Distribution Level Composite Load Modeling

by

Kevin McCormick

Bachelor's in Electrical Engineer Technology from University of Pittsburgh at Johnstown, 2017

Submitted to the Graduate Faculty of the

Electrical and Computer Engineering Department

of the requirements for the degree of

Master of Science in Electrical and Computer Engineering

University of Pittsburgh

2023

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Kevin McCormick

It was defended on

April 7, 2023

and approved by

Mai Abdelhakim PhD, Assistant Professor, Electrical and Computer Engineering

Brandon Grainger PhD, Associate Professor, Electrical and Computer Engineering

Thesis Advisor: Robert Kerestes PhD, Associate Professor, Electrical and Computer Engineering

Distribution Level Composite Load Modeling

Kevin McCormick, MS

University of Pittsburgh, 2023

Traditionally, distribution systems analysis has been done using steady-state power flow solutions. This approach uses empirical or forecasted data for sources and loads, and the system is analyzed using basic engineering principles. Later, quasi-steady state analysis was used to introduce time variation in the model due to variations in weather and other seasonal conditions. However, due to the increasing penetration of distributed energy resources (DER) such as wind, solar, and storage, power systems analysis requires a more dynamic approach.

Traditional analysis has been done primarily by utilities for utilities, and a utility-based approach comes with limited customer-level fidelity. Often, they cannot look any further downstream than the substation level. Since many modern DERs are being installed at the residential level, this is becoming increasingly insufficient in modern modeling.

By applying dynamic modeling techniques at the distribution level, we can add fidelity to the model, and get a more accurate dynamic representation of these systems. For this work, a prototype of the Western Electricity Coordinating Council (WECC) composite load model, was developed for distribution systems dynamic analysis. The WECC model has be implemented and used in dynamic transmission systems analysis, but not at the distribution level.

Using the IEEE 13-node test feeder as the basis, a program was developed that takes inputs from a user and creates a WECC composite load model. This model is added to the desired node in the system. This process is repeatable, and every node may have a composite load if desired.

In this work, the history of the WECC model is discussed, followed by some background on distribution systems simulation. This is followed by some background on the components used

in the model, along with a brief description of the different stages of development towards this model. Finally, we look at how these steps come together to create the final model and analyze the results.

Table of Contents

List of Tables

List of Figures

1.0 Introduction: Load Modeling History

Load Modeling is the practice of developing models and identifying parameters within those models. [1] Unfortunately, due to gaps in data there will never be a perfect model of a power system. This lack of knowledge has always been a problem, but was traditionally met with conservative characteristics. [2] However, as computational power improves, and techniques are developed we can get a progressively better approximation. Modern simulations began development in the 1980s. Prior to that, stability was a question that was asked only in the context of generation. Therefore, characteristics of the loads were only looked at insofar as was needed to help with analysis at the generator [3].

Simple load models were adopted, where the active and reactive powers were represented as constant current and constant impedance, respectively [4]. Unfortunately, these simple loads were not sufficient. It was known for a long time that conservative loads cannot yield universally conservative models [4]. However, through the 1980s EPRI demonstrated the advantages of component-based load modeling. Specifically, between 1981 and 1986 research had moved from determining basic characteristics of a load to the creation of the EPRI Load Modeling Reference Manual. [5,6]

Later, EPRI developed the LOADSYN software that was able to take input information about the load composition, class mix and characteristics to construct a model consisting of a constant impedance (Z), constant current (I) and constant Power (P) component. [4] This is commonly referred to as the ZIP load model.

There are other models that have been used, such as the polynomial load model, exponential load model, and frequency-dependent load model [5] but none of these have stood the

test of time and no modern references to these methods were found. That is to say, other methods of analysis exist but the ZIP model is the most commonly used.

In 1993, IEEE issued a recommendation for the use of dynamic load models in power system stability studies [7], but it was only after a major event took place in 1996 that it began to gain traction. The event in question was a series of two outages, on July 2 and August 10 of 1996. These took place in the Wester Electricity Coordinating Council, or WECC. Model validation studies used in the aftermath of the outage concluded that dynamic motor models were needed to capture the North-South power oscillations [8]. With this, the usefulness of dynamic modeling was made readily apparent.

In response to this need, WECC developed a model in wherein 20% of the load was represented by a dynamic motor and 80% was represented as a static load. This new model was able to capture the oscillations but lacked realism due to a lack of representation of the substation transformer and distribution feeder impedances. Traditionally, the loads were aggregated at high voltage busses, upstream of these components which were then neglected.

This version of the model was called the interim model and was further developed to show Fault Induced delayed voltage recovery (FIVDR) events. These were seen as reliability risks and needed to be captured in simulation studies. FIVDR events, it should be noted, are caused primarily by single phase residential loads. They occur because these loads have a low inertia, and can easily stall if voltage sags, in which case there is a delay as they start again before the bus voltage recovers.

WECC Developed two more load models, one in 2005 that addressed the FIVDR events and one in 2009 that included the addition of a single-phase motor model to represent the residential load issues. This completed model, which will be explored in more detail in section 2

of this thesis, is considered *the* composite load model. This is, technically, untrue, as there are others including Southern Company Edison (SCE) developed a composite load model in 2007. The concept itself is, in fact, generic and simply means that the system has dynamic and static components.

1.1 Modern Dynamic Load Modeling

The WECC model remains the most popular composite load model and is often referred to as the singular example of a composite load model. Indeed, it is widely considered the most accurate model. However, that accuracy comes at a price. It takes over 131 input parameters to run. Attempts to filter these parameters down have been attempted, including an attempt using the Active Subspace Method (ASM) [9]. This method attempts to establish sensitivity and interdependency of the parameters. Sensitivity being the direct effect a parameter has on the system. For example, if doubling parameter A would double the system output, and doubling parameter B would only increase it by 25% you could say that A has a higher Sensitivity. Interdependency, however, only cares how much a parameter effects the other parameters. i.e., what if a third parameter, C, was doubled and in turn increased A by 50% and B by 100%. B and C would have a higher interdependency than A and C, but since A has a higher overall effect, the latter relationship would be weighted more heavily.

It should also be noted that the WECC model is aimed at dynamic solutions. These solutions are usually geared towards system responses to an event such as a fault or lightning strike. There are other simulations that look and plan for overall stability and consider much larger

timescales and the issues associated with them. For example, how do you account for things like sunlight, wind and temperature? How does this affect Distributed Energy Resources (DER), such as wind and solar? These questions are looked at in Stability studies [10], as well as Quasi-Static Time Series Simulations [11], though these simulations tend to be on the order of a year or more in length. As these simulations take place over a longer time period their time step must be raised to match, which is usually on the order of seconds at a minimum.

While these simulations are not dynamic, they are not static either. Both can look at load shapes, but QSTS, specifically, takes inputs from previous states of the system to calculate the next states. This is where the name is derived from, and is the primary difference between QSTS and normal stability analysis [12] QSTS is also an area of active research used in many applications, including DC power flow stability studies [13].

These examples are here to show that there are other vectors of research looking at ways to solve both dynamic simulation issues as well as general modeling considerations. Some of these approaches are theoretical, some use models and other can use measurements in the fields to find new insights. This thesis, however, will be covering just the modeling portion.

