Design and In-Water Testing of a Fault-Detection System for Unmanned Underwater Vehicle Actuators

Matt Kemp[1], Jon Erickson[2], Scott Jensen[3], Sotiria Lampoudi[4], and Eric J. Martin[5]

[1-5] *Monterey Bay Aquarium Research Institute, Moss Landing, CA, 95039, USA*
mkemp@mbari.org
jon@mbari.org
sjensen@mbari.org
slampoudi@mbari.org
emartin@mbari.org

ABSTRACT

We discuss the design of a fault-detection system for unmanned underwater vehicle actuators, and present the results of in-situ testing on a mass-shifter. The following design elements are discussed: design objective, actuator selection, failure modes, fault selection, fault injection, fault detection, hardware selection, and software design. In-situ test results are presented and analyzed, including: nominal operation, faulty operation, false-alarm rate, missed-detection rate, detection time, and redundant residuals. Follow-on application to a second actuator is discussed.

1. INTRODUCTION

Unmanned underwater vehicles (UUV) often use electro-mechanical actuators to perform flight control tasks – e.g. thruster, elevator, rudder, mass-shifter, or variable buoyancy system – (Webb, Simonetti, & Jones, 2001; von Alt, 2003; Wernli, 2000). Model-based fault detection was discussed extensively in (Gertler, 1998; Patton, Frank, & Clark, 1989). Moseler and Isermann applied it to fault detection of DC motors (Moseler & Isermann, 2000). Nandi et al. extended this to condition monitoring (Nandi, Li, & Toliyat, 2006). More recently, Fagogenis et al. (Fagogenis, De Carolis, & Lane, 2016) used a Bayesian model with a hidden switch variable to detect partial loss of thrust.

Kemp et al. presented characterization data for a UUV mass-shifter under nominal and faulty conditions (Kemp & Raanan, 2017). They quantified the baseline fault-detection performance using a gaussian detector, and compared it to a one-class support vector machine.

Kemp and Martin discussed fault isolation of a mass-shifter using model-based methods (Kemp & Martin, 2018). They found that despite the model capturing most of the system dynamics, small model errors still caused a high rate of false-alarms during start of motion.

Kemp discussed fault detection of a UUV thruster using both model-free and model-based methods (Kemp, 2019b), and found that the same fault-detection methods used on the mass-shifter applied equally well to the thruster.

Analysis of nominal characterization data on a UUV rudder/elevator actuator indicated that the device operated mostly under transient conditions, and that model-based methods did not perform well due to difficulties modeling black-box non-linearities introduced by the servo-controller (Kemp, 2019a).

In this paper we discuss the design of a fault-detection system for UUV actuators targeted at the rapid development and in-situ validation of algorithms, and an application to a UUV mass-shifter. The paper is organized as follows. Section 2 describes the design. Section 3 presents results collected during in-situ tests. Section 4 analyzes the results. Finally, Section 5 summarizes the findings.

2. SYSTEM DESIGN

2.1. Design Objective

The design objective was to create a system capable of supporting rapid development and in-situ validation of fault-detection algorithms for any of the actuators typically used by UUVs: thruster, rudder, elevator, mass-shifter, drop-weight, or variable buoyancy system. We relied on four design principles to achieve this:

1. house the electronics in a separate compartment – to minimize vehicle re-designs,
2. select high-equality modular off-the-shelf data acquisi-

tion hardware – to minimize development time and maximize flexibility,

3. use a computer running a standard operating system – to minimize development time,
4. implement the software in Python – to maximize access to existing software, and minimize implementation time.

2.2. Mass-Shifter

Although the design intent is general, the initial implementation was specific. The mass-shifter is an electro-mechanical linear actuator that functions to move a large mass - the vehicle's battery – back-and-forth in order to adjust the pitch of the vehicle (Figure 1). It consists of a DC brushed motor and a planetary gear connected to the battery through a lead-screw, and a PI servo-controller in constant velocity mode that receives position feedback from a quadrature encoder. The mass is constrained to move along parallel rails on four wheels – front and back, left and right. The rails are terminated at either extremity by hard travel limits – blocks of aluminum designed to stop the wheels.

