6-2022

Evaluating Energy Storage Methods for the Stabilization of Rural Microgrids

Colin Pennock
Union College - Schenectady, NY

Evaluating Energy Storage Methods for the Stabilization of Rural Microgrids

Colin Pennock

ECE-499: Electrical Engineering Capstone 3

Supervised by Luke Dosiek

3/13/22

Report Summary

Microgrids are a hot topic in industry at the moment due to their ability to help integrate renewables into the national grid. However, they are mainly used in urban settings, despite the stability benefits they could provide to rural communities. This research aims to help expand microgrids into a rural context by building a streamlined, phasor-domain model of a microgrid in MATLAB-Simulink's toolbox Simscape Electrical and using it to evaluate three energy storage options: a national grid connection, battery, and flywheel. My capstone involved testing models of these three solutions by subjecting my overall microgrid to faults typical of a rural setting. The design requirements for this project were split into two parts. First were the simulation specifications for the overall model, which had to accurately simulate the critical components of a microgrid: traditional diesel generation, wind and solar generation, a connection to the national grid, electrical load, and of course some form of energy storage. Second were the performance requirements for power service and quality, which were evaluated for a variety of fault scenarios and at each important element of the microgrid model. These included measures of power factor and microgrid voltage, voltage angle, and frequency. Both simulation and performance metrics were evaluated through direct observation of simulation output through scopes. It was found that my model was an accurate portrayal of the phasor-domain, power-flow behavior of all three types of energy storage and a general microgrid. In this same domain, all three types of energy storage performed similarly and were superior to having no energy storage at all. There were some issues with the phasor domain limiting the scope of my simulation, especially with the flywheel model. These limitations did not prevent me from reaching meaningful results, and they also provide ample opportunities for future work on real-time simulation and additional energy storage types.

Table of contents

List of Figures and Tables

Introduction

As renewable energy becomes more and more common, one strategy for handling these new resources is the microgrid. Microgrids, when intentionally applied to manage distributed energy resources, are small, self-controlled grids that are capable of disconnecting from the national grid at will. Microgrids used in this way provide all sorts of stabilization, management, and cost benefits [1]. They are great for protecting important locations, such as campuses or military bases, during national grid outages as well. However, there is a second type of microgrid that is much less studied: "by-necessity" microgrids. These grids are permanently "islanded" (lacking a connection to a large grid system) by circumstance; sometimes because they are on small islands or because they are very far away from any other settlements. These grids may have some renewable projects in them, but they usually run mainly off diesel and have limited control methods. Without a national grid connection, these grids struggle a lot with electrical stability. In addition, there is limited attention given to improving these low-efficiency grids and by extension the lives of the people in these impacted communities.

The first problem is due to the fact that islanded microgrids lack the inertia provided by the massive synchronous generators found in national grid-scale power plants. Because the physical and electrical parameters of generators are so closely related, the mechanical spinning of the large rotors on grid-scale machines helps keep the power system stable. This mostly entails keeping the grid frequency at a steady level, which in turn helps maintain a consistent voltage and power factor. Microgrids with no grid connection often have to rely on much smaller generators to provide stabilization, and when combined with the integration of renewable energy and a lack of sophisticated control strategies, “by-necessity” microgrids can be easily knocked out, even by slight faults [2].

The second problem is due to a focus in research on maximizing the potential of complex and well-maintained microgrid technologies. For instance, lots of work has been done on complex grid controllers that use machine learning and/or predictive techniques [3]. There are many papers comparing battery types for perfecting the grid energy flow and maximizing savings, like in [4]. There are plenty of studies on making microgrid installations as cost-effective as possible using mathematical optimization [5]. All of this work is important, but each new method increments only slightly on improving microgrids that are already closely monitored and controlled for optimal performance. There seems to be much less work put into practical improvements for by-necessity microgrids.

My project seeks to address both of the problems discussed above by comparing different energy storage types’ ability to stabilize a simulated islanded microgrid. The first issue will be addressed because energy storage is the primary solution to grid instability. Essentially, you are using energy storage systems (ESSs) to replace the inertia not present in islanded grids. The second problem will be addressed by simulating energy storage methods that are practical to potentially implement. In addition, the model should be simple enough to simulate so as to be understood by an outsider but complex enough to provide meaningful information. With these parameters, my simulation work should help fill the gap in microgrid research.

The rest of this report is organized in the following manner. The background section will provide additional context for the motivation behind this project. Next, design specifications for my model will be given, including constraints and relevant standards. After that, design alternatives, especially the appropriate level of simulation complexity, will be discussed. The most important section follows in the final design and implementation of my project. Then, my testing process, design evaluation, and results are explained in the testing, design evaluation, and final results section. Finally, discussions, conclusions, and recommendations are given.

Background

I don't want to give the impression that no research has been done on rural microgrids. However, the research that is done on rural microgrids tends to be more on the topic of "rural electrification," which is a topic concerned with expanding electricity access into areas without any infrastructure. One common method that has been both researched and implemented as a rural electrification solution is setting up independent microgrids, as in [6]. Essentially, these grids are installed around specific sources dropped into places with no previous electricity access. This is a good strategy, but there are also a lot of possible pitfalls that I believe my research can help with. It is unfortunately somewhat common for these rural electrification systems to use stabilization methods that are far too expensive, complex, and difficult to maintain for the communities the project is set up in. For instance, pacific island electricity access projects funded by external governments will install renewable resources with fancy controllers, only for the system to break down in a few years [7]. No residents of the island were actually trained in operating the cutting-edge microgrid design, so they just end up going back to using inefficient diesel generation. My project will tie into this rural electrification issue by focusing on ensuring that islanded microgrid projects have a strong focus on stability and practicality.

It is worth spending a little time specifying what is meant by “stability” for the purposes of my project. In the power industry, stability is broken up into two components: reliability and resiliency. The former is defined as “the ability of the power system to deliver electricity in the quantity and with the quality demanded by users,” while the latter is “the ability of a system to recover and, in some cases, transform from adversity.” [8]. In reality, these definitions are not absolute, as the features that make a grid reliable often help to make it resilient as well. For my work, stability is largely synonymous with reliability, in that I am mostly concerned with keeping load serviced as much as possible. Any simulation that contains a total grid outage will be considered a failure, so recovering from an outage will not really be a focus. The main goal is to use energy storage to help the microgrid survive faults without a complete collapse.

It would also be pertinent to mention the relevant measurements when considering grid stability. These can be split up into two categories: measurements of service and measurements of quality. The former consists of metrics that are useful in determining if the grid’s load requirement is being met. The most important service metrics are real and reactive power, as well as power factor, which is the ratio of real to apparent power. The latter category consists of metrics that help determine how well grid elements are performing, and subsequently, how well load appliances will perform. Metrics here usually include grid voltage magnitude and angle as well as grid frequency. All power system metrics can be thought of with respect to their ideal “rated” values. If your power quality metrics start to drift too far from what is ideal, it may cause some critical pieces of equipment to fail or be disconnected by a breaker, which in turn will cause an outage of some scale. In short, measuring both service and quality will be critical in understanding how energy storage improves the stability of my microgrid simulation.

The framework for my microgrid model was built previously as part of my research during the summer of 2021 on simulation Union's power grid. This means that I was able to focus the bulk of my work in 499 on building the energy storage models for this project and testing them in an existing environment (see The Final Design and Implementation section for more). This carry-over from the summer is also a major reason why this project is entirely simulation-based. But isn't this at odds with my previously stated focus on the practicality and feasibility of actually implementing my proposed ESSs? Well, if this were any other engineering discipline then yes. However, in power systems, there is only one place to start and that is pure simulation. Because I am not working on a specific type of controller or converter and am focusing more on the grid level, the only way to work with my topic on a capstone project scale is to stay within the realm of simulation. I can't exactly buy a grid battery myself, so I have to simulate the grid instead. That being said, I can still make sure that I am simulating methods that would actually be easy to implement and maintain, and given more time or a continuation of this project by another person, small-scale prototypes would be a possibility in terms of advancing the research to the implementation stage. For one person for two terms of work, however, simulation offers more than enough potential for novel research while still allowing me to focus somewhat on practicality and potential implementation.

The final thing to mention about the feasibility of implementation is cost. Originally, I had "Low-cost" as an element of my project title, as an extra consideration for practicality. However, it seems that complexity is a much more critical component in building a successful and reliable microgrid, which is also something I have a lot more control over than the potential cost of various solutions. In reality, most microgrid project installation costs are government-funded. The actual costs I'll be indirectly concerned with reducing are the cost to the islanded communities when outages occur, as well as the grant money wasted on overly complex, unstable projects.

Design Specifications and Standards

Because of the simulation-only nature of my project, my design specifications can be summarized loosely by "grid values should stay as close to their rated values as possible." I have a relatively high degree of flexibility in setting the required values. For instance, Union's grid runs at 13.2 kV, so that's probably what I'll use for my model. As long as every component agrees on what the grid voltage should be, the actual number is not particularly important. Instead, what is important is how well the simulation can stick to its intended value when disturbed. That being said, there are typical values for grid parameters and plenty of relevant standards. My requirements assume that my microgrid model is subject to some electrical fault that causes a complete outage without a grid connection and is improved by the introduction of an ESS. Below are two lists of requirements. The first list consists of design requirements for the model itself, while the second gives requirements for the performance of the simulated grid.

Simulation Requirements

- My model must consider the following basic microgrid components in some form: small synchronous generators, solar and wind renewable resources, electrical load, a national grid connection point, and of course an energy storage device.
- Although it might seem obvious, the model has to be able to complete its simulations. There are three separate requirements here.

    - First, the model has to converge on initial values for each Simscape component in order for the simulation to begin at steady state. MATLAB does this through “load-flow” analysis, which essentially involves guessing the proper initial conditions repeatedly until a solution emerges that is within a specified tolerance of what the system expects the conditions to be. If this process doesn’t converge, the model will likely behave incorrectly.
    - Second, the model obviously has to be error-free and be able to complete its simulation without getting stuck.
    - Finally, I need to be able to run these simulations in a reasonable amount of time. I would say the absolute limit on how long a trial should take to simulate would be 5 minutes. I will need to run many, many trials, and anything more than 5 minutes per trial will eat up too much time. This is also dependent on the hardware I’m running MATLAB on, of course.
- Finally, as I’ve stated before, my model should be accessible and comprehensible for future use in other research.

Performance Requirements

- The first and most important requirement is that a successful simulation can never feature a complete loss of grid power. Essentially, an outage is an automatic failure.
- Sometimes, absolute outages won’t occur, but the real power supplied to a load will become very low as the power factor also decreases. In order to prevent this phenomenon, a loss of any more than 50% of a particular grid element’s rated real power will not be tolerated.

- International Electrotechnical Commission Standard IEC 61000-4-30 defines a "voltage sag" as a reduction of grid voltage by 10% or more below its rated value for anywhere between half a cycle time and one minute [9]. Most electrical equipment is capable of riding out voltage sags, so for my purposes, I'll require that my grid voltage doesn't lose 10% of its rated value for a period of more than a minute. This is the primary measure of electricity quality.
- The universal standard for the US power grid frequency is 60 Hz. There doesn't seem to be any particular published standard for this, but the 60 Hz standard is enforced by the national government. As such, I will be running all of my grid devices at 60 Hz. I'll measure frequency for the sake of completeness, but anything that causes frequency variation will likely cause simulation issues as well.
- Power factor, on the other hand, is much less universal in its regulation. Because the power factor essentially tells you how much reactive power you have present in your system, having a low power factor isn't nearly as damaging as having low grid voltage or a grid frequency too far off of 60 Hz. That being said, it is wasteful, as the reactive power cannot be used to run very much. As such, there are many standards, usually enforced on the state level, for the national grid power factor. For instance, the New York Power Authority requires connections to the national grid to run at least a 0.95 power factor to avoid fines [10]. I'll require this 0.95 value at my national grid connection point, but I'll require a steady-state power factor of 0.9 on the microgrid bus itself, as that's the rated power factor of the generator model I built this summer [11]. During a fault transient, dipping below this threshold is not an issue, as it would be with voltage. If the microgrid

settles below a power factor of 0.9, it will constitute a significant waste of energy over time.

- Although I intend to focus on a single-load model, the IEEE-1366 standard describes how outage frequency and duration indexes are measured, which could potentially be useful if I want to try and roughly calculate these indexes from my final data [12].

Constraints

The primary constraint for my project is time. Almost every individual component of my project could be its own project. For example, the simple phasor-domain wind and solar models I'm using could be expanded into full time-domain simulations of all of the inverters and power electronics required for actually connecting dc renewables to an ac grid. This simulation would be very complex, and for my purpose, this increased complexity is not worth the minimal increase in accuracy. In fact, the complexity of a full, detailed solar or PV model is such that it's very likely to prevent the overall grid model from running reliably. Ironing out all of these minor issues would easily eat up ten weeks at least, leaving no time for me to carry out my actual research. This applies to every individual component of a microgrid, so I'm constraining myself to grid-level, hour-time-scale operations to avoid getting lost in the weeds.

The other major constraint for this project is one I've mentioned briefly before; I don't want to stray into any ESSs or control methods that are so complex to maintain and operate that they end up coming full circle and destabilizing the microgrid again because the residents using the system don't know how to work with them. Luckily for me, this constraint is pretty much automatically reinforced by the time constraint. Along with making sure my methods are practical comes an accessibility constraint for the actual simulation work. I want my model to be useful for future researchers who wish to continue this research into possible prototyping and implementation stages. This means that the model should be neat, well organized, and easy to expand for future research. It should also be fairly robust, in that it isn't held just on the edge of working with very specific parameters. That way, it will be easy to manipulate in the future for new research.

Design Alternatives

The first major choice I needed to make for this project is what simulation software to use. My work from the summer was done in MATLAB's Simscape Electrical, and obviously, there was a desire to carry this over into my capstone. However, there were a few other possibilities in HOMER and CAD. The former is better suited for cost analysis and grid control studies and is easier to use than MATLAB or CAD, but it doesn't have very detailed electrical parameter read-outs. The latter would allow me to build very detailed models of individual grid components and carry out advanced load flow studies, but it would fall into the trap of being too specialized. In the end, I stuck with Simscape because it has the right balance of simplicity and rigorousness. In addition, MATLAB is used very often in industry, making cross-comparison of models easy.

The second design alternative I needed to consider was the relative level of complexity for each basic grid component. For instance, the "load" portion of the model could be dynamic and change throughout the simulation, it could be partially inductive to simulate loads like air conditioners, or it could just be a static, resistive load. For reference, Figure 1 on the next page shows a section of the model I built during the summer.

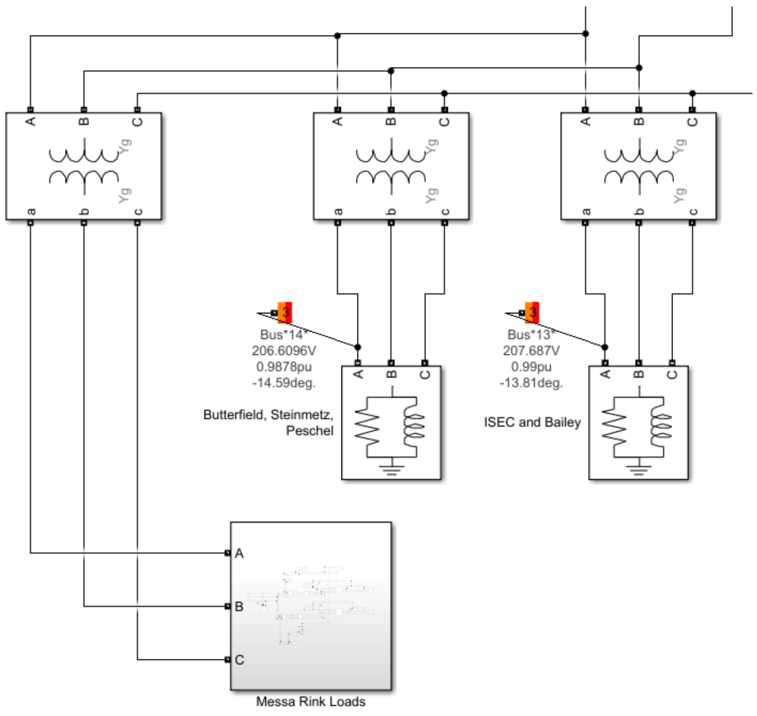


Figure 1: Example Load Types in Simscape

Each block represented a set of buildings connected to the same transformer. Note that the static loads contain inductive components. The Messa Rink subsystem contains an asynchronous motor load, as an example of a dynamic load. Is this level of detail necessary for my capstone? A similar question applies to every component of my microgrid model. See the design section for how this question was answered for each part.

