Hybrid Data-Driven Physics-Based Model Framework Implementation: Towards a Secure Cyber-Physical Operation of the Smart Grid

Valeria Vega-Martinez, Austin Cooper, Brandon Vera, Nader Aljohani
[*]*Electrical & Computer Engineering*
University of Florida
Gainesville, Florida, United States
valeriavega@ufl.edu, austin.cooper@ufl.edu, bvera1@ufl.edu, eng89nader@ufl.edu

Arturo Bretas[*][†]
[†]*Distributed Systems Group*
Pacific Northwest National Laboratory
Richland, Washington, United States
arturo.bretas@pnnl.gov

***Abstract*—False data injection cyber-attack detection models on smart grid operation have been much explored recently, considering analytical physics-based and data-driven solutions. Recently, a hybrid data-driven physics-based model framework for monitoring the smart grid is developed. However, the framework has not been implemented in real-time environment yet. In this paper, the framework of the hybrid model is developed within a real-time simulation environment. OPAL-RT real-time simulator is used to enable Hardware-in-the-Loop testing of the framework. IEEE 9-bus system is considered as a testing grid for gaining insight. The process of building the framework and the challenges faced during development are presented. The performance of the framework is investigated under various false data injection attacks.**

***Index Terms*—false data injection attack, machine learning, state estimation, OPAL-RT, real-time simulation**

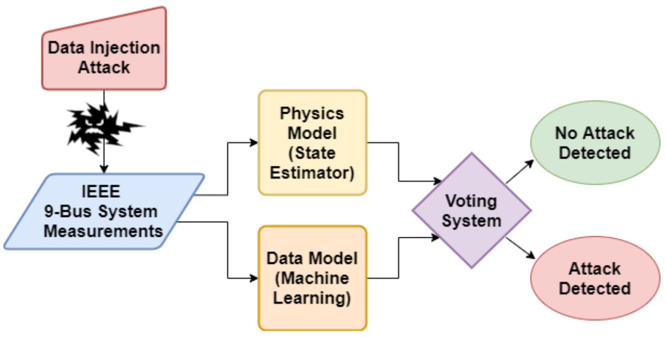
Data Injection Attack
IEEE 9-Bus System Measurements
Physics Model (State Estimator)
Data Model (Machine Learning)
Voting System
No Attack Detected
Attack Detected

Fig. 1. Overview of Hybrid Framework

I. Introduction

Smart Grid (SG) has become the epitome of the next-generation power grid in part to its positive environmental impacts and desirable operational costs. The decentralized integration of controls and communications allow SG to have self-healing capabilities, easier integration of renewable energy generation, and lower operational costs. Consequently, increased dependency on communication networks exposes SG infrastructure to cyber-physical security threats, such as false data injection (FDI) attacks. Currently, real-time monitoring of power systems is done through State Estimation (SE). SE has error processing capabilities; however, it is limited to steady-state assumptions and lacks compatibility to the rapidly evolving demands of cyber-physical security in smart grids. In contrast, machine learning solutions consider temporal and spatial information to validate data and learn the normal state of a properly functioning grid to detect anomalies introduced in field measurements. Together, physics-based and data-driven methods can complement one another to provide a promising outlook for cyber-physical security in smart grids.

In recent work of the Machine Learning (ML) realm, the bad data analysis within SE has been enhanced [1]–[3]. The aim of these works is to detect FDI attacks through the use of the Ensemble CorrDet with Adaptive Statistics (ECD-AS) algorithm. Based on past measurements, this algorithm analyzes the current real-time measurement values in order to detect false data. The feature of temporal information is an add-on to the physics-based SE.

While the aforementioned enhanced solution has not been integrated yet, in this paper, we provide a methodology to integrate such solution in a Hardware-in-the-loop environment for detecting FDI attack embedded with an adaptive voting system towards data fusion. First, we build the power grid to represent the physical model in Simulink. Next, we developed the Machine Learning and State Estimation model in Simulink. The simulink model which represents the hybrid solution is configured in RT-LAB format to be executed in real-time. Therefore, the contributions of this work are threefold:

1) Building Simulink model of hybrid solution for FDI detection;
2) Configuring the hybrid solution to be executed in RT-LAB simulator;
3) Implementing an adaptive voting system for data fusion.

