Modular Interactive Modeling for Control and Simulation of Electric Power Systems

by

Sarah Flanagan

B.S. Electrical Engineering and Computer Science, MIT (2019)

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2021

Author . . . .
Department of Electrical Engineering and Computer Science
June 1, 2021

Certified by. . . . .
Marija Ilic
Senior Research Scientist
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Modular Interactive Modeling for Control and Simulation of Electric Power Systems

By

Sarah Flanagan

Submitted to the Department of Electrical Engineering and Computer Science
on June 1, 2021 in Partial Fulfillment of the
Requirements for the Degree of Master of Engineering in
Electrical Engineering and Computer Science

ABSTRACT

Power systems must ensure reliable service during normal operation and unexpected disturbances. They also should enable decarbonization goals by supporting utilization of new renewable energy resources that are being added to the system. Conventional control used in power plants and generators is becoming insufficient because previously true assumptions no longer hold with the widespread implementation of renewable energy sources. Future electric power systems will comprise of a more distributed grid of loads and Distributed Energy Resources (DERs), all contributing to electricity service goals. Novel modeling and control for their provable performance are actively being pursued. This thesis builds on the idea of novel modeling and controlling future electric power systems using a multi-level modular approach. Particular emphasis is on general simulation tools for assessing dependence of these new architectures on control design. A MATLAB-based Centralized Automated Modeling of Power Systems (CAMPS) software models the primary dynamics of components in a modular way and develops a centralized model of the interconnected system. In this thesis further extensions to CAMPS improve plotting of state variables and their expressions, enable conversion from the dq (direct quadrature) reference frame to the abc-reference frame, and allow substitution of different controllers into an open loop model.

A recently introduced modeling approach, which maps voltage and current variables into the energy space and interactively exchanges energy space variables called interaction variables between components, is used as the starting model for new simulations. One energy space-based controller is simulated using Simulink to test the controller's performance when using a switching model instead of an average model. A new software tool, Plug-And-Play Automated Modeling of Power Systems (PAMPS) based on this recent theoretical work implements distributed algorithms in MATLAB. One example applies PAMPS to a RL (resistive and inductive) circuit controlled by a voltage source and connected to a constant power load. Future work can use PAMPS to model additional electrical components including synchronous machines and solar inverters. Since PAMPS exchanges information within the energy space, it can also be applied in future work to model the interactions between multi-energy sources such as mechanical and thermal energy conversion components.

Thesis Supervisor: Marija Ilic
Title: Senior Research Scientist

Acknowledgements

The work of this thesis was funded by the Department of Energy. I would like to express by gratitude to the Department of Energy for their financial support with DOE grant DE-EE0009031 for the "Secure Monitoring and Control of Solar Power Distribution Systems through Dynamic Watermarking project."

I want to give a very special thank you to my research supervisor Professor Ilic. She has been incredibly supportive and encouraging and has taught me so much. I am inspired by her excitement for power systems, multi-layered modeling, and creating a more renewable future and all that she has been able to accomplish in her academic career including breaking down barriers for women in academia. I am thankful for the many hours she spent helping me, working with me, and teaching me about power systems, and for always looking out for me. I feel so fortunate to have had her as my supervisor.

I thank my academic advisor Joel Voldman for all of his support throughout both my undergrad and my Masters. He has been a huge help with advising me throughout most of my academic, research, and job-related decisions during my time at MIT, and I would always feel better after meeting with him. I would also like to thank the Course 6 Undergraduate Officers, Katrina LaCurts, Vera Sayzew, Brandi Adams, Ellen Reid, and Myriam Berrios for everything they do for the department to support their students and make course 6 a great community.

I would like to thank Dan Wu, Premila Rowles, Miroslav Kosanic, Pallavi Bharadwaj, and Rupamathi Jaddivada for being wonderful collaborators, groupmates, and friends. They made me feel very welcome to the group even with all our meetings being virtual this year. I am looking forward to getting to meet you all in person soon. I would like to thank Pallavi for all that she has taught me about power electronics, per-unitization, and Simulink. I thank Rupa for the countless hours she has spent helping me and for everything that she has taught me about control, energy space modeling, CAMPS, DAMPS, SGRS, PAMPS, and power systems. She has been so supportive and helpful, and I am very grateful for all of her assistance.

I thank the MIT Lincoln Labs engineers Matthew Overlin, Edward Corbett, Chris Smith, and James Macomber for providing their insights about the Simulink simulations of an energy-space controlled inverter connected to a synchronous machine and for sharing their switching mode version of Xia Miao's average mode Simulink model with the group.

I want to thank Professors Joe Steinmeyer and Gim Hom for giving me the opportunity to TA, 6.111, my favorite class at MIT, and for being amazing lecturers. It was a wonderful experience both taking 6.111 and the IAP course 6.S186 with you, and then getting to work with you to help teach the material to other students. I thank Professor Jacob White for introducing me to control theory in 6.320, being a great lecturer, and for his kindness and encouragement.

I want to thank my past grade school, middle school, and high school teachers for their dedication, for getting me excited about math, science, and engineering, and for preparing me well for work at MIT. I would like to especially thank Mrs. McArdle, Mrs. Schweer, Mrs. Barstow, Ms. Lauritsen, Mr. O'Dell, Mr. Minsavage, and Mrs. Lurz.

Finally, I want to give a huge thank you to my friends and family for all of their love and support throughout my entire life and for making this very difficult year a lot better. I thank my grandparents for loving me, seeing the best in me, and all of their words of encouragement. I want to thank my boyfriend Trevor for all of his love over the past 5 years and for being there by my side every step of the way throughout both the good times and stressful times and late nights. I especially want to thank my parents for their love, always being there for me and believing in me, and pushing me to keep going even at times when I wanted to give up. I love you all so much and feel so fortunate to have such an incredible boyfriend, family, and friends.

Table of Contents

List of Figures

1 Introduction

1.1 Thesis Motivation

Large scale electric energy systems such as transmission grids, distribution grids, and micro-grids need a method to be controlled. These systems need to be able to have reliable service during normal operations and be able to handle disturbances and power imbalances [1]. Traditional solutions simply controlled power plants and generators [2].

In order to meet environmental goals towards decarbonization, renewable energy resources need to be deployed. Throughout the years, renewable energy sources have become increasingly widespread, and states such as Hawaii and California have set specific goals relating to renewable energy sources. Both California and Hawaii have set goals for use of 100% renewable energy by 2045 [3] [4] . However, the deployment of renewable energy sources introduces new challenges to the power grid. Some of the previous assumptions that were used to design control of power plants and generators no longer hold.

1.2 State-of-the-Art Practices

Modern energy management systems through Supervisory Control and Data Acquisition (SCADA) are responsible for on-line scheduling and regulation of electric energy systems. Control centers monitor the bulk power system's demand, generation, and transmission status. There is an information exchange from ISOs (independent system operators) to residential customers [5].

Conventional generators use governor and excitation control, and the gains of these controllers are tuned for the static worst-case scenario [6]. Overdesigning the system for the worst-case scenario prevents system operators from fully utilizing all the intermittent renewable resources and results in inefficiencies and inability to meet environmental goals [7]. Traditional control has implied timescale separation which assumes that the transmission and distribution grid is instantaneous and stable relative to the generator's dynamics [8]. However, this assumption only holds when the system dynamics are faster and stable relative to the time constants of the transmission and distribution equipment [8]. Traditional control of power systems also assumes that active and reactive power can be decoupled. With the introduction of renewable energy sources to the system, these assumptions are no longer valid.

In today's industry, operators rely on storage to ensure reliable service during unexpected changes in demand. However, these storage demands are often not economically feasible. For instance, it is estimated that 0.9MW of flexible storage would be required for every 1MW of renewable power [5] .

The power systems of today have a one-way flow from generators to loads [9]. However, future power systems will need to have a two-way flow [9]. Future SCADA will need to support cooperative coordination. Future power systems will need a more distributed grid to allow for the ideal placement of loads and Distributed Energy Sources (DERs) [9] . In order to effectively implement this distributed grid, there needs to be a framework for modeling, controlling, and simulating the power system.

1.3 Previous Work within EESG

Previous work within Professor Marija Ilic's Electric Energy Systems Group (EESG) has proposed that an electrical system can be modeled as being multi-layered in space and time. This means the system can be observed at different time scales and either the entire power network or selected parts of the power network can be observed [10]. This can also be referred to as "zooming in and zooming out" [11]. There are three layers of control: the tertiary, secondary, and primary layer, which are described in [10]. The tertiary level assumes that the dynamics of the electrical system have settled, and acts at a slower time scale. It uses power forecasts and economics of markets to determine set points. The primary layer is the physical model and control of each module. These dynamics occur at a fast time scale at the order of milliseconds or microseconds. The secondary layer or wrapper, is what maps the tertiary layer to the primary layer [10].

The group has created several software tools for modeling different aspects of the multi-layered electrical system. A MATLAB based software, Centralized Automated Modeling of Power Systems (CAMPS), takes a modular approach: separating the dynamics of each component into its own module [12]. Information about voltages and currents is exchanged between modules. CAMPS then computes the state space equations of the interconnected system in a centralized way [12].

CAMPS models the primary dynamics of the system. While it works well for small interconnected systems, it takes a long time to create the interconnected system and simulate system dynamics for larger systems. For larger systems, it is better to model the electrical system as interacting distributed modules. The MATLAB-based software, SGRS (Smart Grid in a Room Simulator) Platform enables distributed modeling of power systems [13]. Previous work within the group used SGRS to develop methods for modeling the primary level and tertiary level, but did not have a secondary layer to map the two layers implemented in software.

The group has also proposed a new method for modeling and control of power systems using the energy space. State variables such as voltage and currents can be mapped to the energy space. Rather than exchanging information about voltages and currents between modules, information is now exchanged in terms of energy, power, and reactive power. One mapping to the energy space and a controller in the energy space for controlling synchronous machines and an inverter-based battery was proposed by Marija Ilic and Xia Miao in [14]. Xia Miao created a Simulink model that uses an average model to simulate a solar inverter connected to a synchronous machine with resistive and inductive (RL) loads. A general component-agnostic multi-layered mapping from conventional state space models of system components to the energy space and multi-layered controller in the energy space was proposed by Marija Ilic and Rupamathi Jaddivada in [15] and [16].

1.4 Thesis Contributions and Organization

The contributions to this thesis include additional functionality added to CAMPS. Improvements were made to the CAMPS plotting tool, which plots the system trajectory of state variables of the interconnected power system. A function that maps the dq-reference state variables back to abc-reference frame converter was also added. Additionally, a new wrapper block models the open loop dynamic of a module and allows for different controllers to be substituted in.

Additional simulations are run using the Simulink model created by Xia Miao. The simulations reveal issues with using Xia’s proposed energy space controller when going from an average model to a switching model of the solar inverter. Changing capacitor values and adding in a moving average block to the control signal partially solved these issues, but open questions still remain.

The thesis also builds upon the theoretical work of [15] and [16] to create a new software which utilizes the PAMPS (Plug-And-Play Automated Modeling of Power Systems) approach. This modeling approach is fundamentally based on mappings from conventional state space to energy state space. PAMPS modules calculate their own state variables, map these state variables to interaction variables in the energy space, and communicate interaction variables with their neighbors. The thesis describes the algorithms for implementing PAMPS main modules and submodules within software. The thesis also describes how the general PAMPS software model can be applied to an RL circuit supplied with a controllable voltage source that is connected to a constant power load.

Chapter 2 explains the modular modeling approach used in CAMPS. It will also detail the additional features that I added to the existing CAMPS software. Chapter 3 looks at the test system for the controller proposed by Xia Miao. It stresses implementation issues when going from an average model to switching model. It suggests methods to fix some of these issues and remarks on lingering open questions. Chapter 4 describes how the newly created software tool PAMPS (Plug and Play Modeling of Power Systems) can be implemented in a distributed way within the SGRS platform. The general algorithm is described and an example shows an application of this software to an RL circuit with a controllable voltage source that is connected to a constant power load. Finally, Chapter 5 details future work. It proposes how using the PAMPS software modules can be created for many types of electrical components including synchronous machines and solar inverters. The chapter also details how PAMPS can enable modeling of the interactions between multiple energy sources such as mechanical and thermal energy.

2 CAMPS: Centralized Automated Modeling of Power Systems

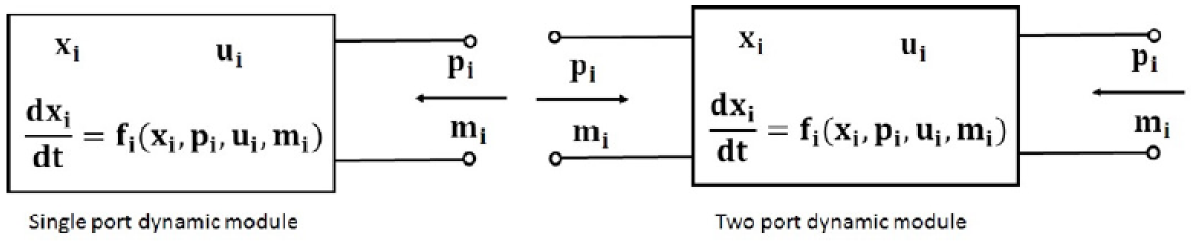
Centralized Automated Modeling of Power Systems (CAMPS), is a MATLAB based software tool originally developed by Kevin Bachovchin under the supervision of Marija Ilic, with further developments from Rupa Jaddivada that can model electrical systems in a centralized way [12]. There are MATLAB classes within the CAMPS library that model different components in power systems including a synchronous machine, transmission line, load, induction machine, flywheel model, solar PV (Photovoltaics) model, battery, and an infinite bus [17]. Each of these classes can be referred to as a CAMPS module.

CAMPS is unique from other circuit modelling tools such as SPICE because in addition to modelling the primary dynamics of the system, it also contains the standard state space model of the connected system [17]. This allows for primary control of the system. CAMPS modules exchange information between each other in terms of voltages and currents. The interconnected state space model is produced using Kirchhoff's Laws for Voltage and Current. CAMPS can also calculate the equilibrium, linearize the system around an operating point, perform participation factor analysis, calculate the system's eigenvalues, and plot the system's dynamic response [12].

2.1 CAMPS module general structure

CAMPS modules all have a similar structure which is described in [12]. Internally, a module i has a vector of state variables x_i, and a vector of controllable inputs, u_i. The module can have exogenous inputs m_i. It also can receive port inputs p_i from the modules that it is connected to. It also sends out port outputs to other modules. These port outputs are a subset of the state variables x_i. The module can be single port if it is connected to a single component or two port if it is connected to component on both sides. [12] Figure 2-1 shows the single port and two port

CAMPS module.



x_i
u_i
$\frac{dx_i}{dt} = f_i(x_i, p_i, u_i, m_i)$
p_i
m_i
Single port dynamic module
Two port dynamic module

Figure 2-1 Single-port and two-port CAMPS module from [12]

Using these state variables, port variables, controllable inputs, and exogeneous inputs, the state space of any module can be written using equation (2.1), which is introduced in [12]. Initial conditions are set in equation (2.2), and u_i is written using equation (2.3) [12]. The variable y_i is an output of interest calculated by using a function of the state variables and y_i^{ref} is the setpoint for this output.

$$\frac{dx_i}{dt} = f_i(x_i, p_i, u_i, m_i) \tag{2.1}$$

$$x_i(0) = x_{i0} \tag{2.2}$$

$$u_i = g_i(y_i, y_i^{ref}, m_i) \tag{2.3}$$

The port inputs and outputs to each module are in terms of voltages and currents. A centralized state space model of the entire interconnected system can be derived using Kirchhoff's Laws for Voltage and Current.

2.2 CAMPS Code

CAMPS is implemented using the programming language MATLAB. Each CAMPS module is implemented as a MATLAB class. Each CAMPS class is a submodule to the superclass called Module. The Module superclass specifies that each submodule will contain properties for its state variables, port inputs, port outputs, controllable inputs, setpoints, and controller gains.

In order to test several different types of controllers, separate CAMPS modules were created for the open loop version of each module and versions of the module with different controllers. For example, for modeling a 7 state synchronous machine there exists an SM7State module and a separate SM7State Control module. There are also three models for modeling the fundamental model of a synchronous machine that are called SMFundamental, SMFundamentalGc, and SMFundamentalGcEc, which are the open loop model, the fundamental model with governor control, and the fundamental model with governor control and exciter control. The properties between the two SM7State modules and the properties between the three SMFundamentalModules are identical except for their controller gains, controllable inputs, controllable input names, control input equations, setpoints, setpoint outputs, and setpoint output equations. The same open loop dynamics is copied and pasted within each module.

To allow for more modularity, a new wrapper is created that allows for the separation of control from the physical dynamics and a new controller object. This wrapper contains only the open loop dynamics, but takes in a controller argument in its constructor. A controller object is defined by specifying values for controller gains, controllable inputs, controllable input names, control input equations, setpoints, setpoint outputs, and setpoint output equations. This controller can then be inputted into the wrapper module which applies the controller to the open loop model. Creating this allows for many types of controllers to be used with the same open-loop physical dynamics module rather than having to copy and paste code to create many different modules.

Within a CAMPS scenario file, the user instantiates any modules to be used within the system and specifies which other modules it is connected to. Based on this information, CAMPS will create a Power System object of the interconnected system [17]. Several functions can be called on this Power System object. This includes creating the state space equations of the interconnected system, doing linearized analysis of the system around an operating point, solving for the system equilibrium, and simulating the system state trajectory [17].

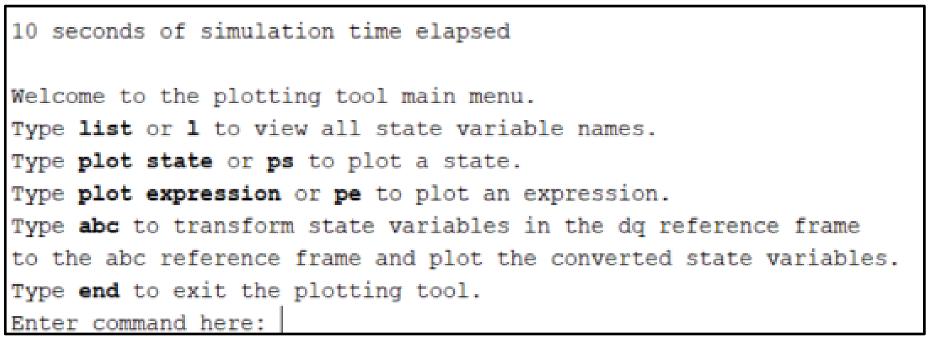
2.3 Plotting the System Dynamics

A key feature of CAMPS is seeing the system trajectory of state variables over time. This simulates the time response of the dynamic equations of the interconnected system written in state space form. Previous versions of CAMPS had a plotting tool that only allowed for the plotting of state variables. In order to plot the state variables, the name of the state variable had to be typed in exactly, but there was not an intuitive way for the user to know the names of the state variables that they could plot. Additionally, the plotting tool did not allow for the plotting of expressions in terms of state variables. In order to plot expressions in terms of state variables, the user would have to follow a confusing, non-user-friendly series of steps. They would need to navigate to the Results/SimulateSystemTrajectory folder, load in variables from a .mat file, and then manually plot the variables themselves by writing their own MATLAB code [17].

To simplify this process, I developed a new plotting tool with additional functionality. I added a “list” command that displayed all the state variables that had been calculated and could be plotted. I also added in a “plot expression” command that allowed for the plotting of expressions for state variables. The new plotting tool also retained the old functionality with the command “plot state”, which plots the trajectory for a single state variable. Section 2.3.1 will show how a user can use the new plotting tool within CAMPS.

2.3.1 Example of plotting the state trajectory

A user can call the "PrintMFileAndSolveSimulate" command within a CAMPS scenario file to simulate the trajectory of state variables. Once the simulation end time is reached, the plotting tool main menu opens. A sample usage screenshot is shown in Figure 2-2.

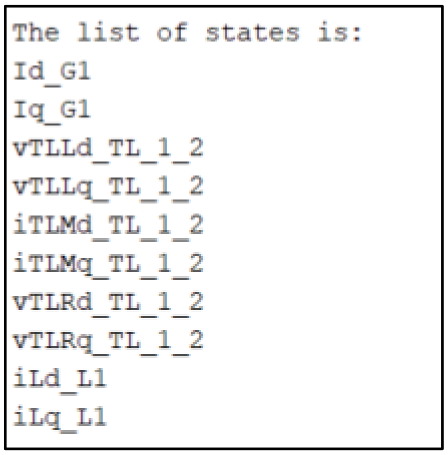


```
10 seconds of simulation time elapsed

Welcome to the plotting tool main menu.
Type list or l to view all state variable names.
Type plot state or ps to plot a state.
Type plot expression or pe to plot an expression.
Type abc to transform state variables in the dq reference frame
to the abc reference frame and plot the converted state variables.
Type end to exit the plotting tool.
Enter command here:
```

Figure 2-2 Plotting tool main menu

The user can choose to view all state variable names, plot a state variable, plot one or more mathematical expressions in terms of state variables, or exit the plotting tool. If the user types "**list**" or "**l**", the user can view the names of all state variables that can be plotted. Then the plotting tool will return the main menu. An example screenshot is shown in Figure 2-3.



