Board 70: Development and Implementation of a Non-Intrusive Load Monitoring Algorithm

Dr. Robert J Kerestes, University of Pittsburgh

Robert Kerestes, PhD, is an assistant professor of electrical and computer engineering at the University of Pittsburgh's Swanson School of Engineering. Robert was born in Pittsburgh, Pennsylvania. He got his B.S. (2010), his M.S (2012). and his PhD (2014) from the University of Pittsburgh, all with a concentration in electric power systems. Robert's academic focus is in education as it applies to engineering at the collegiate level. His areas of interest are in electric power systems, in particular, electric machinery and electromagnetics. Robert has worked as a mathematical modeler for Emerson Process Management, working on electric power applications for Emerson's Ovation Embedded Simulator. Robert also served in the United States Navy as an interior communications electrician from 1998-2002 on active duty and from 2002-2006 in the US Naval Reserves.

Mr. Dekwuan Stokes, University of Pittsburgh

Dekwuan is a senior electrical engineering major at University of Pittsburgh. He plans to enroll in the PhD program with a focus in power, as well as, achieve his MBA throughout the process. His career choice and long term goal is to become a professor and to start his own businesses. Outside of school, Dekwuan likes to play basketball, video games and enjoy time with his family and friends.

Ryan M Brody, University of Pittsburgh

Ryan Brody graduated in April 2018 from the University of Pittsburgh with a bachelor's degree in electrical engineering with a concentration in electric power systems and a minor in computer science. He has since started a master's degree at the University of Pittsburgh studying electrical engineering and electric power systems. He is interested in researching power electronic converters and battery management systems for electric vehicle fast charging and distributed energy resources in smart grids. He is also interested in engineering education, aspiring to be a professor when he is older.

Adam Emes, University of Pittsburgh

Adam Emes completed his B.S. in electrical engineering, with a concentration in electric power engineering, from the University of Pittsburgh in 2018. In his time as an undergraduate, he completed three co-op rotations at Curtiss-Wright EMD, and worked part time as an undergraduate student researcher. From his co-op position, he gained experience with electric motor and generator design. In his undergraduate research, he contributed to projects that utilized signal processing in fault classification and load detection applications. He is currently a second year M.S. student in the electric power systems group at Pitt. His research interests include power converter stability analysis for renewable energy systems.

Mr. Alexander Williams, University of Pittsburgh

Undergraduate Electrical Engineering Student at the Swanson School of Engineering

LOAD DETECTION ALGORITHM FOR SMART GRID APPLICATIONS

Authors: Ryan Brody, Adam Emes, Dekwuan Stokes, Alexander Williams, Robert Kerestes
Department of Electrical and Computer Engineering, University of Pittsburgh
Pittsburgh, PA, USA

Abstract – **This research paper focuses on a non-intrusive load monitoring (NILM) algorithm developed for a senior-design project. The goal of this research is two-fold: to produce a working load detection algorithm for purely resistive, inductive, or capacitive loads, and to use this experience as the basis for creating a new laboratory assignment for undergraduate students. The NILM algorithm was developed in MATLAB and tested with both simulated data and real data collected in the Electric Systems Power Lab (EPSL) at the University of Pittsburgh. Using voltage and current sensors and testing its functionality such that students can replicate the results of this paper as a laboratory assignment. From a pedagogical perspective, this project combines knowledge of power systems, signal processing, and coding – providing students with a relevant assignment that relates to modern day challenges in smart grid and smart home technology.**

I. BACKGROUND

The goal of NILM [1] algorithms is to identify when a load in a residence turns on or off (henceforth called an "event") only by measuring the power consumed by the residence as a whole. In the case of many commercial products, such as Sense [2] and Neurio [3], this is achieved by first measuring the voltage and current through the 220 V line of an average residential fuse box using a spare 220 V breaker to measure the voltage and a clamp-type current sensor. This data then is processed using event detection [4] and artificial intelligence [5] algorithm to determine what loads are consuming power and when they are.

Rather than developing a truly new NILM algorithm capable of performing like the Sense or Neurio, this paper details a proof-of-concept NILM algorithm, how it was prototyped in MATLAB, and how it was tested. It is not practical, based on our experience and available resources, to compete with one of these companies, and therefore, we've approached the subject from an educational perspective. Also, this simplified approach makes it possible to incorporate this project into a laboratory assignment for the future Power Quality class at the University of Pittsburgh.

Experimentation occurred in the EPSL [6], constructed in 2014 [7], which offers undergraduate students at the University of Pittsburgh a space to experiment with the fundamentals of electric power engineering in a safe way. The lab is equipped with six lab benches, each with single phase and three phase resistors, inductors and capacitors, intended to model the loading conditions of a typical household. To measure important quantities and export waveforms to a computer, each bench has a Dranetz PowerVisa Power Quality Meter that can perform important calculations, including event detection.

Accordingly, the scope of this paper is restricted by the limitations imposed by the lab equipment. First, only a 120 V 3-phase system, as opposed to a 220 V system, was studied because of ease of access and safety concerns with students manipulating the lab bench's fuse box. Additionally, the possible loading conditions were limited to no load, shunt resistance, shunt capacitance, and shunt resistance because those components are readily available in each lab bench. Finally, due to limitations of the measurement equipment available, the real-time aspects of NILM algorithms were ignored, and instead the focus was identifying load types when events occur at known times. In the future, each of these limitations will be addressed and accounted for.

