`copilotControlPos_deg_lat`
		
		`copilotControlPos_rad_lat`
			Lateral control position of the copilot.	LEFT
				
				0= vertical when vehicle body is level (pitch and roll attitude = 0)			

		`copilotControlPos_deg_avgPedal`
		
		`copilotControlPos_rad_avgPedal`
			Net Directional control position of the copilot. Normally, Left pedal + right pedal.	Left Pedal in or counter clockwise twist of a sidestick			

		`copilotControlPos_deg_rtPedal`
		
		`copilotControlPos_rad_rtPedal`
			Right Directional control position of the copilot.	NEGATIVE for Pedal in
				
				0= pedal full aft
							0.0

		`copilotControlPos_deg_ltPedal`
		
		`copilotControlPos_rad_ltPedal`
			Left Directional control position of the copilot.	Pedal in.
				
				0= pedal full aft
						0.0	

		`copilotControlPos_deg_collective`
		
		`copilotControlPos_rad_collective`
			Copilot collective control position.	UP
				
				0 = full down
						0.0	

		`copilotControlPos_deg_avgThrottle`
		
		`copilotControlPos_rad_avgThrottle`
			Average copilot throttle control position.	FWD
				
				0= full aft w/o thrust reversing. Thrust reversing is negative			

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

		`copilotControlPos_deg_throttle [number of engines]`
		
		`copilotControlPos_rad_throttle [number of engines]`
			Individual copilot throttle control positions. Order is outboard port (left) to outboard starboard.	FWD
				
				0= full aft w/o thrust reversing. Thrust reversing is negative			

		`controlPos_deg_avgThrottle`
		
		`controlPos_rad_avgThrottle`
			Average pilot and copilot throttle control position.	FWD
				
				0= full aft w/o thrust reversing. Thrust reversing is negative			

		`controlPos_deg_avgPropellor`
		
		`controlPos_rad_avgPropellor`
			Average pilot and copilot propeller blade pitch control position.	FWD
				
				0=flat pitch, Thrust reversing is negative			

		`controlPos_deg_propellor [number of engines]`
		
		`controlPos_rad_propellor [number of engines]`
			Individual propeller blade pitch control position. Order is outboard port (left) to outboard starboard.	FWD
				
				0=flat pitch, Thrust reversing is negative			

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

Standard naming convention for control surfaces
 	
"Pos" indicates a control surface position
 	
"Rate" indicates the derivative of the control surface position
 	
"Accel" indicates the derivative of the control surface rate
 	
"HingeMoment" indicates the hinge moment on the control surface (sign convention = + deflection results in + hinge moment)
			Control surface positions for a vehicle are often uniquely defined for that vehicle, therefore the:
			
			  Description
			
			  Positive Sign Convention
			
			  Initial Value
			
			  Min Value
			
			  Max Value
			
			definitions may all change based on the particular vehicle.
			
			The table below shall be used as default descriptions and values when they are not otherwise explicitly defined for the vehicle.				

δ_{TEF_n}		controlSurfacePos_deg_TEF [number of trailing edge flap control surfaces] controlSurfacePos_rad_TEF [number of trailing edge flap control surfaces]	Vector of trailing edge flap positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	TED 0 deflection is flap retracted			
		controlSurfaceRate_deg_s_TEF [number of trailing edge flap control surfaces] controlSurfaceRate_rad_s_TEF [number of trailing edge flap control surfaces]	Vector of trailing edge flap deflection rates, one for each surface. Order is outboard port (left) to outboard starboard.	TED			
		controlSurfaceAccel_deg_s2_TEF [number of trailing edge flap control surfaces] controlSurfaceAccel_rad_s2_TEF [number of trailing edge flap control surfaces]	Vector of trailing edge flap deflection accelerations, one for each surface. Order is outboard port (left) to outboard starboard.	TED			

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
		`controlSurfaceHingeMoment_ftlbf_TEF [number of trailing edge flap control surfaces]` `controlSurfaceHingeMoment_Nm_s_TEF [number of trailing edge flap control surfaces]`	Vector of the hinge moments on the trailing edge flap, one for each surface. Order is outboard port (left) to outboard starboard.	+ = TEU moment (positive deflection results in positive moment)			
		The convention and examples above apply to all control surface positions defined below					
δ_{TEF_n}		`controlSurfacePos_deg_TEF [number of leading edge flap control surfaces]` `controlSurfacePos_rad_TEF [number of leading edge flap control surfaces]`	Vector of trailing edge flap (TEF) positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	TED			
δ_{TEF}		`controlSurfacePos_deg_avgTEF` `controlSurfacePos_rad_avgTEF`	Trailing edge flap deflection (TEF). Average for all trailing edge flap surfaces.	TED 0 deflection is flap retracted			
$\delta_{\delta_{TEF}}$		`controlSurfacePos_deg_diffTEF` `controlSurfacePos_rad_diffTEF`	Differential trailing edge flap (DTEF)deflection (left deflections {+=TED} -right deflections {+=TED})	LT TED (RWD moment)			
δ_{LEF_n}		`controlSurfacePos_deg_LEF [number of leading edge flap control surfaces]` `controlSurfacePos_rad_LEF [number of leading edge flap control surfaces]`	Vector of leadng edge flap (LEF) positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	LED			
δ_{LEF}		`controlSurfacePos_deg_avgLEF` `controlSurfacePos_rad_avgLEF`	Leading edge flap/slat deflection. Average for all deflected leading edge flap/slat surfaces.	LED			

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

$\delta_{\delta_{LEF}}$		`controlSurfacePos_deg_diffLEF`
		
		`controlSurfacePos_rad_diffLEF`
			Differential leading edge flap (LEF) deflection (left deflections-right deflections)	LT LED
				
				(RWD moment)			

δ_{SP_n}		`controlSurfacePos_deg_spoiler [number of spoiler control surfaces]`
		
		`controlSurfacePos_rad spoiler [number of spoiler control surfaces]`
			Vector of spoiler control surface positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	TEU			

δ_{SP}		`controlSurfacePos_deg_avgSpoiler`
		
		`controlSurfacePos_rad_avgSpoiler`
			Spoiler deflection. Average for all deflected spoilers (sum of positions/number of surfaces)	TEU			

$\delta_{\delta_{SP}}$		`controlSurfacePos_deg_diffSpoiler`
		
		`controlSurfacePos_rad_diffSpolier`
			Differential spoiler control surface position (right deflections-left deflections)	RT TEU
				
				(RWD moment)			

δ_{A_n}		`controlSurfacePos_deg_aileron [number of aileron control surfaces]`
		
		`controlSurfacePos_rad_aileron  [number of aileron control surfaces]`
			Vector of aileron control positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	TEU			

δ_{δ_A}		`controlSurfacePos_deg_diffAileron`
		
		`controlSurfacePos_rad_diffAileron`
			Differential aileron deflection, (right deflections-left deflections)	Right aileron TEU
				
				(RWD moment)			

$\delta_{Atab_{n,i}}$		`controlSurfacePos_deg_aileronTab [number of aileron control surfaces, number of tabs]`
		
		`controlSurfacePos_rad_aileronTab [number of aileron control surfaces, number of tabs]`
			Array of aileron tab control positions (i) for each surface deflected (n). Order is outboard port to outboard starboard	TEU			

δ_{Atab}		`controlSurfacePos_deg_avgAileronTab`
		
		`controlSurfacePos_rad_avgAileronTab`
			Average aileron tab deflection (sum of positions/number of surfaces)	TEU			

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

$\delta_{\delta_{Atab}}$		`controlSurfacePos_deg_diffAileronTab`
		
		`controlSurfacePos_rad_diffAileronTab`
			Differential aileron tab deflection (right tab deflections-left tab deflections)	RT Aileron Tab TEU
				
				(RWD moment)			

δ_{R_n}		`controlSurfacePos_deg_rudder [number of rudder control surfaces]`
		
		`controlSurfacePos_rad_rudder [number of rudder control surfaces]`
			Vector of rudder control positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.	TEL			

δ_R		`controlSurfacePos_deg_avgRudder`
		
		`controlSurfacePos_rad_avgRudder`
			Average rudder deflection (sum of positions/number of surfaces)	TEL			

δ_{δ_R}		`controlSurfacePos_deg_diffRudder`
		
		`controlSurfacePos_rad_diffRudder`
			Differential rudder deflection (right deflections-left deflections)	RT Rudder TEL
				
				(ANR moment)			

$\delta_{Rtab_{n,i}}$		`controlSurfacePos_deg_rudderTab [number of rudder control surfaces, number of tabs]`
		
		`controlSurfacePos_rad_rudderTab [number of rudder control surfaces, number of tabs]`
			Array of rudder tab control positions (i) for each surface deflected (n). Order of tabs is upper to lower.	TEL			

δ_{Rtab}		`controlSurfacePos_deg_avgRudderTab`
		
		`controlSurfacePos_rad_avgRudderTab`
			Average rudder tab deflection (sum of positions/number of surfaces)	RT Rudder Tab TEL
				
				(ANR moment)			

$\delta_{\delta_{Rtab}}$		`controlSurfacePos_deg_diffRudderTab`
		
		`controlSurfacePos_rad_diffRudderTab`
			Differential rudder tab deflection (right deflections-left deflections)	RT Rudder Tab TEL
				
				(ANR moment)			

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
δ_{E_n}		`controlSurfacePos_deg_elevator [number of elevator control surfaces]` `controlSurfacePos_rad_elevator [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			
δ_E		`controlSurfacePos_deg_avgElevator` `controlSurfacePos_rad_avgElevator`	Average elevator (or stabilizer/stabilator) deflection) (sum of positions/number of surfaces)	TEU			
δ_{δ_E}		`controlSurfacePos_deg_diffElevator` `controlSurfacePos_rad_diffElevator`	Differential elevator deflection (right deflections-left deflections)	Right control TEU (RWD moment)			
$\delta_{Etab_{n,i}}$		`controlSurfacePos_deg_elevatorTab [number of elevator tab control surfaces, number of tabs]` `controlSurfacePos_rad_elevatorTab [number of elevator tab control surfaces, number of tabs]`	Array of elevator (or stabilizer/stabilator) tab positions (i) for each surface (n). Order is outboard port (left) to outboard starboard per control surface.	TEU			
δ_{Etab}		`controlSurfacePos_deg_avgElevatorTab` `controlSurfacePos_rad_avgElevatorTab`	Average elevator (or stabilizer/stabilator) tab control surface positions (sum of positions/number of surfaces)	TEU			
$\delta_{\delta_{Etab}}$		`controlSurfacePos_deg_diffElevatorTab` `controlSurfacePos_rad_diffElevatorTab`	Average differential elevator (or stabilizer/stabilator) tab deflection (right deflections-left deflections)	Right control TEU (RWD moment)			

