A Mobile Robot Testbed for Prognostics-Enabled Autonomous Decision Making

Edward Balaban[1], Sriram Narasimhan[2], Matthew Daigle[3], José Celaya[4], Indranil Roychoudhury[5], Bhaskar Saha[6], Sankalita Saha[7], and Kai Goebel[8]

[1,8]*NASA Ames Research Center, Moffett Field, CA, 94035, USA*
edward.balaban@nasa.gov
kai.goebel@nasa.gov

[2,3]*University of California, Santa Cruz, NASA Ames Research Center, Moffett Field, CA, 94035, USA*
sriram.narasimhan-1@nasa.gov
matthew.j.daigle@nasa.gov

[4, 5]*SGT Inc., NASA Ames Research Center, Moffett Field, CA, 94035, USA*
jose.r.celaya@nasa.gov
indranil.roychoudhury@nasa.gov

[6,7]*MCT Inc.., NASA Ames Research Center, Moffett Field, CA, 94035, USA*
bhaskar.saha@nasa.gov
sankalita.saha-1@nasa.gov

ABSTRACT

The ability to utilize prognostic system health information in operational decision making, especially when fused with information about future operational, environmental, and mission requirements, is becoming desirable for both manned and unmanned aerospace vehicles. A vehicle capable of evaluating its own health state and making (or assisting the crew in making) decisions with respect to its system health evolution over time will be able to go further and accomplish more mission objectives than a vehicle fully dependent on human control. This paper describes the development of a hardware testbed for integration and testing of prognostics-enabled decision making technologies. Although the testbed is based on a planetary rover platform (K11), the algorithms being developed on it are expected to be applicable to a variety of aerospace vehicle types, from unmanned aerial vehicles and deep space probes to manned aircraft and spacecraft. A variety of injectable fault modes is being investigated for electrical, mechanical, and power subsystems of the testbed. A software simulator of the K11 has been developed, for both nominal and off-nominal operating modes, which allows prototyping and validation of algorithms prior to their deployment on hardware. The simulator can also aid in the decision-making process. The testbed is designed to have interfaces that allow reasoning software to be integrated and tested quickly, making it possible to evaluate and compare algorithms of various types and from different sources. Currently, algorithms developed (or being developed) at NASA Ames - a diagnostic system, a prognostic system, a decision-making module, a planner, and an executive - are being used to complete the software architecture and validate design of the testbed.

1. INTRODUCTION

Over the last several years, testbeds have been constructed at NASA and elsewhere for the purpose of diagnostic and prognostic research on components important to aerospace vehicles: electronics, actuators, batteries, and others. For examples, please refer to (Poll, et al., 2007), (Smith, et al., 2009), (Balaban, Saxena, Narasimhan, Roychoudhury, Goebel, & Koopmans, 2010), However, there still remained a need for a testbed that supported development of algorithms performing reasoning on both the component and system levels, and optimizing decision-making with system health information taken into account. Such a testbed would also, ideally, be inexpensive to operate and not require lengthy experiment setup times. The main categories of tasks to be performed on the testbed were defined as the following: (1) development of system-level prognostics-enabled decision making (PDM) algorithms; (2) maturation

and standardization of interfaces between various reasoning algorithms; (3) performance comparison among the algorithms from different organizations; and (4) generation of publicly available datasets for enabling further research in PDM.

Adding decision optimization capability to a diagnostic/prognostic health management system will allow to not only determine if a vehicle component is failing and how long would it take for it to fail completely, but also to use that information to take (or suggest) actions that can optimize vehicle maintenance, ensure mission safety, or extend mission duration. Depending on the prediction time horizon for the fault, the character of these actions can vary. If a fault is expected to develop into a complete failure in a matter of seconds, a rapid controller reconfiguration, for example, may be required. If the fault progression takes minutes then, perhaps, reconfiguration of the vehicle (such as switching from the low gain antenna to the high gain antenna) can help remedy the situation. Finally, if the remaining useful life (RUL) for the faulty component is measured in hours, days, weeks, or longer, a new mission plan or adjustments to the logistics chain may be warranted (often in conjunction with lower level actions).

While eventually an aerial test vehicle, such as an unmanned fixed wing airplane or a helicopter, would allow testing of the aforementioned technologies on complex scenarios and with motion in three-dimensional space, operating such a testbed is often expensive. A single flight hour often requires many days of preparation. Safety requirements for an aerial test vehicle, even without a human onboard, are also usually quite stringent. In contrast, a rover whose movement is restricted to two dimensions can operate at low speeds in a controlled environment, making experiments easier and safer to set up. The experiments can still involve motion, complex subsystems interactions, and elaborate mission plans, but the possibility of a dangerous situation occurring is reduced significantly. For technologies in early phases of development in particular, a land vehicle platform could provide a suitable initial test environment for the majority of development goals at a fraction of the cost of an aerial vehicle, usually with a clear transition path to the latter.

Guided by the above reasons and requirements, an effort was started to develop such a platform on the basis of the K11, a rover originally slated to serve as a robotic technologies test vehicle in the Antarctic (Lachat, Krebs, Thueer, & Siegwart, 2006). The rover equipment (such as its batteries) was updated and its sensor suite was expanded. A key distinction from other planetary rover development efforts should be stressed, however. The focus of this research is not to develop next-generation planetary rover hardware, but rather to use the K11 rover platform to create a realistic environment for testing novel PDM algorithms. These algorithms would then be used as blueprints by other organizations in order to create PDM functionality for their particular applications.

Fault modes in components that are common to various types of vehicles (such as electric motors, batteries, or control electronics) were identified and injection methods for some of them were developed – with as much realism as practical. A software simulator, meant for allowing rapid validation of autonomy algorithms and for providing optimization guidance during hardware-in-the-loop experiments, was developed as well. While for the time being algorithms developed at NASA Ames are being used to populate the autonomy architecture on the K11, algorithms from other sources could be tested and evaluated in the future.

The next section of the paper, Section 2 focuses on the testbed hardware, while Section 3 summarizes work on the simulator to date, including experimental validation of the models. Section 4 describes the current reasoning software suite being deployed on the testbed and Section 5 provides a summary of the accomplishments and outlines potential future work.

2. TESTBED

The following section consists of three main parts: the first part describes the hardware of the testbed, including its sensor suite; the second focuses on the testbed (core) software; and the third one describes the methods used for fault injection. It should be noted that there is a distinction made in this work between core software and reasoning (including PDM) software. Examples in the former category include the operating system, the middleware providing communication between components, the data acquisition software, the low-level drivers – essentially the elements that enable the K11 to perform all of its functions under direct human control. The reasoning package, on the other hand, is the software that lessens or completely removes the dependence on a human operator. PDM software is what constitutes the test article for this testbed and its elements will be swapped in and out depending on the test plan. The current set of PDM elements is described in Section 3.

2.1. Hardware

The K11 is a four-wheeled rover (Figure 5) that was initially developed by the Intelligent Robotics Group (IRG) at NASA Ames to be an Antarctic heavy explorer. It had a design capacity to transport 100 kilograms of payload across ice and frozen tundra (Lachat, Krebs, Thueer, & Siegwart, 2006).

The rover was also previously used in experiments to test power consumption models and in a gearing optimization study. It has been tested on various types of terrain, including snow. The lightweight chassis was designed and built by BlueBotics SA. It consists of an H-structure and a

Figure 1: The K11 rover

joint around the roll axis to ensure that the wheels stay in contact with the ground on uneven terrain. The mass of the rover without the payload is roughly 140 kg. Its dimensions are approximately 1.4m x 1.1m x 0.63m.Each wheel on the K11 is driven by an independent 250 Watt graphite-brush motor from Maxon Motors equipped with an optical encoder. The wheels are connected to the motors through a bearing and gearhead system (gearhead ratio r = 308). Motors are controlled by Maxon Epos 70/10 single-axis digital motion controllers, capable of operating in velocity, position, homing, and current modes.

After considering various alternatives, LiFePO4 (lithium iron phosphate) batteries, commonly used in modern electric vehicles, were selected to power the rover. LiFePO4 batteries have a high charge density, perform well in high temperatures, and are not prone to combustion or explosion. Furthermore, they can withstand a high number (approximately 500) of charge/discharge cycles before needing to be replaced. There are four 12.8V 3.3 Ah LiFePO4 batteries on the K11, connected in series. Each battery contains 4 cells.