1.2 The WECC Model

At a high level the WECC model is not overly complicated, as shown in Figure 1. There is a Grid Source, transmission lines, distribution transformer information and finally a load bus. This bus represents the entire downstream load. It could represent a few houses, or a whole suburb or

even an entire city with residential and commercial loads. That input data is broken down and represented here by six components. Four different motors, a static load and a power electronics load.

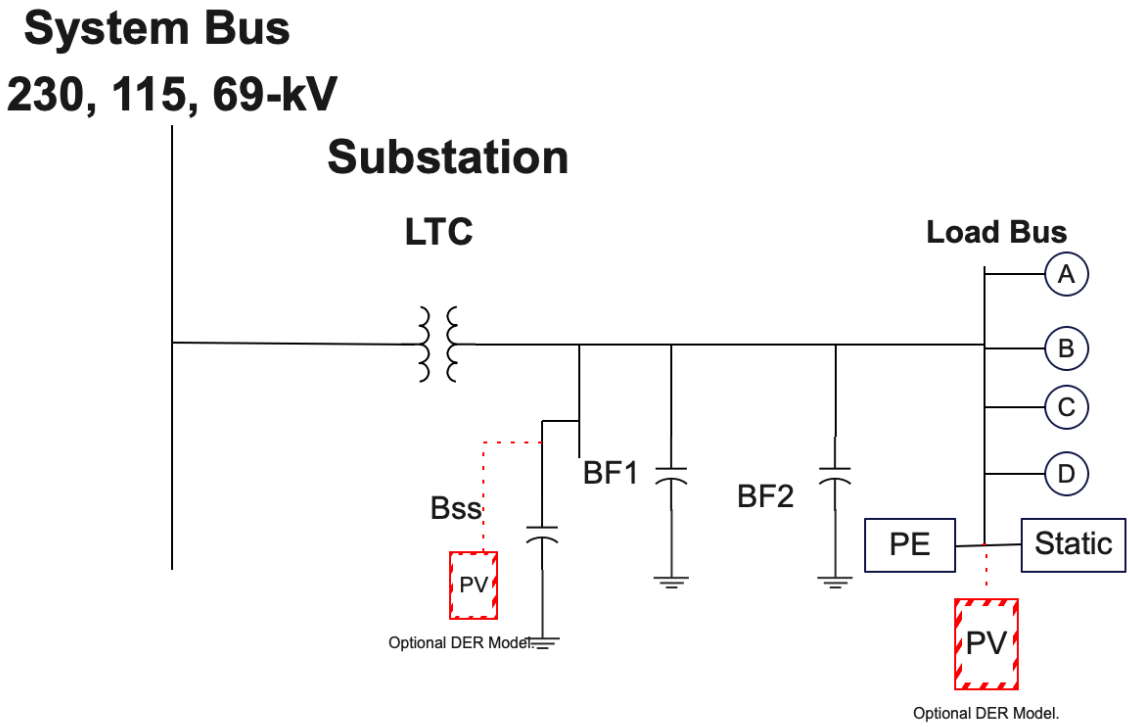

Figure 1. WECC Composite Load Model

Of special note are motor D and the Power Electronic load. Motor D is the single-phase motor for residential use that was previously mentioned, and it operates differently than the other motors. The power electronic load has some overlap with this behavior. Essentially, both loads will, as the voltage drops during a simulation, change. Motor D will, at certain cutoffs, change operating conditions or stall completely. The Power electronics, which represents all manner of VFDs and controls on modern equipment, is represented by a linear drop off as more and more devices switch off. This is shown in figure 2.

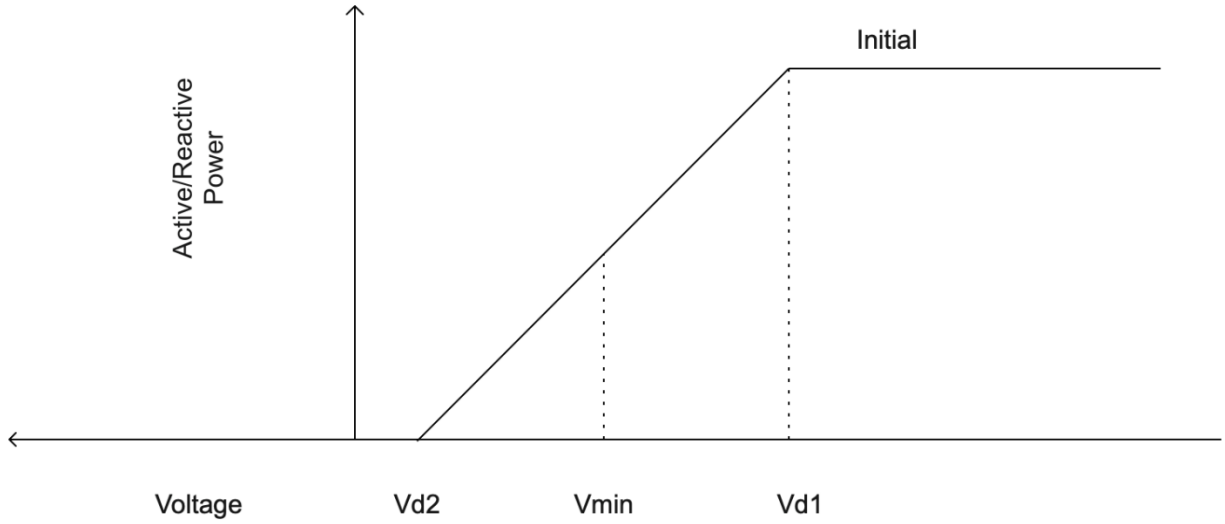

Figure 2. Power Electronic Component of the Composite Load Model

The motor types are:

A–Low Inertia Constant Torque

B – High Inertia Variable Torque Loads

C – Low inertia Variable torque Loads

D – High Torque using low slip. (Easily stalled and easily restarted)

That said, it is important that we break down these uses in greater detail. Above is a detailed description of the components of the WECC model, but the amount of usage each component gets is going to depend on the load class mix. Additionally, it was also mentioned that 80% of the load is static in nature. However, while the 80:20 ratio of dynamics/static loads is a good rule of thumb it is not a requirement in modern WECC models.

The Load class is the type of load that is being serviced in each system. This is different than the specific loads at a given node, which, As mentioned previously, could be a residential suburban home, or a downtown high rise. For example, the houses are residential, and the downtown locations are commercial. Both, however, would be serviced by a distribution circuit which feed primarily Residential and Commercial loads. However, if that circuit is mostly Commercial it may be labeled a "Commercial" system, despite having as much as 20% residential loads [14]. This overlapping of terms can cause some confusion, but both entire nodal systems and the nodes themselves can be classified via load class, with different rules governing each level.

The type of load class determines whether, and to what extent, each load is used. For example, class A motors do not appear in a residential load. Therefore, Residential loading would never have any of these motors as part of the Class Load mix. In practice, these motor types may be present in a residential setting, but depending on the rules of association may remove it entirely from the model.

Conversely, Class D motors are almost exclusively residential in nature. Just from this broad information it becomes obvious that each type of load class can be assigned values that can then be distributed back in based on weights. The details of how to best distribute will be explored later, but the concept is illustrated in Figure 3 and further expanded in Figure 4, which gives a sample table of what the rules of association look like. Remember that, so far, all of these are based on 'best practices' for general loads, and this information can be further refined if you have specific load information in mind.