The remainder of the paper is organised as follows. In Section II, background information of the framework's major components, i.e., State Estimation, Machine Learning and OPAL-RT, is presented. Section III presents the implementation of the framework within RT-LAB environment. Case studies used to evaluate the method's performance are shown in Section IV. Finally, Section V presents the conclusions of this work as well as future remarks.

II. Background

A. State Estimation

The primary goal of the physics-based SE process is to estimate the power system state variables, namely the complex voltages at each bus. This is achieved through a collection of measurements taken throughout the system. Telemetry errors must also be taken into consideration. Thus, the measurement model can be generalized as:

$$z = h(x) + e \tag{1}$$

where x is the vector of estimated state variables of size ($N \times 1$). For a system of n buses and $(n-1)$ bus voltage magnitudes and angles, $N = 2n-1$ where N is the number of states. $h(.)$ is of size $(m-1)$, where m is the number of measurements, and relates the measured quantities and state variables as a vector of linear and/or nonlinear functions [4].

In the weighted-least squares (WLS) SE model, the goal is to estimate the best state vector x which minimizes the $J(x)$ index:

$$J(x) = [z - h(x)]^T R^{-1} [z - h(x)] \tag{2}$$

where R^{-1} is the inverse of the measurement variance errors, referred to as the weight matrix.

Although the WLS method is considered the most computationally efficient method for the overdetermined set of measurements [4], improper estimates can be made if such measurements are taken near leverage points, where highly influential measurements "attract" the SE solution [5]. Such conditions can yield erroneous low errors where, in reality, a large component of the error may be undetectable using traditional Gross Error (GE) detection methods. To better detect and identify GEs at these leverage points, the Innovation Index (II) method and Largest Normalized Error Test are used to incorporate the undetectable error component of the measurement residuals into the GE detection process.

The error vector e can be interpreted as a sum of two components [4]:

$$e = e_U + e_D \tag{3}$$

where e_D and e_U are detectable and undetectable parts of the error e respectively, and K is the projection matrix which is defined as [4]:

$$K = H(H^T R^{-1} H)^{-1} H^T R^{-1} \tag{4}$$

where H is the Jacobian matrix of the first derivatives of the nonlinear functions of $h(.)$, and R is the diagonal matrix whose elements are the measurement variance errors. The measurement residuals r can then be defined as:

$$r = (I - K)e = e_D \tag{5}$$

For the purposes of considering the undetectable error component for GE processing, the II is defined as the ratio between the detectable and undetectable error components, and is defined for the i^{th} measurement as:

$$II_i = \frac{||e_D^i||_{R^{-1}}}{||e_U^i||_{R^{-1}}} = \frac{\sqrt{(e_D^i)^T R^{-1} (e_D^i)}}{\sqrt{(e_U^i)^T R^{-1} (e_U^i)}} = \frac{\sqrt{I - K_{ii}}}{\sqrt{K_{ii}}} \tag{6}$$

where K_{ii} is the i^{th} main diagonal element of matrix K. By knowing the residual of the estimated measurements and that the error components are orthogonal, total error for a measurement i can be defined as the Composed Measurement Error (CME) [4]:

$$CME_i^2 = \left(1 + \frac{1}{II_i^2}\right) r_i^2 \Rightarrow CME_i = r_i \sqrt{1 + \frac{1}{II_i^2}} \tag{7}$$

where r_i is the residual of the i^{th} measurement and II_i is its Innovation Index. This equation can be used to obtain the composed error of measurement in its normalized form CME^N [4], which will be used for GE detection as:

$$CME_i^N = \frac{r_i}{\sigma_i} \sqrt{1 + \frac{1}{II_i^2}} \tag{8}$$

The performance index for GE detection can now be related to the CME. This new index, $J_c(x)$, will also have a chi-square distribution:

$$J_c(x) = \sum_{i=1}^{m} \left(\frac{CME_i}{\sigma_i}\right)^2 \tag{9}$$

Thus, the SE algorithm is as follows:

1) Perform the state estimation and find the residual vector.
2) Find the II of each measurement and obtain the undetectable error component.
3) Calculate the composed error of all measurements.
4) Perform GE identification using the CME^N.