The final, and most critical, design alternative I considered this term is which ESSs I should simulate. I needed to keep my practicality and time constraints in mind while still selecting ESSs that would have robust and complex enough models to provide accurate information about their stabilization performance. Each storage option is listed below, with a brief description of what I considered when making my final decision.

- **Grid Connection:** While not exactly an “energy storage” method, the most obvious solution for an islanded microgrid is to connect it to the national grid. Luckily, including this in my final design was a no-brainer, as I already had a completely functional grid connection point model leftover from the summer. Although it’s not a particularly practical solution for many isolated grids, it is an easy test case for me to simulate.
- **Generic Battery:** Simscape includes libraries that contain generic “grid battery” models, which can store and discharge power but aren’t terribly detailed. This block would simulate the typical energy storage device and would be the bare minimum to interpret the impact of ESSs.
- **Battery Types:** There are many different types of batteries: Lithium-ion, Vanadium-redox, lead-acid, and on and on. These would be interesting to simulate individually, but difficult to do without falling into the trap of excessive complexity. Many of these batteries would require proprietary models built with Simulink logic from the ground up.
- **Flywheels:** One method of compensating for the lack of inertia in microgrids is to use large spinning flywheels, which store energy as potential energy due to their spin. When a fault occurs, the flywheels can take seconds or even minutes to spin down, and their momentum can help carry the grid through the fault. This is a very interesting and relatively cheap solution, but its physical nature could make it difficult to simulate.
- **Pump Storage:** Pumped hydro storage is a common form of national-grid scale energy storage. Essentially, pump storage stores energy in the form of water at high elevation, and then lets the water flow downhill through a turbine to release the energy. While this is usually a large-scale application, there has been some research done on translating pump

storage to smaller applications [13]. Modeling this method would be interesting, but again, it has the potential to be very difficult.

In the end, in addition to choosing to work in MATLAB, I decided to definitely model the grid connection and the generic battery and to model a flywheel or pump storage system if I have time. I abandoned the idea of modeling different battery types for a reason similar to why I decided not to worry about building detailed solar or wind models; the increase in complexity is not at all worth the very slight differences in charge/discharge rate or other attributes, so I'll stick to the generic battery block. If that proves to be quickly completable, I'll move on to simulating the more interesting physically-oriented methods. I also plan to start with the simplest possible versions of each grid component model and expand on them as needed, especially required by the more complex ESS models. See the next section for a more detailed design philosophy for each component of my microgrid model.

Final Design and Implementation

Shown below in Figure 2 is a block diagram of my MATLAB-Simulink Simscape Electrical model at the highest level.

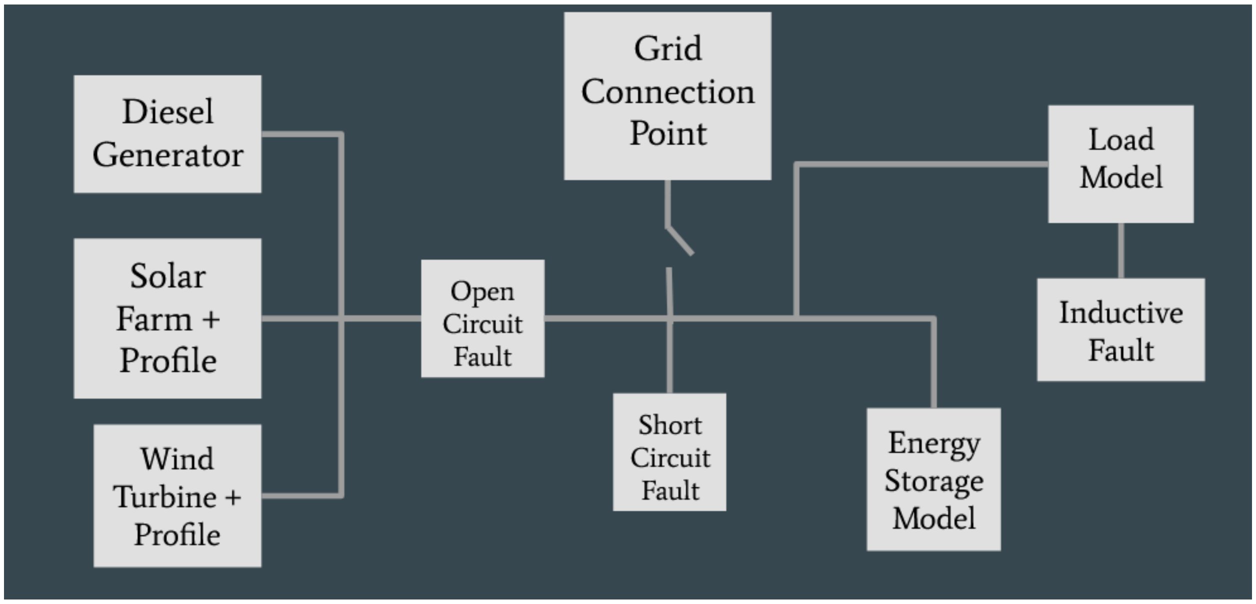


Figure 2: Simscape Microgrid Model Block Diagram

This layout was chosen because it most closely resembles a typical islanded microgrid in that it lacks a central controller. In other words, every subsystem is directly tied to the same bus, outside of individual loads. In addition, this model also ignores grid topography by lumping all possible resources into a single unit. For instance, there is only a single generator, one PV array, one wind turbine, and a single "lump" of load. This simplicity ensures that I can get meaningful data within the time constraints of my capstone.

The simulation was done in the phasor domain, meaning that the current, voltage, and power sine waves were visible as pure magnitude and angle values. Not having to model real-time sinewaves greatly lowers the memory impact of the model, allowing for the model to simulate twelve hours of microgrid activity in a matter of seconds. This is a relatively long simulation for a Simscape model, but it is made possible by the phasor domain. Simulating a twelve hour period allows me to more accurately simulate the hour-scale faults that a rural microgrid would typically face. However, the phasor approach has some drawbacks, as I will explain in the flywheel design section. The pump storage ESS mentioned before was not modeled due both to time constraints and these phasor domain restrictions.

The rest of this section will consist first of subsections explaining the detailed design for the grid connection point, generator, PV, wind, load, battery, and flywheel models. Next, a general overview of the three fault models is given. Finally, I will briefly discuss how this model meets my design requirements. Although each of these models can further be broken down into more detailed Simulink logic, primarily examining the grid-element level makes sense because that is the level at which my model produces its results. Essentially, Simscape Electrical simulates how power flows between the grid elements I listed at the start of this paragraph, so that's why I decomposed my project in this manner.

Grid Connection Point

The model that was used as a representation of my microgrid's connection point with the national grid is shown below in Figure 3.

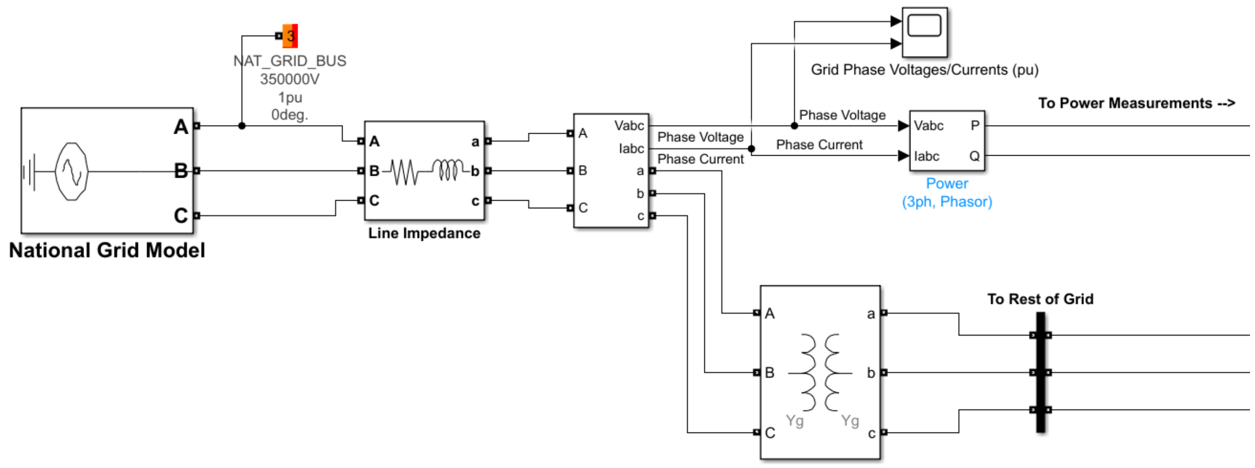


Figure 3: Simscape Grid Connection Subsystem

I built this model as part of my summer research, and it was carried directly over into my capstone. Essentially, this model is just an unlimited voltage source with some impedance to simulate transmission lines. This source, marked "National Grid Model" above, can supply infinite real and reactive power because it is designated as being on a "swing" bus by Simscape. Because of this, it can make up for whatever deficiencies in power supply the microgrid model has, just how the national grid would when the microgrid is connected. Also note the transformer, which steps the grid voltage of 350 kV down to the microgrid voltage of 13.2 kV and introduces a very small inductive load. The last thing to mention about this model is the Vabc and Iabc measurement block, which is used all over the model and is the primary way I will be converting three-phase signals to voltage, current, and power for analysis.

Small Synchronous Generator

The model I used to represent a synchronous generator is shown below in Figure 4.

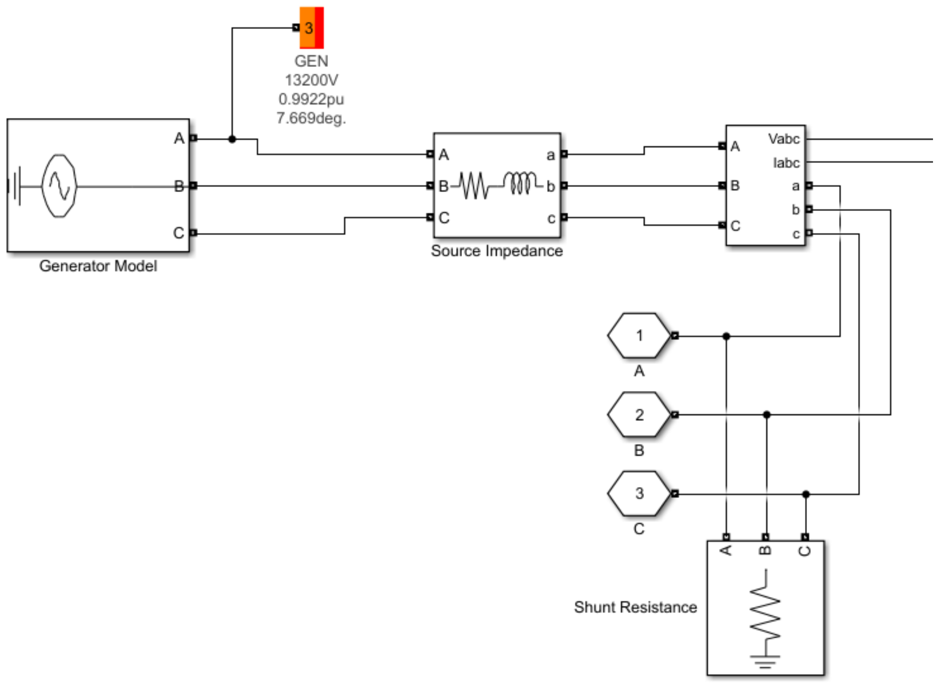


Figure 4: Simscape Generator Model

The first thing to note about this model is how simple it is, it's just a voltage source with a small (~1000 W) active load. By adjusting the impedance of the source, I can reasonably simulate the electrical characteristics of a generator. The small shunt resistance is mostly necessary for making MATLAB happy (you can't connect two sources directly together). The source impedance block simulates the internal generator resistance and inductance. This design was very easy to use and did not run into many issues with load flow. However, not being able to model the physical characteristics of a generator was a minor drawback, especially for interpreting how the generator would interact with the inertia-based flywheel ESSs. Because of this, I originally planned to substitute the more detailed generator model found in Appendix 2 as needed. This turned out to be unnecessary, as the entire phasor domain model was incapable of simulating true inertia, regardless of the generator model used. The generator in my final model was rated by the load flow tool to run at 3MW with a power factor of about 0.91. The former spec is arbitrary to have balanced power production and consumption around the microgrid, but the latter is based on the power factor of Union's cogeneration plant.

PV and Solar Profile

Normally, a PV Simscape model would require time-domain simulation of the complex system of inverters and solar cells. However, MATLAB has released a library of phasor-domain approximations of many common microgrid elements. This library, called "exampleMicrogridLibrary," includes many useful blocks and will be a critical tool for my project [14]. One such block is their phasor-domain "solar farm" block, which I used to make a simple solar panel model, shown below in Figure 5.

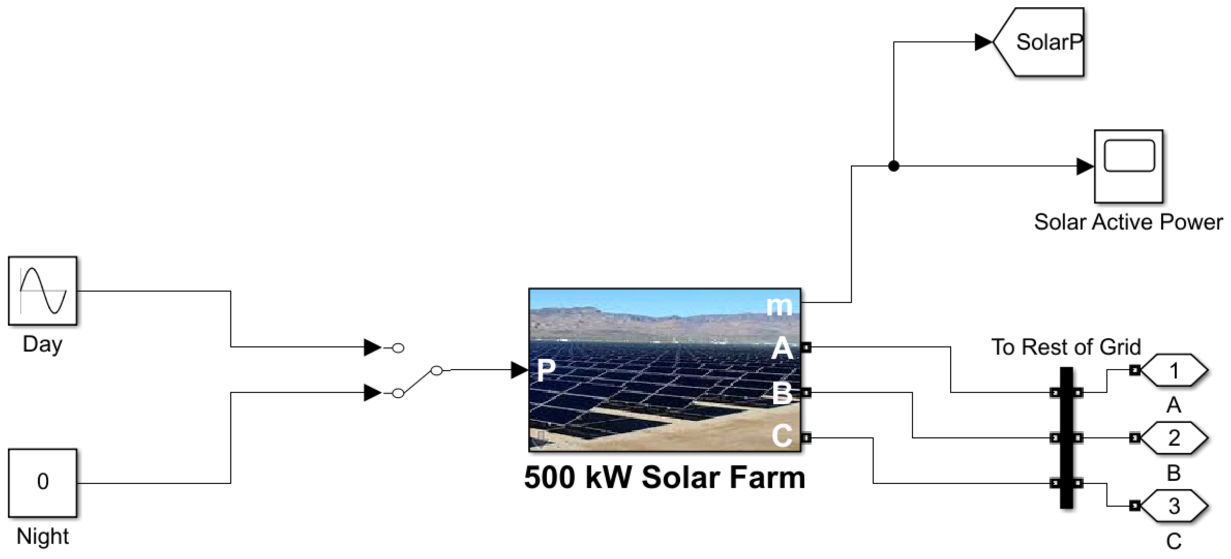


Figure 5: Simscape Phasor-Domain Solar Subsystem

This model is essentially a negative variable load that is controlled by basic Simulink logic. So, I can use it to inject power into the grid in a manner approximating a real solar panel. In the example above, I can manually choose whether the panel outputs nothing for “night” or if it outputs the positive portion of a sine wave to simulate a sunny 12 hour period. This model is very flexible and easy to manipulate, and, unlike a full-fledged time-domain model of a solar array, is very unlikely to cause load flow errors.

Wind Turbine and Profile

My combined wind profile-wind turbine phasor domain model is shown below in Figure 6.

