Generating Aircraft Trajectories encoded with the Aircraft Intent Description Language using the Modeling Language Modelica

Michael Hardt and Robert Höpler

Abstract The integration of Unmanned Aircraft Systems (UAS) into civil, non-segregated airspace remains an open problem for which many distinct proposals have been made. One of the fundamental ingredients for a viable system is the facilitation of the communication of an aircraft's trajectory among the airspace stakeholders so as to introduce a necessary degree of predictability into the system. This is essential for the coordination of aircraft in a densely populated airspace and, in particular, to be prepared for potential time-critical contingencies. The Aircraft Intent Description Language (AIDL) has been proposed in the past to efficiently represent an aircraft's trajectory. It consists of describing the aircraft's flight intent into several parallel sequences of instructions, which are intuitive and easily interpretable, and can be translated into a high resolution flight trajectory which takes into account the aircraft performance model and environmental conditions. The benefits of this representation is that its information content is minimal suitable for reduced bandwidth communications, and it is independent of the aircraft's performance model and environmental conditions, both of which may vary over time, or for which certain stakeholders may later dispose of improved information. Nevertheless, a potential hindrance in the implementation of this trajectory representation consists in the trajectory reconstruction process. A system of differential-algebraic equations (DAE) of possibly high index must be solved. Numerical tools to date have required the consideration of numerous special use cases such as to condition the numerical solution problem accordingly. This paper presents a generic approach by which this trajectory reconstruction process is performed making use of the object-oriented modeling language Modelica and associated tools. Efficient embeddable code can

Michael Hardt
Boeing Research & Technology - Europe, Avenida Sur del Aeropuerto de Barajas 38, Floor 4, Madrid, Spain, e-mail: michael.w.hardt@boeing.com

Robert Höpler
Campus Burghausen, Technische Hochschule Rosenheim, Marktler Strasse 50, 84489 Burghausen, Germany
e-mail: robert.hoepler@th-rosenheim.de

then be generated from this environment for UAS. It is considered that these techniques can be an important enabler to permit a wide sector to fully take advantage of the numerous advantages that this description language offers.

1 UAS Prediction with Trajectory Management

The integration of low altitude UAS operations into civil and military airspace presents a variety of issues and challenges that remain unresolved [3]. The sUAS (small UAS less than 20 kg) fleet alone between hobbyists and commercial enterprises is projected to number as much as 6 million in the U.S. by 2021. The airspace occupied by these aircraft is generally considered to be below 400 feet above ground level. This same airspace is expected to be the emerging sector of flying air taxis, which lie outside the sUAS category. To deal with the inevitable congestion within this airspace, new, advanced techniques of UAS Traffic Management (UTM) are being investigated and tested. The concept of UTM is based upon information sharing and exchange between the relevant stakeholders: aircrafts, operators, and air traffic authority. Future UTM operations shall require all operators to coordinate and share their flight intent with each other to achieve safe operations. Though there are still no existing standards regarding the format of the flight intent content, it is certain that the transmitted information must facilitate full trajectory computation thus permitting flight coordination and safety.

Future UTM concepts shall rely extensively upon automation [3]. A mature operational capability necessary to support a high-density airspace shall require fully autonomous planning, scheduling, separations, entry/exit airspace, interoperability, and contingency management. In particular, trajectory management is vital for the resolution of contingency management scenarios [2]. As is recognized in the latter work, the challenging issue of handling trajectories lies in the diversity of aircraft types, the innumerable system failure sources, and failure modes. Predictability is what is expected from trajectory management techniques which is essential for providing guarantees in terms of airspace restrictions, separation awareness, and especially in handling contingencies.

Some of the desired qualities in a trajectory computational framework are [2, 3]:

(A) *No burden on current system:* weak requirements on communication bandwidth and computational power

(B) *Cooperative and interoperable:* facilitate trajectory reconstruction and interpretation for all stakeholders

(C) *Performance and risk-based:* trajectory parameterization to satisfy performance and/or risk-based criteria

(D) *Efficient:* real-time capable on standard hardware

(E) *Scalability and sustainability:* easily expandable for new vehicle configurations, adaptation for use with other tools, varying degrees of model fidelity

(F) *Constraint information:* no-fly zones, weather, non-conforming flight, other restrictions

2 Aircraft Intent Description Language

The formal language known as Aircraft Intent Description Language (AIDL) [11, 6] was developed for serving as a standard, interoperable means of describing and exchanging predicted aircraft trajectories in Trajectory-Based Operations (TBO) for Air Traffic Management (ATM). The underlying framework satisfies the above desired attributes for a UTM system.

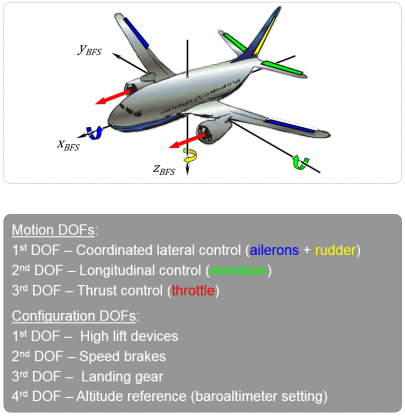

Fig. 1 Models forming the basis for a AIDL Trajectory Computation Framework

The formal language constructs representing aircraft intent imply constraints imposed upon the aircraft motion and configuration degrees of freedom (dof). A three-dimensional motion model is typically used for fixed-wing aircraft as shown in Figure 1, though the framework can be readily adapted for other systems.

