METHODS AND TOOLS FOR MODEL-BASED VIRTUAL SENSORS APPLIED TO CONDITION MONITORING

Mikel GONZALEZ [1,2], Ekaitz ESTEBAN[1], Oscar SALGADO[1], Jan CROES[2], Bert PLUYMERS[2], Wim DESMET[2]

[1] Control and Monitoring Area, IK4-Ikerlan,
J. M. Arizmendiarrieta 2, 20500 Mondragon, Spain
[2] KU Leuven, Department of Mechanical Engineering – member of Flanders Make,
Celestijnenlaan 300 B, B-3001, Heverlee, Belgium

Key words: Modeling, Virtual sensing, FMI, state estimation

Abstract
Model-based virtual sensing techniques are a valuable approach to estimate system variables which are difficult to measure. Instead of measuring these variables directly, physics-based models and estimation algorithms are used to compute them. As the system grows in complexity the use of dedicated modeling tools is required to reduce modeling effort and errors. However the integration of these tools with the estimation algorithms is not always straightforward. In this paper an overview of the main virtual sensor algorithms and the way to connect them with modeling tools is presented. The Functional Mock-up Interface (FMI) is discussed as the most suitable way to accomplish this. The advantages of using a symbolic modeling language such as Modelica in the implementation of virtual sensors are also discussed. These advantages are highlighted by means of an application example.

1 INTRODUCTION
Maintenance services account for a significant part of the operating cost of high-value systems, especially when possible failures result in large down times [1]. Condition based maintenance strategies provide a cost efficient way to cope with the safety and reliability requirements of these systems [2]. Condition monitoring strategies rely on the measurement of specific variables of the system. In many applications, however, the direct measurement of these variables is either too costly or not possible. Virtual sensing is an attractive option to overcome these difficulties. Virtual sensing is a technique to estimate variables and parameters of interest using available measurements and physics-based models instead of direct physical sensors.

A review of fault and damage detection methods can be found in [3], [4] and [5]. In contrast to other fault and damage detection methods, model-based techniques incorporate physical knowledge of the system, allowing a deeper understanding of the process behavior [6]. Therefore these techniques provide not only crucial information on unobservable quantities but also physical insight in why the system's performance is degrading. The efficacy of these techniques highly depends on the capability of the model to accurately represent the physics of the system and identify uncertain system parameters [1].

This can be a major drawback of these techniques, as the high complexity of modern electromechanical systems results in models which are either too costly or not accurate enough. In addition, cyber-physical systems are composed of several domains such as mechanical, electrical or electronic. The required level of detail of each of these disciplines is dependent on which physical phenomena are most dominant. Often a combination of these tools is required to capture the necessary dynamics. Furthermore, the limited interchangeability of these modeling tools hinders the integration of specialized simulation tools with the estimation algorithms [8].

This paper gives an overview of the most widely used model-based virtual sensing techniques and presents a suitable toolchain for the integration of these techniques with highly complex models. The rest of the paper is organized as follows. In section 2 the state of the art of the most relevant algorithms for model-based virtual sensors is provided and discussed. The viability of implementing these algorithms in an industrial context is studied in section 3. In particular the modeling effort required and the possibility of integrating the simulation tools with other platforms is assessed. In fact the use of the Functional Mockup Interface (FMI) is proposed to address this problem. In section 4 a case study for the application of model-based virtual sensors is provided, where a model developed in an efficient modeling language is combined with estimation algorithms in order to show the potential of the presented toolchain.

2 OVERVIEW OF MODEL-BASED VIRTUAL SENSOR APPROACHES

Model-based techniques combine physics based models and real measurements of the system with data processing algorithms to estimate system variables difficult to measure. At a cost of a higher modeling effort, state and parameter estimators are better at detecting multiple faults and dealing with nonlinearities than other model-based methods such as parity equations [6]. Furthermore, these approaches can be used in other applications such as advanced control techniques or model updating.