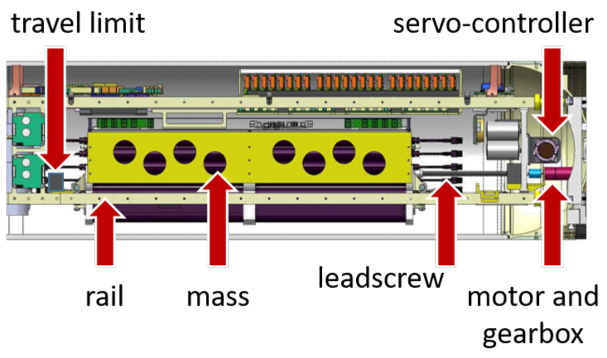

Figure 1. Components of the mass-shifter.

2.3. Failure Modes

The mass-shifter has two modes of failure. The first is an overload fault, which occurs when it exceeds the travel limit. The second is a coupling fault, which occurs when the mechanical coupling between the battery and the motor is lost. The preponderance of failures are due to overload.

2.4. Fault Injection

Fault injection was addressed during the design phase, with a requirement that only faults which can be injected in a reversible and repeatable manner be tested. After reviewing design options, we decided to select overload faults – in-situ reversible coupling faults were discarded because they require substantial vehicle modifications. Reversible overload faults are injected by commanding the mass-shifter past its travel limit but before the overload – i.e. less than 10mm past the travel limit.

2.5. Fault Detection

Mass-shifter fault-detection and isolation can be done using two residuals: 1) a current residual defined as the difference between motor current and its nominal value, and 2) a position residual defined as the difference between absolute battery location – measured with a string potentiometer – and relative battery location – measured with the motor encoder (Kemp & Martin, 2018).

Since the scope of this implementation was limited to overload faults, only the current residual was used. The system however generated four residuals: 1) and 2) above, 3) a speed residual defined as the difference between speed measured with the encoder and its nominal value, which serves to detect a stuck motor shaft or an encoder failures, and 4) a motor residual defined as the difference between speed estimated from the encoder and speed estimated from motor current and voltage, which serves to detect motor or sensor failure:

$$\left.\begin{aligned} r_{current} &= i - i_{nominal} \\ r_{position} &= \alpha x_{motor} - x_{battery} \\ r_{speed} &= \alpha(\omega_{mech} - sign(\omega_{mech})\omega_{nominal}) \\ r_{motor} &= \alpha(\omega_{elec} - \omega_{mech}) \end{aligned}\right\} \tag{1}$$

where α converts motor rotation to linear motion, x_{motor} is the battery position predicted from the encoder, $x_{battery}$ is the battery position from the potentiometer, ω_{mech} is the motor speed calculated from the encoder, and ω_{elec} is the motor speed calculated from the electrical quantities:

$$\omega_{elec} = (V - R_m i)/K_m \tag{2}$$

where V is the motor voltage, i is the motor current, R_m is the motor resistance, and K_m is the torque constant.

The effect of pitch on the nominal current was measured by putting the vehicle on a variable pitch table and measuring the steady-state current over the full pitch range. The resulting data reflected the force required to move against gravity, and the energy dissipation that occurs when moving with it, and is modeled by:

$$i_{nominal} = max(i_{cutoff}, i_0 + K_\phi \phi) \tag{3}$$

when the motor is moving forward and

$$i_{nominal} = min(-i_{cutoff}, -i_0 + K_\phi \phi) \tag{4}$$

when the motor is moving aft, where, ϕ is the vehicle pitch angle.

Parameter identification was done using a combination of manufacturer data (Maxon A-max 22-110138 motor; Maxon GP 22B-110357 planetary gear head; Nook lead-screw with 1 mm/rotation pitch) and model identification. Table 1 summarizes the parameters, their values, and how they were determined.