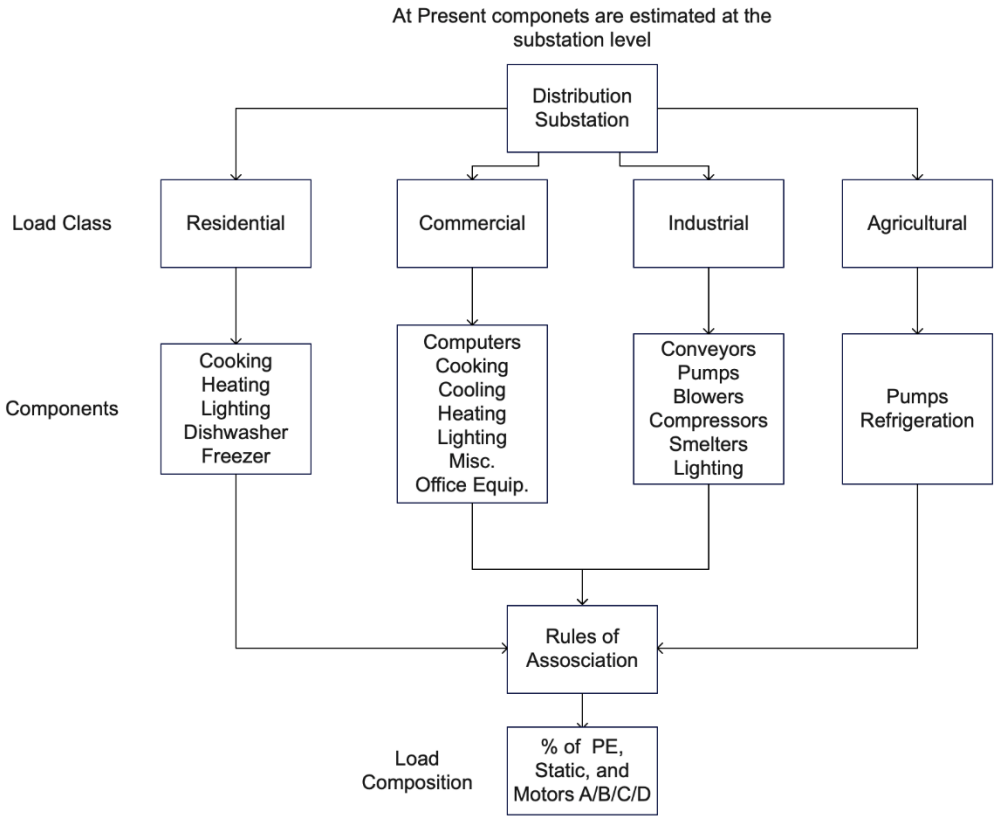

Figure 3. Rules of Association Flowchart

	A	B	C	D	pe	z	i
Clothes Dryer	0	0	0.4	0	0	0.6	0
Cooking	0	0	0	0	0	1	0
Cooling	0	0.1	0.1	0.8	0	0	0
Dishwasher	0	0.5	0	0	0	0.5	0
Freezer	0	0	0	1	0	0	0
Furnace Fans	0	1	0	0	0	0	0
Heating	0	0	0	0.1	0	0.9	0
Lighting	0	0	0	0	0	1	0
Other	0	0	0	0	0.5	0.5	0
Hot Tubs and Spas	0	0	1	0	0	0	0
Pool Pumps and Filters	0	0	1	0	0	0	0
Personal Computers	0	0	0	0	1	0	0
Refrigeration	0	0	0	1	0	0	0
Television	0	0	0	0	1	0	0
Water Heating	0	0	0	0	0	1	0

Figure 4. Rules of Association Table (Residential Load)

The above table shows an example of the current rules of association. To elaborate, these rules define the loads as having certain ratios of different load types. The top row is a list of those loads which are, in order, Motor Classes A, B, C &D, Power Electronics (pe), Impedance (z) and current (i). Each column must add to one. As we can see, residential loads lack Class A motors and current based loads. Additionally, some loads are simple to model, such as furnace fans which are completely a class B motor.

Conversely, cooling, which may include compressors, pumps and fans, has been broken into three types of motors. 10% Class B, 10% Class C and 80% Class D. This distribution is the result of component-based measurements that come from studies that can define load shapes for different load classes. It should be noted that these results are generic, and can be adjusted if

you have the data to drive that decision. For example, if you are running two separate studies and one is in Arizona and one is in Pennsylvania you are likely to have different load shapes given the climate differences. Therefore, you can create different rules of association to accommodate these differences. It should be noted that, while not perfect, the US has publicly available load shape data for such applications [15].

One final topic that is worth mentioning as it becomes more prevalent is DERs and their effect in the dynamic landscape. Currently, DERs are usually just implemented as Photovoltaic (PV) arrays [16]. Other types of DER are modeled as generators or load models. PVs, as well as battery storage, influence the fault response that is much different than previous loads. Normal loads have inertia as the rotating motor or generator spins. During a fault, this spinning is turned into electrical energy that feeds the fault. This is not the case for PVs, which generally use steady state inverters to connect to the grid [17]. This problem, while more pronounced at a residential scale, is also becoming more of an issue even at the generation level [18]. As more solar generation opens up the system will lose momentum, and the fault characteristics will change accordingly.

This change maniefests as a lower overall fault current. Traditional faults, owing to the large physical inertia of both motors and generators, will create large currents on the order of 5-6 times normal current levels. This creates an extremely dangerous but easily detectible overcurrent. Conversely, DER systems create low current faults that are much more difficult to detect. This is still rare in normal systems, but microgrids have created situations where there may be little to no inertia and therefore alternate fault detection methods must be used. This is an area of ongoing research. [19]

DERs have also given rise to 'hybrid' plants which use multiple sources to generate power. This is typically in the form of solar and battery power but could also include wind or gas depending on the use [20]. As a final note, DERs can be added to the WECC model at both the Load Bus and the Bss Bus, depending on the scale of the generation. This addition is beyond the scope of this project, however.

DERs are just one component that goes beyond the standard WECC model. There are also active areas of research at both higher and lower levels. At the high level, there is a growing area of research for seeing system control responses during situations that are designed around maximizing profit instead of minimizing damage. For example, there are a few projects looking at using controls to maximize profit [21, 22]. Other projects seek better ways to generate aggregated load data [23] to increase the fidelity of the model even with low visibility into the system. At the low level, there is research into dynamic modeling all the way to the plug-in load level [24].

2.0 Distribution System Simulation

OpenDSS is a script-driven simulation engine that has the flexibility to model any distribution network. It is open source, and widely used for power flow analysis, harmonic power flows and short circuit analysis [25]. OpenDSS is not used as much for Dynamic modeling, and so literature on the subject is sparse. This document attempts to show the usefulness of developing a prototype in OpenDSS that can model the WECC composite load model at the distribution level.

2.1 Dynamic Modeling

OpenDSS, when downloaded, comes with many resources to help learn and test the software. It also includes several example feeders, including the IEEE 13 node feeder [26], shown in Figure 5. This feeder is for testing, primarily, as it is too small and unrealistic to draw general conclusions. However, for our purposes it was useful as a test to make sure the software was working, and to edit in dynamic components.