Accordingly this paper is focused only in state, parameter and disturbance estimators (i.e virtual sensors). There is a large number of algorithms tailored for specific applications and other algorithms which are too computationally expensive (e.g particle filters, Monte-Carlo simulations [7]) as they are based on exhaustive sampling. For the sake of brevity this comparison is limited to the most widely used algorithms that have potential to cope with industrial systems.

The core of these algorithms consists on using the difference between the real measurements and the prediction of the model to correct this prediction. The most common virtual sensor algorithms are the Luenberger observer (LO), the sliding mode observer (SMO) and the Bayesian estimators. Unlike Bayesian estimators, the LO and the SMO consider deterministic processes (without noise). The LO and SMO algorithms are predominantly used in combination with a linear model as optimality can be proven under such conditions. The notion of optimality however does not stand if the model does not fully represent the physics of the system. Thus both LO and SMO are simpler to implement than Bayesian estimators but under noisy measurement conditions the Bayesian algorithms are proved to perform better ([9],[10]). Therefore the remaining of this section will be focused on Bayesian estimators.

KALMAN FILTER

The Kalman Filter is the optimal linear filter [17]. In the case of Gaussian noise, it provides the maximum a posteriori estimate with the smallest achievable covariance. With non-Gaussian noise, it is optimal in giving the minimal mean square error. It is the most widely used Bayesian estimator and has been successfully used in a number of applications [11]. The KF uses a model defined in state space form as the one shown in equation (1). In the most commonly used derivation of the KF, the process and measurements are assumed to be disturbed by zero mean white Gaussian noise w and v, with covariance **Q** and **R** respectively. The states are assumed to be Gaussian variables with covariance **P**.

$$\begin{aligned} \dot{x} &= Ax + Bu + w \\ y &= Cx + Du + v \end{aligned} \qquad \begin{aligned} &with\ x \sim N(\bar{x}, \boldsymbol{P}) \\ &with\ w \sim N(0, \boldsymbol{Q}) \\ &with\ v \sim N(0, \boldsymbol{R}) \end{aligned} \tag{1}$$

The KF algorithm is shown in Figure 1. In each k-time step the system model is evaluated and compared against measured data. This is done in two steps: prediction and update. In the prediction step an a-priori estimation of the states mean and covariance is obtained from the system´s model. In the update step this a priori estimation is corrected using the system's output. The estimation process is done recursively: all the prior information is summarized in the initial mean and covariance of each step. Therefore the computational effort in each time step is the same regardless the number of measurements.

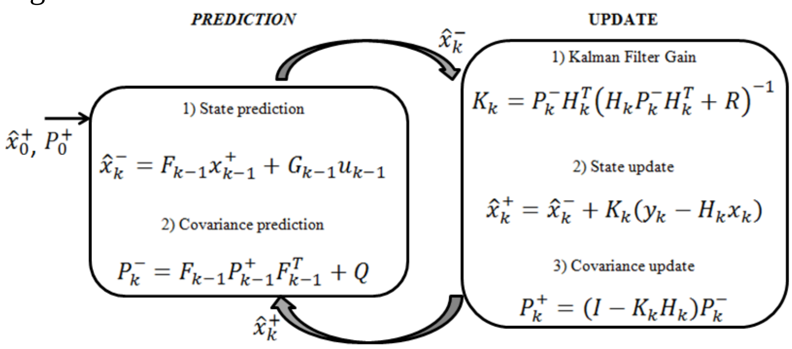

Figure 1: Kalman Filter implementation

Despite being widely used, the KF is limited to linear systems, and the joint estimation of states and parameters is not applicable [7]. Below an overview of the approaches that extend the applicability of the KF to nonlinear systems and systems with constraints is given.