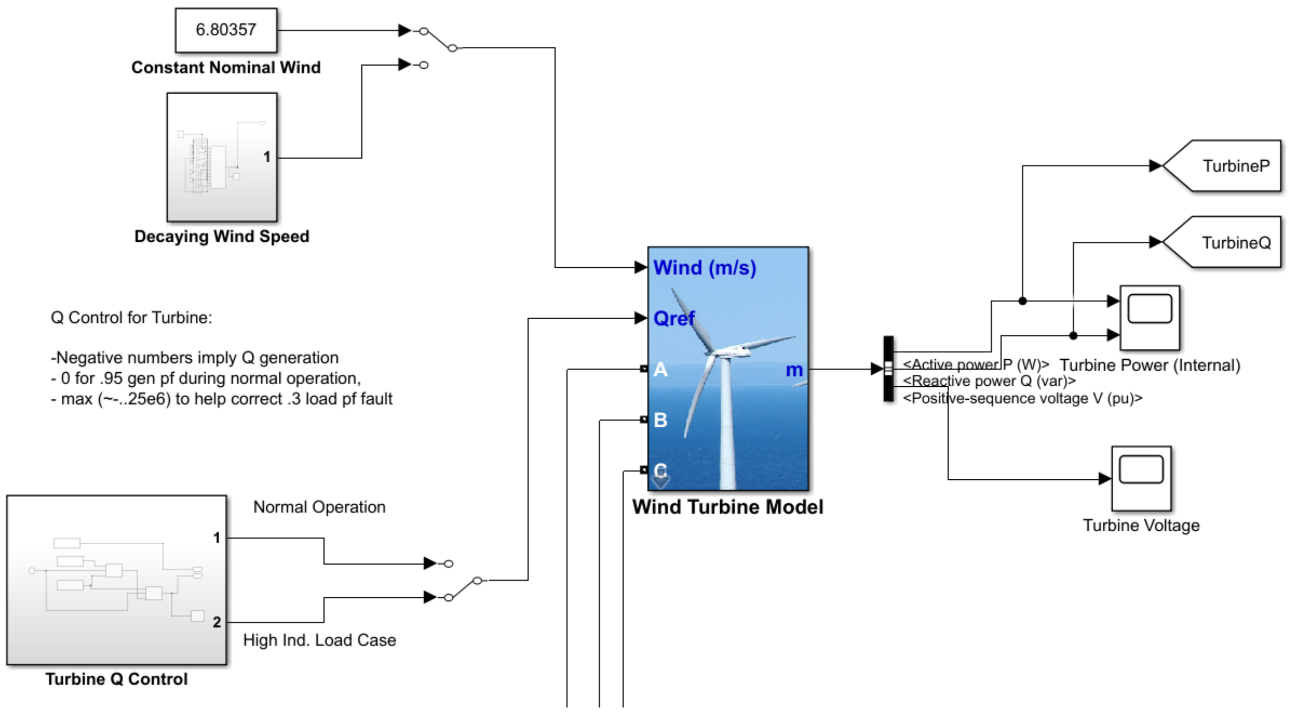


Figure 6: Simscape Phasor-Domain Wind Turbine Subsystem

This model is essentially a simplified version of the one I included in my 498 report, with added Simulink logic (mainly switches and timers) to give it the capacity to behave like a real wind turbine from a power-flow, phasor domain perspective. The main component of this model is the blue turbine model block. This simulates a 1 MW turbine using a variable load and some Simulink logic. Essentially, the block has a nominal wind speed at which it will output its nominal power after some adjustment period determined by its inertia and friction factors. At other wind speeds, it will output an appropriate proportion of its nominal power until a specified min and max speed is reached. For this model, I chose the nominal wind speed to be the average speed in Rockland, Maine, which I had leftover from research I did during the summer of 2020 (because my model is generalized, the actual number is largely irrelevant). Stats for max/min wind speed and turbine height were sourced from the typical values in [15] and [16] respectively.

There are two blocks of Simulink logic that I added to the model in order to control the "wind" and "Qref" inputs of the turbine. These were the "Decaying Wind Speed" and "Turbine Q Control" blocks respectively, as can be seen in Figure 6. The former was simply a 12-port switch, which cycled through decreasing wind values each hour, until eventually the wind speed fell below the turbine's minimum speed and its output power dropped to 0. This allowed me to model the effects of variable wind speed in my final overall model. The Q control block allowed me to roughly model the capacity of a wind turbine to provide emergency reactive power. All this block did was set the turbine's reference Q to 25% of its 1 MW nominal capacity, allowing it to output 250 kVARs of Q to help the microgrid during the heavy inductive load fault case (see the fault modeling subsection). This value was consistent with the reactive power capability of a real wind turbine found in [17]. I was able to manually choose to enable or disable this feature.

With both of these logical additions, I had a complete phasor-domain wind turbine model that worked perfectly for the purposes of my research.

Load

The final completed subsystem is shown below in Figure 7.

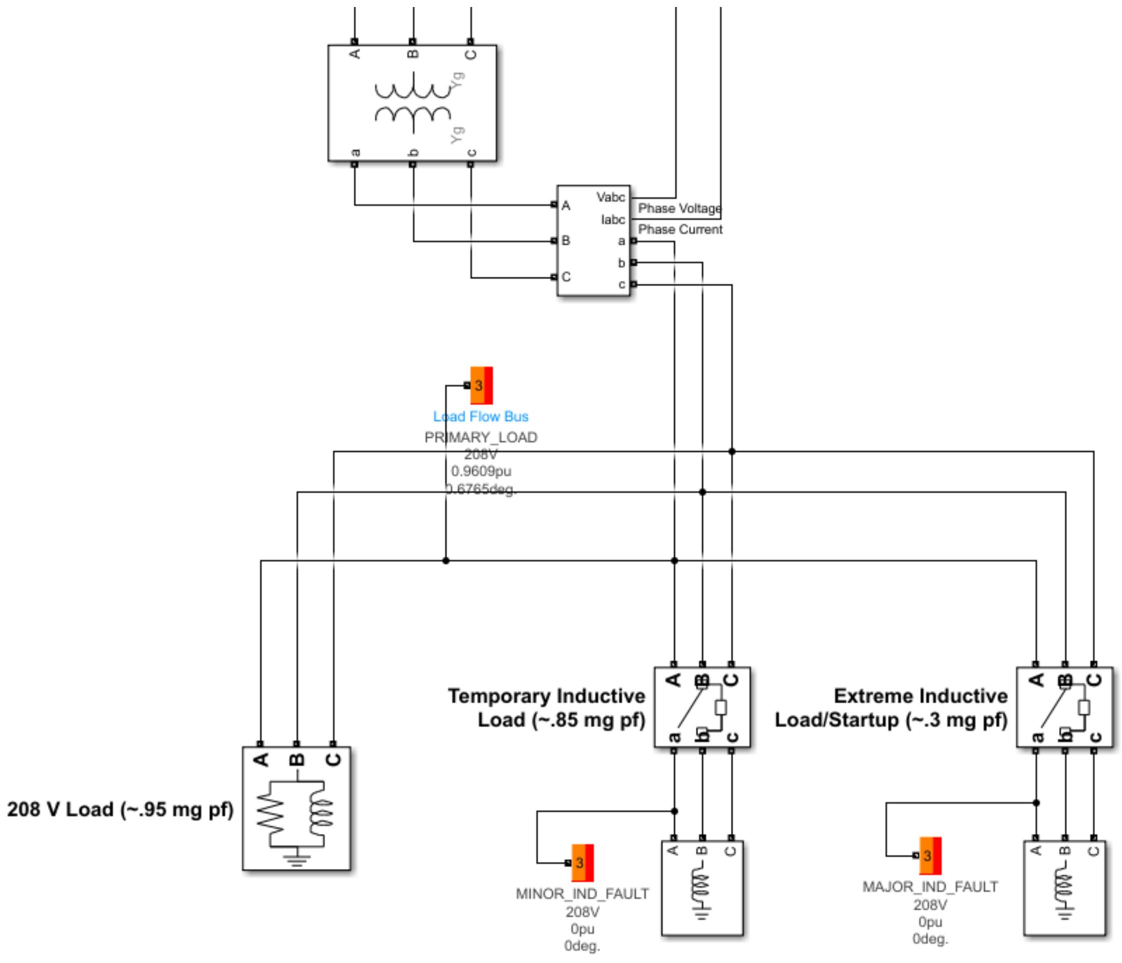


Figure 7: Simscape Load Subsystem

I've chosen to represent the load in my model as a single lump of load for a few reasons. For one, it reduces the overall simulation time. Second, my project is not concerned with things like grid topology; if the grid fails to service this single load, it is marked as a failure. This simple binary performance metric, along with power quality, will be more than enough to judge how well the ESS performs without overcomplicating things.

The other important aspect of this load model is that it considers inductive load as well as real load. While most electrical load is real or active, some machines like air conditioners or compressors require some reactive power. This requirement can be modeled as an inductor, as seen in the “208 V Load” block above. As a baseline, this 2 MVA load runs at a 0.95 power factor, with 1.9 MW of real power required and 100 kVARs of reactive power required. This power factor is fairly typical of a residential load and is a general requirement for connection to the New York power grid to avoid fines [10]. Figure 7 above also shows the additional inductive loads that can be toggled on through breakers as part of the inductive fault test (see the fault modeling subsection).

Note that the voltage is stepped down from the distribution/microgrid level of 13.2 kV to just 208 V on the load level. This is an important aspect to model, both because the transformer introduces some inductive load as it did in the grid connection model and because it accurately reflects the typical voltage range of residential loads (208V phase-phase or 120 phase-ground). The single load can easily be thought of as representing a single building, as one-building microgrids are actually fairly common. In addition, this model is easily expandable into something more closely resembling Figure 1 if things like grid topology and load shedding are studied by others in the future.

Battery

While looking for existing Simscape models of grid batteries, It became apparent that they are usually simulated in real time, as this allows the user to model inverter behavior and ac/dc conversion. So, I needed some phasor domain approximation of a battery, and I found one in the same library that I got the solar model from, in [14]. This model is the "Energy Storage (Grid-Following)" block in Figure 8 below.

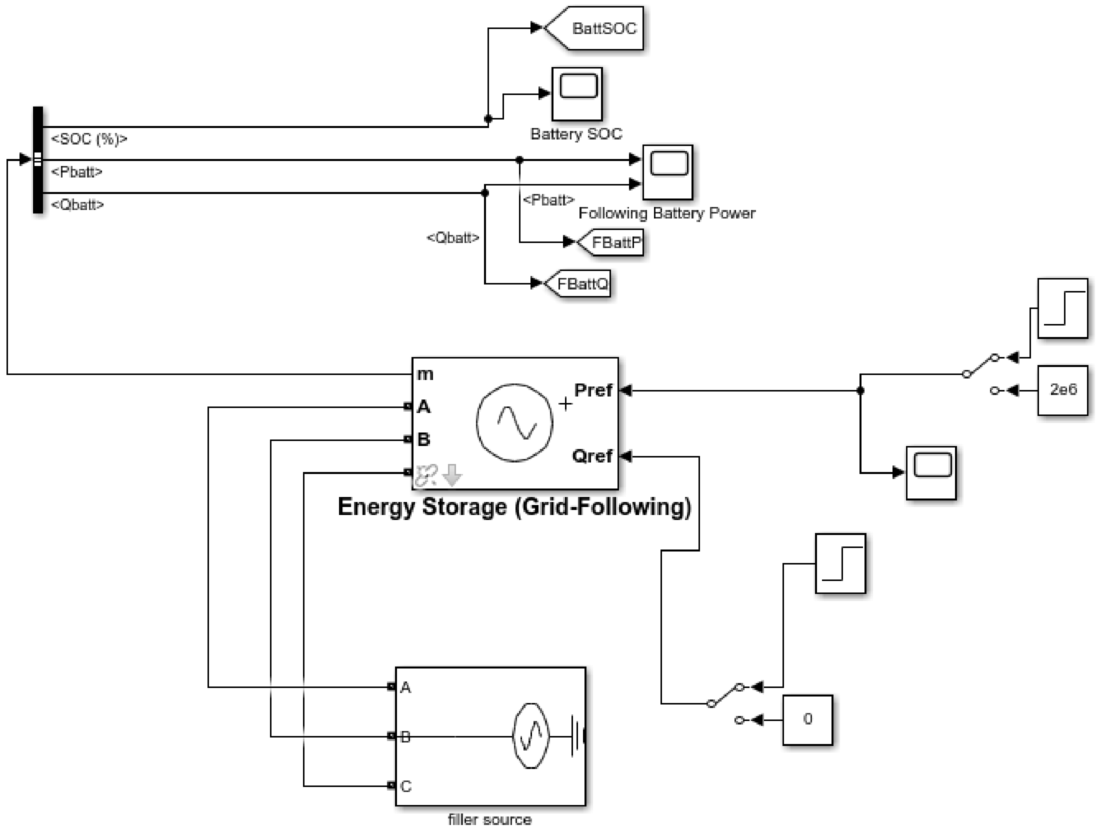


Figure 8: Simscape Grid-Following Battery Model

This phasor-domain battery model actually functioned similarly to the solar one, but with much more sophisticated internal logic. It controlled a dynamic load to simulate the behavior of a battery, charging and discharging on a set cycle, with the user being able to specify the battery capacity and instantaneous power capability. A positive load represented a charging battery, while a negative load would represent a battery discharging power onto the grid. This model allowed the user to specify the desired P and Q output of the battery through the Pref and Qref inputs. I used the formula $P^2 + Q^2 = (P_{rated})^2$ for battery inverter power capability with a desired rated power of 2.5 MW to determine that Pref should be 2 MW and Qref should be 0.5 MVARs [18].

So why is this battery model connected to a “filler source” and not to the rest of the microgrid? Well, the controlled-load model of a battery worked well for my initial tests, but when I attempted to run a load solely off the battery, I ran into a problem. MATLAB does not allow a negative load to power a positive one; it needs some kind of source to be able to run a simulation. To get around this problem, I split the battery model into two parts: the Grid-Following Model, which I’ve already shown, and the Grid-Forming Model shown below in Figure 9. While I used Simulink logic, switching between these modes would be handled by an inverter in a real battery setup.

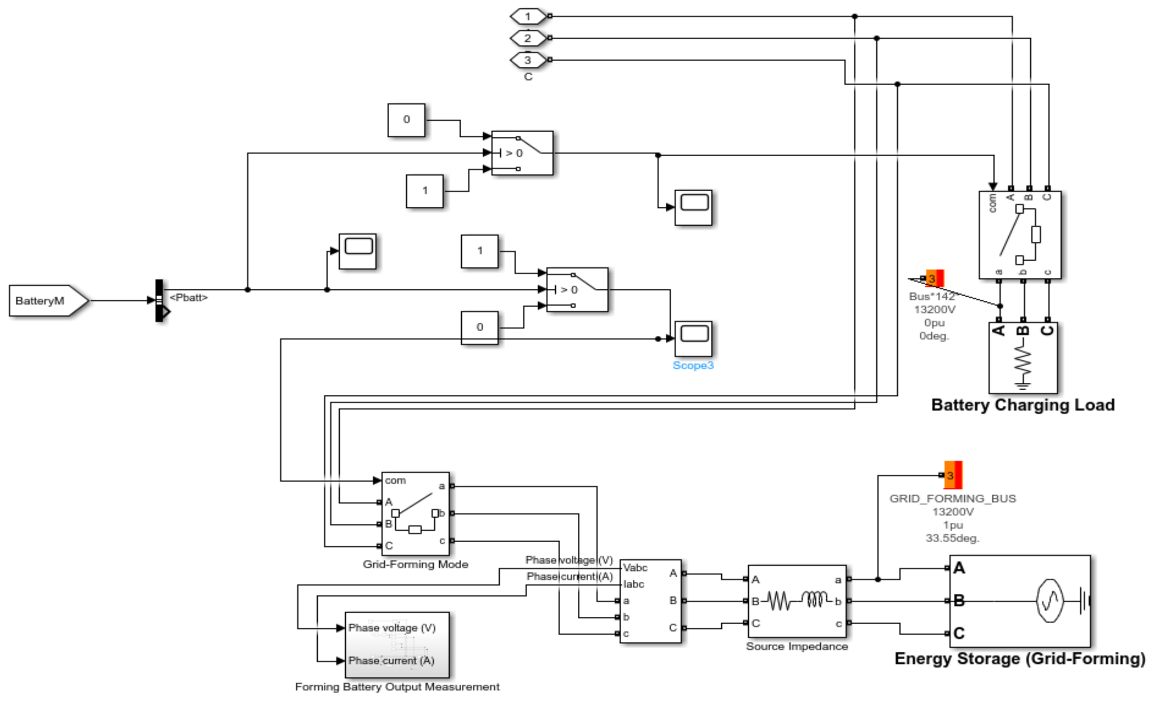


Figure 9: Simscape Grid-Forming Battery Model

The Grid Following model runs completely separate from the rest of the microgrid and is connected to its own swing source. The internal Simulink logic of the following model is used to control a voltage source (Energy Storage (Grid-Forming) in Figure 9 above) and a load (Battery Charging Load) that mimic the behavior of the battery. The load is 1.25 MW and connects only when the Grid-Following model is charging. This is because the battery charges at half its rated power capability of 2.5 MW. The source is limited by Simscape's load flow simulation to the Pref and Qref of the battery and is only connected to the grid when the Grid-Following model is discharging. The final version of the battery model was implemented after I determined that having the Grid Following model ever be connected to the microgrid model was too unreliable. Initially, I had the Grid Following model reconnect to simulate charging, but the Q from this model was leaking through the Simscape breakers, causing tiny oscillations in the microgrid which heavily extended the simulation time, making it far too long to be viable. Completely disconnecting the model and switching in a simple active charging load instead fixed this problem and gave me a fully functional phasor domain model of a battery.