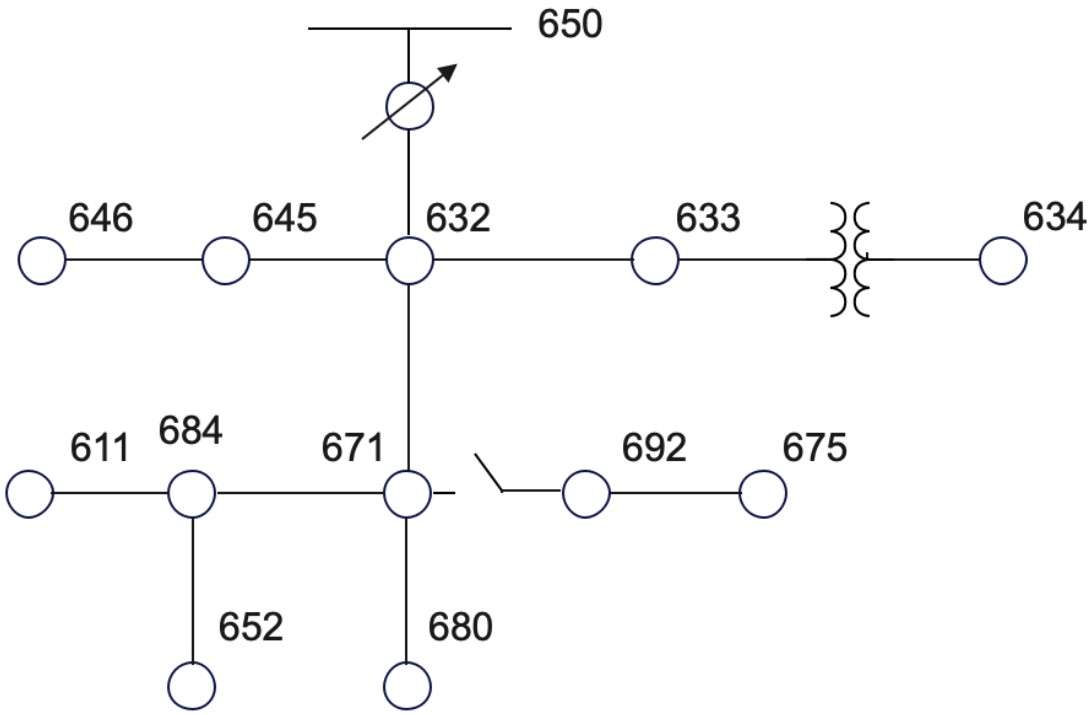

Figure 5. IEEE 13 Node System

2.1.1 IEEE 13 Node Dynamic Feeder

For the first iteration the IEEE Dynamic study the 13-node system was used as a base. From there, we removed the loads at Node 634, and added three dynamic components. One was a generator, and the other two were PV systems [27]. This was the system that was primarily tested against, and the default results will show an unmodified version of the outputs.

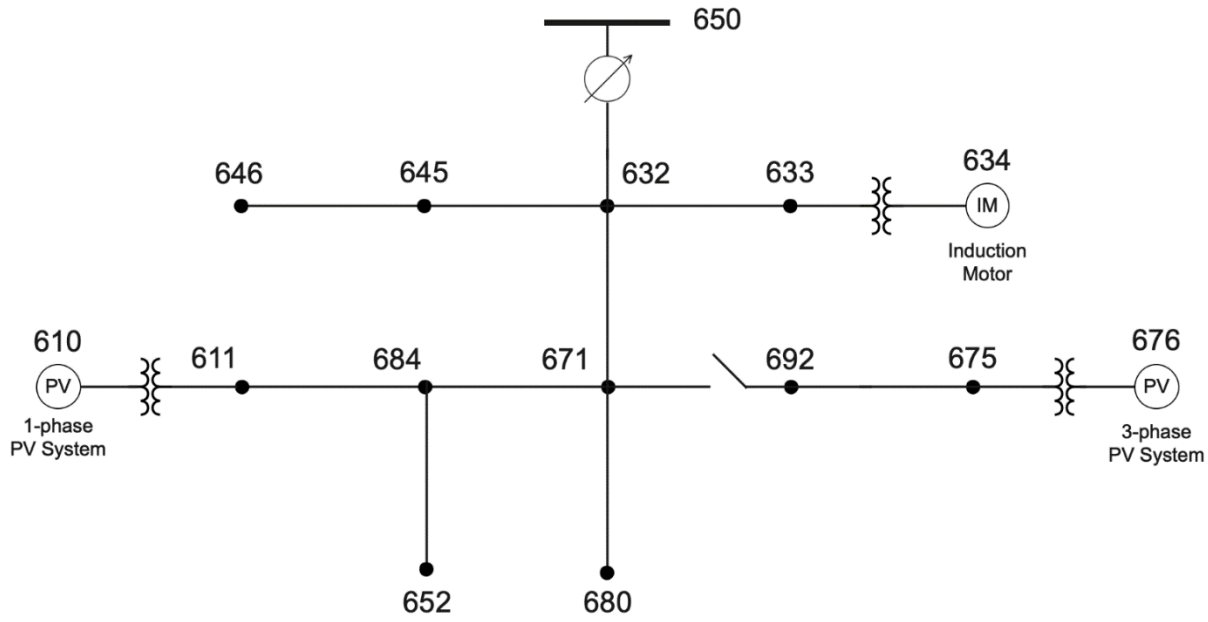

Figure 6: Dynamic 13-Node Test Feeder [27]

2.1.2 WECC Model In OpenDSS

To start, we created and OpenDSS model that was just a faithful representation of the WECC model. In other words, we created four motors, a static load and a power system load. Although the use cases have been previously discussed, the details of how that translates into parameters for the motors needs to be explored. First, there is a disparity between OpenDSS and the NERC Motor parameters. The NERC parameters for each motor type are more limited, so we will explore those first.

NERC motors constitute 8 parameters that have been gained through analysis and experiment. [28]. These are listed in the table below, see Table 1.

Table 1. Motor Parameters

Parameter 1	Meaning	Parameter 2	Meaning
LF	Loading Factor	Ra	Stator Resistance
Ls	Synchronous Reactance	Lp	Transient Reactance
Lpps	Sub Transient Reactance	Tpo	Open Circuit Time Constant
Tppo	Sub Transient Open Circuit T.C.	H	Inertia constant

These 8 parameters define the models for motor types A, B and C. However these can be thought of as the difference between a static and dynamic model, as there are a lot of basic parameters that are not mentioned here. For example, what voltage, frequency and phase angle are you operating at? These need to be accounted for in OpenDSS for every simulation, alongside other variables.

There are default values for the OpenDSS induction motor, such as the per-unit stator and rotor Resistance and Impedance which are as follows:

Table 2: OpenDSS Pre-existing parameters

puRS	Per-unit Stator Resistance: .0053	puRr	Per-Unit Rotor Resistance: .007
puXS	Per-unit Stator Impedance: .106	puXr	Per-unit Rotor Reactance: .12

However, it is important to update these to reflect the values developed by EPRI so that the model is a faithful WECC model.