B. Machine Learning

A data-driven machine learning algorithm is implemented to consider both temporal and spatial data. Ensemble CorrDet with Adaptive Statistics (ECD-AS) combines concepts from the work done with the CorrDet algorithm and the Ensemble CorrDet algorithm [1]–[3]. CorrDet is implemented at the bus level to make use of the distributed architecture. Namely, each bus is considered as a local, spatial region corresponding to one local CorrDet detector, ϕ_m. Further exploration of CorrDet led to ECD-AS in which adaptive statistics from local CorrDet environments are collected, processed, and compared against each other to determine data sample classification (i.e., normal or abnormal).

Algorithm 1 summarizes the training and testing phases of ECD-AS. For each ϕ_m, the training process consists of computing the mean (μ_m) and sample covariance (Σ_m) of all corresponding measurements considered to be normal. Subsequently, the squared Mahalanobis distance ($\delta_{Z,m}$) for each sample in the training set is computed using (10). A threshold (τ_m) is then initialized for every local CorrDet using (11) where $\mu_{thr,m}$ and $\sigma_{thr,m}$ are the mean and standard deviation of the squared Mahalanobis distance values of all normal training samples, respectively, and η is a hyper-parameter value determined through experimentation.

$$\delta_{Z_m} = (z_m - \mu_m)\Sigma_m^{-1}(z_m - \mu_m)^T \tag{10}$$

$$\tau_m = \mu_m + \eta \times \sigma_m \tag{11}$$

Algorithm 1 ECD-AS

```
1: Input: $Z_{train}$, $Z_k$
2: Training Phase:
3: Initialize sliding window (B) of length $\beta$.
4: for Every local CorrDet detector $m = 1 : M$ do
5:     Compute $\mu_m$ and $\Sigma_m$ of samples in $Z_{train}$.
6:     Compute $\delta_{Z,m}$ using (10).
7:     Include $\delta_{Z,m}$ into B.    ▷ Insert the last $\beta$ training samples into B.
8:     Compute the threshold ($\tau_m$) using (11).
9: end for
10: Testing Phase:
11: for Every incoming sample, $Z_k$ do
12:     Compute $\delta_{Z,k}$ with respect to all local classifiers.
13:     if Any $\delta_{Z,k} < \tau_m$ then
14:         Classify sample as normal. ($Y_k$ = 0)
15:         Update $\mu_m$ and $\Sigma_m^{-1}$ using (12) and (13).
16:         Update B by adding $\delta_{Z,k}$ of most recent sample and removing oldest value.
17:         Update $\tau_m$ using (11) with $\mu_{m,\beta}$ and $\sigma_{m,\beta}$ from B.
18:     else
19:         Classify sample as abnormal. ($Y_k$ = 1)
20:     end if
21: end for
22: Output: Y
```

As the program accepts new incoming data samples, statistics are adapted to account for varying trends in data because system conditions evolve as load and generation fluctuate throughout time. The squared Mahalanobis distance of each incoming sample with respect to all ϕ_m is computed using (10). If at least one δ_{Z_m} is greater than its corresponding τ_m, the sample is classified as abnormal. On the other hand, if the sample is classified as normal, then τ_m, μ_m, and Σ_m^{-1} are updated using (11), (12), and (13), respectively. The program is then ready to accept another new sample of data.

$$\mu_{m,new} = (1-\alpha)\mu_m + \alpha(z_m - \mu_m) \tag{12}$$

$$\Sigma_{m,new}^{-1} = \frac{1}{1-\alpha}\left[\Sigma_m^{-1} - \frac{(z_m-\mu_m)^T(z_m-\mu_m)}{\frac{1-\alpha}{\alpha} + (z_m-\mu_m)(z_m-\mu_m)^T}\right] \tag{13}$$

