A Modeling Framework for Prognostic Decision Making and its Application to UAV Mission Planning

Edward Balaban[1], Juan J. Alonso[2]

[1] *NASA Ames Research Center, Moffett Field, CA, 94035, USA*
edward.balaban@nasa.gov

[2] *Stanford University, Stanford, CA, 94305, USA*
jjalonso@stanford.edu

ABSTRACT

The goal of prognostic decision making (PDM) is to utilize information on anticipated system health changes in selecting future actions. One of the key challenges in PDM is finding a sufficiently expressive yet compact mathematical representation of the system for use with decision optimization algorithms. In this paper we describe a general modeling approach for a class of PDM problems with non-linear system degradation processes and uncertainties in state estimation, action effects, and future operating conditions. The approach is based on continuous Partially Observable Markov Decision Processes (POMDPs) used in conjunction with 'black box' system simulations. The proposed modeling framework can be cast into simpler representations, depending on which sources of uncertainty are being included. The approach is illustrated with a mission planning case study for an unmanned aerial vehicle (UAV). In the case study a PDM system is tasked with optimizing the vehicle route after an in-flight component fault is detected. A stochastic algorithm (based on particle filtering) is used for decision optimization, with a second, deterministic algorithm providing a performance evaluation baseline. Both algorithms utilize a UAV physics simulator for generating predictions of future vehicle states. Performance benchmarking is done on a set of mission scenarios of increasing complexity.

1. INTRODUCTION

Decision-making in complex aerospace applications (our area of interest) encompasses the selection of actions at numerous levels of system abstraction. At the lower levels decision making can mean selecting controller gain values in a subsystem or defining the allowable movement range for a control device. At the mission level decision making can involve modifications to the vehicle route. At the highest levels, decision making can extend to allocating assets for a mission from a fleet of vehicles or even reorganizing a logistics chain.

Prognostic Decision Making (PDM) can be defined as the process of selecting system actions informed by predictions of the future system health state. While in many respects PDM is similar to other planning and decision optimization problems (such as those from the fields of optimal control or path planning), there are two key reasons, in our opinion, to consider it as a distinct problem type. First, incorporating information, however imperfect, about the evolution of the system health parameters may help increase decision quality and reduce the state estimation uncertainty. Doing that may prove to be a non-trivial task, as health degradation processes in aircraft or spacecraft components often have complex dependencies on operating conditions, environmental factors, and the degradation processes occurring in the other parts of the system. Second, selecting actions appropriately may, in turn, help improve subsequent prognostic estimates. In nominal operations, incorporation of prognostics into the decision-making could help optimize vehicle performance and minimize maintenance costs. In situations where the vehicle is experiencing in-flight malfunctions, having the additional source of information provided by prognostic methods could be crucial to ensuring a safe mission outcome.

In order to make the prognostic decision making problem tractable for an aerospace application, it may need to be subdivided into smaller sub-problems, with the information exchange organized for overall decision coherency (Balaban & Alonso, 2012). The goal of the work described in this paper is to further develop modeling techniques and algorithms for solving the types of sub-problems that require representation of uncertainties in state estimation (including in payoffs/rewards), action outcomes, and future operating conditions. Our modeling approach is based on continuous Partially Observable Markov Decision Processes (Bertsekas, 1995). As a generalization of the more traditional state-space representations and Markov Decision Processes (Bellman, 1957b), the same POMDP framework can be used to describe problems with some or all of the aforementioned sources of uncertainty absent.

A mission replanning case study for an unmanned aerial vehicle is used as an illustrative example. It is extended from a mission replanning study for a ground vehicle (Balaban & Alonso, 2012). While the general ideas for the two studies are similar, additional challenges present themselves when implementing PDM for a flight vehicle. With motion now conducted in three dimensions (six degrees of freedom), the task of estimating a future vehicle state is inherently more complex. In addition, the system needs to produce its action recommendations as quickly as possible since, unlike a ground vehicle, an airplane cannot simply stop, enter a safe mode, and wait for the decision-making process to complete. The longer the process takes, the worse the fault condition may become and the further from the optimal flight path the vehicle may end up.

Exact solutions of problems formulated as POMDPs are generally not achievable (Pineau, Gordon, & Thrun, 2006), therefore some type of an approximate algorithm needs to be utilized. In this work an algorithm based on particle filtering (Gordon, Salmond, & Smith, 1993) is implemented, with a second, deterministic algorithm used for validation and performance assessment purposes. The latter algorithm is based on backtracking search (Knuth, 1968).

In the near term, the case study described in this work is meant to pave the way for PDM flight demonstrations on the Edge 540 electric UAV (Hogge, Quach, Vazquez, & Hill, 2011). The Swift electric UAV (Denney, Pai, & Habli, 2012), being developed at NASA Ames Research Center, is under consideration as an advanced follow-up test platform. The Swift is significantly larger than the Edge 540 (13.0 meter vs 2.45 meter wingspan) and can fly longer missions (7-8 hours aloft projected vs. approximately 0.3 hours for the Edge 540). The longer flight endurance is expected to allow for a greater variety of gradually developing fault modes to be introduced into the test scenarios.

The rest of the paper is organized as follows. Related research efforts are briefly reviewed in the next section. Section 3 describes our modeling methodology. The mission replanning case study is defined in Section 4. The prototype vehicle health management architecture used to implement the case study is described in Section 5. A UAV physics simulator (Section 6) is one of the components of this architecture. The simulator includes prognostic degradation models for some of the vehicle components. Section 7 presents the two decision-making algorithms (deterministic and stochastic), which are tested on a set of simulated missions of varying complexity (Section 8). Section 9 provides a summary of the work performed and outlines our plans for future research.

2. Related Work

This section highlights some of the research efforts that, we believe, form a representative sample of the current state of the art in PDM. Most of the PDM research to-date has been done for low-level component control, with a few projects using prognostics in solving higher-level planning and scheduling problems.

Pereira, Galvao, and Yoneyama (2010) developed a Model Predictive Control (MPC) system that distributes control effort among several redundant actuators based on prognostic health information. Damage accumulation is assumed to be linearly dependent on the exerted control effort. The approach is tested in simulation on a tank level regulation problem. Brown and Vachtsevanos (2011) also incorporate prognostic information into a model-predictive controller and apply it to optimizing performance (in simulation) of an electro-mechanical actuator. The work includes recommendations for error analysis and for estimation of uncertainty bounds in long-term Remaining Useful Life (RUL) predictions. Bogdanov, Chiu, Gokdere, and Vian (2006) investigate coupling of a prognostic lifetime model for servo motors with a family of LQR controllers.

Edwards, Orchard, Tang, Goebel, and Vachtsevanos (2010) propose a set of metrics to quantify the impact of input uncertainty on non-linear prognostic systems. The metrics are incorporated into a feedback correction loop in order to demonstrate RUL extension for a non-linear, non-Gaussian system (a helicopter gear plate experiencing a fatigue crack fault). An algorithm based on particle filtering is used for uncertainty estimation. The work done by Tang, Hettler, Zhang, and Decastro (2011) contains elements of prognostics-enhanced control, but also extends into prognostic path planning for an unmanned ground vehicle. RUL estimates were used either as a constraint or as an additional element in the cost function of the path-planning algorithm, with the algorithm based on Field D* search (Ferguson & Stentz, 2006). Methods for estimating and managing prediction uncertainty were also developed.