The philosophy in developing the sensor suite on the K11 (summarized in Table 1) was to employ only those sensors or data acquisition hardware that are commonly available on a variety of vehicles or can be added at a reasonable cost, while also providing sufficient data for a PDM system. Each component is utilized to the maximum extent possible. For instance, the motor controllers are not only used for their primary purpose of operating the motors and giving feedback on their velocity and current consumption, but are also used to support external sensors. The unused controller analog input channels are called upon to read battery voltage and current sensors and will be used for collecting temperature readings. In a similar vein, a decision was made to utilize a modern off-the-shelf smartphone for part of the instrumentation suite instead of, for example, a dedicated GPS receiver and a gyroscope. The smartphone also provides a still/video camera, a compass, and data

Measurement Type	Manufacturer	Location/comments
GPS (longitude and latitude)	Motorola	On the smartphone
Gyroscope (roll, pitch, yaw)	Motorola	On the smartphone
Motor temperature	Omega	On each motor (to be implemented)
Battery temperature	Omega	On each battery pack (to be implemented)
Position encoder	Maxon	On each drive motor
Battery voltage	custom	On a custom PCB board measuring individual battery pack voltages
Total current	custom	On a custom PCB board measuring individual battery pack voltages
Individual motor current	Maxon	Part of motor controller

Table 1: Measurements available on the K11

processing and storage resources. It has a built-in wireless capability for communicating with other on-board components and directly with the ground station (as a back-up to the main communication link through the on-board computer). The current phone used on the K11 is a Google Nexus S.

The bulk of the computational resources needed to operate the rover are provided by the onboard computer (an Intel Core 2 Duo laptop). Its responsibilities include executing the motor control software, performing data acquisition, as well as running all of the reasoning algorithms. A second laptop computer currently serves as a ground control station.

2.2. Software

Several of the core software elements on K11 are adopted, or being adopted, from the Service-Oriented Robotic Architecture (SORA) developed by the Intelligent Robotics Group (Fluckiger, To, & Utz, 2008). This includes the navigation software; the middleware, based on Common Object Request Broker Architecture (CORBA) (Object Management Group, 2004) and Adaptive Communication Environment (ACE) (Schmidt, 1994); and the telemetry software, the Robot Application Programming Interface Delegate (RAPID) (NASA Ames Research Center, 2011).

The smartphone (running Google Android 2.2 operating system) hosts a data acquisition module written in Java. That module collects data from the phone's sensors (described in the previous section) and sends it over a User Datagram Protocol (UDP) socket to the onboard computer. The central data acquisition software running on the computer receives the phone data, merges it with data received from other sources (e.g., voltage sensors, current

sensors, controller state, etc) and records it into a unified data file, which can then be transmitted to the ground control station. The central data acquisition software on the K11 is based on LabView from National Instruments.

The ground station graphical user interface (GUI) software is also written in LabView. It allows the operator to take manual control of the rover (via on-screen commands or a joystick) and to set data recording preferences. The operator can switch the GUI from interacting with the K11 hardware to interacting with the K11 simulator. One of the goals in developing the simulator (described further in Section 3) is to make the difference in interacting with it versus the rover hardware as minimal as possible. The operating system currently used on both the onboard computer and the ground station is Microsoft Windows XP. It will be replaced by a UNIX-based operating system in the near future.

2.3. Fault Modes

A number of fault modes have been identified so far for implementation on the testbed (Table 2). The criteria for their selection include relevance to a variety of aerospace vehicles (not just rovers), feasibility of implementation, and progression time from fault to failure. The last criterion is important because if the progression time is too brief (e.g. microseconds), then likely no useful action can be taken in the prognostic context to predict the remaining useful life of the component and remedy the situation. On the other hand, if the fault-to-failure progression time is measured in years, then running experiments on those fault modes may become impractical. Faults in both of the above categories could still be handled by diagnostic systems, however. Out of the fault modes described in Table 2, a few were selected for the initial phase of the project. The methods for their injection on the K11 are covered in more detail next. The methods for modeling progression of these faults in the simulator are described in Section 3.

2.3.1. Mechanical Jam and Motor Windings Deterioration

The first fault mode selected for implementation is a mechanical jam on the motor axle which leads to increased current, overheating of motor windings, deterioration of their insulation, and eventual failure of the motor due to a short in the motor windings. To maintain realism, a performance region for the motor is chosen (using manufacturer's specifications) where a healthy motor would have no problems keeping up with either speed or load requirements. In the presence of increased friction, however, the amount of current needed to satisfy the same speed and load demands is higher, leading to overheating. Unless speed and/or load are reduced or duty cycle (the proportion of time the motor is on versus duration of cool-down intervals) is adjusted, the heat build-up will eventually destroy the insulation of the motor windings and lead to motor failure. This fault mode was first implemented in the simulator and its model verified using experimental data collected on smaller-sized motors that were run to failure under similar conditions (please see section 3.2, Motor Modeling). A hardware fault injection using a mechanical brake on one of the rover motors will be implemented next. The rover motor will not be run to complete failure initially; instead the simulator model parameters and prognostic algorithms will be validated in experiments stopping short of creating permanent damage. Eventually, experiments that will take motors all the way to failure will be performed.

Fault Mode	Injection method	Subsystem
battery capacity degradation	accelerated aging	power
battery charge tracking	normal operations	power
parasitic electric load	programmable	power distribution
motor failure	software	electro-mechanical
increased motor friction	mechanical brake	electro-mechanical
bearing spalls	machined spalls	electro-mechanical
sensor bias/drift/failure	software	sensors
motor driver faults	MOSFET replacement in the controller with an aged component	power distribution

Table 2: Potential fault modes

2.3.2. Parasitic Load

A parasitic electrical load will be injected on the main power distribution line via a remotely controlled rheostat. The rheostat can be set for resistance from 0 to 100 Ohms and can dissipate up to 600 Watts of power. The rheostat will simulate a situation where, for example, an accessory motor is continuously engaged due to a failed limit microswitch.

2.3.3. Electronics Faults

The systems on the K11 provide several opportunities for fault injection in electronics subsystems. Power electronics in the motor drivers allow fault injection in power-switching devices such as Metal–Oxide–Semiconductor Field-Effect Transistors (MOSFETs), Insulated Gate Bipolar Transistors (IGBTs) and electrolytic capacitors used for voltage filtering. These devices have a key role in providing current

to the motors, but are known for relatively high failure rates. Fault injection will also be implemented on the power switches of the motor winding H-bridges, where current will be routed to degraded power transistors during rover operation. In addition, some of the symptoms of power transistors failing will be replicated programmatically by varying the gate voltage. The premise of the fault injection in the H-bridge transistor is that it will diminish the performance of a motor winding, reducing torque and altering motor performance characteristics, making control difficult.

Efforts on accelerated aging of IGBTs and power MOSFETs are presented in (Celaya, Saxena, Wysocki, Saha, & Goebel, 2010). Accelerated aging methodologies for electrolytic capacitors under nominal loading and environmental conditions are presented in (Kulkarni, Biswas, Koutsoukos, Celaya, & Goebel, 2010); methodologies for accelerated aging via electrical overstress are presented in (Kulkarni, Biswas, Celaya, & Goebel, 2011). MOSFETs, IGBTs, and electrolytic capacitors at various levels of degradation will be used to inject component-level electronic faults, with some of the faults expected to have a cascading effect on other electronic and/or mechanical subsystems.

2.3.4. Battery Capacity Degradation

As the rover batteries go through charge/discharge cycles, their capacity to hold charge will diminish. The degradation rate will depend on several factors such as imposed loads, environmental conditions, and charge procedures. For example, Li-Ion chemistry batteries undergo higher rates of capacity fade with higher current draw and operational temperatures. Even at rest, this type of battery has chemical processes occurring that have long-term effects - for instance, latent self-discharge and transient recovery during relaxation. The depth-of-discharge (DoD) and even the storage temperature have major influences on the overall life of the battery as well. There is no specific mechanism required for injecting this fault – the batteries will age naturally in the course of rover operations. Some experiments will, however, utilize battery cells aged to a desired point in their life cycle on the battery aging test stand (Saha & Goebel, 2009)

2.3.5. Remaining Battery Charge Tracking

While not being, in the strict sense, a fault, tracking the remaining battery charge will be one of the main tasks of the prognostic system. End of charge is an end-of-life criterion, so the remaining charge estimate is expected to be a factor in most of actions undertaken by PDM software. Most battery-powered devices have some form of battery state-of-charge (SOC) monitoring onboard. This is mostly based on Coulomb counting, i.e. integrating the current drawn over time, divided by the rated capacity of the battery. The definition used in this work is the following:

$$SoC = 1 - \frac{\int_{t=0}^{t|V=V_{cutoff}} I(t)dt}{\text{Capacity of Current Cycle}} \times \mathbf{100}\%$$

It should be noted that both the numerator and denominator of the fraction are predictions, not the actual measurements: battery voltage prediction for the former and capacity prediction for the latter. Further details are discussed in (Saha and Goebel 2009).

