Computing Probability Distributions Over a Hybrid State Space: Case Study and Practical Limitations *

[Experience Report]

Alessandro Pinto
United Technologies Research Center
2855 Telegraph Avenue
Berkeley, CA 94705
PintoA@utrc.utc.com

George A. Mathew
United Technologies Research Center
2855 Telegraph Avenue
Berkeley, CA 94705
MathewGA@utrc.utc.com

ABSTRACT

The problem of computing the probability distribution over the hybrid state space of a Discrete-Time Stochastic Hybrid System (DTSHS) is considered. A tool called USHVER (UTRC SHS Verifier) is presented that allows to model DTSHSs directly in C++. Once a system is modeled, it can be used by back-end tools to perform Monte-Carlo simulations, reachability analysis, or property checking using the Sequential Probability Ratio Test. Given the complexity of the models usually found in realistic design problems, the development of USHVER has been constrained by the requirement of begin agnostic with respect to the dynamics of the system: flows and reset maps are treated as black-box models. In this paper, the basic ideas on which USHVER is built are presented. In addition, a detailed model of the thermal management system of a prototypical aircraft is used to present some results and discuss the limitations of the approach on realistic applications.

1. INTRODUCTION

In 2010, the United Technologies Research Center (UTRC) conducted a study on the subject of verification of dynamical systems subject to uncertainty [12]. Several tools existed for the verification of probabilistic systems [5, 14, 4, 10]. They either addressed the verification problem for Markov Chains, or they imposed restrictions on the dynamics and uncertainty in the SHS model. The study led to the development of a tool called USHVER to model Discrete Time Stochastic Hybrid Systems (DTSHS) directly in C++ and to perform simulation and analysis. Many of the assumptions typically made on the dynamics of a DTSHS to limit the complexity of the analysis task are often too restrictive to capture realistic systems. Thus, it was intentional to avoid making any assumption on the dynamics of the system and to understand the limitations of the resulting tools. In this article, we report our experience in developing and using the software. In Section 2, we introduce an operator that allows to compute the evolution of probability measures over time. We also briefly discuss why such operator is interesting for the development of USHVER. In Section 3, we describe the tool including the main architectural decisions. In Section 4, we present an example with the intent of showing our experience in creating an abstract model of a system that still includes relevant details. In Section 5 we show some results and in Section 6 comment on the limitations we faced in using the tools. We also believe that the model presented in Section 4 could be used in the future to create a benchmark for stochastic hybrid systems given its level of details including the values of all key parameters.

*Content taken from the DARPA V2D2 Study "Stochastic Analysis and Design of Systems" [12]

2. MARKOV MODELING OF STOCHASTIC HYBRID SYSTEMS

Several methods for the analysis of DTSHS (and dynamical systems) are based on discretization. For example, the approximation technique recently developed by Abate *et al.* [3, 7] starts from a a finite partition of the state space and use Markov Set-Chains [9] as approximate model fo the original DTSHS. The authors focus on providing error bounds between the original system and the Markov Chain approximation but place some restrictions on the smoothness of the probability distributions.

Discretization techniques have been exploited in the work of Dellnitz [6] and Froyland [8] that have used set-oriented methods to model continuous dynamical systems. The basic idea in this work is to partition the domain of the continuous variables (referred to in the sequel as the continuous state space) into a finite number of sets. Each set becomes the state of the Markov chain where transition probabilities are interpreted as the probability of a typical point in one set to move to another set under the constraint imposed by the dynamics of the system. This approach is used in USHVER except that the hybrid nature of the dynamics, namely the jump conditions and the different dynamics for each mode of the hybrid system need to be taken into account. This discretization technique is general and allows removing *restrictions on the dynamics of the system and on the probability distributions of the stochastic processes*. Unfortunately, this freedom comes at the *expense of the complexity of the analysis task and leads to the inability of computing error bounds in the general case*. However, the method used in USHVER has other important practical properties stated later in this Section.

The discrete state space is represented by $\mathcal{Q}$ and we consider a continuous state in $\mathbb{R}^n$. The state space of the hybrid system can then be defined as $\mathcal{S} = \mathcal{Q} \times \mathbb{R}^n = \cup_i \{q_i\} \times \mathbb{R}^n$ (this model has been extended to a more general state space and to IO-DTSHS in [12] and extensions are supported by USHVER). The state space model for a dis-