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

δ_{C_n}		`controlSurfacePos_deg_canard [number of canard control surfaces]`
		
		`controlSurfacePos_rad_canard [number of canard control surfaces]`
			Vector of canard control positions, one for each surface. Order is outboard port (left) to outboard starboard.	TED		0.0	

δ_C		`controlSurfacePos_deg_avgCanard`
		
		`controlSurfacePos_rad_avgCanard`
			Average canard position (sum of positions/number of surfaces)	TED		0.0	

$\delta_{\delta C}$		`controlSurfacePos_deg_diffCanard`
		
		`controlSurfacePos_rad_diffCanard`
			Average differential canard deflection (LEFT deflections-RIGHT deflections)	LEFT control TED
				
				(RWD moment)			

$\delta_{Ctab_{n,i}}$		`controlSurfacePos_deg_canardTab [number of canard tab control surfaces, number of tabs]`
		
		`controlSurfacePos_rad_canardTab [number of canard tab control surfaces, number of tabs]`
			Array of canard tab positions (i) for each surface (n). Order is outboard port (left) to outboard starboard per control surface.	TED			

δ_{Ctab}		`controlSurfacePos_deg_avgCanardTab`
		
		`controlSurfacePos_rad_avgCanardTab`
			Average canard tab deflection (sum of positions/number of surfaces)	TED			

$\delta_{\delta Ctab}$		`controlSurfacePos_deg_diffCanardTab`
		
		`controlSurfacePos_rad_diffCanardTab`
			Average differential canard tab deflection (LEFT deflections-RIGHT deflections)	LEFT control TED
				
				(RWD moment)			

δ_{SB}		`controlSurfacePos_deg_speedbrake`
		
		`controlSurfacePos_rad_speedbrake`
			Speedbrake deflection	Extended			

δ_{LGn}		`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		0.0	

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

		`landingGearWeightOnWheels_lbf [number of landing gear struts]`
		
		`landingGearWeightOnWheels_N [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_rad_s [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	Positive Sign Convention	Initial Value	Min Value	Max Value
C_L	CL	`totalCoefficientOfLift`	Coefficient Of Lift, Total, includes effects of stores	UP			
C_D	CD	`totalCoefficientOfDrag`	Coefficient Of Drag, Total, includes effects of stores	AFT			
		`aeroBodyForceCoefficient[3]`	Vector of total aerodynamic force coefficients in the body coordinate system, comprised of the three components as defined below.				
C_X	CX	`aeroBodyForceCoefficient_X`	X-body Force Coefficient due to aerodynamic loads, includes stores (Body coordinate)	FWD			
C_Y	CY	`aeroBodyForceCoefficient_Y`	Y-body Force Coefficient due to aerodynamic loads, includes stores (Body coordinate)	RT			

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
C_Z	CZ	`aeroBodyForceCoefficient_Z`	Z-body Force Coefficient due to aerodynamic loads, includes stores (Body coordinate)	DN			

		`aeroBodyForce_lbf [3]`
		
		`aeroBodyForce_N [3]`
			Vector of total aerodynamic forces in the body coordinate system, including stores. Comprised of the three components as defined below.				

F_{AX}	FAX	`aeroBodyForce_lbf_X`
		
		`aeroBodyForce_N_X`
			Total X-body Force due to aerodynamic loads, includes stores (Body coordinate)	FWD			

F_{AY}	FAY	`aeroBodyForce_lbf_Y`
		
		`aeroBodyForce_N_Y`
			Total Y-body Force due to aerodynamic loads, includes stores (Body coordinate)	RT			

F_{AZ}	FAZ	`aeroBodyForce_lbf_Z`
		
		`aeroBodyForce_N_Z`
			Total Z-body Force due to aerodynamic loads, includes stores (Body coordinate)	DN			

		`thrustBodyForce_lbf [3]`
		
		`thrustBodyForce_N [3]`
			Vector of total net propulsion system forces in the body coordinate system (includes installation losses, inlet efficiency and propeller efficiency). Comprised of the three components as defined below.				

F_{EX}	FEX	`thrustBodyForce_lbf_X`
		
		`thrustBodyForce_N_X`
			Total net engine thrust Force, X-body axis	FWD			

F_{EY}	FEY	`thrustBodyForce_lbf_Y`
		
		`thrustBodyForce_N_Y`
			Total net engine thrust Force , Y-body axis	RT			

F_{EZ}	FEZ	`thrustBodyForce_lbf_Z`
		
		`thrustBodyForce_N_Z`
			Total net engine thrust Force, Z-body axis	DN			

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

		`gearBodyForce_lbf  [3]`
		
		`gearBodyForce_N [3]`
			Vector of total landing gear ground reaction forces in the body coordinate system. Does NOT include aerodynamic forces on the landing gear that are included in `aeroBodyForce` defined above. Comprised of the three components as defined below.				

F_{GX}	FGX	`gearBodyForce_lbf_X`
		
		`gearBodyForce_N_X`
			Total landing gear ground reaction force, X-body axis	FWD			

F_{GY}	FGY	`gearBodyForce_lbf_Y`
		
		`gearBodyForce_N_ Y`
			Total landing gear ground reaction force, Y-body axis	RT			

F_{GZ}	FGZ	`gearBodyForce_lbf_Z`
		
		`gearBodyForce_N_Z`
			Total landing gear ground reaction force, Z-body axis	DN			

		`totalBodyForce_lbf [3]`
		
		`totalBodyForce_N [3]`
			Vector of total forces in the body coordinate system. Includes all forces exerted upon the aircraft. Comprised of the three components as defined below.				

F_{xTOT}	FX	`totalBodyForce_lbf_X`
		
		`totalBodyForce_N_X`
			Total Forces On a/c, X-body axis	FWD			

F_{yTOT}	FY	`totalBodyForce_lbf_Y`
		
		`totalBodyForce_N_Y`
			Total Forces On a/c, Y-body axis	RT			

F_{zTOT}	FZ	`totalBodyForce_lbf_Z`
		
		`totalBodyForce_N_Z`
			Total Forces On a/c, Z-body axis	DN			

		`aeroBodyMomentCoefficient [3]`	Vector of total aerodynamic moment coefficients in the body coordinate system, including stores. Comprised of the three components as defined below.				
C_l	CLL	`aeroBodyMomentCoefficient_Roll`	Total Aerodynamic Rolling Moment Coefficient including stores. Moment about the X-body axis	RWD			

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
C_m	CLM	`aeroBodyMomentCoefficient_Pitch`	Total Aerodynamic Pitching Moment Coefficient, including stores. Moment about the Y-body axis	ANU			
C_n	CLN	`aeroBodyMomentCoefficient_Yaw`	Total Aerodynamic yawing Moment Coefficient, including stores. Moment about the Z-body axis	ANR			

		`aeroBodyMoment_ftlbf [3]`
		
		`aeroBodyMoment_Nm [3]`
			Vector of total aerodynamic moments in the body coordinate system, including stores. Referenced to the moment reference center. Comprised of the three components as defined below.				

L_A	TAL	`aeroBodyMoment_ftlbf_Roll`
		
		`aeroBodyMoment_Nm_Roll`
			Total Aerodynamic Rolling moment (including attached stores), about the X-body axis	RWD			

M_A	TAM	`aeroBodyMoment_ftlbf_Pitch`
		
		`aeroBodyMoment_Nm_Pitch`
			Total Aerodynamic pitching moment (including attached stores), about the Y-body axis	ANU			

N_A	TAN	`aeroBodyMoment_ftlbf_Yaw`
		
		`aeroBodyMoment_Nm_Yaw`
			Total Aerodynamic yawing moment (including attached stores), about the Z-body axis	ANR			

		`thrustBodyMoment_ftlbf [3]`
		
		`thrustBodyMoment_Nm [3]`
			Vector of total net propulsion system moments in the body coordinate 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	`thrustBodyMoment_ftlbf_Roll`
		
		`thrustBodyMoment_Nm_Roll`
			Total Engine Rolling Moment, about the X-body axis	RWD			

M_E	TEM	`thrustBodyMoment_ftlbf_Pitch`
		
		`thrustBodyMoment_Nm_Pitch`
			Total Engine pitching Moment, about the Y-body axis	ANU			

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

N_E	TEN	`thrustBodyMoment_ftlbf_Yaw`
		
		`thrustBodyMoment_Nm_Yaw`
			Total Engine yawing Moment, about the X-body axis	ANR			

		`landingGearBodyMoment_ftlbf [3]`
		
		`landingGearBodyMoment_Nm [3]`
			Vector of total landing gear ground reaction moments in the body coordinate system. Referenced to the moment reference center. Does NOT include aerodynamic moments on the landing gear that are included in `aeroBodyMoment` defined above. Comprised of the three components as defined below.				