Flywheel

My phasor-domain approximation of a microgrid-sized flywheel installation is shown on the next page in Figure 10.

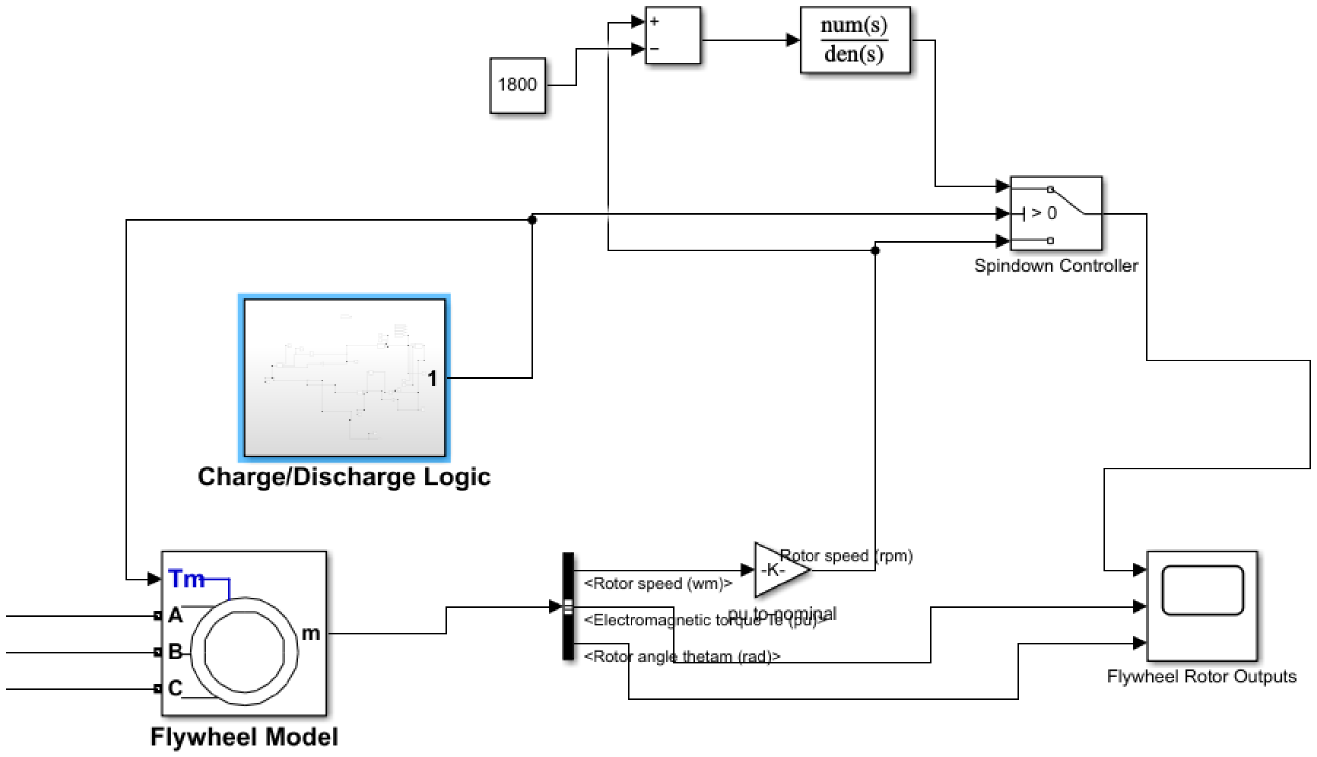


Figure 10: Simscape Phasor-Domain Approximate Flywheel Model

Like batteries, flywheels are usually simulated in real time due to their dependence on Simulink blocks (from toolboxes other than Simscape Electrical) that represent physical phenomena such as inertia. Despite this, I was able to reasonably approximate the power flow behavior of a flywheel using the model above. The main component of this model is the asynchronous machine, labeled “Flywheel Model.” This Simscape Electrical model of a machine is capable of giving speed, inertia, and torque *measurements* to the rotor output scope, but it is not capable of really translating these physical phenomena to the microgrid via their impact on frequency; it only outputs the appropriate P and Q depending on the direction and rate of its spin. So, in order to mimic the behavior of a flywheel, I had the asynchronous machine “charge” by acting as a motor (positive input torque) for a set duration and “discharge” by acting like a generator (negative input torque) for the same duration. For periods in between, the input torque to the machine was set to 0, causing the rotor to spin without generating any power. This is not the same as the spinning behavior of a true flywheel, which would spin up to speed to charge, draw essentially no power when spinning, and then spin down to release its stored power. However, because of the limited simulation of inertia available in the phasor domain, this behavior was not possible to recreate exactly. So, using basic Simulink elements like switches and delays in the "Charge/Discharge Logic" block, I managed to reconstruct the proper behavior of a flywheel on a power-flow level by controlling the input torque of an asynchronous machine, as can be seen in the results section.

The machine I used was rated at 1800 rpm with a rated power of 2 MW and the ability to supply power for 500 seconds with an inertia factor of 3000, based roughly on having twenty of the 100 kW flywheels in Table 1 of [19] installed. The time that the flywheel could output its rated power was calculated using the kinetic energy formula, $KE = 0.5I\omega^2$, with I = 3000*20 for 20 flywheels and $\omega = (2\pi * 1800)/60$. This result over time must equal the flywheel's rated power of 2 MW, as power is energy over time, so I was able to easily calculate the duration of a flywheel charge/discharge to be 532.96 seconds. This was rounded to 500 for simplicity's sake. The reason I chose to consider 20 flywheels lumped into one machine is so that the output power of the flywheel and battery models would be the same (2 MW), making them easier to compare against each other.

Some other things to mention about this flywheel are the "spindown controller" switch and surrounding logic and the way in which it charges and discharges. The former is just responsible for actually decreasing the rotor speed to 0 when the flywheel is discharging, as otherwise, the asynchronous machine would always read out a speed measurement of 1800 rpm (we assume 0 slip) regardless of the direction that it is spinning. The latter is governed by switches in the Charge/Discharge Logic block that detect particular grid faults and tell the flywheel to inject power at that time. There are three separate flywheel models, each with a different fault trigger: Generator active power less than 10 kW, load power factor of less than 0.9, and finally microgrid voltage lower than 0.9 per unit. This behavior helps distinguish the flywheel model from the battery further, as it does not simply charge and discharge on a set cycle.

Finally, it is important to note that for simulation reasons this model needed the microgrid to be grid-connected to function. This is because of the asynchronous machine approximation method and is NOT necessary for a real flywheel. Luckily, by adjusting the load flow values for the sources involved, I was able to make the grid very low priority, which means this flaw had very little impact on the results.

Overall, although this is not a perfect model of a flywheel, it gets the job done by essentially being a modified battery model with some consideration of torque, speed, and inertia. As we will see in the design evaluation section, both models ended up performing relatively similarly, which reflects their similarity and the restrictive nature of phasor domain simulation.

Fault Modeling

The three fault types I considered, shown as subsystems in Figure 2, were chosen to represent typical problems that a rural microgrid might face. The actual implementation of these faults was very simple, but they were changed frequently as part of the model's testing procedure (see design evaluation section). What follows is a brief description of the role and behavior of each fault subsystem.

The open-circuit fault is modeling a complete disconnection of all three lines between the generation (wind, solar, and diesel) and the rest of the microgrid. This would occur if a large tree fell and physically downed the lines or if your generation sources experienced some kind of mechanical failure. This fault was modeled via a three-phase breaker with an internal timer that would open and close at particular points in the simulation.

The short-circuit fault models a low-resistance connection to ground suddenly becoming available on the main microgrid bus. This can happen when trees fall and lay on the power lines, as in [20]. This ground connection results in an extra draw of real and reactive power that affects the entire grid. Simscape has a specific “3-phase fault” block that models this fault type by essentially connecting a specified number of phases to ground and to each other (either 1, 2, or all 3 phases) at a user-specified time.

Finally, the inductive fault is modeled using the breakers and inductive loads shown in Figure 7. One inductive load is 193 kVARS and represents a relatively light fault, and the other load is 960 kVARS and would be a very heavy fault. These numbers were chosen so that the overall load would have an uncorrected power factor of 0.85 and 0.3 during the respective light and heavy faults. These faults were applied temporarily via timed breakers, like the open circuit fault. The light fault was applied for longer to simulate something like a period of air conditioning, while the heavy fault was applied for longer to simulate something like a big industrial motor starting up. For more details on fault timing and testing, see the “Testing Process” section.

Meeting Requirements

My final design met all of my simulation requirements and is a very flexible model. The fact that I got any data out of it at all meant that it managed to run within a reasonable time without critical errors, despite the long simulation time. In addition, all the final models got along with MATLAB's load flow tool, allowing for the elimination of startup transients, as we will see in the results section. For more details on how both simulation and performance requirements were met by each energy storage system, see the next section.

Testing, Design Evaluation, and Final Results

This section is broken up into two subsections: the testing process section will explain how I tested my microgrid model, and the design evaluation and results section will show the results of my testing and explain how they relate to my simulation and performance requirements.

Testing Process

An important aspect of my project to discuss is how exactly I'll go about measuring my service and quality metrics for testing purposes. The primary method for measurement in MATLAB-Simulink is the scope, which simply allows me to view the value of a particular signal over time and output that data as a graph. This will be the primary method for detecting outages, as we can literally see the power drop to 0. An example of this can be seen on the next page in Figure 11, where the national grid is disconnected, causing its active power to collapse to 0.

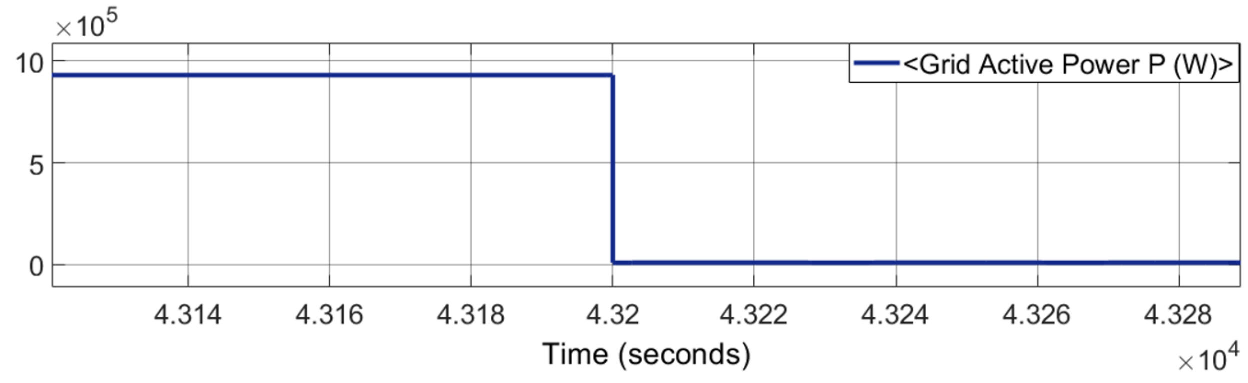


Figure 11: Simulink Power Collapse Example

Similar methods can be used to measure quality metrics like grid voltage and angle, which can be directly viewed through a scope. However, the method used to estimate microgrid frequency was somewhat more involved. The Simulink logic used to estimate frequency is shown below in Figure 12.

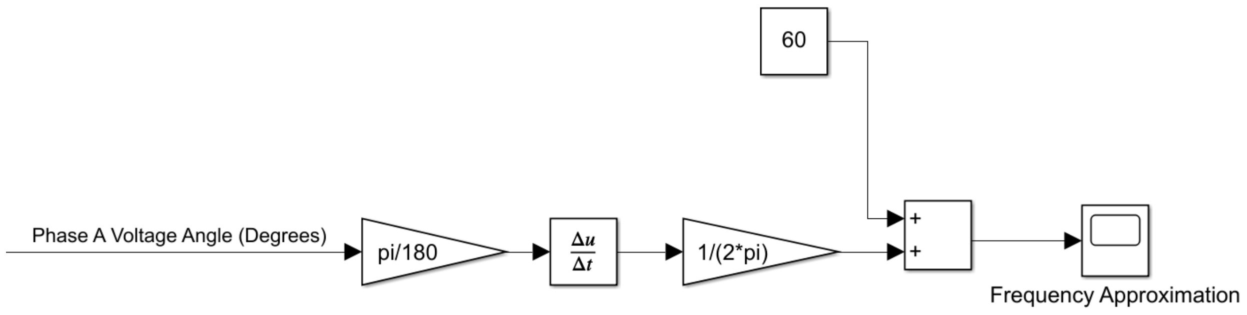


Figure 12: Simulink Microgrid Frequency Estimation Logic

This logic uses the fact that frequency can be estimated as the rate of change of the voltage (or phase) angle, as in [21]. Essentially, this takes the voltage angle from phase A of the microgrid, converts it to radians, takes its rate of change, converts from radians/sec to Hertz, and then adds whatever this deviation from zero is to the nominal frequency of the microgrid, 60 Hertz.

If one of the measurements of service or quality fails to stay at its required value (see design specifications), it will be easy to see. We can increase our ability to precisely detect unacceptable measurements using breakers. For example, if it's a critical measurement, such as power, that can't dip below its required value for even a fraction of a second without causing issues for the customer, we can set a breaker to trip at a certain power level, which will disconnect and force a complete collapse that is easily visible through a scope. This method was used mostly internally in some of the Simulink logic governing the flywheel and battery models and did not end up being a significant part of the direct testing process, which I will now explain.

With scopes placed at every major grid element, I gave each ESS an overall score based on how well they met my performance requirements for various fault scenarios. What follows is a description of my testing protocol, broken up into three parts: the exact fault scenarios, some additional renewable energy parameters, and the logic behind generating my final scores.

The following is a list of fault scenarios, each of which was tested separately for each energy storage type. The abbreviations given will be reused in summary tables of my results.

- NF = No faults occur.
- OC = Open Circuit fault. Disconnects generation from the rest of the microgrid. Lasts for 4 hours, from hour 4 to hour 8.
- 1-PS = 1-phase (Phase A) 1-ohm short circuit fault to ground, occurs on the main microgrid bus. Lasts for 6 hours, from hour 3 to hour 9.
- 2-PS = 2-phase (Phases A and B) short circuit fault. Same timing, value, and duration as above.
- 3-PS = 3-phase short circuit fault. Same timing, value, and duration as above.

- LIL = Light Inductive Load mentioned in the load design section, lasts 8 hours, from hour 2 to hour 10.
- HIL = Heavy Inductive Load also mentioned in the load design section. Lasts 15 minutes. Starts at Hour 6.
- Short OC: 300-second open circuit fault, used only in testing the flywheel. Starts at hour 6.

Some of these cases were repeated with different renewable energy settings to confirm the proper behavior of the model (see the next section for more details). These additional settings were as follows.

- CW = Constant nominal wind speed of the turbine for the entire simulation.
- DW = Wind speed decays over the course of the simulation.
- QC = Wind turbine provides 25% of its real power as Q compensation for heavy inductive load.
- D = Dark conditions.
- S = Full brightness/sunny conditions. Follows a sine wave shape.

For each energy storage type, I evaluated scopes measuring the following service and quality metrics in my model.

- PQX = Real and reactive power flow for a certain grid element. Includes power factor, which is the ratio of real power to total/apparent power. Only service performance measurement, but measured many times per ESS.
- Grid voltage/angle = Main measure of microgrid power quality. Includes an estimate of frequency, as explained previously in this section. Quality performance measurement.

- Battery SOC = Battery State of Charge. Quality performance measurement.
- FW Mechanical Metrics = Rotor speed, electrical torque, and rotor angle for the flywheel model's asynchronous machine. Quality performance measurement.

With all of that setup done, for each combination of renewable settings and faults, for each metric listed above, and for each of the four energy storage cases, I gave the corresponding scope a score based on the following criteria, which were directly related to my design specifications. The full tables of these scores can be seen in the second part of the Design Evaluation and Final Results section.