crete time stochastic hybrid system is a collection $\mathcal{H} = (\mathcal{Q}, Init, T, L, R)$ where $\mathcal{Q} := \{q_1, q_2, ..., q_m\}$ is a finite set of modes with $m \in \mathbb{N}$; $Init : \mathbb{B}(\mathcal{S}) \to [0, 1]$ is an initial probability measure on $\mathcal{S}$; T (the flows) is a stochastic map that describes the dynamics of the continuous variables $x \in \mathbb{R}^n$ in each mode; L is a switching probability function that gives the probability of switching between various modes; R is a stochastic map that probabilistically resets the values of the continuous state variables when a switch occurs from mode q_i to mode q_j. The dynamics of the continuous variables in mode q_i is given as $x(n+1) = T(q_i, x(n), \xi_i(n))$, where $\xi_i(n)$ is an i.i.d process with distribution $\mathcal{N}_i$. $L(x, q_i, .)$ is a probability measure on the discrete state space $\mathcal{Q}$. i.e., $L(x, q_i, q_j)$ gives the probability of the system to jump from mode q_i to mode q_j given the value of the continuous state x. The reset is given as $x(n + 1) = R(q_i, q_j, x(n), \eta_j(n))$ where $\eta_j(n)$ is an i.i.d process with distribution $\mathcal{W}_j$.

The execution of a state space model for the discrete time stochastic hybrid system over a finite time horizon $[0, N]$ is defined as in [3]. We will now define how the probability measure can be propagated over time. With a slight abuse of notation, we use the same symbols for measures and probability distribution functions. For the flow corresponding to each mode q_i, the propagation of measures is described by the Frobenius-Perron operator corresponding to the map $T(q_i, ., .)$. This is the unique operator $[P_i]$ such that

$$\int_A [P_i]\mu(x)dx = \mathbb{E}_{\xi_i}\left[\int_{\mathbb{R}^n} \mu(x).\chi_A(T(q_i, x, \xi_i))dx\right]$$

For all $A \subset \mathbb{R}^n$. Note that if the probability measure at time k is given by $\mu(k)$, then the measure at time $k + 1$ is given as $\mu(k + 1) = [P_i]\mu(k)$. Moreover, eventhough the maps $T(q(k), x(k), \xi_i(k))$ are non-linear, the Frobenius-Perron operators are linear, but infinite-dimensional. For more details on the theory of these transfer operators, see [11]. For the switching between the modes q_i and q_j, we define the switching transfer operator as
$[L_{i,j}]\mu(x) = L(x, q_i, q_j).\mu(x)$. Finally, the change in measures due to probabilistic resets is given by the Frobenius-Perron operator $[M_{i,j}]$ corresponding to the map $R(q_i, q_j, ., .)$ computed similarly to operator $[P_i]$ above.

The evolution of the probability measures over the whole hybrid state space can be described using a single transfer operator given as $\Gamma^{k+1} = \mathbb{P}\Gamma^k$, where $\Gamma^k = [\mu_1^k, \ldots, \mu_m^k]^T$ and μ_i^k is the sub-probability measure restricted to mode i at time k. The transfer operator $\mathbb{P}$ is a block matrix, where block (i, j) is the product $P_i M_{i,j} L_{i,j}$ of the flow, reset and jump operators. Operator $\mathbb{P}$ could be used as input to probabilistic model checkers developed for Markovian models (e.g [10, 1]). In [6], Dellnitz et al. describe set oriented numerical methods to construct finite dimensional approximations for the Frobenius-Perron operator corresponding to a continuous dynamical system. The state space is partitioned into a finite number of connected sets $\{A_1, A_2, ..., A_n\}$. To form the Markov model, each set A_i is identified with a state i of an n-state Markov chain. A $n \times n$ matrix P is constructed, where $P_{ij} = \frac{m(A_i \cap T^{-1}A_j)}{m(A_i)}$ with m the Lebesgue measure.

The attractive features of the Frobenius-Perron operator are the following. As the size of each set A_i gets smaller (and therefore n increases), the approximation gets better meaning that the probability distributions computed using the operator tend to the ones of the original system. Moreover, the operator is linear meaning that computations can be parallelized. In addition, whenever the probabilities P_{ij} of the operator are computed through simuations, such simulation jobs can also be distributed over multiple machines.

3. USHVER

The implementation of the USHVER tool had to address several challenges. First, the DTSHS model described in Section 1 needed to be ultimately represented in memory. We avoided the definition of a language and a parser and we decided to provide a set of base classes directly in C++ that the user could simply extend for modeling a particular system[1]. The base class for a DTSHS is called `Shs` and provides methods to add sub-systems (allowing hierarchical models), modes and transitions. The `Mode` base class is extended by the user to define a mode in a DTSHS. The user overrides the `Flow` function which, given the continuos state and the input, computes the new continuous state. This map can be deterministic or stochastic depending on the model. The `Transition` base class is extended by the user to define a transition between two modes. The user overrides the `Switch` and `Reset` methods and also defines the source and target modes. The switch method computes the probability of taking the transition in a given state and for a given input, while the reset method is a map that, given the continuos state and the input, computes the new continuous state. A leaf DTSHS is built by instantiating the base class and adding modes and transitions to it. Multiple DTSHSs can be added to a parent DTSHS and wired through input-output connections. The `Shs` base class provides a method `Compose` to compose its children (this method is called from the root Shs and it is propagated throuhg the hierarchy). The `Shs` model is the input to several algorithms such as simulation, reachability and the SPRT algorithm. The outputs of these algorithms are hybrid traces or hybrid probability ditributions. A set of observers and filters is also provided to probe the `Shs` model in specific points (inputs, outputs and states).