L_G	TGL	`landingGearBodyMoment_ftlbf_Roll`
		
		`landingGearBodyMoment_Nm_Roll`
			Total Landing Gear Rolling Moment, about the X-body axis	RWD			

M_G	TGM	`landingGearBodyMoment_ftlbf_Pitch`
		
		`landingGearBodyMoment_Nm_Pitch`
			Total Landing gear Pitch Moment, about the Y-body axis	ANU			

N_G	TGN	`landingGearBodyMoment_ftlbf_Yaw`
		
		`landingGearBodyMoment_Nm_Yaw`
			Total Landing Gear Yawing Moment, about the Z-body axis	ANR			

		`totalBodyMoment_ftlbf [3]`
		
		`totalBodyMoment_Nm [3]`
			Vector of total moments in the body coordinate 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	`totalBodyMoment_ftlbf_Roll`
		
		`totalBodyMoment_Nm_Roll`
			Total Rolling Moment, about the X-body axis	RWD			

M_{TOT}	TTM	`totalBodyMoment_ftlbf_Pitch`
		
		`totalBodyMoment_Nm_Pitch`
			Total Pitching Moment, about the Y-body axis	ANU			

N_{TOT}	TTN	`totalBodyMoment_ftlbf_Yaw`
		
		`totalBodyMoment_Nm_Yaw`
			Total Yawing Moment, about the Z-body axis	ANR			

Table A.9 — Simulation control parameters

Symbol	Short Name	Full Variable Name	Description	Positive Sign Convention	Initial Value	Min Value	Max Value
	TIME	`simDuration_s`	Time Since Start Of Operate Mode	NSC			
		`deltaTime_s[number of different integration step sizes]`	Vector of Integration step sizes				
		`simDate`	Date simulated. Date at the start of the simulation is used. (Not the date the simulation run was made) Type yyyy-mm-dd				
		`simTime`	Simulated time of day based on 24 hours Zulu. ==Type hh:mm:ss.ss==				
		`productionDate`	Date the simulation run was made (Not the simulated date [simDate]) Type yyyy-mm-dd				

References

GRAM 07.

```
Justus, C.G.; and Leslie, F. W.: "The NASA MSFC Earth Global Reference
Atmospheric Model—2007 Version" NASA TM-2008-215581
```

For symbols:

ISO 1151-1 : 1988

ISO 1151-2 : 1985

ISO 1151-4 : 1994(E)

ISO 1151-5 : 1987

Not all symbols presented above are defined by ISO. This document attempts to make them consistent where similar.

For Time and Dates:

ISO 8601:2000

DAVE-ML Website (Informative)

The "official" DAVE-ML site is http://daveml.org/. This link contains all DAVE-ML documentation and links and information on DAVE-ML tools and applications. Additional information is available at http://www.aiaa.org/

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

Version 2.0 (Release Candidate 3)

2010-04-23

AIAA Modeling and Simulation Technical Committee [https://info.aiaa.org/tac/ASG/MSTC]

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

Abstract

The Dynamic Aerospace Vehicle Exchange Markup Language (DAVE-ML) is a text-based file format intended for encoding the principal elements of a flight simulation model for an aerospace vehicle. It is based on two other open standards: the Extensible Markup Language (XML) version 1.1 and the Mathematical Markup Language (MathML) version 2.0, both products of the World Wide Web Consortium. DAVE-ML defines additional grammar (markup elements) to provide a domain-specific language capable of aerospace flight dynamics modeling, verification, and documentation.

This markup language represents the encoding format for BSR/AIAA S-119 Flight Dynamic Model Exchange Standard [AIAA10].

This is a draft version of the reference manual for DAVE-ML syntax and markup. DAVE-ML syntax is specified by the `DAVEfunc.dtd` Document Type Definition (DTD) 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 American Institute of Aeronautics and Astronautics (AIAA) members. Contact the author above for more information or comments regarding refinement of DAVE-ML.

Table of Contents

B-1. Changes to this document

This section contains a list of changes during the development of the DAVE-ML DTD.

B-1.1. Changes since version 2.0RC2

This section outlines changes to the DTD and this reference manual since version 2, release candidate 2, was released in October 2008.

The changes are divided into *structural* changes to the DTD and *non-structural* changes to the DTD and reference manual.

B-1.1.1. DTD structural changes since version 2.0RC2

- Corrected the DTD to require at least one `staticShot` child element of the `checkData` element.

- Changed the DTD logic for inclusion of provenance information for elements `griddedTableDef`, `ungriddedTableDef`, `function`, `checkData`, and `staticShot` from

      (provenance? | provenanceRef?)

  to

      (provenance | provenanceRef)?

  which is the more correct way to display an optional selection from two choices. However, use of `provenance` or `provenanceRef` is now deprecated for `checkData` elements as described in the non-structural changes below; this change was made for consistency with the other elements that have provenance information.

- Added optional, mutually exclusive flag elements `isInput`, `isControl`, and `isDisturbance` to conform with the latest draft of [AIAA10]. These flags are intended to help clarify the role of all defined variables and to assist in analyses such as linear model extraction.

- Changed the Uniform Resource Identifier (URI) for the *atan2* function definition to the daveml.org domain.

- Changed SYSTEM ID to reflect new daveml.org domain; changed PUBLIC ID from NASA to AIAA.

- 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 white space 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.

B-1.1.2. Non-structural DTD and reference manual changes since version 2.0RC2

- Amplified and elaborated on possible values for `interpolate` and `extrapolate` attributes for independent variables of function definitions.

- Added an index to the reference manual.

- Removed the sign convention list as this is now in the overlying AIAA draft standard, S-119 [AIAA10].

- Added clarification that, while uncertainty can be applied in multiple places in the model (including input, calculations, functions, and outputs), it is probably not a good practice to do so.

- Changed the non-dimensional signal units-of-measure indication from blank or ND to 'nd', in accordance with the latest draft standard [AIAA10]; removed plus sign from front of sign convention since it doesn't always apply and is redundant to the definition.

- Added references to earlier standards papers by B. Hildreth [Hildreth94], [Hildreth98].

- Cleaned up formatting in the element descriptions found in Section B-8.2 [55]; corrected the grammar in the BNF descriptions of the major elements within this text.

- Added two sentences to the description of the calculation element, explaining how to tie variables in the model to the MathML expression via the MathML content identifier (`ci`) element. Thanks to Missy Hill and Curtis Zimmerman for pointing out the lack of explanation.

- Changed email address for B. Hildreth; added persistent email at daveml.org; changed default Uniform Resource Locator (URL) to daveml.org

- Changed the description of `bounds` element in the DTD to remove out-of-date reference to [un]griddedTable[DeflRef] since this is no longer supported. In the reference manual, removed a misleading second bounds element for the uniformPDF element case and added notes to imply the presence of PCDATA (either one or two, depending on the case). Thanks to Dan Newman for prompting this review and change.

- Did more DTD clean-up to remove redundant 'optional' specifiers on elements and attributes; made the 'internal identifier' descriptions consistent throughout the DTD and reference manual; tweaked the style of the reference pages in the reference manual; fixed poor grammar in the DTD; replaced parentheses with brackets in the BNF syntax in the reference manual for comments to avoid confusion; reformatted the reference pages in the manual for readability; and added references to W3C on-line documentation where necessary. Thanks to Dennis Linse for encouraging the correction of 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 to Dennis Linse 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 both of them have been deprecated in version 2.0. The clarification was made after near-simultaneous suggestions from 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 typographical errors in `fileHeader`, `variableDef`, `griddedTableDef`, `ungriddedTableDef`, and `function` element schematic overview syntaxes in sections 6.2.1-6.2.6. Thanks to Dennis Linse for catching these errors.

- Added a sentence about the `tol` sub-element being an absolute difference to remove ambiguity over its interpretation.

- In the reference manual syntax layout, moved the `provenance` and `provenanceRef` sub-elements out of `checkData` and into `staticShot` sub-elements since `checkData`, a singleton place-holder, 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 of figures and incorporated several new reference citations. Added interpolation paragraph.

- Deprecated `signalID` used for internal signals in check-case data in favor of the more consistent `varID` (which meant the introduction of a formal `varID` element) and made the specification of `signalUnits` sub-element mandatory for input and output signals for consistency. Thanks to Dan Newman for helping solidify the thinking about this issue.

- Changed examples in this text to use updated AIAA variable names to match a 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 check-case examples to match the draft AIAA standard.

- Removed the `symmetric` attribute of the `uniformPDF` sub-element; this attribute was redundant as symmetric distribution is implied with a single `bounds` sub-element, and asymmetric is implied by two `bounds` sub-elements. 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 problems 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 the `staticShot` element in response to a suggestion by Dennis Linse; and added a typical description to example listings.

- Added a section about namespaces and removed the hard link in the DTD that incorrectly set the 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` sub-element 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 sub-elements of the `variableDef` and `checkData` elements. Thanks to Dennis Linse for the correction.

- Added '[Deprecated]' to 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 suggestions of Geoff Brian, Giovanni Cignoni, Randy Brumbaugh, and Dan 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 the DAVE-ML exchange test [Jackson04] paper.

- Added `discrete` enumeration selection to `interpolate` attribute of `independentVarPts` and `independentVarRef` elements at the suggestion of Geoff Brian, Australian Defence Science and Technology Organisation (DSTO).

- Added a section and a `variableDef` example on extending the MathML-2 function set with *atan2*.

- Removed all `xns` attributes from examples.

- Emphasized 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) and fixed a typographical error 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 and added a `date` attribute to the `modificationRecord` element, at the suggestion of Geoff Brian of DSTO.

- Added 2D and 3D ungridded table examples and figures and corrected a typographical error in the ungridded table definition syntax (thanks to Dr. Peter Grant of U. Toronto's UTIAS and Geoff Brian of DSTO).

- Reintroduced <!ENTITY> to include MathML-2 DTD (complete) in the body of this DTD. This entity definition quietly went away in version 1.6 due to a misunderstanding of the proper way to include external DTDs. It was reintroduced to assist with validating parsers.

- Added a `description` sub-element to the `provenance` element, so the provenance entry can contain more information about change justification documents and made `provenance` or `provenanceRef` acceptable sub-elements to `variableDef` and `checkData` elements at the request of Geoff Brian of DSTO.

B-1.6. Changes since version 1.7b1

- Renamed `docID` attribute to `refID` in the `modificationRecord` so the attribute name is consistent; the `docID` attribute is deprecated but remains in place 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 how to contact 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` attribute corrected to `utID`.

- Allowed multiple `author` elements wherever only one had been allowed before.

- Added a new tag, `isStdAIAA`, to indicate that 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 true email from examples 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 that a `variableDef` refers to a state variable in the model.

- Added a new tag, `isStateDeriv`, to indicate that 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-2 content markup over presentation markup. Fixed several grammatical and typographical errors and added figure 1. Added ISO 8601 (Dates and Times) reference.

B-1.9. Changes since version 1.5b2

- Added Bill Cleveland (NASA Ames Research Center's 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 (specifically `function`, `griddedTableDef` and `ungriddedTableDef` elements) of `provenance` elements to enable them to use a `provenanceRef` reference instead to eliminate duplicate `provenance` elements.