In our work we aim to build on the above efforts by exploring the benefits and the challenges of performing prognostic decision making in a more general framework, where the system model is treated as a 'black box'. Our approach to doing this is described next.

3. Modeling Approach

The mathematical modeling approach adapted for this work is meant to be general and applicable to a wide variety of systems with degradation processes. A POMDP formulation was chosen as the basis since it allows for transition costs, rewards, and action outcomes to be expressed in probabilistic terms. The main elements of a model are described below:

State space:	$S \subseteq \mathbb{R}^n$
Action space:	$A \subseteq \mathbb{R}^m$
Observations:	$Z \subseteq \mathbb{R}^p$
Transition function:	$T(s,a,s') = pdf(s'\|s,a) : S \times A \times S \rightarrow [0,\infty)$
Observation function:	$O(z',a,s') = pdf(z'\|s',a) : S \times A \times Z \rightarrow [0,\infty)$
Belief state:	$b(s) = pdf(s)$
Belief space:	B - the set of all belief states
Initial belief:	b_0
Belief update:	$b^{az}(s') \propto O(z',a,s') \int\limits_S T(s,a,s')b(s)ds$
Policy:	$\pi(a,b) = pdf(a\|b) : A \times B \rightarrow [0,\infty)$, Π is the set of all policies
Costs:	$\mathcal{C} = \{c_1(s,a), ..., c_{\|\mathcal{C}\|}(s,a)\} \subseteq \mathbb{R}^{\|\mathcal{C}\|}$
Cost functions:	$c_i(s,a,c) = pdf(c\|s,a) : S \times A \times \mathbb{R} \rightarrow [0,\infty)$, where c is a specific real value, $i \in \{1,\ldots,\|\mathcal{C}\|\}$
Rewards:	$\mathcal{R} = \{r_1(s), ..., r_{\|\mathcal{R}\|}(s)\} \subseteq \mathbb{R}^{\|\mathcal{R}\|}$
Reward functions:	$r_i(s,r) = pdf(r\|s) : S \times \mathbb{R} \rightarrow [0,\infty)$, where r is a specific real value, $i \in \{1,\ldots,\|\mathcal{R}\|\}$
Features:	$\mathcal{X} = \{x_1(s),\ldots,x_{\|\mathcal{X}\|}(s)\} : S \rightarrow \mathbb{R}^{\|\mathcal{X}\|}$
Constraints:	$\mathcal{G} = \{g_1(s),\ldots,g_{\|\mathcal{G}\|}(s)\} : S \rightarrow \mathbb{B}^{\|\mathcal{G}\|}$
Objective function:	$J^\pi(b_0) : \Pi \times B \rightarrow \mathbb{R}$

System states in S are not defined explicitly, but rather obtained through a 'black box' simulator, such as the one described in Section 6. The simulator takes an estimate of the current state and the desired action as inputs and generates a state estimate for the next time step as the output. Compared to discretized state representations, this approach helps to avoid the 'curse of dimensionality' (Bellman, 1957a), resulting from the exponential growth in the size of the state space with the number of state vector components. It also mitigates discretization errors. While a discretized formulation does not require a system simulator, we believe that employing such a formulation would quickly become impractical for all but the simplest PDM problems.

The concept of a continuously valued action space A allows us, in conjunction with the simulator, to represent not only the system actions, but the loads and environmental conditions imposed onto the system as well. Observations Z most often come from sensor readings, but can also be manual measurements of a quantity such as the length of a crack. The transition probability function $T(s,a,s')$ allows us to model the uncertainty in estimating the future loading conditions and their effects on the system state (s denotes the current state, a the action taken in the current state, and s' is the next state). The observation function $O(z',a,s')$ describes the probability of seeing an observation z' if a state s' is achieved as a result of an action a.

Since in real life it is often not possible to know the exact state the system is in, the belief state $b(s)$ is defined as the probability density of being in a particular state s. A policy $\pi(a,b)$ is then defined as the probability density of taking a specific action a given a belief state b. In practical terms, a policy allows us to define what actions the system should take given the belief distribution over possible states.

A cost is defined probabilistically as a value dependent on the current system state and the action taken in it. A cost can be the amount of energy needed to transition to the next system state or the corresponding measure of component wear. Rewards, on the other hand, depend only on the system state achieved. Reward functions define the desirability of a state as a probability density over the real domain. Features are also functions defined over state variables that enable reasoning over the various properties of a state. Unlike rewards, feature values are not meant to be used in an additive manner directly and are thus defined as point estimates.

Constraints are defined as inequalities of the form $g_i(s) \geq 0$ on the components of the state of the system. Strict equality constraints can be defined as well, although they are less useful in the current context. Constraints can be used to define regions of nominal and off-nominal system behavior, as will be done shortly for system health.

Finally, features, costs, and rewards are combined into an objective function J^π that, given an initial belief state b_0 and a policy π, can compute a single evaluation metric of that policy. Several objective functions can be defined for multicriteria reasoning. Given the predominantly non-negative transition costs in a typical PDM application, a separate discount factor (often used in value and objective functions to guarantee convergence and encourage shorter solutions) is likely unnecessary. Negative costs correspond to self-healing or commanded recovery events and, while certainly valid, should not dominate the non-negative costs (e.g. energy consumption or component wear), given a properly designed objective function. Terminal states are defined through the constraints in $\mathcal{G}$, including (but not limited to) constraints on vehicle health parameters or position.

Now that the modeling framework has been described, the key system health management concepts can be defined within it:

- **Health parameters**: $H = \{h_1, \ldots, h_H\}$ is a subset of state vector elements that describes the health state of the system.
- **Fault:** A subset of constraints $\mathcal{G}_{\mathit{fault}} \in \mathcal{G}$ defines significant deviations from the expected nominal behavior (with respect to vehicle health). A fault occurs if $\exists i, g_i(s) = false, g_i \in \mathcal{G}_{\mathit{fault}}$. At least some of the health parameters H are expected to be included in the constraints of $\mathcal{G}_{\mathit{fault}}$.
- **Failure:** Another subset $\mathcal{G}_{\mathit{failure}} \in \mathcal{G}$ defines states where the system loses its functional capability with respect to a health parameter $h \in H$.
- **System failure:** System failure is defined through a boolean function $\mathcal{F} : B \to \mathbb{B}$, which indicates when the entire system is believed to be effectively non-functional ($\mathcal{F}$ is defined via the $\mathcal{G}_{\mathit{failure}}$ set). System failure can be indicated when a single $g_i \in \mathcal{G}_{\mathit{failure}}$ is believed to be violated, for example, or a larger subset of them. More generally, the function could also be defined as $\mathcal{F} : B \to [0, 1]$, mapping the belief space to a probability of system failure.
- **End of Life (EoL)**: End of Life in this set of definitions is synonymous with system failure. Time of EoL is defined similarly to (Daigle, Bregon, & Roychoudhury, 2012) as $t_{EoL} \triangleq \inf\{t \in \mathbb{R} : (t \geq t_p) \wedge (\mathcal{F}(b) = true)\}$, where t_p is the time of prediction (or, alternatively, the present time).
- **Remaining Useful Life (RUL)**: $RUL = t_{EoL} - t_p$

The *diagnostic* problem then becomes the process of determining the current belief state, b_t. The *prognostic* problem can be stated as the process of determining, at time t, the belief state $b_{(t+\Delta t)}$, given the current policy π (with $\Delta t > 0$). The *prognostic decision making* problem is then the process of finding (or approximating) π^*, such that

$$\pi^* = \arg\max_{\pi \in \Pi} J^\pi(b_t),$$

where $J^\pi(b_t)$ is an objective function computed using a policy π given a state of belief b_t at time t. Time (or an alternative index) can be discretized.