Trajectory prediction is achieved by combining the AIDL instructions or aircraft intent rules with the flight dynamic model, which may vary slightly according to the aircraft class (fixed-wing, quadrotor, etc), the specific dynamic parameters of the dynamic model for the aircraft of interest, i.e. aircraft performance model, and a meteorological model describing the wind field and potentially other atmospheric characteristics. Once these elements are in place, by expressing aircraft intent according to the AIDL, it is ensured that each instance of aircraft intent defines a

unique trajectory. The advantage of this approach is that any stakeholder with interest in the aircraft's motion may reconstruct an aircraft's future flight trajectory with the minimal, compact information composing the AIDL instructions whereby the other elements consisting of the aircraft performance model and atmospheric model may be obtained by other means or even updated from independent estimation procedures.

Addressing the desired UTM framework attributes listed in Section 1, Table 1 summarized the benefits of employing AIDL for trajectory representation.

Table 1 AIDL UTM Attributes

(A)	*No Burden on System:* only flight intent rules need to be transmitted, the entire right side of Figure 1 can be preloaded for all available standard aircraft configurations
(B)	*Cooperative, interoperable:* intuitive language constructs that are human-readable
(C)	*Performance and risk-based:* motion dof constraints can be parameterized according to performance and/or risk-based criteria
(D)	*Efficient:* real-time trajectory generation tools exist for solving the differential-algebraic equation resulting from the AIDL Trajectory Computation Framework
(E)	*Scalable, sustainable:* the trajectory computational is modular; hence, vehicle configurations are easily substituted with others, also varying degrees of model fidelity can be adapted
(F)	*Constraint amenable:* no-fly zones are typically incorporated directly into the flight intent, weather inputs into the dynamic guidance model are independent and modular with the advantage that they may be easily updated as better information becomes available

The subject of this paper lies principally in the efficiency aspect (D) of the computational framework. A generic methodology is presented (a) for symbolically expressing the equations of the multiple-phase differential-algebraic equation system including the hybrid (switching) behavior between the AIDL phases using the language Modelica, and (b) creating efficient code, and (c) solving the DAE efficiently and numerically by means of an associated Modelica tool.

AIDL is a domain-specific formal language (DSL) composed of an alphabet (set of "instructions" or atomic ways of describing aircraft behavior), a lexicon (set of rules that govern the legal/meaningful combination of elements from the alphabet) and a sequence control mechanism (set of "triggers" that switch behavioral changes upon reaching conditions). Table 3 lists the recognized AIDL *groups*, each of which contains a set of different *instructions*.

The AIDL alphabet contains the language instructions and their properties. AIDL instructions are composed of three members: the *effect*, which describes the use of a specific degree of freedom, the *execution interval*, which determines the length of time for which the instruction is active, and the *conditions*, which impose additional constraints on the resulting aircraft trajectory. Most intent instructions do not contain

any conditions, but they do always have associated an effect and trigger (which defines the execution interval).

The AIDL instruction effects capture the basic commands and guidance modes at the disposal of the Flight Management System (FMS) to direct the operation of the aircraft. They can be seen as the minimal indivisible pieces of information regarding the operation of the aircraft and represent the language symbols [5]. Table 2 lists 27 representative (non-configuration) instruction effects present in the AIDL alphabet, known as Σ_{AIDL}.

Table 2 AIDL alphabet Σ_{AIDL}

AIDL Alphabet - Σ_{AIDL}					
Effects					
#	**Keyword**	**Effect**	**#**	**Keyword**	**Effect Name**
1	SBA	Set Bank Angle	15	VSL	Vertical Speed Law
2	BAL	Bank Angle Law	16	HVS	Hold Vertical Speed
3	HBA	Hold Bank Angle	17	EL	Energy Law
4	OLBA	Open Loop Bank Angle	18	HE	Hold Energy
5	CL	Course Law	19	HSL	Horizontal Speed Law
6	HC	Hold Course	20	HHS	Hold Horizontal Speed
7	LPL	Lateral Path Law	21	SL	Speed Law
8	VPL	Vertical Path Law	22	HS	Hold Speed
9	AL	Altitude Law	23	TL	Time Law
10	HA	Hold Altitude	24	ST	Set Throttle Control
11	SPA	Set Path Angle	25	TCL	Throttle Control Law
12	PAL	Path Angle Law	26	HTC	Hold Throttle Control
13	HPA	Hold Path Angle	27	OLTC	Open Loop Throttle Control
14	OLPA	Open Loop Path Angle			

The Σ_{AIDL} instruction effects can be categorized into four groups according to the mode in which they influence the aircraft behavior.

1. **Set** instruction effects model *target modes* that represent the change of a given aspect of the aircraft motion or configuration from its initial value towards a target value
2. **Law** instruction effects model *advanced modes* of guiding and controlling an aircraft. They generally involve tracking a predefined path in one or more of the state variables.
3. **Hold** instruction effects model the *basic modes* of guiding or controlling an aircraft or setting its configuration. These basic modes act as simplified Law modes, in which once captured, the desired motion or configuration aspect is maintained constant by the coordinated action of the aircraft control and configuration settings.