Table 1. Model parameters.

parameter	name	value	method
R_m	motor resistance	20.2Ω	manuf
K_m	torque constant	21.8mNm/A	manuf
w_{cmd}	commanded speed	419.4rad/s	calib
α	rotation to linear	0.001888 mm/rad	manuf
K_ϕ	pitch constant	0.055 A/rad	calib
i_0	current at 0 pitch	0.03 A	calib
i_{cutoff}	current at cutoff	0.02 A	calib

2.6. Transient Suppression

Kemp and Martin observed that large transients are present at the beginning and end of a commanded move (Kemp & Martin, 2018). To reduce false alarms, we defined a state machine with four states: *idle*, *start transient*, *moving*, and *stop transient*: The state machine transitions from *idle* to *start transient* when the mass shifter starts moving. It transitions to *moving* 500 ms later, at which point it starts the fault detection logic. The *moving* state lasts until the distance-to-target falls below a threshold. The software then enters the *stop transient* state and stops detecting; 500 ms later, it transitions to the *idle* state.

2.7. False-Alarm Rate and Missed-Detection Rate

We defined the false-alarm and missed-detection rates as follows:

- False-alarm rate: fraction of nominal runs during which the current residual exceeded the detection threshold N times in a row.
- Missed-detection rate: fraction of the faulty runs during which the current residual did not exceed the detection threshold N times in a row.

2.8. Component Selection

The hardware design has two blocks. The first consists of a custom signal conditioning printed circuit board (PCB) placed between the motor and the servo-controller, and a string potentiometer attached to the battery and connected to the PCB (Tensor Solutions SP1-4). The PCB performs a series of analog operations: conversion of the motor current to a voltage (shunt resistor), low-pass filtering, conversion to line levels, and buffering. The PCB also provides a clean reference voltage to the potentiometer. All fault detection signals from the PCB – ground, motor current, motor voltage, potentiometer, and motor encoder A and B phases – are sent to the fault-detection computer over a wet-mateable connector located on the main pressure housing's fore bulkhead.

The second block has five components (Figure 2):

- Signal acquisition is done with National Instruments CompactDAQ hardware. NI's CompactDAQ is a family of high-quality customizable and modular signal acquisition cards capable of measuring most signals of interest. Motor current, motor voltage, and potentiometer signals are converted by an NI-9239 analog acquisition module. The motor encoder's A and B signals are converted to motor position by an NI-9361 counter module configured for quadrature counting. The two modules are slotted in an NI-9174 chassis – the 9174 can support up to 4 modules. Synchronization between the modules was achieved by slaving the 9361's sampling clock to the 9239's, and by starting acquisition on the 9239's sampling clock rising edge.
- Vehicle pitch is measured by a Microstrain 3DM-GX5-25 attitude heading reference system (AHRS). The AHRS is mounted underneath the mounting frame.
- The fault-detection software is implemented on a CongaTec PA5 single-board computer (4x1.6 GHz Atom cores) running Windows 10. Windows 10 was selected in order to support the NI-DAQ interface library to the CompactDAQ modules.
- A ConnectTech Com Express CCG020 carrier board provides the interface between the computer and the DAQ and AHRS.
- A custom interface PCB, which performs two functions: converting input power from the vehicle to all necessary supply voltages, and exporting the signals from the signal conditioning PCB to the CompactDAQ modules.

The second block's electronics sits in a 300m-rated pressure housing mounted in the vehicle nose section (Figure 3). Power to the payload is provided by the vehicle. Communication with the vehicle is done over Ethernet

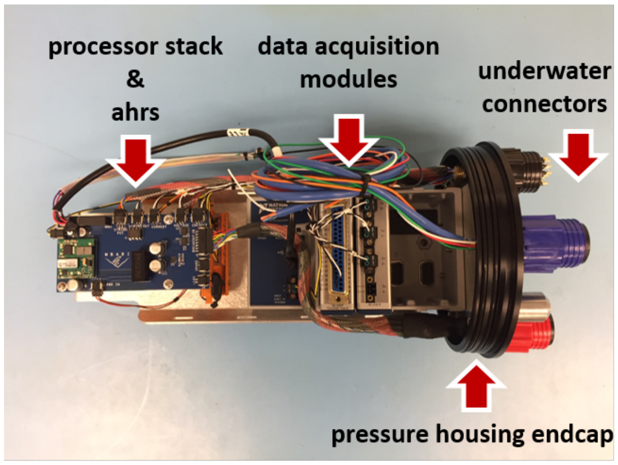

Figure 2. Fault-detection housing.