EXTENDED KALMAN FILTER

The Extended Kalman Filter (EKF) is the most widely used extension of the KF for nonlinear systems. The EKF uses a system model as the one in equation (2) and linearizes it around the KF estimate [12]. The standard KF shown in Figure 1 is then applied at this linearized point. The linearization of such a system can be cumbersome and may have a significant influence on the performance. In simple systems analytical Jacobians can be obtained, but this is seldom the case. Normally numerical Jacobians have to be used, which are less accurate, more computationally expensive and in some cases cannot be obtained (e.g discontinuities). This requirement is the main drawback in the implementation of the EKF.

$$\begin{aligned} \dot{x} &= f(x,u) + w \\ y &= h(x,u) + v \end{aligned} \qquad \begin{aligned} &with\ x \sim N(\bar{x}, \boldsymbol{P}) \\ &with\ w \sim N(0, \boldsymbol{Q}) \\ &with\ v \sim N(0, \boldsymbol{R}) \end{aligned} \tag{2}$$

UNSCENTED KALMAN FILTER

The EKF has been shown to be an effective solution in several systems. However for a nonlinear transformation the mean and covariance of a random variable propagate in a different way than the linearized simplification of the EKF [12]. This can lead to inaccuracies if used with highly nonlinear systems. The Unscented Kalman Filter (UKF) addresses this problem by finding a set of individual points (sigma points) in the state space, whose sample probability density function (pdf) approximates the real pdf of the state vector. The pdf is thus approximated by the mean and covariance of the sigma points instead of by a nonlinear transformation. The differences of these approaches are shown in Figure 2 [13].

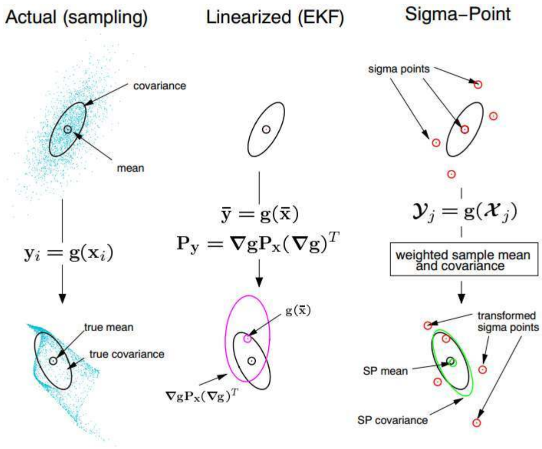

Figure 2: Unscented transformation compared to linear simplification [13].

On the one hand no Jacobians of the system have to be calculated, which simplifies its implementation. On the other hand multiple simulations are required in each step (to calculate the mean and covariance of each sigma point), which also increases the computational cost. The computational cost compared to the EKF, will depend on the linearization strategy in the EKF and the solver efficiency of the UKF.

MOVING HORIZON ESTIMATOR

In contrast to the above defined algorithms the MHE takes into account more than one single measurement at each estimation step. However, not all the measurements available are used as the computational cost would be too expensive. Instead, a sliding window with M-steps to the past that moves one step ahead every k-time step is used. This implies that when a new measurement is available it is included in the data window and the oldest is removed [14]. The estimation consists then in solving an optimization problem to calculate the optimum states and parameters of the system. Thus, the resulting estimation technique can be interpreted as the repeated solving of fixed-size optimization problems that are generated by moving a fixed size observation window over a growing measurement (as shown in Figure 3) [15]. One important feature of the MHE is that it allows for the estimation of systems with constraints and severe nonlinearities. This procedure results in more robust estimations to external disturbances than other Kalman Filter based methods [16].