4. **Open Loop** instruction effects model a flight command issued by the FMS that acts directly over the controls. Hence, they do not depend on the state variables, just the time.

The Σ_{AIDL} instruction effects can be classified into 12 different *groups* (not considering those which affect aircraft configuration) based on the *target variable* affected by the effect (refer to table 3 in which the color code used for each group coincides with the color code of its representative instructions). All effects with the same target variable belong to the same group. The target variable is the specific state or control (or combination of them) at which the instruction guidance or control mode is directed. Effects from the same group cannot be active at the same time (concurrent execution intervals) as they are incompatible in terms of solvability according to the analysis performed in [5].

Table 3 AIDL effect groups

Groups		
Code		
#	**Keyword**	**Group Name**
1	**LDC**	Lateral Directional Control
2	**LDG**	Lateral Directional Guidance
3	**LPG**	Lateral Positional Guidance
4	**VPG**	Vertical Positional Guidance
5	**AG**	Altitude Guidance
6	**PAC**	Path Angle Control
7	**VSG**	Vertical Speed Guidance
8	**EG**	Energy Guidance
9	**HSG**	Horizontal Speed Guidance
10	**SG**	Speed Guidance
11	**TG**	Time Guidance
12	**TC**	Throttle Control

The *specifiers* of target variables indicate concretely which state variable is being constrained. As previously mentioned, the instruction groups are defined by the collection of instructions / effects which share the same set of specifiers. For an instruction to be implemented, its corresponding specifier must also be indicated. The variable definitions as shown in the specifier list in Table 3 are listed in the following table:

Associated with each instruction is a trigger. Triggers are denoted as the conditions which indicate the end of an instruction. Once that condition is fulfilled, the instruction ends, and the next instruction in the given thread is executed. The trigger condition consists of an algebraic relation of a function of the aircraft state or time variable. Each trigger is also associated with a unique ID. Table 5 lists typical trigger dependencies which are compared with a polynomial function of time for a zero-crossing event.

Table 4 Specifier Flight Variables

Specifier Flight Variables		
Symbol	**Units**	**Variable Name**
χ_{TAS}	[rad]	aerodynamic heading angle
γ_{TAS}	[rad]	aerodynamic path angle
μ_{TAS}	[rad]	aerodynamic bank angle
χ	[rad]	bearing angle
γ	[rad]	path angle
μ	[rad]	bank angle
φ	[rad]	geodetic latitude angle
λ	[rad]	geodetic longitude angle
h	[m]	altitude above mean sea level
H_P	[m]	barometric altitude
$\dot{h}$	[m/s]	altitude rate
$\dot{H}_P$	[m/s]	barometric altitude rate
$v_{TAS}\cos(\gamma_{TAS})$	[m/s]	magnitude of true airspeed when projected onto horizontal plane in ned coordinates
$v_{GRD}\cos(\gamma)$	[m/s]	magnitude of ground speed when projected onto horizontal plane in ned coordinates
v_{TAS}	[m/s]	true airspeed magnitude
v_{CAS}	[m/s]	calibrated airspeed
v_{GRD}	[m/s]	ground speed magnitude
t	[sec]	elapsed time in current instruction
δ_T	[-]	throttle value [0–1]

Typically with fixed-wing aircraft, a 3-*dof* aircraft dynamic guidance model is sufficient for trajectory specification. It's expression in wind relative coordinates permit a convenient decoupling between the experienced longitudinal forces, and the lateral and vertical motion degrees of motion described by coordinated turns.

The fundamental interpretation of an AIDL instruction is the imposition of a mathematical constraint upon the equations of aircraft motion. Thus, in the case of fixed-wing aircraft, these three flight *dof* can then be constrained via the implementation of three parallel AIDL *threads*. Table 6 shows a further categorization of the Σ_{AIDL} groups into *profiles*. The allowed combinations of simultaneous AIDL profiles is illustrated in Table 7 when assigned to the Lateral, Longitudinal, and Propulsive *dof* / threads. As all three *dof* have been constrained, the calculated trajectory is unique. This is important in maximizing the predictability of an aircraft's trajectory simply by communicating its flight intent via its intended AIDL instructions.

The calculation of a trajectory given by a succession of AIDL instructions and triggers in the three parallel threads, as given in Table 7, implies the solvability for the numerical solution of a differential-algebraic equation (DAE), i.e. the 3-dof flight dynamic model coupled with the three AIDL constraints, whose index depends upon the nature of the currently active AIDL instructions. The AIDL formal language constructs are designed such that the accepted combinations of AIDL instructions permit the solvability of the DAE as described in detail in [5].