3. TESTBED SIMULATOR

As mentioned previously, a simulator has been developed to aid in the design of PDM algorithms for the testbed. It captures both nominal and faulty behavior, with the controlled ability to inject faults. In this way, it serves as a virtual testbed through which algorithms can be initially tested and validated. Faults in the simulator are modeled as undesired changes in system parameters or configuration. In addition to serving as a virtual testbed, the simulator will also be utilized in guiding the decision making process. A graphical user interface was developed for interacting with the simulator, and is shown in Figure 2. In this section, the models used by the simulator are reviewed and some model validation results are presented.

3.1. Rover Dynamics Modeling

The rover consists of a symmetric rigid frame with four independently-driven wheels. Generalized rover coordinates are shown in Figure 3. The F subscript stands for “front”, the B subscript for “back”, the L subscript for “left”, and the R subscript for “right”. The rover pose is given by (x, y, θ). The independent dynamic variables describing the motion include the body longitudinal velocity v, the body rotational velocity ω, and the wheel rotational velocities ω_{FL}, ω_{FR}, ω_{BL}, and ω_{BR}. Note that the body velocities and wheel velocities are independent due to the presence of slip. Velocity in the lateral direction is negligible (Mandow, 2007).

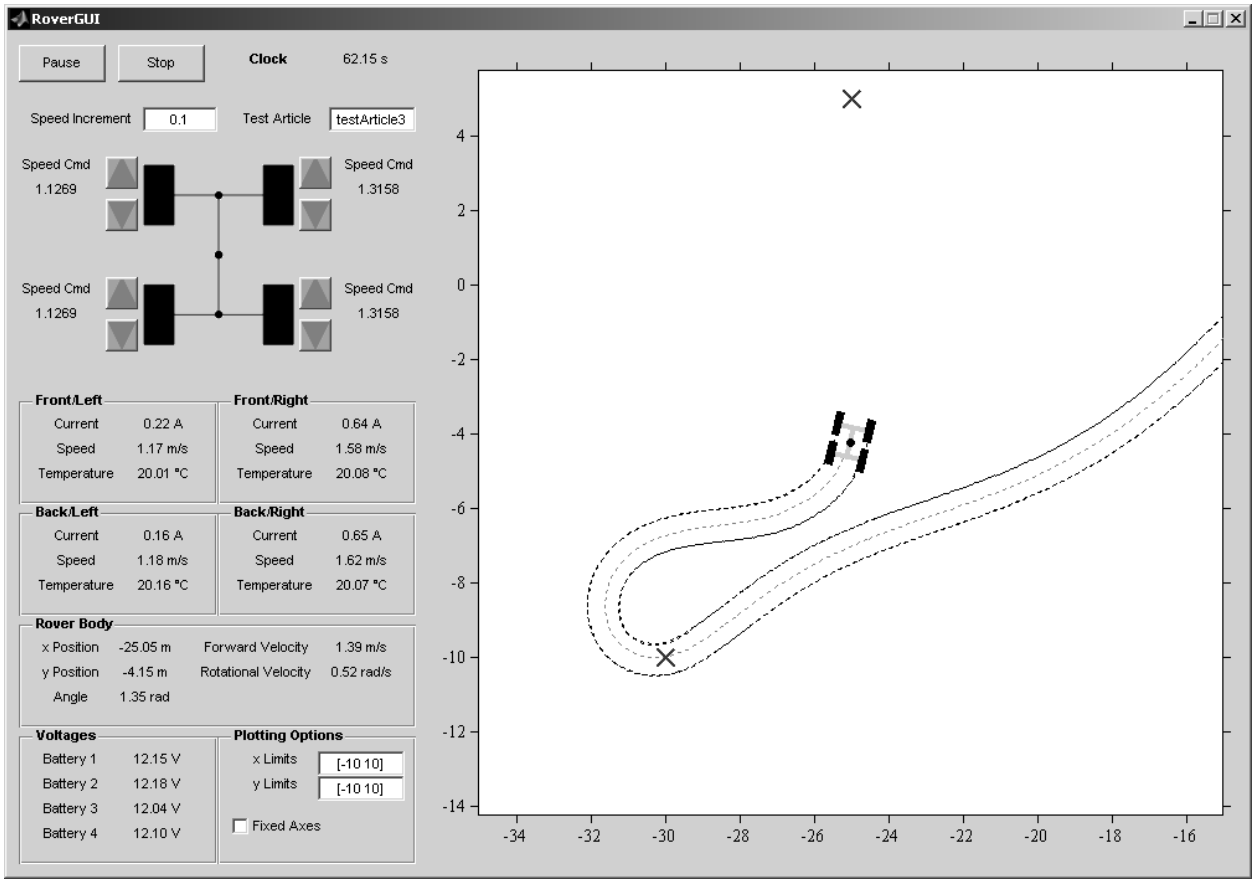

Figure 2: Rover simulation GUI

Since the rover exhibits both longitudinal and rotational velocity, it will experience forces opposing both of these movements. The rover forces are shown in Figure 4. Each motor produces a torque that drives its wheel. When the longitudinal velocity of the rover is equal to the rotational velocity of the wheel times its radius, then there is no slip and no force. Otherwise, some amount of slip will be present and the difference in the relative velocities of the wheel and the ground produce a ground force F_{gl} that pushes the wheel along the ground. These forces are transmitted to the rover body, moving it in the longitudinal direction. The F_{gl} forces produce torques on the rover body, producing a rotation. The rotation is opposed by additional friction forces F_{gr}. The friction forces are defined as:

$$F_{gl} = \mu_g(v_w - v)$$

$$F_{gr} = \mu_r \omega$$

Note that μ_g and μ_r are not in the same units. The F_{gr} forces, opposing the rotation, act at a right angle from the diagonal going from the robot center to the wheel, and in the direction that opposes the rotation. The forward component of this force affects the forward velocity of the rover, just as the component of a F_{gl} force perpendicular to the diagonal affects the rotational velocity. The angle γ is of interest here, given by

$$\gamma = \arctan l/(2b)$$

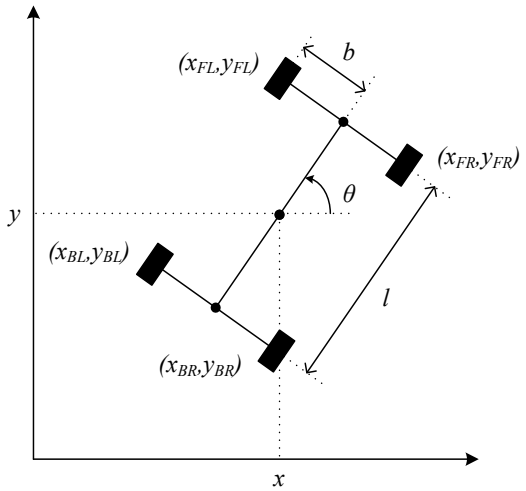

Figure 3: Generalized rover coordinates

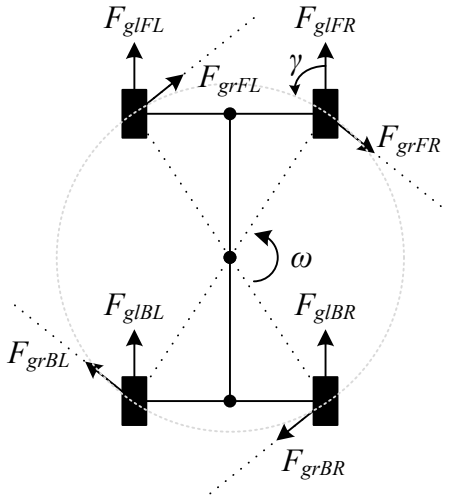

Figure 4: Rover forces

For a given wheel w, the rotational velocity is described by

$$\dot{\omega}_w = \frac{1}{J_w}(\tau_{mw} - \tau_{fw} - r_w F_{glw} + r_w F_{gr}\cos\gamma),$$

where J_w is the wheel inertia, τ_{mw} is the motor torque, and τ_{fw} is the friction torque:

$$\tau_{fw} = \mu_w \omega_w.$$

The forward velocity is described by

$$\dot{v} = \frac{1}{m}(F_{glFL} + F_{glFR} + F_{glBL} + F_{glBR}),$$

assuming that μ_r are the same for all wheels so that the contributions from the F_{gr} forces cancel out. The rotational velocity is described by

$$\dot{\omega} = \frac{1}{J}(d\cos\gamma F_{glFR} + d\cos\gamma F_{glBR} - d\cos\gamma F_{glFL} - d\cos\gamma F_{glBL} - 4dF_{gr})$$

We note that the F_{gl} forces are at distance d from the rover center with the perpendicular component at angle γ. The $cos\gamma$ factor projects the force onto the tangent of the rotation.