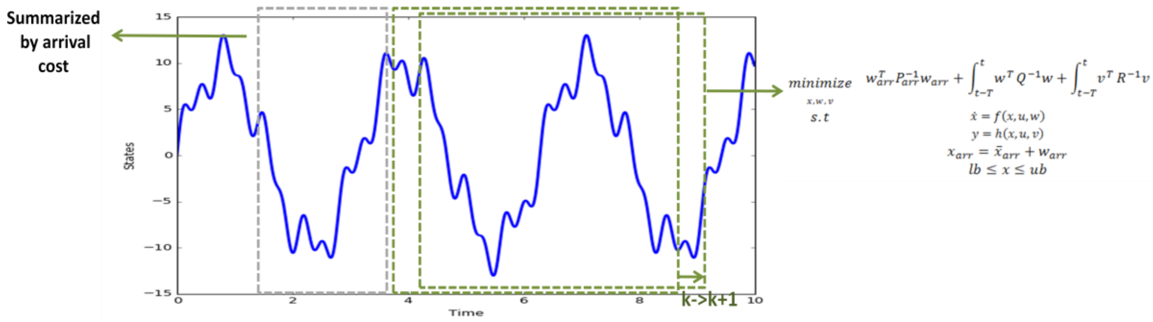

Figure 3: Implementation of the MHE

The MHE has successfully been used in different applications for parameter and state estimation of nonlinear systems both with and without constraints. Depending on the severity of the nonlinearities and the complexity of the constraints, the MHE may have a considerable computational cost, limiting thus the number of real-time processing applications. This drawback, however, can in most cases be solved by efficiently computing the optimization problem. Although other approaches for the constraint handling can be found for single step estimators, the MHE is the best suited estimator for systems with constraints [17].

PARAMETER AND INPUT ESTIMATION

The algorithms explained in the previous section can be extended to estimate not only the states of the system but unknown parameters and inputs too. Based on the state-space system representation, an augmented version of the system can be obtained if the unknown inputs and/or parameters are included in the states vector and their directional derivatives are included in the system matrices. Then a random walk model is used for the unknown input/parameters: they are assumed to remain constant except for an additive noise [18]. Once defined in the proper way, this augmented vector is estimated by means of any of the techniques shown in the previous section. The joint estimation of parameters makes the system nonlinear, which means a nonlinear filter is required. The estimation of inputs will be linear as long as the input matrix (B matrix in equation (1)) is decoupled from the state matrix (A). In this situation a linear filter can be used for the estimation.

	ALGORITHM	**Non Linear**	**Time-varying systems**	**Joint state-parameter estimation**	**Computational cost**	**Robustness against noise**
Deterministic Observers	Lueneberger Observer[1]	No	slow	No	Low	Low
	Sliding Mode Observer	Yes	slow	No	Low	Low
Bayesian Estimators	Kalman Filter	No	Yes	No	Low	Moderate
	Extended Kalman Filter	Slightly	Yes	Yes	Moderate	Moderate
	Unscented Kalman Filter	Yes	Yes	Yes	Moderate	Moderate
Optimization based	MHE	Yes	Yes	Yes	High	High

Table 1: Summary of the capabilities of the virtual sensor algorithms presented in this work

Table 1 summarized the described algorithms. The selection of the appropriate algorithm will depend on the system to be estimated. Each method has its advantages and disadvantages, and can be the optimal one depending on the application. The KF is the most cost efficient way to cope with linear systems regardless of their complexity or time varying nature. For instance, in [10] the KF is used for the monitoring of an elevator (i.e a LTV system) performance by estimation of its acceleration, using only sensors available in the system. The KF can also be augmented as previously explained. In [19] the KF is augmented to estimate both states and input forces in structural dynamics problems. When the system is nonlinear one of the above mentioned extensions has to be used. The computational cost of these algorithms will depend on way each particular problem is solved. In general the UKF

[1] There is an extension for nonlinear systems that also allows for Joint estimation. As explained in section 2 they are more suited for deterministic systems and therefore are not covered in this paper.

performs better than the EKF for highly nonlinear problems [13], [20]. The MHE might have an even higher computational cost, but can handle outer disturbances and model constraints better. In [14] several example applications are shown where the MHE outperforms the EKF.