Given this general framework for describing the system, its actions, and the state of system health, we can recast it depending upon the problem at hand. While in some circumstances representation and quantification of uncertainty is necessary to increase the decision accuracy, in others the downsides of having a random process present in the reasoning system outweigh the benefits. When in real-time use aboard an air vehicle, for example, running the simulation in the deterministic mode (supplying the expected value of the next state) may often be preferable. For the purposes of development and verification, it may also be beneficial to start with a deterministic state transition system. Enabling uncertainty in the effects of an action would turn the system into a continuous Markov Decision Process (MDP) (Bertsekas, 1995). Adding uncertainty in state estimation would turn it into a continuous POMDP.

While exact solutions for problems posed as deterministic state-space representations can be produced using a variety of search techniques, as the amount of uncertainty increases computing exact solutions becomes more and more challenging. In certain cases (linear system model, quadratic objective function) continuous MDP problems can be solved exactly using Differential Dynamic Programming (Jacobson & Mayne, 1970). In other cases approximate methods are required. The two algorithms described in Section 7, backtracking search and Particle Filter, can be used with both deterministic and stochastic next-state representations. The former will produce an exact solution if the states are defined deterministically, while the latter will generate approximate solutions in both cases.

4. Mission Replanning Case Study

Our case study is based on a UAV mission scenario where the vehicle is tasked with visiting a set of waypoints and returning back to the point of origin. Such a mission profile is typical of reconnaissance or geophysical survey missions, for example. For tasks of this type the order of visiting the waypoints is often less important than which waypoints were actually reached. Further scenario details are listed below.

Given:

- The waypoint set is defined as $wp = \{wp_i\}_{i=1}^N$, where a waypoint $wp_i = \{(x_i, y_i, z_i), \mathcal{G}_i, r_i\}$ is specified by its Cartesian coordinate vector (x_i, y_i, z_i), waypoint-specific constraints subset $\mathcal{G}_i$ (e.g. on airspeed or bank angle), and a reward value r_i.
- A path $p = (wp_j)_{j=1}^M$ is defined as an ordered subset of wp (i.e. $M \leq N$). P is the set of all possible paths.
- The aircraft starts its mission at waypoint wp_1 (home runway) and is required to return there for a mission to be considered a full or a partial success.
- An initial path p_0 is specified (p_0 is not necessarily optimized).
- $h_h \in [0, 1]$ is the normalized system health index (1 is full health and 0 indicates failure). The constraint on system health is defined as $g_h(s) = h_h \geq 0$.
- $h_e \in [0, 1]$ is the normalized remaining energy index (1 is full charge and 0 indicates depleted energy). The constraint on energy is defined as $g_e(s) = h_e \geq 0$.
- A healthy vehicle is able to complete p_0 within the energy and component health constraints (before either reaches 0).
- Energy and health transition costs between a pair of waypoints a and b are defined as $c_e(wp_a, wp_b)$

and $c_h(wp_a, wp_b)$, respectively (the costs are history-dependent). $\mathcal{C}_e(p)$ and $\mathcal{C}_h(p)$ are the corresponding cumulative costs for a path. Similarly, the cumulative path reward is denoted by $\mathcal{R}(p)$.

- A fault f occurs at a time t_f, that makes it impossible to complete p_0 before EoL. The fault magnitude changes with time, loading conditions, and environmental factors.

Find:

$$p^* = \arg\max_{p \in P} \mathcal{R}(p),$$

where p^* is a path that maximizes cumulative reward.

A waypoint wp_i is considered to be reached if the vehicle achieves a position $(x_i \pm \epsilon_x, y_i \pm \epsilon_y, z_i \pm \epsilon_z)$, with all of the applicable constraints satisfied. Here ϵ_x, ϵ_y, and ϵ_z are the predefined maximum position error values.

5. PHM Reasoning Architecture

The reasoning architecture used in this work consists of the four main components: the diagnoser (DX, for diagnostics), the decision maker (DM), the vehicle simulator (VS), and the vehicle itself (Figure 1).

The execution process is initiated with an input route p_0 and an initial fault set F_0 supplied to the decision maker. This can be done at the beginning of a mission (in that case the fault set may be empty) or if a fault is diagnosed in flight. A fault set F is one or more fault descriptors. A fault descriptor f_i consists of a fault type d and a fault magnitude m. Fault magnitude m is generally normalized to the interval $[0, 1]$.

DM utilizes VS to evaluate and, optionally, optimize the input route (according to the criteria in Section 4). An initial optimization can be performed if a successful completion of the input route is deemed unlikely. Alternative routes sent to VS consist of an ordered set of waypoints (with vectors Γ of specific values for waypoint parameters). DM also informs VS of the relevant fault modes. VS simulates the candidate path, p_c, and returns the reward and cost estimates for it. Once DM finalizes the route recommendation p^*, it is sent to the vehicle for execution.

As the mission proceeds, DX continues to monitor observations Z (e.g. sensor readings) generated by the vehicle to detect any new fault conditions. The current prototype implementation, meant to primarily test the PDM algorithms, includes only a minimal diagnostic functionality. Future versions will incorporate a fully-featured diagnoser, such as HyDE (Narasimhan & Brownston, 2007) or QED (Daigle & Roychoudhury, 2010). One of our long-term goals is to unify diagnostic and decision making processes around a single, comprehensive system model. For the near future, however, if the diagnoser utilized is model-based (as is the case for both HyDE and QED), a separate diagnostic model would need to be constructed.

A new fault condition triggers a reevaluation of the vehicle route by DM. Another event type that triggers a reevaluation is a significant deviation of the predicted component degradation rates from the rates observed as the flight progresses further. This mechanism is implemented by storing the predicted component degradation curves for the proposed route (p^*) before it is sent to the vehicle for execution.

6. Vehicle Simulator

Given an estimate of the current vehicle state and the desired action, the simulator is used to generate an estimate of the next state in its innermost loop. The state vector includes the aircraft position, velocity, acceleration, orientation, lift, drag, thrust, battery voltage, battery charge remaining, component temperatures, and other data. When an entire path p is provided, the simulator can generate reward and cost estimates $\mathcal{R}(p)$, $\mathcal{C}_h(p)$, and $\mathcal{C}_e(p)$. For that, the route is divided into segments where the key elements of vehicle dynamics can be considered constant (i.e. acceleration, bank angle, angle of climb/descent, etc). The segments are further divided into time steps. The size of a time step, dt, can be specified as one of the simulation parameters (it is adjusted automatically for segments where higher precision in position control is required, such as during take-offs and landings). As the simulator is primarily intended to provide estimates of transition costs and rewards for DM in a computationally efficient manner, it does not employ a closed loop controller to achieve precise trajectory following.