- Realization dawned that there was little difference between `griddedTables` and `griddedTableDefs` but that the latter was more flexible (ditto `ungriddedTables` and `ungriddedTableDefs`). By making the `gtID` and `utID` attributes 'implied' instead of 'required,' we can use the `Def` versions in both referenced-table and embedded-table `functions`. 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 `griddedTables` as `griddedTableDefs`.

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 does not 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 dynamics models. The intent of DAVE-ML is to significantly expedite the process of re-hosting a simulation model from one facility to another and function 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/].

The exchange of aerospace vehicle flight dynamics models may derive many benefits from the application of XML in general, and DAVE-ML in particular:

- Provides a human-readable text description of the model

- Provides an unambiguous machine-readable model description, suitable for conversion into programming language or direct import into object-oriented data structures at run-time

- Allows use of the same source file for computer-aided design and real-time piloted simulation

- Based on open, non-proprietary, standards that are language- and facility-independent

- Allows inclusion of statistical properties, such as confidence bounds and uncertainty ranges, suitable for Monte Carlo or other statistical analysis of the model

- Complies with emerging AIAA simulation data standards

- Represents a self-contained, complete, archivable data package, including references to reports, wind-tunnel tests, author contact information, and data provenance

- Is 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.org*   [http://daveml.org]

B-3. Purpose

DAVE-ML is intended to encode (for exchange 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, and variable names/units/sign conventions. The staff 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 to automating most of this tedious, manual process. Often, when a pair of facilities has already 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 develop a common exchange format for these flight dynamic models. The advantage gained is to enable any simulation facility or laboratory, after having written a DAVE-ML import and/or export script, to 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 most aircraft simulation packages: the aerodynamics model. This initial version of DAVE-ML can be used to transport a complete aerodynamics model, including descriptions of the aerodynamic build-up equations and data tables, and include references to the documentation about the aerodynamics 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 and inertia model, or a weapons load-out model, or perhaps a navigational database. The only requirement is that model outputs must be unambiguously defined in terms of inputs, with no past history (state) information required.

DAVE-ML forms the encoding portion of the Flight Simulation Model Exchange Standard, BSR/ AIAA S-119-2010 (currently in draft form). More information is available at the S-119 web site [AIAA10].

B-4. Background

The idea of a universally understood flight dynamics data package has been discussed for at least two decades within the AIAA technical committees [Hildreth94], [Hildreth98]. 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 Flight Research Center, developed a hybrid 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 in the early 1990s. Some other simulation standards have subsequently been proposed by the AIAA and are under active consideration at this writing [AIAA10].

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 includes a methodology for accomplishing the gradual standardization of simulation model components, a mechanism for standardizing variable names within math models, and proposed Hierarchical Data Format 5 (HDF5) as the data format. This document is included as an Annex to the standard [AIAA01], [AIAA03], [AIAA10].

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.

The 2002 proposal included several key aspects from the draft standard, including allowing use of a standard variable-name convention and 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 the results of a demonstration using DAVE-ML to exchange 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 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 (LaRC) [Hill07] and at NASA Ames Research Center (ARC). 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, to define mathematical relationships. MathML-2 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-2 home web page, found at http://www.w3.org/Math/.

MathML-2 provides a mostly 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 2D coordinate set. DAVE-ML provides a means of extending MathML-2 for a predefined set of functions (currently only the *atan2* function is defined). Thus, a DAVE-ML-compliant processing tool should recognize this extension (which is accomplished by using the MathML-2 `csymbol` element). See Section B-6.2.2 [20] for a discussion and example B-6 [25].

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 inputs 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-2 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 and calculations through the subsystem model into outputs. 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 from the model, and/or the results of intermediate calculations. Variables must be defined for each input and output of any function element as well as any input or output of the subsystem represented. MathML-2 [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, but any *presentation* markup is not required and should be ignored by the processing application (except as required to generate documentation). 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 or white space. 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 aerodynamics 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 1 [15] contains excerpts from an example model, showing five of the six major parts of a DAVE-ML file.

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE DAVEfunc SYSTEM "DAVEfunc.dtd">
<DAVEfunc>
<fileheader> ... </fileheader>
<variableDef varID="DBFLL" ...>
    ...... (DBFLL variable description) .....
</variableDef>
<variableDef varID="XMACH" ...>
    ... (Mach variable description) ....
</variableDef>
<variableDef varID="CLBFLL" ...>
    ... (CLBFLL variable description) ...
</variableDef>
        .
<breakpointDef bpID="XMACH1_PTS" ... >
    <bpVals>0.3, 0.6, 0.8, ... </bpVals>
</breakpointDef>
<breakpointDef bpID="DBFL_PTS_PTS" ... >
    <bpVals>0., 15., 30., ... </bpVals>
</breakpointDef>
        .
<griddedTableDef gtID="CLBFL_table">
    <breakpointRefs>
        <bpRef bpID="DBFL_PTS"/>
        <bpRef bpID="XMACH1_PTS"/>
    </breakpointRefs>
    <dataTable>
-0.864E-02, -0.1026E-01, ...
-0.863E-02, -0.5367E-02, ...
    </dataTable>
</griddedTableDef>
        .
<function ... >
    <independentVarRef varID="DBFLL" .../>
    <independentVarRef varID="XMACH" .../>
    <dependentVarRef varID="CLBFLL" .../>
    <functionDefn ... >
        <griddedTableRef gtID="CLBFL_table"/>
    </functionDefn>
</function>
        .
        .
</DAVEfunc>
```

DBFLL is the lower-left body flap deflection (input to model)

XMACH is flight Mach number (input to model)

CLBFLL is the lift coefficient contribution due to lower-left body flap deflection (function output)

Breakpoint set definitions; may be reused by several functions

Multi-dimensional nonlinear function description; may be shared (left- and right-lower body flap lift contribution, for example)

Function element ties together input/output variables, breakpoint sets, and function table points

Check-case (implementation verification) data not shown

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.

In DAVE-ML 2.0, all signals are real and scalar.

The `variableDefs` 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 multiple `function` definition if desired for efficiency. Alternatively, some or all `functions` 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 multiple `function` definitions if necessary for efficiency. Alternatively, `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 inputs is important and must match the order given in the parent `function`. Thus, functions that share an ungridded table definition 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 1 [15]. 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 are 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 modification history.

```
fileHeader : [name]
    author+ : name, org, [email]
        contactInfo* : [contactInfoType, contactLocation]
            {text describing contact information for author}
    creationDate : date {in ISO 8601 YYYY-MM-DD format}
    fileVersion?
```

```
                        {file version identifier}
            description?
                {textual description of model}
            reference* : refID, author, title, date, [classification, accession, href]
                description? :
                    {textual information about reference}
            modificationRecord* : modID, date, [refID]
                author+ : name, org, [email]
                    contactInfo* : [contactInfoType, contactLocation]
                        {text describing contact information for author}
                description?
                    {textual description of modification}
                extraDocRef* : refID
            provenance* :
                    author+ : name, org, [email]
                        contactInfo* : [contactInfoType, contactLocation]
                            {text describing contact information for author}
                    creationDate : date {in ISO 8601 YYYY-MM-DD format}
                    documentRef* : [docID,] refID
                    modificationRef* : modID
                    description?
                        {textual description of the background of the model}
```

`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 [49] 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 a good practice would include an automated revision number from a version 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. This sub-element can include a description of the reference.
`modificationRecord`	An optional list of one or more modifications with optional reference IDs, 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. Use of a `provID` attribute in the `fileHeader` is unnecessary since this provenance applies to the entire model unless otherwise specified.

Example B-1. An 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
     NASA TM-107580. This aero model was used for HL-20 approach and
     landing studies at NASA Langley Research Center during 1989-1995
     and for follow-on studies at NASA Johnson Space Center in 1994
     and NASA Ames Research Center in 2001. This DAVE-ML version was
     created in March 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 make models more readable by humans.
❷ Start of the `fileHeader` element.
❸ Creation date of file, in ISO-8601 format. See Section B-6.5.4 [49]
❹ In this example, the revision number is automatically inserted by a version control system.
❺ All documents referenced by notation throughout the file should be described in the `fileHeader` as `reference` elements.

⑥   All modifications made to the contents of this file should be listed in the `fileHeader` as `modificationRecord` sub-elements for easy reference by later `modificationRef` 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 and unique `varID` identifier (used for linking inputs, functions and outputs), 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. 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. Another optional sub-element, `uncertainty`, captures the statistical properties of a (normally constant) parameter.

Other optional sub-elements are provided to identify inputs, disturbances, and simulation control parameters, as well as the ability to identify a variable as a state or state derivative for linear model purposes.

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}
    (
        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 MathML-2 DTD}
    (isInput | isControl | isDisturbance)?
    isState?
    isStateDeriv?
    isOutput?
    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 internal identifier that is unique within the file.
units	The units-of-measure for the signal, using the AIAA standard units convention [AIAA10].

axisSystem	An optional indicator of the axis system (body, inertial, etc.) in which the signal is measured. See [AIAA10] or Section B-6.5 [49] below for recommended practice for nomenclature.
sign	An optional indicator of which direction is considered positive (+RWD, +UP, etc.). See [AIAA10] or the section on Section B-6.5 [49] 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-2 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].
	A MathML-2 calculation can include both constants (using the content numeric cn element) and references to other variables internal to the parent DAVEfunc description. The variables (which can include the output, or dependent variable of a function table) are identified using its varID attribute string in the appropriate MathML content identifier (ci) element of the expression.
	Examples of MathML expressions appear later in this reference.
isInput	This optional element, if present, signifies that this variable is an input to the model, such as a pilot inceptor deflection or Mach number. Useful for linear model extraction tools. It must not be the result of a calculation or be cited as the dependent variable of a function.
isControl	This optional element, if present, signifies that this variable is a simulation control parameter, such as a trim flag or simulation time step measurement. Simulation control parameters should have no influence on the dynamic

behavior of the model and should be ignored by a linear model extraction tool.

isDisturbance This optional element, if present, signifies that this variable represents an external disturbance input to the model; this is useful for linear model extraction tools to partition this input separately from the other model inputs.

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 as Annex A to [AIAA10]. 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.

Example B-2. An example of two `variableDef` elements defining input signals

In this example , two input variables are defined: XMACH and DBFLL. These two variables are inputs to a table lookup function shown in example B-11 [35] below.

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


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

 <variableDef name="mach"❶ varID="XMACH"❷ units="nd" symbol="M">
   <description>
       Mach number (dimensionless)
   </description>
   <isInput/>❸
   <isStdAIAA/>❹
 </variableDef>
   .
   .
   .
 <variableDef name="dbfll" varID="DBFLL" units="deg"❺ sign="TED"❻
              symbol="&#x3B4;bfll"❼>
   <description>
       Lower left body flap deflection, deg, positive trailing-edge-down (so deflections are
       always zero or positive).
   </description>
```

```
        <isInput/>
    </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 variable names.

❷ The `varID` attribute is intended for the processing application to read. This is an internal identifier that must be unique within this model.

❸ The optional `isInput` attribute indicates this variable should be treated as an input to the model for model hierarchy and linear model extraction (for example).

❹ 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 Annex A of [AIAA10] or subsequent standards document

❺ The mandatory `units` attribute describes the units of measure of the variable. See Section B-6.5.6 [50] below for a recommended list of units-of-measure abbreviations.

❻ 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 [50] below for a recommended list of sign abbreviations.

❼ 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 definition example

This DAVE-ML excerpt defines CLBFLLO which is the dependent variable (output) from a table lookup function (shown later in Example 11 [35]). It is subsequently used in the calculation of the lower-left body flap lift coefficient (shown in Example 4 [23]).

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

<!-- PRELIMINARY BUILDUP EQUATIONS -->

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

<!--    table output signal   -->
  <variableDef name="CLdbfll_0" varID="CLBFLL0" units="nd">
    <description>
        Output of CLBFLL0 function; lift force contribution of
        lower left body flap deflection due to alpha^0 (constant
        term).
    </description>
  </variableDef>
```