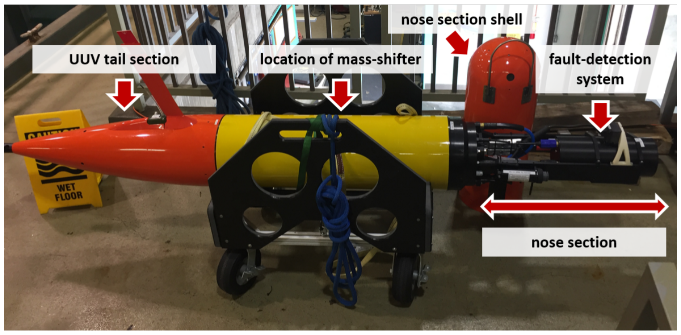

Figure 3. The fault detection system (right) is mounted in the nose section of the UUV.

2.9. Software Design

The monitored quantities (voltage, current, encoder position, position of the mass and pitch) are read by software written in Python running on the payload computer. The software runs in a loop, each iteration of which consumes the last 200 samples from the NI-DAQ – NI-DAQ is NI's API for CompactDAQ modules – the last pitch value from the AHRS, and context supplied by the UUV. The software decides whether to run the fault detection calculations depending on the context, and logs raw and processed data. The loop runs synchronously, at a rate of 2.5 Hz.

While NI-DAQ supplies a low overhead asynchronous logging solution based on the TDMS file format, we found it impractical to use as it could not be read by existing MATLAB TDMS drivers due to binary incompatibility. We chose instead to log all data in the HDF5 format.

Under Windows 10, the Python software is started by the Task Scheduler at boot. This seemingly simple design decision led to the discovery of a long failure cascade and some lessons learned: during integration testing, we found that at times NI-DAQ simply ceased to flow, with no outward sign of error. We traced the root cause to a Windows policy, which prioritizes interactive tasks (e.g. logging in) over scheduled ones (e.g. our application). The result of this policy is that when a user logs in, occasionally our application was being pre-empted, causing the NI-DAQ buffer to overrun. When this occurs, a Python exception is raised, but since the read operation occurs in a callback – which is a Python function called by a C DLL – , the overrun exception must be handled in the callback, or risk being lost as it cannot propagate up the (C) call stack.

To prevent this, two measures were taken. First, all exceptions that may occur in the NI-DAQ callback are handled in the callback itself, as they cannot propagate out of it – this is probably safe practice for callbacks, although it is in conflict with the design imperative to keep their execution time to a minimum. Second, the priority of the scheduled task was elevated above that of interactive tasks. Because process priority is not exposed by the Task Scheduler GUI, we accomplished this by exporting the scheduled task as XML, editing the XML, and re-importing the scheduled task into the Task Scheduler application.

2.10. Communication with Vehicle

The payload was integrated to a Tethys-class Long-Range Autonomous Underwater Vehicle (LRAUV). LRAUVs perform unmanned basin-scale oceanographic measurements, and have an operational envelope of 14 days (Bellingham et al., 2010).

The vehicle's flight control software includes a message passing component based on the Lightweight Communications and Marshalling (LCM) (Huang, Olson, & Moore, 2010). LCM messages are published over multicast on the vehicle's on-board Ethernet, and can be consumed by payloads and peripherals. In our case, the software driver for the mass-shifter was instrumented so as to publish its state (idle, homing or moving) over LCM. The Python software running on the payload was, in turn, written to consume the state messages from the mass-shifter, and used them to drive its own internal state machine, whose states included all those of the mass-shifter driver, as well as two additional states representing a start and stop transient.

Once the vehicle is in the water, the payload is controlled via the vehicle's own flight control software. A C++ component module was added to the vehicle's flight controller to control payload power-up.

3. RESULTS

3.1. Test Protocol

To conduct in-situ experiments, we instructed the vehicle to dive to 4 m, shut its propeller down, and null its buoyancy and pitch. The vehicle was then instructed to move the mass-shifter in one of three ways:

- Nominal: from the center position (50%), move to 75% range (+20mm), then to 25% (-18mm), and finally back to center.
- Developing fault: move to within 1mm of the travel limit, then past the limit within 4mm of the failure point, then back to 1mm before the limit.
- Critical fault: move to within 1mm of the travel limit, then to within 1mm of the failure point, then back to 1mm before the limit.