The aerodynamic forces are calculated in a right-handed, velocity-oriented, local-level frame of reference, with the origin at the center of mass of the aircraft. The position of the center of mass $\{x(t), y(t), z(t)\}$ is computed in an inertial right-handed Cartesian frame of reference, with the origin at the take-off/touch-down end of the runway. There are currently three elements of the vehicle model with non-linear, action-dependent degradation aspects: battery voltage, battery temperature, and motor temperature (subsections 6.5 and 6.6). The simulator is implemented in MATLAB (*MATLAB version 7.11.0.584 (2010b)*, 2010). The rest of the section provides further details of the simulator, describing the key equations and assumptions.

6.1. Lift and drag

Lift L and drag D are calculated as

$$L = \frac{1}{2}\rho_\infty v_\infty^2 S C_L, \tag{1}$$

$$D = \frac{1}{2}\rho_\infty v_\infty^2 S C_D, \tag{2}$$

$$C_D = C_{D,p} + C_{D,i}, \tag{3}$$

where L is lift, ρ_∞ is the freestream density, v_∞ is the freestream velocity, $C_{D,i}$is the coefficient of induced drag,

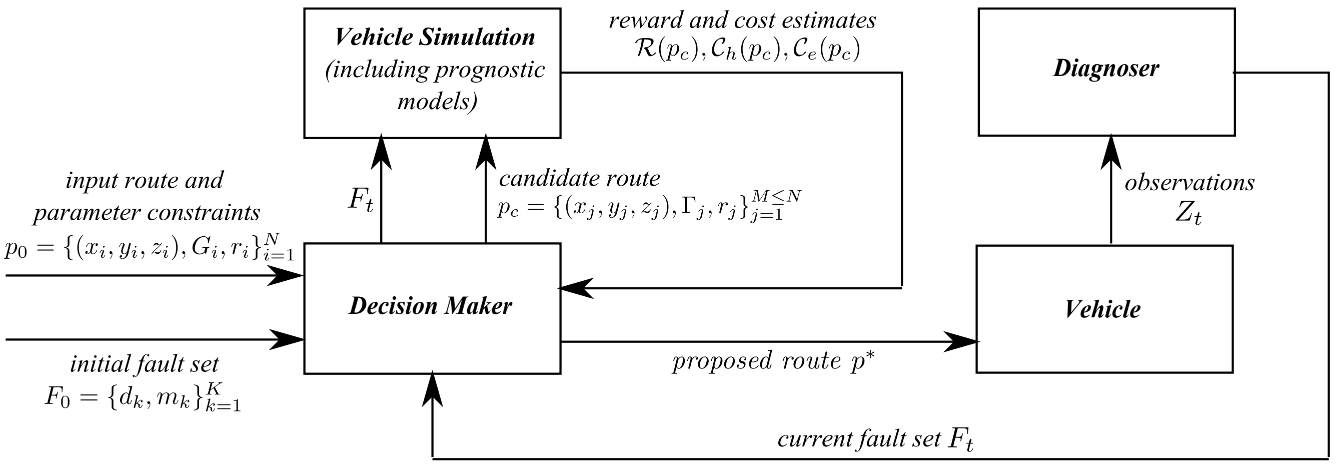
Vehicle Simulation (including prognostic models)
reward and cost estimates $\mathcal{R}(p_c), \mathcal{C}_h(p_c), \mathcal{C}_e(p_c)$
Diagnoser
input route and parameter constraints $p_0 = \{(x_i, y_i, z_i), G_i, r_i\}_{i=1}^N$
F_t
candidate route $p_c = \{(x_j, y_j, z_j), \Gamma_j, r_j\}_{j=1}^{M \leq N}$
observations Z_t
Decision Maker
Vehicle
initial fault set $F_0 = \{d_k, m_k\}_{k=1}^K$
proposed route p^*
current fault set F_t

Figure 1. Reasoning architecture

and $C_{D,p}$ is the coefficient of profile drag. Using Prandtl's Lifting Line theory:

$$C_{D,i} = \frac{C_l^2}{\pi e AR}, \tag{4}$$

where AR is the planform aspect ratio and e is the span efficiency factor. From (3) and (4):

$$C_D = C_{D,p} + \frac{C_l^2}{\pi e AR}, \tag{5}$$

$C_{D,p}(\alpha, Re)$ is obtained from experimental wind tunnel data for a similar airfoil (Miller, 2008; Lewis, 1984), with α denoting the angle of attack (in radians) and Re the Reynold's number. The thin airfoil assumption is made, thus

$$C_l = 2\pi\alpha. \tag{6}$$

6.2. Equations of motion for straight, accelerated flight

Force balance equations in x and z directions give

$$\ddot{x} = \frac{1}{m}(T - D) - g\sin\gamma, \tag{7}$$

$$\ddot{z} = \frac{1}{m}L - g\cos\gamma, \tag{8}$$

where γ is the flight path angle, g is the acceleration of gravity, and m is the mass of the aircraft.

6.3. Equations of motion for turning flight

Assume a constant airspeed (i.e. $dv/dt = 0$, v here equivalent to the freestream velocity), balanced flight (no skidding or slipping) during turns. From the equation of equilibrium in the horizontal direction:

$$F_c = L\sin\phi\cos\gamma = \frac{v^2}{r}m, \tag{9}$$

where ϕ is the bank angle, r is the turn radius, and F_c is the centrifugal force. Then

$$r = \frac{v^2}{L\sin\phi\cos\gamma}m. \tag{10}$$

From the vertical equation of equilibrium, the lift required is now

$$L = mg\frac{\cos\gamma}{\cos\phi}, \tag{11}$$

Considering the trajectory of a generalized climbing (or descending) turn to be a helix, the x, y, z coordinates are parametrized as follows:

$$x(t) = r\cos(t), \tag{12}$$

$$y(t) = r\sin(t), \tag{13}$$

$$z(t) = 2r\sin(\gamma)t. \tag{14}$$

6.4. Power and current

The power P required to generate the desired thrust T is expressed as

$$P = \frac{1}{\eta_p}Tv \text{ and} \tag{15}$$

$$P = \eta_m E\frac{dq}{dt}, \tag{16}$$

where q is the battery charge required, E is the battery voltage, and η_p and η_m are the propeller and motor efficiency coefficients, respectively. Charge q can then be expressed as

$$dq = \frac{1}{\eta_t\eta_m\eta_p}\frac{Tv}{E}\,dt, \tag{17}$$

The electrical transmission efficiency coefficient is denoted by η_t. Propeller efficiency for a specific airspeed is currently approximated as a quadratic function, with the maximum efficiency achieved at cruise airspeed v_c is

$$\eta_p = \eta_{p,max} \left(-\left(\frac{v}{v_c}\right)^2 + 2\frac{v}{v_c} \right). \tag{18}$$

In the future versions of the simulator equation 18 will be replaced with experimentally-derived curves for the propellers used.

