AN ABSTRACT OF THE THESIS OF

Eunice Naswali for the degree of Master of Science in Electrical and Computer Engineering presented on July 22nd 2011.

Title: Modeling and experimental validation of supercapacitors for use in an In-lab grid developed for wind integration applications.

Abstract approved: _____

Annette Renee von Jouanne

The fast growth of wind energy utilization has necessitated research into wind energy integration. Due to the variable nature of wind and the forecasting challenges, it is desirable to utilize wind energy alongside energy storage sources for reliable wind energy integration. This work details the design of a 25 kW supercapacitor storage system that is integrated into an in-lab grid. The in-lab grid, which features high power sources and loads, was developed to research methods aimed at optimizing wind energy production while increasing the predictability of wind farm outputs. In order to predict the performance of the storage system for purposes of ensuring its efficient and safe use, a model representing the supercapacitor is required. This work also develops and validates a model for a supercapacitor which is used in characterization, lifetime and Matlab/Simulink simulation testing. The results from testing identify the supercapacitor's electrical and electrochemical properties as well as their degradation rates. These are then used to provide estimates for an expected performance of the storage system.

Modeling and experimental validation of supercapacitors for use in an in-lab grid developed for wind integration applications

by
Eunice Naswali

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented July 22nd 2011
Commencement June 2012

Master of Science thesis of Eunice Naswali presented on July 22nd 2011

APPROVED:

Major Professor, representing Electrical and Computer Engineering

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Eunice Naswali, Author

ACKNOWLEDGEMENTS

I would like to thank my advisors Dr. von Jouanne and Dr. Alexandre Yokochi for their valuable input, guidance and support in my work. I would also like to thank Dr. Ted Brekken for his valuable input, support and guidance. The members of my research group were of great assistance at various points in my work and I would like to thank David Naviaux, Hai-yue Han, Alex Bistrika and Chianna Alexander. I would also like to extend my gratitude to the rest of the energy systems group members for making it an enjoyable and comfortable environment to work in.

Lastly I am grateful to my fiancée, my mom, the rest of my family, Dr Chris and Elizabeth Bell and my friends for their love, support and prayers.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF TABLES

Modeling and experimental validation of supercapacitors for use in an in-lab grid developed for wind integration applications

1 INTRODUCTION

1.1 Background

Though wind is a clean source of energy and the dependence on a decreasing amount of fossil energy is endangering the environment, it is still a small percentage of the total energy supply in the United States. This is due to the variable and unpredictable nature of wind which presents wind farms as an extra challenge to be managed by grid operators. 3 MW to 5 MW of additional frequency regulation electric ancillary service is required for every 100 MW wind power installed [1]. However, utilizing stored energy alongside wind energy is shown to greatly improve energy flow management thus enabling wind to compete with other traditional energy sources. This realizable benefit has fostered the growth of wind energy utilization in the United States; the installed capacity is expected to double in the next five years [2] and hence more research into wind energy integration on a large scale is being carried out.

The Wallace Energy Systems and Renewables Facility (WESRF) has developed an in-lab grid for research into methods to optimize energy production while improving on the predictability of the energy output from a wind farm using energy storage [3]. The energy storage system integrated into this in-lab grid therefore acts as an energy buffer smoothing the fluctuations in energy output from the wind farm.

1.2 Wind Energy Storage

1.2.1 Types of Wind Energy Storage

Energy storage systems typically considered for wind applications are briefly summarized below [4]-[8]:

1) Pumped hydro

This is a form of hydro electricity; when there is low electricity demand (e.g., at night) water is pumped using a turbine from a low level reservoir to a high level reservoir and pumped back when there is high electricity demand (e.g. , during the day). The energy derived from pumped hydro is represented in (1), where η is turbine efficiency, ρgh is the potential energy per unit volume and V is the volume of water flowing. The advantages of pumped hydro are that it can respond to sudden and continuous electricity demand. It also has a high efficiency of approximately 70%-80% [6]-[8]; however, it is limited by terrain. To mitigate this, underground pumped storage is considered however this has a high capital cost. Because of its high capital costs, pumped hydro is therefore only commonly used in developing countries.