3.0 The Model

The model used in this project was coded in OpenDSS. OpenDSS has dynamic capabilities, though they are currently limited. Fortunately, there is a growing body of literature on the subject [29]. Fortunately, the dynamic limitations of OpenDSS do not apply to this thesis but will be explored in the conclusion of this paper as an area of further research. For the purposes of this model, there is a need to recreate the WECC Dynamic Load Model in OpenDSS.

3.1 Methodology

As previously discussed, the WECC Model is broken into several components. These are Static Loads, Power Electronics, and Motors. Motors, in turn, have 4 classes: A, B, C & D. This makes 6 unique loads that need to be programmed and accounted for. As a note, Figure 1 also has transformer and transmission components. These are ignored for this model because we are looking at distribution modeling, and so all of our modeling is looking downstream of these components.

3.1.1 Static and Power Electronic Loads

Static Loads are straightforward. For this study, we are going to look at them as completely unchanging. They will also represent most of the load by kVA. In traditional composite models

they represent 80% of the load, but the rules of association in more modern approaches mean that this number is in flux not just generally, but on a per-class basis. However, with DER technology becoming more common at all levels of distribution it is likely that this static load percentage will be declining for the foreseeable future as Power Electronic loads consume a larger percentage.

These power electronic loads are, however, programmable and can respond to any number of conditions. Whereas traditional components, such as breakers, would only respond during fault or other high current conditions these allow much more nuanced control. The representation in the WECC model is to respond only to voltage and to do so in a linear fashion. When the terminal voltage drops below a certain setpoint, Vd1, the load is shed linearly until it is completely cut off at a second Terminal Voltage, Vd2.

3.1.2 Class A, B, C & D Motors

Class A, B and C motors and their modeling components have been previously covered, but there is a difference between the WECC model and the model in OpenDSS. The WECC model parameters are shown in Table 1, and most of these have direct counterparts in OpenDSS. However, in order to use them we have to use the Generator component and run it as a negative value. This is because the generator is synchronous, while the motors in OpenDSS are all induction motors. That said, it doesn't all translate one to one, and notably the time constants Tpo and Tppo cannot be used.

Class D motors are different, however. Rather than modeling a different class of motors, the fundamentals are also completely different and therefore must be implemented differently. As previously defined, motor D needs to operate in different states. This was implemented in the

model by having it run in either a ‘run’ state or a ‘stall’ state. In a run state the motor operates as normal, and in a stall state the motor becomes a high kVA load with no dynamics. The stall state is achieved when the voltage drops below 45% of nominal voltage. After this, if the voltage is restored, the motor can re-start after .3 seconds.

4.0 The Prototype

In the previous chapters we've gone over the background that led to the creation of this prototype, along with the detailed steps of that process. Now, we can discuss the outcome. First, we will discuss the needs of the program, then the inputs, and finally we can discuss the outputs. Future expansions on the program will be explored in section 5.0, conclusion.

First, this program requires a functioning OpenDSS file. This was designed, specifically, around the IEEE Test Feeder systems. Although the 13 node system was the primary example and what was tested the most thoroughly, it would only take a minor edit or two to work with an arbitrarily sized system.

4.1 Inputs and Outputs

Additionally, since this system was designed for engineers, it does require knowledge of the system to be used effectively. As previously discussed, the system takes inputs in the form of ratings for the system and nodal locations and outputs WECC composite load models to those node locations, allowing for dynamic simulations to be added throughout the simulation. However, it is up to the engineer using the system to provide a model they are familiar with so that they can input the proper information.

Now, let’s walk through the implementation of these inputs and outputs.

4.2 Python Code

OpenDSS comes with a feature called the COM Interface, which allows it to interface with other programming languages. For this project Python, and the compiler Pycharm, were used. The program requires a completed OpenDSS script for the program to begin with. For testing purposes the 13 node test feeder was used here, but any finished system would work. From here, the program asks the user if they want to add a WECC model. If yes, they must input information about the location, size, voltage and phases of the node they are connecting to. Once done, the model will generate the WECC model and place it at the specified node. Then, it will loop until the user declines to place a WECC model, as shown in Figure 6.

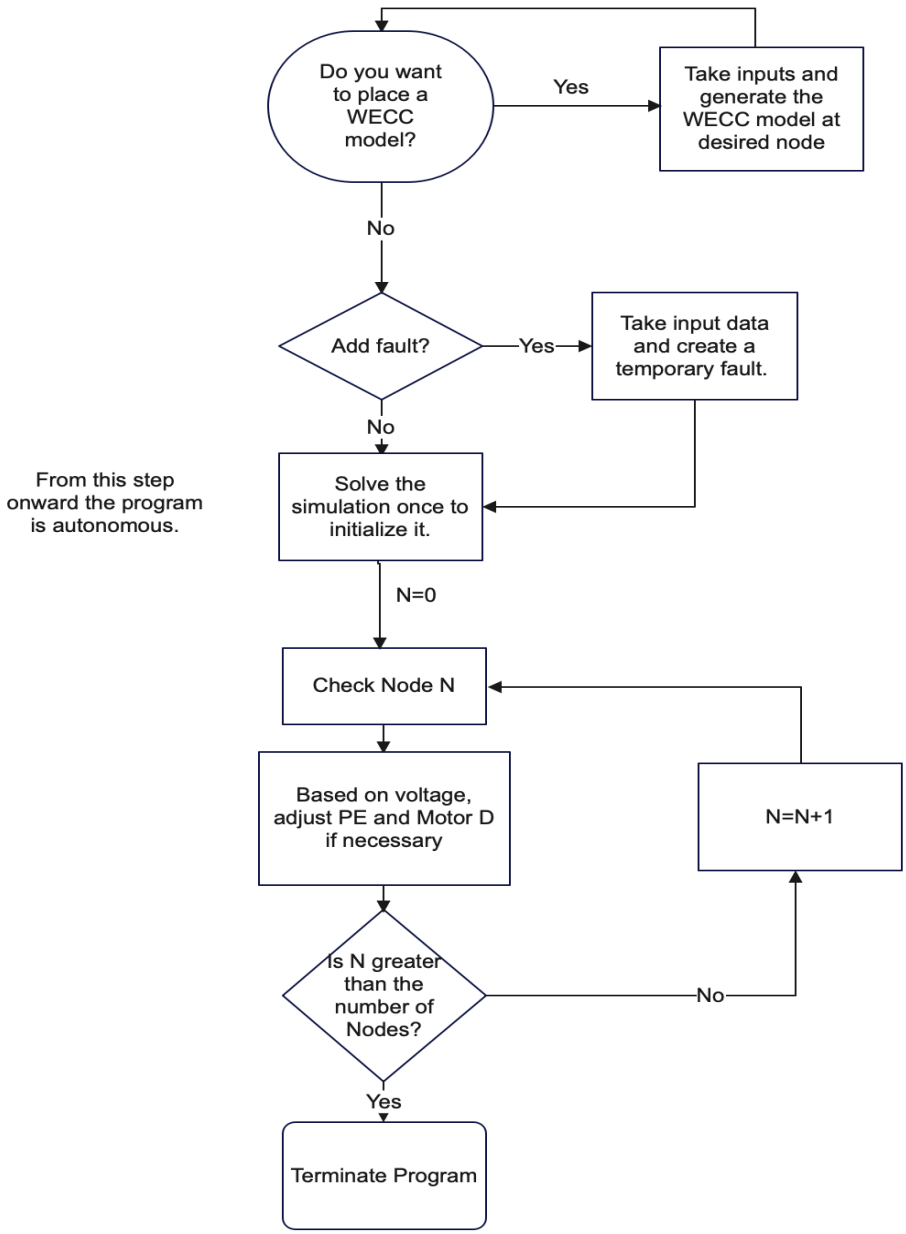

Figure 7. Python Programming Logic

The result is a program that can inject a WECC model anywhere in the system and run a simulation showing results of that placement. As part of the Start phase the program activates the 13 node test feeder file. Then it adds the WECC components as necessary, before finally executing the program and running the simulation.