6.5. Battery charge and voltage at the terminals

The battery model is adapted from the work described by Daigle, Saxena, and Goebel (2012), which, in turn, is based on battery models by Barsali and Ceraolo (2002), M. Chen and Rincon-Mora (2006), and Saha, Quach, and Goebel (2012). Three main processes are captured in the model: the *ohmic drop* (also known as the *I-R drop*), the *parasitic resistance* (accounting for self-discharge), and the *concentration polarization*. Out of the three, the concentration polarization resistance is the primary contributor to the non-linearity of the battery output voltage as a function of its state of charge. The state of charge (SoC) is defined as

$$SoC = 1 - \frac{q_{max} - q_b}{C_{max}}, \tag{19}$$

where q_b is the current charge in the battery (assumed to be held by capacitance C_b), q_{max} is the maximum possible charge, and C_{max} is the maximum possible capacity. The concentration polarization resistance is expressed as:

$$R_{CP} = R_{CP0} + R_{CP1} \exp\left(R_{CP2}(1 - SoC)\right), \tag{20}$$

where R_{CP0}, R_{CP1}, and R_{CP2} are empirical parameters. The resistance (and, consequently, the voltage drop at the battery terminals) increases exponentially as SoC decreases (Saha et al., 2012). End of discharge is considered to have occurred when the voltage drops below a predefined threshold. Further details of the model can be found in (Daigle, Saxena, & Goebel, 2012).

6.6. Battery and motor temperatures

Being able to predict battery temperatures is important for the following reasons: (a) temperature influences the internal battery resistance (although this effect is, at present, not included into the model) and (b) excessive temperatures can lead to premature capacity degradation and, beyond a certain point, to thermal runaway and battery failure (Y. Chen, Song, & Evans, 1996). In an electric motor overheating can also result in failure due to stator winding insulation damage or rotor magnet delamination (Milanfar & Lang, 1996). Battery and motor temperatures are currently estimated with the following simple model:

$$dT = \frac{1}{C_t}(RI^2 + h(T_a - T))dt, \tag{21}$$

where R is the electrical resistance of the component, C_t is the thermal inertia coefficient, h is the thermal transfer coefficient, I is the current, T is the component temperature, and T_a is the ambient temperature.

The thermal transfer coefficient h is estimated using the assumption that the motor and battery cooling systems are designed to keep these components operating at some nominal temperatures T_o in straight and level cruise flight, with I_c current being drawn from the batteries. Given these conditions, $dT \approx 0$, thus

$$h = \frac{RI_c^2}{T_o - T_s}, \tag{22}$$

where T_s is the standard ambient temperature for which the cooling system was calibrated. C_t is estimated empirically.

7. Algorithms

For the case study described in Section 4, the model is recast as a constraint-satisfaction problem, with states under consideration concentrated in multi-dimensional clusters representing the waypoint coordinate vicinities, acceptable airspeed and bank angle ranges, possible system health conditions, etc.

7.1. Backtracking Search

Backtracking Search (BT) is a recursive, depth-first algorithm that, in our case, attempts to sequentially build up a path p from a set of waypoints $\{wp_i\}_{i=1}^{N}$ until one or more of the constraints in $\mathcal{G}$ are violated (Algorithm 1). When that happens, the algorithm will back up to the last known nominal state and attempt to build the rest of the path through a different set of waypoints. The algorithm keeps track of the best objective function value found, J^*, and the path that produced it, p^*, returning them after the search is complete (for this algorithm $J = \mathcal{R}$). If the system simulator is used in the deterministic mode, then the algorithm is guaranteed to find the optimal solution (or solutions). For the purposes of benchmarking, the deterministic mode is used, with specific values assigned to waypoint parameters. With this the worst case computational complexity is $O(N!)$, where N is the number of waypoints. Unlike an exhaustive search algorithm, however, BT is capable of skipping regions of search space where constraints are violated, which, in practice, results in a significant execution speed up.

7.2. Particle Filter

Particle Filter algorithms (also known as Sequential Monte Carlo algorithms) are a family of non-Gaussian/non-linear methods for approximating posterior distributions in partially observable, controllable Markov Chains (where time is discretized). While often used for system state estimation and prediction (Orchard, 2007), the general approach is suitable for other appropriately formulated problems and in our case is applied to vehicle path selection.

Algorithm 1 BT

```
1: inputs: $p, p^*, J^*, \{wp_i\}_{i=1}^N$
2: outputs: $p^*, J^*$
3: if $|p| > N$ then
4:     return
5: end if
6: $\{b, \mathcal{R}, \mathcal{C}_h, \mathcal{C}_p\} \leftarrow$ simulate($p_{test}$)
7: if $\mathcal{F}(b) = false$ then
8:     $J \leftarrow R$,
9:     if $J > J^*$ then
10:        $J^* \leftarrow J$
11:        $p^* \leftarrow p$
12:    else
13:        return
14:    end if
15: end if
                        ▷ recurse through the remaining waypoints
16: for $i \leftarrow 1, N$ do
17:    if $wp_i \notin p$ then
18:        BT($b, \{p, wp_i\}, p^*, \{wp_i\}_{i=1}^N$)
19:    end if
20: end for
```

The PF algorithm (Algorithm 2) is initialized with a set of k particles, each particle p_i containing the starting waypoint wp_1 and assigned the weight of $w_i = 1/k$. During each of the iterations of the algorithm (and for each particle), the path associated with a particle is sampled randomly out of the set of unvisited waypoints up to the maximum length of N. Each sample is tested in the simulator and the particle weight updated proportionally to the objective function value (now incorporating costs in addition to rewards). Unless system failure is believed to be likely for even the shortest path extensions, the particle path is extended by one waypoint (the first one in the randomized remaining waypoints set τ).

The number of algorithm iterations, D, is equal to N for the deterministic simulator mode and can be set to $D > N$ otherwise, to help prevent potentially promising particles from being ruled out too early. The highest weight particle is identified and stored after each iteration, to enable interruptibility. Particle weights are then normalized and the particles are resampled. The overall computational complexity of the algorithm is $O(N^2)$.

8. Experiments

For the experiments described, the physics simulator (Section 6) was used with the parameters for the Edge 540 electric UAV (Figure 2). The Edge 540 is 2.44 m long, with a 2.54 m wingspan (Hogge et al., 2011). It is powered by two electric motors connected in series through a single drive shaft. The motors drive a 0.66 m two-bladed propeller. The current for the motors is supplied by a set of four Li-Poly rechargeable batteries, which can store a total of 43200 coulomb. The average flight time is approximately 20 minutes.