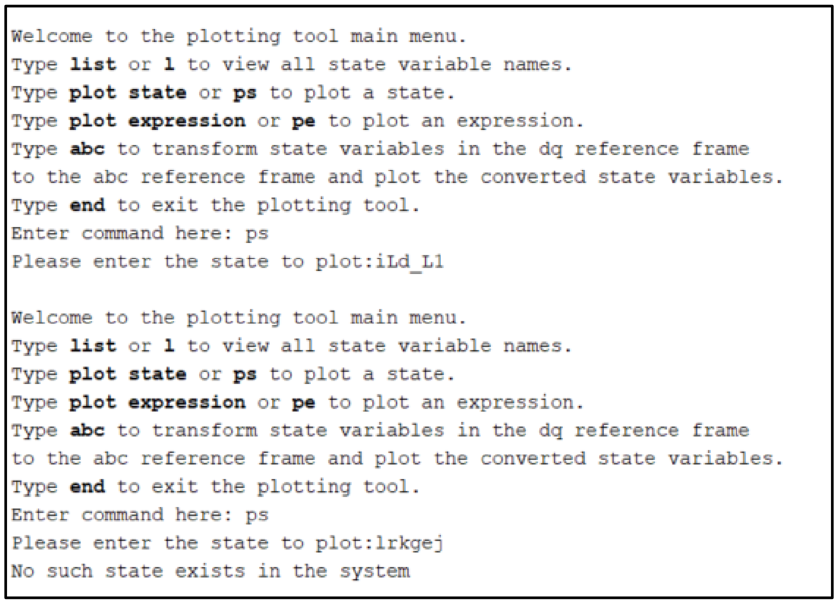
```
The list of states is:
Id_G1
Iq_G1
vTLLd_TL_1_2
vTLLq_TL_1_2
iTLMd_TL_1_2
iTLMq_TL_1_2
vTLRd_TL_1_2
vTLRq_TL_1_2
iLd_L1
iLq_L1
```

Figure 2-3 Plotting tool list of states

If the user types "**plot state**" or "**ps**", the user can plot a single state variable. Another prompt appears waiting for the user to input the state to plot.

- If the state entered by the user is a part of the interconnected state space, the state trajectory is plotted. The plotting tool will then return the main menu.
- If the state entered by the user is not a part of interconnected state space, it will print the message ‘No such state exists in the system’. The plotting tool will then return the main menu.

An example screenshot is shown in Figure 2-4. The plot produced based on the commands in Figure 2-4 is shown in Figure 2-5.



```
Welcome to the plotting tool main menu.
Type list or l to view all state variable names.
Type plot state or ps to plot a state.
Type plot expression or pe to plot an expression.
Type abc to transform state variables in the dq reference frame
to the abc reference frame and plot the converted state variables.
Type end to exit the plotting tool.
Enter command here: ps
Please enter the state to plot:iLd_L1

Welcome to the plotting tool main menu.
Type list or l to view all state variable names.
Type plot state or ps to plot a state.
Type plot expression or pe to plot an expression.
Type abc to transform state variables in the dq reference frame
to the abc reference frame and plot the converted state variables.
Type end to exit the plotting tool.
Enter command here: ps
Please enter the state to plot:lrkgej
No such state exists in the system
```

Figure 2-4 Plotting tool plot state

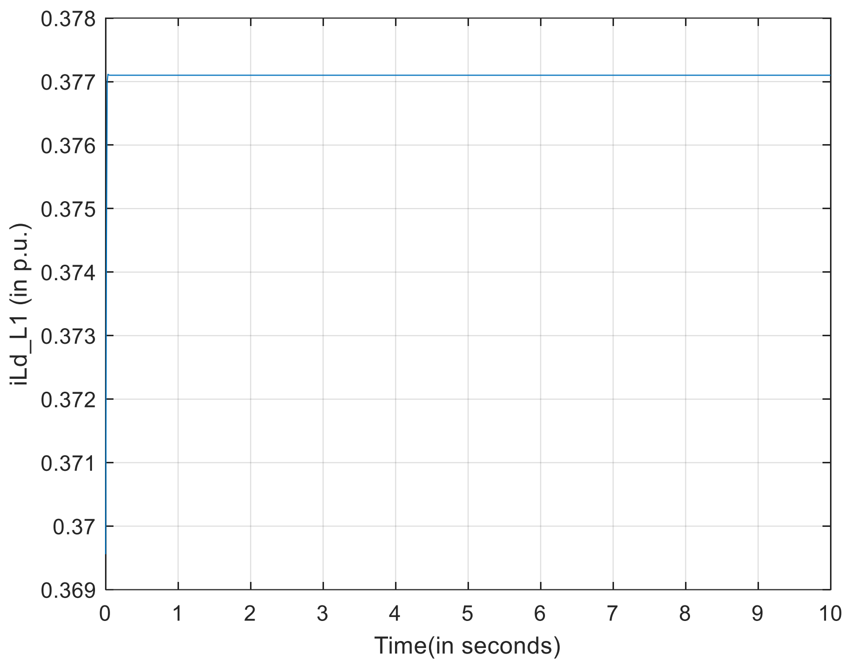


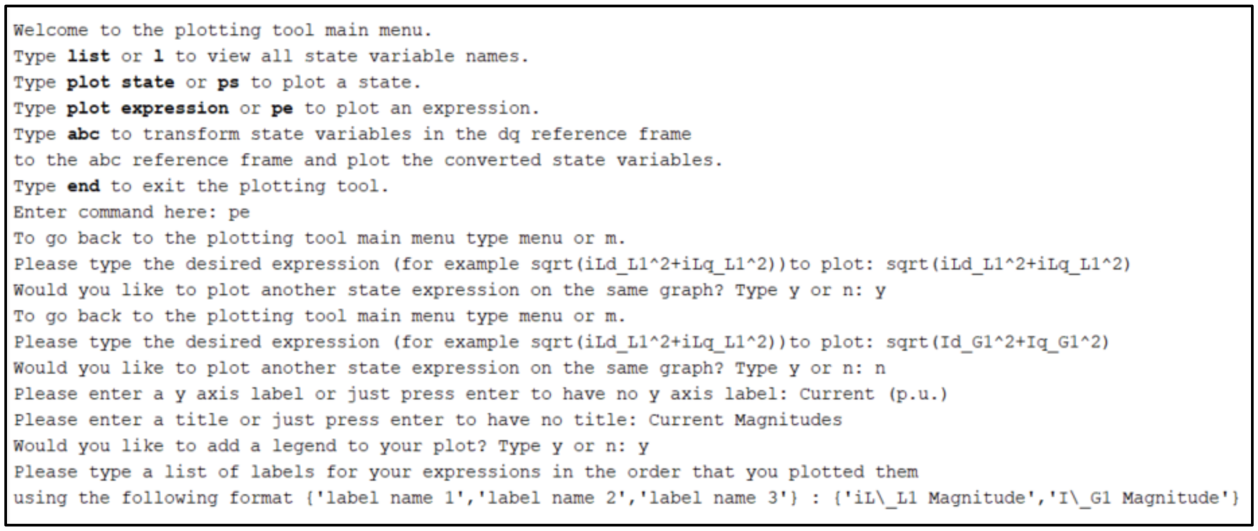
Figure 2-5 Plot produced by plotting tool "plot state" command

If the user types "**plot expression**" or "**pe**", the user can plot one or more mathematical expressions in terms of state variables.

- Another prompt appears waiting for the user to input the state variable expression to plot. Basic mathematical operators (+,-,/,*,^,sqrt(),etc.) can be used in these expressions. One example expression would be sqrt(iLd_L1^2+iLq_L1^2), which computes the current magnitude of the load current.
- Next, the user will be prompted to choose if they would like to plot an additional expression on the same graph. The user should type either "y" for yes or "n" for no. If the user types "y", they will be prompted to enter an additional expression.
- After this, the user will be prompted to enter a label for the y axis of the plot. If the user does not wish to have a y-axis label they can just press enter.

- The user will now be prompted to either enter a title for the plot, or the user can press enter to skip this.
- Then the user can choose to add a legend to the plot. If they say yes, they should enter a list of strings of expression names in the following format: {'label name 1','label name 2','label name 3'} (Note that some special characters such as '_' need to use the escape character '\' before it and be written as '_' to be displayed properly)
- The figure with the plot is now finished being created, and the user will return to the plotting tool main menu.

A sample usage screenshot is shown in Figure 2-6. The commands shown in Figure 2-6 produce the plot in Figure 2-7.



```
Welcome to the plotting tool main menu.
Type list or l to view all state variable names.
Type plot state or ps to plot a state.
Type plot expression or pe to plot an expression.
Type abc to transform state variables in the dq reference frame
to the abc reference frame and plot the converted state variables.
Type end to exit the plotting tool.
Enter command here: pe
To go back to the plotting tool main menu type menu or m.
Please type the desired expression (for example sqrt(iLd_L1^2+iLq_L1^2))to plot: sqrt(iLd_L1^2+iLq_L1^2)
Would you like to plot another state expression on the same graph? Type y or n: y
To go back to the plotting tool main menu type menu or m.
Please type the desired expression (for example sqrt(iLd_L1^2+iLq_L1^2))to plot: sqrt(Id_G1^2+Iq_G1^2)
Would you like to plot another state expression on the same graph? Type y or n: n
Please enter a y axis label or just press enter to have no y axis label: Current (p.u.)
Please enter a title or just press enter to have no title: Current Magnitudes
Would you like to add a legend to your plot? Type y or n: y
Please type a list of labels for your expressions in the order that you plotted them
using the following format {'label name 1','label name 2','label name 3'} : {'iL\_L1 Magnitude','I\_G1 Magnitude'}
```

Figure 2-6 Plot expression using plotting tool

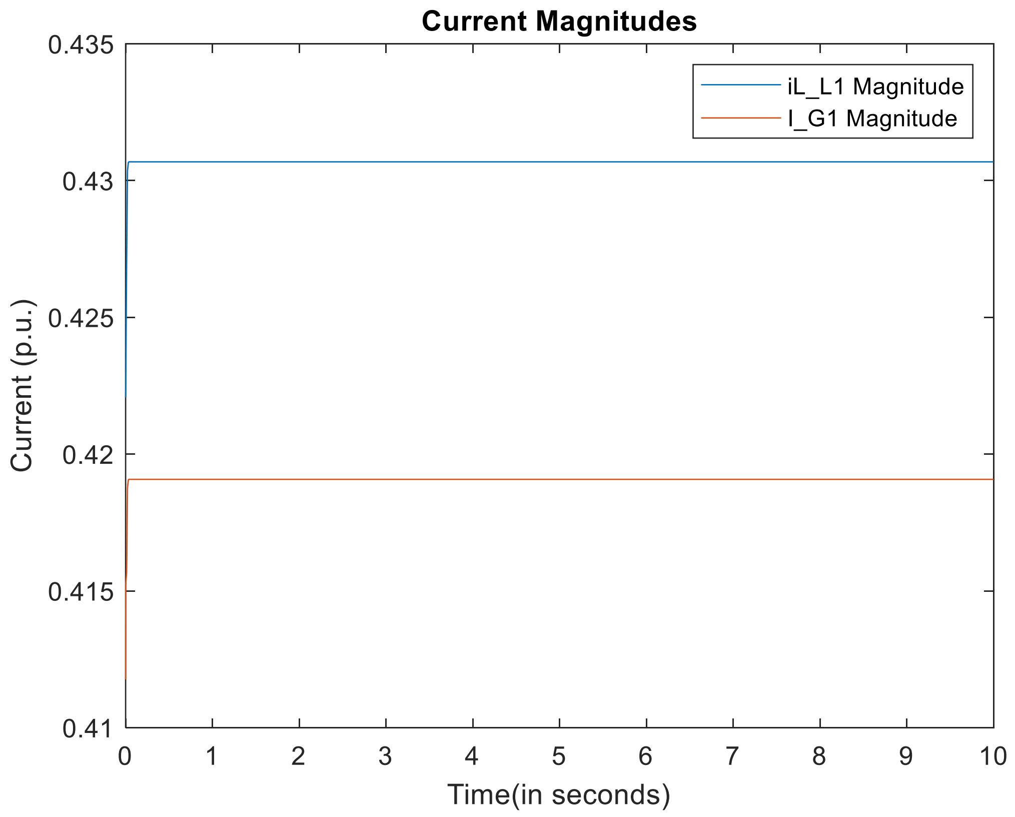


Figure 2-7 Plot of "plot expression" using plotting tool

Finally, if the word **'end'** is typed, the user can exit plotting tool.

2.4 Reference frame mapping from the dq-reference frame to the abc-reference frame

Electrical power systems traditionally use three-phase voltages that are sinusoids that are 120 degrees phase shifted from each other [7]. This is known as abc voltage and is shown in Figure 2-8. It is difficult to develop methods of control for the abc quantities of voltage and current [6]. To make it easier to control, the abc quantities can be converted to the dq (direct-quadrature) reference frame. When the abc quantities are in steady-state, the converted dq values will be constant [6].

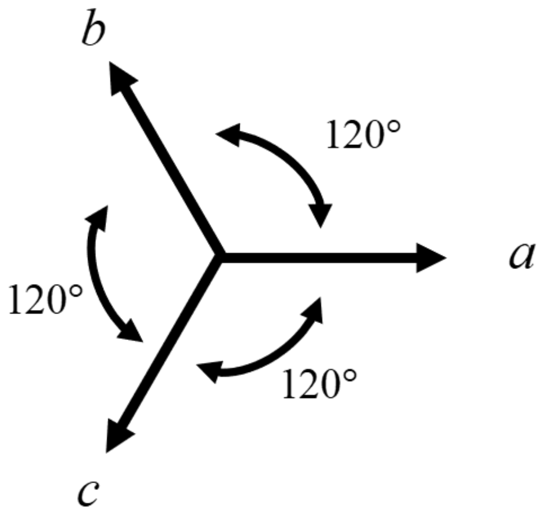


Figure 2-8 The abc-reference frame

To convert from abc to dq, the abc quantities must first be transformed to the stationary alpha-beta reference frame, which is shown in Figure 2-9. Then the alpha-beta quantities can be transformed to the dq-reference frame as shown in Figure 2-10. The dq-reference frame rotates at a speed $\frac{d\phi}{dt}$ and is at angle ϕ relative to the stationary reference frame alpha-beta [7]. The dq-reference frame’s rotational speed, $\frac{d\phi}{dt}$, should be consistent for all electrical components in the system.

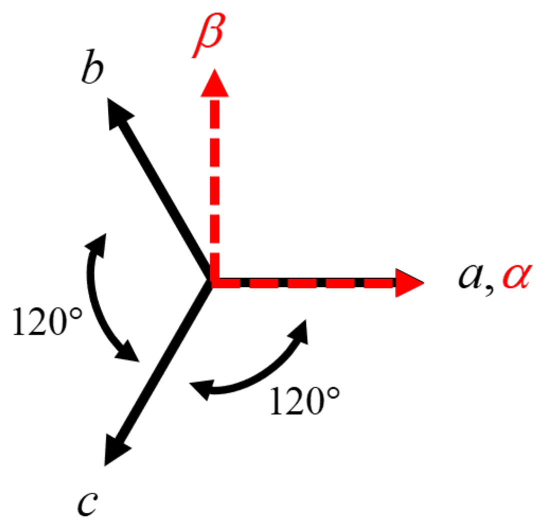


Figure 2-9 Going from abc-reference frame to alpha-beta reference frame

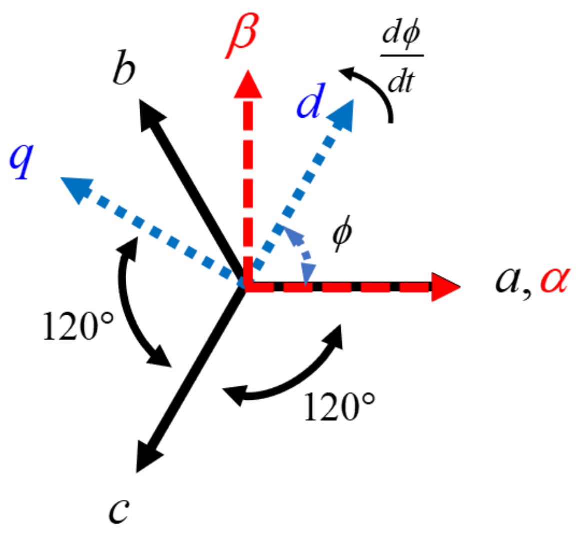


Figure 2-10 Going from abc-reference frame to dq reference frame

To map from abc to dq, Park's transformation is used. This is defined in equation (2.4) from [6].

$$x_{dq0} = \underbrace{\sqrt{\frac{2}{3}}\begin{bmatrix} \cos\phi & \cos\left(\phi-\frac{2\pi}{3}\right) & \cos\left(\phi+\frac{2\pi}{3}\right) \\ \sin\phi & -\sin\left(\phi-\frac{2\pi}{3}\right) & -\sin\left(\phi+\frac{2\pi}{3}\right) \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}}_{P(\phi)} \begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} \tag{2.4}$$

If the c phase is equal to -a plus -b, the transformation can be simplified to be equation (2.5) from [6].

$$\begin{bmatrix} x_d \\ x_q \end{bmatrix} = \underbrace{\sqrt{\frac{2}{3}} \begin{bmatrix} \cos\phi - \cos\left(\phi + \frac{2\pi}{3}\right) & \cos\left(\phi - \frac{2\pi}{3}\right) - \cos\left(\phi + \frac{2\pi}{3}\right) \\ -\sin\phi + \sin\left(\phi + \frac{2\pi}{3}\right) & -\sin\left(\phi - \frac{2\pi}{3}\right) + \sin\left(\phi + \frac{2\pi}{3}\right) \end{bmatrix}}_{P(\phi)} \begin{bmatrix} x_a \\ x_b \end{bmatrix}$$

(2.5)

CAMPS v1.0 uses the dq-reference frame for calculating all of its state variables. However, it did not previously have a feature for mapping dq quantities back to the abc-reference frame. In this thesis, an additional feature in CAMPS was added to have this additional functionality. The variables x_a, x_b, x_c are calculated using the equations (2.6) and (2.7) [6].

$$\begin{bmatrix} x_a \\ x_b \end{bmatrix} = \left[\underbrace{\sqrt{\frac{2}{3}} \begin{bmatrix} \cos\phi - \cos\left(\phi + \frac{2\pi}{3}\right) & \cos\left(\phi - \frac{2\pi}{3}\right) - \cos\left(\phi + \frac{2\pi}{3}\right) \\ -\sin\phi + \sin\left(\phi + \frac{2\pi}{3}\right) & -\sin\left(\phi - \frac{2\pi}{3}\right) + \sin\left(\phi + \frac{2\pi}{3}\right) \end{bmatrix}}_{P(\phi)} \right]^{-1} \begin{bmatrix} x_d \\ x_q \end{bmatrix}$$

(2.6)

$$x_c = -(x_a + x_b) \tag{2.7}$$

I created a CAMPS scenario file that connected an infinite bus to a resistive and inductive load. I tried applying the dq to abc converter to the state variables produced from the system trajectory. Through experimental results, I found that the dq to abc converter yielded strange looking plots when an adaptive timestep with minimum step size 10^{-2} seconds had been used in the solver for calculating the system trajectory of the dq variables. Figure 2-11 and the zoomed in version of the plot in Figure 2-12 show an example of this. This was fixed by applying MATLAB's linear interpolation function to the vector of the dq variables to make them have a fixed time-step of 10^{-4} seconds. Then, when the dq to abc converter was applied to the linear interpolated vector of dq variables, the plots looked like the expected three phase 60 Hz sinusoids. The fixed figure is shown in Figure 2-13 and the zoomed-in version of the plot in Figure 2-14. An example usage of the dq to abc converter is shown in section 2.4.1.