Adaptation to varying system conditions is achieved through careful selection of sliding window size and hyper-parameters responsible for adjusting the updated statistics. The hyper-parameters (η, α, and β) enable the model to initialize statistics during training and update the statistics after a sample is classified as normal. η is a parameter determining how far the classification threshold should be away from the mean of the normal samples, β defines the size of the sliding window, and α determines how much value is given to the new sample data as opposed to the previous mean. The range of values for the hyper-parameters are the following: η can range from 4 to 18, β can range from 0 to 450, and α can range from 0 to 10^{-4}. Multiple combinations of the these hyper-parameters were applied during the testing phase to achieve optimal performance given a load profile. The following hyper-parameter values were selected: η = 6, β = 65 and α = 0.001.

C. OPAL-RT

RT-LAB is a software fully integrated with the Simulink/ MATLAB environments. It is used in many industries, including power systems, power electronics, aerospace, and automotive [6]. Simulink models can interact in real-time with the real world through RT-LAB by interfacing with the OPAL-RT hardware, a component which enables HIL testing. It is capable of running vast Simulink models and I/O capability in parallel, allowing the load to be shared across multiple CPUs [6]. Working with RT-LAB is broken down as follows:

1) Import and edit the Simulink model through RT-LAB.
2) Compile and load the model into a real-time application.
3) Execute on multiple cores within OPAL-RT hardware.

The upper level structure for producing a model consists of two blocks: computation and console. In the console block, the GUI and outputs are configured. OPAL-RT expects only one master subsystem which must be labeled with the prefix ”SM_”. In contrast, there can be multiple slave subsystems within the design as long as they are labeled with the prefix ”SS_”. The console subsystem is optional; however, if included, it must be labeled with the prefix ”SC_”. Lastly, if the subsystems within the model expect inputs, the inputs must be passed with a Simulink block known as ”OpComm”. Once the model is structured correctly, it is imported into RT-Lab and compiled [7].

III. Simulation Approach

A. Modeling

The parameters of the transmission lines for the IEEE 9-bus system were computed from the R, X, and B values of MATPOWER [8] branch data of its 9-bus case. The buses were added by creating nine subsystems, each containing several ”3-Phase V-I Measurement” blocks from the Simscape library in Simulink. The loads of the original system were modified such that ”3-Phase Dynamic Load” blocks were included to simulate changing load conditions. The real and reactive load values are controlled externally to follow a user defined load profile. Lastly, measurement blocks within the main subsystem gather and transmit selected measurements from the buses to the SE and ECD-AS blocks.

The main integration challenge was attributed to the configuration of the dynamic loads and generators to consider varying load conditions corresponding to a load profile. The ”3-Phase Dynamic Load” block in Simulink can implement a three-phase load by varying P and Q through external control. According to MathWorks, the ”3-Phase Dynamic Load” is represented by current sources; therefore, it cannot be connected to an inductive network in series [9]. A small resistive load of approximately 5 MW is connected in parallel to mitigate the limitation of the equivalent model. On a similar note, the three-phase equivalent sources were replaced with synchronous machines as the generators of the model to enable interaction with dynamic loads.

B. Detection Methods

1) State Estimator: The SE MATLAB program includes functions for generating the state variable estimates and identifying GEs. The initial WLS SE function obtains the residuals, projection matrix K, and measurement standard deviations. The *II* function then obtains the initial innovation indices,

$J_c(x)$ index, and composed measurement errors. From the composed measurement errors are derived the CME^N for GE identification. $J_c(x)$ is then compared to the chi-square critical value, which if exceeded indicates GE detection. The CME^N then identifies the index of the GE.

MATPOWER was used to obtain the branch data for the IEEE 9-bus system. Next, MATPOWER power flow was run to obtain the real/reactive power injections at buses 1, 2, and 3 as well as the real/reactive power flows between each bus for a 9-bus case. MATPOWER also supplied the complex voltages at each bus for that case, providing valuable reference for the physics-based model. With the branch and power flow data, the black-box SE model was created.