Algorithm 2 PF

```
1: inputs: $\{wp_i\}_{i=1}^N, K$
2: outputs: $p^*$
3: $p_1, \ldots, p_K \leftarrow \{wp_1\}$
4: $w_1, \ldots, w_K \leftarrow 1/k$
5: for $d \leftarrow 1, D$ do
6:     for $k \leftarrow 1, K$ do
7:         $\tau \leftarrow$ permute($\{wp_i\}_{i=1}^N - p_k$)
8:         $l \leftarrow -1$
9:         repeat
10:            $l \leftarrow l + 1$
11:            $p_{test} = \{p_k, \{wp_1, \ldots, wp_l\}\}$
12:            $\{b, \mathcal{R}, \mathcal{C}_h, \mathcal{C}_p\} \leftarrow$ simulate($p_{test}$)
13:            $w_k \leftarrow \Theta^T \cdot \{\mathcal{R}, -\mathcal{C}_h, -\mathcal{C}_p\}$,
14:        until $\mathcal{F}(b) = true$
15:        if $l \geq 1$ then
16:            $p_k \leftarrow \{p_k, \{wp_1\}_\tau\}$
17:        end if
18:    end for
19:    $j \leftarrow \arg\max_j w_j$
20:    $p^* \leftarrow p_j$
21:    $\{w_1, ..., w_K\} \leftarrow \{w_1, ..., w_K\} / \sum_{i=1}^K w_i$
22:    $\{p_1, ..., p_K\} \leftarrow resample(\{p_1, ..., p_K\}, \{w_1, ..., w_K\})$
23: end for
```

8.1. Setup

A set W of ten sequentially numbered waypoints is created, with each waypoint associated with a reward value (Table 1 and Figure 3). Test scenarios with progressively increasing numbers of waypoints are then created (the 7-waypoint scenario contains waypoints 1 through 7, the 8-waypoint scenario contains waypoints 1 through 8, and so on). As the UAV transitions between waypoints 2 and 3 (in the original, unoptimized order), a fault is injected into one of the motors, m_2. The motor loses power, however the rotor can still spin. The fault has the following consequences on the aircraft performance:

Figure 2. Edge 540 UAV (courtesy of NASA LARC)

- The available maximum thrust is reduced;
- There is now a parasitic mechanical load on the remaining motor, m_1, which not only has to provide propulsion, but also spin the rotor of m_2 (there is no disconnect mechanism);
- As a consequence, m_1 draws more current in order to execute the intended flight path;
- Increase in current results in heat build-up inside the motor housing which is not sufficiently dissipated by normal cooling mechanisms;
- The heat, if allowed to build up, will eventually raise the internal temperature of m_1 to the level where the motor windings insulation begins to melt and a short circuit can occur, irreversibly damaging the motor (assumed to be $70°C$, as measured on the motor housing);
- The increased amount of current required to drive the remaining motor also means a higher rate of battery discharge and a higher rate of heat build-up inside the battery;
- Reducing the airspeed to decrease the motor current would increase the traverse time and, below a certain threshold, result in an aerodynamic stall;

The fault is injected by changing the motor efficiency coefficient η_{m_1}(nominally 0.85, reduced to 0.4). The waypoints are selected in such a way as to make it impossible for the UAV to visit all of them in the original order before either energy depletion or vehicle health deterioration beyond the point of failure. DM is then expected to rearrange and/or reduce the set of waypoints to maximize the mission payoff (i.e. find an optimal path), while remaining within the constraints on energy consumption and vehicle health degradation:

$$p^* = \arg\max_{p \in P} \mathcal{R}(p),\ s.t.\ \mathcal{C}_e(p) \geq 0, \mathcal{C}_h(p) \geq 0$$

The waypoint ordering matters because load profiles associated with the different routes may affect the degrading component and the energy consumption differently. Scenarios with greater numbers of waypoints offer the algorithms more choices on how to maximize mission payoff, but with the choices comes an increase in computational time.

Note that for this case study alternative strategies of handling the fault (such as switching between powered and gliding flight to let the motor cool down) are not considered.

Experiments were conducted by running the two algorithms on the same set of scenarios and recording the reward values achieved and the number of simulation calls made. BT search was executed once for each scenario, while PF was run 30 times per scenario. The results for the latter were averaged and the standard deviation of them was computed. The vehicle simulator was used in the deterministic mode, in order to enable comparison between the two algorithms.

Table 1. Waypoint parameters

	$x(m)$	$y(m)$	$z(m)$	$V(m/s)$	$\phi(deg)$	r
1	0	0	0	30	20	41
2	500	2500	450	30	10	52
3	4500	4200	900	35	20	60
4	3800	1440	550	35	10	71
5	500	-1500	750	45	25	39
6	-2300	-500	850	40	30	46
7	1100	4000	400	30	20	33
8	1800	-700	700	30	25	95
9	1200	6200	500	25	30	85
10	150	5200	600	40	15	30

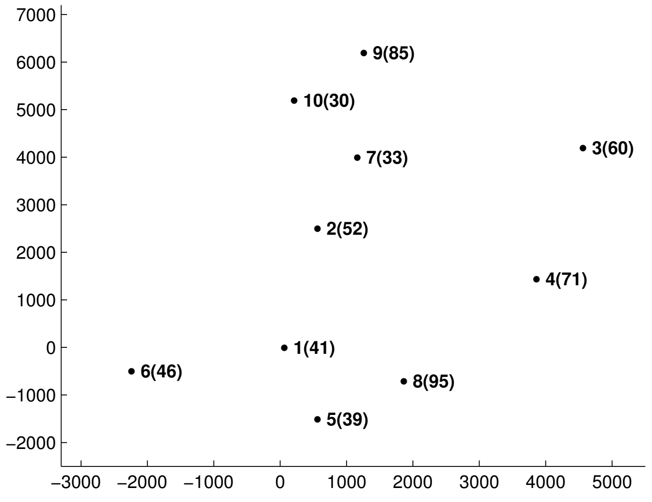

Figure 3. Waypoint locations and reward values

8.2. Results

In the result tables below (Tables 2-5), $\mathcal{R}$ is the path reward and n is the number of simulator calls made during a particular scenario. The number of simulator calls, rather than the actual algorithm runtime, is used as a platform-independent metric*. In the case of PF, mean and standard deviation values are provided for $\mathcal{R}$ and n. Ratios $\mathcal{R}/n$ and $\mu_{\mathcal{R}}/\mu_n$ are used as metrics of algorithm efficiency.

Backtracking Search

As can be seen in Table 2, BT search starts becoming impractical for the more complex scenarios, as the number of simulation calls grows exponentially. Still, the results serve as a useful evaluation baseline for PF, providing the maximum reward values achievable for a scenario.

*As a point of reference, a simulator call during a 10-waypoint scenario ($dt = 5$ seconds) typically took on the order of $0.005 - 0.010$ seconds to complete on a system with an Intel i7-2620M dual core CPU (2.70 GHz), with 8 Gb of RAM, running Windows 7 Professional (64-bit).

Table 2. Backtracking search results

$scenario$	$\mathcal{R}$	n	$\mathcal{R}/n$
7 $waypoints$	236	57	4.14
8 $waypoints$	298	241	1.23
9 $waypoints$	383	1598	0.24
10 $waypoints$	413	10117	0.04

Particle Filter

Since PF is stochastic, the algorithm was executed 30 times for each scenario and each K (number of particles) setting. The results were then averaged. Three K settings were used: 10, 20, and 30 (tables 3, 4, and 5, respectively). The average reward performance of the algorithm (and its standard deviation) steadily improve as the number of particles is increased (second column). The performance increase obviously comes at an additional computational cost (third column).

