2014 ASHRAE/IBPSA-USA
Building Simulation Conference
Atlanta, GA
September 10-12, 2014

AN FMI-BASED TOOLCHAIN FOR THE ADOPTION OF MODEL-BASED FDD

Marco Bonvini, Michael Wetter, and Michael D. Sohn
Environmental Energy Technologies Division, Lawrence Berkeley National Laboratory
1 Cyclotron Road, 94720, Berkeley, CA

ABSTRACT

This work presents a toolchain based on the Functional Mockup Interface (FMI) standard for accelarating the adoption of model-based Fault Detection and Diagnostics (FDD) algorithms. The presented toolchain uses nonlinear state estimation techniques to detect multiple simultaneous faults in presence of noisy sensor measurements. The toolchain enables the use of a dynamic model that emulates correct and faulty equipment, while keeping the computing effort low. The FDD toolchain allows importing dynamic models through an interface that complies with the FMI standard, an application programming interface for the exchange of models which is supported by many simulation tools. This functionality reduces the modeling effort required to setup a model-based FDD algorithm because it allows taking advantage of models available in open-source libraries for building components and HVAC systems that can model various fault scenarios.

INTRODUCTION

In large commercial building energy systems, equipment routinely degrades or malfunctions. Set-points, valves, controls, and schedules are often adjusted manually for a specific or unique event and then not returned to their normal operation. Unfortunately, a sufficient number of sensors are rarely deployed in most buildings or building systems to detect these faults in a reasonably short period of time. In many commercial heating, ventilation, and air conditioning (HVAC) systems, various types of substandard operations can occur, leading to uncomfortable occupant conditions, damage to equipment, and energy waste. Just 13 of the most common faults in U.S. commercial buildings in 2009 are estimated to have caused over $3.3 billion in energy waste (Mills 2009). Building hardware and software for simple and rapid fault detection and diagnosis (FDD) are a key tool to reduce some of these significant energy wastes.

This work presents a toolchain for accelerating the adoption of FDD algorithms that use advances in the nonlinear state estimation techniques to detect multiple simultaneous faults even in presence of noisy sensor measurements. The presented approach allows the use of first principles dynamic models that may account for several operating modes, while keeping the computing effort low enough to provide fault estimates within few seconds. The presented toolchain also reduces the modeling effort required to setup a model-based FDD because dynamic models can be integrated into the FDD algorithms using the Functional Mockup Interface (FMI) standard, a standard for exporting simulation models supported by many simulation programs (e.g. Dymola, OpenModelica, Matlab, etc.). The framework takes advantage of the models available in open-source libraries such as the Modelica Buildings Library (Wetter et al. 2014) to setup real-time FDD strategies for buildings components and HVAC systems.

The major contribution of the work is the integration of state estimation and filtering techniques with models that comply with the FMI standard. The result is a FDD toolchain that can easily leverage the models used during the design phase in the commissioning or operation. Reusing models lead to a better identification of the causes of a fault; nonetheless it saves time and money.

The paper is structured as follows. The first section contains a brief literature review that supports the need for model-based FDD strategies. The second section provides a description of the proposed FMI model-based FDD framework. The last section contains an example that shows how the proposed framework can be coupled with a Modelica model to set up an FDD algorithm.

REVIEW ON FDD

There is a large body of literature about FDD strategies, both generally (Venkatasubramanian et al. 2003c; Venkatasubramanian et al. 2003b; Venkatasubramanian et al. 2003a), and in the context of building-specific applications (Katipamula and Brambley 2005a; Katipamula and Brambley 2005b). These works classify three categories of FDD methods: quantitative model-based, qualitative model-based and process history-based. The authors also provide a systematic comparison of these methods.

Although FDD is not used in the business-as-usual operation and maintenance of today's commercial buildings, the literature provides many examples of process history-based and qualitative and quantitative model-based FDD developed for building HVAC applications. Desired attributes of FDD approaches for building applications include:

- Robustness to sensor errors and data paucity.
- Applicability to both dynamic and steady-state equipment operation.
- Ability to identify numerous or simultaneous faults.

- Fast enough for near-real time execution and feedback to operators.