5.0 Conclusion

The challenge of distribution level dynamic simulations is becoming more prevalent, and to that end I have developed a prototype that can successfully take inputs and generate a composite load model. This is generated at the nodal level, allowing for a much higher level of fidelity than previous iterations. The existence of this program, and the source code provided, allows other researchers to utilize it as a base with which to start their own investigations.

With that said, there are limitations to the program. First is that it has a narrow scope and application at present. This is a first step into distribution dynamic modeling, and so in practice any other studies would likely wish to modify the program to make it more suited to their own needs. The second limitation is that, as was covered earlier, information downstream of the transformer is limited. While that makes this kind of program appealing to run, as it has a higher fidelity than previous models, it also makes it difficult to confirm. While the results have shown some behaviors we would expect, it is hard to say how accurate the responses are without recreating a real system event. This is extremely difficult to do even by a utility, and was therefore outside the scope of this project. However, this project opens the doors for other researchers in several areas.

First, we kept the scope of this project confined to residential loads. Even then, the load shape was summarized and estimated to create a general case. Future iterations could add other segments, including Commercial, Industrial and Agricultural. Even further down the line it could be possible to add this data as an arbitrary case, and create a new set of data and rules of association using user specified inputs. Additionally, it should also be possible to include load shape data that

accounts for various factors, such as time of day, time of year and even breaking it down by other factors such as local climate or weather.

Second, we could expand this to be able to generate the nodes themselves rather than requiring a pre-made system. This would require narrowing in on the most important aspects of the system. How much detail should be taken as an input? Can that level of detailed be filtered by the user? If so, what should the base assumptions be for optional information?

Third, looking at creating higher fidelity models could be of interest. Although this model can generate WECC models at the node level, in practice most utilities can only 'see' at the substation transformer. There is feedback on occasion, but turning that data into a more detailed system would create a better opportunity for this simulation.

Fourth, this could be modified to look at more stable systems. Seeing how a house might respond to time-of-use pricing given pre-defined parameters should be possible with the COM interface.

Fifth, and this is an area of ongoing research, but adding and expanding to the Dynamic Library in OpenDSS which is currently limited in nature. Although sufficient for this project, there is a lot of room for expansion on this front.

Finally, it could be turned into a package that can be downloaded. Right now, the set up required to run this program requires knowledge of both Python and OpenDSS, but making it more accessible to design professionals would be a potential area of interest.