3 TOOLS FOR VIRTUAL SENSOR INTEGRATION

In the previous sections the importance of a highly reliable model has been highlighted. In addition a great modeling flexibility and simplicity is also required, helping thus avoiding errors and speeding up the process. Therefore the use of advanced modeling tools is desirable in order to achieve cost efficiency and model reliability in the design phase. Thus the heterogeneity of modern cyber-physical systems requires highly flexible modeling tools or exchanging models from different engineering domains (like electrical or mechanical). Unfortunately these simulation tools are often incompatible. Furthermore, modeling environments do not generally allow the user to manipulate the solution at each time step, as required by estimation algorithms. As a consequence it is common to hand-code system models in less efficient languages resulting in overly simplified models or increased modeling complexity (either in modeling the system or in developing an ad hoc interface).

Several modeling environments include model exchange capabilities. However, they are usually developed ad-hoc to interface with one particular tool in a certain context. As an example, Dymola offers model portability to Matlab and Simulink. This feature is however only limited to those tools, and is version dependent. A standard software connectivity method is therefore desirable. One example of the industrial importance of tool connection standards is AUTOSAR, which is a standard for automobile software applications. AUTOSAR is widely accepted in the automobile industry but it does not support common modeling tools [21].

The Functional Mock-up Interface (FMI) is a tool independent standard developed to address this problem. It supports both model exchange and co-simulation of dynamic systems [22]. According to the FMI standard the model is exported (either for exchange or co-simulation) as a so called Functional Mock-up Unit (FMU). The FMU is a compressed file containing an XML-file with the model description, a set of C-functions to execute model equations and further data required by the model (including tables, icons and documentation). Simulation programs or software packages that implement the FMI standard can import FMUs, access their resources, and run simulation [24].

This approach allows dynamic modeling of different software systems to be used together for software-, model- and hardware-in-the-loop (HiL) simulation and for embedded software [25]. Thus it enables the combination of efficient modeling software with custom applications, settling a general approach that reduces errors and speeds up the modeling process. The FMI is a promising candidate to become the industry standard for model exchange [26].

The FMI is currently supported by 84 tools [22]. In particular the Modelica language [23] based tools such as Dymola, JModelica.org or Openmodelica are suitable tools for the combination with estimation algorithms by means of the FMI. Modelica allows for model reuse and counts with several libraries for different engineering domains. This allows for simple model implementation. Additionally the Jacobians (which are required by the EKF and MHE) of the system can be obtained analytically, which is has less computational cost than other linearization techniques such as numerical differentiation.

The combination of Modelica with other programming languages by means of the FMI provides thus a suitable approach for the implementation of model-based virtual sensors. The main focus of the FMI is model exchange for simulation and not for estimation purposes, but it has already been successfully used for estimation [26]-[28]. For instance in [27] and [28] the FMI is used to allow for the implementation of nonlinear observers within Dymola. In [29] an UKF is used to prove the suitability of the method for Fault Detection and Diagnosis.

4 APPLICATION EXAMPLE

In this section the advantages of the presented toolchain are highlighted with the case study presented in [30]. In this example the friction forces in a scaled vertical transportation test rig are estimated by means of an EKF. The measurements used for the estimation are those already available in the installation: car position and velocity, machine encoder position and velocity, and machine quadrature current and voltage. The current example shows an upward travel of the car frame of the used test rig, traveling from 0.7m to 2.5m height.

A multi-physical model including the field oriented control architecture, electrical subsystem and mechanical subsystem are included in this example. A Permanent Magnet Synchronous Machine (PMSM) is used to move the driving sheave. The angular velocity of the PMSM is controlled with a PI controller, and the reactive current in the machine is assumed to be zero. The stiffness of the cabin and counterweight cable sides varies with their length, which makes of this a Linear Parameter Varying (LPV) system. The system has therefore 9 states: Cabin position and velocity, counterweight position and velocity, control reference current, machine current and Voltage and pulley angular position and velocity. In addition, a friction pararameter that takes into account both the friction coefficient and the cabin’s geometry as defined in [30], is identified both for cabin and counterweigth. Given the complexity of such a system the use of the above mentioned toolchain can save modeling effort and errors.