Near real-time execution enables embedding FDD in commercial performance management tools. It has the potential to minimize energy wastes, shorten unsatisfactory occupant conditions, or reduce the chance of rapid component degradation. While critical in some FDD applications, the need to detect novel faults not known a priori is less important in buildings because the most common problems with highest energy and operational impacts are well documented. For example Comstock, Braun, and Groll (2001) provide a comprehensive review of chiller faults and similar documents exist for other building systems and components. The need for FDD techniques that are robust to the poor-quality sensor data in buildings has been reported by Dexter and Pakanen (2001). Several solutions have been investigated to build robust FDD strategies. Two examples are algorithms that use adaptive thresholds (Zhou, Wang, and Ma 2009), or principal component analysis (PCA) techniques for the detection of sensor faults in air handling units (Wang and Xiao 2004) and centrifugal chillers (Wang and Cui 2006). While robustness to sensor error was demonstrated in the PCA approach by Wang and Cui (2006), the comparison of FDD techniques provided by Venkatasubramanian et al. (2003a) indicates that both PCA and neural networks (NN) approaches may have difficulty in identifying multiple simultaneous faults, which is another desired characteristic of FDD for building applications. (Venkatasubramanian et al. 2003a) showed that the approaches based on state observers, also known as state estimators, are thought to be more suitable for detecting multiple faults and dealing both with nonlinearities of the models and uncertainties. A drawback of observer-based strategies is that they require a higher modeling effort, e.g., to include explicit fault descriptions in the model itself. However, various modeling tools (e.g. EnergyPlus, TRNSYS), and modeling libraries for buildings (Modelica Buildings library (Wetter et al. 2014)) are available, and they provide two main advantages: reducing the effort required to set up the model, and reducing the risk of modeling errors since they are validated and tested. This work presents a model-based FDD toolchain that is based on advanced nonlinear state estimation techniques and leverages the FMI standard. The main advantage of this approach is the ability to connect the FDD algorithm with models that are already available in the building simulation community, such as the Modelica Buildings library, thereby reducing the effort required to set up model-based FDD algorithms.

THE FMI STANDARD

The Functional Mockup Interface (FMI) is a tool independent standard that supports the exchange of dynamic models for co-simulation. The development of FMI was initiated within the ITEA2 project MODELISAR, a project that involved 29 partners among manufacturers, simulation tool vendors and research institutes. The project started in 2008 and ended in 2011. The primary goal of FMI was to support the exchange of simulation models between suppliers and equipment manufacturers when different simulation programs are used. Today more than 40 simulation programs support the FMI, and the standard is now managed by a group of companies, institutes and universities organized under a noprofit association. Applications that use the FMI standard range from software/model/hardware-in-the-loop simulation to embedded systems.

A simulation model exported according to the FMI standard is called a Functional Mockup Unit (FMU). An FMU comes in the form of a zip-file, which contains the FMI model description file, which is an XML-file with information needed by an import tool, C-code and/or shared libraries required to interface with the model or simulation tool, resource files such as tables, and documentation. Simulation programs or software packages that implement the FMI standard can import FMUs, access their resources, and run simulation.

THE FMI-BASED toolchain

The use of computer based tools for the design of buildings and HVAC systems is increasing. These tools embed the knowledge needed to predict performances and behavior of HVAC and buildings. These information are helpful to support designers' work. For example engineers use simulation programs to study the behavior of the building or HVAC system under certain conditions like hot summer and cold winter days. If the results of the simulations satisfy the design requirements, they proceed with the construction, commissioning and finally operation. During the operation, problems causing energy wastes or component degradation may appear. At this stage FDD techniques are needed. Ideally, FDD techniques should be able to directly access the knowledge established during the design phases and reuse it during the operation.

The presented FDD toolchain allows taking advantage of the models developed during the design stages and establishing a connection between design and operation, two typically disconnected tasks. The presented toolchain provides such a connection based on the FMI standard. The toolchain reduces the effort required to set up FDD strategies that use model-based approaches, easing the adoption of these techniques. The grey area shown in Figure 1 represents the FMI-based FDD toolchain. The toolchain links the simulation programs, typically used by designers, with Energy Information Systems (EIS) that are used by building operators, energy manager and technicians. The FMI standard interface allows such a link. The