Motivation for our work is twofold. First, this paper serves as documentation of an academic exploration of load detection algorithms. As credit for a design elective, the research focused on using the tools available to create a proof-of-concept NILM to determine what types of loads are consuming power in a residence in real-time.

Second, the ultimate goal of the project is to develop a laboratory procedure that can be integrated into a new smart grid course at the University of Pittsburgh. The lab is intended to familiarize students with concepts related to coding and signal processing, while encouraging students to consider how the smart gird paradigm can fundamentally change the way utilities and consumers interact with the grid. A basic procedure will be outlined for students to follow, which will mimic at least part of the design process for the load-detection algorithm developed in this report.

II. ALGORITHM DETAILS

Considering the simplifications detailed in the previous paragraph, the flow chart in Figure 1 shows an abstraction of the NILM algorithm. First, an event is detected and the phase current and voltage waveforms are measured several cycles before and after the event. Then, two cycles of current data are sampled based on the location the positively-sloped zeros. Doing so creates a uniform initial frame of reference – the voltage is zero and increasing – for the classification algorithm. Then, the algorithm groups the pre- and post-transient waveforms based on each events similarity the groups of training data. The following subsections will explain each of these steps in more detail.

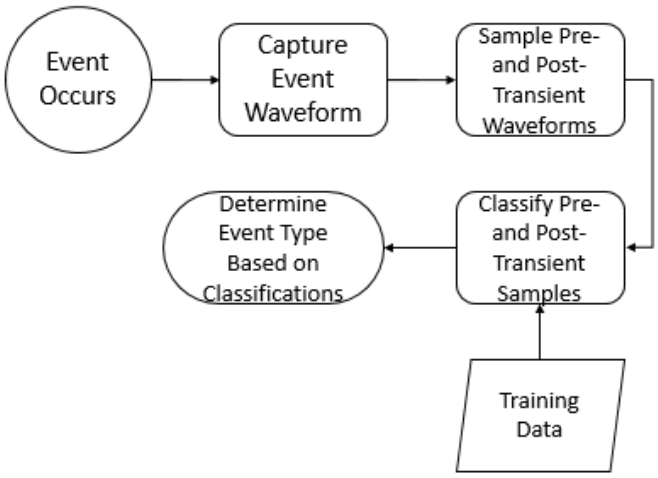

Fig 1. Abstraction of Load Detection Algorithm

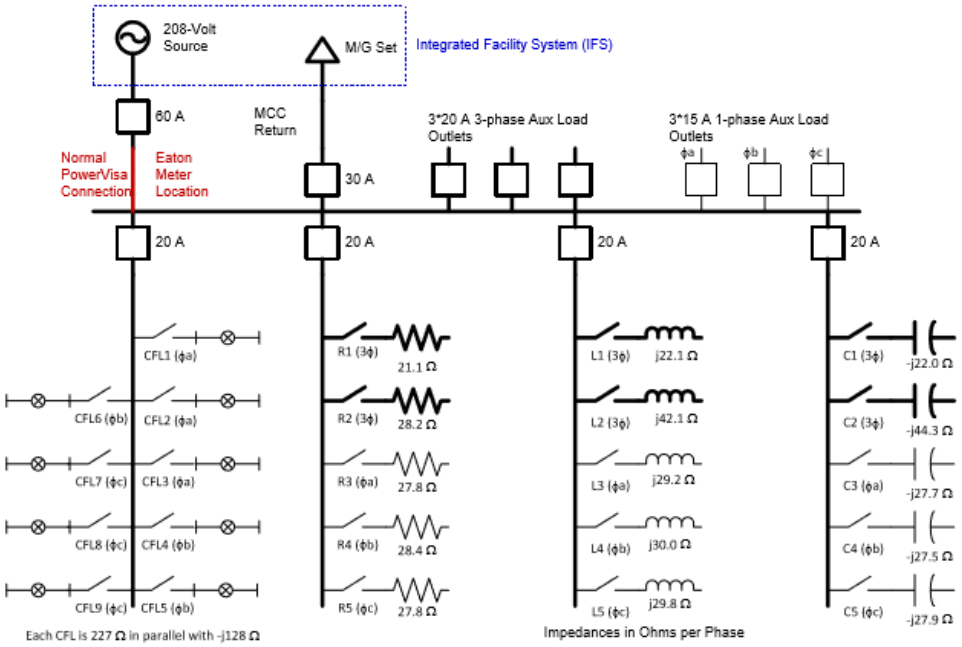

Fig 2. Schematic for data collection

A. Event Detection

In order to simulate events occurring, single phase and three phase shunt resistors, capacitors, and inductors, along with single phase CFLs, were switched on and off using controls within the lab benches of the EPSL. A one-line diagram of the workbench is shown in Figure 2 below. The switches were manually operated and left on for

approximately 2 seconds before switching them back off to allow for most transients to settle. Note that two different event types are simulated in this testing procedure: one turn-on event, and one turn-off event.

A Dranetz PowerVisa was used to measure the waveforms and detect when events occur. The PowerVisa built-in event detection feature can use a variety of metrics to determine when an event has triggered, but given the types of events anticipated, the only criteria used to detect an event is a cycle-to-cycle variation in RMS current. The amount of allowable variation in current before an event is triggered can be set by the user, but for this study, the minimum threshold value of 1 A is used. Changes in variations in RMS voltage are ignored because, for the loads tested in this procedure, the voltage waveforms remain relatively unchanged throughout all events. However, when an event is detected, the phase voltages are recorded along with the phase currents because the inflection points of the voltage waveforms will be used to determine when to sample the current. That means for each event that occurs, six waveforms are recorded. These waveforms will be referred to as the voltage and current profiles.