Table 3. Particle Filter results (K=10)

$scenario$	$\mu_{\mathcal{R}}(\sigma_{\mathcal{R}})$	$\mu_n(\sigma_n)$	$\mu_{\mathcal{R}}/\mu_n$
7 $waypoints$	230.2(12.8)	145.6(8.4)	1.58
8 $waypoints$	285.5(17.4)	199.2(13.8)	1.43
9 $waypoints$	368.6(21.7)	292.0(18.7)	1.26
10 $waypoints$	382.9(39.9)	331.2(20.5)	1.16

Table 4. Particle Filter results (K=20)

$scenario$	$\mu_{\mathcal{R}}(\sigma_{\mathcal{R}})$	$\mu_n(\sigma_n)$	$\mu_{\mathcal{R}}/\mu_n$
7 $waypoints$	232.4(2.5)	288.4(10.2)	0.80
8 $waypoints$	291.3(8.3)	391.2(15.8)	0.74
9 $waypoints$	372.3(14.6)	585.9(20.2)	0.63
10 $waypoints$	393.2(21.5)	670.2(29.6)	0.59

Table 5. Particle Filter results (K=30)

$scenario$	$\mu_{\mathcal{R}}(\sigma_{\mathcal{R}})$	$\mu_n(\sigma_n)$	$\mu_{\mathcal{R}}/\mu_n$
7 $waypoints$	234.0(1.1)	440.7(11.5)	0.53
8 $waypoints$	293.8(5.7)	603.8(21.6)	0.49
9 $waypoints$	375.2(10.2)	855.4(35.8)	0.44
10 $waypoints$	399.3(15.6)	1012.6(60.1)	0.39

8.3. Analysis

A comparison of algorithm performance with respect to reward values is provided in Figure 4. The dashed lines represent the reward benchmark set by BT, the dots depict the average PF rewards, and the error bars show the corresponding standard deviations. The algorithm, in general, performed well in approximating the results of the exact BT algorithm, but at a fraction of its number of simulation calls during the more complex scenarios. Even with $K = 10$ PF on average achieved over 92% of the maximum possible rewards.

Figure 5 shows a comparison with respect to the number of simulation calls. The dashed lines, again, correspond to the BT performance benchmark, the dots are the average numbers of calls for the PF algorithm, and the error bars show the standard deviations of the latter.

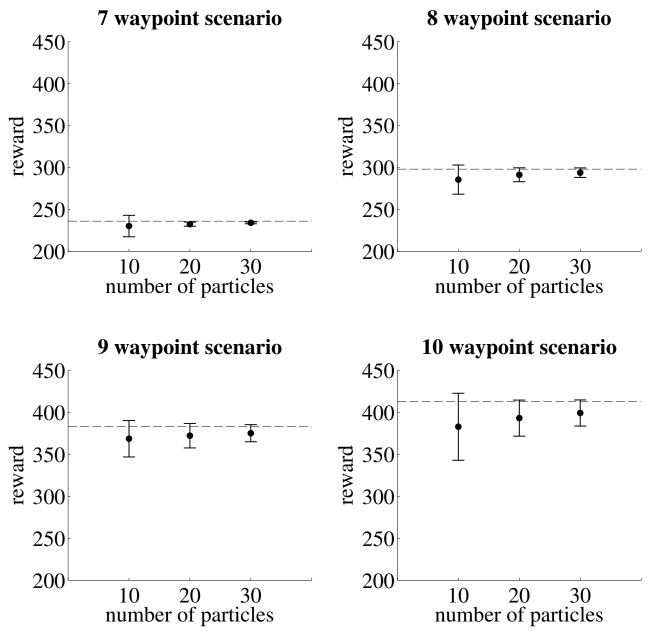

Figure 4. Reward value comparison

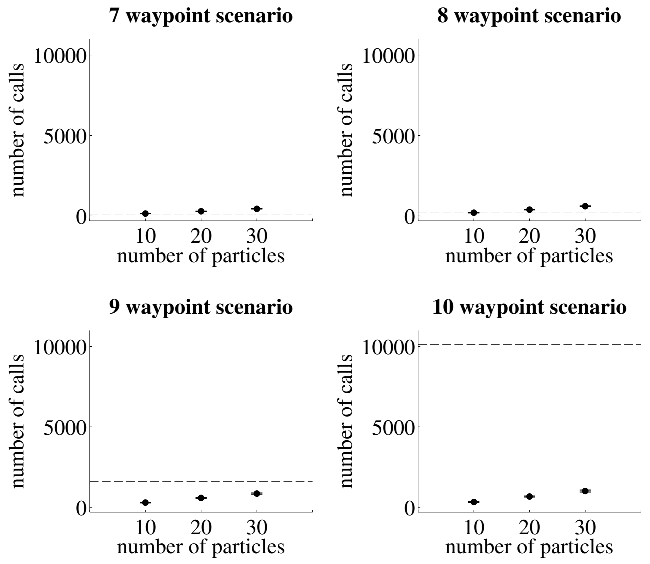

Figure 5. Number of simulation calls comparison

Even when using a fairly coarse 5-second time step in the simulator, executing BT on scenarios larger than ten waypoints became impractical. PF, on the other hand, showed the potential to be suitable for far more complex scenarios (informal tests performed on 25-waypoint scenarios still resulted in acceptable execution times, although the quality of the solutions could not be independently verified at this time).

9. Concluding Remarks

The main goal of this work is to describe a modeling framework suitable for a class of aerospace prognostic decision making (PDM) problems with complex dynamics, non-linear degradation processes, and uncertainties present in state estimation, action outcomes, and future operating conditions. Continuous Partially Observable Markov Decision Processes (POMDPs) are chosen as the foundation for the framework, with the paper providing the details on how the concepts needed for implementation of PDM are represented within it. The framework can be recast (simplified) depending on which sources of uncertainty are desired to be included. A case study serving as an illustration of the approach involves replanning a UAV mission after an in-flight fault is detected. A prototype PDM system takes into account the dependency of energy consumption and component degradation rates on the route chosen and attempts to maximize the value of the mission, while prioritizing the safe return of the aircraft. The system is demonstrated with a deterministic decision-making algorithm based on backtracking search (BT) and a stochastic algorithm based on particle filtering (PF). While likely too computationally expensive for practical applications, the BT algorithm is used to establish a performance baseline. The PF algorithm serves as an example of how a stochastic algorithm can be structured to utilize a continuous POMDP PDM model and that, potentially, such an algorithm can achieve a level of performance approaching that of an exact method, but at a fraction of its computational cost. On a set of test scenarios, the PF algorithm on average achieved over 92% of BT reward values. As the scenario complexity increased, the efficiency advantage of a stochastic algorithm became more pronounced. On the 10-waypoint scenario the PF algorithm achieved the above results while making an order of magnitude fewer calls to the vehicle simulator compared to BT.