3.2. Motor Modeling

The wheel motors are DC motors with PID control. The DC motor model is given by

$$\dot{i}_m = \frac{1}{L}(V_w - i_m R - k_w \omega_w)$$

where V_w is the motor voltage, L is the winding inductance, R is the winding resistance, and k_w is an energy transformation term. The motor torque given by

$$\tau_m = k_\tau i_m$$

where k_τ is an energy transformation term.

Increased motor/wheel friction for wheel w is captured by an increase in μ_w. A change in motor resistance is captured by a change in R. The motors windings are designed to withstand temperatures up to a certain point, after which, the insulation breaks down, the windings short, and the motor fails. It is therefore important to model the temperature behavior of the motor.

The motor thermocouple is located on the motor surface. The surface loses heat to the environment and is heated indirectly by the windings, which, in turn, are heated up by the current passing through them. The temperature of the windings is given by

$$\dot{T}_w = \frac{1}{J_w}(i^2 R - h_w(T_w - T_m)),$$

where J_w is the thermal inertia of the windings, h_w is a heat transfer coefficient, and T_m is the motor surface temperature (Balaban, et al., 2009). It is assumed that heat is lost only to the motor surface, and that winding resistance R is approximately constant for the temperature range considered. The surface temperature is given by

$$\dot{T}_m = \frac{1}{J_s}(h_w(T_w - T_m) - h_a(T_m - T_a))$$

where J_s is the thermal inertia of the motor surface, h_a is a heat transfer coefficient, and T_a is the ambient temperature. Heat is transferred from the windings to the surface and lost to the environment.

This model was validated for DC motors using experimental data collected on the Flyable Electro-Mechanical Actuator (FLEA) testbed (Balaban, Saxena, Narasimhan, Roychoudhury, Goebel, & Koopmans, 2010). The unknown parameters J_w, J_s, h_w, h_a, and R were identified to match data acquired from a scenario where the motor was overloaded and, as a result, heated up considerably. The motor current and surface temperatures were measured. A comparison of predicted vs. measured temperature is shown in Figure 5.

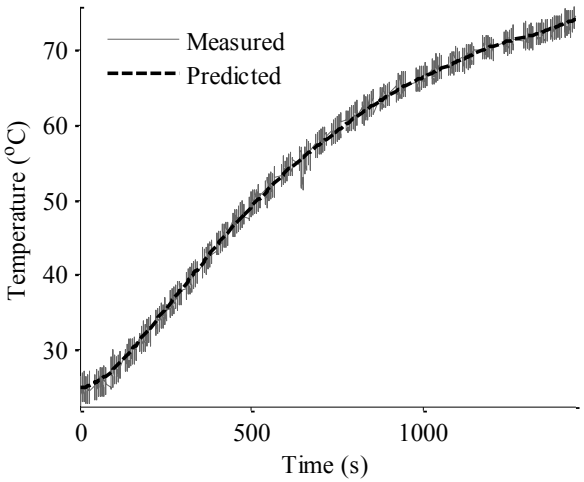

Figure 5: Comparison of measured and model-predicted motor surface temperature for a DC motor

3.3. Sensor Fault Modeling

Sensor faults are captured with bias, drift, gain, and scaling terms. Ranges for typical fault magnitude values have been identified through a literature search and discussions with manufacturers (Balaban, Saxena, Bansal, Goebel, & Curran, 2009). Faults in common sensors such as current, voltage, temperature, and position will be modeled.

3.4. Battery Modeling

The key challenge in modeling a battery is estimating its open-circuit voltage, E_0 . The theoretical open-circuit voltage of a battery is traditionally assessed when all reactants are at 25^oC and at 1M concentration (or 1 atm pressure). However, this voltage cannot be measured directly during battery use due to the influence of internal passive components such as the electrolyte, the separator, and the terminal leads. The measured voltage will be lower; the factors contributing to the voltage drop are characterized in the following paragraphs.

The first factor considered is the *ohmic drop*. The term refers to the diffusion process through which Li-ions migrate to the cathode via the electrolytic medium. The internal resistance to this ionic diffusion process can also be referred to as the IR drop. For a given load current, this drop usually decreases with time due to the increase in internal temperature, which results in increased ion mobility.

The next factor is *self-discharge*, which is caused by the residual ionic and electronic flow through a cell even when there is no external current being drawn. The resulting drop in voltage has been modeled to represent the activation polarization of the battery. All chemical reactions have a certain activation barrier that must be overcome in order for the reaction to proceed and the energy needed to overcome this barrier leads to the activation polarization voltage drop. The dynamics of this process are described by the Butler–Volmer equation, which, in this work, is approximated by a logarithmic function.

C*oncentration polarization* is the voltage loss due to spatial variations in reactant concentration at the electrodes. This occurs primarily when the reactants are consumed faster by the electrochemical reaction than they can diffuse into the porous electrode. The phenomenon can also occur due to variations in bulk flow composition. The consumption of Li-ions causes a drop in their concentration along the cell, which causes a drop in the local potential near the cathode. The magnitude of concentration polarization is usually low during the initial part of the discharge cycle, but grows rapidly towards the end of it or when the load current increases.

Finally, the degradation of battery capacity with aging, as encapsulated by the cycle life parameter, can be modeled by the concept of *Coulombic efficiency*, η_c , defined as the fraction of the prior charge capacity that is available during the following discharge cycle (Huggins, 2008). As mentioned previously, this depends upon a number of factors, particularly on current and depth of discharge in each cycle. The temperature at which the batteries are stored and operated under also has a significant effect on the Coulombic efficiency. For further details on battery modeling, please refer to (Saha and Goebel 2009).

3.5. Electronics Fault Modeling

The field of electronics prognostics is relatively new compared to prognostics for mechanical systems. As a result, research efforts to develop physics-based degradation models that take into account loading and operational conditions are in their early stages. There are several well-known electronics reliability models that deal with failure rates under specific stress factors and corresponding failure mechanisms. However, such models do not take into account usage time, thus making them less suitable for prediction of remaining useful life.

Empirical degradation models of IGBTs, based on the turn-off tail of the drain current, have recently been used for prediction of their future health state (Saha B., Celaya, Wysocki, & Goebel, 2009). Their collector-emitter voltage has been used as precursor of failure as well (Patil, 2009).

In the case of power MOSFETs, the on-state drain to source resistance has been identified as a precursor to failure for the die-attach failure mechanism (Celaya, Saxena, Wysocki, Saha, & Goebel, 2010). For gate-related failure, empirical degradation models based on the exponential function have also been developed (Saha S., Celaya, Vashchenko, Mahiuddin, & Goebel, 2011).