As noted in the introductory section, reachability analysis is performed over a discretized state space. USHVER provides a grid data structure to encode the discretization without in fact creating an actual grid. Each mode q_i is characterized by an invariant region $Inv(q_i) \subset \mathbb{R}^{d(q_i)}$, where $d(q_i)$ is the dimension of the continuous state space in mode q_i. We limit our discussion to invariant regions that are orthotopes, meaning products of intervals: $Inv(q_i) = \left[x_{1,l}^{(i)}, x_{1,u}^{(i)}\right) \times \ldots \times \left[x_{n,l}^{(i)}, x_{n,u}^{(i)}\right)$, where we denoted by $x_{j,l}^{(i)}$ and $x_{j,u}^{(i)}$ the lower and upper bound of the j-th variable in the i-th mode, respectively. For a mode q_i, let $G^{(i)} = (g_1^{(i)}, \ldots, g_n^{(i)})$ represent the grid, where $g_d^{(i)}$ is the number if intervals for the d-th state. The length of each interval is $l_d^{(i)} = \frac{x_{d,u}^{(i)} - x_{d,l}^{(i)}}{g_d^{(i)}}$ The

[1]Please refer to `http://www.alessandropinto.net/data/ushver/html/bouncing_ball_example.html` for an example of how DTSH, modes and transitions can be defined in USHVER.

j-th interval is $I_{d,j}^{(i)} = \left[x_{d,l}^{(i)} + (j-1)l_d^{(i)}, x_{d,l}^{(i)} + jl_d^{(i)}\right)$. The discretization induces a new state space which is now finite and defined as $\hat{S} = \bigcup_{i=1}^{m}\{q_i\} \times [1, g_1^{(i)}] \times \ldots \times [1, g_n^{(i)}]$. Let $s \in \hat{S}$ be a state such that $s = (q_i, j_1, \ldots, j_n)$. Then, we use the following notation: $mode(s) = q_i$, $B(s) = (j_1, \ldots, j_n)$ and $H(s) = I_{1,j_1}^{(i)} \times \ldots \times I_{n,j_n}^{(i)}$. The finite state space is now encoded into a linear array for each mode. For a given mode, the total number of discrete states is $n_m^{(i)} = \prod_{d=1}^{D(q_i)} g_d^{(i)}$. The discrete linear hybrid state space is $\hat{S}_L = \cup_{q \in Q}\{q\} \times [1, n_m^{(i)}]$. The encoding is implemented by a function $map : \hat{S} \to \hat{S}_L$ that given a hybrid state $s \in \hat{S}$ computes $s' \in \hat{S}_L$ as follows: $mode(s) = mode(s') = q$ and $B(s')|_1 = B(s)|_{D(q)} + \sum_{d=1}^{D(q)} \left(\prod_{d'=d+1}^{D(q)} g_d^{(i)}\right)(B(s)|_d - 1)$. Finally, grids of multiple DTSHSs can be composed within a system by a `grid composition operator`.

To perform reachability analysis, it is necessary to compute operator $\mathbb{P}$ over the discretized state space. The generic entry P_{ij} of a flow operator can be interpreted as the probability that a typical point in the set A_i moves into the set A_j under one iteration of the map T describing the flow. The expression $P_{ij} = \frac{m(A_i \cap T^{-1}A_j)}{m(A_i)}$ does not have an explicit form in our case becasuse the flow and reset maps are generic, directly expressed in code (a design choice as explained in the introduction). Thus, the quantity P_{ij} is computed by a Monte-Carlo approach. One randomly selects a large number of points $\{a_1, ..., a_N\} \subset A_i$ and sets $P_{ij} \approx \#\{a \in \{a_1, ..., a_N\} : T(a) \in A_j\}/N^2$. Obviously, there are cases where P_{ij} can be computed in closed form and no sampling is required.

The first simple approach to reachability analysis is to generate the entire matrix representing the transfer operator. This approach would require to compute the quantity $\mathbb{P}_{ij}$ for each state i of the finite Markov Chain approximation even if state i is actually never reached by the dynamics of the system when considering the initial distribution. Thus, in the implementation of USHVER, the computation of transfer operators and the probability measure propagation are interleaved to avoid considering those parts of the state space that are never reached. The reachability analysis algorithm explores the set of reachable states explicitly. The basic data structure used to hold the reachable state space is a reachability graph $R_G(C, E, P)$ where C is a set of vertexes, E is a set of edges and $P : E \to [0, 1]$ is a function that associates a probability to each edge. A cell is a tuple $c(s, \mu) \in C$ such that $s \in \hat{S}_L$, $\mu \in [0, 1]^2$ representing the probability of being in state s – the probability measure is a vector of two elements for efficiency reasons so that measure propagation can be done with a single sweep over the set of reached states. Algorithm 1 shows the basic algorithm. A pair of queues Q_n is used to avoid expensive copy operations when new states are discovered and during measure propagation.