- P = Pass. Implies behavior of the particular measurement was satisfactory in all aspects and entirely fulfilled all of my relevant performance requirements.
- P* implies a pass with a minor caveat, more due to the simulation than the actual grid performance. For instance, a particular source might be supplying slightly more power than it should be capable of, but the overall power flow of the grid is still balanced. Has a numerical weight of 0.9.
- ? = Questionable. This represents a more significant problem with the trial that still might not be catastrophic. This was most frequently assigned in situations where grid elements had power factors less than 0.9, as this is not ideal but not necessarily disastrous and un-correctable. Could also represent a load losing 50% of its rated active power but maintaining a proper power factor. Has a numerical weight of 0.5.
- F = Fail. Automatically occurred if the model didn't simulate, if the power factor was very low (less than 0.5) for over a minute, if the grid voltage sagged below 0.9, or if the grid frequency deviated from 60 Hz. Has a numerical weight of 0.

- F* implies a failure that would occur even under ideal grid behavior, for example, a generator disconnecting when an open circuit fault occurs at its terminals. Has a numerical weight of 1.

Design Evaluation and Results

This section will be broken up into two parts: graphs demonstrating that each of the four energy storage cases functioned as expected on a power-flow level, and tables giving the test results for each storage type. The tables reference all of the parameters in the lists given in the previous section, and the graphs will both confirm proper model behavior and give context for how the scores were determined.

Shown on the following pages in Figures 13, 14, 15, and 16 are Simulink scopes showing the No Storage, Grid Connection, Battery, and Flywheel cases respectively. Each graph is followed by a brief description of why that particular plot confirms proper model behavior. For all four of these plots, the wind profile was set to a constant nominal wind speed, and the solar model did not output any power. This graphical method fulfills the simulation requirement of the model being streamlined and easy to understand while still complex enough to give meaningful data; visualizations are accessible but still tell the user a lot.

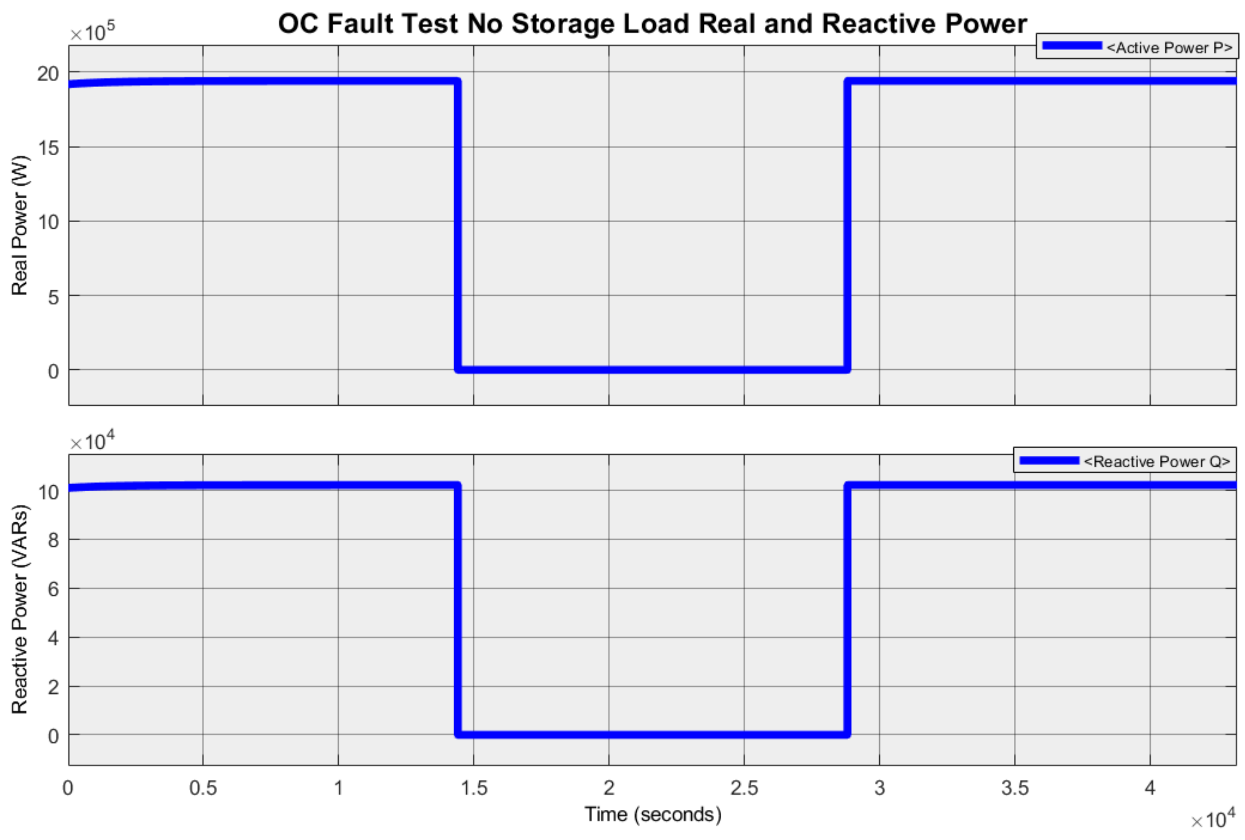


Figure 13: Four Hour Open Circuit Fault Load Power with no Storage

The above plot demonstrates the complete inability of the microgrid to handle the open circuit fault without a form of energy storage, which is exactly what we would expect. Both the real and reactive power at the load completely drop to 0 for the duration of the fault, which would represent a 4-hour outage on the part of the customer. This is obviously not acceptable and demonstrates why having some kind of energy storage in a microgrid is so important.

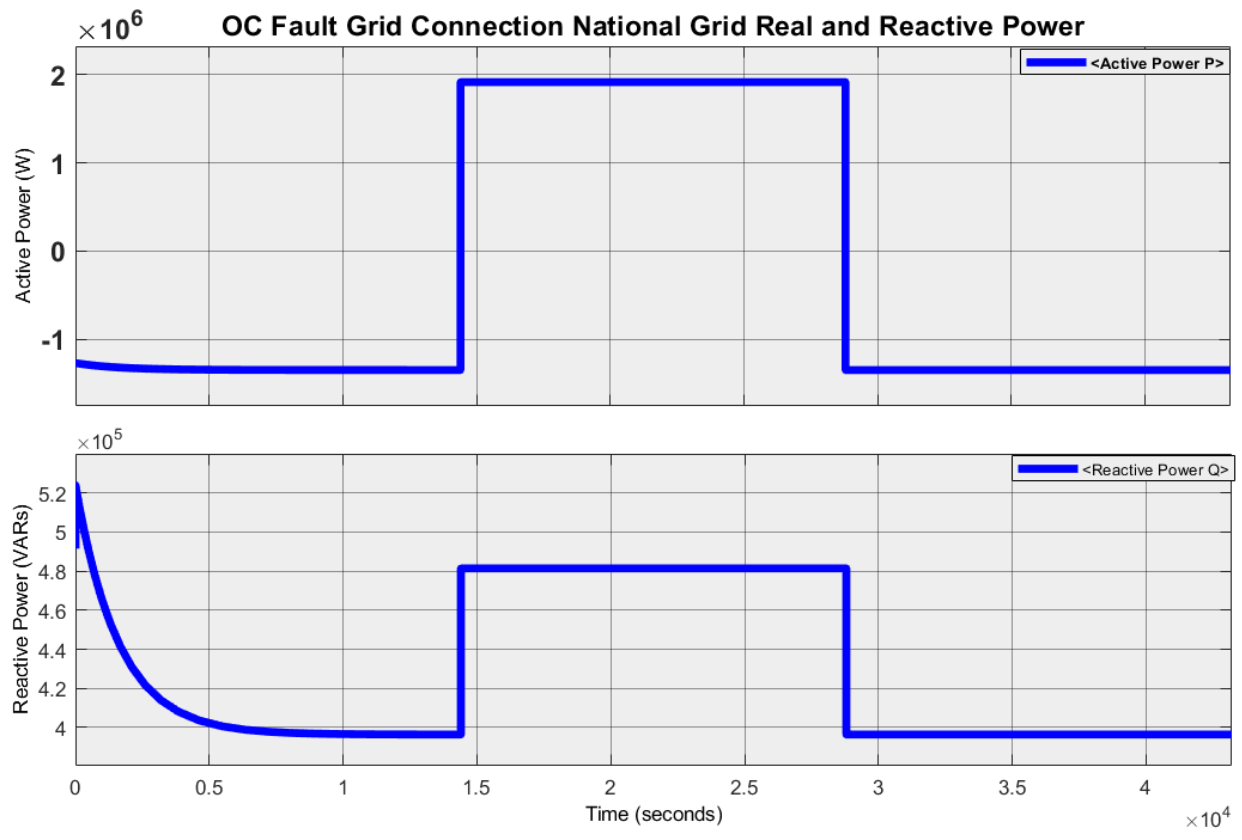


Figure 14: Four Hour Open Circuit Fault Grid Connection Grid Corrective Power

The plot above demonstrates the corrective power of having a connection to the national grid as a form of energy storage. Here, the national grid can essentially supply infinite real and reactive power, as explained before. So, when faced with the same fault as the no storage case, the grid supplies the 2 MW required to power the load. Outside of that fault, the microgrid generation has priority and the grid even absorbs the extra real power supplied by the 1MW wind turbine and the 3 MW generator (hence the negative power flow outside of the fault duration). This is the ideal micro-macro grid relationship and is typically achieved via a smart central microgrid controller. Note that the decreasing reactive power curve is due to the wind turbine, which requires some Q as it spins up to full speed at the start of the simulation.

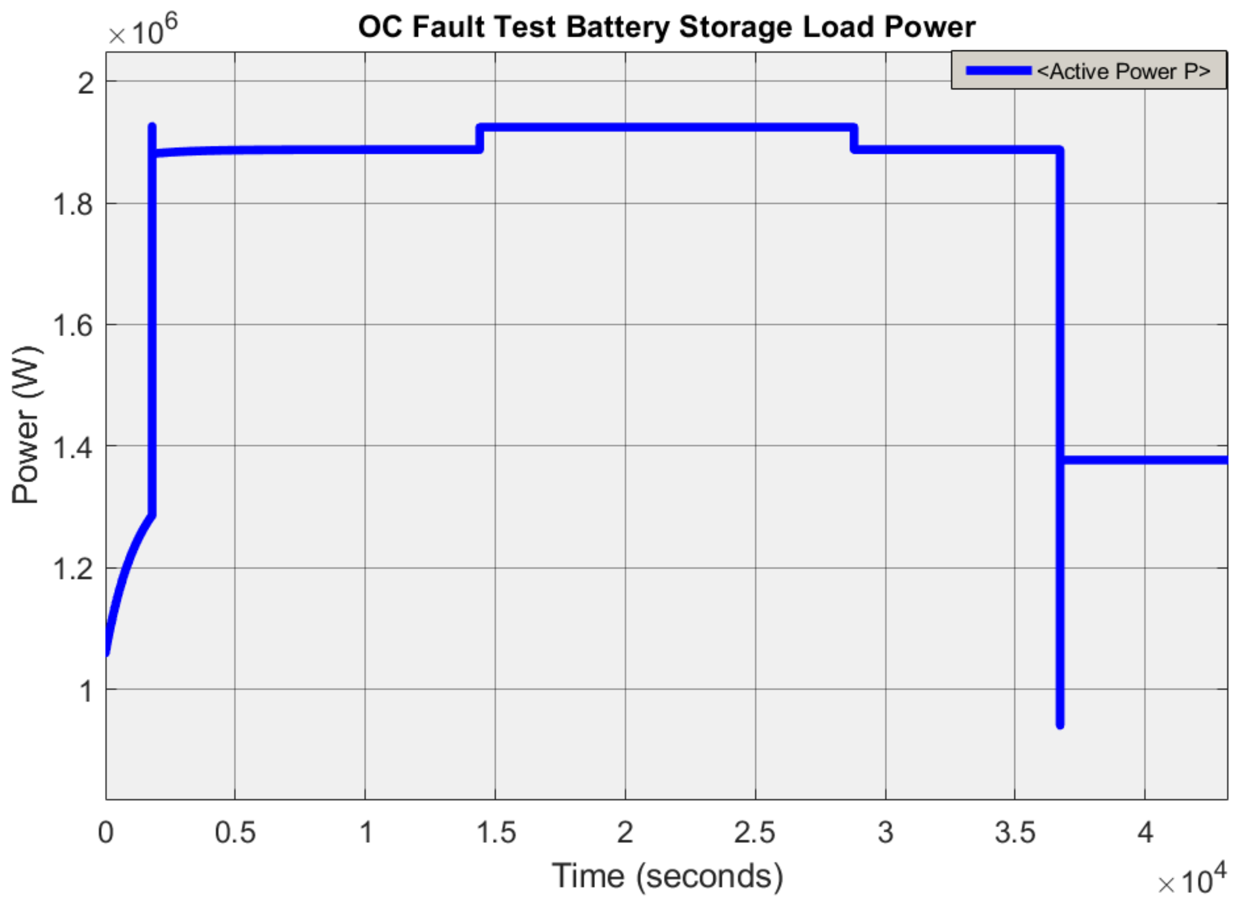


Figure 15: Four Hour Open Circuit Fault Battery Example

The microgrid that produced the plot of load active power above is also facing the same open-circuit fault, but this time the grid battery model is installed. This graph demonstrates proper battery behavior in the following manner. First, because of some background Simulink logic, it always has to start in “charging” mode for just a little bit, in this case, half an hour (this time period is arbitrary), hence the low starting load power. Then, as the battery is slowly discharging, the load is mostly running off the generator, until just before 15 kilo-seconds (hour 4) when the fault occurs. Then, the load switches to running off the battery, and the load power jumps slightly as no power is being lost to the generator’s internal resistance anymore. Then the fault ends four hours later and we go back to the generator. Finally, the battery runs out of power and starts charging, forcing the generator to power it and the load. This results in a drop in load power in the last few hours of the plot. From this one graph, we can see that every important function required by the battery design is fulfilled: charging, discharging, and carrying the grid through losses of generation.