Table 5 Trigger Codes

Trigger Codes			
Code#	**Variable z**	**Units**	**Description**
0	ID	-	unique trigger identification
1	d	[m]	distance traveled since instruction begin
2	t	[s]	elapsed time since instruction begin
3	T	[s]	elapsed time since AIDL guidance begin
4	d_{LPL}	[m]	projected distance traveled along LPL since instruction begin
5	y	-	unknown trigger
10	v_{GRD}	[m/s]	ground speed magnitude
11	M	-	Mach number
12	v_{TAS}	[m/s]	true airspeed magnitude
13	v_{IAS}	[m/s]	indicated airspeed
14	v_{CAS}	[m/s]	calibrated airspeed
15	v_{EAS}	[m/s]	equivalent airspeed
20	h	[m]	geometric altitude
21	H_p	[m]	geopotential pressure altitude
22	H_i	[m]	geopotential indicated altitude
23	H	[m]	geopotential altitude
30	μ_{TAS}	[rad]	aerodynamic bank angle
40	δ_T	–	throttle
50	χ_{TAS}	[rad]	aerodynamic heading angle
51	χ	[rad]	true bearing angle
52	$\chi_{TAS,MAG}$	[rad]	magnetic heading angle
53	χ_{MAG}	[rad]	magnetic bearing angle
60	γ_{TAS}	[rad]	aerodynamic path angle
61	γ	[rad]	path angle
70	$\dot{h}$	[m/s]	geometric rate of climb
71	$\dot{H}_p$	[m/s]	rate of climb
80	m	[kg]	mass

Table 6 AIDL Profiles

Profiles			
#	**Keyword**	**Profile Name**	**Groups**
1	**L**	Lateral	**LDC**, **LDG**, **LPG**
2	**V**	Vertical	**VPG**, **AG**, **PAC**, **VSG**
3	**E**	Energy	**EG**
4	**S**	Speed	**HSG**, **SG**, **TG**
5	**T**	Thrust	**TC**

3 AIDL Differential Algebraic Equation Formulations

Despite its many favorable attributes listed in Table 1, AIDL has had only limited acceptance. This may partly be due to the numerical complexity of resolving the DAEs in order to compute the aircraft trajectory. The primary objective of this paper is to provide a generic manner by which standard tools can be used to circumvent

Table 7 AIDL Threads & Motion Profile Allowed Combinations

Threads						
#	Keyword	DOF	Profiles			
1	LAT	Lateral	L	L	L	L
2	LON	Longitudinal	V	V	E	S
3	PROP	Propulsive	S	T	T	T

the need for specially designed software to formulate the DAE according to each possible combination of AIDL instructions.

In this section the DAE formulation and its implications shall be presented. Below, equations 1-3 represent the force equations of the three-*dof* fixed-wing model, 4 is a mass variation equation, and equations 5-7 are the navigation equations describing the aircraft position.

$$\dot{\mathrm{v}}_{\mathrm{TAS}} - \frac{T - D - W\sin(\gamma_{\mathrm{TAS}})}{m} + \dot{w}_1^{WFS} = 0 \tag{1}$$

$$\dot{\gamma}_{\mathrm{TAS}} - \frac{1}{\mathrm{v}_{\mathrm{TAS}}}\left[\frac{L\cos(\mu_{\mathrm{TAS}}) - W\cos(\gamma_{\mathrm{TAS}})}{m} + \left(\dot{w}_3^{WFS}\cos(\mu_{\mathrm{TAS}}) + \dot{w}_2^{WFS}\sin(\mu_{\mathrm{TAS}})\right)\right] = 0 \tag{2}$$

$$\dot{\chi}_{\mathrm{TAS}} - \frac{1}{\mathrm{v}_{\mathrm{TAS}}\cos(\gamma_{\mathrm{TAS}})}\left[\frac{L\sin(\mu_{\mathrm{TAS}})}{m} + \left(\dot{w}_3^{WFS}\sin(\mu_{\mathrm{TAS}}) - \dot{w}_2^{WFS}\cos(\mu_{\mathrm{TAS}})\right)\right] = 0 \tag{3}$$

$$\dot{\mathrm{m}} + F = 0 \tag{4}$$

$$\dot{\lambda} - \frac{\mathrm{v}_{\mathrm{TAS}}\cos(\gamma_{\mathrm{TAS}})\sin(\chi_{\mathrm{TAS}}) + w_2^{WFS}}{(N+h)\cos(\varphi)} = 0 \tag{5}$$

$$\dot{\varphi} - \frac{\mathrm{v}_{\mathrm{TAS}}\cos(\gamma_{\mathrm{TAS}})\cos(\chi_{\mathrm{TAS}}) + w_1^{WFS}}{(M+h)} = 0 \tag{6}$$

$$\dot{\mathrm{h}} - \mathrm{v}_{\mathrm{TAS}}\sin\gamma_{\mathrm{TAS}} - \dot{w}_3^{WFS} = 0 \tag{7}$$

In addition to the specifier flight variables presented in Table 4, Table 8 describes the remaining variables found in Equations 1-7.

As highlighted in [5, 11], the variables may be grouped into state X and control u vectors, i.e.

$$X = \{\mathrm{v}_{\mathrm{TAS}}, \gamma_{\mathrm{TAS}}, \chi_{\mathrm{TAS}}, \lambda, \varphi, h, m\} \tag{8}$$

$$u = \{\mu_{\mathrm{TAS}}, L, \delta_{\mathrm{T}}\} \tag{9}$$

while the remaining variables are determined as a function of these within the Aircraft Performance Model (APM) and the Earth Model (E). The AIDL instructions are equivalent to constraints acting upon the state and control vectors. In the case