The algorithm alternates between the discovery of new reachable states and measure propagation. The queue of reached cells Q and the current queue of states to be processed $Q_n[1]$

[2]The curious reader may find an example of how this is computed inside the reachability analysis code by an operator called `LocalPf` also discussed later in this article

```
Input: DTSHS H, N
Output: R_G(Q, E, P)
k ← 0 ;
/* Initialize the reachable set                    */
Q ← Init(H) ;
/* Q_n is the frontier of new reached cells        */
Q_n[0] ← Q ;
while k < N do
    /* Previous and current indices                */
    i ← 1 − kmod2, i_n ← kmod2 ;
    Q_n[i_n] ← ∅ ;
    forall the c(s, μ) ∈ Q_n[i] do
        (S', P') ← LocalPf(s) ;
        forall the s' ∈ S' do
            c' ← (s', (0, 0)) ;
            if c' ∉ Q then
                Q ← Q ∪ {c'} ;
                Q_n[i_n] ← Q_n[i_n] ∪ {c'} ;
            end
            E ← E ∪ {(c, c')} ;
            P(c, c') ← P'(s') ;
        end
    end
    /* Propagate the probability measure           */
    forall the c(s, μ) ∈ Q do
        μ[i_n] ← 0 ;
    end
    s forall the (c(s, μ), c'(s', μ')) ∈ E do
        μ'[i_n] ← μ'[i_n] + μ[i] · P(c, c');
    end
end
```

Algorithm 1: Reachability and measure propagation

(i.e. new states found in the previous iteration) are initialized with the set of initial states with a non-zero probability. The index of the queue containing the cells found in the previous iteration is i, and the index of the queue which will contain all the new states reachable in one step is i_n. In the discovery phase, for each cell in $Q_n[i]$, a function `LocalPf` (local Frobenius-Perron operator) computes a set of new states $S' \in \hat{S}_L$ with associated probabilities P' (i.e. the probability of reaching state $s' \in S'$ from s). This function takes into account flows, jumps and reset maps. Each new state is added to the next queue $Q_n[i_n]$ and to queue Q. Finally, the edges of the graph and the associated probability are also added to the reachability graph. In the second phase, the algorithm propagates the probability measures. For each cell in the reachability graph, probability mass is moved to the output cells according to the previous value of the probability measure and to the transition probabilities P.

Optimizations. The basic algorithm has several drawbacks in terms of memory requirements and speed. A few improvements have been implemented to deal with clocks and to limit memory requirements. If the evolution of a clock variable is independent from other variables (free clock), then the dynamics of the system is invariant with respect to the value of the clock. Thus, for a state $s = (q_i, \delta, j_1, \ldots, j_n)$ where δ is a free clock, we generate a state $s_t = (q_i, 0, j_1, \ldots, j_n)$ which is queried for any other value of the free clock. This method allows avoiding unnecessary calls to the `LocalPf` function which is in general expensive.

As the probability measure is propagated over time, the probability mass of some cells descreases and may even be-

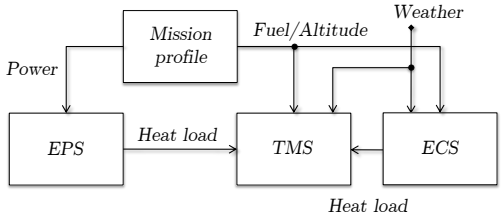


Figure 1: High-level model of the system under study.

come zero. For example, if a clock is used to encode time, then the probability mass at step $t+1$ of all cells with the value of the clock variable less than $t+1$ should be equal to zero. Thus, these cells can be removed from the queue with large memory savings. During the execution of the reachability algorithm we periodically traverse Q and we remove states that have zero probability mass. Traversing the queue of reached states is an expensive operation but can reduce memory requirements. The removal is non-trivial and requires multiple passes starting from cells which do not have input edges in the reachability graph. This is important as it is not possible to remove cells with zero probability values but with some active cells in their input cone of influence.

4. EXAMPLE

We model the thermal management system of a prototypical aircraft executing a certain class of missions. USHVER is then used to computed the fuel temperature distribution over time from take-off to landing. The purpose of this exercise is to use the tools and show their limitations on a realistic case study. At the same time, we hope that the models described in this section will be of use in setting up a benchmark for future studies.