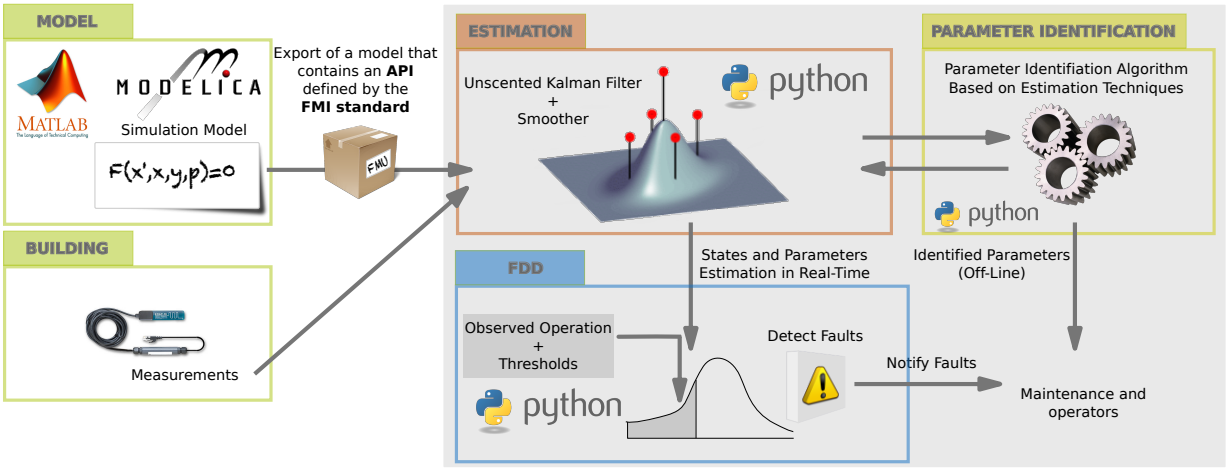

Figure 1: The FMI-based FDD toolchain

toolchain is composed of three main components. (1) A real-time state estimator and smoother. The state estimator is a technique that estimates any variables in real-time, while the smoother improves the quality of the estimations when measurements are noisy. (2) A parameter estimator that estimates in real-time parameters or can be used to calibrate the models off-line. (3) The FDD module that post-processes the results of the estimators and computes fault status and probabilities. The following subsections describe the details of the toolchain as well some details of the three main components.

The FMI Standard Interface

Figure 1 shows the FDD toolchain. The toolchain can import a model exported according to the FMI standard (i.e., an FMU). A model that has been developed in a simulation program like OpenModelica, Dymola or Matlab/Simulink, can be exported as an FMU and used within the toolchain. The FMU model has to be compliant with the FMI standard for model exchange version 1.0. An FMU that complies with this version of the FMI standard provides two main features required by the FDD algorithm: state and parameter reinitialization. This constraint on the FMUs limits the applicability of this approach. For example EnergyPlus provides an FMU interface but it does not provide access to its internal state variables. Models developed using the Modelica modeling language and exported as FMUs by tools like Dymola, OpenModelica or JModelica satisfy this requirement. The main components of the toolchain using the FMU are the state estimator, the smoother and the parameter estimator. The estimators (as well as the other components of the toolchain) have been written in Python and they use PyFMI, a python package for the simulation of models compliant with the FMI standard (Modelon 2013).

Python Based Infrastructure

The infrastructure of the FDD toolchain has been implemented in Python. Python is available for all major operating systems: Windows, Linux/Unix, OS/2, Mac as well as cheap embedded hardware prototyping platforms based on ARM architectures (e.g., RaspberryPI, BeagleBone, etc.). We selected Python as it has been particularly used for the development of scientific applications because it allows for rapid prototyping of algorithms, and provides different modules optimized for scientific computing and data visualization (numpy, SciPy, matplotlib). In the recent years, Python has proven to be a good solution for the development of web servers and web-based applications. All these characteristics make the presented toolchain flexible since it can scale from embedded systems to web-based tools running on cloud computing platform (e.g., Amazon Web Services), depending on the nature of the application. The toolchain can simulate and work with models compliant with the FMI standard using PyFMI.

The toolchain interfaces with an Energy Information Systems or any other software that can acquire measuremenst from sensors located in the building and save them into a Data Base Management Systems (DBMS). It is important that the toolchain can get the data from the DBMS in different formats. For example, in real world applications, the data may be stored in excel files every month in order to be processed by energy managers. The toolchain takes data from Excel files, CSV files, or directly from a DBMS system and move them to its internal data structure. For an application currently developed for the U.S. Department of Defense, we are using IBM DB2 as the DBMS. However the toolchain can be extended to interface with other data bases.