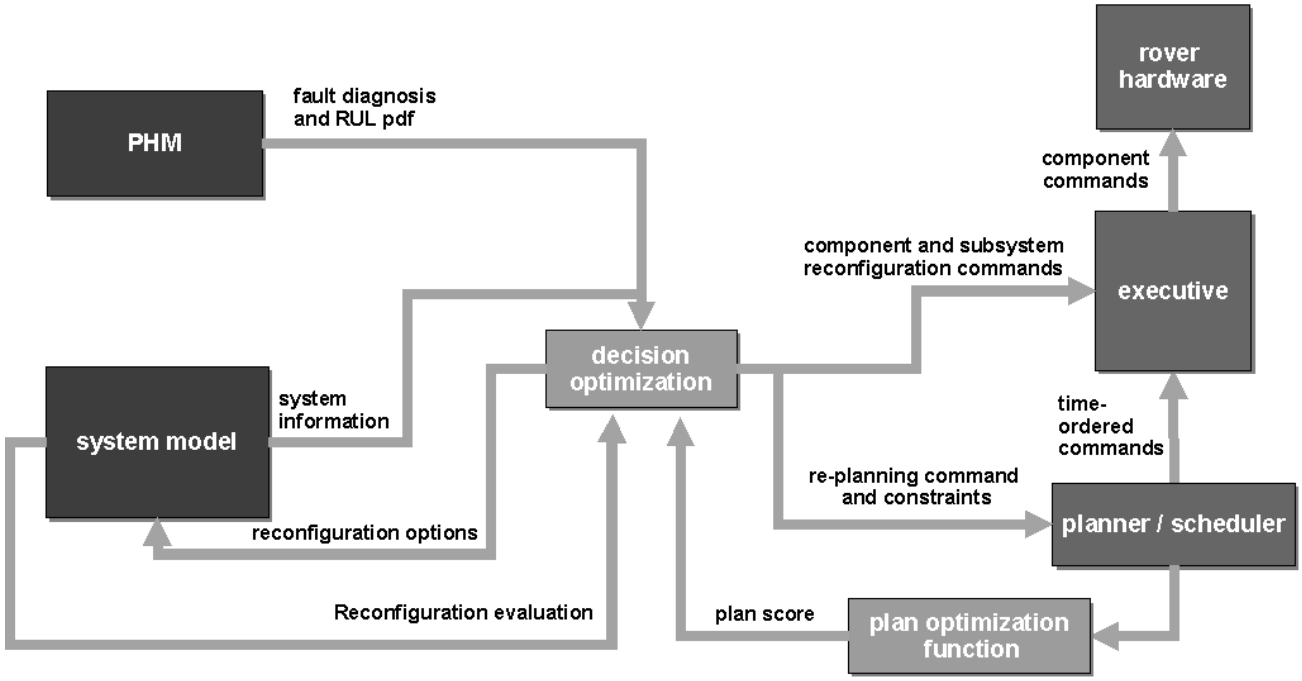

Figure 6: Autonomy architecture

For electrolytic capacitors, an empirical degradation model, also based on an exponential function is presented in (Celaya, Saxena, Vaschenko, Saha, & Goebel, 2011). This model is built based on accelerated degradation data. Details on its physical foundations are presented in (Kulkarni, Biswas, Celaya, & Goebel, 2011).

4. PROGNOSTICS-ENABLED DECISION MAKING ALGORITHM DEVELOPMENT

This section describes the reasoning architecture currently being deployed on the K11 and is meant to mainly provide an overview of the types of algorithms that can be plugged into it for testing and comparison. The current set of algorithms is expected to evolve as this research progresses and other organizations become involved. Figure 6 outlines the architecture and depicts the information flow among its components. The Prognostic Health Management (PHM) element on the figure combines diagnostic and prognostic reasoning engines. If a system fault occurs, the diagnostic engine is tasked with detecting it and identifying what it is, followed by invocation of an appropriate prognostic algorithm to track fault progression. Once a prognosis of the remaining useful life is made, the information is passed to the decision optimization module, which identifies the best way to respond. The K11 simulator, with its nominal and fault-progression models, is used to guide the decision optimization process in some of the cases. The response chosen may involve reconfiguration of low-level controllers or requesting the planner to come up with a new mission plan. Planner output is used to generate action schedules and then, through the executive module, time-ordered commands for individual components.

4.1. Diagnostics

Diagnosis can be defined as the process of detecting, isolating, and identifying faults in the system. A fault is defined as an undesired change in the system that causes the system to deviate from its nominal operation regime. Diagnostic approaches can be broadly divided into two types: model-based and data-driven (Gertler, 1998). Model-based methods rely on a system model built from *a priori* knowledge about the system. Data-driven methods, on the other hand, do not require such models but instead require large, diverse sets of exemplar failure data, which are often not available. The decision of whether to adopt a model-based or a data-driven diagnostic approach depends on the sensor suite properties and the fault modes of interest, among other factors.

Currently a model-based approach is adopted for providing a diagnostic system for the rover, as the sensors (Table 1) and fault modes (Table 2) lend themselves to physics-based modeling. Once sensors measuring more complex dynamics (e.g. accelerometers) are added to the system, data-driven diagnosis methods may be required. Additionally, model-based and data-driven algorithms can be synergistically combined to improve upon either approach implemented individually (Narasimhan, Roychoudhury, Balaban, & Saxena, 2010).

Typically, model-based methods require a nominal system model, as well as a model that captures the relations between faults and symptoms. The overall goal is to use the model to generate estimates of nominal system behavior, then compare them with observed system behavior. If any deviation from nominal, i.e., a symptom, is observed, the fault-symptom model is used to isolate the fault modes that explain the symptoms.

A model-based diagnosis approach is generally split into three tasks: fault detection, fault isolation, and fault identification, with the following event flow:

- Fault detection involves determining if the system behavior has deviated from nominal due to the occurrence of one or more faults. The fault detector takes as inputs the measurement readings, y, and the expected nominal measurement values, $\hat{y}$, generated by the nominal system observer. The detector indicates a fault if the residual, $r = y - \hat{y}$, is statistically significant.
- Once a fault is detected, the fault isolation module generates a set of fault hypotheses, F, and, at every time step, reasons about what faults are consistent with the sequence of observed measurements in order to reduce F. The goal of fault isolation is to reduce F to as small a set as possible. If only single faults are assumed then, ideally, the goal of fault isolation is to reduce F to a singleton set.
- Once the fault (or faults) are isolated, fault identification is invoked. It involves quantitatively evaluating the magnitude of each fault, $f \in F$.

Once the fault magnitude is identified, prognostic algorithms can be invoked to predict how the damage grows over time and estimate the remaining useful life of the affected component and the overall system.

4.2. Prognostics

For the purposes of this research, prognostics is defined as the process which predicts the time when a system variable or vector indicating system health no longer falls within the limits set forth by the system specifications (End-of-Life or EOL). The prediction is based on proposed future usage. In some cases the trajectory of the aforementioned variable or vector through time is predicted as well. Similarly to diagnostic methods, prognostics methods are generally classified as either data-driven or model-based: (Schwabacher, 2005); (Saha and Goebel 2009); (Daigle & Goebel, 2011).

Generally, the inputs to a prognostic algorithm include information on the fault provided by the diagnostic algorithm (e.g. fault type, time and magnitude). Output of a prognostic algorithm could be then presented to a PDM algorithm in one of the following ways:

a. as an estimate $\phi^{q}{}_{t_j,t_i,L}$ of the variable of interest (e.g. accumulated damage or remaining life) at a specific time t_j given the information up to time t_i for a component q, where $t_j > t_i$. L is the anticipated average load up to t_j.

b. As a discrete point trajectory $\Phi^{q}_{i,L}(j)$ given information up to the point i, where $L=\{l_1, l_2, \ldots, l_{EoP}\}$ are the anticipated load values for each point on the

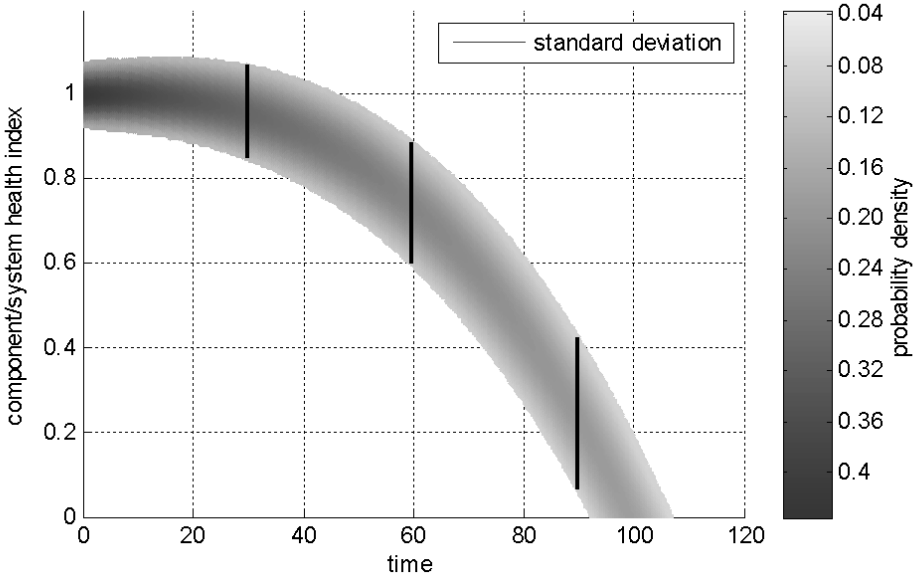

Figure 7. Prognostic prediction example

prediction trajectory, EOP is the end of prediction index, and $I < j < EOP$.

c. As a continuous function $\Phi^{q}_{t_i,L}(t_j)$ given information up to a time t_i and an anticipated load function $L(t)$

The estimate produced in all of the above cases may be expressed as a probability density function (pdf) or as moments derived from the probability distribution. Each of the three options assumes time-variability of the prognostic function, which is one of the main factors that make PDM an interesting and challenging research problem. The function may change from one time of prediction to the next as more information about the behavior of the system becomes available.