B. Sampling Waveforms

The process of sampling raw captured waveforms entails determining which data to feed into the machine learning algorithm. The PowerVisa has a sample rate of 256 samples/cycle [8], or 15,360 Hz for a 60 Hz grid. As such, analyzing the transient events would likely produce unreliable results. Because of this, the transient portion of the waveforms are ignored, and the steady-state waveforms from the pre-transient and post-transient portions of each event were the chosen as the input for the NILM algorithm. For consistency, each waveform sample represents two cycles of data. The pre-transient data was cut to include the first two cycles of data, while the post-transient data was cut to include the final two cycles of data. Due to measurement limitations imposed by the equipment, these sampling locations end up approximately two full cycles away from the transient event (exactly 511 steps, or approx. 33.3 ms).

When sampling these waveforms, a common point of reference time is needed for the artificially intelligent portion of the algorithm. Because the voltage waveforms only experience a slight change in magnitude during events and because the current phase shift is defined with respect to the voltage, the positively-sloped zero crossings of the phase voltages were used as the common reference point when sampling waveforms. The starting point of each sample is determined by iteratively analyzing the data for zero crossings. The end point then occurs 511 points after the start because the sampling frequency is 256 samples/cycle. Cutting the current waveform of each phase relative to their phase voltage removes the 120-degree phase shift (for a 3-phase system). Any differences between the waveforms should now only be attributed to the differences in load rather than any other factors that affect phase shift. Note that the sampling method outlined above is used for both the training data and test (non-training) data.

Seen in Figure 3 is a flowchart demonstrating how the waveforms are sampled. After reading the raw waveform, the corresponding voltage waveform for the phase is analyzed for positive zero crossings on both the pre- and post-transient sides of the waveform (step 1). In the example in Figure 3, the post-transient data is being sampled. Once two cycles of the voltage data are determined (step 2), the current waveform is cut at those values (step 3).

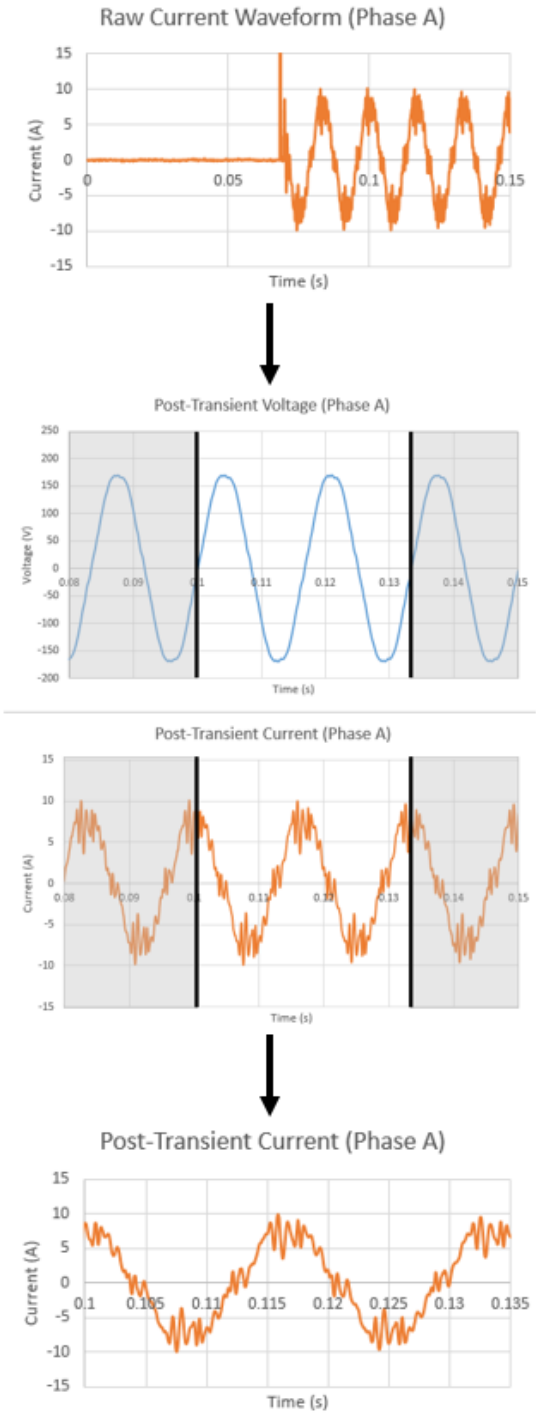

Fig 3. Sampling waveform procedure

In order to sample the pre-transient and post-transient waveforms as described and shown in Figure 3, a MATLAB function was designed. Titled "get_crossings.m", the function first increments through every voltage waveform that is input into it, grouped by phase. To determine where to sample the pre-transient data, the function starts from time 0 s, and positively increments while comparing individual values of Va, Vb, and Vc until the first positive zero crossing is determined for each voltage. The times for each of these first crossings are saved. Because every cycle has 256 data points, the end point for the sample is determined by adding 511 to the time index. The end times for each voltage are also saved. For the post-transient data, the function starts from the maximum time (end of the waveform),

instead of the beginning of the waveform. Again, individual values are compared, this time while negatively incrementing. Once the first crossing is determined, the end of the sample is found by subtracting 511 from the time index. Both times are saved. The final step is simply plugging the start and end time values into each phase current, resulting in the properly sampled waveforms.