Since this signal is not flagged as an input, control, disturbance or output, this variable is an intermediate signal local to this model.

Example B-4. A more complete `variableDef` example with a `calculation` element

In this example, the local variable CLBFLL is defined as a calculated quantity, based on several other input or local variables including the CLBFLL0 function output variable defined in the previous example [23]. 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="nd">
    <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>  ❹
          <apply>
            <times/>
            <ci>ALP</ci>
            <apply>
              <plus/>
              <ci>CLBFLL1</ci>
              <apply>
                <times/>
                <ci>ALP</ci>
                <apply>
                  <plus/>
                  <ci>CLBFLL2</ci>
                  <apply>  ❺
                    <times/>
                    <ci>ALP</ci>
                    <ci>CLBFLL3</ci>
                  </apply> <!--                a*c3   -->  ❻
                </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 The content identifier (`ci`) MathML-2 element gives the `varID` of the previously defined variables used in this equation; this variable represents the output of the `CLBFLL0` function found in Example 11 [35] that is captured in the `CLBFLL0` variable defined in Example 3 [23]. The other `ci` elements are not defined in this manual but are defined in the full model.

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

❻ 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. Another example of an output variable based on a `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="nd" 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> ❷
            <ci>CLBFLR</ci>
            <ci>CLWFL</ci>
            <ci>CLWFR</ci>
            <ci>CLRUD</ci>
            <ci>CLGE</ci>
            <ci>CLLG</ci>
          </apply>
        </math>
      </calculation>
      <isOutput/> ❸
    </variableDef>
```

❶   Here `apply` simply sums the value of these variables, referenced by their `varID`s.
❷   This `ci` element refers to the lower left body flag lift contribution calculated in the previous example [23].
❸   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 function extension to the default MathML-2 function set

In this excerpt, we demonstrate a means to encode a math function, *atan2*, that is not available in the default MathML-2 function set. 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-2 `csymbol` element, specifically provided to allow extension of MathML-2 for cases such as this.

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

   <variableDef name="Wind vector roll angle" varID="PHI" units="rad">
     <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.org/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 aerodynamics data book [NAA64].

❷   The `csymbol` element is provided by MathML-2 as a means to extend the function set of MathML-2. An extension for *atan2* is the only function defined at present but others may be added to the set in the future. Note the specific URI that uniquely identifies this function; it is also the URL (web 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- or white space-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- or white space-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- or white space-separated breakpoints}
```

`breakpointDef` attributes:

bpID	An internal reference 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 [50] below.

`breakpointDef` sub-elements:

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

Example B-7. Two `breakpointDef` examples in a DAVE-ML model excerpt

As an example, two breakpoint sets are defined which are used in the `function` element given below (example B-11 [35]). 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 [28].

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


  <breakpointDef name="mach" bpID="XMACH1_PTS" units="nd">  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="deg">     ❸
    <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- or white space-separated list and must be in monotonically increasing numerical order.

❸   This breakpoint set defines the breakpoints for lower body flap deflection.

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 each intersection of a set of specified independent input values. The coordinates along each dimension are defined in separate `breakpointDef` elements that are referenced within this element by `bpRefs`, one for each dimension.

The data contained within the data table definition are a comma- or white space-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 2D 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 (8 x 10) comma- or white space-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 [49] below.

An optional `uncertainty` element may be provided that represents the statistical variation in the values presented. See Section B-6.4 [43] 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+ : bpID
    uncertainty? : effect
        (normalPDF : numSigmas | uniformPDF)
    dataTable
        {character data of comma- or white space-separated table values}
```

`griddedTableDef` attributes:

gtID	An internal reference 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 of the table's output signal. See Section B-6.5.6 [50] 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 `bpRefs` is important; see the text above.
uncertainty	This optional element, if present, describes the uncertainty of this parameter. See Section B-6.4 [43] 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- or white space-separated list, representing an unraveled multi-dimensional table. Parsing this list and storing it in the appropriate array representation is up to the implementer. By convention, the last breakpoint value increases most rapidly.

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

This nonlinear function table is used by a subsequent `function` in Example 11 [35] to specify an output value based on two input 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 lower-right 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 good practice for human readers

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

❸ Descriptions make for good practice 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 `bpRefs` 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 practice.

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+ :
        {coordinate/value sets as character data}
```

`ungriddedTableDef` attributes:

utID	An internal reference 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 its input space. This element includes one coordinate for each function input variable. Parsing this information into a usable interpolative function is up to the implementer. By convention, the coordinates are listed in the same order that they appear in the parent function.

Example B-9. An excerpt showing an `ungriddedTableDef` element, encoding the data depicted in Figure 2 [31].

This 2D 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 does not 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="nd">
  <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 2D table such as this one, data points give two columns of independent breakpoint values and a third column of function values 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.
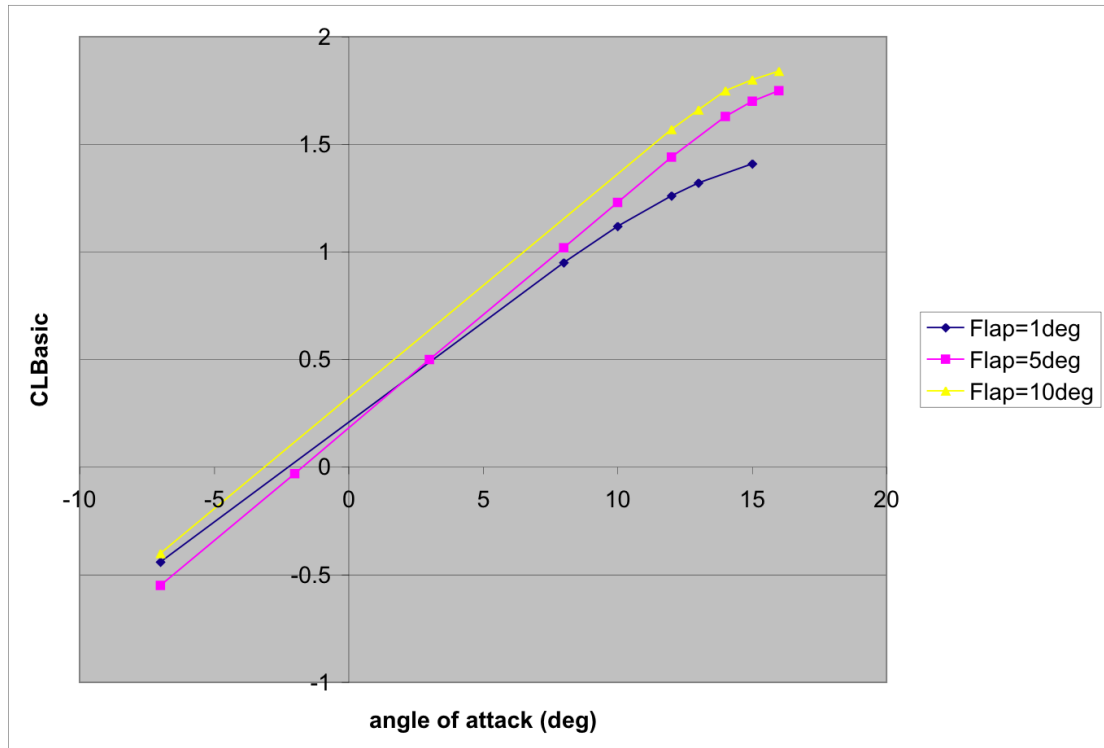
Figure B-2. The 2D lift function given in Example 9 [30]

Example B-10. An excerpt from a sample aerodynamics model giving an example of a 3D `ungriddedTableDef` element, encoding the data shown in Figure 3 [33].