Integration challenges including converting code from MATLAB 2021 to Simulink 2011. The RT-LAB version made available to build this model is exclusively compatible with MATLAB/Simulink 2011. Migrating and converting functions involved careful initialization of variables and matrices used in the SE, at the expense of some modularity.

2) Ensemble CorrDet with Adaptive Statistics: The vision for implementing ECD-AS in a Simulink environment was inspired by the definitions of ECD detector and local CorrDet detector. The ECD detector is defined as a set of local CorrDet detectors of data samples with a few, spatial neighboring measurements, compared to full measurements in the CorrDet detector. On the other hand, a local, spatial region, such as a power system bus, is considered to be a local CorrDet detector.

Within Simulink the ECD detector is represented by a MATLAB function block responsible for prompting individual local CorrDets to process data and update statistics. Similarly, each local CorrDet is represented by a single user-defined function containing statistics unique to it. Each local classifier retains the values of μ_m, Σ_m^{-1}, and B with persistent variables. When prompted by the ECD detector block, each local CorrDet function computes and reports squared Mahalanobis distance values and threshold values to the ECD detector block. Congruent with Algorithm 1, the ECD detector block then compares the set of squared Mahalanobis distances obtained from all local CorrDet detectors with their corresponding threshold values and classifies the sample accordingly. Once the classification is determined, the ECD detector prompts the local CorrDet detectors to process new samples.

3) Adaptive Voting System: The results from both detection methods are combined with a voting system to reap the desirable attributes of each method [1]. Previous implementations of a voting system (VS-Fixed) only consider a fixed threshold and were constrained to constant system conditions. Since constant system conditions are not realistic in the realm of power systems, an adaptive voting system (VS-Adaptive) is presented to account for varying system conditions and decision scores as the program accepts samples. VS-Adaptive is represented by a MATLAB function block which accepts outputs from SE and ECD-AS. Algorithm 2 summarizes the implementation of the voting system.

$$\Psi_{fusion} = \Psi_{ECD_{normalized}} + \Psi_{SE_{normalized}} \tag{14}$$

The overall decision score from ECD-AS ($\Psi_{ECD,k}$) is considered to be minimum $\delta_{Z,k}$ when ECD-AS classifies a sample as normal; however, the $\Psi_{ECD,k}$ is $\delta_{Z,k}$ corresponding to the local CorrDet detector where an abnormal sample was found. In the event there are multiple local CorrDet detectors that detect an abnormal sample, $\Psi_{ECD,k}$ is the maximum $\delta_{Z,k}$ among them. With regard to SE, the overall decision score from SE ($\Psi_{SE,k}$) is considered to be $J_c(x)$ of every sample computed using (9). A fusion decision (Ψ_{fusion}) is computed using (14) and compared to a threshold (τ_{fusion}) similar to (11). **B** is updated to prevent bias by ensuring equal consideration of error detection between the two models.

Algorithm 2 Adaptive Voting System

```
1: Input: $Z_{train}$, $Z_k$, $\Psi_{ECD}$, $\Psi_{SE}$, $Decision_{SE}$, $Decision_{ECD}$
2: Training Phase:
3: Initialize sliding windows (B) of length $\beta$ for both $\Psi_{ECD,k}$ and $\Psi_{SE,k}$.
4: Insert the most recent $\beta$ minimum $\delta_{Z,m}$ from training phase into $B_{\Psi_{ECD}}$.
5: Insert the most recent $\beta$ $J_c(x)$ indices from SE into $B_{\Psi_{SE}}$.
6: Testing Phase:
7: Normalize $B_{\Psi_{ECD}}$ and $B_{\Psi_{SE}}$.
8: Initialize sliding window ($B_{\Psi_{fusion}}$) of length $\beta$ using (14).
9: for Every incoming sample, $Z_k$ do
10:     Normalize $\Psi_{ECD}$ and $\Psi_{SE}$ with their respective B.
11:     Compute $\Psi_{fusion}$ and $\tau_{\Psi_{fusion}}$ using (14) and (11) respectively.
12:     if $\Psi_{fusion} < \tau_{\Psi_{fusion}}$ then
13:         Classify sample as normal. ($\Psi_{Decision} = 0$)
14:         Update $B_{\Psi_{fusion}}$ by adding recent value and removing oldest.
15:         if $Decision_{SE}$ = "normal" then
16:             Update $B_{\Psi_{SE}}$.
17:         end if
18:         if $Decision_{ECD}$ = "normal" then
19:             Update $B_{\Psi_{ECD}}$.
20:         end if
21:     else
22:         Classify sample as abnormal. ($\Psi_{Decision} = 1$)
23:     end if
24: end for
25: Output: $\Psi_{Decision}$
```