$$E = \eta \rho g h V \quad (1)$$

2) Compressed air

This technology also stores energy when electricity demand is low by compressing air into a reservoir and when electricity is needed air is expanded by heat and run through turbines to generate electricity. The reservoir could be salt caverns, abandoned mines, rock structures and artificial reservoirs. The energy from compressed air is given by (2)-(3) where n are moles of air compressed from an initial volume V_i to a final volume V_f. The advantages include; a quick start-up time (10minutes) and long storage capability. Its disadvantages are; a risk of explosion, low round trip efficiency (50%) [7] and geographical limitation. The low efficiency can be improved by heat storage where waste heat is stored up and used to heat compressed air in a later cycle. This is the case in adiabatic compressed air storage and the efficiency improvement obtained is 70%.

$$E = -\int_{V_i}^{V_f} pdV\,, \qquad \text{where } p = \frac{nRT}{V} \tag{2}$$

$$E = nRT\ In\left(\frac{V_i}{V_f}\right) \tag{3}$$

3) Flywheels

These store and supply energy in the form of kinetic energy for a rotating mass. The rotating mass is a rotor which contains a motor/generator that generates electricity. Equation (4) represents the

energy of a flywheel with radius r, mass m and rotating at angular velocity ω. k is a constant equal to 0.5 for a uniform disc and 1 for a thin ring. To increase the storage capacity, it is preferred to increase the rotational speed over increasing the mass, therefore flywheels consisting of glass or carbon fiber rather than metal which is a heavier material are now used. More flywheel developments include operating a flywheel in a vacuum and using levitated magnetic bearings rather than standard bearings to reduce on friction losses. This makes it difficult to eliminate heat as it is enclosed in a vacuum. Also the magnetics contribute a loss due to eddy currents. Other limitations of flywheel energy storage are; a high self discharge rate (1%-10% per hour) [7], a high cost and catastrophic damage in case of failure. On the flip side flywheels have a high round trip efficiency (80%-85%) [6],[8], large storage capacity and fast charge/discharge rate.

$$E = \frac{1}{2} k m r^2 \omega^2 \qquad (4)$$

4) Supercapacitors

These achieve energy storage through the accumulation of charge at the electrode-electrolyte interface when a voltage is applied. Energy of a supercapacitor with capacitance C over a charge/discharge voltage window $V_{max} - V_{min}$ is given in (5). Supercapacitors have a high

power charge and discharge capability. They also have long cycle life as no chemical reactions are involved in the charge and discharge process like in the battery. Their low self discharge rate (10% after 3 days) and high efficiency (90%) [6]-[7] due to a low equivalent series resistance (ESR) makes them beneficial to use. For large scale energy applications, supercapacitors are limited by their high cost.

$$E = \frac{1}{2} C V_{max}{}^{2} - V_{min}{}^{2} \quad (5)$$

5) Superconducting magnetics

The storage of energy is achieved by feeding a direct current into a superconducting coil; this energy can then be retrieved by discharging the current in the coil. The energy is in the form of magnetic energy, (6) is the magnetic energy per unit density that can be obtained from a superconducting magnetic coil. The benefit of using superconducting magnetic energy storage is high efficiency (95%) [5]-[8], immediate availability of power and a high power density. Its disadvantages are; a high cost, low energy density and they require low temperatures for operation.

$$E = \frac{B^2}{2\mu_o} \quad (6)$$

6) Batteries

Batteries store energy in the form of electrochemical energy. Equation (7) is the energy available from a battery with charge Q at a voltage V. The charge and discharge process in batteries involves chemical reactions which limits its cycle-life. They also are temperature sensitive. The advantages of using batteries are that they can supply a large amount of power for a long time. There are various types of batteries used for large scale energy applications. Lead acid batteries are cheap but heavy, only capable of medium efficiencies, have a low energy density and a limited cycle life. Lithium-ion batteries are light, have a high efficiency, a low self-discharge and high energy density. However they can be damaged by full charge and long periods of no use, they are also costly. Sodium sulfur batteries have a high capacity (3-4 times that of lead-acid batteries) and a high efficiency. Flow batteries consist of an electrolyte flowing through an electrochemical cell which converts the chemical energy into electricity. These have a large capacity, they can be discharged and recharged after long periods of time however they have a low energy density.