Figure 7 shows the important features of an example prediction curve produced at a specific time *t*. The points on the time axis are relative to the moment of prediction. The health index values are normalized to be in the [0, 1] interval. Probability density of health index values for each point in time is illustrated using a grayscale map (shown on the right side of the figure). The solid black bars are drawn to show one standard deviation of probability distributions at different time points into the future. End-of-Life is a time value corresponding to the health index chosen to indicate that the component or a system can no longer provide useful performance. In this example EOL corresponds to health index of 0, however the threshold can be defined as any other value in the [0, 1] interval.

The prediction step requires knowledge of the future usage of the system. For the rover, this involves the expected future trajectory and environmental inputs, such as the terrain and the ambient temperature. The physics models developed for the simulator can be utilized in both the estimation and prediction phases. Damage progression processes that are difficult to model may require use of data-driven prognostics methods. In the remainder of this section, more details are provided on the prognostic methods currently investigated for the fault modes of interest.

4.2.1. Mechanical Jam/Windings Insulation Deterioration

The thermal build-up model as described in the simulator section will be used to predict when the interior of a motor would reach the temperature at which insulation of the windings is likely to melt, thus disabling the motor. A machine-learning prediction method will also be utilized for comparison. The method is based on the Gaussian Process Regression (GPR) principles and was previously tested on another testbed developed at NASA Ames, the FLEA (Balaban, Saxena, Narasimhan, Roychoudhury, & Goebel, Experimental Validation of a Prognostic Health Management System for Electro-Mechanical Actuators, 2011). The FLEA was used to inject and collect data progression of the same type of fault in the motors of electro-mechanical actuators. Several motors were run to complete failure and GPR demonstrated a high accuracy in predicting their remaining useful life.

GPR does not need explicit fault growth models and can be made computationally less expensive by sampling techniques. Further, it provides variance bounds around the predicted trajectory to represent the associated confidence (Rasmussen & Williams, 2006). Domain knowledge available from the process is encoded by the covariance function that defines the relationship between data points in a time series. In the present implementation, a Neural Network type covariance function is used.

Sensor data is processed in real-time to extract relevant features, which are used by the GPR algorithm for training during the initial period. The longer is the training period, the better are the chances for the algorithm to learn the true fault growth characteristics. However, to strike a balance between the length of the training period and the risk of producing an insufficient prediction horizon, a limit for the training period is set. Once this limit is reached, the algorithm starts predicting fault growth trajectories. EOL is subsequently determined by where these trajectories intersect the predetermined fault level threshold. As time progresses, the GPR model is updated with new observations and, subsequently, the predictions are updated as well. Best fitting hyper-parameters for the covariance function are determined via a maximum-likelihood optimization. The uncertainty created by this process is handled by drawing a large (p~50) number of samples from the observed data at each prediction instance (t_p) and training p different Gaussian Process models on these distinct data sets. Their prediction results are then averaged.

4.2.2. Battery Capacity Deterioration and Charge Tracking

There are several methods widely in use for batteries that relate capacity and SOC to the number of cycles a battery has undergone and its open circuit voltage. Most such methods, however, are reliability based, i.e. they assume certain usage profiles are maintained throughout the cycle life of the battery. Such assumptions (for instance that the battery undergoes full discharge followed by full charge repeatedly until its end-of-life) are not always realistic. This is especially true for a platform such as the K11, where individual missions may have different goals with different load profiles. Under such circumstances, it is advantageous to model the internal processes of the battery and let a Bayesian inference framework, such as the Particle Filter (Arulampalam, Maskell, Gordon, & Clapp, 2002) manage the uncertainty in the model (Saha & Goebel, 2009)

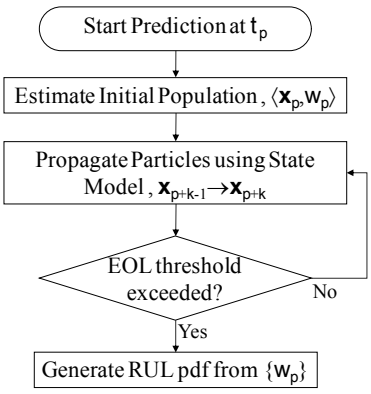

Figure 8: Prediction flowchart

In the remaining battery charge prediction application, the main state variable is the battery voltage, *E*, and in the objective is to predict when *E* reaches the low-voltage cutoff. During the prognostic process a tracking routine is run until a long-term prediction is required, at which point the state model is used to propagate the last updated particles until the end-of-discharge (EOD) voltage cutoff is reached. The RUL pdf is computed from the weighted RUL predictions of the particles in the last tracking step. Figure 8 shows the flow diagram of the prediction process.

In the case of the overall battery life cycle, the main variable that needs to be tracked is the capacity of the battery itself, and the goal is to predict when the battery capacity will fade by more than 20% from when it was new. At that point the battery is said to have reached its EOL. For a description of the main physical phenomena behind these processes, please refer to Section 3.2. A battery lifecycle model and a Particle Filter algorithm utilizing it are presented in (Saha & Goebel, 2009) and (Saha, et al., 2011).

4.2.3. Electronics Faults

Previously researched data-driven and model-based (direct physics and empirical/Bayesian) techniques are being utilized for addressing electronics faults. The particle filter approach has been used in conjunction with an empirical degradation model for IGBTs experiencing failures related to thermal overstress (Saha B. , Celaya, Wysocki, & Goebel, 2009). For power MOSFETs, a data-driven prognostics approach based on Gaussian Process Regression has recently been implemented for die-attach degradation (Celaya, Saxena, Vaschenko, Saha, & Goebel, 2011). For electrolytic capacitors, a remaining useful life prediction based on a Kalman filter has been developed using a degradation model based on an empirical exponential function (Celaya, Kulkarni, Biswas, & Goebel, 2011). It should be noted that the aforementioned efforts make the assumption of usage levels and operational conditions staying constant in the future. New accelerated aging experiments aimed at producing degradation models without these limitations are currently underway.

4.3. Decision Making

As stated previously, one of the main objectives of the K11 testbed is to investigate PDM algorithms in order to enhance an aerospace vehicle's capability to achieve its high-level goals – be it under a faulty condition, degraded operation of a subsystem, or an anticipated catastrophic failure. There has been an increasing amount of research conducted over the last several years in prognostic methodologies for various types of components or systems. The effort described in this paper aims to bring more attention to the "management" aspect of prognostic health management, i.e. what could be done after a fault is detected and the trajectory of its progression is predicted.

Several factors are being used to select the appropriate system level (or levels) on which to respond to an off-nominal condition. These factors include the severity of a fault, its criticality, and predicted time-to-failure interval. A faulty electronic component in an electric motor driver could prompt the decision-making system to trigger a controller reconfiguration - so as to ensure the dynamic stability of the system and a certain level of retained performance. At a different level, a control effort reallocation can be triggered by a supervisory mid-level controller in order reduce the torque required from a faulty drive motor and compensate for the reduction with the other motors. Reallocating the load could, potentially, extend the remaining useful life of the affected component long enough to ensure achievement of the mission objectives. At the highest level, the rover mission can be re-planned based on prognostic health information so as to achieve maximum possible utility and safety. The above examples call on different system components in their response; there are, however, commonalities for all of them. There is always an objective (or a set of objectives) to be met and a series of actions to be selected by the decision making process in order the meet those objectives. Therefore, the decision making process is, essentially, an optimization process which tries to achieve specified objectives by considering system performance and health constraints.