IV. Case Study

The Simulink model was imported and executed using RT-LAB to generate and collect a total of 21,600 samples with Gaussian noise. The number of samples considered represents 24 hours of real-time operations given SCADA obtains measurements every four seconds. The measurement set of the samples includes real and reactive power flows, real and reactive power injections, and voltage magnitudes, of which 43 measurements were selected for the SE and ECD-AS. For the purposes of ECD-AS, the first 1,800 samples (around 8%) were used for training while the remaining 19,800 samples were used to validate performance.

A series of FDI attacks were introduced into the model via a MATLAB function block. Out of the 19,800 testing samples, 1080 samples (around 5%) were selected at random. For this work, ten different FDI scenarios were developed in which a single measurement was intercepted by introducing a gross error between 7% and 23% of measurement's magnitude. In this particular case, one standard deviation is defined as one percent of the ideal measurement value (i.e., a measurement without Gaussian noise). The duration of each FDI attack varied from 3 to 15 samples to simulate a sustained attack by an adversary.

FDI detection is treated as a classification problem in which expected and unexpected measurements are discerned. Confusion matrices are created by comparing the output of the detection methods to the known classification of data. Each sample from the testing phase is classified as either True Negative (TN), True Positive (TP), False Negative (FN), or False Positive (FP). TN refers to a normal sample predicted as normal while FN refers to a normal sample predicated as abnormal. Similarly, TP refers to an abnormal sample predicted as abnormal while FP refers to an normal sample predicted as abnormal. Once samples are labeled/classified, metrics such as Accuracy, Precision, Recall and F1-score can be calculated [2]. Matthews correlation coefficient (MCC) is used as an evaluation metric which is described using (15) as:

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{15}$$

The MCC coefficient, also known as phi coefficient (r_ϕ), is chosen as an evaluation metric because it produces a high score only if the prediction of the classification model yielded favorable results in all confusion matrix categories [10]. MCC varies from -1 to +1 where +1 indicates a perfect prediction, 0 represents random prediction, and -1 indicates observations in disagreement with predictions. Fundamentally, MCC is defined as the correlation coefficient between predicted values and true values (15), and can be an ideal metric when there is a predefined threshold, as is the case with SE and VS-Fixed. The performance of each detection method with respect to MCC is summarized in Fig. 2.

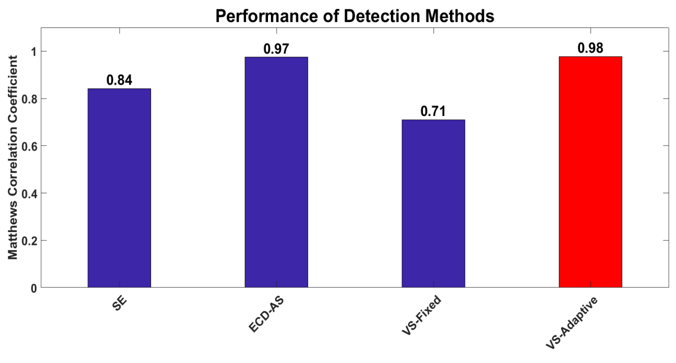

Fig. 2. Performance considering Matthews Correlation Coefficient.

The performance of the detection methods is evaluated using conventional classification performance metrics (i.e., precision, accuracy, recall, and F1-score). TABLE I presents the overall performance of each method across ten scenarios.