The fuel temperature on an aircraft is affected by the mission profile, weather conditions and heat loads such as the engine and the electric power system. All these elements are uncertain. Figure 1 shows a high level model of the system to be analyzed. The mission profile drives the dynamics of all the aircraft sub-systems inducing a power requirement profile $w(t)$, thrust profile $f(t)$, velocity profile $v(t)$, and altitude profile $h(t)$. The heat generated by the electric power system depends on the efficiency of the generators η_G and the efficiency η_C of other components (such as power converters) that are present on the aircraft[3]. The altitude and velocity of the aircraft affect the air temperature T_A and air density d_A and therefore the ability to extract heat through a Fuel/Air heat exchanger. Weather conditions also impact T_A as well as the heat that needs to be rejected by the thermal management system (due to kinetic effects). Finally, the amount of fuel consumption affects the fuel flow rate through heat exchangers, thereby affecting the temperature of the fuel at their outlets. All these effects must be modeled: the mission profiles, the environment such as weather conditions, and the system under study.

Modeling the mission profile. Several standard profiles of missions are available in literature [13], ranging from short

[3]We have omitted other heat loads such as the engine in Figure 1 for brevity.

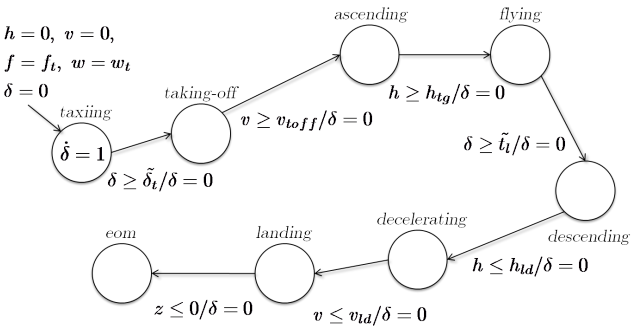


Figure 2: Mission profile of a mission without refueling.

Mission phase	H (kW)	W (kW)	F (kg)
Taxiing	18.44	84.1	4,325
Take-off/Decelerating/ Landing	26.63	84.4	17,300
Ascending/Descending	27	83	17,300
Flying	20	76.4	8,000

Table 1: Heat load, power and thrust requirements for a prototypical UAV.

to long missions which may include refueling. The stochastic hybrid system model of the mission profile we selected is shown in Figure 2. Each state represents one phase of the mission and it is characterized by a set of differential algebraic equations that defines the way in which altitude h, velocity v, thrust f and power requirement w change over time. Variables δ is used to encode time. The aircraft remains on the ground for a certain amount of time $\tilde{\delta}_t$ which is a random parameter. During taxing, the aircraft requires some level of thrust f_t and some level of power w_t that are both considered constant.

The take-off phase is characterized by a constant altitude, a constant acceleration of the vehicle and constant thrust and power requirement. When the aircraft reaches a velocity v_{toff}, the mission switches to a phase where the aircraft starts ascending. When a target altitude h_{tg} is reached, the mission changes phase to a state where the aircraft flies at constant altitude. During this phase thrust, power requirement and velocity change according to a set of stochastic differential equations. After a random amount of time $\tilde{t}_l$, the aircraft starts its descending phase and switches to the deceleration phase when the altitude reaches a landing altitude h_{ld}. Finally, the aircraft lands and ends the mission when the ground is touched. The end-of-mission (eom) state is an accepting state of this automaton.

The maximum total take-off weight of the aircraft is approximately $50,000\ kg$. We consider that the aircraft is initially full of fuel. The dry mass of the aircraft is $10,400\ kg$, and the total fuel capacity is $8,400\ kg$. The average heat, power and thrust requirements (H, W and F, respectively) in the different phases of the missions are shown in Table 1.

The thrust needed to keep compressing air into the engine in the taxiing mode is 25 % of the take-off thrust. We as-

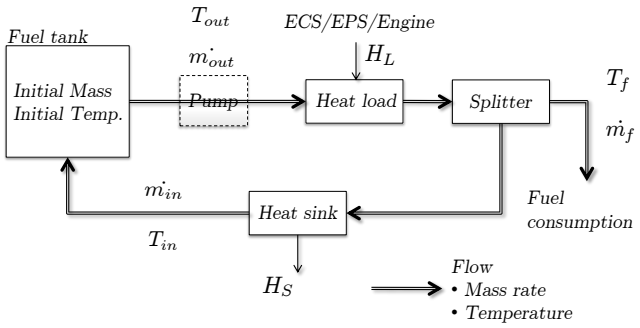


Figure 3: Model of the thermal Management System.

sume that the time spent in this mode of operation is uniformly distributed between 5 and 15 minutes. During take-off, velocity increases according to the following equation: $\dot{v} = \frac{f - f_{drag}}{m}$ with $f_{drag} = c_d \cdot \rho \cdot A \cdot v^2$, where the drag coefficient $c_d = 0.048$, ρ is the air density and A is the wing area which we consider to be 28 m^2. The take-off velocity is $v_{toff} = 240\ km/hr$. The climb angle is $\pi/3$ and the target altitude is $h_{tg} = 10,000\ m$. The descend rate is $h_d = 300\ m/s$ and the landing altitude is $h_{ld} = 200\ m$.