C. Classifying Events

In order to classify new data based on load type, a training array and corresponding class array must first be created. Note that waveforms were collected for four different loads – CFL, resistive, capacitive, or inductive. In Figures 4 through 8, the four possible waveforms are displayed, plus the waveform for when there is no load connected (open circuit). For each lab bench, there is a total of 48 events-worth of data - 18 CFL events, and 10 resistive, capacitive, and inductive events. For each of the 48 events, the pre- and post-transient current waveforms for each phase were sampled using "get_crossings.m" and saved into a training array to be input into the MATLAB "classify.m" function. Thus, there is a total of 240 rows of data in the training array for each lab bench.

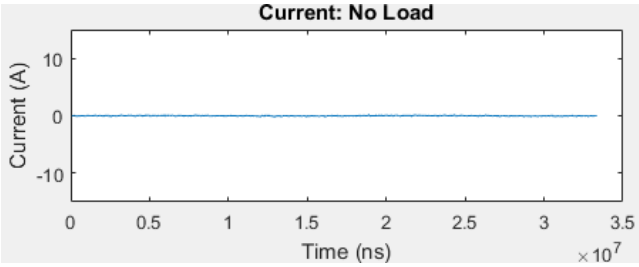

Fig 4. Current waveform: no load

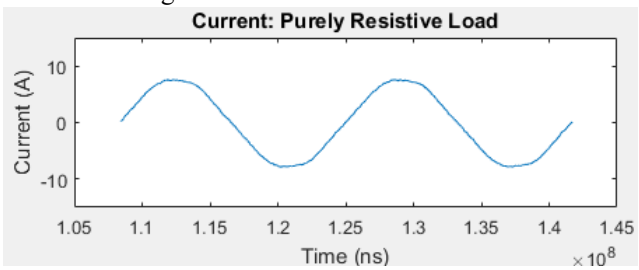

Fig 5. Current waveform: purely resistive load

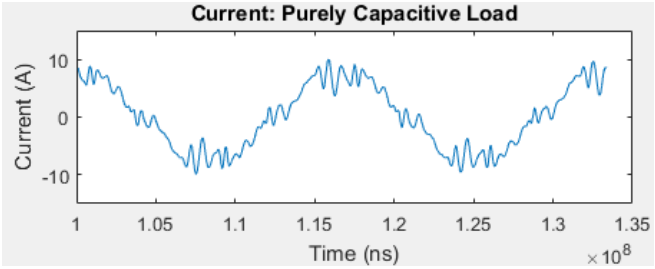

Fig 6. Current waveform: purely capacitive load

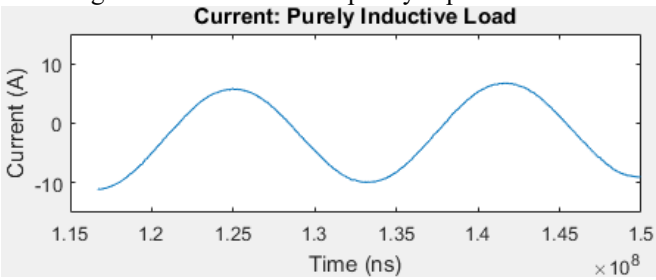

Fig 7. Current waveform: purely inductive load

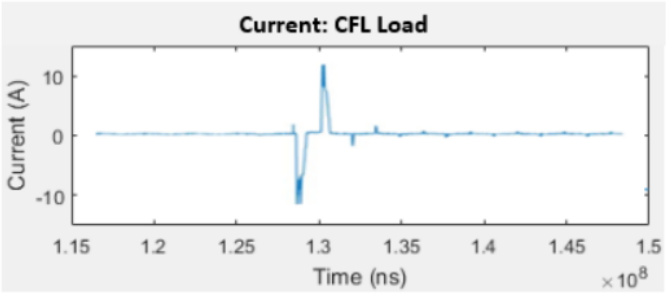

Fig 8. Current waveform: CFL load

In addition to the training array, a class array is needed. The class array specifies which waveform in the training array corresponds to which load type – no load, resistive, capacitive, inductive, or CFL. Because of the large number of waveforms being used to train the algorithm, a function titled "generate_class.m" was created to generate this array. A brute force method was used to populate the training array based on the waveform's filename and the rms value of each waveform. First, if the RMS value of the current is below 1 A, the training data for that row is classified as "no load". If the RMS is above the threshold, the function simply looks into the filename for that set of training data and pulls the load type from the filename. The resulting "class" array has 240 rows, with each row specifying the classifier for the corresponding row in the training array. With the "training" and "class" arrays created, "get_crossings.m" is now used to sample input data that needs classifying. Once the input data is sampled, all three arrays are input into "classify.m", with the resulting output specifying the classifier for each row of input data.

In this stage of development, more training data will lead to more accurate results. Because of the limited number of sample waveforms, when input data is accurately classified, the data will be added to the training array in order to continuously improve the accuracy of the algorithm. In order to automate this process, a new function is needed. One limitation to automating this process, however, is the possibility of the algorithm incorrectly classifying data. There will need to be a manual process to ensure that all classified data is accurately classified, a process that also requires further development.