We performed at total of 495 nominal runs, and 30 fault runs.

3.2. Nominal Behavior

Figure 4 shows the current and current residual during a nominal run from 75% to 25% range (i.e. from 20mm to -18mm). The insert shows the pitch of the vehicle during the move. The current is highest during the initial part of the move, as the mass-shifter is ascending against gravity. As the pitch decreases, the current decreases. When the pitch changes sign, the mass-shifter acts as a brake. The current residual is O(10mA).

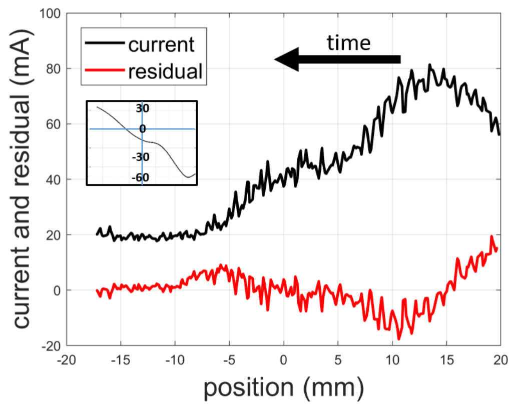

Figure 4. Current (blue) and current residual (red) during nominal operation. Motion is from right to left. The current decreases from 80mA to 20mA due to pitch changes. The residual is O(10mA).

3.3. Fault Behavior

Figure 5 shows the current and residual during an overload fault: motion starts just short of the travel limit (at 37mm), and continues until the overload failure (at 46mm). The current increases 10X during the move, and the residual 15X.

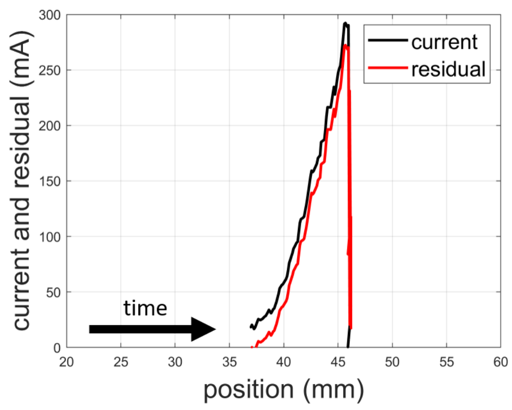

Figure 5. Current (blue) and current residual (red) during an overload fault. Motion is from left to right. The overload occurs at 46mm. Current and residual grow from O(10mA) to 300mA just before the overload.

3.4. False-Alarm Rate

Per our definition, the false-alarm rate is the fraction of nominal runs during which the current residual exceeded the detection threshold N times in a row. Figure 6 shows the false-alarm rate over 495 runs, as a function of the detection threshold, and for different minimum fault lengths N. We found that the false-alarm rate was 0 when the threshold exceeded 30mA, independent of N.

For thresholds below 30mA, we found that 1) the false alarm rate decreased exponentially fast with threshold, and 2) that the false-alarm rate decreased with N.

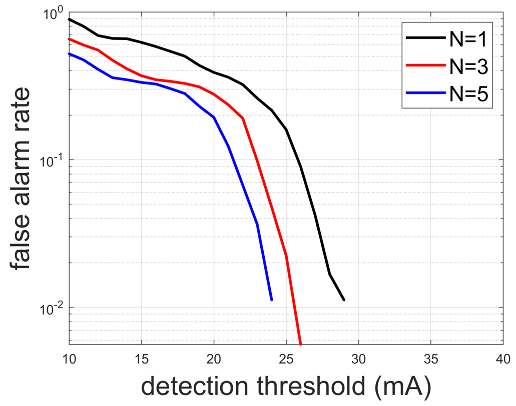

Figure 6. False-alarm rate versus detection threshold for different minimum fault lengths N.

3.5. Missed-Detection Rate

Per our definition, the missed-detection rate is the fraction of faulty runs for which the current residual did not exceed the detection threshold N times in a row.