Modeling the weather conditions. The ram air temperature depends on the air density and temperature (as well as the speed of the aircraft). We use models similar to the U.S. Standard Atmosphere models [2]. The key variables of interest to us are: The altitude h, the pressure p, the temperature T, the acceleration due to gravity $g = 9.8m/sec^2$, the radius of earth $R = 6356.766km$, the sea level temperature $T_0 = 15.0^0C$ and the sea level pressure $p_0 = 101,325N/m^2$. The model comprises a series of six layers, each defined by a linear temperature gradient also called *lapse rate*. The base altitude for a layer is denoted by h_n while the lapse rate is λ_n (in K/km). The temperature distribution within layer n is given by $T = T_n + (h - h_n)\lambda_n$. Using the ideal gas law equations and the equations for hydrostatic equilibrium, the pressure distribution within layer n is given by the expression $\frac{p}{p_n} = \left(1 + \frac{(h-h_n)\lambda_n}{T_n}\right)^{-g/(\lambda_n R)}$. For isothermal layers ($\lambda_n = 0$), the above expression reduces to $\frac{p}{p_n} = e^{-(h-h_n)g/(RT_n)}$. The above equations link the altitude to the pressure and temperature of the air that is used for the fuel-air heat exchangers on the aircraft. We only consider the first two layers with $h_0 = 0\ m$, $\lambda_0 = -6.5\ K/km$, $h_1 = 11\ km$, $\lambda_1 = 0$.

Modeling the Thermal Management System. Figure 3 shows the model of the thermal management system where we consider the flow rates at steady state or slowly varying. Also, the volume of the fluid in this circuit changes relatively slowly. If this assumptions do not hold, then it is possible to better approximate the behavior of the system by using a series of models at constant flow rate, and capturing the transient effects in the transitions among modes.

A pump is used to push fuel from the tank through the fuel circuit. The heat produced by the environmental control system, the electric power system and the engine is absorbed by the fuel through heat exchangers that we have abstracted in this model. Part of the fuel is used by the engine while the rest returns to tank. Before entering the tank, the fuel is cooled to an appropriate temperature by an Air/Fuel heat exchanger that uses ram air. The variables of interest in this model are the total fuel level in the tank, the fuel flow rate, and the fuel temperature.

Using the nomenclature in Figure 3, we model the system with the following equations:

- $\dot{M} = \dot{m_{in}} - \dot{m_{out}} = -\dot{m_f}$, $M(0) = M_0$. This equation links the engine fuel consumption $\dot{m_f}$ to the total fuel mass M (where M_0 is the total fuel mass at the beginning of the mission). The fuel rate $\dot{m_f}$ can be derived from the thrust and from the thrust specific fuel consumption (TSFC) of the engine. For example, one representative engine consumes 0.7 $lb/lbt/hr$ (pounds per pounds of thrust per hour) without afterburner, and 2 $lb/lbt/hr$ with afterburner.

- $\dot{m_{out}} c_f (T_f - T_{out}) = H_L$ where H_L is the total heat rate from the heat loads on the aircraft. In this equation c_f is the specific heat of the fuel which is assumed to be 0.2 kJ/kg K.

- $\dot{m_{in}} c_f (T_f - T_{in}) = H_S$ where H_S is the heat rate that the sink is able to reject. The heat rate H_S depends on the velocity of the aircraft, the air density and the air temperature.

- The continuous dynamics for fuel-mass(M) and fuel-tank-temperature(T) for all modes are given as $\dot{M} = -m_f$ and $\dot{T} = \frac{1}{M}\left(m_{in}T_{in} - m_{out}T + m_f T\right)$

5. PARAMETERS AND RESULTS

The system that we consider has five continuous variables and eight modes, although the last mode has a trivial behavior. The continuous variables are the altitude (h), the velocity (v), the fuel-tank mass (M), the fuel-tank temperature (T) and a clock variable (δ). We are interested in computing the marginal probability distribution of the fuel-tank temperature at the end of the mission (i.e. when the system enters the eighth mode). Some of the mode transitions are triggered by the clock variables. For example, the switching probability from the *taxing* mode to *take-off* mode depends only on the clock variable δ which is reset to zero after mode transition. We set $m_{out} = 2m_f$ and $m_{in} = m_{out} - m_f$. This means that we are not modeling any additional control on the fuel rate. A controller could be added to the model, albeit an increase in the number of states and, therefore, in the complexity of the analysis. H_L is the heat-load. Part of the fuel that is not consumed by the combustor is recirculated through the fuel-air heat exchanger back to the fuel-tank. The temperature drop in the fuel after passing through the fuel-air heat exchanger is assumed to be a fraction (f) of the difference in the temperature of the fuel and air. This temperature drop is a function of the heat-exchanger effectiveness and depends on how often the ram air inlet can be opened. We set $f = 0.1$. The outside air temperature is modeled using different layers with different lapse rates. For our studies, only the first two layers are important. The temperature in the first two layers is given by the following

model:

$$\begin{aligned} T_{air} &= T_0 - 6.5h \quad \text{for } 0 \leq h \leq 11.0 \\ T_{air} &= 0.751865 T_0 \quad \text{for } 11.0 < h \leq 20.0 \end{aligned} \tag{1}$$

where $T_0 = 15^0 C = 288.15K$ and h is in km. We assume an that the fuel temperature is initially distributed uniformly between 288 and 298 degrees Kelvin. Also, we discretize the dynamics of the system using a Euler Forward Scheme. The discretization step is 1 second.