The use of Modelica based software in such multi-physics systems presents great advantages in terms of flexibility and model reusability. It simplifies the modeling phase, reducing the effort and making it less prone to errors. Besides, further configuration changes (such as evolving from simple to detailed models) are simpler than with causal models. These advantages become more relevant as the models grow in size and complexity. In this case example the software OpenModelica [31] is used to model the system, however, any other FMI compliant modeling tool is suitable for the development of the model. The model is exported as FMU as explained in section 3, and thus it can be imported in any other FMI compatible tool.

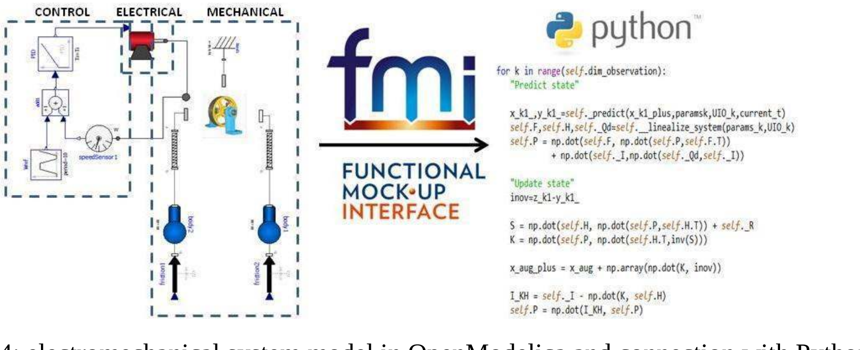

Figure 4: electromechanical system model in OpenModelica and connection with Python

The EKF used for the joint estimation of the states and parameters of the system is developed in Python. Several scientific computing packages are available in Python that aid the development of custom made algorithms and applications. The interconnection of Python with the model has been done using the FMI by means of the package PyFMI [32].

The friction coefficient of cabin and counterweight are identified by augmenting the state vector as explained in section 2, which makes the system nonlinear. The use of the FMI allows for the extraction of analytical state Jacobians and derivatives. The Jacobian for the parameters can be obtained by means of numerical differentiation using the derivatives of the states. This is a great advantage in the implementation of the EKF, as it simplifies the linearization of the system in a simple and time efficient way.

The estimation of the states and the identification of parameters lead to the estimation of other variables of interest. In this particular example the friction forces in the system are derived from the estimated states and the identified friction parameter. Additionally, as explained in section 2 the estimated covariance gives an idea of the uncertainty of the estimation. The estimation of the system's states, parameters and the output of the virtual sensor are shown in Figure 5.