The internal data structure of the FDD toolchain is based

on data series that are stored in memory. There are two reasons for working with data in memory: first it improves the access speed and, second, data are already stored in a DBMS need not be replicated. The data are managed using Pandas (McKinney 2012), a Python module for data analysis that is typically used to manage up to 10 GB of data. Pandas provides advanced querying, filtering as well as analytic functionalities.

Real-Time State Estimator and Smoother

The state estimation technique used in the presented FDD algorithm is the key component of the entire toolchain. Our implementation uses the Unscented Kalman Filter UKF (Julier and Uhlmann 1996). UKF is a Bayesian model-based technique that recursively estimates the state variables of a generic system, as defined by a set of ordinary differential equations (ODEs). The ODEs may represent the thermal model of the building, an HVAC plant or a chiller. The state variables of the system that can be estimated by the UKF are typically temperatures, pressures, and energies. Since complications from building variability and data uncertainties are usual, introducing a back smoother can reduce their effect on the estimation. The back smoother is an additional refinement of the UKF to improve the model-to-data reconciliation and rejecting the effect of noise on the measurements as more data become available (Sarkka 2008). The FDD algorithm will have difficulty with some forms of non-random noise, such as auto-correlation, miss- calibration, and calibration drift. The type of noise would result in less information content in a measurement, so the algorithm may take more samples, and thus take more time, to detect with high probability a fault. Both the UKF and the smoother require running different simulations. This requirement can affect the computing time of the toolchain, in particular when the model is complex. However, since the simulations are independent from each other, they can be run in parallel. This feature may be important when real-time estimations are needed. The toolchain uses a parallelized implementation of said techniques that is compliant with the FMI standard (Bonvini, Wetter, and Sohn 2014). The parallel simulation does not involve the simulation programs that generates the FMUs. The toolchain dispatches the simulation to the available processors and then they are performed by one of the ODE solvers provided by PyFMI. The requirements of the UKF and the smoother are limited to functionalities provided by models exported according the FMI standard for model exchange version 1.0. The requirements include the possibility to initialize state variables, parameters, and running simulations.

Parameter Estimator

The toolchain contains a UKF and a smoother that have been extended to estimate parameters in addition to state

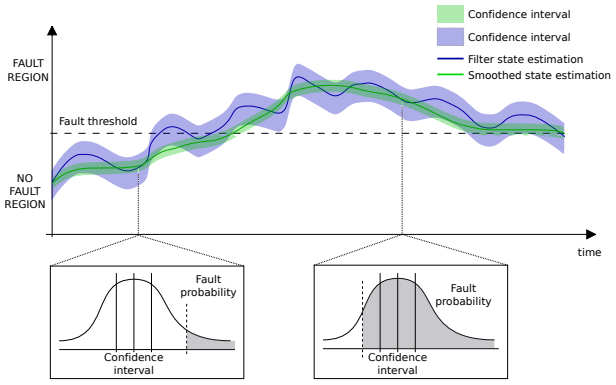

Figure 2: Estimation of the variable representing the fault, together with its σ-confidence interval, and how it can be seen as a fault probability.

variables (Wan and Van der Merwe 2000). State variables vary with respect time and their evolution is governed by differential equations. Parameters are quantities that do not change as a function of time during normal operation. Examples of parameter are heat exchange coefficients, or the thermal conductivity of a wall. The UKF and the smoother are capable of computing parameter estimations in real-time. The real-time estimations of parameters are used together with the state variables estimations to detect anomalies. There are other techniques that use the ability of state and parameter estimation and smoothing techniques for identifying parameters. One example is the coupling between the Expectation Maximization algorithm and the unscented smoother proposed by Yokoyama (2011). Unfortunately, these type of algorithms require a higher computational effort and, hence, may not be applicable to identify parameters in real-time. However they can be used to calibrate models, for example during the commissioning phase.