A simulation trace of the system obtained with USHVER is shown in Figure 4. The figure shows variables M (fuel mass in the tank) and T (fuel temperature in the tank) as a function of time. These simulations are just reported to provide an understanding of system works and the effect of a descreasing fuel level on the fuel temperature.

A simulation trace of the system obtained with USHVER is shown in Figure 4. The figure shows variables M (fuel mass in the tank) and T (fuel temperature in the tank) as a function of time. These simulations are just reported to provide an understanding of how the system works and the effect of a decreasing fuel level on the fuel temperature. At the same time, simulation traces can provide insights on how to abstract certain parts of the system. For example, the take-off and ascending phases account for a small fraction of the mission and could be abstracted away. It is interesting to observe also how the fuel temperature increases according to a non-linear law. However, for the first 3000 seconds the temperature profile appears to be fairly linear. Thus, it would be possible to approximate the fuel temperature dynamics by a clock variable $\dot{T} = c$, for $t \in [0, 3000]$, where c needs to be determined.

Figure 5.c shows the marginal probability distributions for the mass(M) and temperature (T) variables at the end of the mission computed using the reachability algorithms in USHVER . The discretization parameters for this case study can be found in [12]. The generation of the reachability results required roughly half an hour on a laptop with a Intel Core2 Duo CPU P9400 running at 2.40 GHz, and 2.95 GB RAM. Up to 3 million states were generated by the reachability computation.

6. CONCLUDING REMARKS

We have developed a tool called USHVER that allows modeling DTSHSs and performing simulation and analysis using different tools. We have used USHVER on a model of a thermal management system with 5 continuous variables and 7 modes. There are practical limitations in the adoption and use of the techniques and tools presented in this article:

- The size of systems that can be analyzed seems to be limited. The complexity arises from the branching factor in Algorithm1 which depends on the dynamics of the system and on the nature of the uncertainty.
- The technique used in USHVER provides some guarantees (see [8] for a review). However, explicit error bounds between the approximate probability measure and the real ones are not always available. The error depends on the selection of the partition and on the way in which the probabilities of the Markov model are computed. The user of USHVER may have to rely on empirical estimates to judge when the computed distributions are acceptable and stop refining the discretization grid. This is a limitation counterbalanced by the possibility of modeling general dynamics. The availability of other tools such as the Sequential Probability Ratio Test can also help in validating reachability results and guiding the user in assessing their quality.
- The finite partition is assigned manually and is uniform in each mode. This is an important limitation that impacts the adoption of the tool, the complexity and the quality of the results.
- The adoption of USHVER has been limited to internal research projects within the United Technologies Research Center. There are several adoption barriers. Stochastic hybrid system models of realistic systems are too complex for probabilistic analysis. The use of tools such as USHVER requires an initial step where an abstract model is constructed. The development of such model requires knowledge of the verification methods and of the system at the same time. This combination of skills is often difficult to find.

Areas for improvement for USHVER include the automatic generation of the grid and the computation of confidence bounds for the results generated by the reachability algorithm for which we could leverage the work in [3].

7. ACKNOWLEDGMENT OF SUPPORT AND DISCLAIMER

This material is based upon work supported by the United States Air Force under Contract No. FA9550-10-C-0116. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force.

8. REFERENCES

[1] "http://www.mrmc-tool.org."

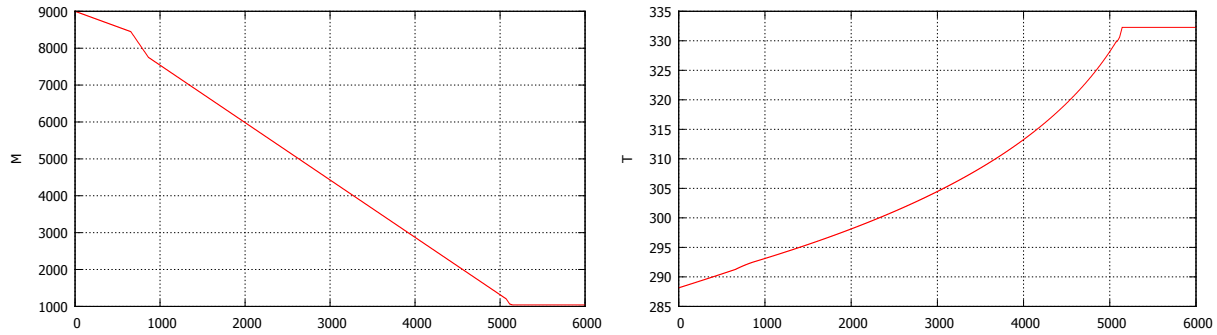
[2] "U.s. standard atmosphere," U.S. Government Printing Office, Tech. Rep., 1976. [Online]. Available: http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19770009539_1977009539.pdf