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

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

<ungriddedTableDef name="yawMomentCoefficientTable1" units="nd"
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  -0.0093307</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 labeled 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 parent function definition(s) which links all independent (input) variables and the dependent (output) variable to the function.
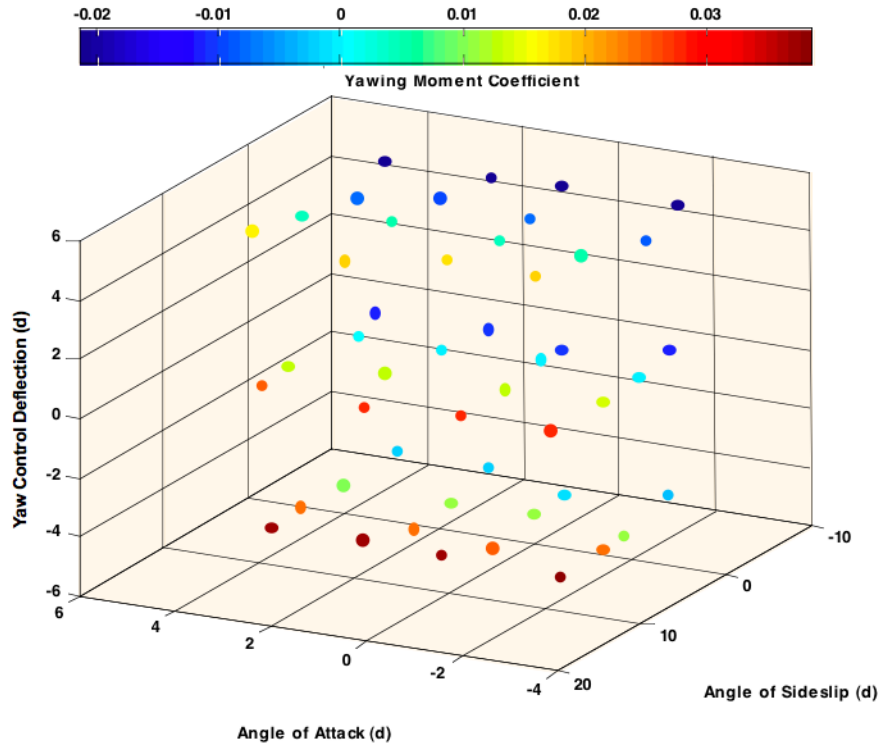
Figure B-3. The 3D function given in the previous example

B-6.2.6. The function definition element

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

```
function : name
    description? :
        {text description of the function}
    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, she can express a linearly interpolated functions by specifying the independent (breakpoint) value sets as one or more `independentVarPts` which are comma- or white space-separated, monotonically increasing floating-point coordinate values corresponding to the `dependentVarPts`. 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` set 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 to use a linear spline interpolation between points. The performance of interpolation of various orders is left up to the processing application. See Section B-6.3 [40].

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- or white space-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 refers to separately defined `variableDef`s. The order of specification is important and must match the order in which breakpoint sets 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 default expectation would be a linear spline interpolation between points. The performance of interpolation of various orders is left up to the implementer. See Section B-6.3 [40].
dependentVarRef	A single `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.
ungriddedTableRef	If not using a simple function table, the author may use this element to point to separately specified `ungriddedTableDef` element.

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. See Section B-6.3 [40].

❷　The dependent variable (identified as `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` with an internal table

In this example, the function `CLRUD0` returns, in the variable `CLRUD0`, the value of function `CLRUD0_fn` represented by gridded `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 the `CLRUD0` string for both the function `name` attribute and as the `varID` for the dependent (output) variable reference does not interfere (although they are confusing); `name` is 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"> ❷
    <description>
        Rudder contribution to lift coefficient,
        polynomial multiplier for constant term.
    </description>
    <provenance> ❸
      <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"/> ❹

    <functionDefn name="CLRUD0_fn">
      <griddedTableDef name="CLRUD0_table"> ❺
        <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 `name` attribute is used for documentation purposes, not for internal variable linkage.
❸   The `provenance` element is required by the AIAA standard [AIAA10].
❹   The `varID` attribute is used to link the output of this function with a previously defined variable as given in Example 3 [23].
❺   This example has an embedded gridded table.

Example B-13. A simple 1D `function`

At the other end of the spectrum, a simple 1D nonlinear function can be defined with no reuse, as shown below; however, multiple-dimension functions are supported by adding additional `independentVarPts` definitions.

```
<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+
                varID
                signalValue
        checkOutputs
            signal+
                signalName
                signalUnits
                signalValue
                tol
```

`checkData` sub-elements:

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

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>deg</signalUnits>
        <signalValue>   5.000</signalValue>
      </signal>
       .
       .        (similar input signals omitted)
       .
      <signal>
        <signalName>delta elevator</signalName>
        <signalUnits>deg</signalUnits>
        <signalValue>   0.000</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs> ❹
      <signal>  ❺
        <signalName>CX</signalName>
        <signalUnits>nd</signalUnits>
        <signalValue>-0.00400000000000</signalValue>
        <tol>0.000001</tol>
      </signal>
       .
       .        (similar output signals omitted)
       .
    </checkOutputs>
  </staticShot>
  <staticShot name="Positive pitch rate"> ❻
    <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 `signal` elements are usually given; taken together, these scalar `signals`s represent the check-case input "vector."

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

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

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

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

This example shows another check-case; this one includes intermediate values as an aide to debugging a new implementation.

```
<checkData>
   <( provenance | provenanceRef )?>
   <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>f_s</signalUnits>\
         <signalValue> 300.000</signalValue>
       </signal>
       <signal>
         <signalName>angleOfAttack</signalName>
         <signalUnits>deg</signalUnits>
         <signalValue>  16.200</signalValue>
       </signal>
        .
        .        (similar input values omitted)
        .
       <signal>
         <signalName>bodyPositionOfCmWrtMrc</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>aeroBodyForceCoefficient_X</signalName>
         <signalValue> 0.04794994533333</signalValue>
         <signalUnits>nd</signalUnits>
         <tol>0.000001</tol>
       </signal>
       <signal>
         <signalName>aeroBodyForceCoefficient_Z</signalName>
         <signalValue>-0.72934852554344</signalValue>
         <signalUnits>nd</signalUnits>
```