Appendix A Source Code: Python

Note: Source Code will not just run, it requires setup.

```
1 import py_dss_interface
2 import os
3 import pathlib
li
5
6 dss_file = r"C:\Users\Batman\Desktop\Thesis Code\Test
TEXT\Mod13NodeTest.dss" # Filepath for the opendss file
, contained in a variable
7 dss = py_dss_interface.DSSDLL(r"C:\Program Files\
OpenDSS")
8 dss.text(f"compile [{dss_file}]") # compile dss_file by
running the compile command through opendss.
9
10
11 #INITIALIZATIONS AND SETTING FLAGS
12 k=0
13 Fault1=0
lli kV=0
15 Total=0
16 monitor=0
17 Volt=0
18 dss.text(f"New Generator.670 Phases=l Bus1=670.1 kV=li.
16 kVA=-1 H=.5 Model=6 Usermodel=indmach012a conn=
Delta\nN userdata = (Rs=.02 Xs=l.8 Xdp=.12 Xdpp=.10li)\n
II)
19 timer=[0]*1000 #Initializing for multi loads
20 PowerElec=[0]*1000
21 VD=[0]*1000
22 RatedVoltage=[0]*1000
23
2li
25 i = float(input('Do you want to add one or more
composite loads? l=yes 0=no\n'))
26 while(i>0):
27 #Take node infromation
28 Node= float(input('Which node will this load be
placed at? Include .1 .2 and/or .3 for phase placement.
\n'))
Page 1 of?
File - C:\WECC Python 1.0\Thesis.py
29 kV= float(input('Define the voltage at the node in
kV\n'))
30 kva_res = float(input('Define the size of your load
in kVA\n'))
31 Conn=input('Delta or Wye Connection?\n')
32 kva_com = 0 #Commercial modifier. Included
initially but removed from scope.
33 #Values are from rules of assosciation.
```

```
3~ A= 0 * kva res+ .1239 * kva_com
35 B = .068 * kva_res + .1272 * kva com
36 C = .06~ * kva res+ .03 * kva com
37 D = .118 * kva res+ .0192 * kva com
38 Pe= .2~5 * kva_res + .1697 * kva_com # unused
39 z = .~15 * kva_res + .23 * kva_com
~0 k=k+l #tracks which load this is
~1 Total=k #tracks to total number of composite loads
~2
~3 #Initilaize Generators at the node
~~ dss.text(f"New Generator.B{k} Phases=l Busl={Node}
kV={kV} kVA=-{B} H=.5 Model=6 Usermodel=indmach012a
conn={Conn}\nN userdata = (Rs=.02 Xs=l.8 Xdp=.12 Xdpp=.
10li)\n") #conn={Conn}
~5 dss.text(f"New Generator.C{k} Phases=l Busl={Node}
kV={kV} kVA=-{C} H=l Model=6 Usermodel=indmach012a
conn={Conn}\nN userdata = (Rs=.02 Xs=l.8 Xdp=.19 Xdpp=.
lli)\n11 )
~6 dss.text(f"New Generator.D{k} Phases=l Busl={Node}
kV={kV} kVA=-{D} H=.1 Model=6 Usermodel=indmach012a
conn={Conn}\nN userdata = (Rs=.02 Xs=l.8 Xdp=.19 Xdpp=.
lli)\n11 )
~7 dss.text(f"New Load.Z{k} Phases=l Busl={Node}
kV={kV} kva={z} Model=l conn={Conn}")
~8 dss.text(f"New Load.Pe{k} Phases=l Busl={Node}
kV={kV} kva={Pe} Model=l conn={Conn}")
~9 dss.text(f"new monitor.Pow_{k} element=Generator.D{
k}")
50 i = float(input('Do you want to add an additional
load? l=yes 0=no\n')) #Can be looped any number of
Page 2 of?
File - C:\WECC Python 1.0\Thesis.py
50 times by responding with a nonzero number.
51 monitor=l #Sets flag so results are displayed later
52 timer[k]=0 #sets a timer value
53 PowerElec[k]=Pe #Stores Pe total value for later
5~ RatedVoltage[k]=kV #Stores rated voltage for later
55
56 #Optional Fault Loop. Note: Not repeatable, though
could easily become so.
57 Faultl=float(input('Do you want to simulate a fault? 1=
yes 0=no\n'))
58 if(Fault1>0):
59 Fault_Bus=float(input('Where should the fault occur
? This will be a one phase fault.\n'))
60 Fault_Time=float(input('How long into the
simulation do you want it to activate?\n'))
61 Fault_Length=float(input('And how long will it last
?\n'))
62 Fault_Length=int(Fault_Length*100000) #Scaled up
for later
63 dss.text(f"new Fault.1 busl={Fault_Bus} phases=l R
=.00001 ONtime={Fault_Time} temporary=yes")
6~ Fault_Length=Fault_Length+Fault_Time*100000 #Scaled
up for later. Addition is due to how ONtime operates.
65
66
67 #Solve existing system for the first time.
```

```
68 dss.solution_solve()
69 dss.text(f"solve mode=dynamics stepsize=.0000111 )
70 dss.text(f"solve number=l")
71 dss.text(f"new monitor.Pow_Feeder element=Generator.670
II)
72 step=float(input('How long should the dynamic
simulation last? A few seconds is considered long.\n'
))/.00001
73
7~ #more initializations
75 count=int(0)
76 L=100
Page 3 of?
File - C:\WECC Python 1.0\Thesis.py
77 kV=kV*577.3
78 R=10000 #Regular R, assigns a large resistance to the
fault to remove it.
79 #R=.00001 #Alternate R for a slower fault recovery
80 frcel = .75
81 stall=0
82 timers=0
83 k=0
8li outer=0
85 inner=0
86 wait=0
87 trip=0
88 Vmink=[100]*1000 #Needs to be arbitrarily high to
avoid an issue when initializing.
89
90
91 #This loop does 3 things: 1) Controls Motor D 2)
Controls PE and 3) Controls the Fault
92 while(L<step): #Counter based solution. Compares
input to counter value L.
93 dss.text(f11solve number=l11 ) #Simulation moves
forward exactly once
9li k=0 #The same counter from before, indicates which
node we are looking at.
95
96 while(Total>k): #Total number of composite loads,
run until each has been observed.
97 #Iterates k before using k. This is done
above too, so k[B] is actually 0.
98 k = k + 1
99
100
101
102
103
10li
105
}")
#Initating Values
Vdl = .9*RatedVoltage[k]/1.732
Vd2 = Vdl * .9
dss.circuit_set_active_element(f11 Generator.D{k
VD= dss.cktelement_voltages_mag_ang()
dss.circuit_set_active_bus(f1167011 )
```

```
Base=dss.bus_voltages()
Page 4 of?
File - C:\WECC Python 1.0\Thesis.py
106 MotorD=-VD[0]/Base[~]
107 V=VD[0]*.001
108 #This next section is for Power Electronics
controls.
109 Vmin=Vmink[k] #This is important for multiple
composite loads.
110 if (V<Vmin): #If Vmin isn't initialized high
111
112
113
11~
115
116
117
118
119
120
enough it will be B for the first iteration and ruin
the loop.
Vmink[k] = V #This resets Vmin to V anyway
, so the starting value can be any number as long as
it's large.
if (Vmin<Vd2):
Vmink[k] = Vd2
if (V<Vd2):#All Load is tripped for V below
Vd2
Fvl = 0
elif (V<Vdl): #While Decreasing between Vd1
and Vd2
if (V<=Vmin):
Fvl = (V - Vd2) / (Vdl - Vd2)
else: #While recovering above Vmin.
partial reconnection.
Fvl = ((Vmin - Vd2) + frcel * (V -
Vmin)) / (Vdl - Vd2)
121 else:
122 if (Vmin >= Vdl): #If V has not fallen
123
12~
below Vd1
recovered
Fvl = 1.0
else: #V has been below Vd1 but has
125 Fvl = ((Vmin - Vd2) + frcel * (V -
Vmin)) / (Vdl - Vd2)
126 # print(f"{Fv1}")
127 #print(f"{V} {Vd1} {Vmin} {Vd2} {frcel}")
128 # print (f "4 ")
129 Pe= PowerElec[k] * Fvl # needs to take the
real value
Page 5 of?
File - C:\WECC Python 1.0\Thesis.py
130
131
132
```

```
133
13li
135
136
137
138
139
1li0
1li1
1li2
1li3
llili
1li5
=11811 )
#print (f "{Fv1} ")
dss.text(f 11 Edit Load.Pe{k} kVA={Pe} 11 )
#Motor D Controls
if(MotorD<.li5):
stall=l
dss.text(f 11 Edit Generator.D{k} kVA=350 11 )
#print(f"STALL ")
if(MotorD>.95):
#print(f"WORKING")
if(stall>0):
timer[k]=timer[k]+l
timers=timer[k]
if(timers>.3):
timer[k]=0
stall=0
dss.text(f 11 Edit Generator.D{k} kVA
1li6 #print(f"RESTARTED{k} ")
1li7
1li8 L = L + 1 #The Counter for the master while loop.
1li9 if(Fault1>0): #Doesn't activate if fault was not
chosen.
150 count=count+l #Whole digit counter.
151 if(count>Fault_Length): #Fault Length is a
combination of Length and Duration.
152 #R=R+.001 #This loop allows the fault to
dissapear slowly, which gives more visiblity to PE
loads.
153 dss.text(f11 Fault.l.R={R}11 )
15li
155
156 #Checks monitor
157 if(monitor>0):
158 dss.text(f 11Show monitor Pow_l 11 )
159 dss.text(f 11Plot monitor object=Pow_l channels=[l 3
] II)
160 dss.text(f 11Plot monitor object=Pow_l channels=[S 7
] II)
Page 6 of?
File - C:\WECC Python 1.0\Thesis.py
161 #dss.text(f"Show monitor Pow_0")
162 #dss.text(f"Show Monitor Pow_Feeder")
163 #dss.text(f"Plot Monitor object=Pow_Feeder")
16~ #dss.text("Show voltages LN node")
```

Appendix A.1 Code Requirements

Generated by pip freeze

colorama==0.4.6

numpy==1.24.2

pandas==1.5.3

py-dss-interface==1.0.2

python-dateutil==2.8.2

pytz==2022.7.1

six==1.16.0

You will also need OpenDSS, and you will need to enable scripts on your device, which requires administrative access to your PC.

Bibliography

[1] Z. Ma, Z. Wang, Y. Wang, R. Diao and D. Shi, "Mathematical Representation of WECC Composite Load Model," in *Journal of Modern Power Systems and Clean Energy*, vol. 8, no. 5, pp. 1015-1023, September 2020, doi: 10.35833/MPCE.2019.000296.

[2] W. W. Price, K. A. Wirgau, A. Murdoch, J. V. Mitsche, E. Vaahedi and M. El-Kady, "Load modeling for power flow and transient stability computer studies," in *IEEE Transactions on Power Systems*, vol. 3, no. 1, pp. 180-187, Feb. 1988, doi: 10.1109/59.43196.

[3] C. Concordia and S. Ihara, "Load Representation in Power System Stability Studies," in *IEEE Power Engineering Review*, vol. PER-2, no. 4, pp. 41-42, April 1982, doi: 10.1109/MPER.1982.5519422.