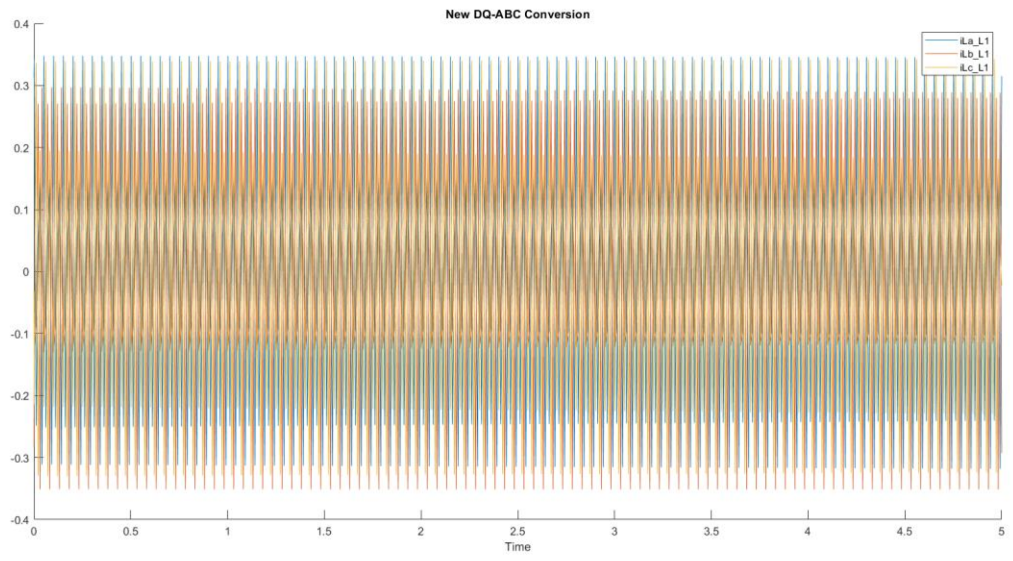


Figure 2-11 Zoomed-out plot from dq to abc conversion no linear interpolation

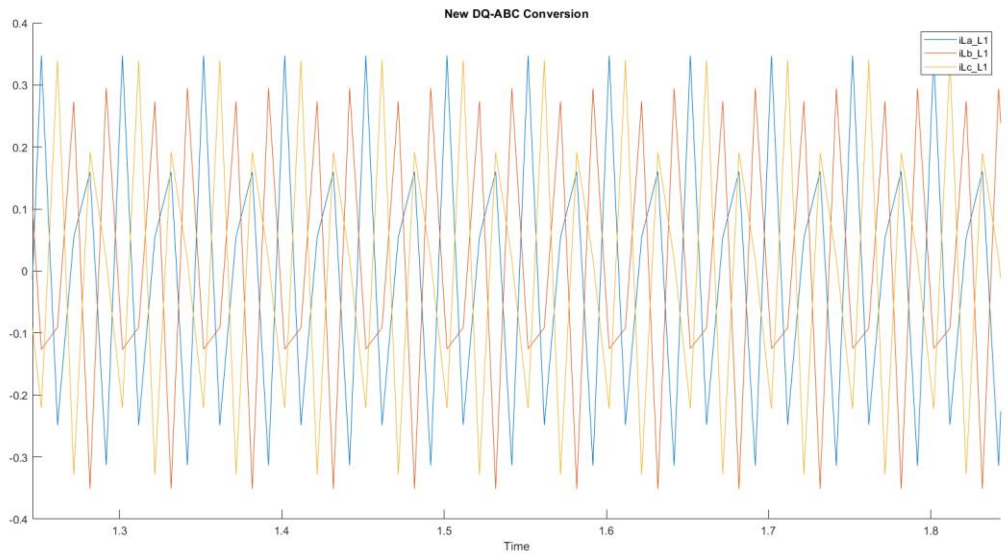


Figure 2-12 Zoomed-in plot from dq to abc conversion no linear interpolation

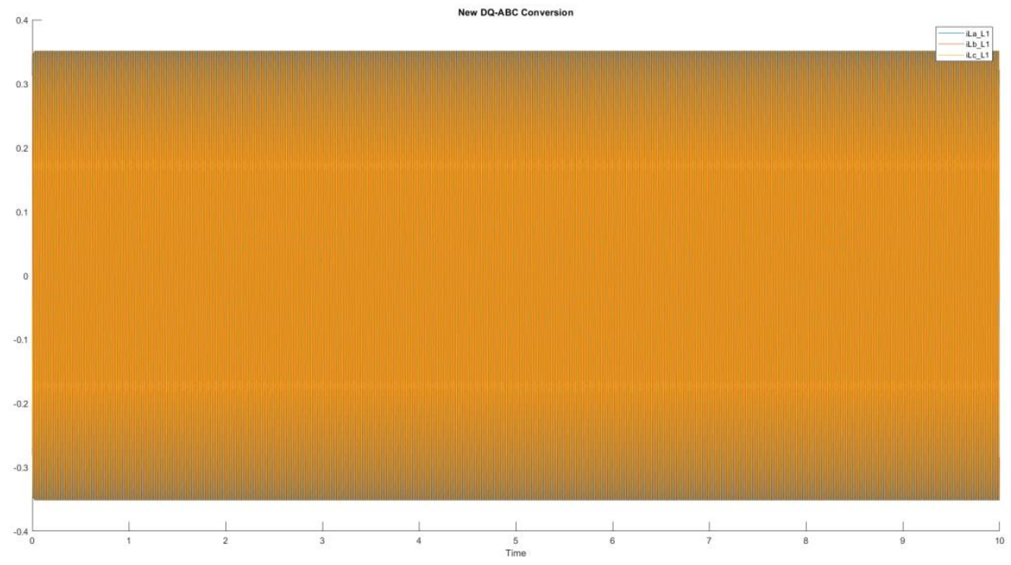


Figure 2-13 Zoomed out plot from dq to abc conversion with linear interpolation

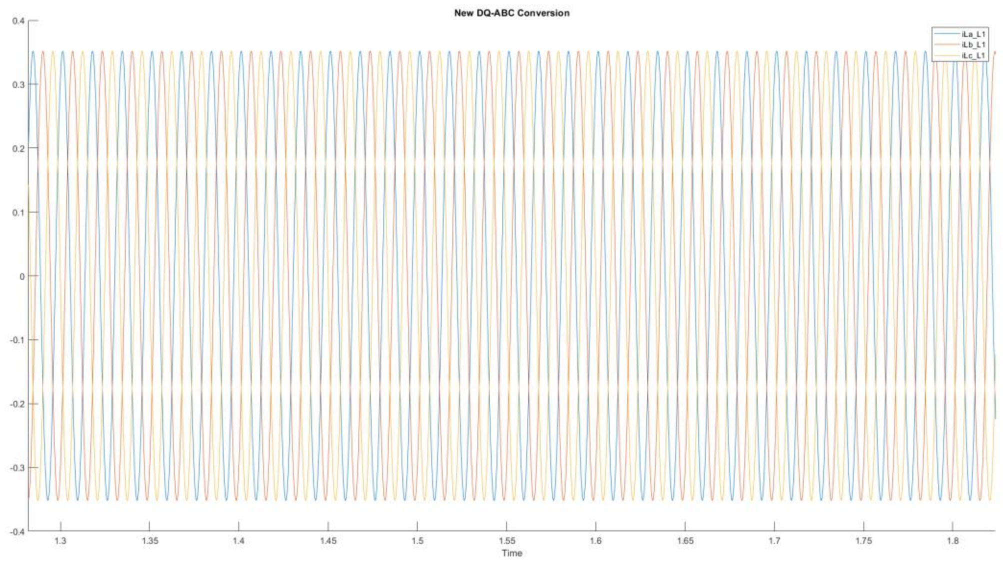


Figure 2-14 Zoomed in plot from dq to abc conversion with linear interpolation

2.4.1 Example of using the dq to abc converter

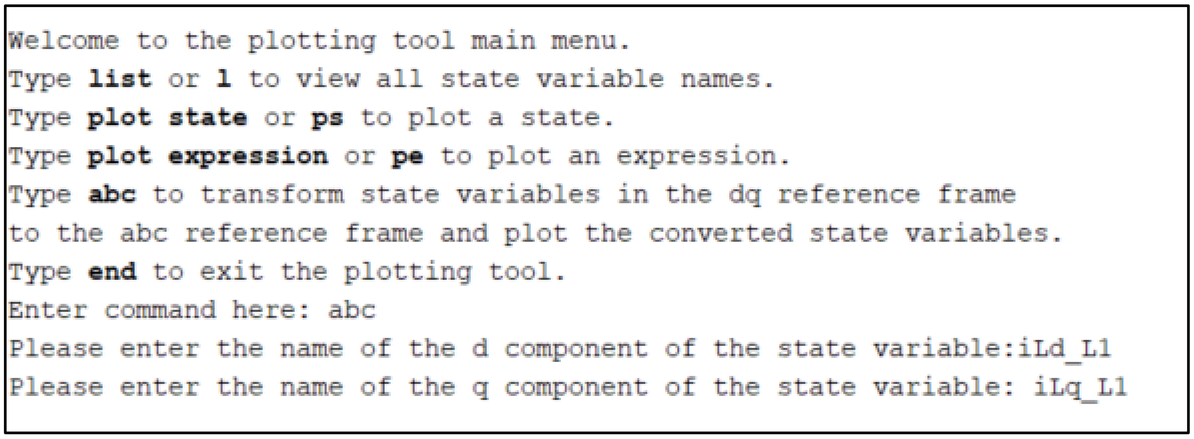
Within the plotting tool described in section 2.3, the user can also use the command “abc”. If the user types “abc”, the user can transform state variables in the dq-reference frame to the abc-reference frame using Park’s transformation. The plotting tool will then plot these converted abc state variables.

- Once the user types “abc”, another prompt appears asking for the name of the d component of the state variable. (For example, one valid input would be the state variable for the d component of the load current, iLd_L1.)
- Next, the user will be prompted to type the name of the q component of the state variable. (For example, one valid input would be iLq_L1.)
- This will produce 4 new plots. The first plot is the a, b, and c components all plotted together. The second plot has just the a component. The third plot is of the b component. The fourth plot is of the c component. (Note, the user may want to zoom in on these plots

since these are plots of sinusoidal outputs with a fast frequency).

- The user will then return to the plotting tool main menu.

A sample usage screenshot is shown in Figure 2-15.



```
Welcome to the plotting tool main menu.
Type list or l to view all state variable names.
Type plot state or ps to plot a state.
Type plot expression or pe to plot an expression.
Type abc to transform state variables in the dq reference frame
to the abc reference frame and plot the converted state variables.
Type end to exit the plotting tool.
Enter command here: abc
Please enter the name of the d component of the state variable:iLd_L1
Please enter the name of the q component of the state variable: iLq_L1
```

Figure 2-15 Dq to abc conversion using plotting tool

The resulting plots are shown below in Figure 2-16.

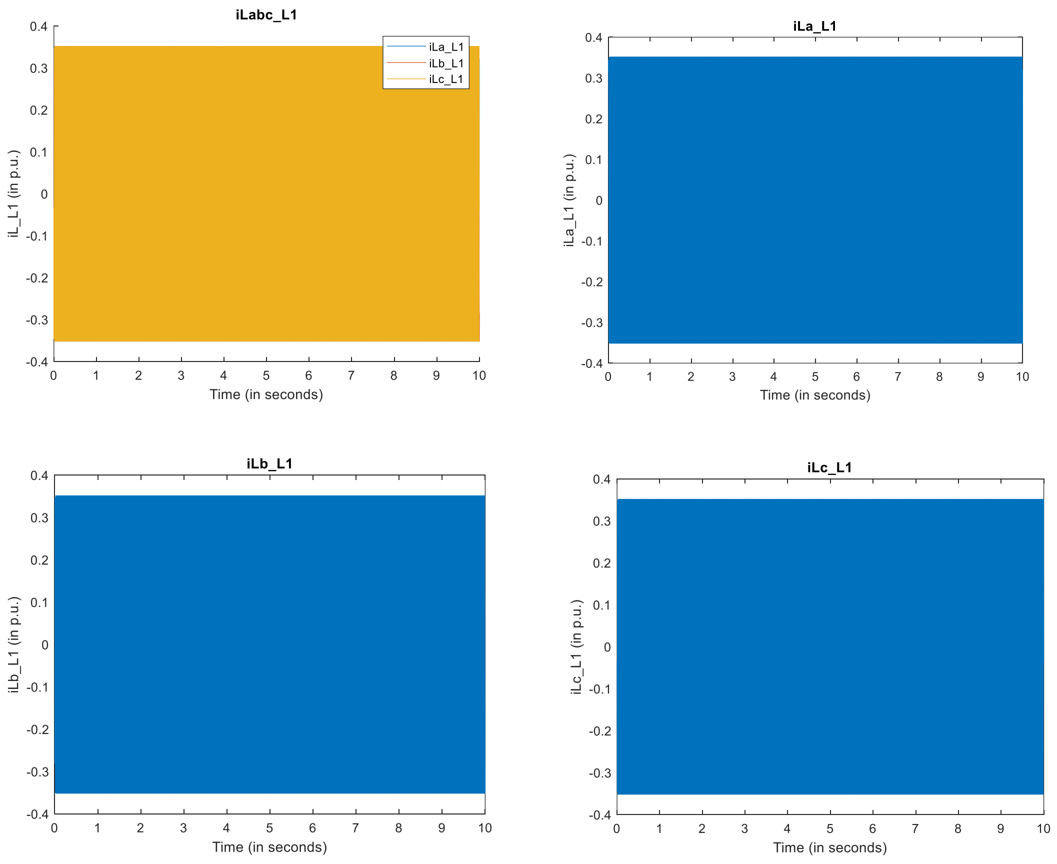


Figure 2-16 Plots from dq to abc converter

The user can zoom in on these plots to further understand the system trajectory of the abc components. Figure 2-17 shows an example of zooming in on the iLabc_L1 plot.

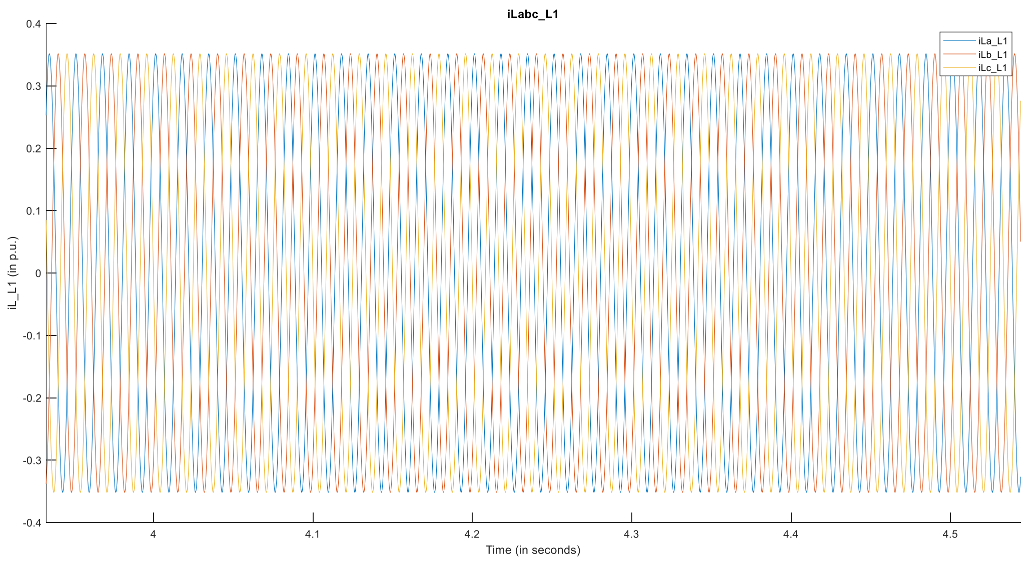


Figure 2-17 Zoomed in plot of dq to abc converter

3 Simulations of a Proposed Energy Space Controller

While CAMPS communicated voltages and currents between modules, a recent type of modeling uses the exchange of energy variables. In [14] a new time-domain model using energy variables and a new nonlinear controller is proposed for inverter control. This nonlinear controller strives to ensure high quality of service by minimizing distortions of voltages and current in the event of disturbances. As a proof of concept, Xia Miao created a Simulink model for modeling a solar inverter connected to a synchronous machine. Both the solar inverter and synchronous machine also have RL loads connected to them. The overall setup of the system is shown in Figure 3-1.

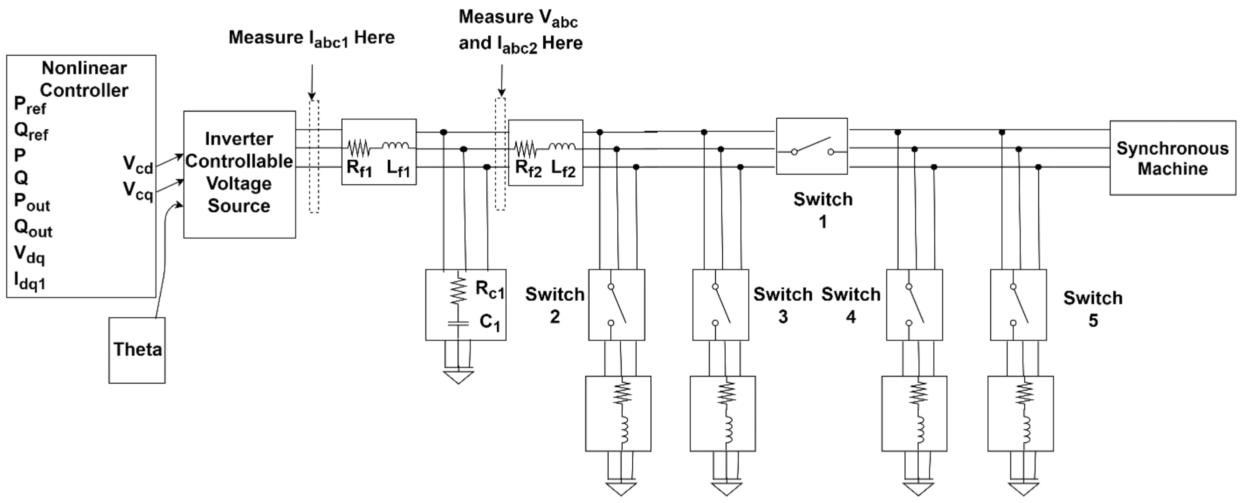


Figure 3-1 Diagram of solar inverter connected to synchronous machine as posed in [14]

The nonlinear controller has inputs of the power setpoint, reactive power setpoint, power, reactive power, output power, output reactive power, the d and q components of voltage, and the d and q components of current 1, which are written as $P_{ref}, Q_{ref}, P, Q, P_{out}, Q_{out}, v_{dq}$, and i_{dq1} respectively. The variable v_{dq} is calculated by applying Park's transformation to V_{abc} using equation (2.4) and converting to per-units. Similarly, i_{dq1} applies Park's transformation to I_{abc1} and converts to per-units , and i_{dq2} applies Park's transformation to I_{abc2} converts to per-units. Per-units is a unitless dimension. To convert the voltage to per-units, the voltage is divided by the base voltage, V_{base}. To convert the currents to per units, the current is divided by the base current, I_{base}. In the Simulink simulation V_{base} equals 208V and the power base, S_{base}, equals 6.25 kVA. The I_{base} can be calculated by dividing V_{base} from S_{base} as shown in equation (3.1)

$$I_{base} = \frac{S_{base}}{V_{base}} \tag{3.1}$$

When applying Park's transformation using equation (2.4), θ is substituted in for ϕ. Since the power system should provide 60 Hz voltages and currents, θ is defined as the 60Hz frequency multiplied by 2π and the time in seconds, t. This is shown in equation (3.2).

$$\theta = 2\pi \times 60t \tag{3.2}$$

P and Q are calculated using equations (3.3) and (3.4) from [14]. The output power and output reactive power, P_{out} and Q_{out}, are calculated using equations (3.5) and (3.6) from [14].

$$P = v_d i_{d1} + v_q i_{q1} \tag{3.3}$$

$$Q = v_q i_{d1} - v_q i_{q1} \tag{3.4}$$

$$P_{out} = v_d i_{d2} + v_q i_{q2} \tag{3.5}$$

$$Q_{out} = v_q i_{d2} - v_q i_{q2} \tag{3.6}$$

In order to compute the setpoints P_{ref} and Q_{ref}, equations (3.7) and (3.8) from [14] are used. K_V is the controller gain and C is the capacitance from the capacitor labeled C1 in Figure 3-1.

V is the voltage magnitude of v_{dq} in per-units and is calculated using equation (3.9). V_{ref} is the magnitude of the voltage reference point. The variable ω_0 is defined as $2\pi \times 60$.

$$P_{ref} = P_{out} - K_V \left(V^2 - V_{ref}{}^2\right) \tag{3.7}$$

$$Q_{ref} = Q_{out} - CV^2\omega_0 \tag{3.8}$$

$$V^2 = \sqrt{v_d{}^2 + v_q{}^2} \tag{3.9}$$

Based on these inputs, the nonlinear controller computes output signals v_{cd} and v_{cq}.

Xia Miao created a Simulink model for testing the nonlinear controller. In his simulations, he used an average model for converting the v_{cd} and v_{cq} inputs into physical abc voltages. This average model block occurs at the block labeled “Inverter Controllable Voltage Source” in Figure 3-1. In the test simulation, switch 1 that connects the inverter and synchronous machine starts out open and closes after 2 seconds. Switches 2,3,4, and 5 also start out open. At 1.5 seconds, switch 3 closes, connecting the RL load.

Using the average model, the nonlinear controller appears to produce 60 Hz sinusoidal three phase voltages and currents and responds well to the transients introduced at 1.5 seconds and 2 seconds when the switch connecting the RL load closes and the switch connecting the invert to the synchronous machine close. The plots for the inverter side V_{abc}, I_{abc}, and power and reactive power are shown in Figure 3-2 and are further zoomed in from 1.4 seconds to 2.2 seconds in Figure 3-3.