```
        <tol>0.000001</tol>
      </signal>
      <signal>
        <signalName>aeroBodyMomentCoefficient_Pitch</signalName>
        <signalValue>-0.10638585796503</signalValue>
        <signalUnits>nd</signalUnits>
        <tol>0.000001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
</checkData>
```

❶   The `internalValues` element, if present, usually is a list of all model-defined internal
     variable values. This is normally not required for a model exchange, but such information
     is very useful 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 nonlinear function tables by
use of the `interpolate` attribute of the `independentVarPts` and `independentVarRef`
elements. This attribute, combined with the `extrapolate` flag, provides several different ways
of realizing the intermediate values of the function when not at one of the specified intersections
of independent values.

Possible values for the `interpolate` attribute are:

discrete	Output uses value associated with nearest breakpoint
floor	Output uses value associated with next (numerically higher) breakpoint
ceiling	Output uses value associated with last (numerically lower) breakpoint
linear (default)	Output is linearly interpolated between breakpoints
quadraticSpline	Output follows a quadratic spline fit through values associated with two nearby breakpoints
cubicSpline	Output follows a cubic spline fit through values associated with three nearby breakpoints

Possible values for the `extrapolate` attribute are:

neither (default)	Output is held constant at value associated with closest end of breakpoints if the input value is outside the limits of the associated breakpoint set
min	Output follows extrapolated values of function if the input is below the minimum breakpoint value
max	Output follows extrapolated values of function if the input is above maximum breakpoint value
both	Output follows extrapolated values of function if the input is outside the limits of the associated breakpoint set

Implementation of the specific interpolation algorithm is left up to the implementer. One reference is the Wikipedia entry on interpolation [wiki01].

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) is recommended when the `extrapolate` attribute is `none`. When the `extrapolate` attribute is `both`, a clamped cubic spline that matches the extrapolated slope of the last two data points is suggested.

- For the discrete interpolation values (`discrete`, `ceiling`, or `floor`), the value of the `extrapolate` attribute is meaningless.

For discrete interpolation,

- `discrete` implies the change between output values occurs midway between independent breakpoint values, as shown in the top plot of Figure 4 [42].

- `ceiling` means the output takes on the value of the next-higher dependent variable breakpoint as soon as each independent breakpoint value is passed (assuming the input value is increasing) as shown in the middle plot of Figure 4 [42].

- `floor` means the output retains the value of the last dependent variable breakpoint until the next independent breakpoint value is reached (assuming the input value is increasing) as shown in the bottom plot of Figure 4 [42].

The default value for `interpolate` is `linear`. The default value for `extrapolate` is `neither`.

Figures 4 [42] and 5 [43] below give nine different examples for a 1D table whose independent values are [1, 3, 4, 6, 7.5] with dependent values of [2, 6, 5, 7, 1.5].
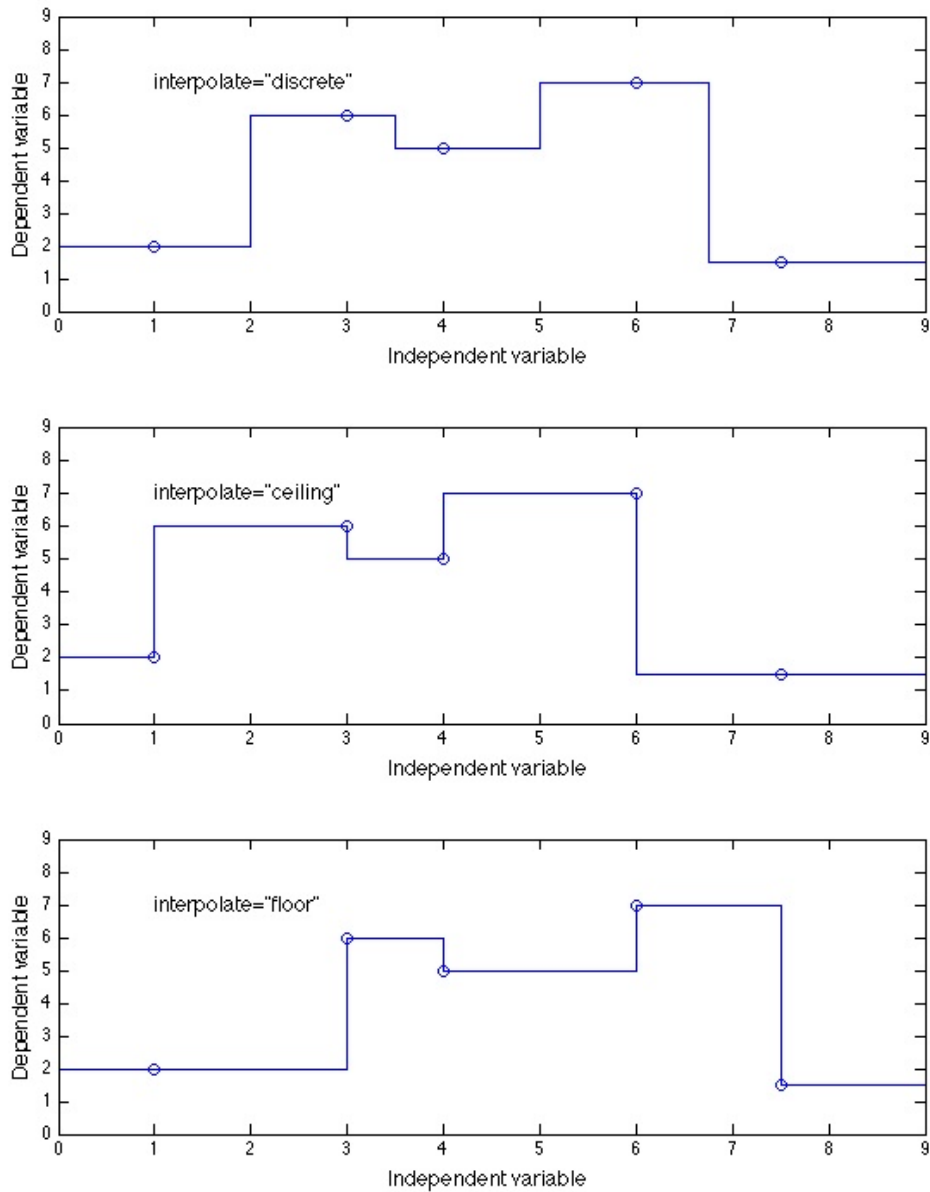
Figure B-4. Example of the three discrete enumeration values of `interpolate` attribute of the `independentVarPts` and `independentVarRef` elements for a 1D function table.
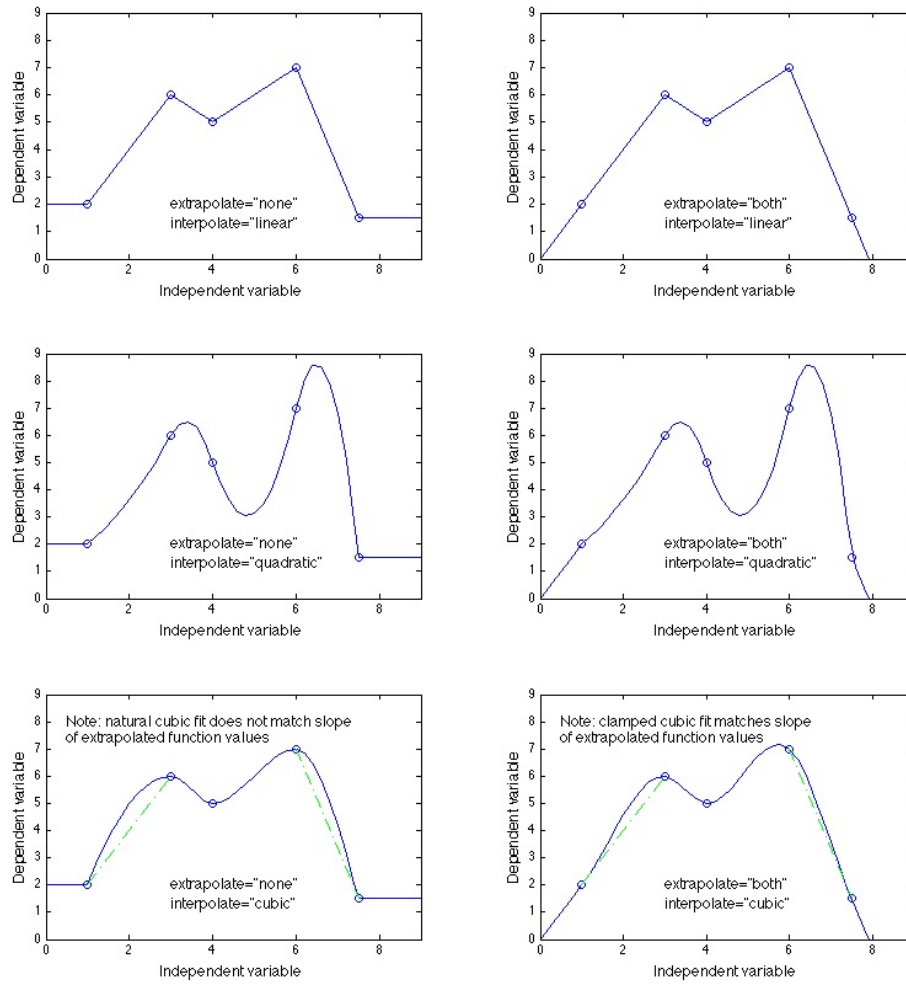
Figure B-5. Examples of the three higher-order interpolation methods showing the effect of the `interpolate` attribute of the `independentVarPts` and `independentVarRef` elements for a 1D function table.

B-6.4. Statistical information encoding

Statistical measures of variation of certain parameters and functions can be embedded in a DAVE-ML model in several ways. This information is captured in an `uncertainty` element, which can be referenced by `variableDef`, `griddedTableDef` and `ungriddedTableDef` elements. For maximum modeling flexibility, it is possible to add uncertainty to the independent value arguments to a function or calculation, to the output of a function itself, as well as to any output signal. Applying uncertainty at more than one location in a calculation change is probably not a good practice, however.

Details on providing the random values for uncertainties is left to the implementer.

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 does not have a dependency direction (i.e., the statistical uncertainty of one parameter is specified in terms of the other parameter, therefore the calculation order does not 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 { scalar value representing the one, two or three sigma bound }:
        (correlatesWith* : varID |
         correlation* : varID, corrCoef )
  OR
    uniformPDF
        bounds { one or two scalar values for abs. or min/max bounds }
```

uncertainty attributes:

effect   Indicates, by choice of four enumerated values, how the uncertainty is modeled: as an additive, multiplicative, or percentage variation from the nominal value, or a 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: multiplicative and absolute bounds do not 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 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="nd" 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 a 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 nonlinear 1D function and the uncertainty is applied to the output of the table.

```
<DAVEfunc>
    <fileHeader>
        .
        .
        .
    </fileHeader>
    <variableDef name="angleOfAttack" varID="Alpha_deg" units="deg">
        <isStdAIAA/>
    </variableDef>
    <variableDef name="Cm_u" varID="Cm_u" units="nd">
        <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 1D 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 nonlinear 1D function and the uncertainty is applied to the output of the table.

```
<DAVEfunc>
    <fileHeader>
      .
      .
      .
    </fileHeader>
    <variableDef name="angleOfAttack" varID="Alpha_deg" units="deg">
        <isStdAIAA/>
    </variableDef>
    <variableDef name="Cm_u" varID="Cm_u" units="nd">
        <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 1D Cm_u function table description).

❷   This function gives the nominal values of Cm_u as a 1D function of angle-of-attack (alpha).

Example B-19. A 1D point-by-point, Gaussian distribution function

In this example, a Gaussian (normal) distribution function is applied to *each* point in a 1D 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="deg">
        <isStdAIAA/>
    </variableDef>
    <variableDef name="Cm_u" varID="Cm_u" units="nd">
        <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 lists three-sigma bounds of each point 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="deg">
        <isStdAIAA/>
    </variableDef>
    <variableDef name="CL_u" varID="CL_u" units="nd">
        <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="nd">
        <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 that varies with angle-of-attack according to a nonlinear 1D table (given in the "Nominal CL" table defined in this example). 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 lift coefficient correlates directly with the variation in pitching moment coefficient.

❸ Pitching-moment coefficient has a nominal value that varies as a function of angle-of-attack, according to a nonlinear 1D table (given in the "Nominal Cm" table defined in this example). 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 conventions 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 such 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 the data in the correct sequence in the model realization is left up to the implementer.

Figure 6 [49] below shows how a 3D table is represented as an unraveled list of points.
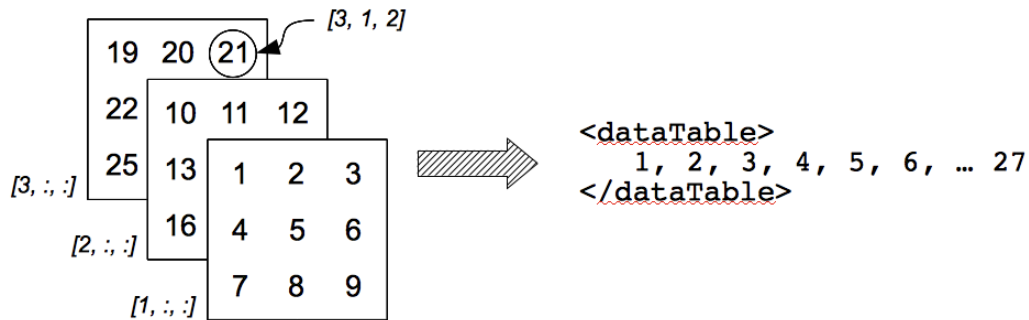
Figure B-6. An unraveled list compared to the original 3D table

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 dynamics subsystems

It is recommended that a vehicle's flight dynamics 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 attributes and creation date elements, is numerically in the order YYYY-MM-DD. Thus, July 15,

2003 is given as 2003-07-15. This formatting convention conforms to the ISO-8601 standard for representing dates. [ISO8601]

B-6.5.5. Common sign convention notation

A convention for indicating positive sign conventions for measurements is included in Annex A of the draft AIAA Flight Dynamics Model Exchange Standard [AIAA10]. These acronyms, if applicable, should be used whenever possible to enhance communications.

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 non-dimensional) 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 either set of units (English or ISO). For greater re-usability, it is recommended that the set of measurements listed in the *AIAA Flight Dynamics Model Exchange Standard* [AIAA10] (of which this document is a part) be used. The Standard recommends how to encode powers of units (f_s2 for ft/sec^2, 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 XML specification for Names [W3C-XML]. They must start with a character or underbar, may not start with "xml" or "XML," and may not contain colons, among other restrictions. In addition, the identifiers must be unique within a single DAVE-ML file. See the XML [W3C-XML] [125] for more information regarding valid XML Names.

B-6.5.8. DAVE-ML Namespace

The XML standard allows for namespace domains, in which element names belong to either the empty (null) namespace or to 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, in order that uniqueness is guaranteed.

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

```
<DAVEfunc xmlns="http://daveml.org/2010/DAVEML">
```

This will allow DAVE-ML models to be embedded in other XML documents without confusion. Note that this general URI 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-2 elements are normally defined in a MathML namespace, so any calculation defined using MathML-2 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

- Support for vector and matrix variables and math operations.