Fault Detection and Diagnosis

Once the state and parameter estimations have been computed, they need to be converted to fault probabilities. As shown in Figure 1, this step requires additional information based on the knowledge of the monitored system. This knowledge can be used to define thresholds or values associated to the fault free operation. Once these thresholds have been identified, they define a fault region, a subset of the parameter or variable space representing the faulty operation. Given the fault region, it is possible to compute the probability of the fault region containing the estimated variable. Figure 2 shows this process. The estimator and the smoother compute, sequentially, the mean and covariance of the fault variable or parameter. The estimations are Gaussian variables. The probabilities of faults are computed as

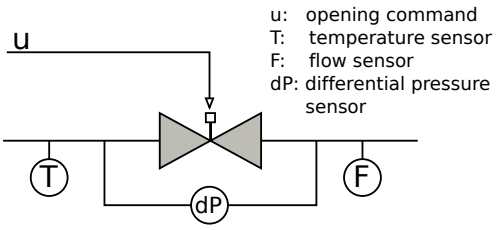

Figure 3: Schematic of the system.

$$\int_{\Omega} P(s)ds \tag{1}$$

where $P(\cdot)$ is the probability distribution function (pdf) of the fault variable or parameter, and Ω indicates the fault region (in Figure 2 the gray area subtended by the pdf). Note that the fault region boundaries can be adjusted adaptively, for example to enhance the ability of the algorithm to detect faults or avoid false positives.

EXAMPLE

This section contains an example that shows how the FMI-based fault detection algorithm can be used to identify faults in a valve in presence of noisy and wrong measurements. The example emulates measurements using simulations for the subsystem shown in Figure 3. During the simulation, two faults have been added: the first fault represents a leakage while the second fault is a blockage. The data generated by simulation have been corrupted by noise in order to test the robustness of the algorithm. The considered system is a valve that regulates the water flow rate in a water distribution system. The system is

$$\dot{m}(t) = \phi(x(t))A_v\sqrt{\rho(t)}\sqrt{\Delta p(t)}, \tag{2a}$$

$$x(t) + \tau\dot{x}(t) = u(t), \tag{2b}$$

where $\dot{m}(\cdot)$ is the mass flow rate passing through the valve, $\Delta p(\cdot)$ is the pressure difference across it, $u(\cdot)$ is the valve opening command signal, $x(\cdot)$ is the valve opening position, τ is the actuator time constant, $\phi(x(\cdot))$ is the power-law opening characteristic, A_v is the flow coefficient and $\rho(\cdot)$ is the fluid density. In our implementation the square root of the pressure difference is regularized around zero flow for numerical reasons. The system has three sensors that measure the pressure difference across the valve $\Delta p^N(\cdot)$, the water temperature $T^N(\cdot)$ and the mass flow rate passing through it $\dot{m}^{N+D}(\cdot)$. All the sensors are affected by measurement noise. In addition, a thermal drift affects the mass flow rate sensor. The measurement equations are

$$T^N(t) = T(t) + \eta_T(t), \tag{3a}$$

$$\Delta p^N(t) = \Delta p(t) + \eta_P(t), \tag{3b}$$

$$\dot{m}^{N+D}(t) = (1 + \lambda(T(t) - T_{ref}))\dot{m}(t) + \eta_m(t), \tag{3c}$$

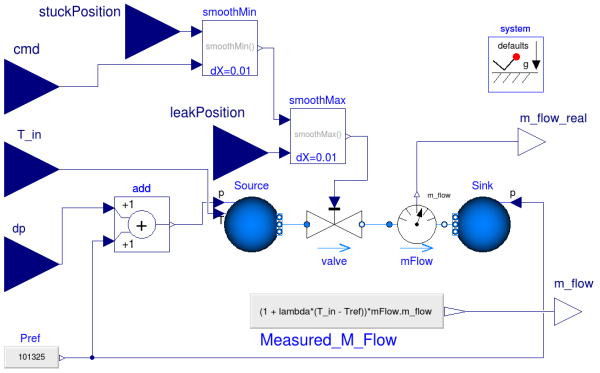

Figure 4: Schematic diagram of the Modelica model.