[4] A. Arif, Z. Wang, J. Wang, B. Mather, H. Bashualdo and D. Zhao, "Load Modeling—Review," in *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 5986-5999, Nov. 2018, doi: 10.1109/TSG.2017.2700436

[5] W. W. Price, K. A. Wirgau, A. Murdoch, J. V. Mitsche, E. Vaahedi and M. El-Kady, "Load modeling for power flow and transient stability computer studies," in *IEEE Transactions on Power Systems*, vol. 3, no. 1, pp. 180-187, Feb. 1988, doi: 10.1109/59.43196.

[6] W. W. Price, K. A. Wirgau, A. Murdoch, J. V. Mitsche, E. Vaahedi and M. El-Kady, "Load modeling for power flow and transient stability computer studies," in *IEEE Transactions on Power Systems*, vol. 3, no. 1, pp. 180-187, Feb. 1988, doi: 10.1109/59.43196.

[7] IEEE Task Force on Load Representation for Dynamic Performance, “Load Representation for Dynamic Performance Analysis,” *IEEE Transactions on Power Systems*, vol.8, no.2, May 1993, pp.472-482.

[8] July, 2018. RTDS Technologies, “IEEE 13 Bus Feeder”. Retrieved from: http://images.shoutwiki.com/mindworks/7/7e/IEEE_13_Bus_Power_System.pdf

[9] Z. Ma, B. Cui, Z. Wang and D. Zhao, "Parameter Reduction of Composite Load Model Using Active Subspace Method," in *IEEE Transactions on Power Systems*, vol. 36, no. 6, pp. 5441-5452, Nov. 2021, doi: 10.1109/TPWRS.2021.3078671.

[10] A. Argüello, W. L. Cunha, T. R. Ricciardi, R. Torquato and W. Freitas, "Dynamic Modeling in OpenDSS: An Implementation Sequence for Object Pascal," *2018 IEEE Power & Energy Society General Meeting (PESGM)*, Portland, OR, USA, 2018, pp. 1-5, doi: 10.1109/PESGM.2018.8586532.

[11] P. Pourbeik, R. J. Koessler and B. Ray, "Addressing voltage stability related reliability challenges of San Francisco Bay Area with a comprehensive reactive analysis," *2003 IEEE Power Engineering Society General Meeting (IEEE Cat. No.03CH37491)*, Toronto, ON, Canada, 2003, pp. 2634-2639 Vol. 4, doi: 10.1109/PES.2003.1271063.

[12] M. J. Reno, J. Deboever and B. Mather, "Motivation and requirements for quasi-static time series (QSTS) for distribution system analysis," *2017 IEEE Power & Energy Society General Meeting*, Chicago, IL, USA, 2017, pp. 1-5, doi: 10.1109/PESGM.2017.8274703.

[13] L. Tang *et al*., "Analysis of Quasi-steady-state Sensitivity Matrix Based on DC Power Flow," *2022 7th Asia Conference on Power and Electrical Engineering (ACPEE)*, Hangzhou, China, 2022, pp. 459-463, doi: 10.1109/ACPEE53904.2022.9783742.

[14] June 12, 2012. WECC, "WECC MVWG Load Model Report ver 1.0" Retrieved from: https://www.wecc.org/Reliability/WECC%20MVWG%20Load%20Model%20Report%20ver%201%200.pdf

[15] Ecotope. "2011 Residential Building Stock Assessment". Retrieved from: http://ecotope.com/project/2011-residential-building-stock-assessment/

[16] April 5, 2021. Green, I. "Behind the Meter DER Representation in Composite Load Model" Retrieved from: https://www.wecc.org/Administrative/Green%20-%20BTM%20DER%20Representation%20in%20CMPLDWG.pdf

[17] January 2010. J. Keller, B. Kroposki. "Understanding Fault Characteristics of Inverter-Based Distributed Energy Resources". Retrieved from: https://www.osti.gov/biblio/971441

[18] December 9, 2019. WECC. "Solar Voltaic Power Plant Modeling and Validation Guide" Retrieved from: https://www.wecc.org/Reliability/Solar%20PV%20Plant%20Modeling%20and%20Validation%20Guidline.pdf

[19] March 23, 2021. N. Carnovale. "Fault Detection in Inverter-Based Microgrids Utilizing a Nonlinear Observer"

[20] August 27, 2020. WECC. "Modeling Renewable Energy/Battery Eenrgy Storage System Hybrid Power Plants. Retrieved from: https://www.wecc.org/Administrative/WECC%20White%20Paper%20on%20Modeling%20Hybrid%20Power%20Plant.pdf

[21] Wen Liang, M. Liu, Fangchao Song, Wencheng Wu, Kai Zhou and Aimin Jin, "Power system dynamic economic dispatch with controllable air-conditioning load groups," *2016 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, Xi'an, 2016, pp. 962-967, doi: 10.1109/APPEEC.2016.7779637.

[22] S. M. N. Hasnaeen, S. Rahman and M. F. Uddin, "Load Scheduling of a Refrigerated Warehouse with Homogeneous Compressors Under Dynamic Pricing," *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, Dhaka, Bangladesh, 2019, pp. 1-6, doi: 10.1109/ICASERT.2019.8934558.

[23] Yizheng Xu and J. V. Milanovic, "Framework for estimation of daily variation of dynamic response of aggregate load," *IEEE PES ISGT Europe 2013*, Lyngby, Denmark, 2013, pp. 1-5, doi: 10.1109/ISGTEurope.2013.6695241.

[24] Yizheng Xu and J. V. Milanovic, "Framework for estimation of daily variation of dynamic response of aggregate load," *IEEE PES ISGT Europe 2013*, Lyngby, Denmark, 2013, pp. 1-5, doi: 10.1109/ISGTEurope.2013.6695241.

[25] March 18, 2015. P. Pourbeik. "Simple Model Specification for Battery Energy Storage System". Retrieved From: https://www.wecc.org/Reliability/REEC_C_031815_rev3%20Model%20Spec.pdf

[26] D. N. Kosterev, C. W. Taylor and W. A. Mittelstadt, "Model validation for the August 10, 1996 WSCC system outage," in *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 967-979, Aug. 1999, doi: 10.1109/59.780909.

[27] KERESTES SOURCE

[28] C. Concordia and S. Ihara, "Load Representation in Power System Stability Studies," in *IEEE Power Engineering Review*, vol. PER-2, no. 4, pp. 41-42, April 1982, doi: 10.1109/MPER.1982.5519422.

[29] A. Argüello, W. L. Cunha, T. R. Ricciardi, R. Torquato and W. Freitas, "Dynamic Modeling in OpenDSS: An Implementation Sequence for Object Pascal," *2018 IEEE Power & Energy Society General Meeting (PESGM)*, Portland, OR, USA, 2018, pp. 1-5, doi: 10.1109/PESGM.2018.8586532.

[30] January 27, 2015. "WECC Dynamic Composite Load Model (CMPLDW) Specifications. Retrieved from: http://home.engineering.iastate.edu/~jdm/ee554/WECC%20Composite%20Load%20Model%20Specifications%2001-27-2015.pdf

[31] "Load representation for dynamic performance analysis (of power systems)," in *IEEE Transactions on Power Systems*, vol. 8, no. 2, pp. 472-482, May 1993, doi: 10.1109/59.260837.

[32] Mitra, P. September 2020. "Technical Reference on the Composite Load Model" in Electrical Power Research Institute. 3420 Hillview Avenue, Palo Alto, California