III. EXPERIMENTAL RESULTS

As explained in Section II, the current and voltage waveforms for each load type were captured using the Dranetz PowerVisa meter. Outlined in the following section, the algorithm was first tested using real lab bench data. The algorithm was then tested using a Simscape model of the lab bench. The model allowed waveforms of varying amplitudes to be used in testing, which was not possible using the physical lab benches. Last, power calculations were included in the algorithm to determine the amount of power used by each load. These calculated results were compared to the readings from the Dranetz meter.

A. Laboratory Data

Data was collected using the physical devices in the six ESPL lab benches for the loads in Figure 2, except the auxiliary loads. Turning each device on and off yields two events per test component (one *turn on* and one *turn off*), meaning 10 events each for resistors, capacitors and inductors and 18 CFL events, totaling 48 events per bench and 240 total events. One bench was used to train the algorithm, while the remaining five benches tested it. The results are summarized in Table 1 below.

Table 1. Load Detection Success Rate for [lab workbenches]

Component	Event Type	Testing Sample Size	Correct Identifications	Success Rate
Resistor	*Turn ON*	25	25	**100%**
	Turn OFF	25	25	**100%**
Capacitor	*Turn ON*	25	25	**100%**
	Turn OFF	25	25	**100%**
Inductor	*Turn ON*	25	25	**100%**
	Turn OFF	25	25	**100%**
CFL	*Turn ON*	45	0	**0%**
	Turn OFF	45	45	**100%**
All Components	***Turn ON and Turn OFF***	**240**	**195**	**81.25%**

Note that the event identifications had the format of "single-phase/three-phase", "load type", and "event type". An example of an event identification for a single-phase capacitor turning on would be: "Single-phase Capacitor Turn ON". An example of an event identification for a three-phase resistor turning off would be: "Three-phase Resistor Turn OFF".

B. Simulation Data

A simulated model of the lab bench was created in Simscape Power Systems and used to generate test data with different impedance values than what are in the lab bench. Note that the operation of the model is very similar to that of the lab bench. Pictured in Figure 9, a 120 V three-phase voltage source drives the system, where each load can be switched on or off at any time. A segment of resistive loads is shown in Figure 10. The set-up for each section of loads (resistive, capacitive, or inductive) include two three-phase impedances, and three single-phase impedances - all three-phase and single-phase loads are controlled by individual switches. Also note that the CFL loads were not able to be simulated using Simscape due to difficulties in representing the a CFL load accurately. After an event is simulated, the three-phase voltage and current waveforms are saved to the MATLAB workspace, as seen in Figure 9. The saved waveforms were then fed into the algorithm for testing.