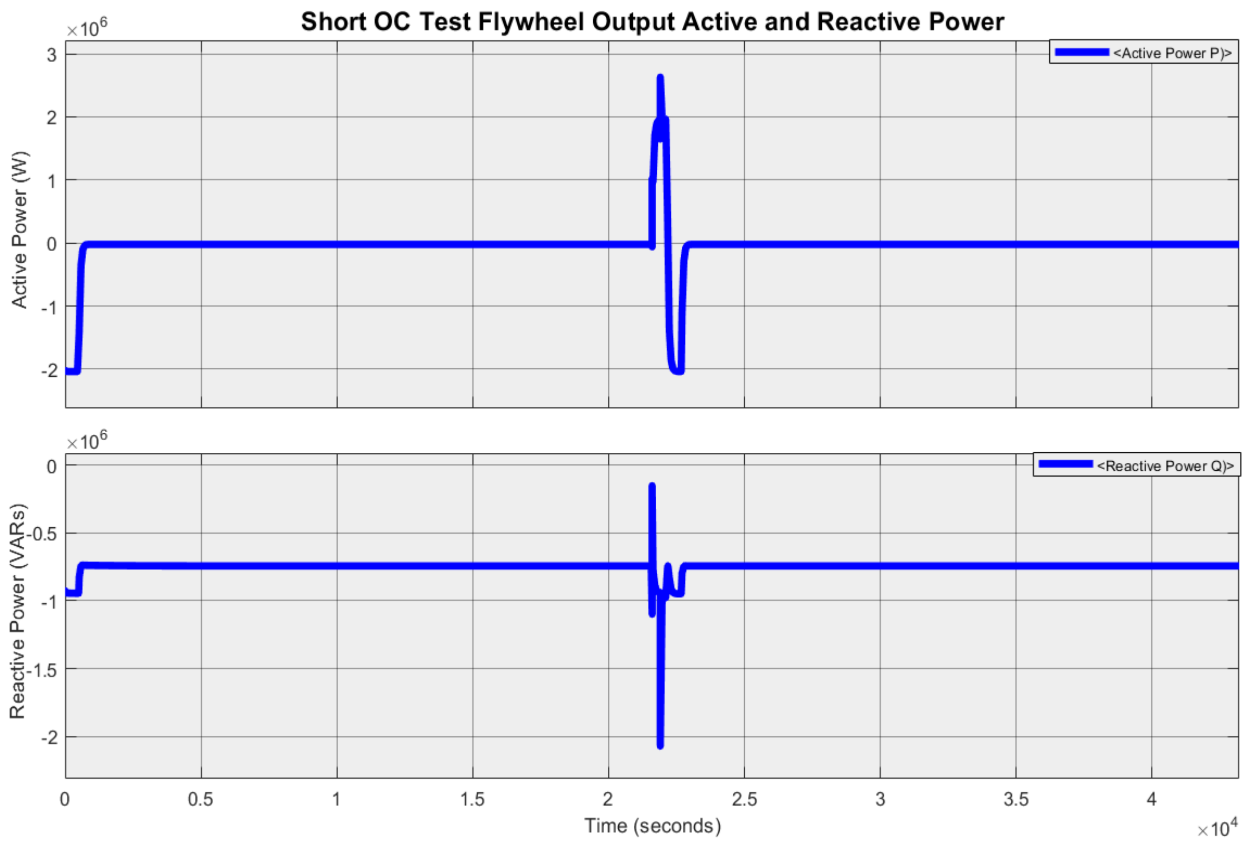


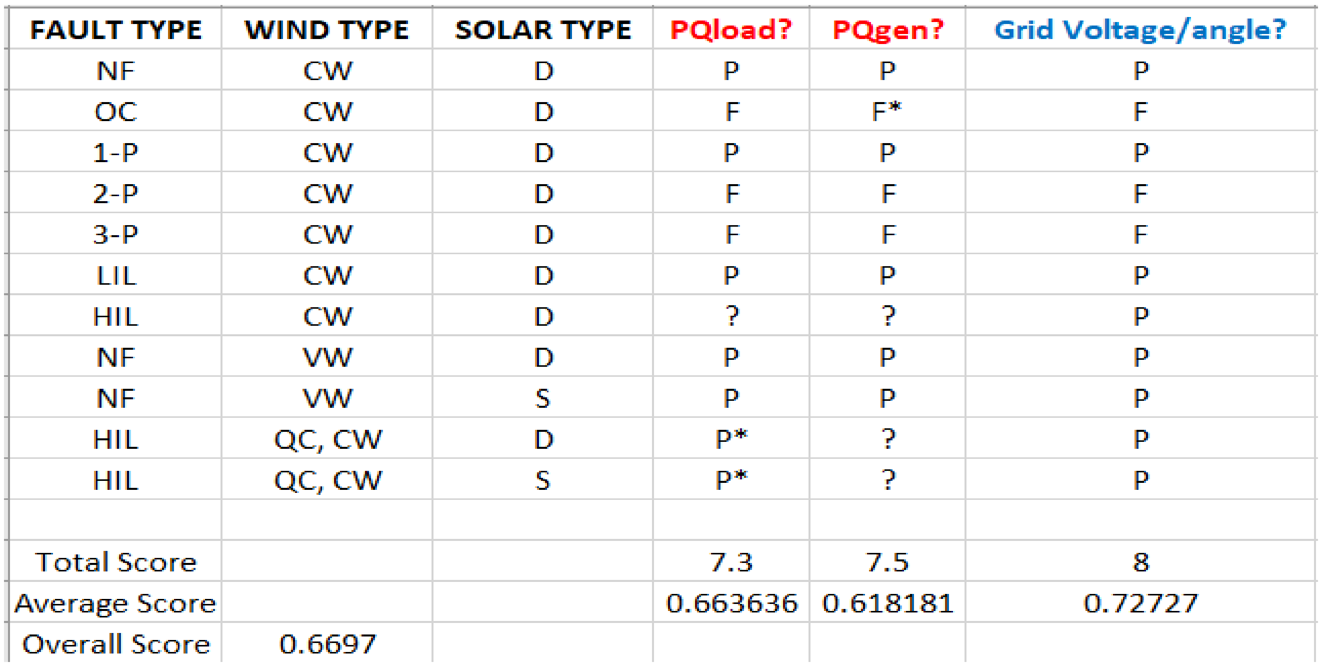
Figure 16: Five Minute Open Circuit Fault Flywheel Output Power

The final plot above shows the real and reactive power that the flywheel model is outputting during a much shorter (5 minute) version of the open circuit fault test. This graph perfectly shows both the successes and limitations of this flywheel model, as discussed in its corresponding design section. First, note that the flywheel starts by drawing its 2 MW of rated power from the microgrid for a short duration. This duration is its rated capacity time of 500 seconds of 8.33 minutes (see flywheel design section). Because the short OC fault starts at hour 6 (21600 seconds), the flywheel also discharges at that time, for its entire 8.33 minute capacity. It is triggered to do so because it detects that the generator power output drops to 0 at that time, as this plot comes from the flywheel model set to trigger on low generator active power. This carried the microgrid through a short open circuit fault. Note that the response of the flywheel is curved and somewhat gradual, which makes sense given how much inertia it has. So, even though it could technically output power for 8.33 minutes, it can only output the full required power for less time, hence the 5 minute fault to demonstrate. Once the flywheel is done discharging, it immediately spins up again, drawing power as it does. In between charging and discharging, the flywheel draws no power, as friction is ignored here.

The limitations of this model start to become apparent upon examining the reactive power output. Because I am using an asynchronous machine as an approximation, the flywheel absorbs reactive power regardless of whether it is charging or discharging. This is because it is really spinning in one direction to charge and another to discharge, not slowing down to release its energy through inertia as it would in reality. This is not perfect behavior, but it is exactly what I would expect from my design. In order to improve the accuracy of the flywheel model, it would need to be translated somehow into a real-time simulation, which would involve actually simulating the 3 phases of sine waves present on the ac microgrid. This would allow for the simulation of physical phenomena like inertia and more advanced power electronics like inverters, which would increase the accuracy of my energy storage models with respect to their real-world counterparts, especially the flywheel. For a phasor domain analysis mainly concerned with grid-level power flow, however, my flywheel model was sufficient.

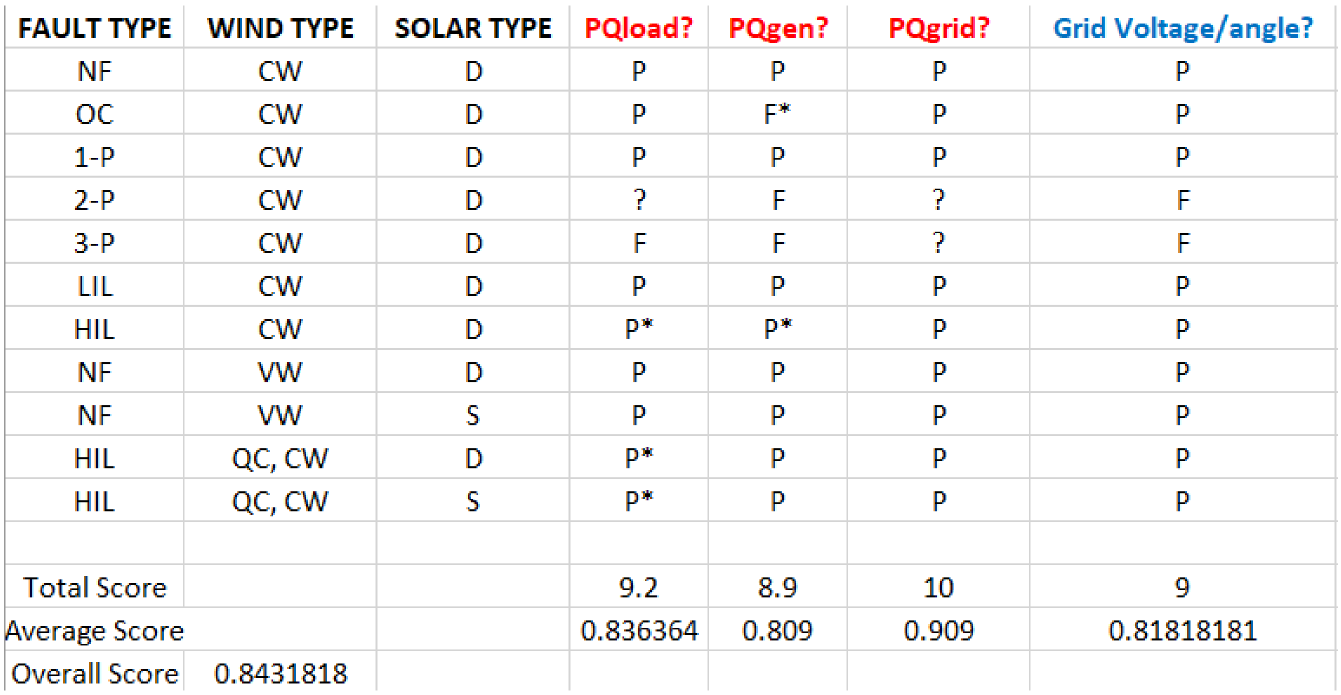
Having been convinced by graphical data that my four models behaved satisfactorily, I produced the following four scoring tables, shown below in Tables 1, 2, 3, and 4. Service metrics are shown in red while quality metrics are shown in blue. For more information and to see all of the relevant scopes/visualizations used throughout the four models to produce these tables, see Appendix 3.

Table 1: No Storage Case Testing Results



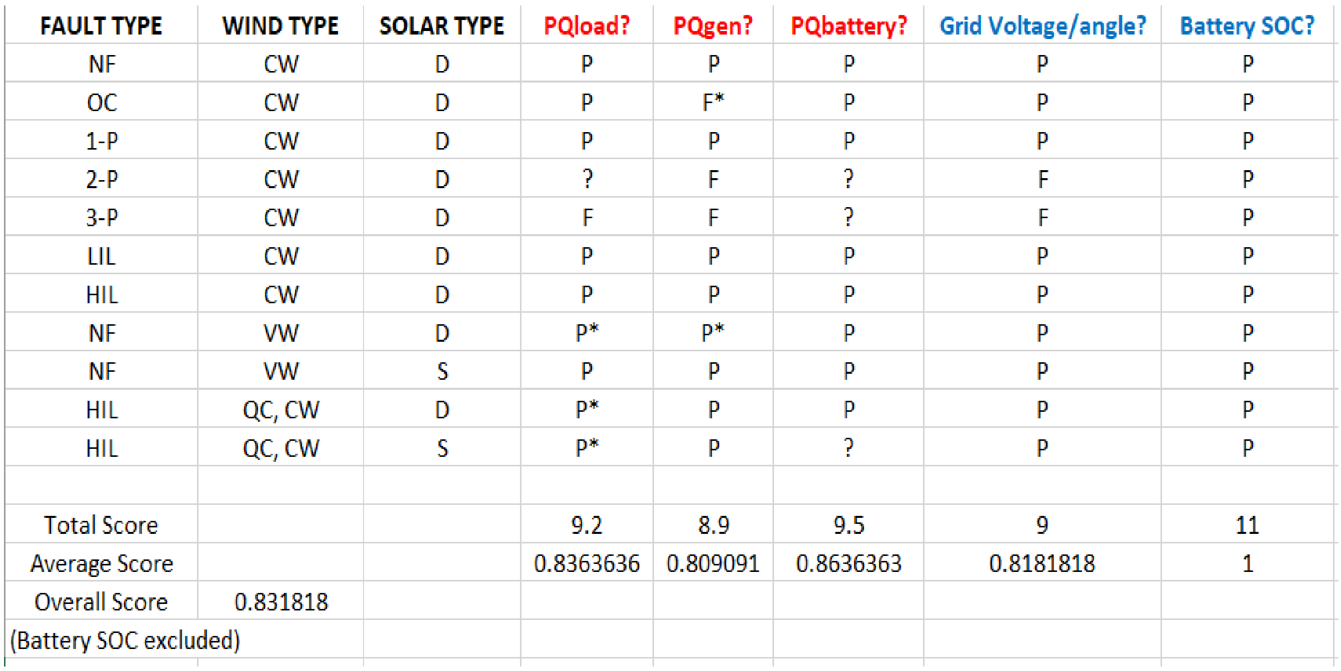
FAULT TYPE	WIND TYPE	SOLAR TYPE	PQload?	PQgen?	Grid Voltage/angle?
NF	CW	D	P	P	P
OC	CW	D	F	F*	F
1-P	CW	D	P	P	P
2-P	CW	D	F	F	F
3-P	CW	D	F	F	F
LIL	CW	D	P	P	P
HIL	CW	D	?	?	P
NF	VW	D	P	P	P
NF	VW	S	P	P	P
HIL	QC, CW	D	P*	?	P
HIL	QC, CW	S	P*	?	P
Total Score			7.3	7.5	8
Average Score			0.663636	0.618181	0.72727
Overall Score	0.6697				

Table 2: Grid Connection Case Testing Results



FAULT TYPE	WIND TYPE	SOLAR TYPE	PQload?	PQgen?	PQgrid?	Grid Voltage/angle?
NF	CW	D	P	P	P	P
OC	CW	D	P	F*	P	P
1-P	CW	D	P	P	P	P
2-P	CW	D	?	F	?	F
3-P	CW	D	F	F	?	F
LIL	CW	D	P	P	P	P
HIL	CW	D	P*	P*	P	P
NF	VW	D	P	P	P	P
NF	VW	S	P	P	P	P
HIL	QC, CW	D	P*	P	P	P
HIL	QC, CW	S	P*	P	P	P
Total Score			9.2	8.9	10	9
Average Score			0.836364	0.809	0.909	0.81818181
Overall Score	0.8431818					

Table 3: Battery Case Testing Results



FAULT TYPE	WIND TYPE	SOLAR TYPE	PQload?	PQgen?	PQbattery?	Grid Voltage/angle?	Battery SOC?
NF	CW	D	P	P	P	P	P
OC	CW	D	P	F*	P	P	P
1-P	CW	D	P	P	P	P	P
2-P	CW	D	?	F	?	F	P
3-P	CW	D	F	F	?	F	P
LIL	CW	D	P	P	P	P	P
HIL	CW	D	P	P	P	P	P
NF	VW	D	P*	P*	P	P	P
NF	VW	S	P	P	P	P	P
HIL	QC, CW	D	P*	P	P	P	P
HIL	QC, CW	S	P*	P	?	P	P
Total Score			9.2	8.9	9.5	9	11
Average Score			0.8363636	0.809091	0.8636363	0.8181818	1
Overall Score	0.831818						
(Battery SOC excluded)							

Table 4: Flywheel Case Testing Results

FAULT TYPE	WIND TYPE	SOLAR TYPE	FW RESPONSE TYPE	PQload?	PQgen?	PQgrid?	PQflywheel?	Grid Voltage/angle?	FW Mechanical Metrics?
NF	CW	D	Low Gen P	P	P*	P*	P	P	P
OC	CW	D	Low Gen P	P	F*	P*	P	P	P
1-P	CW	D	Low MGrid V	P	P*	P*	P	P	P
2-P	CW	D	Low MGrid V	?	F	?	P	F	P
3-P	CW	D	Low Green V	F	F	?	P	F	P
LIL	CW	D	Low Load pf	P	P	P	P	P	P
HIL	CW	D	Low Load pf	P*	?	P	P	P	P
NF	VW	D	Low Gen P	P	P	P*	P	P	P
NF	VW	S	Low Gen P	P	P*	P*	P	P	P
HIL	QC, CW	D	Low Load pf	P*	?	P*	P	P	P
HIL	QC, CW	S	Low Load pf	P*	?	P*	P	P	P
FW SPECIFIC TESTS									
OC-short	CW	D	Low Gen P	P	P*	P*	P	P	P
OC-short	CW	D	Low MGrid V	P	P*	P*	P	P	P
OC-short	CW	D	Low Load pf	n/a	n/a	n/a	n/a	n/a	n/a
Total Score				11.2	9	11.1	13	11	13
Average Score				0.8615385	0.69231	0.85385	1	0.8461538	1
Overall Score	0.85077								
NOTE: Mech metrics excluded from overall score									

For each table above, the numerical equivalent of every column of scores was added up to get the "total score" row, and then this number was simply divided by the number of tests that energy storage type had undergone to get an average score for each metric. Finally, these average scores were averaged again to give an overall score for that energy storage type; in this final stage of averaging, any quality metrics that came out to have an average score of 1 (Battery SOC and FW mechanical metrics) were excluded, as the logic governing their measurement was completely separate from the actual power-flow performance of the model (see Appendix 3).

The "core" 11 tests included every fault type, as well as a demonstration of the different renewable settings and the Q capability of the wind turbine. These tests were shared between all four cases. To highlight the additional capability of the flywheel to respond to different grid faults, it underwent three additional short open circuit tests to demonstrate its ability to handle short-duration losses of microgrid generation. In addition to these, the core tests also each required different flywheel triggers, for example, the short circuit fault mostly affected the grid voltage, and so that trigger was used for the flywheel. The not-applicable readings in the final test of the flywheel table are due to the power factor never dropping low enough from the short OC fault to trigger the flywheel. Because my flywheel model technically requires a (very low-priority) grid connection, the flywheel failing to trigger makes this model essentially equivalent to the grid connection case, and thus the data it would give is irrelevant for the flywheel table.