[3] A. Abate, A. D'Innocenzo, and M. D. Benedetto, "Approximate abstractions of stochastic hybrid systems," *IEEE Transactions on Automatic Control*, 2009.

[4] A. Abate, J. Katoen, J. Lygeros, and M. Prandini, "Approximate model checking of stochastic hybrid systems," *European Journal of Control*, 2010.

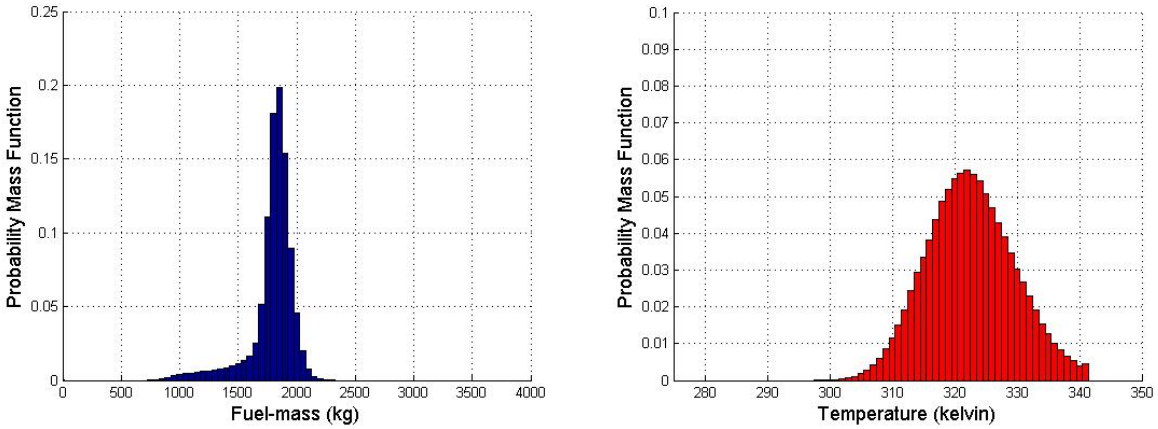
[5] A. Abate, "Probabilistic reachability for stochastic hybrid systems: Theory, computations, and applications," Ph.D. dissertation, University of California, Berkeley, November 2007. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/440.html

[6] M. Dellnitz and O. Junge, *Set Oriented Numerical Methods for Dynamical Systems*. World Scientific, 2002, pp. 221–264.



(a) Fuel mass profile (b) Fuel temperature profile

Figure 4: Results obtained using the USHVER **simulator**



(a) Fuel Mass (R) (b) Fuel temperature (R)

Figure 5: Results obtained using the USHVER **reachability analysis (R).**

[7] A. D'Innocenzo, A. Abate, M. D. D. Benedetto, and S. S. Sastry, "Approximate abstractions of discrete-time controlled stochastic hybrid systems," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, December 2008, pp. 221–226. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/443.html

[8] G. Froyland, "Extracting dynamical behaviour via markov models," in *Nonlinear Dynamics and Statistics*, A. Mees, Ed., Newon Institute, Cambridge. Birkhauser, 1998, pp. 283–324.

[9] D. Hartfiel, *Markov set-chains*, ser. Lecture notes in mathematics. Springer, 1998. [Online]. Available: http://books.google.com/books?id=79wZAQAAIAAJ

[10] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen, "The Ins and Outs of The Probabilistic Model Checker MRMC," in *Quantitative Evaluation of Systems (QEST)*. IEEE Computer Society, 2009, pp. 167–176, www.mrmc-tool.org.

[11] A. Lasota and M. C. Mackey, *Chaos, Fractals and Noise*, ser. Applied Mathematical Sciences. Springer, 1994.

[12] G. A. Mathew and A. Pinto, "Stochastic analysis and design of systems," DARPA V2D2 Study (#FA9550-10-C-0116) Final Report, Tech. Rep., 2011.

[13] J. Mattingly, W. Heiser, and D. Pratt, *Aircraft engine design*, ser. AIAA education series. American Institute of Aeronautics and Astronautics, 2002, no. v. 1. [Online]. Available: http://books.google.com/books?id=2Wy5rpdm3DMC

[14] D. Parker, "Implementation of symbolic model checking for probabilistic systems," Ph.D. dissertation, University of Birmingham, 2002.