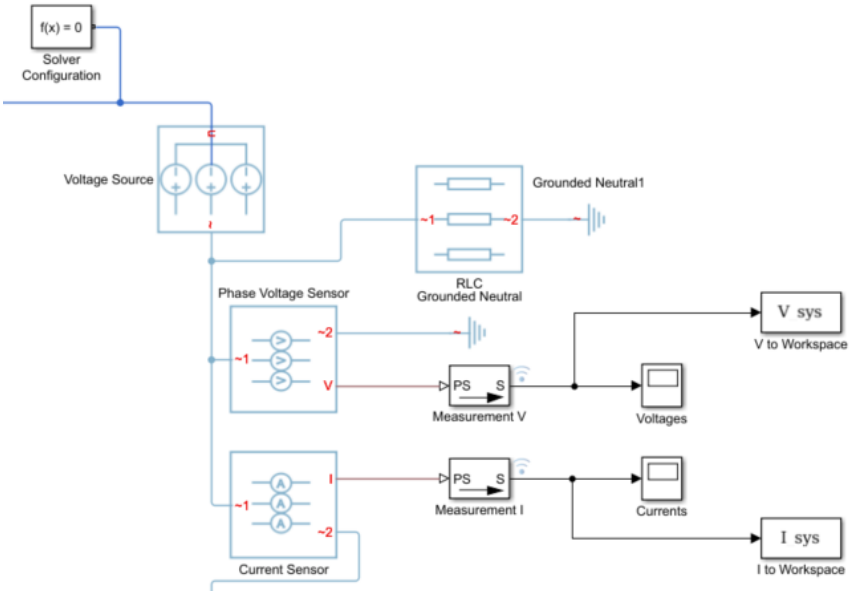

Fig 9. Lab bench model voltage source and measurement location

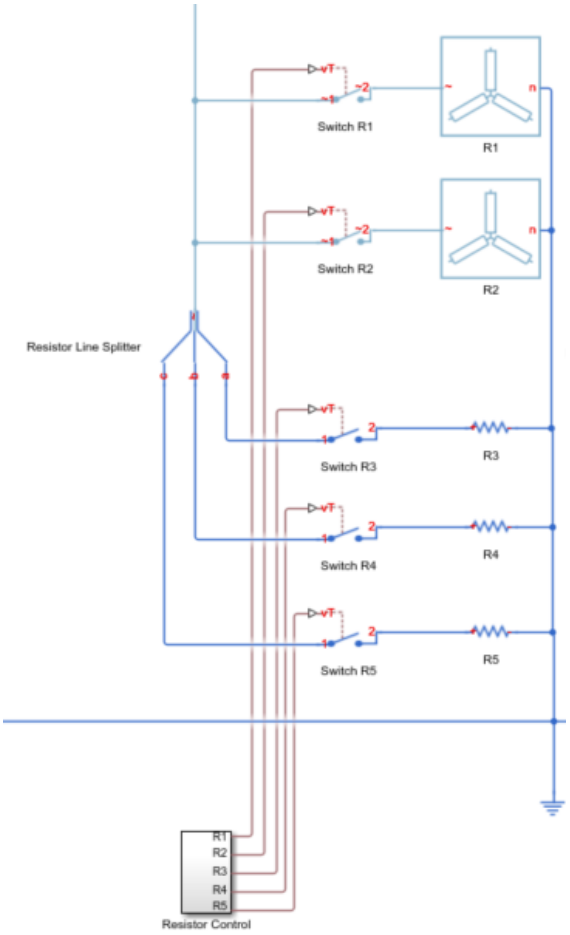

Fig 10. Resistive loads for lab bench model

For testing, the resistance, inductance, and capacitance of each three-phase load were varied. The resulting waveforms were then fed into the algorithm and classified. Seen in Tables 2 through 4 is a summary of the classification results for each varied load value.

Table 2. Simulation results: resistive load

Resistance (Ω)	**Classification Success (Yes or No)**
1E-7	Yes
1E-5	Yes
1E-3	Yes
0.1	Yes
1	Yes
10	Yes
21.2*	Yes
50	Yes
55	Yes
> 60	No – classifies as CFL

Table 3. Simulation results: inductive load

Inductance (H)	Classification Success (Yes or No)
< 0.001	No – classifies as resistor
0.0025	Yes
0.005	Yes
0.01	Yes
0.02	Yes
0.04	Yes
0.05862*	Yes
0.1	Yes
> 0.2	No – classifies as no event

Table 4. Simulation results: capacitive load

Capacitance (µF)	Classification Success (Yes or No)
< 40	No – classifies as CFL
50	Yes
60	Yes
80	Yes
120.57*	Yes
200	Yes
400	Yes
1E3	Yes
1E6	Yes
> 1.5E6	No – classifies as resistor

C. Power Consumption Estimation

Seen in Tables 5 and 6 are the Dranetz power readings compared to the algorithm calculations, with the percent error between the quantities expressed in the right-most column.

Table 5. Real Power Results: Dranetz Reading vs. Algorithm Calculation

Real Power (P) Results			
Load Type	Dranetz Reading (W)	Algorithm Calculation (W)	Percent Error (%)
Resistor 3-phase	2161.6	2265.637	4.81%
Resistor 3-phase	1600	1610.11	0.63%
Resistor phase A	525.26	525.503	0.05%
Resistor phase B	532.78	567.496	6.52%
Resistor Phase C	523.88	528.487	0.88%
Inductor 3-phase	167.88	170.762	1.72%
Inductor 3-phase	93.76	93.719	0.04%
Inductor phase A	46.032	56.813	23.42%
Inductor phase B	45.64	25.916	43.22%
Inductor Phase C	47.382	40.678	14.15%
Capacitor 3-phase	67.126	69.915	4.15%
Capacitor 3-phase	37.828	38.382	1.46%
Capacitor phase A	14.582	11.298	22.52%
Capacitor phase B	14.34	13.056	8.95%
Capacitor Phase C	16.968	16.252	4.22%

Table 6. Reactive Power Results: Dranetz Reading vs. Algorithm Calculation

Reactive Power (Q) Results			
Load Type	Dranetz Reading (var)	Algorithm Calculation (var)	Percent Error (%)
Resistor 3-phase	86.328	69.1009	19.96%
Resistor 3-phase	65.222	41.316	36.65%
Resistor phase A	18.632	22.59	21.24%
Resistor phase B	19.18	27.882	45.37%
Resistor Phase C	23.424	18.593	20.62%
Inductor 3-phase	1995.8	2108.973	5.67%
Inductor 3-phase	1024.48	1107.249	8.08%
Inductor phase A	502.5	532.615	5.99%
Inductor phase B	495.78	500.525	0.96%
Inductor Phase C	494.92	521.443	5.36%
Capacitor 3-phase	-1964.8	-2071.728	5.44%
Capacitor 3-phase	-968.54	-1054.241	8.85%
Capacitor phase A	-522.24	-561.079	7.44%
Capacitor phase B	-523.64	-534.744	2.12%
Capacitor Phase C	-521.02	-556.722	6.85%

IV. DISCUSSION

A. Laboratory Data

The most notable finding in this experiment is that the algorithm never correctly identified a CFL turn on event. Further investigation reveals that the waveforms of the CFL turn on and turn off events, unlike those of the other components, are distinctly different due to the slow settling time of the transient response and restrictions imposed by the measurement hardware. Figure 8(a) superimposes the 45 current signatures collected for the CFL turn on events, with current peaks of about 4 A. Figure 8(b) does the same for the turn off events, but the current peaks are only at about 0.2 A.