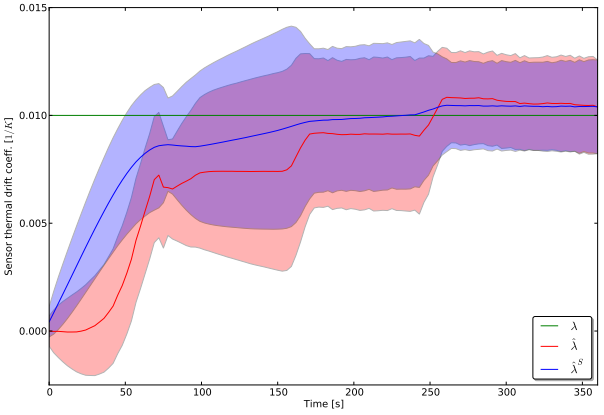

Figure 5: The green line is the sensor thermal drift coefficient, while the red and blue lines are the estimations of the thermal drift coefficient computed by the UKF and smoother (the area surrounding the estimation is the σ-confidence interval).

where the superscript ^{N} indicates a measurement affected by noise, the superscript $^{N+D}$ indicates the presence of both noise and thermal drift, T_{ref} is the reference temperature at which the sensor has no drift, λ is the thermal drift coefficient and $\eta_T(\cdot)$, $\eta_P(\cdot)$, and $\eta_m(\cdot)$ are three uniform white noises affecting the temperature, pressure and mass flow rate measurements. These signals are sampled every two seconds. The model has been implemented in Modelica and its schematic diagram is shown in Figure 4.

Discussion

During the operation, at t = 80 s, the valve becomes faulty. The fault affects the ability of the valve to control its opening position. The valve opening cannot go below 20% (causing a leakage) and over 60% (it gets stuck). At t = 250 s, the valve stuck position changes from 60% to 90%. Both faults are observable only when the opening command signal reaches the fault area (i.e., the leackage is

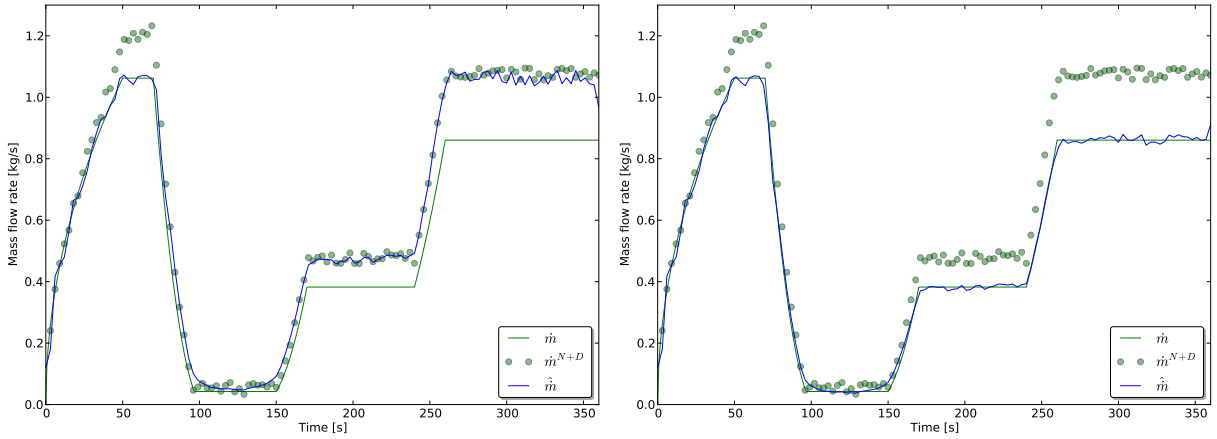

Figure 6: Measured mass flow rate (green scattered points); real mass flow rate (green line); estimated mass flow rate using the UKF+Smoother (blue line). The image on the left shows the estimation that does not account for sensor drift, while the image on the right includes the estimation of the sensor drift.