$$E = QV \qquad (7)$$

7) Hydrogen

Fuel cells and Fossil-based fuels store energy in the form of hydrogen. The challenges with using hydrogen stored energy are; the difficulty in storing and transporting due to its low density, a low round trip efficiency (25%) [7] due to heat loss from fuel cells and utilization of energy to make hydrogen. It is however an environmentally friendly source of fuel as water is emitted in the process of energy generation from hydrogen.

Figure 1.1 summarizes the power and energy characteristics of the above energy storage mechanisms.

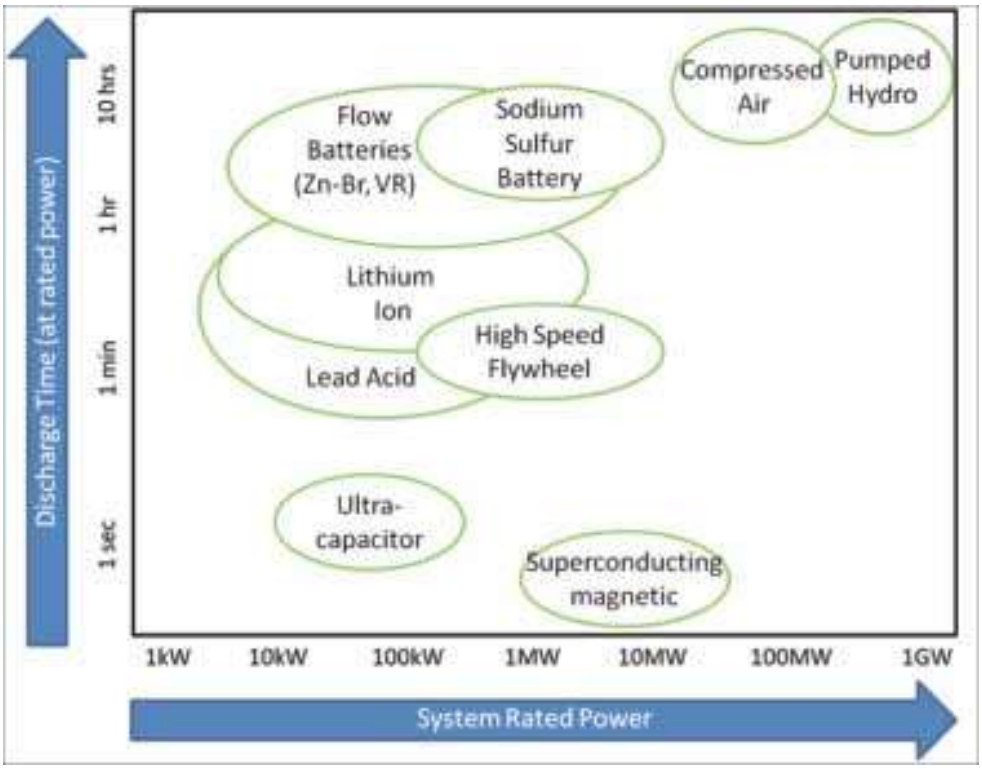


Figure 1.1: Power and energy positioning of energy storage systems [1]

1.2.2 Applications of Wind Energy Storage

Other research work carried out on compensating wind energy with energy storage for a similar application is detailed in [9]-[13]. Besides energy buffering, the coupling of stored energy with wind energy can also benefit energy production through providing low voltage ride through (LVRT) capability and power quality control. In [14] a supercapacitor energy storage system sized for LVRT capability is shown to be superior to other topologies. STATCOMs are used in reactive power compensation on wind farms, and when complemented with energy storage systems as in [15]-[17] they can also be used for real power control. Superconducting magnetics are shown in [18] to provide reactive power control and power factor correction for a micro-grid.