We performed a total of 30 faulty runs, and found no missed-detections across the range of thresholds and minimum fault lengths.

Figure 7 shows the average time elapsed between initial contact with the travel limit and the detection of a fault, as a function of detection threshold and for different minimum fault lengths N. At a detection threshold of 30mA, the detection time is 3 second or less depending on N. For other values, the detection time increases monotonically with threshold, and shifts up with increasing N.

4. DISCUSSION

4.1. False-Alarms Rate

We found that the false-alarm rate taken over 495 nominal runs can be driven to 0 with a modest detection threshold. The fundamental reason for this is that the residual is 15X larger at overload then in nominal condition (Figures 4 and 5).

Going deeper, if we assume that the nominal residual has a gaussian distribution of the correct standard deviation (10mA), then setting the desired false-alarm rate to a very large value, say once per 10^{10} samples, could be met with

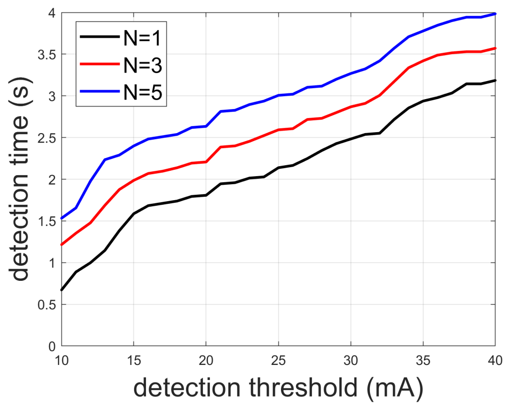

Figure 7. Detection time versus detection threshold for different minimum fault lengths N.

a threshold of 60mA, still 5X below the observed overload residual!

Detailed examination of the data indicated that the false-alarms observed at low thresholds occurred at the start of motion. We also found that they were caused by data-buffering latencies. Referring to Figure 8, when the fault-detection system receives a *start of motion* message from the vehicle, it pulls the most recent available data buffer for processing (shown in black). Because buffer size is 125ms, the processed data therefore precedes the event by anywhere between 125 and 250ms. The result is that the 500ms transient rejection delay is occasionally insufficient, i.e. part of the large transient filters through.

Predictions of this analysis are consistent with the data: 1) increasing the minimum fault length should decrease false-alarms, as it effectively decreases the latency (Figure 6), and 2) increasing the detection threshold should decrease false-alarms, by filtering out those false-alarms with a smaller buffering latency.

Possible remedies include increasing the transient rejection delay, and reducing the data buffer size.

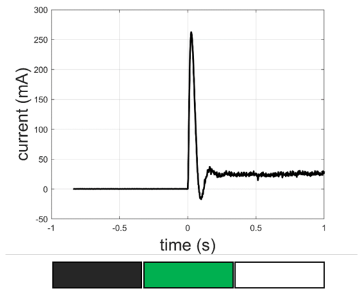

Figure 8. Origin of the false-alarms. Top: startup transient. Bottom: data buffering diagram. The data buffer currently being filled is in green, and the available data buffer is in black.

4.2. Detection Horizon

We showed (Figure 7) that the detection time is of order 3 seconds. Since it takes 10s from initial contact to system failure, it means that the detection horizon is 7s.

Seven seconds is large compared with the vehicle's 2.5Hz control loop. As a result, fault avoidance is quite possible: one could envision, for example, a scheme where after detecting a developing overload fault, the actuator would stop and return to the point where the fault first occurred, and possibly update its internal software travel limit accordingly.

4.3. Detection Time

We showed that the detection time increases with the detection threshold and with the minimum fault length N. To explain this, we refer to the data showing the growth of the residual during a fault (Figure 5), and ask: how much time does it take for a detection to occur?

The detection time is the sum of two terms: how long it takes to cross the threshold the first time, and how long it takes to cross it $N-1$ more times. We can write down the result by inspection:

$$t_{detection} = \frac{r^{-1}(threshold)}{v} + (N-1)t_{detectionloop} \quad (5)$$

where $r(x)$ is the current residual from Figure 5 and v is mass-shifter speed, and where we've taken account of the constant speed maintained by the servo-controller, and the rapid growth of the current. This result shows that a higher threshold implies a larger detection time, and that N shifts the detection time up.