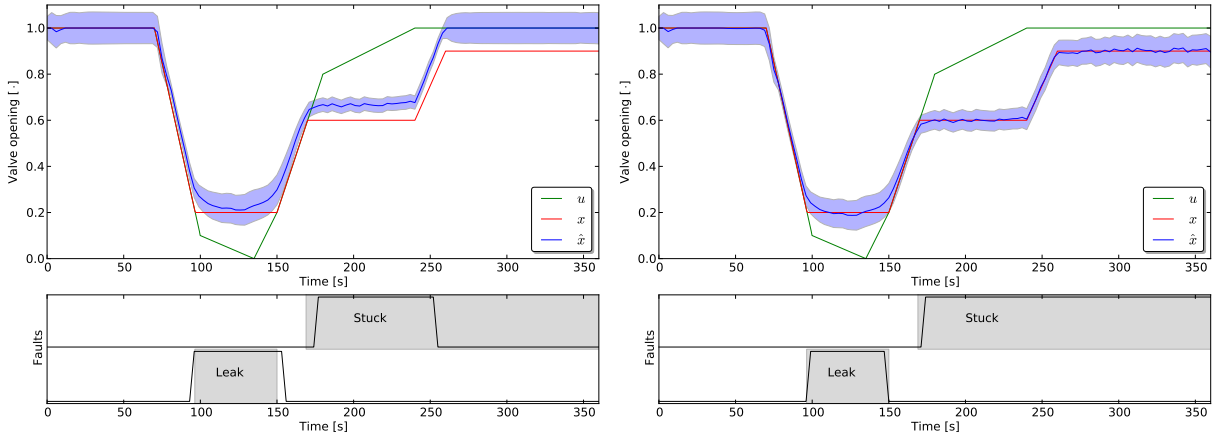

Figure 7: Valve opening command (green line); actual opening (red line); estimated opening using the UKF and Smoother (blue line). The blue area around the estimation represents the σ-confidence interval. The black lines represents the fault status identified by the FDD algorithm, while the grey area represents the presence of a fault. The image on the left shows the estimation that does not account for sensor drift, while the image on the right includes the estimation of the sensor drift.

observable only when the valve is almost closed).

The fault identification procedure identifies when the valve does not work as expected, that is its opening position differs from the command signal. The fault identification uses the UKF that has as input signals for its model the noisy pressure difference, the noisy water temperature and the command signal. Two experiments have been performed. The first experiment tries to estimate the fault conditions (i.e., valve stuck or valve leaking) without accounting fro the sensor thermal drift; the second experiment includes the model of the sensor (see equation (3)) affected by thermal drift.

Figure 6 compares the outputs of the UKF and the smoother with the estimation of measured mass flow rate that is affected by both noise and thermal drift. The effect of the thermal drift is visible where the green dots representing the measured mass flow rate diverge from the green line that is the actual mass flow rate. The UKF and the smother estimate the valve opening position $x(\cdot)$. Figure 5 shows that during the experiment that includes the detailed model of the sensor, the thermal drift coefficient λ is estimated too. Figure 6 shows that the measured mass flow rates (green dots) are far from the real mass flow rates (green line). The UKF and the Smoother compute an estimation of them. As shown in Figure 6 (left) the filter and the smoother try to reconcile the estimation with the

measured data. As shown in Figure 6 (right), an advanced description of the measurement process introduces a benefit, in fact the estimated mass flow rates $\hat{x}(\cdot)$ converge to the real unknown values $x(\cdot)$. Figure 5 shows the UKF and the smoother estimates of the thermal drift coefficient λ. They both start with an initial value equals to zero and converge to a neighborhood of the real unknown value. The smoother improves the estimation of λ at the beginning of the simulation experiment.

Figure 7 shows the estimation of the opening position $\hat{x}(\cdot)$ with respect to the command signal $u(\cdot)$. The green line is the opening command while the red line is the actual opening position. Every time there is a discrepancy between the lines, the valve has a fault. Since the actual position is not known and cannot be measured, the UKF and the smoother estimate it. The blue line is the estimation of the opening position. The blue area around the estimation represents the σ-confidence interval. Once the estimate of the position is available, it can be used to determine if the valve has a fault or not. The black lines in Figure 7 represent the detection of faults associated to the valve, while the grey areas indicate when the fault conditions have been introduced. As expected, when the FDD algorithm uses a model that includes the measurement equations, the faults are correctly identified, as shown in Figure 7 (right). When the model does not contain the measurement equation it does not recognize a fault when the valve gets stuck at 90% at t = 250 s. As expected the smoother is able to provide a better estimation since it uses more data.

CONCLUSION

We presented an FMI-based FDD toolchain that uses advanced state and parameter estimation techniques for dynamic nonlinear systems. As the tool-chain allows importing of FMI standard compliant models, it allows coupling the FDD-algorithm with models developed using general purpose simulation programs like Matlab/Simulink, OpenModelica or Dymola. The toolchain facilitates bridging the gap between the design stage of buildings or HVAC systems and their operation. By means of this connection, it is possible to reuse the knowledge and know-how embedded into the models developed during the design stages for monitoring the performances and detecting faults that potentially affect the energy performance during operation. This toolchain represents a step towards the adoption of model-based FDD strategies in real world applications because. The main advantage introduced by this toolchain is the reuse of models in conjunction with advanced FDD algorithms. Reusing models allows to cut the costs associated with their development as well as their integration with FDD algorithm. The authors believe that all these favorable properties will make FDD algorithms developed using this technology available to a broader audience.