It is sometimes more beneficial to utilize more than one energy storage system in an application. Hydrogen energy storage alone could not balance load and supply fluctuations from wind or solar farms due to its low efficiency and slow response time, therefore, superconducting magnetic storage energy was added to provide short-term energy supply in [19]. In [20], a hybrid energy storage system consisting of batteries and superconducting magnetics is able to provide high power density and high energy density to compensate for fluctuating loads such as railway substations and wind turbines.

1.2.3 Scope of Thesis

The characteristics mentioned in Section 1.2.1 are used to select the suitable storage system for an application. The characteristics considered in selecting an energy storage system for the application in this work included: a fast – medium time response, high power density, high energy density, a long cycle lifetime, accessibility to other energy sources such as natural gas and a compact size. A composite energy storage system consisting of a flow cell battery and supercapacitors was chosen. The supercapacitors deliver a fast time response to energy demands and have a high power density while the battery serves for a medium time response to energy demands and have a high energy density. Also the combination is chosen since supercapacitors can be cycled several times without a significant effect on their lifetime, while the case is not true with batteries without electrolyte replacement.

This thesis presents how to design a supercapacitor storage system for the application. In addition, a model for a supercapacitor is developed and validated. The model is then used in various tests to study and predict the performance of the supercapacitor storage system in order to ensure its safe and efficient use.

2 SUPERCAPACITOR THEORY

2.1 Description of a Supercapacitor

Supercapacitors belong to a class of capacitors known as Electrochemical Double Layer Capacitors (EDLCs). The physical structure of an EDLC is shown in figure 2.1. Energy storage in an EDLC is achieved by charge accumulation under an applied voltage at the interface between the electrode and the electrolyte. The charged surface of the electrode attracts ions of opposite charge and repels ions of like charge. Two parallel layers of charge are formed hence the name Double Layer Capacitors. The ions of opposite charge are bound to the surface and form the layer referred to as surface layer. The ions of like charge are free to move around in the electrolyte and form the diffuse layer; negative ions are found in the region closest to the positive plate (current collector) while positive ions are found closest to the negative plate (current collector). The separator is a porous insulating membrane which allows for conduction of ions in the diffuse layer. The capacitance of an EDLC is shown in (7) where ε is the dielectric constant, A is the surface area of the electrode and d is the charge separation distance. It is clear from (7) that to achieve a high capacitance, a large surface area and a small charge separation distance are needed. The electrodes in EDLCs are porous which increases the surface area of the electrode thus allowing for a high capacitance to be realized without increasing the volume.

$$C = \frac{\varepsilon A}{4\pi d} \quad (7)$$

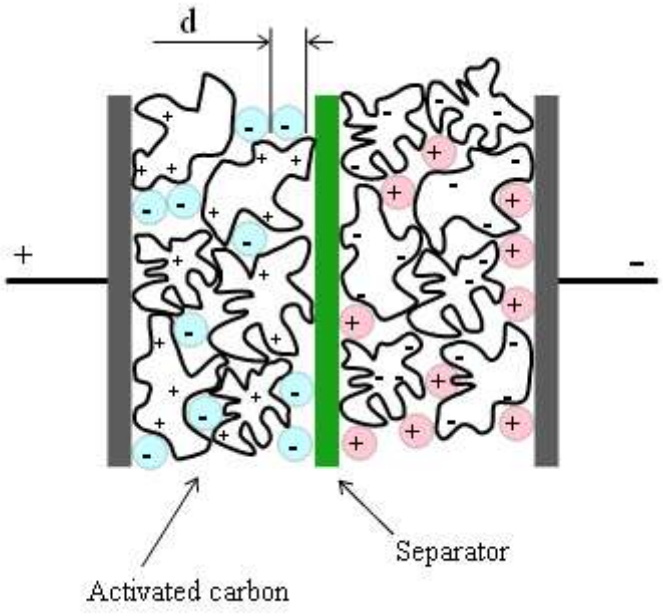


Figure 2.1: Physical structure of an EDLC

2.2 Model of a Supercapacitor

Models used for EDLCs in previous research fall in the following three categories: a RC circuit model, a three branch RC circuit model and a transmission line model. The first model is a simple model and its parameters can be obtained from the specification document of the capacitor. It is shown in figure 2.2, ESR is the equivalent series resistance representing heat losses, C is an ideal capacitor and R_{lk} is the leakage resistance which is proportional to the leakage current. This model can't be used to show the behavior over a

frequency range accurately. Furthermore it does not represent the porous nature of the electrodes [21].