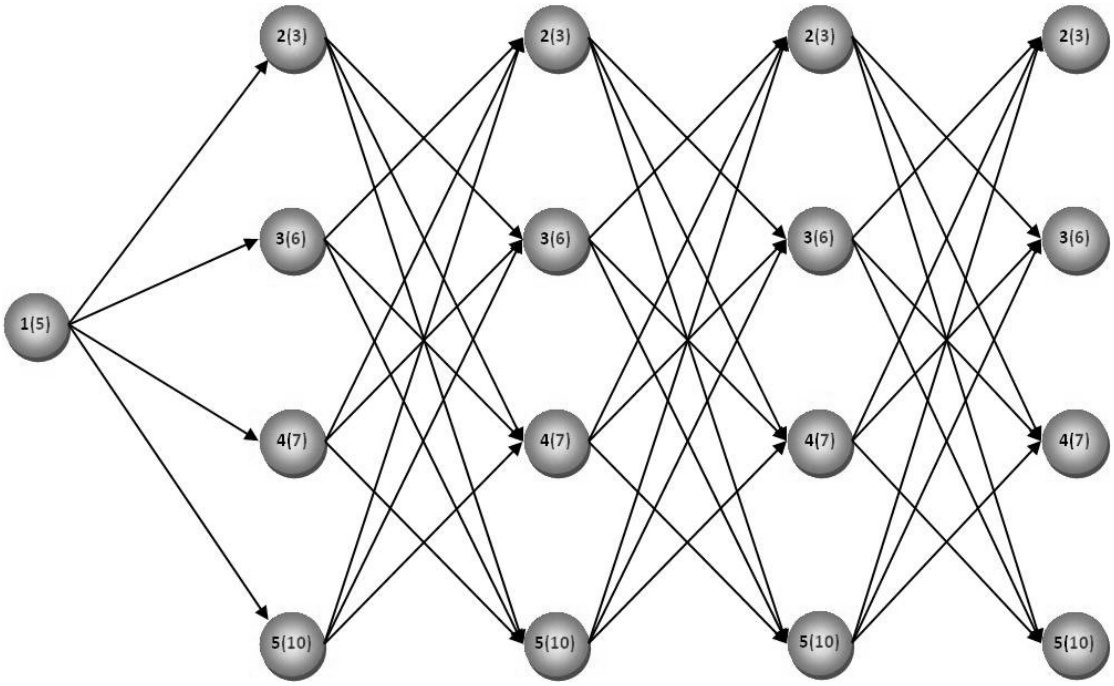

Figure 9. Decision optimization solution space

The scope for the decision-making module in the current implementation is defined as the following: getting vehicle health information from the prognostic health reasoner and the simulator, the decision-making module evaluates the best course of action to take (e.g., controller reconfiguration or mission replanning), while stopping short of performing the actual reconfiguration or re-planning. Instead, the decision-making module adjusts goals and constraints for other software components. To use the planner as an example, the module could set a constraint on rover speed or on the total mission duration, then request the planner to come up with a detailed new plan. In the future, however, it may be necessary to consider whether making PDM-specific modifications to the planner or the adaptive controller, for instance, would improve performance.

The rover is assumed to be an autonomous planetary vehicle, operating with only high level control from human operators (initial sets of goals and constraints). The following definitions and assumptions are used in the current phase of the work:

- The initial sets of goals $G=\{g_1, g_2, ..., g_N\}$ and constraint variables $K=\{k_1, k_2, ..., k_M\}$ are provided as inputs.
- The initial constraint ranges are: $\forall k_j \in K$, $\exists [a_j^0, a_j^1]$, $k_j \in [a_j^0, a_j^1]$, The constraint ranges are adjusted given information from the diagnostic, prognostic, and decision optimization routines.
- Some elements of G may be eliminated as a result of the optimization process. The size of K (M) will remain constant.
- Goal rewards: $\forall g_i \in G, \exists r_i \in [0, r_{max}]$. r_{max} is the maximum possible value of goal reward.
- Goal locations: $\forall g_i \in G,\ \exists l(x_1,\ x_2,\ x_3,\ \theta_1, \theta_2, \theta_3)$, $l \in L$. The preceding location definition is general for a three-dimensional space. In the case of a rover it simplifies to $l(x_1,\ x_2,\ \theta_1,)$.
- Constraint ranges: $\forall k_j \in K$, $\exists [a_j^0, a_j^1]$, $k_j \in [a_j^0, a_j^1]$
- Transition cost: $\forall (l_i,\ l_j) \exists \bar{c},\ where\ \bar{c} = \{c_e,\ c_h\}$, c_e is the energy cost and c_h is the health (life) cost for the system. The former is calculated based on the distance between the goal locations, proposed velocity, and the load index (terrain difficulty). The later is estimated using the health prognostic function, which takes the distance, velocity, and the load index as its inputs.
- Mission starts with energy amount E_0 available
- $E(t)$ and $H(t)$ are the energy and system health 'amounts' at time t, respectively. $E(t)=0$ or $H(t)=0$ constitutes an end-of-life (EOL) condition
- The objective is to accumulate the maximum possible reward before energy and health budgets are exceeded

Two PDM approaches are currently implemented and verified in a simulation involving a small number of goals: Dynamic Programming (DP) and a variant of Probability Collectives (PC) (Wolpert, 2006). A (more computationally expensive) exhaustive search method is used for verification of the DP and PC algorithms. A simple five-goal example is presented next for illustration purposes.

Each of the nodes (goals) is associated with a reward value in the parenthesis (

Figure 9). The vehicle starts out at goal 1, but can choose to traverse the rest of the goal in any order. For some of the degradation modes system health cost may correlate to the energy cost (which, in turn, could be proportional to the sum of loads on the system, whether they are mechanical or electrical). Motor windings insulation deterioration due to increased friction in the system could be one such example. One the other hand, deterioration of an electronic component in one of the subsystems may be determined primarily by the amount of current flowing through that component, ambient temperature, and time.

The implemented DP algorithm uses forward propagation, evaluating the best solution for transitioning from stage to stage, while assuming optimality of the previously made decisions. An 'efficiency index' is used for guiding the stage-to-stage decision process:

$$e_{ij} = r_j/(\sum \bar{c}_{ij})$$

If either health or energy values become less or equal to zero (or all the nodes are visited), the forward propagation phase is stopped. After the forward traverse is completed, an optimal path p_{opt} is 'backed-out' by traversing the stages in the opposite (right-to-left) direction, starting with the node associated with the highest accumulated reward.

The algorithm based on Probabilities Collectives principles is structured in the following manner:

- P is defined as the enumerated set of all possible paths p
- An initial probability distribution $f(p)$ for P is assigned
- 'Related' paths are defined as those that share $n \in [2, N]$ initial nodes, with n incremented progressively every m iterations of the algorithm
- P is sampled using $f(p)$, obtaining a sample path p_i. The cumulative path reward is evaluated by 'walking' the sampled path and taking into account energy and health budgets. If the reward is equal or greater than the current maximum, the probability of the sampled path **and** paths that are 'related' to it are increased and P is re-normalized.

Uncertainty in transition costs and node rewards is incorporated by associating them with probability distributions as well. These distributions are then sampled when 'walking' a path during its evaluation.

Experimenting with DP- and PC-based algorithms showed that both would work well on relatively uncomplicated problems, such as the one described in this section. Limitations of the two approaches started to become evident as well, however. A DP implementation will become more challenging if will multi-objective problems are posed (i.e. optimization over component(s) RUL in addition to cumulative mission reward is desired), unless the multiple optimization variables lend themselves to being aggregated into a single 'composite' variable. The PC-based method, on the other hand, will likely have a lower limit on the size of the goal set it can practically process, at least in its current form. Nevertheless, PC appears to be well-suited for the problem of optimizing system parameters and constraints for maximum RUL, which is being pursued next.

4.4. Task Planning and Execution

Once the high level goals and constraints are determined by the prognostics-enabled decision making module, the detailed task planning for the rover will be generated using NASA's Extensible Universal Remote Operations Architecture (EUROPA) (Frank & Jonsson, 2003). EUROPA provides the capability of solving task planning, scheduling, and constraint-programming problems.. In a complex system, such as a rover, scheduling specific tasks to be executed is often a non-trivial problem. There are resources that are shared by different processes that may not necessarily be available at all times, so EUROPA supports generation of a schedule of activities. Plans and schedules generated by EUROPA (either nominal or those generated in response to a fault) will be passed for automated execution via Plan Execution Interchange Language (PLEXIL) (Dalal, et al., 2007).

5. CONCLUSIONS

The work described in this paper is aimed at providing an inexpensive, safe platform for development, validation, evaluation, and comparison of prognostics-enabled decision making algorithms. Technologies resulting from this research are planned to be transferred for further maturation on unmanned aerial vehicles and other complex systems. At present, the K11 testbed already constitutes a promising platform for PDM research. A list of fault modes of interest has been identified and a number of them have already been implemented in software and/or hardware. A software simulator has been developed that incorporates models of both nominal and off-nominal behavior, with some of the models verified using experimental data. The software architecture for the testbed has been defined in such a way as to allow quick replacement of autonomy elements

depending on testing objectives and customers. The first set of reasoning algorithms, developed at NASA Ames, is being deployed.

Plans for the near future include addition of further injectable fault modes, field experiments of greater complexity, simulator model refinement, and extension of PDM methods to handle more complex problems, including constraints adjustment for optimal RUL . Data collected on the testbed is planned for distribution to other researchers in the field.