VS-fixed is not an appropriate method when considering changing system conditions given its low MCC and poor F1-score relative to other methods. In contrast, VS-Adaptive outperforms both SE and ECD-AS by yielding the highest MCC and F1-score. The hybrid solution in the form of VS-Adaptive provides a promising outlook for cyber-security by showcasing the ability to handle varying system conditions, evaluating results from physics-based and data-driven solutions, and enhancing their desirable features.

TABLE I
COMPARISON OF DETECTION METHODS

Method	Accuracy	Precision	Recall	F1-Score
	$\mu_{DM} \pm \sigma_{DM}$	$\mu_{DM} \pm \sigma_{DM}$	$\mu_{DM} \pm \sigma_{DM}$	$\mu_{DM} \pm \sigma_{DM}$
SE	98.60±1.73	100±0.00	72.03±34.58	78.23±30.62
ECD-AS	99.76±0.29	99.41±0.14	95.75±5.74	97.47±3.11
VS-Fixed	97.59±0.80	99.84±0.21	51.88±16.05	66.99±13.67
VS-Adaptive	**99.79±0.28**	**99.79±0.14**	**96.00±5.52**	**97.78±2.98**

V. CONCLUSION

In this paper, an implementation of a hybrid solution toward false data injection attack in smart grid is presented. Integrating an enhanced solution in a Hardware-in-the-loop environment for detecting FDI attacks provides a promising outlook on the future of the cyber-physical security of smart grids. The feature of the framework is to fuse data from Machine Learning as well as State Estimation for providing an overall system that takes advantage of both models. In addition, the fused model updates the detection threshold since the statistics of the data change over time. Simulation results show that the voting system with adaptive statics outperformed individual systems. The framework is implemented in Simulink with OPAL-RT for real time simulation.

ACKNOWLEDGMENT

This material is based upon work partially supported by the National Science Foundation under Grant No. 1809739.

REFERENCES

[1] C. Ruben, S. Dhulipala, K. Nagaraj, S. Zou, A. Starke, A. Bretas, A. Zare, and J. McNair, "Hybrid data-driven physics model-based framework for enhanced cyber-physical smart grid security," *IET Smart Grid*, vol. 3, no. 4, pp. 445–453, 2020.

[2] K. Nagaraj, S. Zou, C. Ruben, S. Dhulipala, A. Starke, A. Bretas, A. Zare, and J. McNair, "Ensemble corrdet with adaptive statistics for bad data detection," *IET Smart Grid*, vol. 3, no. 5, pp. 572–580, 2020.

[3] K. Nagaraj, N. Aljohani, S. Zou, C. Ruben, A. Bretas, A. Zare, and J. McNair, "State estimator and machine learning analysis of residual differences to detect and identify fdi and parameter errors in smart grids," in *2020 52nd North American Power Symposium (NAPS)*. IEEE, 2021, pp. 1–6.

[4] A. Bretas, N. Bretas, J. B. London Jr, B. Carvalho *et al.*, *Cyber-Physical Power Systems State Estimation*. Elsevier, 2021.

[5] N. G. Bretas, S. A. Piereti, A. S. Bretas, and A. C. Martins, "A geometrical view for multiple gross errors detection, identification, and correction in power system state estimation," *IEEE Transactions on Power Systems*, vol. 28, no. 3, pp. 2128–2135, 2013.

[6] "Software simulation — real time applications — rt labs," https://www.opal-rt.com/software-rt-lab-2-2/, November 2021.

[7] S. Mikkili and A. K. Panda, "Review of rt-lab and steps involved for implementation of a simulink model from matlab to real-time," *International Journal of Emerging Electric Power Systems*, vol. 14, no. 6, pp. 641–658, 2013.

[8] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12–19, 2010.

[9] MathWorks, "Three-phase dynamic load," *MathWorks* Help Center. [Online]. Available: https://www.mathworks.com/help/physmod/sps/powersys/ref/threephasedynamicload.html

[10] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, 2020.