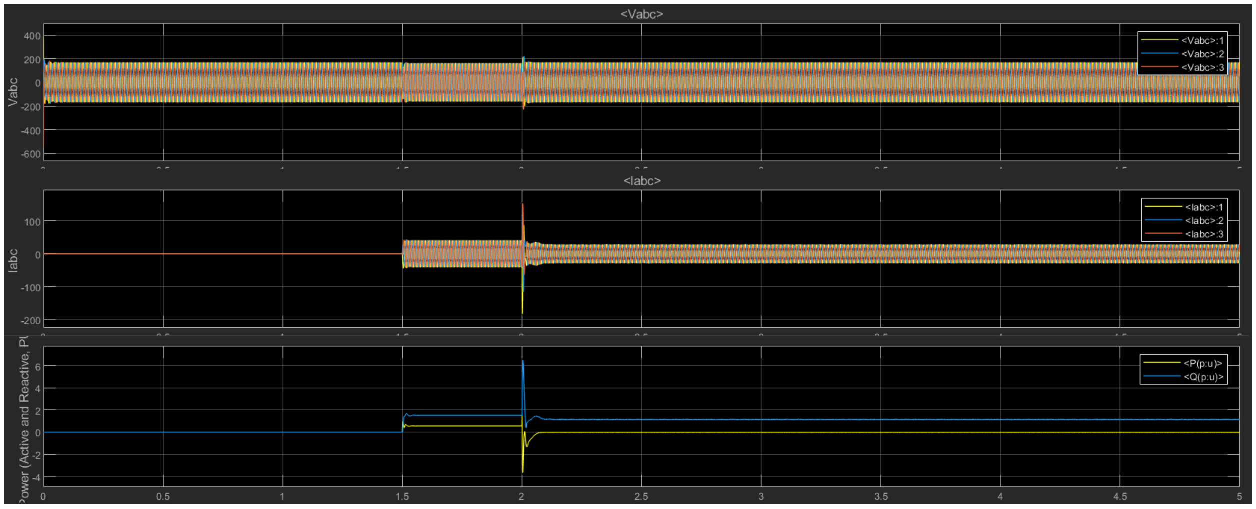


Figure 3-2 Plot of average model V_{abc}, I_{abc}, P, and Q

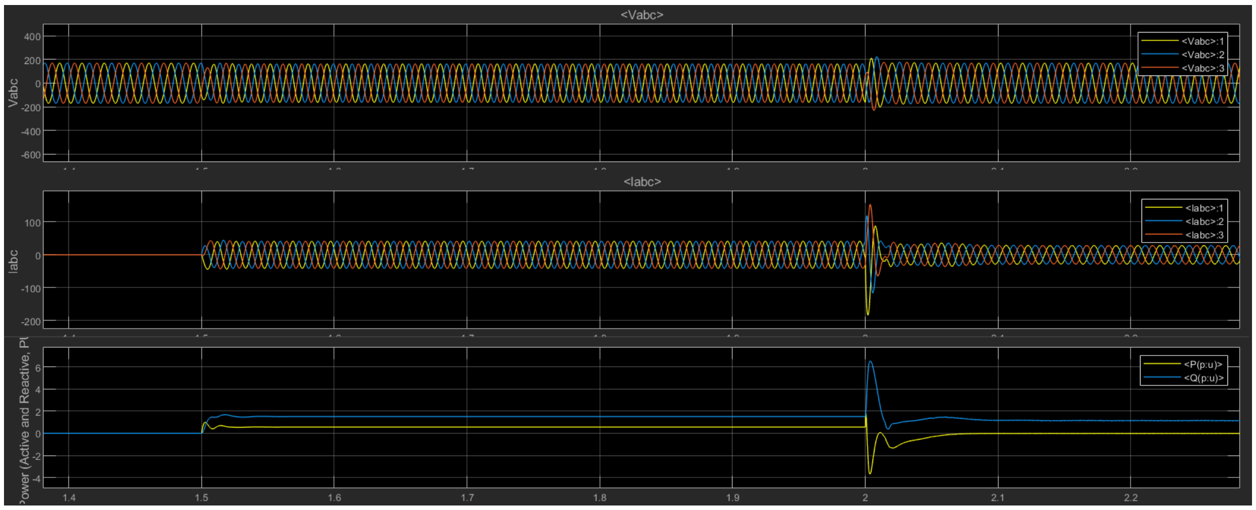


Figure 3-3 Zoomed in plot of average model V_{abc}, I_{abc}, P and Q

3.1 Average Model to Switching Model

To extend Xia Miao's Simulink model, several engineers from MIT Lincoln Labs wanted to test how the system behaved if the average model block was replaced with a switching model block. They created a new Simulink model with this change. The switching model block takes the control signals v_{cd} and v_{cq} and maps them to per-unitized voltages in the abc-reference frame. These per-unitized voltages can be referred to a $U_{abc,ref}$. Then, $U_{abc,ref}$ is inputted into a three-phase PWM (Pulse Width Modulation) generator with a carrier frequency of 25 kHz and a sample time of 50 microseconds. This PWM output is inputted into a three-phase two-level power converter with a nominal DC voltage. The three-phase two-layer power converted produces the abc voltages. These voltages and currents are then filtered by the LCL filter.

When this system was simulated, it yielded strange looking plots for V_{abc}, I_{abc}, power, and reactive power. These plots are shown in Figure 3-4. Rather than being sinusoidal, they looked more like constants with a ripple. In collaboration with EECS Masters student Premila Rowles, and advice and guidance from Professor Marija Ilic and Post-Doc, Pallavi Bharadwaj, we investigated why the switching model produced these plots.

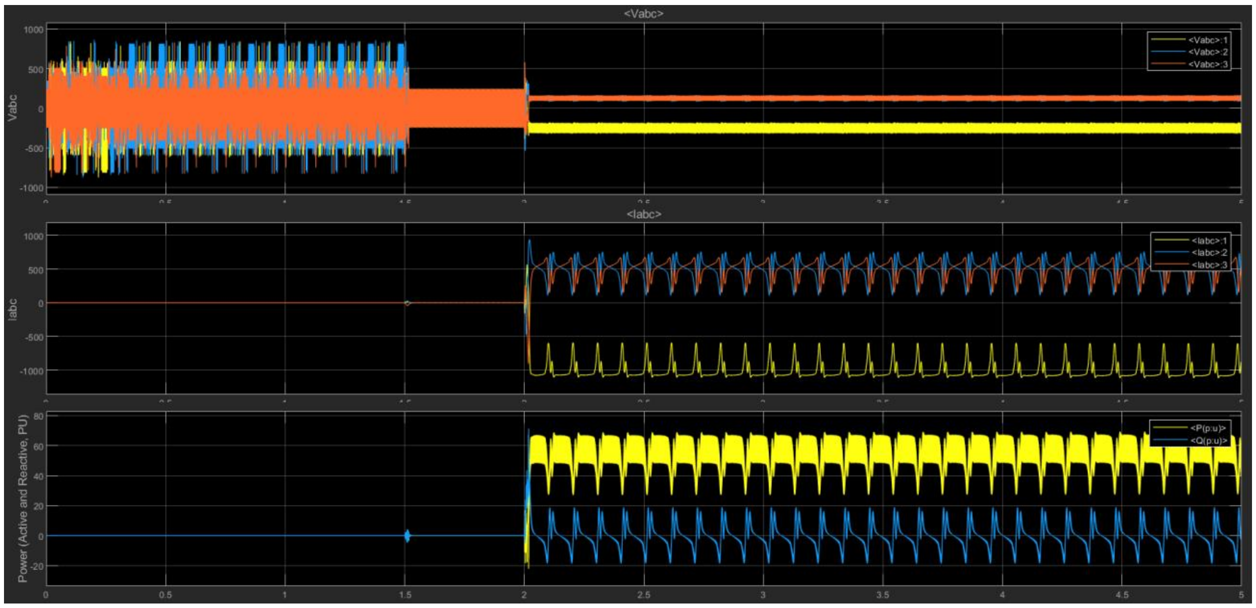


Figure 3-4 Original plots for moving to switching model

To start our investigation, Premila and I went back to the Simulink model that used the average model. We added an additional scope to view the v_{cd} and v_{cq} outputs of the controller.

This yielded the plot shown in Figure 3-5.

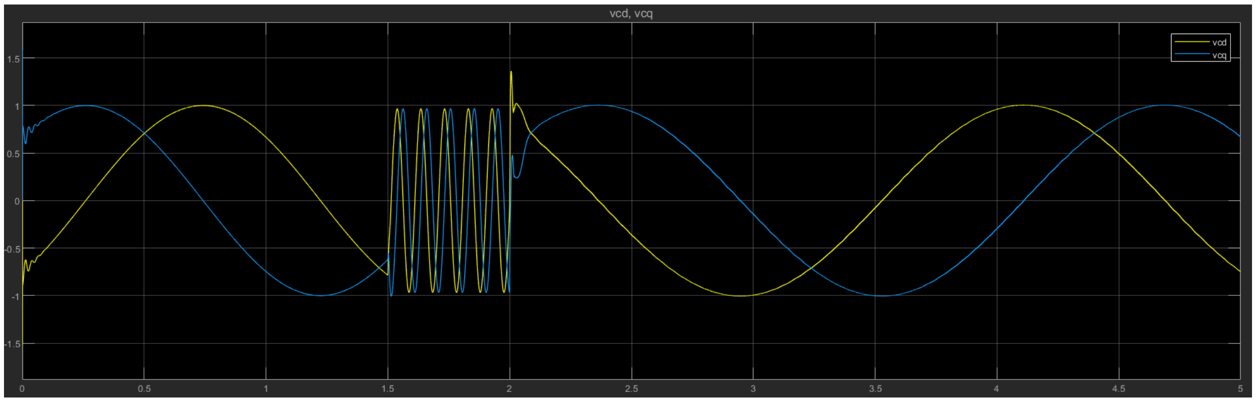


Figure 3-5 Plot of v_{cd} and v_{cq} signals using the average model

We were surprised to see that rather than being a constant value, v_{cd} and v_{cq} were sinusoidal. Looking closer at v_{cd} and v_{cq} we noticed that from 0 to 1.5 seconds they had a frequency of 0.5186 Hz, from 1.5 to 2 seconds (when switch 3 closes) it had a frequency of 9.626Hz, and from 2 to 5 seconds (after switch 1 closes) it had a frequency of 0.4327 Hz. We then used a Simulink block to measure the frequency of V_{abc}, and the resulting plot is shown in Figure 3-6.

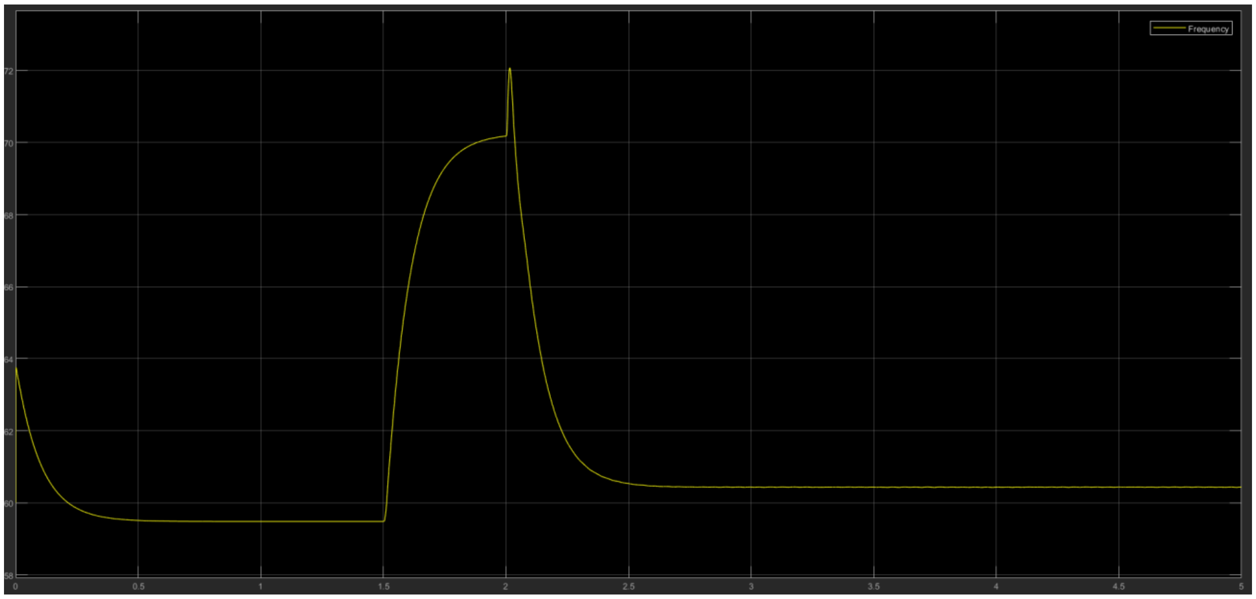


Figure 3-6 Plot of V_{abc} Frequency

While at first glance, V_{abc} had appeared to be 60 Hz, we realized that V_{abc} actually had a frequency of 59.48 Hz from 0 to 1.5 seconds, 70.1 Hz from 1.5 to seconds, and 60.43 Hz from 1.5 to 2 seconds. The difference between this measured frequency and 60Hz was equal to 0.52 Hz between 0 to 1.5 seconds, 10.1 Hz between 1.5 to 2 seconds, and 0.43 Hz between 1.5 to seconds. These differences were roughly the same as the frequency of the v_{cd} and v_{cq} signals. This made us realize that v_{cd} and v_{cq} were sinusoidal because the frequency of V_{abc} did not match the 60Hz frequency used in Park's transform from the abc-reference frame to the dq-reference frame.

To fix this issue, we tried using a PLL (Phase-Locked Loop) to measure the frequency of V_{abc} and using this measured frequency in Park's transform from the abc-reference frame to the dq-reference frame. This made v_{cd} and v_{cq} constant, as shown in the plot of v_{cd} and v_{cq} in Figure 3-7. However, it made the frequency much lower as shown in Figure 3-8. The frequency of V_{abc} dropped down all the way to 0 Hz.

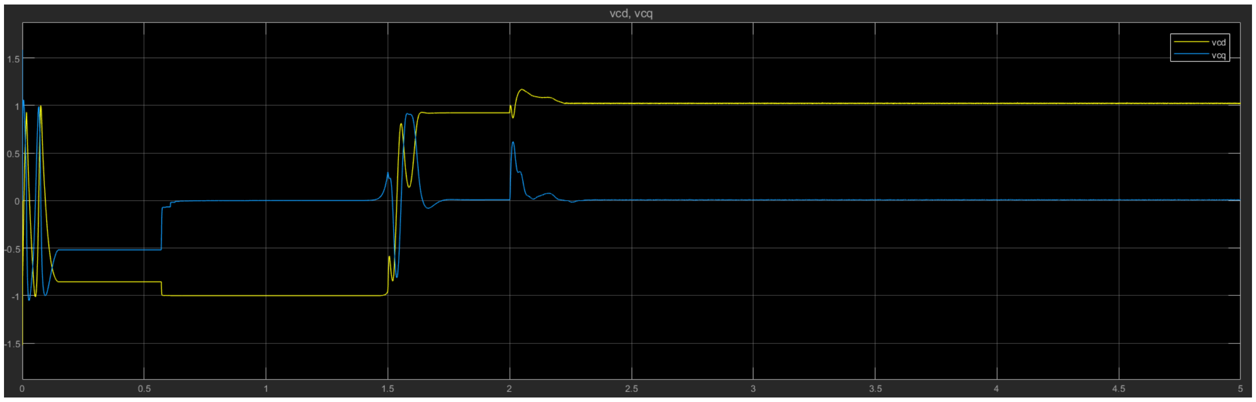


Figure 3-7 Plot of v_{cd} and v_{cq} using a PLL

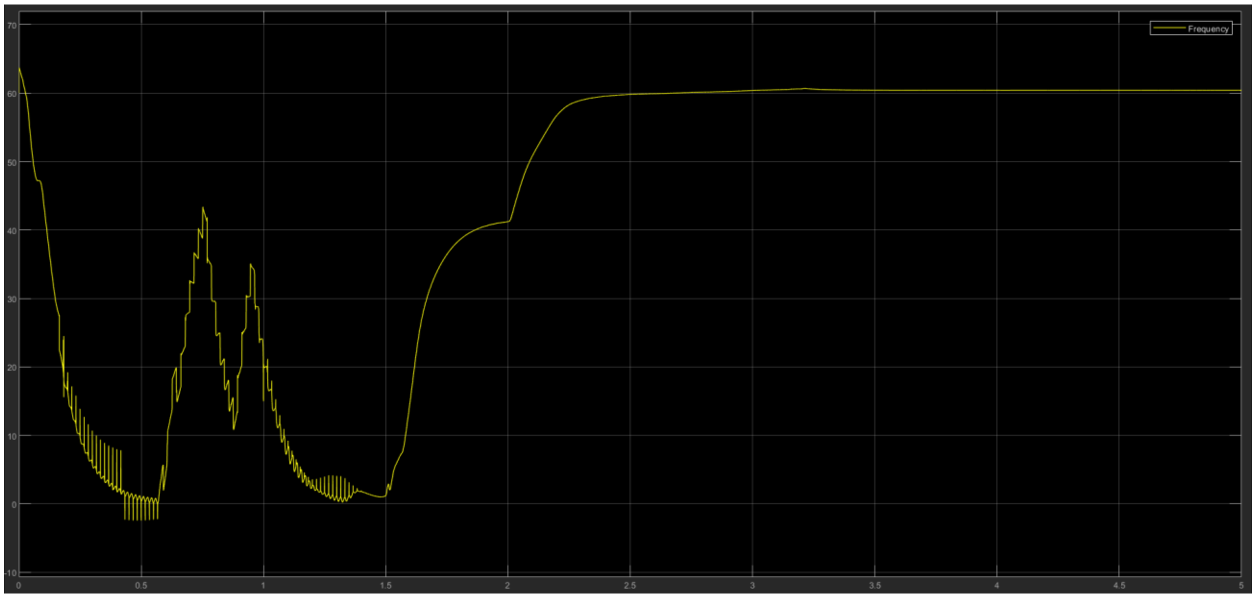


Figure 3-8 Frequency of V_{abc} using a PLL

3.2 Switching Model with V_{cdq} Inputs from Average Model

Next, we looked at feeding in the v_{cd} and v_{cq} signals produced by the average model into the switching model to further understand how changing the switching frequency and filter capacitance would affect the PWM's filtered the output voltages. We created a test system as shown in Figure 3-9.

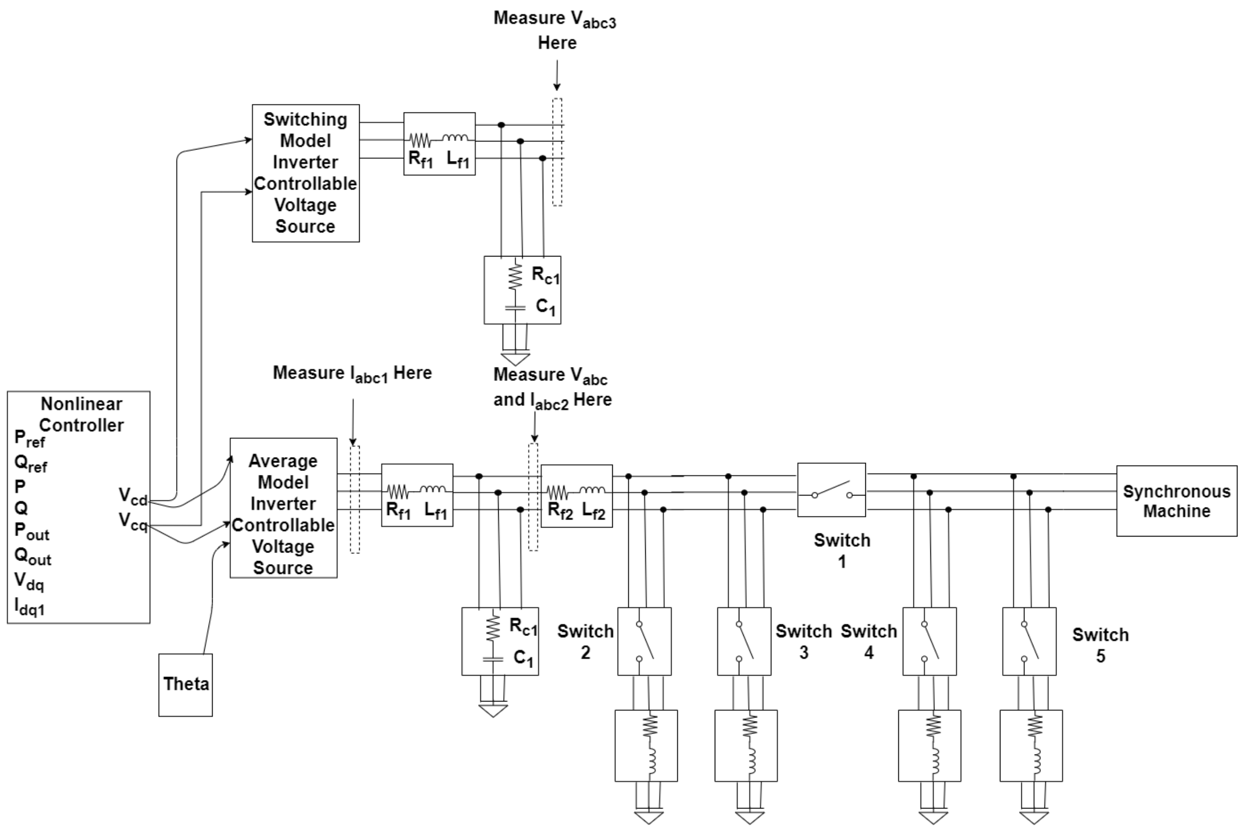


Figure 3-9 Use v_{cdq} signals created using average model as inputs to switching model [14]

Using this test system, we tried changing the PWM carrier frequency to 2.5 kHz. Figure 3-10 shows V_{abc3} produced using a carrier frequency of 25 kHz and Figure 3-11 shows the V_{abc3} produced using a carrier frequency of 2.5 kHz. Within each figure, the first plot below shows the V_{abc3} voltage from 0 to 5 seconds. The second zooms in to show the V_{abc3} voltage from 1.75 seconds to 2.25 seconds. The third is zoomed in further to show the V_{abc3} voltage from 1.93 seconds to 1.98 seconds.