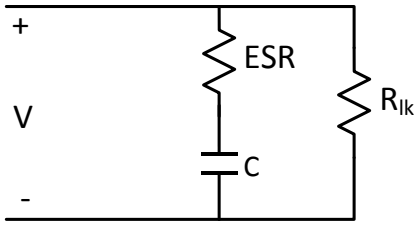


Figure 2.2: Simple RC circuit model for an EDLC

The three branch RC circuit defined in [22] is shown in figure 2.3, it consists of the leakage resistance R_{lk} and three branches which correspond to different time constants for charge transfer. The immediate branch is for time constants within seconds and it consists of a resistance R_i in series with two capacitors; a voltage dependent capacitor C_{i1} and a fixed capacitor C_{i0}. The delayed branch uses R_d, C_d to represent time constants within the minutes range while the long term branch is uses R_l, C_l for time constants greater than ten minutes.

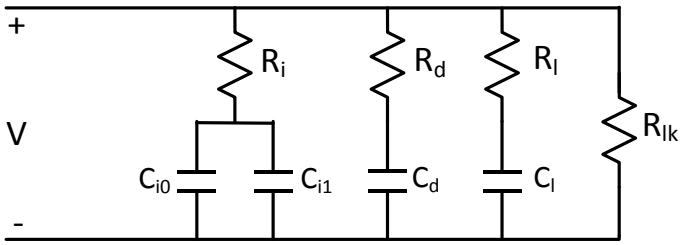


Figure 2.3: Three RC circuit model for an EDLC

The transmission line model is used in [23]-[26] to represent accurately the porous nature of the electrode in EDLCs. The transmission line is a distributed circuit of resistances and capacitances; these are shown in figure 2.4. The series resistance R_s accounts for the resistances due to contacts between materials, the electrolyte resistance and resistance of the collector. The resistances r_{ed}, r_{el} are impedances per unit length (Ωm^{-1}) that depend on the conductivity of the electrode and electrolyte respectively. The capacitance c_{dl} is an impedance per unit length (Ωm^{-1}) for the electrode-electrolyte interface (double layer). There are other parameters shown in figure 2.4 which account for behavior observed other than the transmission line behavior. These include; the leakage resistance R_{lk}, an inductance L_s which is dominant at high frequency, a resistance R_p and inductance L_p which are observed above resonant frequency. The model can be simplified further by setting r_{ed} to zero as the electrode is assumed to be highly conductive in comparison to the electrolyte.

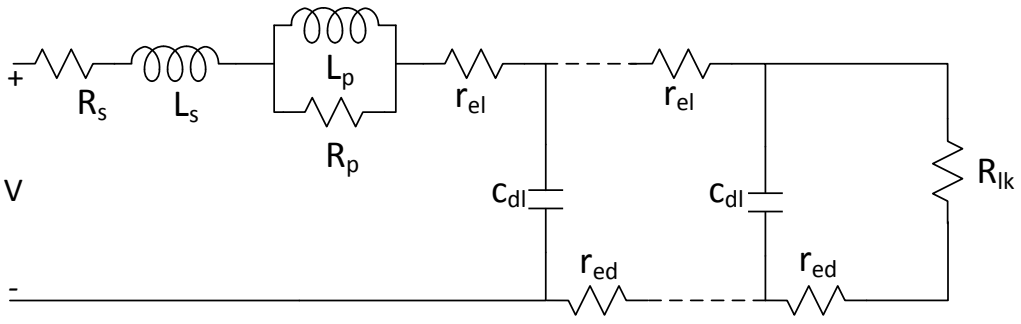


Figure 2.4: Transmission line model for an EDLC

The impedance of the transmission line is derived in [27] using (8)-(19).