4.4. Position Residual

The results we presented are based on the current residual, but as explained earlier three additional residuals are being generated: position residual, speed residual, and motor residual. Whereas the original intent of the position residual was to detect coupling failures, we found that it is in fact a strong indicator of overload faults.

The position residual is the difference between the position estimated from the motor encoder and a string potentiometer attached to the mass-shifter. Figure 9 shows that the residual is less than 0.5mm during nominal operation, but grows 16X during an overload fault, to 8 mm.

The reason for the growth is that, whereas the servo-controller maintains constant motor speed even when pushing against the travel limit, the mass slows down because the wheels and the frame compress (Figure 10).

The position residual could provide a fault confirmation

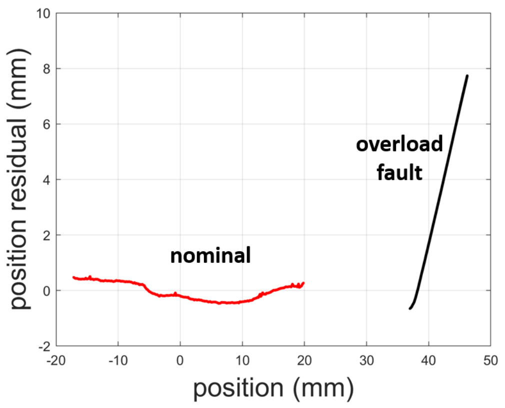

Figure 9. Position residual during nominal operation (red) and during an overload fault (black). The residual drastically increases when contact with the travel limit occurs.

mechanism for faults generated by the current residual; additionally, it could provide a way of discriminating between current overloads due to the travel limit and the short current spikes that are occasionally observed far from the travel limit.

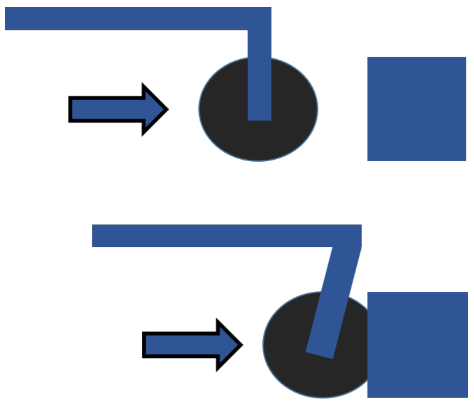

Figure 10. Deformation of the mass-shifter wheel and frame during an overload fault.

4.5. Thruster Fault Detection

The same fault-detection system used for the mass-shifter can be used for thruster fault detection. As discussed in (Kemp, 2019b), the LRAUV thruster is a 3-phase DC brushless motor, driven by the same EZSV23 servo controller as the mass-shifter but with feedback from the Hall effect sensors instead of an encoder. The net motor current can be measured with the same two wires used by mass-shifter, using only one of the three current phases and conputing 1.5 times its mean absolute value. Similarly for the motor voltage, it can be measured with two wires using any of the three inter-phase voltages and performing the same operation. Motor speed can be measured using two of the three Hall sensors, as if they were the A and B phases of a quadrature encoder.

From a hardware perspective, the only difference between the two is signal level, i.e. different sense resistors have to be used on the vehicle-side PCB.

From a data acquisition perspective, the same hardware used on the mass-shifter – voltage and counter modules – will work for the thruster.

From a software perspective, the main differences are the context data provided by the vehicle – motion status, commanded and actual position for the mass-shifter; motion status, commanded speed, and vehicle depth for the thruster –, and the algorithm. Because of a design decision to implement the software in Python using a computer running a standard OS, conversion of an algorithm developed in Matlab to an online implementation is expected to be seamless.

The biggest difference between the two is expected to be the fault-detection algorithm. The mass-shifter depends on a single environmental variable – vehicle pitch – whereas the thruster operates in a host of conditions that depend on external factors – straight and level flight, idle, surface swimming, diving, climbing, etc. Unlike the mass-shifter where a complete characterization was possible in the lab, only limited lab characterization can be done on the thruster. At-sea characterization tests will take place in Fall 2019.