Using the carrier frequency of 2.5 kHz we used different capacitor values for the capacitor C1 (labeled in Figure 3-9) that is used within the LCL filter. The original capacitance used was 30 microfarads. We first tried using one-tenth the capacitor value and ten times the capacitor value. Using one-tenth the capacitor value made the plots look worse, whereas using ten times the capacitor value made the plots look more sinusoidal. The best result found used a capacitor value of 150 microfarads, which was 5 times larger than the original capacitor used. Figure 3-12, Figure 3-13, and Figure 3-14 show the plots for the different capacitor values. Once again, within each figure the first plot below shows the V_{abc} voltage from 0 to 5 seconds. The second zooms in to show the V_{abc} voltage from 1.75 seconds to 2.25 seconds. The third is zoomed in further to show the V_{abc} voltage from 1.93 secods to 1.98 seconds.

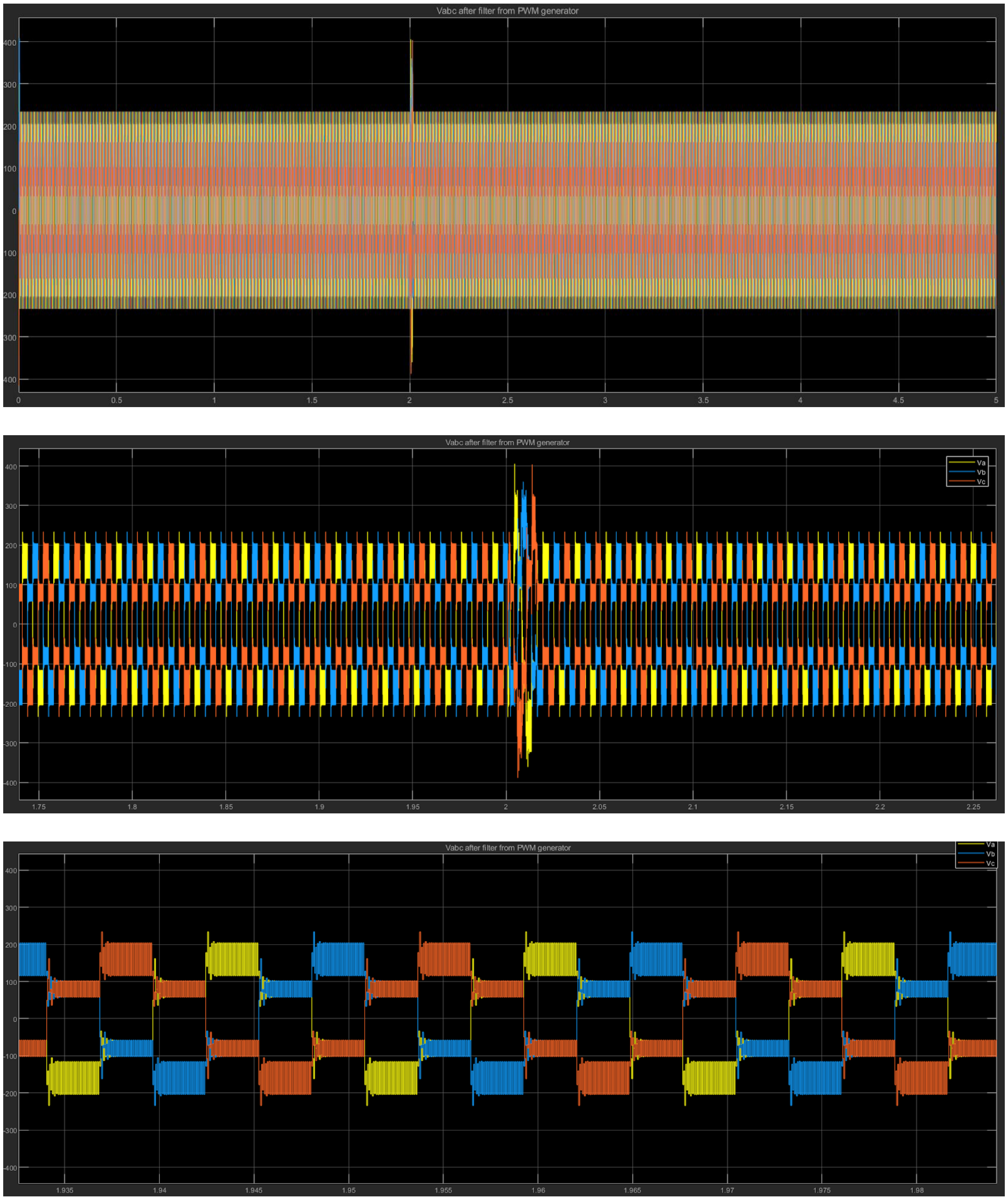
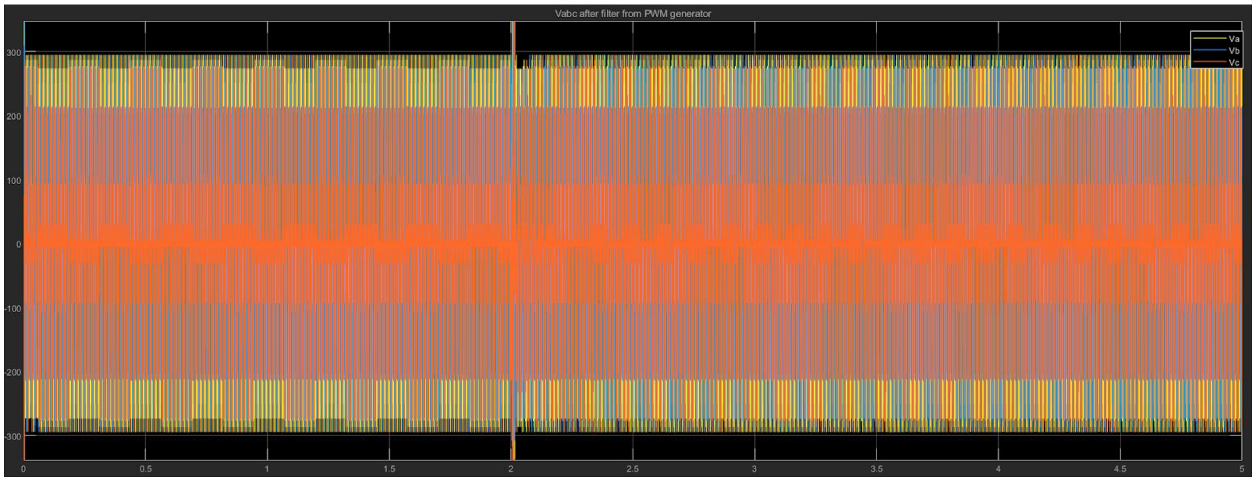
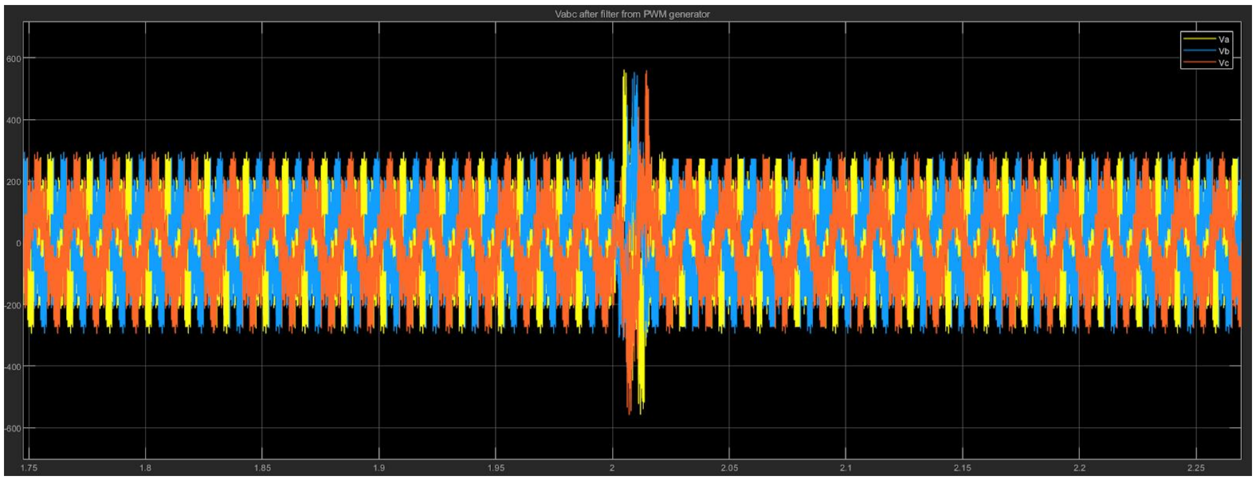


Figure 3-10 V$_{abc}$ plots using default values of a 30 microfarad capacitance, 25kHz carrier frequency and a sample rate of 5E-5s





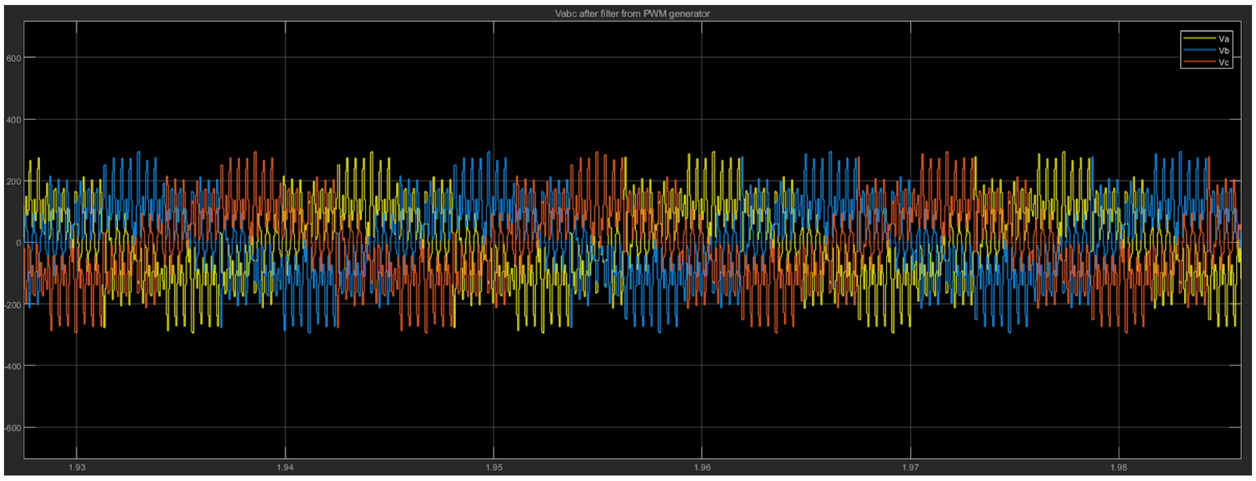
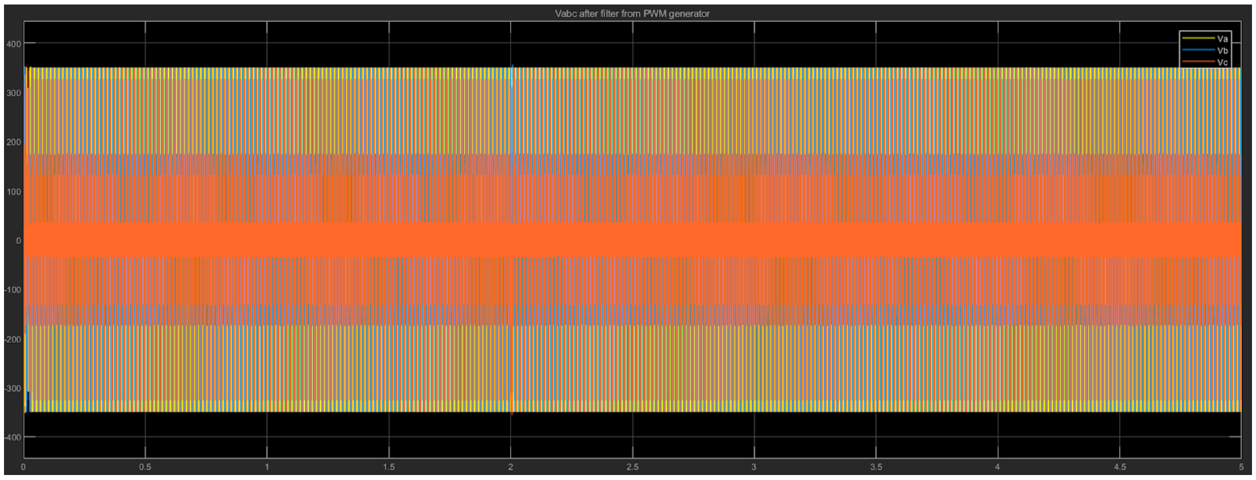
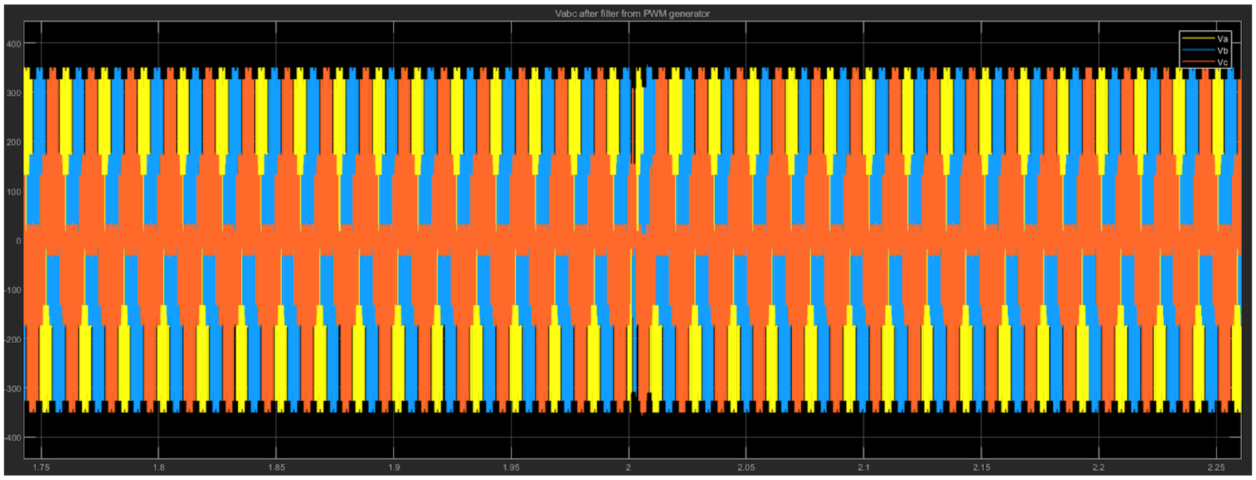


Figure 3-11 V_{abc} plots using a 30 microfarad capacitance, 2.5kHz carrier frequency and a sample rate of 4E-5s





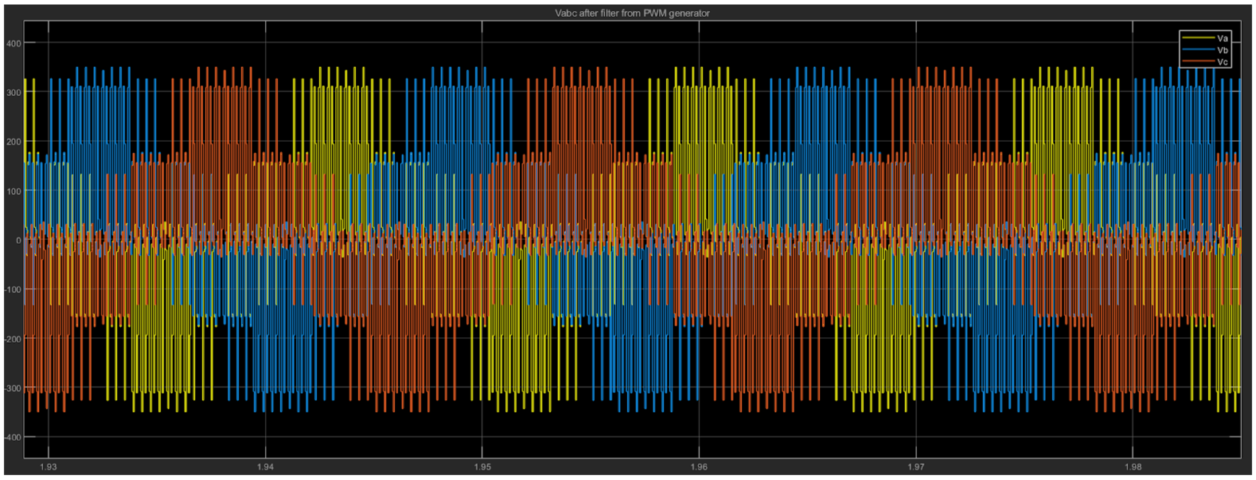
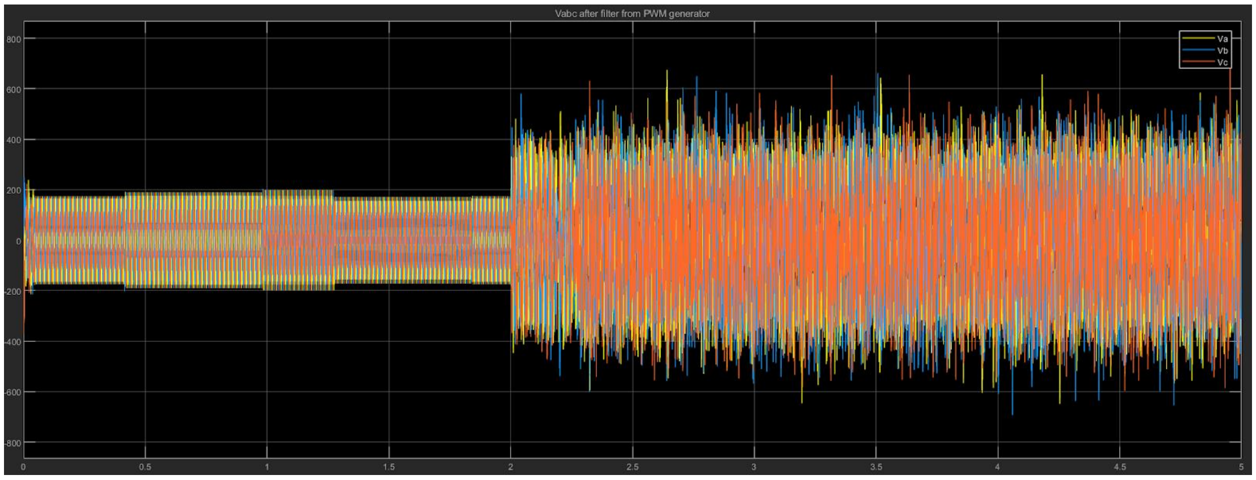
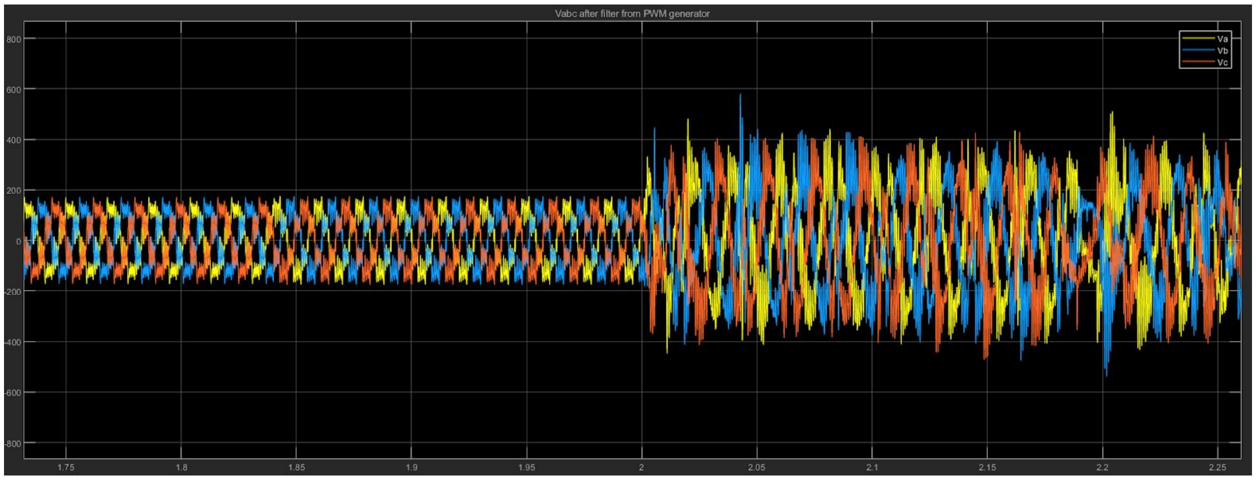


Figure 3-12 V_{abc} plots using a 3 microfarad capacitance (1/10 original value), 2.5kHz carrier frequency and a sample rate of 4E-5s