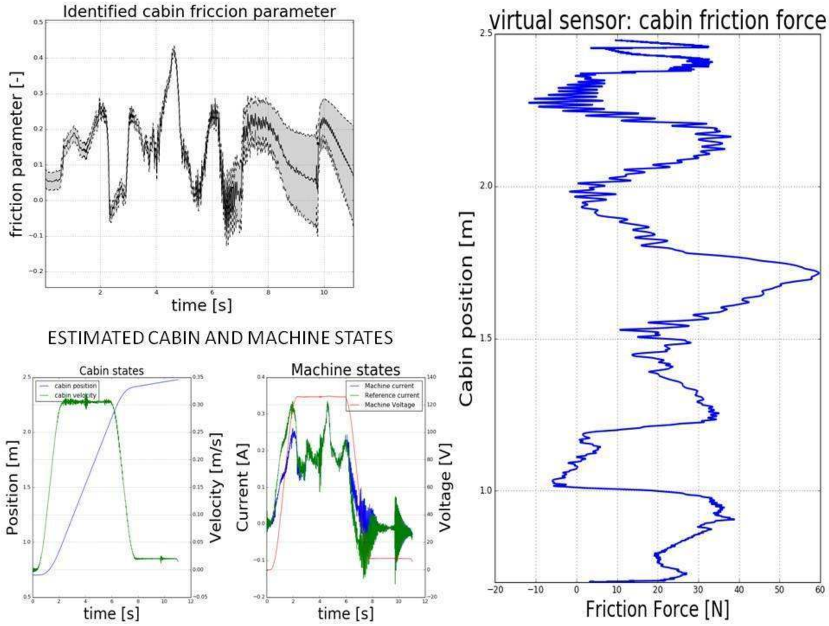

Figure 5: Identified cabin friction parameter, cabin and electrical machine states and estimated friction force

The friction force is following mainly the acceleration profile of the cabin, but the defects of the guiding system are also represented. For instance, the effect of the rail joints can be seen as friction peaks every 0.25m. Therefore this force can be used for the health monitoring of the guiding system. At some points of the simulation it can be seen that the estimated friction parameter (and thus the friction force) go below zero. This can be due to numerical instabilities at these particular positions (e.g local loss of observability) or due to a relatively high uncertainty in positions where the parameter is close to zero. The use of the MHE can be a solution to this issue, as the constraint of a friction parameter strictly higher than zero could be included in the estimation. In the represented test rig this 1.8 long travel takes 11.049 seconds, while the estimation takes circa 5 minutes. The main bottleneck here is that the integrator has to be initialized at each k-step of the filter. Therefore furtherwork is required to reach real-time capabilities using this technique.

This example shows how the presented toolchain can simplify the implementation of model-based virtual sensors. As stated in section 2 the implementation of algorithms such as EKF and MHE is hindered by the requirement of the system's Jacobians. With the presented toolchain however the Jacobian's are extracted analytically, allowing a simple implementation of the presented algorithms. The user can therefore focus only on the modeling part using already available models, facilitating thus model flexibility and reusability. The main added value of this toolchain is therefore helping the end user in the implementation of model-based virtual sensor algorithms.

5 CONCLUSIONS

In this paper an approach to overcome the difficulties associated with the integration of estimation algorithms with high fidelity modeling languages is presented. The use of advanced modeling tools reduces modeling cost and errors, and thus simplifies the implementation of virtual sensing techniques. Namely the computation of the Jacobians of the system is greatly simplified, as it can be obtained analytically. The real time implementation of these toolchain is, however, not yet available, and further work is required on this field. In the future these tools will be applied to the estimation of contact and friction forces in guides of electro-mechanical systems.

6 ACKNOWLEDGEMENTS

The authors gratefully acknowledge the European Commission for its support of the Marie-Sklodowska Curie program through the ITN ANTARES project (GA 606817) and the support from the KU Leuven research fund. The work is also partially supported by Flanders Make.

7 REFERENCES

[1] P. Amorim-Melo, E. Shehab, , L. Kirkwood, P. Baguley "Cost Drivers of Integrated Maintenance in High-value Systems" Procedia CIRP Volume 22, 2014, Pages 152–156

[2] Rosmaini Ahmad, Shahrul Kamaruddin. "An overview of time-based and condition-based maintenance in industrial application" Computers &Industrial Engineering. Vol. 63 Issue 1 135-149 2012

[3] 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.

[4] 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.

[5] 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.

[6] Rolf Isermann. "Model-based fault-detection and diagnosis status and applications". Annual Reviews in control 29 (2005) 71-85

[7] A. T. Nelson, "Nonlinear estimation and modeling of noisy time-series by dual kalman filtering methods". PhD Thesis, Oregon Graduate Institute of Science & Technology, 2000.