Table 8 Dynamic Equation Variables

Dynamic Equation Variables		
Symbol	**Units**	**Variable Name**
v_{TAS}	[mps]	true airspeed
T	[N]	aircraft thrust force magnitude
L	[N]	aircraft aerodynamic lift force magnitude
D	[N]	aircraft aerodynamic drag force magnitude
W	[N]	aircraft weight magnitude
F	[massps]	fuel consumption rate
$\dot{w}_j^{WFS}$	[mps2]	wind acceleration, j^{th} component, in wind relative coordinate system
w_j^{WFS}	[mps]	wind velocity, j^{th} component, in wind relative coordinate system

of a 3-*dof* fixed wing model, three such constraints selected according to compatible categories as given in Table 7 then uniquely define the trajectory. The model equations may then be represented as

$$\begin{aligned} f(X,u,E(X,u),t) &= \dot{X} \\ g_1(X,u,E(X,u),t) &= 0 \\ g_2(X,u,E(X,u),t) &= 0 \\ g_3(X,u,E(X,u),t) &= 0 \end{aligned} \tag{10}$$

The differential index of a the DAE system in Eq. (10) is defined to be the maximum number of differentiations of each of the algebraic constraints $g_i(X,u,E(X,u),t)$ necessary such that the system may be represented as an ODE system. It is a common indicator of the difficulty for solving such a system. Many numerical integration tools can solve systems of differential index 1, but not of a higher index. Mechanical systems generally do not present an index higher than 3, but even these are challenging to solve.

Table 9 Minimum DAE Differential Index

Minimum DAE Index	Sample AIDL Instructions
1	TL, HBA
2	HS, HPA, HC
3	LPL, VPL, HA

As described in [5], the implications of applying AIDL instructions (constraints) to the flight equation of motion upon the DAE index are listed in Table 9. The maximum DAE index resulting from each of the three instructions implies the overall DAE index for the equation system (10). Note that only a set of example instructions are given. In general, constraints acting directly upon the control variables imply a minimum index of 1, constraints upon velocity variables a minimum index of 2, and constraints upon position variables an index of 3.

The system diagram representing the combination of the Trajectory Computation Model (TCM) with the Aircraft Performance Model (APM), the Earth Model (E), and the motion constraints resulting from the AIDL instructions is illustrated in Figure 2 [5].

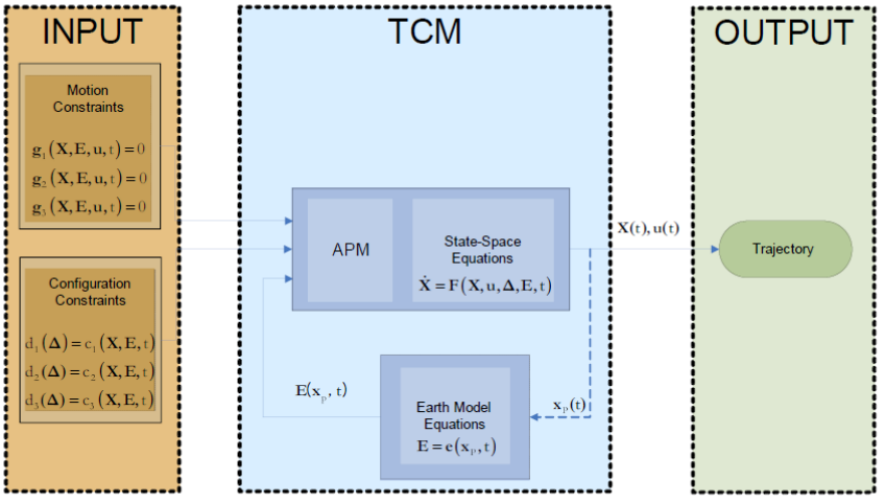

Fig. 2 AIDL Trajectory Computation Framework System Diagram

4 Modelica Formulation of AIDL Trajectory Generation

Modelica is an object-oriented and declarative language for modeling large, complex, and heterogeneous physical systems. Modelica provides language elements which allow to concisely describe system models using differential, algebraic, and discrete equations. A Modelica compiler will automatically transform these equations into a unified mathematical description form called *Hybrid DAE* [8]. This hybrid DAE forms the basis for many potential realizations of the model equations, such as simulator code for numerical integration of the equations of motion, plant models to do mathematical optimizations or control design, or simply executable code of the model equations.

The solution of the AIDL trajectory generation problem sketched in Section 3 can benefit from several key features of the Modelica language and its transformation into a hybrid DAE. The declarative nature of Modelica strongly facilitates the implementation of differential equations. First, the differential and algebraic equations do not have to be manipulated symbolically to obtain a causal model; this is done automatically during translation by the Modelica tool. Second, and more essential in our case, AIDL introduces constraint equations to the flight dynamic

model. Since Modelica facilitates the switching of any state-dependent symbolic (constraint) equation, there is no need to provide a distinct DAE system for each AIDL constraint pattern. This leads to a minimum implementation effort and concise and maintainable models.