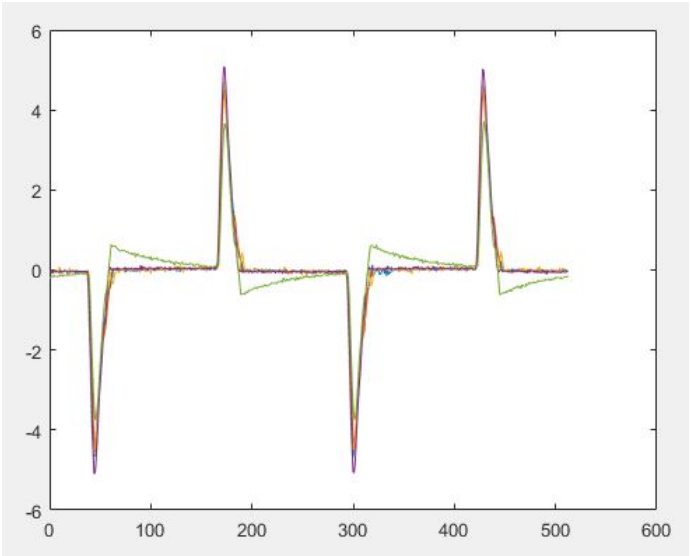

Fig 8(a) - Current signatures collected for all CFL turn on events as a function of the sample index

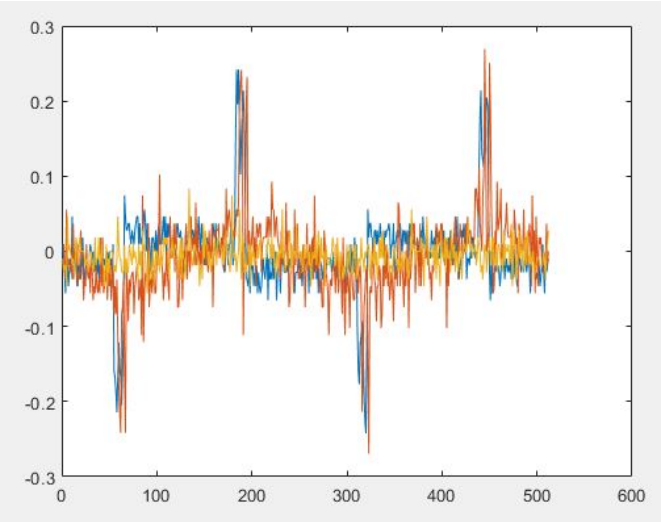

Fig 8(b) - Current signatures collected for all CFL turn off events as a function of the sample index

The measurement device responsible for event detection only records data for a user-selected number of cycles after an event occurs. For the CFLs, the maximum settings of recording two cycles of transient data and four cycles of data before and after the transient did not adequately allow for the CFL transients to dissipate, demonstrated by the drastic difference in the peak currents of Figure 8(a) and (b).

It is therefore suspected that a more sophisticated event detection algorithm that allows for transient responses to properly die out would improve the success rate to up to 100% for the given test components, although other approaches, such as modifying the classification software, could be implemented to allow for different turn on and turn off waveforms. However, such software modifications may add more complexity than is necessary.

B. Simulation Data

To test impedances that could not easily be produces in the lab bench, a simulation was created in Simulink. This simulation used real training data measured from lab benches to classify simulated components. Note that due to nonlinear nature of CFL loads, they were not simulated. However, for consistency, this data remained in the training set.

Seen in Tables 2 through 4, the simulation data is able to be successfully classified for the bench impedance values and impedances within a certain range of those values. However, there are clear thresholds for each impedance where the algorithm begins to classify incorrectly. When the inductance is less than 0.001 H and the capacitance is larger than 1.5E6 μF, the waveform is classified as a resistive load. This result can be explained by the fact that there is a small series resistance for both the inductor and capacitor. When the inductance gets small enough and the capacitance gets large enough, the equivalent impedance is simply going to look like that series resistor because the impedance of the inductor and capacitor are negligible. When the inductance is greater than 0.2 H, the waveform is classified as no event (or no load). This result makes sense because with an inductance that large the impedance is large enough such that the current waveform is essentially flat (close to 0 A) due to the large time constant and small measurement window.

The slightly less explainable results come from the events that were classified as CFL loads. When the resistance was set to greater than 60 Ω and the capacitance was set to less than 40 μF, the events were classified as CFL loads. One would expect, rather, that the events be classified as no event (or no load) based on the same logic explaining why the waveforms for an inductance greater than 0.2 H were classified as no event - the impedances are so large that the current waveform appears to be roughly 0 A. However, these events may be different in the sense that the tested impedances are not quite large enough yet to produce a “no event” classification. Instead, they may be just large enough to make the current waveform appear roughly flat (similar to large segments of the CFL waveforms), and there are more similarities in identifying characteristics to the CFL waveforms rather than the no event waveforms. Another possible explanation is that the standard deviation of CFL turn off events is exceptionally small (see Fig 8b) – so small that it is actually less than the standard deviation of noise during no load conditions. The CFL turn on events, on the other hand, have exceptionally large standard deviations (see Fig 8a). The

phenomena could explain both why large simulated resistors, small simulated capacitors, and CFL turn on events are erroneously classified by this algorithm.

C. Power Consumption Estimation

The results in Section III, part C demonstrate that the power calculations are accurate to the measured power values from the Dranetz meter. Highlighted in Tables 5 and 6, the key result is that the loads that draw predominantly active power (resistors), have very small error between the measured active power and the calculated active power. The maximum deviation in active power results for the resistive loads is 6.5%, while all but two of the calculations are below 2% error. Additionally, loads that draw predominantly reactive power (inductors and capacitor), have very small error between the measured reactive power and the calculated active power. The maximum deviation in reactive power results for both capacitive and inductive loads is 8.85%, while the majority of calculations are blow a 7% error.