[8] Himanshu Neema et al. "Model-Based Integration Platform for FMI Co-Simulation and Heterogeneous Simulations of Cyber-Physical Systems", Proceedings of the 10th International ModelicaConference 235-246, 2014

[9] Yongchang Zhang et al. "A Comparative Study of Luenberger Observer, Sliding Mode

Observer and Extended Kalman Filter for Sensorless Vector Control of Induction Motor Drives" ECCE 2009, pages 2466-2473. IEEE, 2009.
[10] E.Esteban, O. Salgado, A. Iturrospe, and I. Isasa. "Model-based approach for elevator performance estimation. Mechanical Systems and Signal Processing", 68-69:125-137, 2016
[11] Z. Chen. "Bayesian filtering: From Kalman filters to particle filters and beyond" Statistics, 182(1):1-69,2003
[12] Dan Simon. "Optimal State Estimation: Kalman, H Infinity and Nonlinear Approaches. John Wiley & Sons Inc. 2006
[13] Rudolph Van der Merwe , "Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models" PhD thesis Oregon Health & Science University, 2004
[14] Eric L. Haseltine and James B. Rawlings. "Critical Evaluation of Extended Kalman Filtering and Moving Horizon Estimation". Ind. Eng. Chem. Res. 2005, 44, 2451-2460
[15] Christopher V. Rao, "Moving horizon strategies for the constrained monitoring and control of nonlinear discrete-time systems". PhD thesis university of Wisconsin, 2000.
[16] Max Boegli, Tinne De Laet, Joris De Schutter, and Jan Swevers, "Moving Horizon for Friction State and Parameter Estimation" European Control Conference (ECC), 2013.
[17] D. Simon, "Kalman Filtering with State Constraints: A Survey of Linear and Nonlinear Algorithms", IET Control Theory & Applications, vol. 4, no. 8, pp. 1303-1318, 2010
[18] Naets, F., Croes, J., Desmet, W. "An online coupled state/input/parameter estimation approach for structural dynamics" (2015) Computer Methods in Applied Mechanics and Engineering, 283, pp. 1167-1188.
[19] E. Lourens, E.Reynders, et. al. "An augmented Kalman filter for force identification in structural dynamics" Mechanical Systems and Signal Processing Vol 27, 446–460, 2012.
[20] Dan Simon, "A comparison of filtering approaches for aircraft engine health estimation" Aerospace science and technology 12 276-284, 2008
[21] Christian Bertsch et. al. "FMI for physical models on automotive embedded targets", Proceedings of the 11th International Modelica Conference September 21-23, 2015.
[22] https://www.fmi-standard.org/
[23] https://www.modelica.org/
[24] T. Blochwitz et. al., "Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models". Proceedings of the 8th International Modelica Conference, 2011
[25] Patrick Chombart, "Modelisar innovation report"
[26] Christian Bertsch et. al. "The Functional Mockup Interface - seen from an industrial perspective" Proceedings of the 10th International Modelica Conference, 2014.
[27] Jonathan Brembeck,Martin Otter, Dirk Zimmer, "Nonlinear observes based on the Functional Mockup Interface with applications to electric vehicles". Proceedings of the 8th International Modelica Conference, 2011.
[28] Jonathan Brembeck,Martin Otter, Andrreas Pfeiffer et. al. "Nonlinear State Estimation with an Extended FMI 2.0 Cosimulation Inteface" Proceedings of the 10th International Modelica Conference, 2014.
[29] Marco Bonvini, Michael Wetter, Michael D.Sohn "An FMI-based toolchain for the adoption of model-based FDD" ASHRAE Building simulation conference, 2014
[30] Ekaitz Esteban, Oscar Salgado et. al., "Model-Based Estimation of Elevator Rail Friction Forces". Advances in Condition Monitoring of Machinery in Non-Stationary Operations, 2015.
[31] https://www.openmodelica.org/
[32] https://pypi.python.org/pypi/PyFMI