The final overall scores for each case are as follows: No Storage scored 0.6697 per test, Grid Connection scored 0.8432, Battery scored 0.8312, and the Flywheel scored 0.8508. Note that, rough nature of this scoring process aside, the three energy storage types performed fairly similarly, with all of them being better than having no storage at all.

Discussions, Conclusions, and Recommendations

My results are discussed in more detail in the next three subsections. The discussions subsection handles technical details, while the conclusions subsection talks more about general takeaways. Finally, the recommendations section suggests avenues for future research.

Discussions

Because of the restriction of my model to the phasor domain, the three forms of energy storage I tested behaved somewhat similarly, which is reflected in their similar scores. For example, none of these energy storage methods could handle the short circuit faults effectively; most metrics failed by the time two phases shorted across all models. This makes sense, as no matter how much extra power you are getting from additional storage, it is just going to get “sucked” down the fault like everything else. The actual solution to this type of fault would be switching logic, which would require some kind of study of grid topography, which was not included in my model. The similarity of all three ESSs is the main takeaway, but some more minor notes about my results are given below.

- The inductive faults seem to have been the least impactful, causing mostly “P*” and “?” scores. These were caused by unlikely distributions in power supply between generation types. Because Simulink was so heavily biased toward keeping the load power as close to rated as possible, sometimes it would draw significantly more Q out of a source than should be possible in the face of a large demand for it. This is not great for the microgrid, but not necessarily disastrous for customers like an outage would be, as microgrid controllers tend to be more sophisticated than the power flow algorithm used in Simulink. For a rural microgrid that lacks a central controller, however, this could be quite damaging, hence the reduction in score from 1 to 0.9 or 0.5.
- The Q correction provided by the wind turbine was not particularly useful for correcting the heavy inductive fault case, mostly because of its low reactive power capability.
- The open and short circuit faults were the only faults to significantly affect the microgrid voltage, but only the former was correctable using energy storage methods.
- The variable wind and sunny settings for the renewable generation had very little impact on the performance of the model; at most they would occasionally cause a P* because the

increase or decrease of power required from the diesel generator throughout the course of the model would cause a dip in its power factor barely below 0.9, to 0.85 or so.

- The complexity of the flywheel model and its dependence on a national grid connection lead to a lot of "P*" and "?" scores, even in trials lacking faults. This reflects its approximate nature.
- None of the trials for any of the four cases involved any deviation in frequency; any failures in the voltage/angle category were due to voltage sags. This is likely due to the fact that frequency is the main metric that Simscape Electrical's algorithm attempts to keep constant when running a simulation. Essentially, the fact that we have results at all for the full 12-hour simulation means there was no significant deviation in frequency.

Conclusions

The primary goal of this project was to design a streamlined, phasor-domain simulation of a microgrid and to use it to compare common energy storage types and their ability to handle faults common to a rural environment on a power flow level. The ultimate goal of building this model was to help islanded, rural microgrids get some form of low-impact, reliable energy storage and to add to the small pool of research on these isolated grids. With my knowledge from previous work modeling in Simscape Electrical, creating the microgrid and setting up relevant faults went very well. Building the energy storage models was the bulk of the project work, and the limitations of the phasor domain started to become clear while working on the flywheel model. Even with these limitations, I am confident in my robust final model's performance and applicability in the phasor domain, as displayed in the design evaluation section. In addition, the conclusion that each energy storage type behaves very similarly on a power-flow level and that they are all preferable to a complete lack of storage was clearly shown in the final results. Each

ESS was able to fulfill my service and quality requirements to a similar degree; they handled open circuit faults well but had trouble with short circuits. Inductive faults generally gave mixed results.

The primary conclusion for this project is that, when faced with a new microgrid installation in a rural area, the standard grid-scale battery is definitely the preferred method, with support with some kind of connection to the national grid being very useful. The latter is not always doable because of circumstance, but the former should be an automatic inclusion in any microgrid. Because all three storage types perform similarly in the most critical way, installing flywheel storage does not make very much sense. As discussed in this report, you need about 20 flywheels to give you the same instantaneous power capability as a single battery. These are large machines, and installing a huge warehouse of them in a rural (and likely somewhat low-income) location makes little sense. Even with the twenty flywheels installed, their energy capacity is much lower than the typical battery, as could be seen with my flywheel model which was based on a real flywheel and could supply power for 8.33 minutes as opposed to the battery, which could supply power on the scale of hours. If you happened to have a decommissioned power plant full of old generators that could be easily converted into flywheels, there wouldn't be much harm in including them in your microgrid, as they could help you ride out very short power interruptions (such as trees hitting power lines during wind storms). However, intentionally installing them makes very little sense.

The primary lesson that this capstone taught me was the critical importance of considering project scope before starting your work. The flexible nature of my simulation-only project and the fact that it allowed me to easily scale up or down the amount that I would attempt to accomplish in 20 weeks was an absolute blessing. My topic was fairly open in comparison to many projects, so not attempting to lock myself into any hardware expectations was a very good decision. Had I run into a problem akin to the restrictions of the phasor domain with some kind of hardware component, I would have had very little to show for my work at the end of this project. Instead, I have a complete model of a microgrid, a battery, and a functional if approximate model of a flywheel. Especially because I worked alone, having a flexible goal and reasonable expectations for my project scope was a good planning decision. Obviously, this project was more research-oriented than corporate- or product-oriented, but it still taught me an enormous amount about project planning and how real engineering work is actually done. Not to mention that, combined with my work in the summer, my capstone has given me nearly a year's worth of MATLAB-Simulink experience.

Recommendations

The main direction for future work would be to expand my model to include real-time simulation. I made several attempts over the course of this work to include real-time simulation of various components, but even when the Simulink blocks and logic to convert from the phasor domain to the real-time domain cooperated, the simulation was still time-limited by the real-time component, making the simulation take far too long. The restriction to pure phasor domain modeling impacted both the flywheel and battery models, so one area of possible future work would be to make these more detailed by actually simulating the inverter and switching behavior that would be responsible for handling the Simulink logic approximations I used. Being able to simulate true inertia and other real-time phenomena could also allow future researchers to simulate more energy storage types. Originally, I wanted to include pumped hydro storage as an alternative, but after struggling with the limitations of the flywheel model, I abandoned that as an option as it would essentially be another modified battery model with different specs. So, someone who wished to continue this work could look into that storage type once its various physical attributes could be modeled in real-time. This more demanding modeling might have to include some kind of special hardware to increase the simulation speed.

If a future researcher wanted to remain in the phasor domain, another interesting area of research would be case studies. Essentially, given that you have a real or proposed microgrid installation, you could plug the actual load, generation, and ESS numbers into my model and get a quick power flow simulation to help inform storage type and sizing decisions. This was part of the original intention of my capstone; to build an accessible model that others could use to help understand how a particular microgrid's power flow might behave.

References

[1] Kroposki, Benjamin et al. "Making Microgrids Work." *IEEE Power & Energy Magazine, 2008,* pp. 40-53.

[2] O. Dag and B. Mirafzal, "On Stability of Islanded Low-Inertia Microgrids." *IEEE, Kansas State University,* 2016. Manhattan, Kansas, USA.

[3] J. Sachs and O. Sawondy, "A Two-Stage Model Predictive Control Strategy for Economic Diesel-PV-Battery Island Microgrid Operation in Rural Areas." *IEEE Transactions on Sustainable Energy, Vol. 7, No.3,* July 2016, pp. 903-913.

[4] T. Nguyen et al, "Optimal sizing of a Vanadium Redox Battery System for Microgrid Systems." *IEEE Transactions on Sustainable Energy, Vol. 6, No. 3*, July 2015, pp. 729-737.

[5] Khodaei, A. et al. "Microgrid Planning Under Uncertainty." *IEEE Transactions on Power Systems, Vol. 30, No. 5,* September 2015, pp. 2417-2425.

[6] Francklyn, Lili. "Improving Energy Access in Rural Africa Depends on Renewable Energy Microgrids." *HOMER Microgrid News.* February 12, 2019: https://microgridnews.com/improving-energy-access-in-rural-africa-depends-on-renewable-energy-microgrids/

[7] A. Dayant and M. Nguyen, “Dimly Lit Renewable Energy Initiatives for the Pacific.” December 2018: https://www.lowyinstitute.org/the-interpreter/dimly-lit-renewable-energy-initiatives-pacific.

[8] A. Clark-Ginsberg, “What’s the Difference between Reliability and Resilience?” *Stanford University*. Accessed 3/3/2022: https://www.researchgate.net/profile/Aaron-Clark-Ginsberg/publication/320456274_What%27s_the_Difference_between_Reliability_and_Resilience/links/59e651230f7e9b13aca3c2ba/Whats-the-Difference-between-Reliability-and-Resilience.pdf.

[9] “IEC 61000-4-30:2015+AMD1:2021 CSV Consolidated Version.” *International Electrotechnical Commision,* March 2021: https://webstore.iec.ch/publication/68642.

[10] “Design Criteria for Developer Connection to the New York Power Authority Transmission System.” *NY Power Authority,* April 2020,

[11] Kawasaki Heavy Industries, Ltd. “Kawasaki Gas Turbine Generator Sets.” Printed October 2020, accessed July 2021.

[12] M. Kipness, “P1366 - Guide for Electric Power Distribution Reliability Indices.” *IEEE Standards Association,* February 2018: https://standards.ieee.org/project/1366.html.

[13] N. Mousavi, et al. “A novel photovoltaic-pumped hydro storage microgrid applicable to rural areas.” *Applied Energy 262,* 2020.

[14] Jonathan LeSage, “Systems-Level Microgrid Simulation from Simple One-Line Diagram.” May 15, 2018. Accessed July 2021. Contains “exampleMicrogridLibrary” used for simulation: https://www.mathworks.com/matlabcentral/fileexchange/67060-systems-level-microgrid-simulation-from-simple-one-line-diagram.

[15] Level, “Wind turbine systems.” Accessed on 3/8/22 from https://www.level.org.nz/energy/renewable-electricity-generation/wind-turbine-systems/.

[16] Center for Sustainable Systems, "Wind Energy Factsheet." University of Michigan. 2021. Pub. No. CSS07-09.

[17] J. A. Martin and I. A. Hiskens, "Reactive power limitation due to wind-farm collector networks," *2015 IEEE Eindhoven PowerTech*, 2015, pp. 1-6, doi: 10.1109/PTC.2015.7232809.

[18] Zubiaga, Markel, Alain Sanchez-Ruiz, Eneko Olea, Eneko Unamuno, Aitor Bilbao, and Joseba Arza. 2020. "Power Capability Boundaries for an Inverter Providing Multiple Grid Support Services" *Energies* 13, no. 17: 4314.

[19] X. Li and A. Palazzolo, "Multi-Input-Multi-Output Control of a Utility-Scale, Shaftless Energy Storage Flywheel with a 5-DOF Combination Magnetic Bearing." *Journal of Dynamic Systems Measurement and Control 140,* 2018.

[20] J. Goodfellow and P. Appelt, "How Trees Cause Outages." *Environmental Consultants, Inc.* Accessed January 2022.

[21] R. Quint et. al, "Phase Angle Calculations: Considerations and Use Cases." *North American SynchroPhasor Initiative,* September 2016. Accessed 3/11/22.

[22] MATLAB Help Center, "Simulate Simulink Model at Specific Operating Point." *MathWorks.* Accessed July 2021 https://www.mathworks.com/help/slcontrol/ug/simulate-simulink-model-at-specific-operating-point.html.

Appendices

Appendix 1: General Simscape Electrical User's Manual

What follows is a general user's manual for building a phasor-domain model in Simscape Electrical. This particular manual references a model of Union's power grid that I built during the summer of 2021 (see appendix 3 for more information), but the general principles of model building are the same as those used in this project.

The final 24-hour, steady-state model of Union's power grid was built in MATLAB-Simulink version R2019 with the Simscape Electrical toolbox. The following user manual will guide the user through the general process of building this model (or a similar one). The manual is organized into the following sections: Basic Principles, Generation, Solar, Grid Connection, Loads, Motors, Load Flow, and finally Measurement/Analysis. Each section will provide tips on simulating their respective parts of the model. Each of these parts should be built and tested separately before attempting to build the final model. DO NOT ATTEMPT TO

BUILD EVERYTHING AT ONCE, as there is essentially no chance you can troubleshoot the inevitable deluge of errors you will get with this approach. This manual will not provide an exact step-by-step guide, but it should give the most relevant information to make rebuilding the model fairly easy.

A. Basic Principles

Before we start, the most important thing to remember while working on a Simulink model of any kind is to make heavy use of examples, both in the MATLAB software itself and on the file exchange. The best place to start for this model would probably be the example in [14], which provides a basic microgrid model as well as a library called "exampleMicrogridLibrary," which contains many blocks commonly used in microgrid models.

The first thing that must be included in every Simscape Electrical model is the powergui block. This block has to be placed at the highest level of the model and does not connect to anything. In the block, you can set the universal model frequency (for us, 60 Hz), and use a number of helpful model analysis tools, most notably the load flow tool (see Section G).

Most of the model simulates a balanced three-phase line, with separate signals for phases A, B, and C. These lines use a special signal type, which is denoted by the square input/output ports on the blocks that use it. These blocks, which all form the critical basic components of our model, are found in the Simulink Library Browser under Simscape>Specialized Power Systems. The three-phase signal used by these blocks is proprietary and cannot be connected to either Simulink logic signals (denoted by an arrow at the end of the signal) or physical Simscape signals (denoted by color, usually blue or brown) without being converted through a block first. For instance, in Figure 1 you can see that the block labeled “Main Synchronous Generator” outputs on the Simscape ABC bus, but it also outputs its measurements as logic signals, which the controllers work with and feed back into the generator. Simulink logic signals are used fairly often in our model, but it is important to remember that they can never be connected to the main three-phase bus. Physical signals, while not used in this model, could also be used without a direct connection to model things like controllers in the future.

The load layout in Figure 4 is the highest-level view of the finished model. Each gray block is simply a link to another, smaller model called a subsystem. The load subsystems in this figure mostly contain individual building loads (see section E), but the cogen plant and grid connection point are also subsystems, just connected at the busses labeled “To Cogen Plant” and “To National Grid Connection.” When building this model, one should start by constructing each important section separately as a standalone model before connecting them as subsystems in the manner shown in Figure 4. To create a subsystem, simply highlight the blocks you want to put in it, go to the three blue dots and select “create subsystem.” Simulink will automatically create inputs for the subsystem based on what signals are left unconnected in your selection. Sections B-F below will explain how to go about creating each important subsystem in the model.

B. Generation

The most important subsystem to understand is the generator model. There are two required blocks here: Synchronous Machine pu Standard (labeled “Main Synchronous Generator” in Figure 1) and Excitation System (“Generator Excitation System”). These blocks are always required and can simply be connected as shown in the figure. The third block shown, Steam

Turbine and Governor ("Turbine and Governor") is not strictly required but is the standard choice for controlling the synchronous generator's input power. There are other options for doing this, including speed control and a hydraulic turbine, but they will not be discussed here. Instead, the layout in Figure 1 is recommended.

As for the parameters inside each block, there are some key ones, but the others can largely be ignored.

First of all, go to Synchronous Machine>Load Flow and set the generator type to PV. If you are still simulating Union's cogen plant, the active power generation is 1.816e6 W and the Q limit is +/- 879529 VAR. These limits can be set to whatever is correct for your generator, but the generator type always needs to be PV for load flow to behave properly.

Very little in the "Parameters" tab of the synchronous machine needs to be changed from default values except for the first line, which specifies nominal power, voltage, and frequency and the. For our model, this line is [2.668e6 13200 60], but can be adjusted to fit your generator. The other parameters do change generator behavior, and if you happen to know things like inertia coefficient, friction factor, or machine reactances, you can include them here for accuracy. In addition, if you know you have a number of pole pairs other than the default/standard 2, you can change that as well. Note that the way to specify the synchronous speed of your generator is by setting the pole pairs and frequency, which determine the machine's speed according to speed = (frequency*60)/(#pole pairs). You cannot set the speed directly. For the sake of our model, you can simply estimate the other parameter lines handling time constants and reactances to produce the desired post-loss power, in our case the 1.76 MW shown in Figure 8. Note that the "initial conditions" line is set by the load flow tool and should not be changed manually.