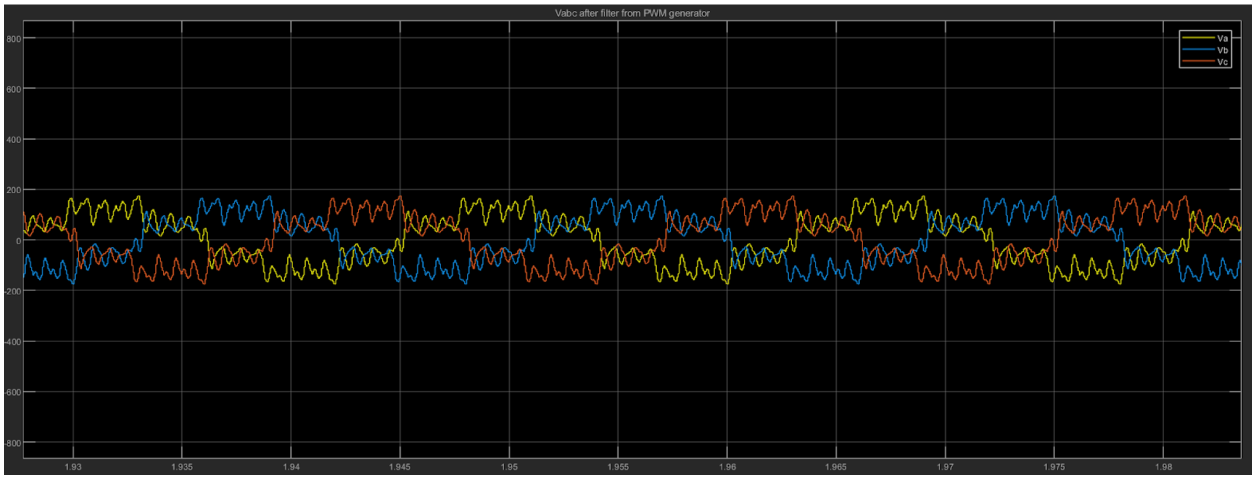


Figure 3-13 V_{abc} plots using a 300 microfarad capacitance (10 times original value), 2.5kHz carrier frequency and a sample rate of 4E-5s

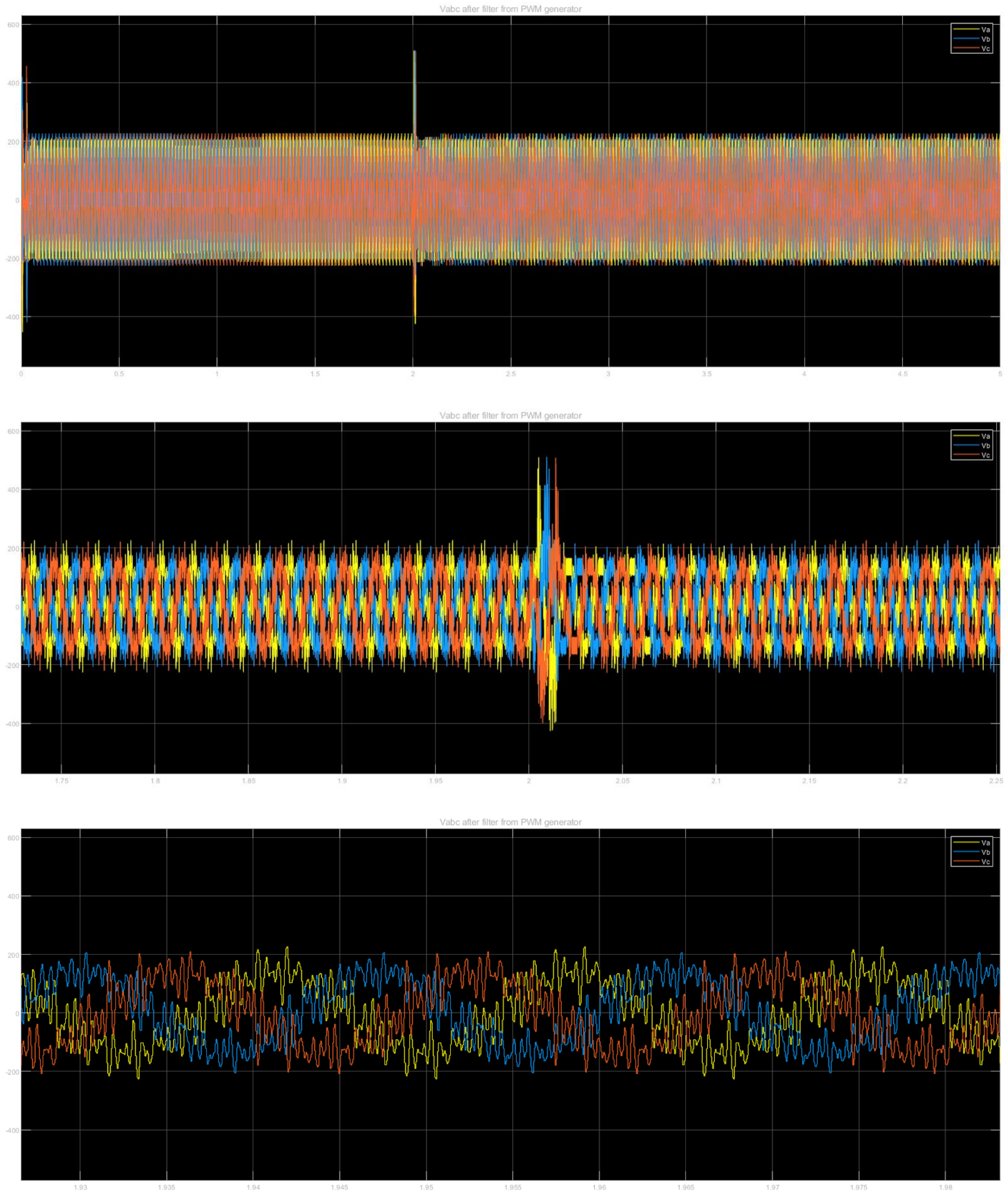


Figure 3-14 V_{abc} plots using a 150 microfarad capacitance (5 times original value), 2.5kHz carrier frequency and a sample rate of 4E-5s

We later learned that using a carrier frequency of 2.5 kHz was infeasible for a practical implementation into hardware because the filter components would be too large. We pivoted back to using a PWM carrier frequency of 25kHz. Using a PWM carrier frequency of 25 kHz, the capacitor value of 150 microfarads was still the best.

3.3 Moving Average of V_{cd} and V_{cq}

Next, we applied the new capacitor value directly to the switching model simulation. Changing the capacitor value alone was not enough to improve the strange looking V_{abc} plots. We added in a scope to view the controller output of v_{cd} and $v_{cq.}$ Examination of these plots, model shown in Figure 3-15 and Figure 3-16, showed that the v_{cd} and v_{cq} signals from the switching were much noisier than the v_{cd} and v_{cq} signals from the average model. To help reduce the noise, we added a moving average block to help filter out this noise. Then, the output of the moving average block was inputted into the switching model block. Through experimentation of different window sizes, we found that using a sliding window of 5000 samples yielded the cleanest v_{cd} and v_{cq} signals. We discovered that adding the moving average block significantly improved the V_{abc} plots. V_{abc} now looked sinusoidal as shown in Figure 3-18.

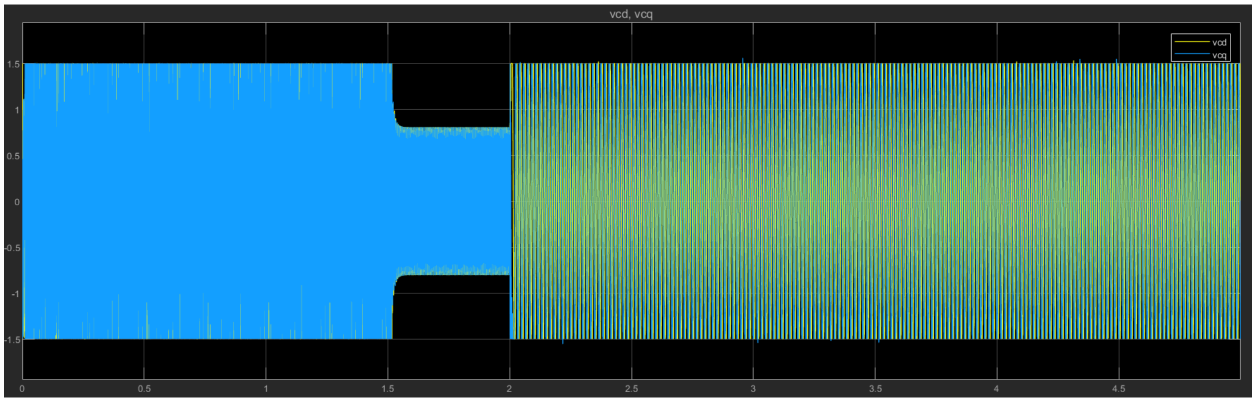


Figure 3-15 Noisy v_{cd} and v_{cq} signals from switching model

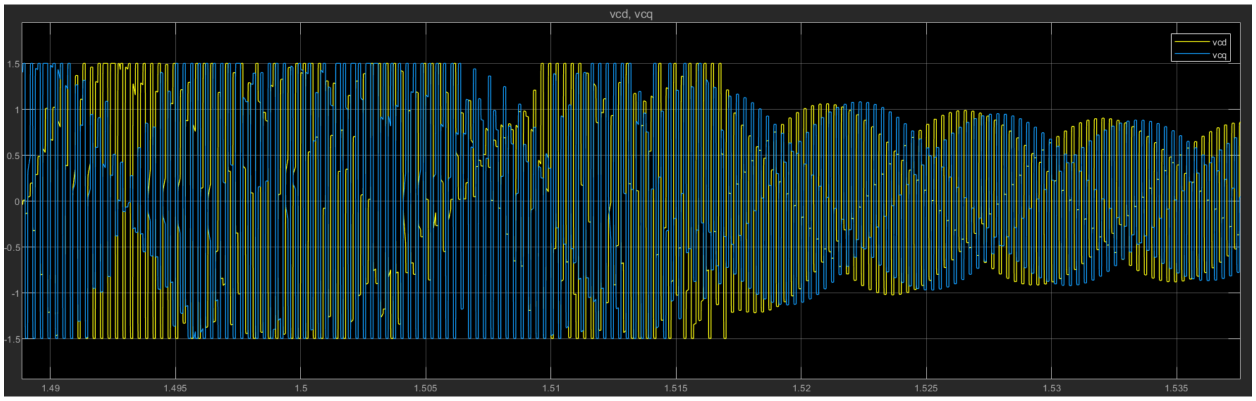


Figure 3-16 A zoomed in view of the noisy v_{cd} and v_{cq} signals from the switching model

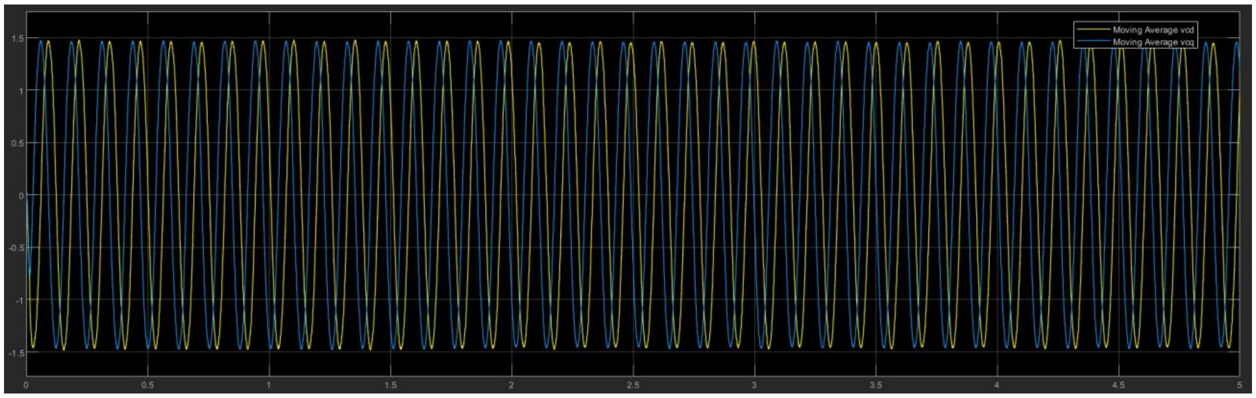


Figure 3-17 The v_{cd} and v_{cq} signals after going through moving average block

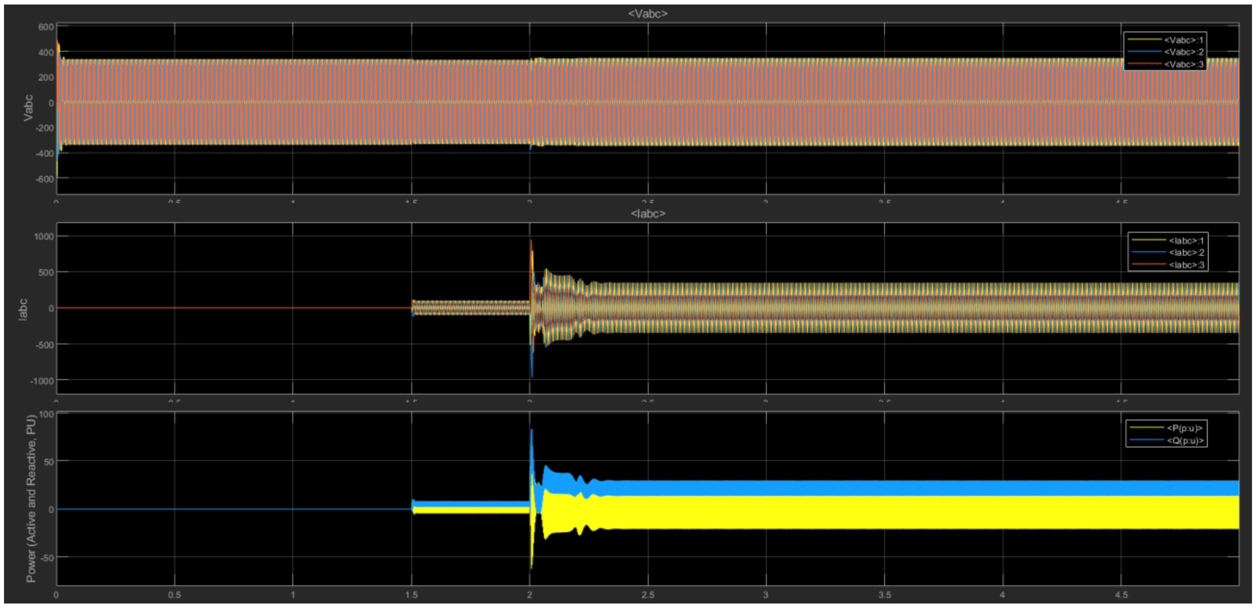


Figure 3-18 V_{abc}, I_{abc}, P, and Q plots after applying a moving average filter to the v_{cd} and v_{cq} signals

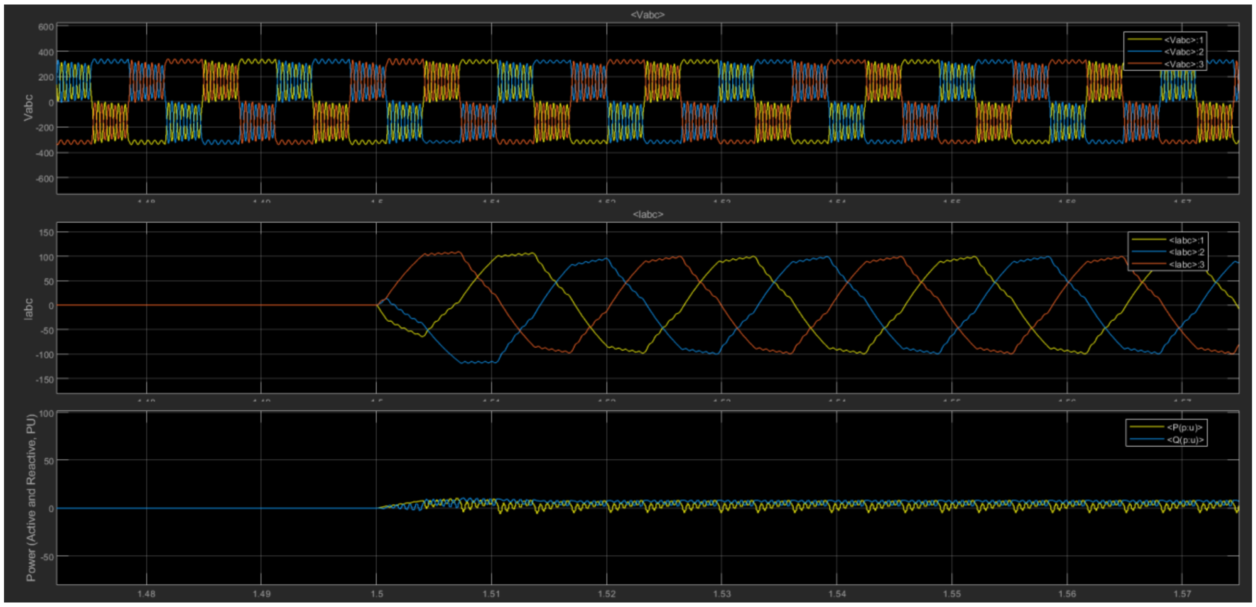


Figure 3-19 Zoomed in V_{abc}, I_{abc}, P, and Q plots after applying a moving average filter to the v_{cd} and v_{cq} signals

Although V_{abc} looked sinusoidal, it still contained a lot of harmonic frequencies as seen in the zoomed in plot in Figure 3-19. Additionally, the amplitude of V_{abc} changed when the capacitance was adjusted. The cause of these harmonics remains an open question. One theory for these harmonics proposed in [7] is that the controller assumes steady state and does not use the definition for the rate of change of reactive power of equation (3.10), which applies to nonlinear dynamics. This definition for $\dot{Q}$ was first introduced in [18].

$$\dot{Q} = v\frac{di}{dt} - i\frac{dv}{dt} \tag{3.10}$$

Future work will control the inverter connected to a synchronous machine system with controllers that use the definition in equation (3.10) for the rate of change of reactive power and see if it helps to reduce some of these effects.

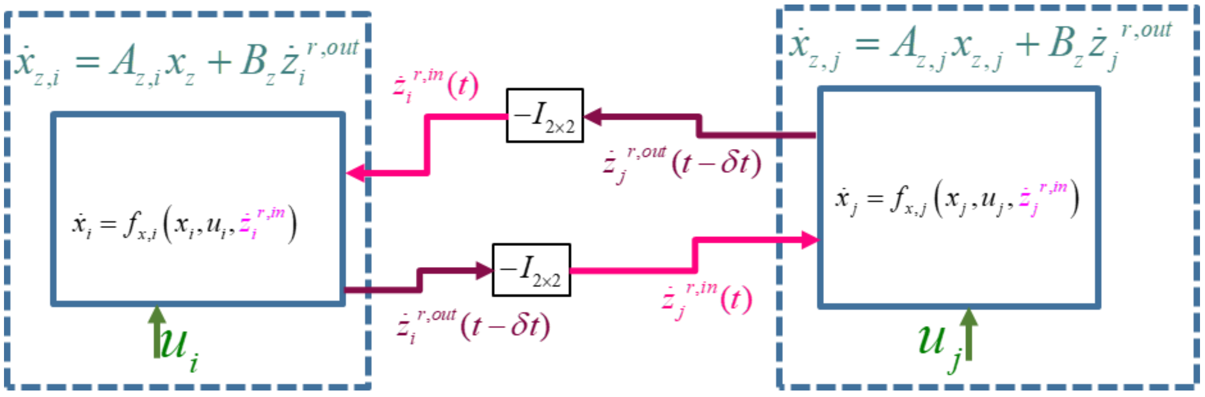
4 PAMPS: Plug-and-Play Automated Modeling of Power Systems

The hierarchical control of today relies on a couple assumptions that may not always hold. One of these assumptions is the decoupling of power and reactive power [7]. A new MATLAB based software tool using the PAMPS, Plug-and-Play Automated Modeling of Power System, approach is introduced. Unlike today's hierarchical control, in PAMPS power and reactive power are coupled. Additionally, in contrast chapter 3, the rate of change of reactive power is now defined using equation (3.10).

PAMPS has both a "zoomed out" view and a "zoomed in" view which will be introduced in the following sections. The ideas for the PAMPS software architecture in this chapter build upon the theoretical model for PAMPS introduced in [15] and [16].

4.1 PAMPS Main Module: A "Zoomed Out" View

The communication between PAMPS main modules is shown in Figure 4-1 from [7] and [19]. This is based on the model introduced in [15]. The only information exchanged between two PAMPS modules are the interaction variables $\dot{z}^{r,in}$ and $\dot{z}^{r,out}$. They can be defined as a vector of power, P ,and the derivative of reactive power, $\dot{Q}$, and are defined in equations (4.1) and (4.2) [15]. No knowledge of the technology specific internal dynamics within a PAMPS module is necessary. This allows for a "zoomed out" view of the system.



$\dot{x}_{z,i} = A_{z,i}x_z + B_z\dot{z}_i^{r,out}$
$\dot{x}_{z,j} = A_{z,j}x_{z,j} + B_z\dot{z}_j^{r,out}$
$\dot{x}_i = f_{x,i}(x_i, u_i, \dot{z}_i^{r,in})$
$\dot{x}_j = f_{x,j}(x_j, u_j, \dot{z}_j^{r,in})$
$\dot{z}_i^{r,in}(t)$
$-I_{2\times2}$
$\dot{z}_j^{r,out}(t-\delta t)$
$-I_{2\times2}$
$\dot{z}_j^{r,in}(t)$
$\dot{z}_i^{r,out}(t-\delta t)$
u_i
u_j

Figure 4-1 A "zoomed out" view of the interaction between PAMPS modules from [7] and [19]

$$\dot{z}^{\,r,in} = \begin{bmatrix} P^{\,r,in} \\ \dot{Q}^{\,r,in} \end{bmatrix} \tag{4.1}$$

$$\dot{z}^{\,r,out} = \begin{bmatrix} P^{\,r,out} \\ \dot{Q}^{\,r,out} \end{bmatrix} \tag{4.2}$$

4.2 PAMPS Submodules: A "Zoomed in" View of PAMPS

A user can also "zoom in" to a single PAMPS module i. Each PAMPS module consists of several submodules. These include the plant dynamics module, a mapper from traditional dynamics to the energy space, an energy dynamics module, energy observer module, energy space controller module, and a mapper from energy control to traditional state variables. This "zoomed in" view of a PAMPS model is shown in Figure 4-2, which is adapted from the work in from [7] and [19].