In the near future we plan to incorporate measures of prognostic uncertainty associated with a particular policy into the policy selection process. This could be useful in situations where reduction of prognostics uncertainty over a period of time is desired or where limits on it are defined. Methods for performance evaluation of algorithms using such measures will need to be identified, as comparison against deterministic algorithms may not be meaningful. Other directions of future research include development of multi-fidelity reasoning (where system model complexity is adjusted depending upon the requirements and the circumstances) and further work on methods for PDM problem decomposition.

Acknowledgment

The authors would like to thank their colleagues at NASA Ames Research Center and Stanford University for their assistance with performing this work and preparing the manuscript for publication. Funding for this work is provided by NASA System-Wide Safety Assurance Technologies (SSAT) project (Aviation Safety Program).

Nomenclature

α	angle of attack
η_m	motor efficiency coefficient
η_p	propeller efficiency coefficient
η_t	power transmission efficiency coefficient
ϕ	bank angle
ρ_∞	freestream density
AR	planform aspect ratio
C_D	coefficient of drag (total)
C_t	thermal inertia coefficient
$C_{D,i}$	coefficient of induced drag
$C_{D,p}$	profile (airfoil) drag coefficient
D	drag
E	battery voltage
e	span efficiency factor
h	thermal transfer coefficient
I	current
L	lift
m	mass
P	power produced by the motor
q	battery charge required
R	component electrical resistance
r	radius of turn
Re	Reynolds number
S	planform area
T	thrust
T_a	ambient temperature
$T_{[c]}$	component temperature
v	airspeed
v_c	cruise airspeed
v_∞	freestream velocity
DM	Decision Maker
DX	Diagnostics (or Diagnoser)
EoL	End of Life
MDP	Markov Decision Process
PDM	Prognostic Decision Making
POMDP	Partially Observable Markov Decision Process
RUL	Remaining Useful Life
SoC	State of Charge (battery)
VS	Vehicle Simulation

REFERENCES

Balaban, E., & Alonso, J. J. (2012). An Approach to Prognostic Decision Making in the Aerospace Domain. In *Annual conference of the prognostics and health management society.* Minneapolis, MN.

Barsali, S., & Ceraolo, M. (2002, March). Dynamical Models of Lead-Acid Batteries: Implementation Issues. *IEEE Transactions on Energy Conversion*, *17*(1), 16–23.

Bellman, R. (1957a). *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Bellman, R. (1957b). *A Markovian Decision Process* (Tech. Rep.). Santa Monica, CA: RAND Corporation.

Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*. Athena Scientific.

Bogdanov, A., Chiu, S., Gokdere, L. U., & Vian, J. (2006). Stochastic Optimal Control of a Servo Motor with a Lifetime Constraint. In *Proceedings of the 45th ieee conference on decision and control.*

Brown, D. W., & Vachtsevanos, G. J. (2011). A Prognostic Health Management Based Framework for Fault-Tolerant Control. In *Annual conference of the prognostics and health management society.* Montreal, Canada.

Chen, M., & Rincon-Mora, G. (2006). Accurate Electrical Battery Model Capable of Predicting Runtime and I-V Performance. *IEEE Transactions on Energy Conversion*, *21*(2), 504–511.

Chen, Y., Song, L., & Evans, J. W. (1996). Modeling Studies On Battery Thermal Behaviour, Thermal Runaway, Thermal Management, and Energy Efficiency. In *Proceedings of the 31st intersociety energy conversion engineering conference (iecec).*

Daigle, M. J., Bregon, A., & Roychoudhury, I. (2012). A Distributed Approach to System-Level Prognostics. In *Annual conference of the prognostics and health management society.* Montreal, Canada.

Daigle, M. J., & Roychoudhury, I. (2010). Qualitative Event-Based Diagnosis: Case Study on the Second International Diagnostic Competition. In *21st international workshop on principles of diagnosis.*

Daigle, M. J., Saxena, A., & Goebel, K. (2012). An Efficient Deterministic Approach to Model-based Prediction Uncertainty Estimation. *Annual Conference of the Prognostics and Health Management Society*.

Denney, E., Pai, G., & Habli, I. (2012). Perspectives on Software Safety Case Development for Unmanned Aircraft. In *Ieee/ifip international conference on dependable systems and networks.* Boston, MA.

Edwards, D., Orchard, M. E., Tang, L., Goebel, K., & Vachtsevanos, G. (2010). Impact of Input Uncertainty on Failure Prognostic Algorithms: Extending the Remaining Useful Life of Nonlinear Systems. In *Annual conference of the prognostics and health management society.* Portland, OR.

Ferguson, D., & Stentz, A. (2006). Using Interpolation to Improve Path Planning: The Field D* Algorithm. *Journal of Field Robotics*, *23*(2), 79–101.

Gordon, N., Salmond, D., & Smith, A. (1993). Novel Approach to Nonlinear/non-Gaussian Bayesian State Estimation. *IEE Proceedings F (Radar and Signal Processing)*, *140*(2), 107–113.

Hogge, E., Quach, C., Vazquez, S., & Hill, B. (2011). *A Data System for a Rapid Evaluation Class of Subscale Aerial Vehicle, NASA/TM2011-217145* (Tech. Rep.). Hampton, VA.

Jacobson, D. H., & Mayne, D. Q. (1970). *Differential Dynamic Programming*. American Elsevier.

Knuth, D. E. (1968). *The Art of Computer Programming*. Addison-Wesley.

Lewis, K. W. (1984). *The Cumulative Effects of Roughness and Reynolds Number on NACA 0015 Airfoil Section Characteristics* (Tech. Rep.). Texas Tech University.

MATLAB version 7.11.0.584 (2010b). (2010). Natick, Massachusetts: The MathWorks Inc.

Milanfar, P., & Lang, J. (1996). Monitoring the Thermal Condition of Permanent-Magnet Synchronous Motors. *IEEE Transactions On Aerospace And Electronic Systems*, *32*(4), 1421–1429.

Miller, S. (2008). *Lift, Drag, and Moment of a NACA 0015 Airfoil* (Tech. Rep.). Ohio State University.

Narasimhan, S., & Brownston, L. (2007). HyDE - A General Framework for Stochastic and Hybrid Model-based Diagnosis. In *Proceedings of the international workshop on the principles of diagnosis.* Nashville, TN.

Orchard, M. E. (2007). *A Particle Filtering-based Framework for On-line Fault Diagnosis and Failure Prognosis*. Doctoral thesis, Georgia Institute of Technology.

Pereira, E. B., Galvao, R. K. H., & Yoneyama, T. (2010). Model Predictive Control using Prognosis and Health Monitoring of Actuators. In *2010 ieee international symposium on industrial electronics.*

Pineau, J., Gordon, G., & Thrun, S. (2006). Anytime Point-Based Approximations for Large POMDPs. *Journal of Artificial Intelligence Research*, *27*, 335–380.

Saha, B., Quach, C., & Goebel, K. (2012). Optimizing Battery Life for Electric UAVs using a Bayesian Framework. In *2012 ieee aerospace conference.*

Tang, L., Hettler, E., Zhang, B., & Decastro, J. (2011). A Testbed for Real-Time Autonomous Vehicle PHM and Contingency Management Applications. In *Annual conference of the prognostics and health management society.*