ACKNOWLEDGEMENT

Student interns George Gorospe, Zachary Ballard, Sterling Clarke, Tabitha Smith and Kevin Rooney have contributed tremendously to this effort through their creativity and hard work. The team also gratefully acknowledges all the advice and assistance from the Intelligent Robotics Group (Terry Fong, Liam Pedersen, Vinh To, Susan Lee, Hans Utz, Lorenso Fluckiger, Maria Bulaat, Vytas SunSpiral, and others). Jeremy Frank and Michael Dalal of the Planning & Scheduling Group have also been very generous with their time, providing expertise on planning, scheduling, and automated execution technologies. Many ideas in this work have been borne out of discussions with colleagues at Impact Technologies (Liang Tang, Eric Hettler, Bin Zhang, and Jonathan DeCastro) who are actively collaborating with NASA Ames on prognostics-enabled automated contingency management, testbeds, and other topics. The funding for this research is provided by NASA ARMD System-wide Safety & Assurance Technology (SSAT) project.

NOMENCLATURE

ARMD	Aeronautics Research Mission Directorate
CORBA	Common Object Request Broker Architecture
DDS	Data Distribution Service
DM	Decision Making
EOD	End Of Discharge
EOL	End Of (Useful) Life
EUROPA	Extensible Universal Remote Operations Architecture
GPR	Gaussian Progress Regression
GPS	Global Positioning System
IGBT	Insulated Gate Bi-polar Transistor
LiFePO4	Lithium Iron Phosphate
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
PDM	Prognostics-enabled Decision Making
PHM	Prognostics and Health Management
PLEXIL	Plan Execution Interchange Language
RUL	Remaining Useful Life
SOC	State Of Charge

REFERENCES

Arulampalam, M., Maskell, S., Gordon, N., & Clapp, T. (2002). A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing , 50* (2), 174-189.

Balaban, E., Saxena, A., Bansal, P., Goebel, K., & Curran, S. (2009). Modeling, Detection, and Disambiguation of Sensor Faults for Aerospace Applications. *IEEE Sensors Journal , 9* (12), 1907 - 1917.

Balaban, E., Saxena, A., Goebel, K., Byington, C., Watson, M., Bharadwaj, S., et al. (2009). Experimental Data Collection and Modeling for Nominal and Fault Conditions on Electro-Mechanical Actuators. *Annual Conference of the Prognostics and Health Management Society.* San Diego, CA.

Balaban, E., Saxena, A., Narasimhan, S., Roychoudhury, I., & Goebel, K. (2011). Experimental Validation of a Prognostic Health Management System for Electro-Mechanical Actuators. *AIAA Infotech@Aerospace.*

Balaban, E., Saxena, A., Narasimhan, S., Roychoudhury, I., Goebel, K., & Koopmans, M. (2010). Airborne Electro-Mechanical Actuator Test Stand for Development of Prognostic Health Management Systems. *Annual Conference of the Prognostics and Health Management Society.* San Diego, CA.

Celaya, J., Kulkarni, C., Biswas, G., & Goebel, K. (2011). Towards Prognostics of Electrolytic Capacitors. *AIAA Infotech@Aerospace.* St. Louis, MO.

Celaya, J., Saxena, A., Vaschenko, V., Saha, S., & Goebel, K. (2011). Prognostics of power MOSFETs. *23rd International Symposium on Power Semiconductor Devices and ICS.* San Diego,CA.

Celaya, J., Saxena, A., Wysocki, P., Saha, S., & Goebel, K. (2010). Towards Prognostics of Power MOSFETs: Accelerated Aging and Precursors of Failure. *Annual Conference of the Prognostics and Health Management Society.* Portland,OR.

Daigle, M., & Goebel, K. (2011). Multiple Damage Progression Paths in Model-based Prognostics. *IEEE Aerospace Conference.* Big Sky, Montana.

Dalal, M., Estlin, T., Fry, C., Iatauro, M., Harris, R., Jonsson, A., et al. (2007). *Plan Exectution Interchange Language (PLEXIL).* Moffett Field: NASA Ames Research Center.

Fluckiger, L., To, V., & Utz, H. (2008). Service oriented robotic architecture supporting a lunar analog test. *International Symposium on Artificial Intelligence, Robotics, and Automation in Space.* Los Angeles, CA.

Frank, J., & Jonsson, A. (2003). Constraint-Based Attribute and Interval Planning. *ournal of Constraints, Special Issue on Constraints and Planning , 8* (4), 335-338.

Gertler, J. (1998). *Fault Detection and Diagnosis in Engineering Systems.* New York: Marcel Dekker Inc.

Huggins, R. (2008). *Advanced Batteries: Materials Science Aspects* (1st Edition ed.). Springer.

Kulkarni, C., Biswas, G., Celaya, J., & Goebel, K. (2011). Prognostic Techniques for Capacitor Degradation and

Health Monitoring. *Maintenance & Reliability Conference.* Knoxville, TN.

Kulkarni, C., Biswas, G., Koutsoukos, X., Celaya, J., & Goebel, K. (2010). Aging Methodologies and Prognostic Health Management for Electrolytic Capacitors. *Annual Conference of the PHM Society.* Portland, OR.

Lachat, D., Krebs, A., Thueer, T., & Siegwart, R. (2006). Antarctica Rover Design and Optimization for Limited Power Consumption. *MECHATRONICS - 4th IFAC-Symposium on Mechatronic Systems.*

Mandow, A. M.-C. (2007). Experimental kinematics for wheeled skid-steer mobile robots. *Intelligent Robots and Systems, 2007 (IROS 2007). IEEE/RSJ International Conference on*, (pp. 1222-1227).

Narasimhan, S., Roychoudhury, I., Balaban, E., & Saxena, A. (2010). Combining Model-based and Feature-driven Diagnosis Approaches - A Case Study on Electromechanic Actuators. *21st International Workshop on Prinicples of Diagnosis (DX 10).* Portland, Oregon.

NASA Ames Research Center. (2011). *The Robot Application Programming Interface Delegate Project*. Retrieved from http://rapid.nasa.gov/

Object Management Group. (2004). *CORBA/IIOP specification.* Framingham, MA: OMG.

Patil, N. C. (2009). Precursor parameter identification for insulated gate bipolar transistor (IGBT) prognostics. *IEEE Transactions on Reliability* , 58(2), 271-276.

Poll, S., Patterson-Hine, A., Camisa, J., Nishikawa, D., Spirkovska, L., Garcia, D., et al. (2007). Evaluation, Selection, and Application of Model-Based Diagnosis Tools and Approaches. *AIAA Infotech@Aerospace Conference and Exhibit.* Rohnert Park, CA.

Rasmussen, C., & Williams, C. (2006). *Gaussian Processes for Machine Learning.* Boston, MA: The MIT Press.

Saha, B., & Goebel, K. (2009). Modeling Li-ion Battery Capacity Depletion in a Particle Filtering Framework. *Proceedings of Annual Conference of the PHM Society.* San Diego, CA.

Saha, B., Celaya, J., Wysocki, P., & Goebel, K. (2009). Towards Prognostics for Electronics Components. *IEEE Aerospace*, (pp. 1-7). Big Sky, MT.

Saha, B., Koshimoto, E., Quach, C., Hogge, E., Strom, T., Hill, B., et al. (2011). Predicting Battery Life for Electric UAVs. *AIAA Infotech@Aerospace.*

Saha, S., Celaya, J., Vashchenko, V., Mahiuddin, S., & Goebel, K. (2011). Accelerated Aging with Electrical Overstress and Prognostics for Power MOSFETs. *IEEE EnergyTech, submitted* .

Schmidt, D. (1994). The ADAPTIVE Communication Environment: Object-Oriented network programming components for developng client/server applications. *Proceedings of the 12 th Annual Sun Users Group Conference* (pp. 214–225). San Francisco, CA: SUG.

Schwabacher, M. (2005). A Survey of Data-drive Prognostics. *AIAA Infotech @ Aerospace Conference.*

Smith, M., Byington, C., Watson, M., Bharadwaj, S., Swerdon, G., Goebel, K., et al. (2009). Experimental and Analytical Development of a Health Management System for Electro-Mechanical Actuators. *IEEE Aerospace Conference.* Big Sky, MT.

Wolpert, D. (2006). Information Theory - The Bridge Connecting Bounded Rational Game Theory and Statistical Physics. (D. Braha, A. Minai, & Y. Bar-Yam, Eds.) *Complex Engineered Systems , 14*, 262-290.