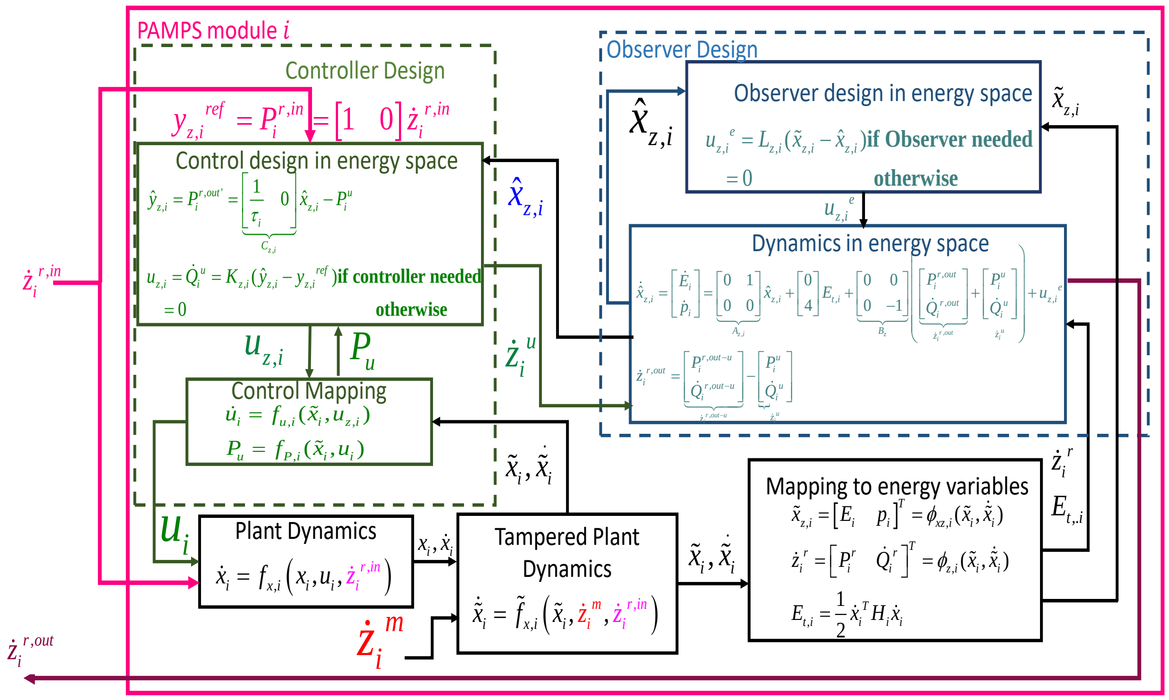


Figure 4-2 "Zoomed in" view of second-order general PAMPS model [7] [19]

4.2.1 Plant Dynamics

Plant dynamics are component specific. They calculate the state variables, which are in terms of voltages and currents. These are calculated using a function of control input u_i and the input from the main PAMPS module $\dot{z}_i^{r,in}$ shown in equation (4.3).

$$\dot{x}_i = f_{x,i}\left(x_i, u_i, \dot{z}_i^{r,in}\right) \tag{4.3}$$

4.2.2 Tampered Plant Dynamics

These plant dynamics may get corrupted by an external disturbance to the system. This is modeled by the tampered plant dynamics block, which can receive an external disturbance $\dot{z}_i^{m}$.

$$\dot{\tilde{x}}_i = \tilde{f}_{x,i}\left(\tilde{x}_i, \dot{z}_i^{m}, \dot{z}_i^{r,in}\right) \tag{4.4}$$

4.2.3 Energy Mapper

The state variables of the tampered plant dynamics are then mapped from being in terms of voltages and currents into the energy space. The energy state variables are E_i, which is the stored energy, and p_i, which is the derivative of the stored energy. They are calculated using a function of the tampered plant state variables, $\tilde{x}_i$ and their derivatives $\dot{\tilde{x}}_i$ as shown in equation (4.5). P_i^r and $\dot{Q}_i^r$ are calculated using equation (4.6) , which makes up the vector $\dot{z}_i^r$. In the second order model, $E_{t,i}$, stored energy in the tangent space, is treated as a disturbance. It is calculated using the derivative of the tampered plant state variables, $\dot{\tilde{x}}_i$, and the inertia matrix, H_i ,as shown in (4.7).

$$\tilde{x}_{z,i} = \begin{bmatrix} E_i & p_i \end{bmatrix}^T = \phi_{xz,i}(\tilde{x}_i, \dot{\tilde{x}}_i) \tag{4.5}$$

$$\dot{z}_i^r = \begin{bmatrix} P_i^r & \dot{Q}_i^r \end{bmatrix}^T = \phi_{z,i}(\tilde{x}_i, \dot{\tilde{x}}_i) \tag{4.6}$$

$$E_{t,i} = \frac{1}{2}\dot{\tilde{x}}_i^T H_i \dot{\tilde{x}}_i \tag{4.7}$$

These energy state variables can be used to calculate the approximate dynamics in the energy space. $E_{t,i}$ and $\dot{z}_i^r$ are sent to the dynamics in energy space submodule block, and the state variables E_i ,and p_i are sent to the observer in energy space.

4.2.4 Energy Dynamics Submodule

The energy dynamics submodule receives $\dot{z}_i^r$ and $E_{t,i}$ from the energy mapper, and control input $\dot{z}^u$ from the energy controller submodule. It computes $\dot{z}^{r,out}$ using the following equation:

$$\dot{z}^{r,out} = \begin{bmatrix} P^r - P^u \\ \dot{Q}^r - \dot{Q}^u \end{bmatrix} = \dot{z}^r - \dot{z}^u \tag{4.8}$$

It then sends $\dot{z}^{r,out}$ to the main PAMPS module, which can then send this value to other PAMPS modules connected to it as shown in Figure 4-1. The energy dynamics module also computes the approximate energy dynamics of E_i and p_i. The state space equation for the energy dynamics is equation (4.9).

$$\dot{\hat{x}}_{z,i} = \begin{bmatrix} \dot{E}_i \\ \dot{p}_i \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}}_{A_{z,i}} \hat{x}_{z,i} + \begin{bmatrix} 0 \\ 4 \end{bmatrix} E_{t,i} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}}_{B_z} \left(\underbrace{\begin{bmatrix} P_i^{r,out} \\ \dot{Q}_i^{r,out} \end{bmatrix}}_{\dot{z}_i^{r,out}} + \underbrace{\begin{bmatrix} P_i^u \\ \dot{Q}_i^u \end{bmatrix}}_{\dot{z}_i^u} \right) + u_{z,i}^e \tag{4.9}$$

4.2.5 Energy Observer

The energy observer module tries to minimize the difference between the reported state variables $\tilde{x}_z$ from the energy mapper and the calculated values $\hat{x}_{z,i}$ from the energy dynamics. It does so by computing $u_{z,i}^e$ in equation (4.10) and sends this to energy dynamics submodule. This observer module is being developed by Masters student Premila Rowles.

$$u_{z,i}^e = L_{z,i}(\tilde{x}_{z,i} - \hat{x}_{z,i}) \tag{4.10}$$

The calculated energy dynamics state variables $\hat{x}_{z,i}$ are sent to the energy controller module.

4.2.6 Energy Controller Module

The energy controller module works to control a desired output $\hat{y}_{z,i}$ to be close to the reference point $\hat{y}_{z,i}^{ref}$. It sends the controllable input in the energy space, $u_{z,i}$, to the energy control mapper.

One such type of control is the sliding mode controller described in [15]. This controller operates under the assumption that the energy dynamics have settled and the derivative of stored energy, p_i , is equal to 0. In the controller from [15] , $\hat{y}_{z,i}$ is defined using equation (4.11). It receives the controlled real power, P_i^u , from the energy control mapper. The controllable input in the energy space $u_{z,i}$ is a controllable input to the derivative of reactive power, $\dot{Q}_i^u$. It is set in equation (4.12).

$$\hat{y}_{z,i} = P_i^{r,out'} = \underbrace{\begin{bmatrix} \frac{1}{\tau_i} & 0 \end{bmatrix}}_{C_{z,i}} \hat{x}_{z,i} - P_i^u \tag{4.11}$$

$$u_{z,i} = \dot{Q}_i^u = K_{z,i}(\hat{y}_{z,i} - y_{z,i}^{ref}) \tag{4.12}$$

The vector $\dot{z}_i^u$ is defined using P_i^u and $\dot{Q}_i^u$. This is sent to the energy dynamics submodule to help with computing approximate energy dynamics. It is also sent to the energy control mapper.

4.2.7 Energy Control Mapper

The energy control mapper takes the controllable interaction variable $\dot{z}_i^u$ of the controllable inputs in the energy space and maps them into physical variables that can be used by the plant dynamics module. It does this using $\dot{z}_i^u$, the state variables, and their derivatives from the tampered plant dynamics block to calculate the derivative of the controllable input, $\dot{u}_i$, as shown in equation (4.13). Once it is calculated, u_i is sent to the plant dynamics block to apply the control to the physical dynamics of the plant.

$$\dot{u}_i = f_{u,i}(u_i, \tilde{x}_i, \dot{\tilde{x}}_i, \dot{z}^{u,i}) \tag{4.13}$$

In addition to calculating $\dot{u}_i$, the energy control mapper also maps the physical variables from the tampered plant dynamics back to the energy space by calculating P_u. P_u is calculated using equation (4.14). Once it is calculated P_u is sent to the energy controller submodule.

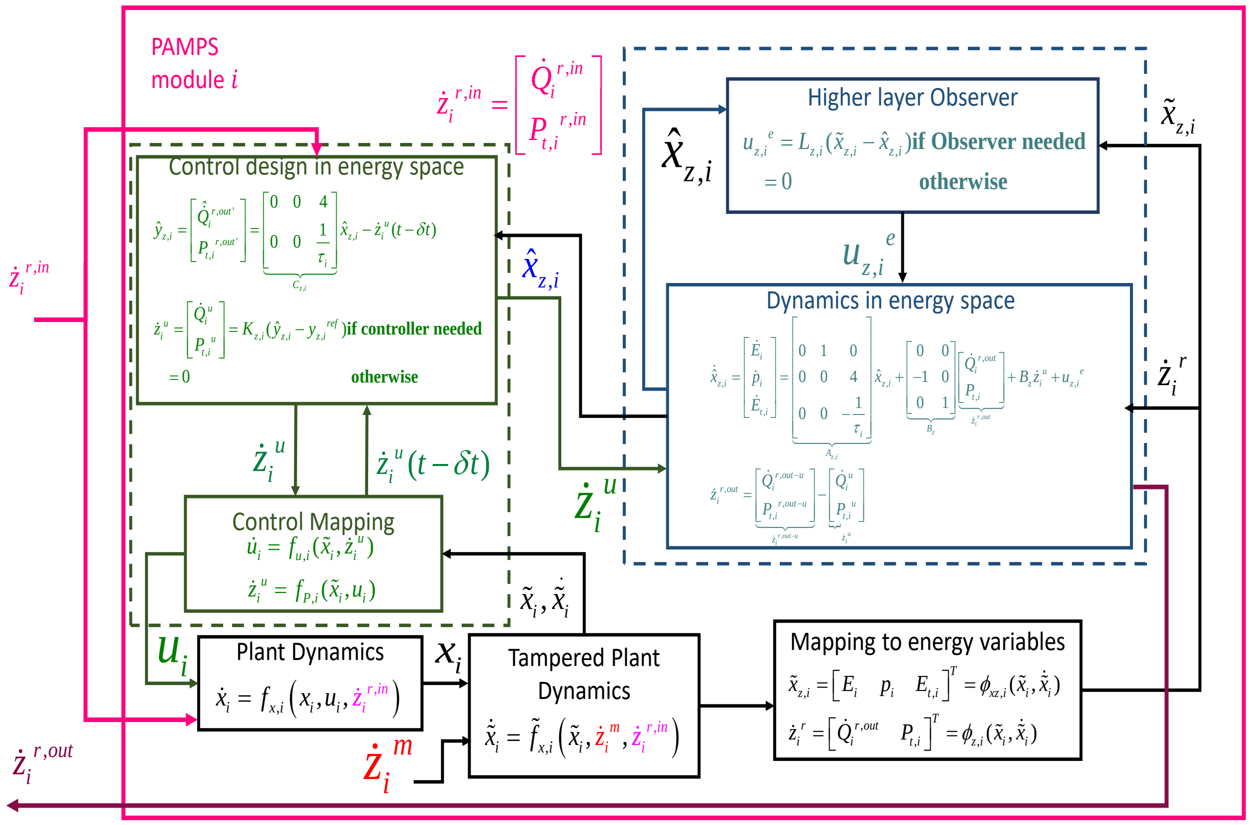
$$P_u = f_{P,i}(\tilde{x}_i, u_i) \tag{4.14}$$

4.3 Third-Order Model of PAMPS module

The third-order model of PAMPS, shown in Figure 4-3, which is adapted from [7] and [19] is very similar to the second order model. However, in the third order model, E_t, stored energy in the tangent space, is treated as a state variable, whereas in the second order model E_t is treated as a disturbance. Additionally, the interaction variables $\dot{z}^{r,in}$ and $\dot{z}^{r,out}$ are now defined as a function of power in the tangent space, P_t, and the derivative of reactive power, $\dot{Q}$, rather than a function of power, P, and the derivative of reactive power, $\dot{Q}$. The new definitions for interaction variables $\dot{z}^{r,in}$ and $\dot{z}^{r,out}$ are shown in equations (4.15) and (4.16).

$$\dot{z}^{r,in} = \begin{bmatrix} P_t^{r,in} \\ \dot{Q}^{r,in} \end{bmatrix} \tag{4.15}$$

$$\dot{z}^{r,out} = \begin{bmatrix} P_t^{r,out} \\ \dot{Q}^{r,out} \end{bmatrix} \tag{4.16}$$



PAMPS module i
$\dot{z}_i^{r,in} = \begin{bmatrix} \dot{Q}_i^{r,in} \\ P_{t,i}^{r,in} \end{bmatrix}$
Control design in energy space
$\hat{y}_{z,i} = \begin{bmatrix} \hat{\dot{Q}}_i^{r,out'} \\ P_{t,i}^{r,out'} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & 4 \\ 0 & 0 & \frac{1}{\tau_i} \end{bmatrix}}_{C_{z,i}} \hat{x}_{z,i} - \dot{z}_i^u(t-\delta t)$
$\dot{z}_i^u = \begin{bmatrix} \dot{Q}_i^u \\ P_{t,i}^u \end{bmatrix} = K_{z,i}(\hat{y}_{z,i} - y_{z,i}^{ref})$ if controller needed
$= 0$ otherwise
$\dot{z}_i^u$
$\dot{z}_i^u(t-\delta t)$
Control Mapping
$\dot{u}_i = f_{u,i}(\tilde{x}_i, \dot{z}_i^u)$
$\dot{z}_i^u = f_{P,i}(\tilde{x}_i, u_i)$
u_i
Plant Dynamics
$\dot{x}_i = f_{x,i}(x_i, u_i, \dot{z}_i^{r,in})$
x_i
Tampered Plant Dynamics
$\dot{\tilde{x}}_i = \tilde{f}_{x,i}(\tilde{x}_i, \dot{z}_i^m, \dot{z}_i^{r,in})$
$\dot{z}_i^m$
$\tilde{x}_i, \dot{\tilde{x}}_i$
$\hat{x}_{z,i}$
$\hat{x}_{z,i}$
Higher layer Observer
$u_{z,i}^e = L_{z,i}(\tilde{x}_{z,i} - \hat{x}_{z,i})$ if Observer needed
$= 0$ otherwise
$u_{z,i}^e$
Dynamics in energy space
$\dot{\hat{x}}_{z,i} = \begin{bmatrix} \dot{\hat{E}}_i \\ \dot{\hat{p}}_i \\ \dot{\hat{E}}_{t,i} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 4 \\ 0 & 0 & -\frac{1}{\tau_i} \end{bmatrix}}_{A_{z,i}} \hat{x}_{z,i} + \underbrace{\begin{bmatrix} 0 & 0 \\ -1 & 0 \\ 0 & 1 \end{bmatrix}}_{B_z} \underbrace{\begin{bmatrix} \dot{Q}_i^{r,out} \\ P_{t,i} \end{bmatrix}}_{\dot{z}_i^{r,out}} + B_z \dot{z}_i^u + u_{z,i}^e$
$\dot{z}_i^{r,out} = \underbrace{\begin{bmatrix} \dot{Q}_i^{r,out-u} \\ P_{t,i}^{r,out-u} \end{bmatrix}}_{\dot{z}_i^{r,out-u}} - \underbrace{\begin{bmatrix} \dot{Q}_i^u \\ P_{t,i}^u \end{bmatrix}}_{\dot{z}_i^u}$
$\dot{z}_i^u$
$\tilde{x}_{z,i}$
$\dot{z}_i^r$
Mapping to energy variables
$\tilde{x}_{z,i} = [E_i \quad p_i \quad E_{t,i}]^T = \phi_{xz,i}(\tilde{x}_i, \dot{\tilde{x}}_i)$
$\dot{z}_i^r = [\dot{Q}_i^{r,out} \quad P_{t,i}]^T = \phi_{z,i}(\tilde{x}_i, \dot{\tilde{x}}_i)$
$\dot{z}_i^{r,in}$
$\dot{z}_i^{r,out}$

Figure 4-3 "Zoomed in" view of third-order general PAMPS model [7] [19]

4.4 Coding PAMPS

Communication between the PAMPS software's main modules and submodules occurs in a distributed way. This allows for abstraction within modules. Rather than having a centralized model, each PAMPS main module and submodule only needs to have knowledge of their own states and any port inputs. The communication between modules is facilitated by the SGRS (Smart Grid in a Room Simulator) Platform, which was created by previous students in the group [10]. The SGRS platform creates a framework for running MATLAB objects in a distributedway. It enables simulations to be run on either a local machine or a lab cluster [20]. SGRS objects are defined within the SGRSObjects folder. Previous objects had been defined for modeling objects at the tertiary level within the “TE_models” folder and for modeling objects at the secondary level within the “Dynamic_Models” folder which comprises DAMPS. For implementing the PAMPS software, I created a new folder called “PAMPS_Models.” Within this folder I created MATLAB classes for the PAMPS main module and each PAMPS submodule. These classes were the main module: “PAMPSMain”, and the submodules: “EnergyControllerSubmodule”, “EnergyControlMapper”, ”EnergyDynamicsSubmodule”, ”EnergyMapper”, “EnergyObserver”, “PlantDynamicsMain”, and “TamperedPlantDynamics”. “PAMPSMain” corresponds to the main PAMPS module described in section 4.1. The submodules correspond to the PAMPS submodules defined in section 4.2.

The user sets up the PAMPS system using a scenario file, which is an XML file. The user first specifies a start and end time to the simulation. They also specify a slow and fast step size. All time values are encoded using the ISO 8601 standard [20]. The user instantiates all the PAMPS modules and submodules to be used in the simulation. They can also specify any parameters specific for a given module, such as resistances, inductances, and initial conditions. The scenario file also specifies which components communicate with each other.
Figure 4-4 shows a coding snippet of the section of the scenario file that specifies these communication links. Note that PampsMain_1 and PampsMain_2 have a communication link between them, and PampsMain_1 and PampsMain_2 have communication links to their

respective submodules. However, PampsMain_1 does not need a communication link to any of the internal PampsMain_2's submodules and PampsMain_2 does not need a communication link to any of the internal PampsMain_1 submodules.

```
<!-- a link from object to object describes a communication channel
   communication channels for the simulation -->
   <link A='PampsMain_1' B='PampsMain_2'/>

   <link A='PlantDynamicsMain_1' B='PampsMain_1'/>
   <link A='PlantDynamicsMain_1' B='TamperedPlantDynamics_1'/>
   <link A='TamperedPlantDynamics_1' B='EnergyControlMapper_1'/>
   <link A='TamperedPlantDynamics_1' B='EnergyMapper_1'/>
   <link A='EnergyMapper_1' B='EnergyDynamicsSubmodule_1'/>
   <link A='EnergyMapper_1' B='EnergyObserver_1'/>
   <link A='EnergyDynamicsSubmodule_1' B='EnergyObserver_1'/>
   <link A='EnergyDynamicsSubmodule_1' B='PampsMain_1'/>
   <link A='EnergyDynamicsSubmodule_1' B='EnergyControllerSubmodule_1'/>
   <link A='EnergyControllerSubmodule_1' B='PampsMain_1'/>
   <link A='EnergyControllerSubmodule_1' B='EnergyControlMapper_1'/>
   <link A='EnergyControlMapper_1' B='PlantDynamicsMain_1'/>

   <link A='PlantDynamicsMain_2' B='PampsMain_2'/>
   <link A='PlantDynamicsMain_2' B='TamperedPlantDynamics_2'/>
   <link A='TamperedPlantDynamics_2' B='EnergyControlMapper_2'/>
   <link A='TamperedPlantDynamics_2' B='EnergyMapper_2'/>
   <link A='EnergyMapper_2' B='EnergyDynamicsSubmodule_2'/>
   <link A='EnergyMapper_2' B='EnergyObserver_2'/>
   <link A='EnergyDynamicsSubmodule_2' B='EnergyObserver_2'/>
   <link A='EnergyDynamicsSubmodule_2' B='PampsMain_2'/>
   <link A='EnergyDynamicsSubmodule_2' B='EnergyControllerSubmodule_2'/>
   <link A='EnergyControllerSubmodule_2' B='PampsMain_2'/>
   <link A='EnergyControllerSubmodule_2' B='EnergyControlMapper_2'/>
   <link A='EnergyControlMapper_2' B='PlantDynamicsMain_2'/>
```

Figure 4-4 Coding snippet of the scenario file section that specifies communication links

When the scenario file is read, the SGRS platform will automatically create all the PAMPS main modules and submodules defined in the scenario file and establish all of the communication channels between modules as specified by the communication links. Each PAMPS main module and submodule is initialized with its own framework handle in the constructor. This framework handle is a MATLAB structure created by the SGRS platform that includes each main module and submodule's name, communication object, log object, time object, and parameter object [20]. The name is a string of the unique name of the PAMPS module or submodule. Its communication object enables communication to all the modules it is linked to [20]. The log object allows each main module and submodule's states to be logged to a csv file [20]. The time object keeps track of the time of each module or submodule and contains the start time, end time, and step sizes specified in the scenario [20]. The function "timeStep" can be called on the time object to increment the time by one step size. The parameter object contains the parameters specified in the scenario file for that particular module or submodule.