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

B-8. Element references and descriptions

This section lists the XML tags, or elements, that make up the DAVE-ML grammar. They are listed alphabetically in two sub-sections; the first is a short description of each element; the second sub-section gives details on the element, attributes, and sub-elements.

B-8.1. Alphabetical list of elements

address	Street address or other contact information of an author [Deprecated.]
author	Gives name and contact information for originating party of the associated data
bounds	Describes limits or standard deviations of statistical uncertainties
bpRef	Reference to a breakpoint definition
bpVals	String of white space- or comma-separated values of breakpoints
breakpointDef	Defines breakpoint sets to be used in model
breakpointRefs	A list of breakpoint reference (bpRefs)
calculation	Used to delimit a MathML v2 calculation
checkData	Gives verification data for encoded model
checkInputs	Lists input values for check case
checkOutputs	Lists output values for check case
confidenceBound	Defines the confidence in a function [Deprecated]
contactInfo	Provides multiple contact information associated with an author or agency
correlatesWith	Identifies other functions or variables whose uncertainty correlates with our random value
correlation	Indicates the linear correlation of this function's or variable's randomness with a previously computed random variable
creationDate	Gives date of creation of entity
dataPoint	Defines each point of an ungridded table
dataTable	Lists the values of a table of function or uncertainty data
DAVEfunc	Root level element
dependentVarPts	Defines output breakpoint values
dependentVarRef	Identifies the signal to be associated with the output of a function

description	Verbal description of an entity
documentRef	Reference to an external document
extraDocRef	Allows multiple documents to be associated with a single modification event
fileCreationDate	Gives date of creation of entity [Deprecated]
fileHeader	States source and purpose of file
fileVersion	Indicates the version of the document
function	Defines a function by combining independent variables, breakpoints, and tables
functionCreationDate	Date of creation of a function table [Deprecated]
functionDefn	Defines a function by associating a table with other information
griddedTable	Definition of a gridded table; associates breakpoint data with table data [Deprecated]
griddedTableDef	Defines an orthogonally gridded table of data points
griddedTableRef	Reference to a gridded table definition
independentVarPts	Simple definition of independent breakpoints
independentVarRef	References a predefined signal as an input to a function
internalValues	A dump of internal model values for debugging check-cases
isControl	Flag to identify a model control parameter
isDisturbance	Flag to identify a model disturbance input
isInput	Flag to identify a model input variable
isOutput	Flag to identify non-obvious output signals from model
isState	Flag to identify a state variable within a dynamic model
isStateDeriv	Flag to identify a state derivative within a dynamic model
isStdAIAA	Flag to identify standard AIAA simulation variable
modificationRecord	To associate a reference single letter with a modification event
modificationRef	Reference to associated modification information
normalPDF	Defines a normal (Gaussian) probability density function
provenance	Describes origin or history of the associated information
provenanceRef	References a previously defined data provenance description

reference	Describes an external document
signal	Documents an internal DAVE-ML signal (value, units, etc.)
signalID	Gives the internal identifier of a varDef [Deprecated]
signalName	Gives the external name of an input or output signal
signalUnits	Gives the unit-of-measure of an input or output variable
signalValue	Gives the value of a check-case signal/variable
staticShot	Used to check the validity of the model once instantiated by the receiving facility or tool.
tol	Specifies the tolerance of value matching for model verification
uncertainty	Describes statistical uncertainty bounds and any correlations for a parameter or function table.
ungriddedTable	Definition of an ungridded set of function data [Deprecated]
ungriddedTableDef	Defines a table of data, each with independent coordinates
ungriddedTableRef	Reference to an ungridded table
uniformPDF	Defines a uniform (constant) probability density function
variableDef	Defines signals used in DAVE-ML model
variableRef	Reference to a variable definition
varID	Gives the internal identifier of a varDef

B-8.2. Element descriptions

This section lists each element in detail, giving the name, content model, attributes, possible parent elements, allowable children elements, and any future plans for the element (such as deprecation).

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) (deprecated) The eXtensible Name Service identifier for the author.
email	(optional) (deprecated) The e-mail address for the primary author.

Description

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

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' sub-element 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 :
      (#PCDATA | 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 private dataTable, or a variableRef. 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, either previously defined or defined in-line with this element. It does not make sense to have a dataTable cited if this bounds element is associated with anything other than an identically shaped function table.

Possible parents

```
normalPDF
uniformPDF
```

Allowable children

```
dataTable
variableDef
variableRef
```

bpRef

>   bpRef — Reference to a breakpoint definition

Content model

```
bpRef : bpID
      EMPTY
```

Attributes

>   `bpID`   The internal identifier for a breakpoint set definition.

Description

>   The bpRef element provides references to a previously-defined breakpoint set so breakpoints can be defined separately from, and reused by, several data tables.

Possible parents

>   `breakpointRefs`

Allowable children

>   NONE

bpVals

bpVals — String of white space- or comma-separated values of breakpoints

Content model

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

Attributes

NONE

Description

bpVals is a set of breakpoints (i.e., 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 2D 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 lists gridded table breakpoints. Since these are separate from function data they may be reused.

Possible parents

DAVEfunc

Allowable children

```
description
bpVals
```

breakpointRefs

       breakpointRefs — A list of breakpoint reference (bpRefs)

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 [12]
```

Attributes

>NONE

Description

>The calculation element is MathML 2 content markup describing how the signal is calculated. The calculation may include both constants and variables; other variables are included by using their varID string in a MathML content identifier (ci) element.

Possible parents

>`variableDef`

Allowable children

>`math` [12]

checkData

>checkData — Gives verification data for encoded model

Content model

```
checkData :
    (provenance | provenanceRef)?
    staticShot+
```

Attributes

>NONE

Description

>This top-level element is the place-holder 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 compatibility.

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 place-holder 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's 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's or variable's immediate value will be used to calculate another function's or variable's value.

Possible parents

normalPDF

Allowable children

NONE

correlation

> correlation — Indicates the linear correlation of this function's 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)'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 an ungridded table, the interpreting 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. The table data point values are specified as comma- or white space-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 un-prefixed elements is this DTD.

Description

Root element is DAVEfunc, composed of a file header element followed by one or more variable definitions and zero or more breakpoint 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 white space-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 do not 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 textual description of an entity. The full UNICODE character set is supported by XML but may not be available in all processing applications.

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) (deprecated) The internal identifier of a reference definition element.

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. Its use is now deprecated in favor of the simpler 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 and a creation date; optional content includes version indicator, description, references, and modification 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
    )
    OR
    (
        independentVarRef+
        dependentVarRef
        functionDefn
    )
```

Attributes

>`name`   The name of this function.

Description

>Each function has optional description, optional provenance, and either a simple input/output table 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. Its use is now deprecated in favor of the simpler 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. Required if table is to be reused by another function or is defined outside of a function.

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 a 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 white space- 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 the 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 do not 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 (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 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 check-cases

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`

isControl

> isControl — Flag to identify a model control parameter

Content model

```
isControl :
     EMPTY
```

Attributes

> NONE

Description

> The presence of an isControl element indicates that this signal is a simulation control parameter used to vary the operation of the model, e.g. the time step size. Such parameters should be ignored when performing linear model extraction (for example) and should not significantly modify the dynamic behavior of the model.

Possible parents

```
variableDef
```

Allowable children

> NONE

isDisturbance

isDisturbance — Flag to identify a model disturbance input

Content model

```
isDisturbance :
    EMPTY
```

Attributes

NONE

Description

The presence of an isDisturbance element indicates that this signal is an external disturbance input to the model and can be ignored when performing linear model extraction (for example). Such parameters should not significantly modify the nominal dynamic behavior of the model.

Possible parents

```
variableDef
```

Allowable children

NONE

isInput

> isInput — Flag to identify a model input variable

Content model

```
isInput :
    EMPTY
```

Attributes

> NONE

Description

> The presence of an isInput element indicates that this variable is an input signal to the model.

Possible parents

> `variableDef`

Allowable children

> NONE

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 may assume a signal defined with a calculation and used subsequently in the model is only 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 continuous 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 presence 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 continuous 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 that this variable is one of the standard AIAA variable names which should be recognizable exterior to this module (e.g. AngleOfAttack_deg). This flag should assist importing tools in determining 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 sub-element references other functions or variables that have a linear correlation to the current parameter or function. The correlation sub-element 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 defined here 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 a previously defined provenance.

Description

When the provenance of a set of several data is identical, the first provenance element should be given a provID and referenced by later provenanceRef elements.

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
    )
    OR
    (
        (varID | signalID)
    )
    signalValue
    tol?
```

Attributes

NONE

Description

This element is used to document the name, ID, value, tolerance, and units of measure for check-cases. When used with checkInputs or checkOutputs, the signalName sub-element must be present (since check cases are viewed from "outside" the model); when used in an internalValues element, the varID sub-element 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 sub-element 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 check-case 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 for the table. Required if table is to be reused by another function or is defined outside of a function.

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?
     (isInput | isControl | isDisturbance)?
     isState?
     isStateDeriv?
     isOutput?
     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 (i.e., 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 sub-element.

Possible parents

```
DAVEfunc
bounds
```

Allowable children

```
description
provenance
provenanceRef
calculation
isInput
```

```
isControl
isDisturbance
isState
isStateDeriv
isOutput
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

[AIAA10] : American Institute of Aeronautics and Astronautics : *American National Standard: Flight Dynamics Model Exchange Standard (Draft) [http://info.aiaa.org/Standards/AS/DAVE]*. BSR/AIAA S-119-201x

[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.

[Hildreth94] : Hildreth, B. L.: *A Process for the Development of Simulation Standards*. AIAA 94-3430, presented at the AIAA Flight Simulation Technologies Conference, August 1994, Baltimore, Maryland.

[Hildreth98] : Hildreth, Bruce L.: *The Draft AIAA Flight Mechanics Modeling Standard: An Opportunity for Industry Feedback*. AIAA 98-4576, presented at the AIAA Modeling and Simulation Technologies Conference, August 1998, Boston, Massachusetts.

[Hildreth08] : Hildreth, Bruce; Linse, Dennis J.; and Dicola, 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

Index

A

B

D

E