Controlled technical system models are often characterized by continuous, piecewise continuous, and discrete behavior, all at the same time. This is also termed *hybrid* behavior. Prominent examples are digitally controlled plants, the modeling of friction and hard stops, or analog-digital converters. In AIDL trajectory generation, the discontinuous component is introduced by the timely piecewise structure of AIDL threads, which is not part of the intrinsically continuous aircraft dynamics. Modelica allows for the crisp definition of *events*, special points in time, where the system behavior switches between different regimes. So called *if equations* allow to switch between sets of equations according to the system state, in our case, the active set of constraints prescribed by the AIDL threads. In order to perform proper and physically meaningful transitions between the continuous phases, Modelica provides so called *when equations* which allow activating a behavior (a set of equations) *at* a specific event. The events are triggered by Modelica operators like `change()` or `edge()` which observe changes in variables. This can be used to transfer the system state properly from the regime before the event to the appropriate regime after. In the context of AIDL, this implies catching the trigger events, processing the AIDL threads, manipulating the constraint 'state', and finally introducing jumps in the physical states of the aircraft performance model if required.

In order to illustrate aspects of the Modelica implementation, a simplified fragment of the thread logic is presented in Listing 1. One can see the intuitive definition of trigger events in the when-clause where continuous and discrete variables are mixed. The aircraft variables are available in the `Thread` class' connector component `apmVariables`, and the constraint state is exposed through the connector `constraints`. The AIDL thread is made up of a list of `Instruction` objects, each containing the current command type, trigger type, and potentially required numerical values such as throttle position. Please note the Thread class uses a special type of equations placed in a section called `algorithm` where the equation order is relevant.

Listing 1 Modelica fragment of the AIDL Thread class

```
model Thread  "AIDL Thread class"
  parameter Instruction instructions[:] "AIDL sequence";
  Integer ip(start=1) "AIDL instruction pointer";
  CommandType currentCommand;
  TriggerType currentTrigger;
  Real triggerTime(start=-1) "Simulation time next trigger event";
  Boolean tl "Throttle law active";
  APMVariables apmVariables "Physical aircraft state";
  ...

algorithm

  // Testing for a set of Trigger Events; time:=simulation time
  when ({
```

```
    currentTrigger==TriggerType.T_elap and time>triggerTime,
    change(currentTrigger==TriggerType.H  and apmVariables.h<
        triggerHeight),
    ...}) then

    // fetch next AIDL command
    ip := ip+1;
    currentInstruction := instructions[ip];
    currentCommand := currentInstruction.cmd;
    currentTrigger := currentInstruction.trg;
    ...

    // process next command
    if currentCommand==CommandType.TL then
      tl := true;
      constraints.value := currentCommand.argument;
    elseif ...
      ...
    end if;

    // process next trigger
    if currentTrigger==TriggerType.T_elap then
      triggerTime := currentTrigger.argument;
    elseif ...
      ...
    end if;

  end when;
end Thread;
```

The Modelica approach pays off when it comes to the translation of the model equations presented in Section 3 into the hybrid DAE. At the very heart of all component based modeling formalisms like Modelica lies the problem of handling high index DAEs arising from multiple constraint equations. High index DAEs either require very special numerical integration schemes with often poor performance or can be transformed by a method called *index reduction* to a less demanding formulation [1]. Modelica tools rely upon methods proposed in [7, 9] which enable solving the index reduction problem including state selection and consistent initialization for a wide class of DAE systems automatically. The Modelica language itself also provides powerful language elements to alleviate consistent initialization of DAE systems [10]. In our case, the AIDL results in a vast amount of possible constraint configurations, which may even involve the permutation of state and control variables. The general index reduction method liberates the modeler from the tedious task of preparing separate DAE or ODE systems for each constraint situation.

Numerical solution of the continuous aircraft dynamics, Eqs. 1-7, imposes no special problems, but manual implementation of the hybrid behavior of AIDL sequences can be challenging. Since events as well as constraints are modeled in our approach symbolically by means of Modelica equations, the Modelica compiler is able to generate code for the time integration of the differential equations which takes special care of the events. This event handling embedded in the model equa-

tions can then be tightly coupled to standard integration schemes to allow for robust and fast simulation [4].

Similar to other object-oriented languages such as Java and C++, Modelica incorporates programming concepts such as classes, instantiation, and encapsulation. Equations implementing the behavior of a specific model component, e.g. the aircraft performance model, are placed into a distinct model class. An incarnation or instance of a class is called *component* in Modelica. Variables and states residing in a component are accessed through specific model ports called *connectors*. A complete simulation model is formed by a collection of components, hence, realizations of all the associated model equations. Physical interaction or exchange of information between model components is defined by a relation between two model connectors called *connection*.

Though Modelica is a purely textual language, it allows for convenient component-based graphical modeling through its object-oriented nature and a special language element called *annotation*. This is inert to the modeled dynamics but amenable to graphical modeling and rendering within an authoring tool.

The visualization of the Modelica trajectory framework used in the example model in the next chapter is shown in Figure 3. One can identify several entities: The `earthModel` provides global environmental conditions such as air pressure and wind. The `aircraftPerformanceModel` implements the equations of motion and the possible constraint equations required by the AIDL Eq. 1-7. This block exposes the physical state of the aircraft, essentially a unique selection of variables listed in Table 4, to the thread components via a connector depicted by an orange circle. The green triangle represents a connector through which the APM expects a set of control signals containing the AIDL constraint pattern and the associated values of the control variables. Our example requires three thread components which implement the AIDL thread logic and triggers. These blocks expect an AIDL thread sequence and output a constraint pattern finally shared with the APM to switch the constraint equations accordingly. The `sync` block merges and coordinates the several flows of constraint patterns stemming from the longitudinal and lateral AIDL threads. Lines in the graphical layer represent connections between connector objects.