Every PAMPS main module and submodule is a subclass to the "PAMPS_Module" superclass. The superclass specifies that every PAMPS module and submodule will have a name, communication object, log object, time object, and parameter object. Additionally, it specifies that each PAMPS module and submodule has the properties of input variables, output variables, old variables, input communicators, output communicators, state variable equations, and state. The input variables and output variables are structures that store the values for the variables that the PAMPS main module or submodule receives and sends respectively. Old variables is a structure that stores input or output variables from the previous timestep. Input communicators is an array of every communication channel that the PAMPS main module or submodule receives input variables from. Similarly, output communicators is an array of every communication channel that the PAMPS main module or submodule sends output variables to. The state variable equations are a structure containing the equations for calculating every output variable. They are written with symbolic variables which are from MATLAB's Symbolic Math Toolbox. This allows output variables to be calculated by substituting these symbolic variables with the numerical values from the relevant input variables and output variables at each timestep. The step function within each main module and submodule is written as a state machine. The property state keeps track of which state the main module or submodule is in.

Within the constructor of each PAMPS main module and submodule, the properties of name, communication object, log object, time object, and parameter object are set based on the corresponding properties from the framework handle. Next, the communication object initializes all its communication channels. It also sets the input communicators, output communicators, and state variable equations properties. The property state is also initialized.

Once every main module and submodule is initialized, SGRS will call the step function of each main module and submodule in a distributed way. The step function is written as a state machine for each main module and submodule. The algorithm for the step function for one main PAMPS module is shown in Figure 4-5.

State 0:

- The output variable $\dot{z}^{r,in}$ is initialized to $\dot{z}^{r,in,0}$.
- The output variable $\dot{z}^{r,out}$ is initialized to $\dot{z}^{r,out,0}$.
- Set state to equal 1.

State 1:

- Send out a message with $\dot{z}^{r,in}$ to any PAMPS submodules in the output communicators array. (If the communication links match Figure 4-2 ,this is "PlantDynamicsMain" and "EnergyControllerSubmodule").
- Send out a message $\dot{z}^{r,out}$ to the other PAMPS main modules in the output communicators array.
- Set state to equal 2.

State 2:

- Check to see if messages have been received from all input communicators.
- If messages have been received from all communicators:
    - Set input variables to the values received from the communicators.
    - Set $\dot{z}^{r,in}$ equal to - $\dot{z}^{r,out}$ from the other PAMPS main module

Figure 4-5 Algorithm for the step function of a PAMPS's software main module

The algorithm for the plant dynamics is written in Figure 4-6. The algorithms for all the other submodules follow a similar structure. They also have an initialization state, sending variables state, and reading and calculating new variables state.

State 0:

- The output variable x_i is initialized to $x_{i,0}$.
- The output variable $\dot{x}_i$ is initialized to $\dot{x}_{i,0}$.
- The input variable u_i is initialized to $u_{i,0}$.
- The input variable $\dot{z}_i^{r,in}$ is initialized to $\dot{z}_{i,0}^{r,in}$.
- Save x_i, $\dot{x}_i$, u_i, and $\dot{z}_i^{r,in}$ within the old variables array.
- Set state to equal 1.

State 1:

- Send out a message with x_i, $\dot{x}_i$, to the "TamperedPlantDynamics" communicator.
- Set state to equal 2.

State 2:

- Check to see if messages have been received from all input communicators.
- If messages have been received from all communicators:
    - Set input variable u_i to the value received from the "EnergyControlMapper" communicator.
    - Set input variable $\dot{z}_i^{r,in}$ to the value received from the "PAMPSMain" communicator.
    - Calculate x_i and $\dot{x}_i$ by using the state variable equations and

Figure 4-6 Algorithm for the step function of the PAMPS software's Plant Dynamics module

4.5 RL Circuit Example

One example of PAMPS is to model an RL circuit with a controllable voltage source connected to a constant power load. This is shown in Figure 4-7 from [15]. The block diagram of the submodules within component sigma1 is shown in Figure 4-8, which is adapted from [7] and [19].

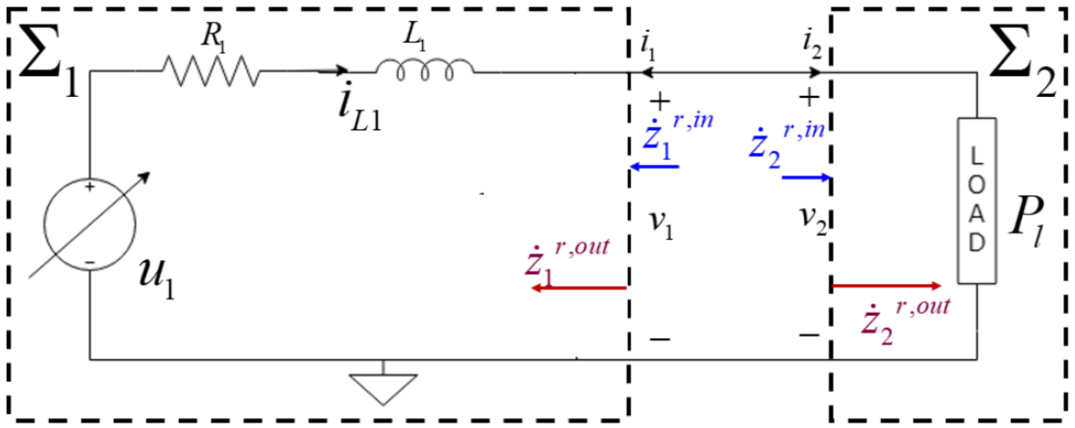


Figure 4-7 Diagram of an RL circuit with a controllable voltage source connected to a constant power load [15]

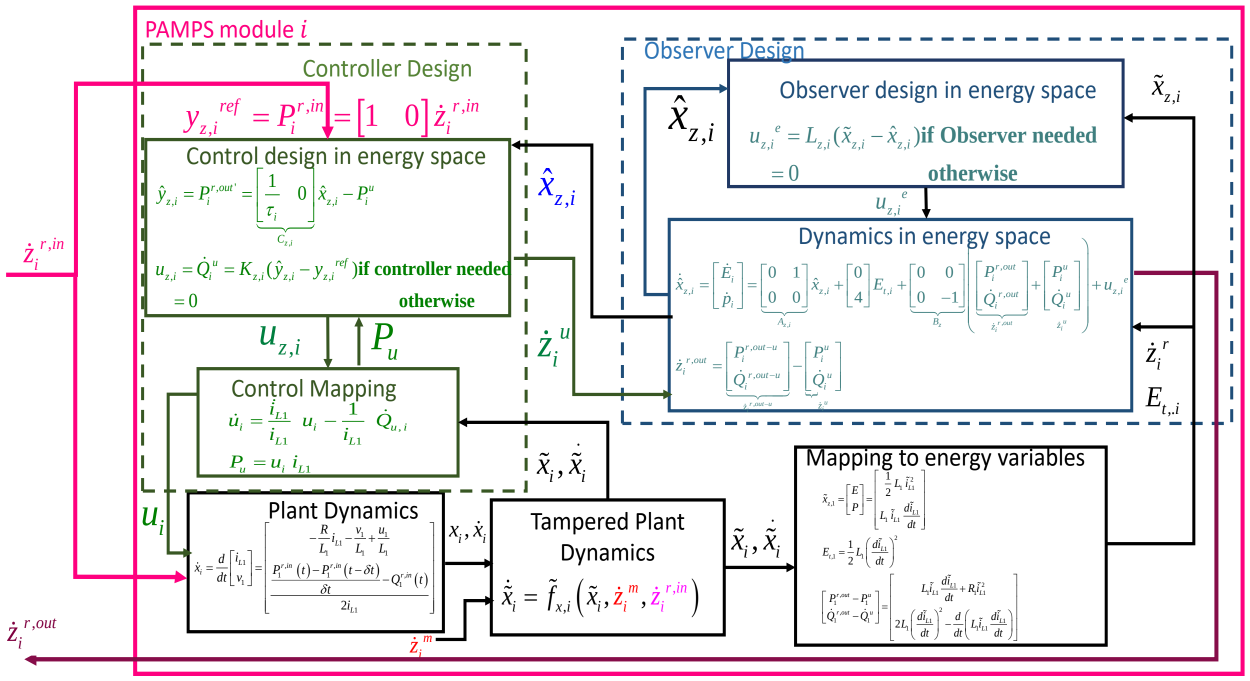


Figure 4-8 "Zoomed in" view of second-order of the component specific PAMPS model for a RL circuit with a controllable voltage source [7] [19]

The general function that represents the state space model of the plant dynamics in Equation (4.3) can now be defined by the technology specific plant dynamics of component 1. The state variables x_i are the current across the inductor, i_{L1}, and the voltage at the port , v_1. The technology specific plant dynamics for component 1 can be represented by equation (4.17) [15].

$$\dot{x}_i = \frac{d}{dt}\begin{bmatrix} i_{L1} \\ v_1 \end{bmatrix} = \begin{bmatrix} -\frac{R}{L_1} i_{L1} - \frac{v_1}{L_1} + \frac{u_1}{L_1} \\ \frac{\dot{P}_1^{r,in} - Q_1^{r,in}(t)}{2 i_{L1}} \end{bmatrix} \qquad (4.17)$$

Note that equation (4.17) contains the derivative $\dot{P}_1^{r,in}$, however, it is only receiving $P_1^{r,in}$. As a result, an approximate derivative needs to be calculated. Several methods for calculating this derivative are proposed in [16]. The best method found in [16] was to use the equation (4.18).

The derivative is calculated using $P_1^{r,in}$ from the current and previous timestep. The denominator, δt, can be defined as the step size of the timestep. Equation (4.19) shows the result of substituting equation (4.18) into equation (4.17).

$$\dot{P}_1^{r,in} \approx \frac{P_1^{r,in}(t) - P_1^{r,in}(t-\delta t)}{\delta t} \tag{4.18}$$

$$\dot{x}_i = \frac{d}{dt}\begin{bmatrix} i_{L1} \\ v_1 \end{bmatrix} = \begin{bmatrix} -\frac{R}{L_1}i_{L1} - \frac{v_1}{L_1} + \frac{u_1}{L_1} \\ \dfrac{\dfrac{P_1^{r,in}(t) - P_1^{r,in}(t-\delta t)}{\delta t} - Q_1^{r,in}(t)}{2i_{L1}} \end{bmatrix} \tag{4.19}$$

Equations (4.5), (4.6), and (4.7) maps the tampered state variables to energy variables. These can now be defined by technology specific equations (4.20), (4.21), and (4.22) [15].

$$\tilde{x}_{z,1} = \begin{bmatrix} E \\ P \end{bmatrix} = \begin{bmatrix} \frac{1}{2}L_1\,\tilde{i}_{L1}^2 \\ L_1\,\tilde{i}_{L1}\,\frac{d\tilde{i}_{L1}}{dt} \end{bmatrix} \tag{4.20}$$

$$E_{t,1} = \frac{1}{2}L_1\left(\frac{d\tilde{i}_{L1}}{dt}\right)^2 \tag{4.21}$$

$$\begin{bmatrix} P_1^{r,out} - P_1^u \\ \dot{Q}_1^{r,out} - \dot{Q}_1^u \end{bmatrix} = \begin{bmatrix} L_1\tilde{i}_{L1}\frac{d\tilde{i}_{L1}}{dt} + R_1\tilde{i}_{L1}^2 \\ 2L_1\left(\frac{d\tilde{i}_{L1}}{dt}\right)^2 - \frac{d}{dt}\left(L_1\tilde{i}_{L1}\frac{d\tilde{i}_{L1}}{dt}\right) \end{bmatrix} \tag{4.22}$$

The equation for mapping control interaction variable to the physical controllable input u_i had a general form in equation (4.13) . The component specific equation is now defined in equation (4.23).

$$\dot{u}_i = \frac{\dot{i}_{L1}}{i_{L1}}\, u_i - \frac{1}{i_{L1}}\, \dot{Q}_{u,i} \tag{4.23}$$

The general equation for mapping the physical variable x_i and controllable input u_i to the energy space variable P_u had a general form in equation (4.14) . The component specific equation is now defined in equation (4.24).

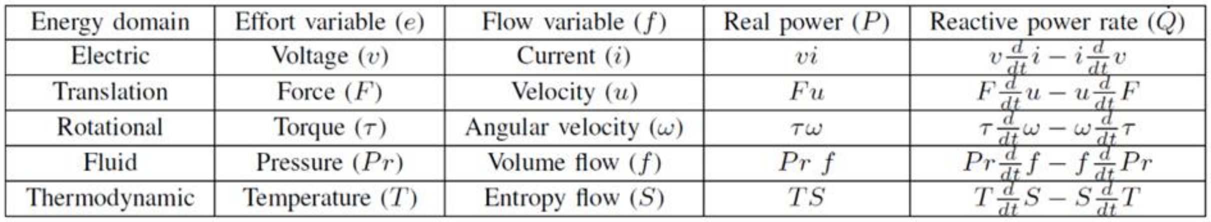
$$P_u = u_i\, i_{L1} \tag{4.24}$$

In order to implement this within the PAMPS software, the state variable equations property should be updated to equal the technology specific changes.

5 Future Work

The structure of PAMPS modules makes it easy to adapt the modules to different technologies. In order to use PAMPS for other types of technology, all that needs to change is the state variable equations defined within the structure. This can be done in a similar way to the RL circuit example shown in section 4.5 Some future applications of PAMPS could be to model synchronous machines, solar inverters, and different types of loads.

The modular nature of PAMPS also makes it easy to swap out a component for a new one. For instance, a new type of controller could be tested by creating a new controller for the energy controller submodule. Assuming the inputs and output to the controller remain the same, only the energy controller submodule block would need to be replaced. The rest of the submodules could remain the same. Another example is that different observers can be tested by swapping out the energy observer block with a new one.



Energy domain	Effort variable (e)	Flow variable (f)	Real power (P)	Reactive power rate ($\dot{Q}$)
Electric	Voltage (v)	Current (i)	vi	$v\frac{d}{dt}i - i\frac{d}{dt}v$
Translation	Force (F)	Velocity (u)	Fu	$F\frac{d}{dt}u - u\frac{d}{dt}F$
Rotational	Torque (τ)	Angular velocity (ω)	$\tau\omega$	$\tau\frac{d}{dt}\omega - \omega\frac{d}{dt}\tau$
Fluid	Pressure (Pr)	Volume flow (f)	$Pr\ f$	$Pr\frac{d}{dt}f - f\frac{d}{dt}Pr$
Thermodynamic	Temperature (T)	Entropy flow (S)	TS	$T\frac{d}{dt}S - S\frac{d}{dt}T$

Figure 5-1 Table showing calculations for power and rate of reactive power in multiple energy domains [7]

Since PAMPS is within the energy space, it is not limited to just modeling the exchange of power between electrical circuits. Real power and the rate of reactive power are defined by flow and effort variables [7]. Figure 5-1 from [7] shows how power and the rate of reactive power can also be defined within the translation, rotational, fluid, and thermodynamic domain. The flow and effort variables defined for each technology could be used within the Plant Dynamics submodule. The equations for power and the rate of reactive power could be used within the Energy Mapper submodule and Energy Control submodule to map these effort and flow variables into the energy space.

6 References

[1] M. Ilic and R. Jaddivada, "Unified Value-based Feedback, Optimization and Risk Resource Management in Complex Electric Energy Systems," in *MOPTA Conference*.

[2] M. Ilic, J. Ilic, S. Ray, J. Joo, M. Cvetkovic, K. Bachovchin, A. Hsu, J. Donadee, X. Miao and F. F. T. Cui, "“Smart Grid in a Room Simulator–SGRS”, NIST Year 1 Report," Gaithersburg, Maryland, June 2014.

[3] C. Domonoske, "California Sets Goal of 100 Percent Clean Electric Power By 2045," NPR, 10 September 2018. [Online]. Available: https://www.npr.org/2018/09/10/646373423/california-sets-goal-of-100-percent-renewable-electric-power-by-2045. [Accessed May 2021].

[4] "Hawaii Clean Energy Initiative," Hawaii State Energy Office, [Online]. Available: http://energy.hawaii.gov/testbeds-initiatives/hcei. [Accessed May 2021].

[5] M. Ilic, “Model-based Control for Sustainable Electric Energy Systems”, Volume 1, Cambridge University Press, 2020.

[6] K. D. Bachovchin, *Design, Modeling, and Power Electronic Control for Transient Stabilization of Power Grids Using Flywheel Energy Storage Systems,* Ph.D. dissertation, CMU, 2015.

[7] M. Ilic and R. Jaddivada, 6.247 Principles of Modeling, Simulations and Control for Electric Energy Systems, Cambridge, MA: MIT, Lecture Notes 2021.

[8] M. Ilic and R. Jaddivada, "Multi-layered interactive energy space modeling for near-optimal electrification of terrestrial, shipboard and aircraft systems," *Elsevier,* 2018.

[9] L. Dickerman, "How should utilities evolve resource planning to meet complex future energy challenges? Distribution-Level Integrated Resource Plans: A Necessity for the Future," *White Paper, Landis+Gyr,* 2016.

[10] M. Ilic, M. Wagner, F. Franchetti, K. Bachovchin, R. Jaddivada, S. Ray, J. Donadee, P. S. Junlakarn, J. Ilic, A. Hsu and C. Y. Tee, "“Smart Grid in a Room Simulator–SGRS”, NIST Year 3 Final Report," Gaithersburg, Maryland, December 2016.

[11] M. Ilic, J. Ilic, S. Ray, J. Joo, M. Cvetkovic, K. Bachovchin, A. Hsu, J. Donadee, X. Miao, T. Cui and F. Franchetti, "“Smart Grid in a Room Simulator–SGRS”, NIST Year 2 Report," Gaithersburg, Maryland, May 2015.

[12] M. D. Ilic, R. Jaddivada and X. Miao, "“Modeling and analysis methods for assessing stability of microgrid”," International Federation of Automatic Control (IFAC), Toulouse, France, July 2017.

[13] M. R. Wagner, K. Bachovchin and M. Ilic, "Computer architecture and multi time-scale implementations for smart grid in a room simulator," *IFAC-PapersOnLine,* vol. 48, no. 30, pp. 233-238, 2015.

[14] X. Miao and M. Ilic, "High Quality of Service in Future Electrical Energy Systems: A New Time-Domain Approach," *IEEE Transactions on Sustainable Energy,* vol. 12, no. 2, pp. 1196 - 1205, 2021.

[15] R. Jaddivada and M. Ilic, "A feasible and stable distributed interactive control design in energy state space," in *IEEE Conference on Decision and Control, CDC,* (submitted(under review) ,2021).

[16] b. M. Ilic, R. Jaddivada and A. H. Gebremedhin, "Unified Modeling for Emulating Electric Energy Systems: Toward Digital Twin that Might Work," 2020.

[17] K. D. Bachovchin, M. Ilic, R. Jaddivada and S. Flanagan, *Centralized Automated Modeling of Power Systems (CAMPS) V1.0 User's Manual*, 2021.

[18] J. L. Wyatt and M. Ilic, "Time-domain reactive power concepts for nonlinear, nonsinusoidal or nonperiodic networks," in *IEEE International Symposium on Circuits and Systems,* New Orleans, LA, USA, 1990.

[19] M. lic, R. Jaddivada, D. Wu, P. Bharadwaj, S. Flanagan and P. Rowles, "Cyber-secure inverter-based controller/observer apparatus using energy state space". Provisional patent under filing–2021.

[20] M. Ilic, J. Ilic, S. Ray, J.-Y. Joo, M. Cvetkovic, K. Bachovchin, A. Hsu, J. Donadee, P. S. Junlakarn, X. Miao, T. Cui, F. Franchetti and M. Wagner, "Smart Grid in a Room Simulator–SGRS," NIST Year 3 - Semi-Annual Report, November 2015.