Seemingly large discrepancies are seen in the reactive power calculations for resistive loads, and the active power calculations for capacitive and inductive loads. While both of these cases show errors as large as 45%, it is important to note that the quantities of these types of powers are much lower than the predominant power being used by the load. In other words, the large power factors of the resistive loads indicate that the amount of reactive power calculated for those loads is insignificant and possibly due to parasitic affect. The inverse argument can be made for the reactive loads.

V. SMART GRID LAB PROPOSAL

A goal of this design project is to translate some of the algorithm design process into a laboratory assignment for the newly-designed undergraduate Smart Grid and Power Quality classes at the University of Pittsburgh. In this section is a procedure outline that guides the students in creating a basic load-detection algorithm that is based off of the results of this report. Note that this procedure could be altered to fit the needs of a given set of students.

A. Data Collection

The data collection component of the lab will closely copy the data collection process outlined in Section II, Part A. First, the Dranetz PowerVisa will be set up as described earlier in the paper. These settings ensure that the Dranetz will automatically capture all necessary waveforms when each load is turned on and off. The captured events will match what was captured in Section II. Every single-phase and three-phase resistor, capacitor, and inductor in the [lab workbench] will be manually switched on and off. Note that each component should be left on for approximately 2 seconds to ensure that steady-state operation has been reached for each load. The same procedure will be carried out with 6 of the CFL loads (2 for each phase).

After data collection is complete, the voltage and current waveforms for each event can be transferred from the Dranetz memory card the lab computer using the DranView software package. For the sake of easy import into MATLAB, it is recommended that the data is saved as text files. It is also recommended that each data set is saved in a way that easily identifies which load (or no load) corresponds to each set. This convention will make it much easier to create the classification array.

B. Algorithm Development

With the data saved on each students' laptop, the next step is for them to design their MATLAB script to read and analyze the data. The first script will be designed to sample the data based on the description in the *Sampling Waveforms* section. The goal of the student is to read the first and final two cycles of each captured waveform and cut the signals in a way that mimic what is seen in Figure 3. The resulting sampled waveforms should closely resemble what are seen in Figures 4 through 8.

Once a matrix is saved with the sampled waveforms, the training and class arrays can be generated. The training array should simply contain the majority of the sampled signals. Note that the student should keep a handful of signals (10 or so) separate from the training array – these will be the signals used to test the algorithm. The class array will correspond to each row of data in the training array, labeling which row is which load (or no load). To create the class array, it is suggested that the students create a code that can read the original file names to determine which row of data corresponds to which load. For more details generating the training and class arrays, see the *Classifying Events* section.

The final part of the algorithm is to implement the built-in *classify* function. The inputs of this function are the training, class, and test arrays. Note that based on the research in this paper, the 'diaglinear' method in *classify* produced the most accurate results. As stated previously, the test array should contain 10 or so test sets of data. The

output of the function will be the classifications for each row of the test data. The students should report on the accuracy of the results and if there are any discrepancies or unexpected outcomes.

VI. CONCLUSION

Using the equipment available to all electrical engineering students at the University of Pittsburgh, the authors were able to successfully identify resistive, capacitive and inductive loads across different phases with *100%* accuracy. Furthermore, simulation shows that a variety of different impedances can successfully be detected by the same algorithm. The major shortcoming of the algorithm described in this paper is the inability to precisely detect nonlinear loads (i.e. CFL's). This outcome is to be expected when only using raw waveforms to classify loads [1].

As suggested by these results, NILM in practical situations (i.e. with highly nonlinear loads) is a nontrivial process that has taken decades of research to develop. However, this work shows that by simplifying the problem to identifying a predetermined set of linear loads, undergraduate electrical engineering students can readily replicate a simplified NILM algorithm and stand to learn a variety of skills in doing so. Even, when restricting the scope to identifying linear loads only, students obtain experience coding, measuring and processing data, working with basic artificial intelligence algorithms, and thinking about the role of demand side management in the smart grid paradigm. Additionally, by introducing some nonlinear loads, like the CFL, students begin to understand the limits of basic artificial intelligence algorithms, like discriminant analysis. Furthermore, this procedure can be easily built upon and customized to fit a variety of skill levels. For example, students more interested in power systems could restrict the scope to only linear loads and focus on how NILM can be used in power systems. On the other hand, students more interested in signal processing and artificial intelligence can research and implement methods used to identify more complex loads. Therefore, the curriculum at many institutions could be improved by implementing a NILM lab exercise, such as the one detailed in this paper.

REFERENCES

[1] G. W. Hart, "Nonintrusive appliance load monitoring," in *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870-1891, Dec. 1992. doi: 10.1109/5.192069

[2] Sense Home Energy Monitor. (2019). The Sense Home Energy Monitor. [online] Available at: http://sense.com/ [Accessed 1 Feb. 2019].

[3] Neurio. (2019). Home - Neurio. [online] Available at: https://www.neur.io/ [Accessed 1 Feb. 2019].

[4] K. D. Anderson, M. E. Bergés, A. Ocneanu, D. Benitez and J. M. F. Moura, "Event detection for Non Intrusive load monitoring," *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, Montreal, QC, 2012, pp. 3312-3317. doi: 10.1109/IECON.2012.6389367

[5] P. Bilski and W. Winiecki, "Generalized algorithm for the non-intrusive identification of electrical appliances in the household," 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Bucharest, 2017, pp. 730-735. doi: 10.1109/IDAACS.2017.8095186

[6] [citation contains information that displays the name of the institution]