5 Results from Simulation Experiments

A numerical experiment is defined to demonstrate the AIDL switching behavior. The Modelica solution is compared with existing proprietary, specially designed solution software with identical results. The experiment consists of an initial horizontal flight for a small UAS, followed by a deceleration phase, then a descent phase, and finally another horizontal flight.

The AIDL threads shown in Table 10 were used as parameters for the thread components in the Modelica model. Note that the thread definitions are categorized as {*Longitudinal 1, Longitudinal 2, Lateral*} rather than {*Propulsive, Longitudinal,*

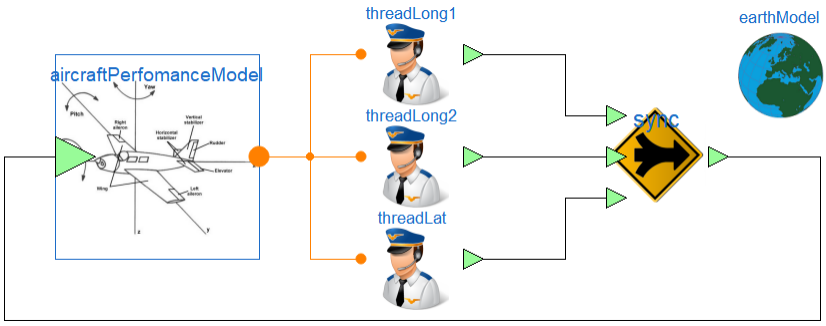

Fig. 3 Graphical representation of the Modelica version of the trajectory computation framework for the simulation experiments presented in Section 5.

Lateral} as indicated in Table 7. This is because a priori the nature of the non-Lateral motion threads cannot be determined from observing a single instruction. In particular, instructions from the Speed profile (see Table 6) may belong to either the Propulsive or Longitudinal threads depending upon the instructions in the other non-Lateral thread as shown in Table 7. Once both non-Lateral threads are observed, however, the assignment to the {*Propulsive, Longitudinal*} threads is straightforward.

Longitudinal thread 2 was augmented by a TERM command to stop simulation after the last trigger. Initial values used for the experiment were $m = 20\,\text{kg}$, $\lambda = -4.3680°$, $\varphi = 40.907051°$, $v_{CAS} = 30\,\text{m/s}$, $h = 2000\,\text{m}$, $\chi_{TAS} = 350°$, $\mu_{TAS} = 0°$. The solver used was DASSL with a tolerance of 10^{-4}. CPU time used for resolving the problem on a standard PC are almost neglegible at 100 milliseconds.

Table 10 AIDL Threads for Experiment 1

Longitudinal 1 Thread						
Inst	**SpecVar**	**TargVal**	**TrigCode**	**TrigVar**	**TrigVal**	**TrigID**
HA	h	2000 [m]	14	v_{CAS}	22 [mps]	—
HS	v_{CAS}	22 [mps]	0	ID	*end*	—
Longitudinal 2 Thread						
Inst	**SpecVar**	**TargVal**	**TrigCode**	**TrigVar**	**TrigVal**	**TrigID**
HS	v_{CAS}	30 [mps]	2	t	25 [s]	—
TL	ffi_T	0.1 [-]	20	h	1100 [m]	—
HA	h	1100 [m]	2	t	15 [s]	*end*
Lateral Thread						
Inst	**SpecVar**	**TargVal**	**TrigCode**	**TrigVar**	**TrigVal**	**TrigID**
HBA	μ_{TAS}	0 [deg]	0	ID	*end*	—

No special care had to be taken about the initialization of state variables and dependent variables for our simulation experiment, neither at simulation start nor after the trigger events. Selection of states and consistent initialization is done by the Modelica tool using [9]. All model equations are available in symbolic form, so the Modelica tool is able to either treat nonlinear systems on an equation level, e.g. by a method called tearing, or is able to introduce suitable solvers for the nonlinear systems to solve when it comes to start or restart the simulation.

Table 11 AIDL Realization

Phase #	1	2	3	4
LON1	HA	HA	HS	HS
LON2	HS	TL	TL	HA
LAT	HBA	HBA	HBA	HBA
DAE Index	3	3	2	3

The resulting instruction combinations within the four phases of the experiment are shown in Table 11. The DAE index for each phase is also given according to Table 9. Figure 4 shows the current AIDL command and trigger types valid at each time. These are obviously discrete variables potentially changing their values *only* at the trigger events in the manner of a typical state machine.

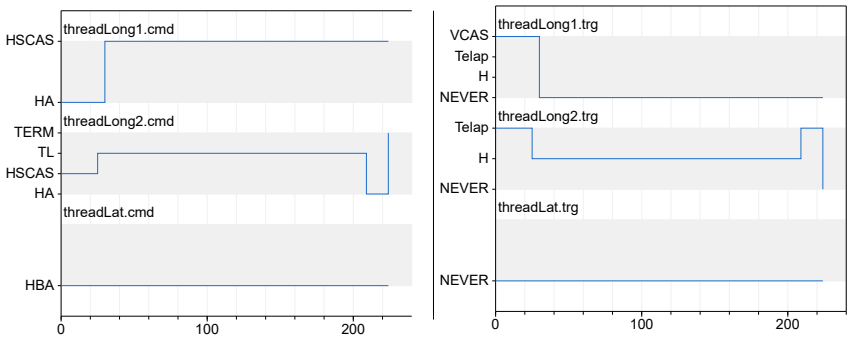

Fig. 4 AIDL command state (left picture) and AIDL trigger state (right picture) in Modelica model over simulation time for Experiment 1. Please note: the y-axis shows a discrete state and does not represent a numerical value.