ACKNOWLEDGMENT

This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231. The research was also supported by the U.S. Department of Defense under the ESTCP program. The authors thank Mary Ann Piette, Jessica Granderson, Oren Shetrit, Wangda Zuo, and Rong Lily Hu for the support provided through the project.

REFERENCES

Bonvini, Marco, Michael Wetter, and Michael Sohn. 2014. "An FMI-Based Framework for State and Parameter Estimation." *10th International Modelica Conference, Lund, Sweden*.

Comstock, M. C., J. E. Braun, and E. A. Groll. 2001. "The Sensitivity of Chiller Performance to Common Faults." *HVAC&R Research* 7 (3): 263–279.

Dexter, A., and J. Pakanen. 2001. "Fault detection and diagnosis methods in real buildings." *Energy Conservation in Buildings and Community Systems, IEA Annex 34: Computer-aided evaluation of HVAC system performance*.

Julier, S. J., and J. K. Uhlmann. 1996. "A General Method for Approximating Nonlinear Transformations of Probability Distributions." *Robotics Research Group Technical Report, Department of Engineering Science, University of Oxford*, November, 1–27.

Katipamula, Srinivas, and Michael R. Brambley. 2005a. "Review Article: Methods for Fault Detection, Diagnostics, and Prognostics for Building Systems – A Review, Part I." *HVAC&R Research* 11 (1): 3–25.

Katipamula, Srinivas, and Michael R. Brambley. 2005b. "Review Article: Methods for Fault Detection, Diagnostics, and Prognostics for Building Systems – A Review, Part II." *HVAC&R Research* 11 (2): 169–187.

McKinney, Wes. 2012. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media.

Mills, Evan. 2009. "A Golden Opportunity for Reducing Energy Costs and Greenhouse Gas Emissions." Technical Report, California Energy Commission PIER.

Modelon, AB. 2013, September. PyFMI A package for working with dynamic models compliant with the Functional Mock-Up Interface standard.

Sarkka, S. 2008. "Unscented Rauch–Tung–Striebel Smoother." *Automatic Control, IEEE Transactions on* 53 (3): 845–849.

Venkatasubramanian, Venkat, Raghunathan Rengaswamy, Kewen Yin, and Surya N. Kavuri. 2003a. "A review of process fault detection and diagnosis: Part III: Process history based methods." *Computers & Chemical Engineering* 27 (3): 327 – 346.

Venkatasubramanian, Venkat, Raghunathan Rengaswamy, Kewen Yin, and Surya N. Kavuri. 2003b. "A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies." *Computers & Chemical Engineering* 27 (3): 313 – 326.

Venkatasubramanian, Venkat, Raghunathan Rengaswamy, Kewen Yin, and Surya N. Kavuri. 2003c. "A review of process fault detection and diagnosis: Part I: Quantitative model-based methods." *Computers & Chemical Engineering* 27 (3): 293 – 311.

Wan, E.A., and R. Van der Merwe. 2000. "The unscented Kalman filter for nonlinear estimation." *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. 153–158.

Wang, Shengwei, and Jingtan Cui. 2006. "A robust fault detection and diagnosis strategy for centrifugal chillers." *HVAC&R Research* 12 (3): 407–428.

Wang, Shengwei, and Fu Xiao. 2004. "AHU sensor fault diagnosis using principal component analysis method." *Energy and Buildings* 36 (2): 147–160.

Wetter, Michael, Wangda Zuo, Thierry S Nouidui, and Xiufeng Pang. 2014. "Modelica Buildings library." *Journal of Building Performance Simulation* 4 (7): 253–270.

Yokoyama, Nobuhiro. 2011. "Parameter Estimation of Aircraft Dynamics via Unscented Smoother with Expectation-Maximization Algorithm." *Journal of Guidance, Control, and Dynamics* 34 (2): 426–436.

Zhou, Qiang, Shengwei Wang, and Zhenjun Ma. 2009. "A model-based fault detection and diagnosis strategy for HVAC systems." *International Journal of Energy Research* 33 (10): 903–918.