5. CONCLUSION

We discussed the design and testing of a fault-detection system for UUV actuators, intended to support the rapid development and in-situ validation of actuator fault-detection algorithms. An application to a UUV mass-shifter was implemented, and results of in-situ testing were presented. We found no false-alarms in 495 nominal runs, and 30 out of 30 successful detections of injected faults. We analyzed these results, and found them to be consistent with expectations. Plans for in-situ testing of thruster fault-detection using the same hardware were discussed.

ACKNOWLEDGMENT

This work was done with financial support from the Packard Foundation.

REFERENCES

Bellingham, J. G., Zhang, Y., Kerwin, J. E., Erikson, J., Hobson, B., Kieft, B., & Banka, A. (2010). Efficient propulsion for the tethys long-range autonomous underwater vehicle. *Proceedings of IEEE/OES Autonomous Underwater Vehicles Conference*.

Fagogenis, G., De Carolis, V., & Lane, D. M. (2016). Online fault detection and model adaptations for underwater vehicles in the case of thruster failures. *IEEE International Conference on Robotics and Automation*.

Gertler, J. (1998). *Fault detection and diagnosis in engineering systems*. Marcel Dekker Editor.

Huang, A. S., Olson, E., & Moore, D. C. (2010). Lcm: Lightweight communications and marshalling. *Proceedings of IEEE/RSJ Conference on Intelligent Robots and Systems*.

Kemp, M. (2019a). High variability in the nominal response of a rudder. *Proceedings of Prognostics and Health Management Society conference*.

Kemp, M. (2019b). Underwater thruster fault detection and isolation. *Proceedings of AIAA*.

Kemp, M., & Martin, E. (2018). Fault isolation of an electromechanical linear actuator. *Proceedings of Prognostics and Health Management Society conference*.

Kemp, M., & Raanan, B. (2017). Actuator fault detection for autonomous underwater vehicles using unsupervised learning. *Proceedings of the Annual Conference of the PHM Society*.

Moseler, O., & Isermann, R. (2000). Application of model-based fault detection to a brushless dc motor. *IEEE Transactions on Industrial Electronics*.

Nandi, S., Li, X., & Toliyat, H. (2006). Condition monitoring and fault diagnosis of electrical motors. *IEEE Transactions on Energy Conversion*.

Patton, R., Frank, P., & Clark, R. (1989). *Fault diagnosis in dynamic systems*. Prentice Hall.

von Alt, C. (2003). Remus 100 transportable mine countermeasure package. *Proceedings of Ocean 2003*.

Webb, D., Simonetti, C., & Jones, C. (2001). Slocum: an underwater glider propelled by environmental energy. *IEEE Journal of Oceanic Engineering*, *26*, 447-452.

Wernli, R. (2000). Auv commercialization - who's leading the pack. *Proceedings of Ocean 2000*.

BIOGRAPHIES

Dr. Matt Kemp is a Principal Engineer at the Monterey Bay Aquarium Research Institute in Moss Landing CA. He holds a Ph.D. in Physics from the University of North Carolina at Chapel Hill. Dr. Kemp served as Director of Concept Development with Bluefin Robotics for 5 years, and Director of Concept Development with Nekton Research for 7. His research interest is in unmanned underwater vehicle health management. He is a member of IEEE, PHM, AIAA, and AAAS.

Jon Erickson is a Mechanical Engineer with forty years in the Marine Research field. Jon's primary interests are in new materials and manufacturing technology, and their application to a marine environment.

Scott Jensen is an Electrical Engineer at MBARI. His work encompasses microprocessors, digital, analog and power electronics, system design, and instrument support.

Dr. Sotiria Lampoudi is a Software Engineer. She holds a Ph.D. in Computer Science from the University of California in Santa Barbara. Her research interests include field robot coordination and robotic OS and Command and Control architecture.

Eric Martin is an electrical engineer at MBARI. He holds an M.S. in Ocean Engineering from the University of Rhode Island. His research interests are ocean instrumentation, remotely operated vehicles, and virtual reality.