Most of the parameters in the two controller blocks are not worth changing from their defaults unless you have specific numbers for a particular device you want to model. The two exceptions are found in the “Turbine and Governor” block. These are the “steam turbine time constants” and “Nominal speed of synchronous machine” lines. The former controls how fast the generator will be able to adjust its output, while the latter must agree with the frequency and pole pairs set in the synchronous machine. For our model, the cogen plant’s gas turbine is considerably more responsive than the average steam turbine, so the time constants were reduced to [0 2 .8 0.1] to reflect that. The cogen plant is also a standard 2-pole 60Hz machine, so the nominal speed is set to 1800 rpm. Finally, both generation controller blocks also have initial condition lines set by the load-flow tool.

Finally, outside the blocks themselves, reference speed and voltage should always be set to 1 (the blocks themselves are simply called “Constant”) and the reference power input will be controlled later by the load flow tool. If you are just experimenting with generator output before building the rest of the model, the reference power is just the proportion of the generator’s rated apparent power that it will attempt to output as active power. This can go above 1, but overloading the generator will cause a huge spike in torque and generally unstable behavior.

C. Solar

The solar model itself, shown in Figure 3, seems simple enough to construct from inspection, but there are a few tricks to getting it running:

First of all, you need the library “exampleMicrogridLibrary” to use the blue solar block at all. The block is called “Grid-Connected PV Array (PQ Model)” in the library and requires the library to be in MATLAB’s file path to work.

The block’s parameters are straightforward, but one internal variable needs to be changed in order to produce the solar output used by our model, shown in Figure 7. To do this, go to the arrow in the bottom left corner of the solar block to open the mask, and in the “Simple Solar Inverter” block you will see as a result, change the active power at initial voltage to -1. In other words, your second line should be [-1 -6.7762e-13].

The actual output of the solar model is specified by a Simulink logic signal on the input P. In our model, this is simply a switch block, a clock as a control signal, and a 0 constant and sinewave to switch between. The “threshold” parameter is set to 43200 seconds or half a day, and thus the sinewave switches to a constant 0 at noon in the simulation.

To specify the peak power output, set the amplitude of the sinewave. For a full-day simulation, the frequency should be set to 7.27222e-5, rad/sec in order to get a single “hump” before it goes to 0. This number can be played with depending on how long your simulation is and how you want your solar estimate to look. All other sinewave parameters can be left alone.

Note that the solar model is not placed in its own subsystem, but there are actually two of them placed amongst the building loads, according to where the solar installations are actually placed on campus. There is one 3 kW and one 12 kW installation. This is a small detail unique to our Union model, and you can put this solar model wherever you want to add some active power to a three-phase bus.

Finally, when you use the load flow tool on a model with this solar block, it will ask you if you want to set initial conditions for the library block. ALWAYS SAY NO TO THIS.

D. Grid Connection

The subsystem simulating a national grid connection point, shown in Figure 2, is simply made up of a Three-Phase Source block ("National Grid Model"), a Three-Phase Transformer Block, and loads. "Feeder Impedance" is a Three-Phase Parallel RLC Branch, while "Capacitor Bank" is a Three-phase Parallel RLC Load. The VI Measurement block between the voltage source and the transformer does not impact model performance and is just for measurement as the name implies (see Section H). Although it isn't shown, there is a three-phase breaker on the line connecting Figure 2 to Figure 4, allowing for the islanding tests done in Figure 9.

The most critical thing to do after setting up the grid point as shown is to go into the voltage source's load flow tab and change the generator type to swing. This is what allows this block to supply (or receive) however much P and Q is required to keep the system stable. As for the other parameters for the source, the phase-to-phase voltage is arbitrary, as long as it agrees with the voltage on the primary side of the transformer and the base voltage parameter at the bottom of the source. The frequency for the source should match the system frequency, and the other source parameters can be left alone (The resistance and impedance shown on the block itself are very small, and load flow will set the voltage phase angle itself).

The important parameters to set in the transformer block are the nominal power and frequency, as well as the voltages for each side. Note that the side with the capital letters is winding 1, while the lowercase letters mark winding 2. The frequency is obviously set to the grid frequency, but the nominal power does not need to be so precise, it just needs to be more than the total load present in your grid. In our model, it's 3e6 W. The transformer resistances and inductances can be set if you know specific device values or are able to estimate them based on standards, but this is more relevant when modeling loads, and will not matter much for this grid connection transformer (see Section E). This transformer should step your arbitrary grid voltage down to whatever voltage you want your main busses to run at. In our case, this is 13.2 kV.

Lastly, the two loads present at the grid connection point are what represent the capacitor bank discussed earlier in this report. The branch should be set to RL branch type, but its values are somewhat arbitrary as it is only supposed to roughly represent the line impedance of the national grid. In fact, we would recommend completely leaving this load out of your model until you are ready to handle sizing the capacitor bank, which should be the final step in preparing your model for analysis, as we will describe now.

The capacitor bank load should be set to the obvious nominal voltage and frequency values for your grid, and should only generate capacitive reactive power (setting the other powers to 0 will make it show up as only a capacitor). To properly size the capacitor bank for a grid power factor as close to 1 as possible (as we see in Figure 6), first make sure that its load type is set to constant Z in the load flow tab. Now, we can use this capacitor to adjust for the inductive load that commonly arises from building loads, transformer leakage, and most notably the fact that generators always run at a rated power factor below 1, which in our case is 0.9. For this model, we sized the capacitor bank manually, meaning we picked capacitance values arbitrarily until the grid power factor was as close to 1 as possible. This is a somewhat inefficient method, as having too small a capacitance often causes the load flow tool to fail. This can be fixed by increasing the feeder impedance, but overall it is a tedious balancing act that should only be done after you are confident that your model is finished. Also, if you don't care about the power factor at your grid connection, don't bother doing it at all.

So, strictly speaking, all you need to model the macro-grid is a three-phase source set to be a swing generator, but there are a number of improvements you can make depending on what results you need.

E. Loads

While modeling the entire load layout of your grid seems like it would be the most complex part of building this kind of model, it's actually fairly simple. First, break down your grid into loads that share a step-down transformer. In our case, this was done for us in the form of a campus one-line diagram. Then, every load "chunk" (It can be multiple buildings, one building, or parts of a single building) can be simply modeled by a parallel RL load and an accompanying step-down transformer, as seen in Figure 15, which shows the "Social Studies Loop" subsystem and its respective loads. This approach to modeling loads is very customizable; you can change how much of each load is inductive (for our model, half the real load value was inductive for every load), the voltage each load runs at, and as we will see in the next section, how advanced each individual load model is. Of course, there can be as many or as few loads as you want total, as connecting them in parallel simply adds their power consumption to the grid. In our case, all we had to do was follow what was shown on the one-line diagram. For instance, the "Social Studies Loop" subsystem comes from a bus that is labeled as such on the diagram, meaning these three buildings are actually connected together in this manner in Union's actual grid. Also notice that the solar model is providing power directly to the Olin + Wold load. This is because, in reality, the solar is on the roof of the Wold building.

The last thing to note about this section is that even though the modeling here is pretty simple because there are so many transformers used, the leakage impedances mentioned in Section D really start to matter on a system-wide scale. Ideally, you could set the parameters for every transformer based on real devices, as % leakage is a common rating. However, these values are actually pretty standard and can be easily looked up, as we did earlier in this report. As long as you keep your voltages straight, this is actually the easiest part of building this model, even if it is a bit tedious. It also has the most potential to increase in complexity in the future, as we will see next.

F. Motors

Most of the design process for the motor model is straightforward from Section 3D, where the theory behind the feedback setup for each load is discussed thoroughly. Here are some details:

The asynchronous machine block used is called "Asynchronous Machine pu Units." The configuration tab went untouched in our model, and the parameters tab is very similar to that of the synchronous machine. Like you did with the generator, you should only really set the nominal power, voltage, and frequency, as well as the number of pole pairs. If you happen to know specific values for the other inductances/resistances, you can set them, but they aren't that impactful. The only other important parameter is under the load flow tab. For the sake of a steady-state model, you will want your mechanical power to match the nominal value.

The constant blocks with lines through them are actually initial condition blocks, annoyingly just called "IC." These blocks set the initial values for the input torque to each motor.

When running the load flow tool, it will not be able to set these blocks itself, but it will tell you what they should be in a popup, so you can do it manually.

The other two blocks are not critical for a functional model, but they do make the motor model slightly more accurate. The triangular gain block simply multiplies the speed by a constant before it becomes torque and can be used to simulate slip. In our model, it's 5%, but this only really affects the rotor angle measurement and is not very important. The other block is a rate limiter, which helps reduce startup transients. At steady state this block is irrelevant.

The final thing to note is that despite being used as a replacement for the simpler static load model, these motors cannot be connected directly to another specialized power block, like a transformer. You have to include a small "parasite load" in parallel with the motor. This load can be very small and has essentially no impact on grid performance. We recommend putting this parasite load right next to the entrance to your motor load subsystem, as this allows you to put any number of motors in parallel with it without any problems.

G. Load Flow

Despite the many provisions we've made throughout this manual for the load flow tool, actually using it is pretty easy. First, you will want to look at the load flow settings in the powergui block>tools. This is where you set the standard grid frequency. 100e6 is always a safe choice for base power, although if your machine base powers are all very low, you can lower it to 10e6. Max iterations is the number of times the tool will attempt to find a solution. For a model like ours of limited complexity, you can get away with increasing this to say, 500. Similarly, the PQ tolerance is the maximum distance between solution "guesses" that the tool will expect as the solution conversions. This can be quite small for our model; the final model had this set to 1e-5.

As for the actual tool itself (click on the “load flow” bar above the settings), it's very straightforward. The first thing you want to make sure you do before running load flow on your model is to add bus blocks. This will add the orange labels we’ve seen in figures throughout this report. This is important, as without these labels the tool will simply choose a random element to provide bus voltage, which can cause a lot of problems. After doing this, the actual parameters the tool sees are shown on the left side of the table: bus type, base voltage and angle, P and Q, and Q limits. These should all be set when designing the other parts of your model. Upon pressing “Compute,” new values for all of these will show up in the right side of the table, in the columns labeled “LF.” Pressing “Apply to Model” will update all of the initial condition parameters in your model’s machine blocks (and will give you any relevant popups with respect to solar or motor models) to force the simulation to start as close to steady state as it can manage. The “compute” stage is the most likely to fail, giving either a modeling error or a failure to converge message. If this happens, (which it will), check over your model and make sure that everything is set up properly, as mentioned in other parts of this manual. The capacitor back sizing process is particularly finicky when it comes to load flow, so do that step last. So, using the tool is easy, but getting it to actually work is not.

H. Measurement/Analysis

Once you have finished your model and gotten it to agree with the load flow tool, the last thing to understand is how to get meaningful data out of your simulation. The crux of getting nice plots can be seen in the top left of Figure 15, where the three-phase line is converted first into voltage and current through the “Three-Phase V-I Measurement” block and then into P and Q through the bizarrely named Power (3ph, Phasor) block. Because the measurement block passes the three-phase signal through it without modifying it all, you can simply place it around your model to feed powers into a scope block and get plots similar to those in the results sections. Comparing P and Q around your model is largely enough to make sure everything is working correctly. However, you can also easily calculate things like power factor from the P and Q buses using basic Simulink logic blocks (add, multiply, etc.).

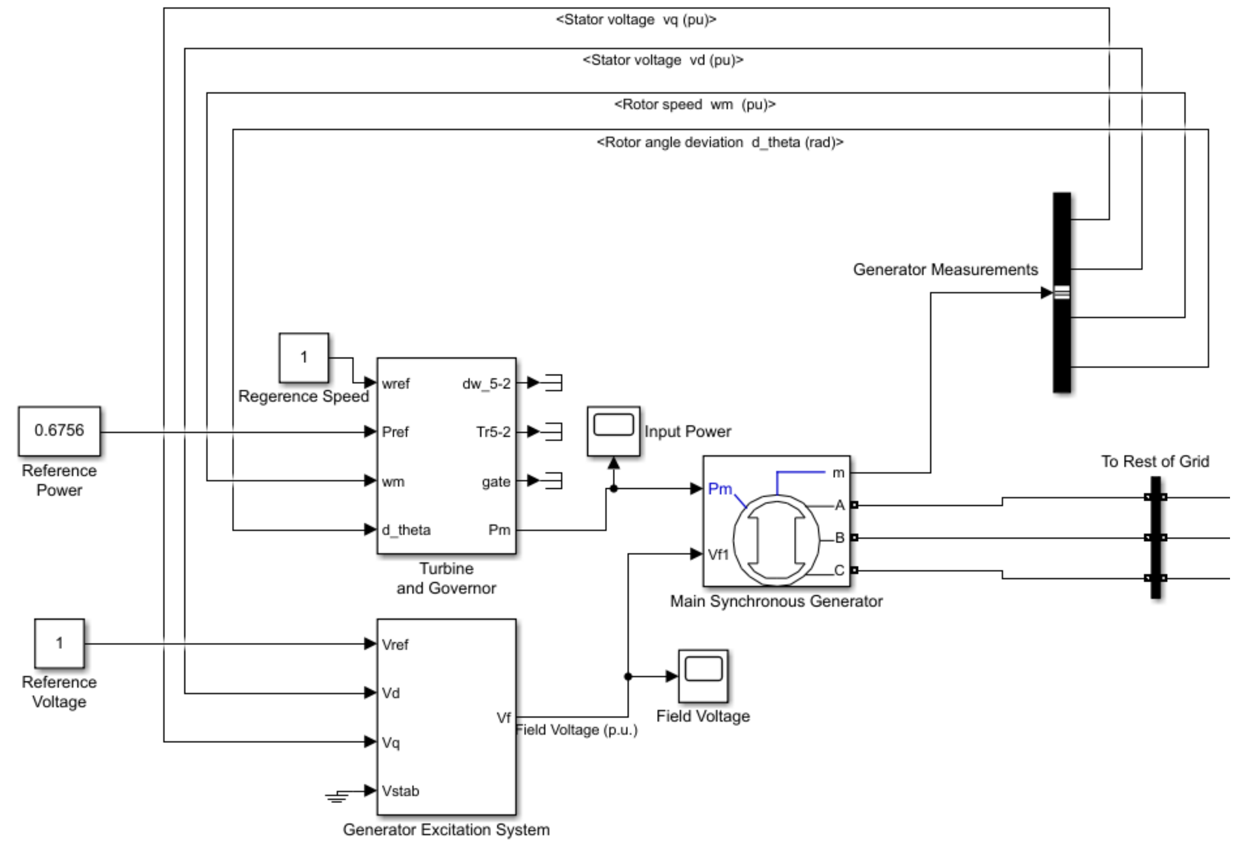
That's all you need for a model of similar complexity to ours, so here are a few miscellaneous tips for improving the model's presentation and functionality:

Most of the time, MATLAB can pick a system solver on its own. However, sometimes the model is able to compile when you run it but the actual progress through your simulation period is super slow. If this happens, try going to the very bottom-right of the simulation window, where the solver type will be highlighted. Click on it and then click on the gear icon to open the solver settings. From there, pick different solver types (other than the default "auto") until the model runs in a more reasonable amount of time. This issue can occur when doing very long simulations or simulations with a lot of transients; MATLAB sometimes will not know which solver to use.

Another thing to mess around with in the solver settings is the max step size under "solver details." For shorter simulations that involve transients (like motor startup tests), reducing the max step size can help reduce mathematical noise.

Make sure to use "goto" tags and their corresponding "from" blocks for moving around measurement signals (SSP and SSQ in Figure 15), as this will reduce the need to constantly go in and out of subsystems.

As mentioned in the report, if your load flow tool manages to converge but generates some questionable initial conditions resulting in big startup transients, a last-ditch effort for a nice steady state graph is manually setting the model's initial conditions using the steady state manager. For more on how to do this, see [22].



Appendix 2: Sophisticated Simscape Synchronous Generator Model

Appendix 3: Additional Files

See the included files for my final 6 models' .slx Simulink files, the example library used throughout this report, an excel file of my test results, and my report from the summer of 2021. The test results file is called "TestResults" and the library file is called “exampleMicrogridLibrary.” My summer 2021 report is labeled “PennockCSummer2021Report_v2.” The 6 final models are “finalNSModel” for the no source case, “finalGridModel” for the national grid connection, “finalBattModel” for the battery, and three versions of the flywheel model depending on what grid fault triggers it: “finalFWmodelP” for low generator P, “finalFWmodelV” for low grid voltage, and “finalFWmodelpf” for low load power factor.