The concurrently calculated aircraft variables are shown in Fig. 5. One can see that some states such as v_{CAS} and aircraft mass result in trajectories with non-differentiable corners and some variables like γ_{TAS} even show jumps. The same holds for the current forces acting on the aircraft shown in Fig. 6.

Those interested in the Modelica code for this experiment may contact the authors.

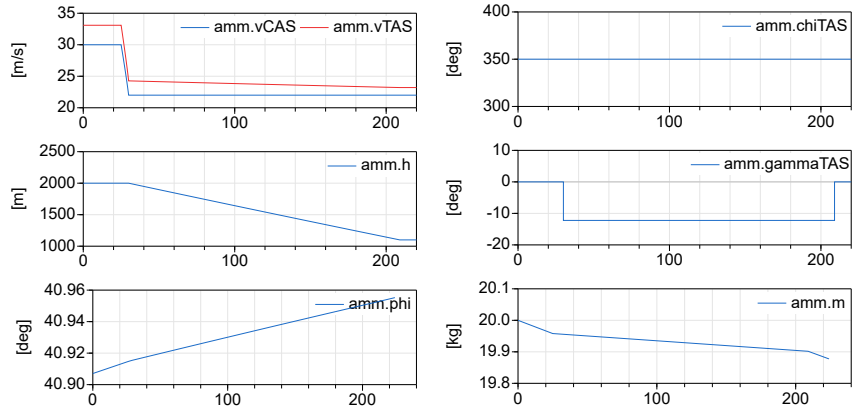

Fig. 5 Aircraft specifier variables over simulation time for Experiment 1.

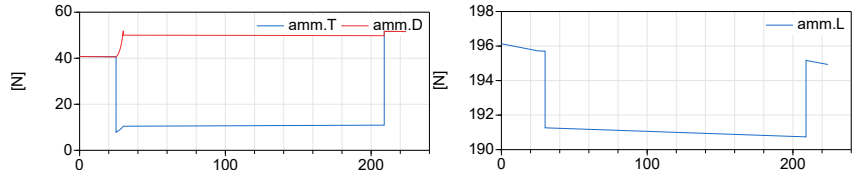

Fig. 6 Dynamic equation variables acting on aircraft over simulation time for Experiment 1.

6 Conclusion

A concise, generic Modelica implementation of the AIDL trajectory computation framework is presented. This implementation is powerful as it is perfectly adapted to handle the complexities implicit in the resolution of the differential-algebraic equations resulting from the AIDL trajectory definition framework. The simplicity of this scheme is considered to be an enabler, in contrast to previous solution schemes, which may facilitate the use of AIDL as a UTM solution. The much improved ease of AIDL calculation lends itself to a more automated, cooperative, interoperable, criterion-based, efficient, scalable, and sustainable UTM alternative. Furthermore, Modelica tools permit the automatic generation of embeddable code for ease of deployment.

References

1. K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. North-Holland, New York, Amsterdam, London, 1989.
2. Mauricio Castillo-Effen, Liling Ren, Han Yu, and Corey A. Ippolito. Off-nominal trajectory computation applied to unmanned aircraft system traffic management. In *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, St. Petersburg, FL, USA, 2017. IEEE.

3. FAA. Nextgen, concept of operations v1.0, unmanned aircraft system (uas) traffic management (utm), 2018. https://utm.arc.nasa.gov/docs/2018-UTM-ConOps-v1.0.pdf.
4. Peter Fritzson. *Object-oriented Modeling and Simulation with Modelica 3.3*. Wiley, 2 edition, 2015.
5. Javier López-Leonés. *The Aircraft Intent Description Language*. PhD thesis, University of Glasgow, 2007.
6. Javier López-Leonés, Miguel A. Vilaplana, Eduardo Gallo, Francisco A. Navarro, and Carlos Querejeta. Towards a formal language for the common description of aircraft intent. In *IEEE/AIAA Digital Avionics Systems Conference*. IEEE, 2007.
7. Sven Erik Mattsson and Gustaf Söderlind. Index reduction in differential-algebraic systems using dummy derivatives. *SIAM Journal of scientific and statistical computing*, 14:677–692, 1993.
8. Modelica Association. *Modelica Language Specification - Version 3.3*, 2012.
9. C.C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM Journal of scientific and statistical computing*, 9:213–231, 1988.
10. Michael Sielemann, Francesco Casella, Martin Otter, Christoph Clauss, Jonas Eborn, Sven Erik Mattsson, and Hans Olsson. Robust initialization of differential-algebraic equations using homotopy. In *Proceedings of the 8th Modelica Conference, Dresden, Germany*, 2011.
11. Miguel A. Vilaplana, Eduardo Gallo, and Francisco A. Navarro. The aircraft intent description language: A key enabler for air-ground synchronization in trajectory-based operations. In *Digital Avionics Systems Conference*, Brussels, Belgium, 2005. IEEE.