$$\frac{\partial v(x,t)}{\partial x} = -r_{el} i(x,t) \tag{8}$$

$$\frac{\partial i(x,t)}{\partial x} = -c_{dl} \frac{\partial v(x,t)}{\partial t} \tag{9}$$

Combining (8) and (9) gives

$$\frac{\partial^2 v(x,t)}{\partial x^2} = r_{el} c_{dl} \frac{\partial v(x,t)}{\partial t} \tag{10}$$

Transforming (10) into Laplace gives

$$sV(x,s) = \frac{1}{r_{el} c_{dl}} \frac{\partial^2 V(x,s)}{\partial x^2}, s = i\omega \tag{11}$$

From which the general solutions for current and voltage shown in (12) and (13) are obtained

$$V(x,s) = Ae^{-\sqrt{sr_{el}c_{dl}}x} + Be^{\sqrt{sr_{el}c_{dl}}x} \tag{12}$$

$$I(x,s) = \frac{\sqrt{sr_{el}c_{dl}}}{r_{el}} (Ae^{-\sqrt{sr_{el}c_{dl}}x} - Be^{\sqrt{sr_{el}c_{dl}}x}) \tag{13}$$

Using boundary conditions $I(x,s) = 0$ for $x = L$, (14) is obtained which simplifies voltage and current expressions to (15) and (16)

$$Ae^{-\sqrt{sr_{el}c_{dl}}x} = Be^{\sqrt{sr_{el}c_{dl}}x} \tag{14}$$

$$V(x,s) = 2Be^{-\sqrt{sr_{el}c_{dl}}L} \cosh\left(\sqrt{sr_{el}c_{dl}(L-x)}\right) \tag{15}$$

$$I(x,s) = 2B \frac{\sqrt{sr_{el}c_{dl}}}{r_{el}} e^{\sqrt{sr_{el}c_{dl}}L} \sinh\left(\sqrt{sr_{el}c_{dl}(L-x)}\right) \tag{16}$$

The impedance can then be determined from (15) and (16). Setting x=0 s=iω gives

$$Z(0,s) = \frac{V(0,s)}{I(0,s)} \tag{17}$$

$$Z(i\omega) = \frac{r_{el}}{\sqrt{i\omega r_{el} c_{dl}}} \coth\left(\sqrt{i\omega r_{el} c_{dl}} L\right) \tag{18}$$

To include the effect of resistances due to contacts between materials, R_s is added to (18) giving

$$Z(i\omega) = R_s + \frac{r_{el}}{\sqrt{i\omega r_{el} c_{dl}}} \coth\left(\sqrt{i\omega r_{el} c_{dl}} L\right) \tag{19}$$

For an ideal capacitor, the electrode is assumed to have a uniform pore size distribution therefore (19) is sufficient to represent the transmission line behavior. In order to include the effect of a non-uniform pore size distribution in the electrode, a constant phase element is used [24]-[25], [27]. Equation (20) is the impedance of a constant phase element when inserted into (19) gives a new expression for the transmission line impedance shown in (21). γ is related to the extent of non-ideality and it can be estimated from a Nyquist plot, which is a plot obtained from an electrochemical impedance spectroscopy (EIS) analysis on the capacitor. Figure 2.5 is a Nyquist plot for an ideal capacitor (blue dashed line) and non-ideal capacitor (red curve), Z' is the real part of the impedance and Z'' is the imaginary part of the impedance. The difference between the two curves is seen in the angle between the slope and the x-axis at low frequencies (ϕ), (22) relates the parameters ϕ and γ [24]. An ideal capacitor has a value of $\phi = -90°$ therefore $\gamma = 1$ and (21) reduces to (19), while $\phi < -90°$ for a non-ideal capacitor and therefore $\gamma < 1$.

$$Z_{CPE}(\omega) = \frac{1}{i\omega^{\gamma}C_{dl}} \tag{20}$$

$$Z(i\omega) = R_s + \frac{r_{el}}{\sqrt{i\omega^{\gamma} r_{el} c_{dl}}} \coth\left(\sqrt{i\omega^{\gamma} r_{el} c_{dl} L}\right) \tag{21}$$

$$\gamma = -\frac{\phi}{90°} \tag{22}$$

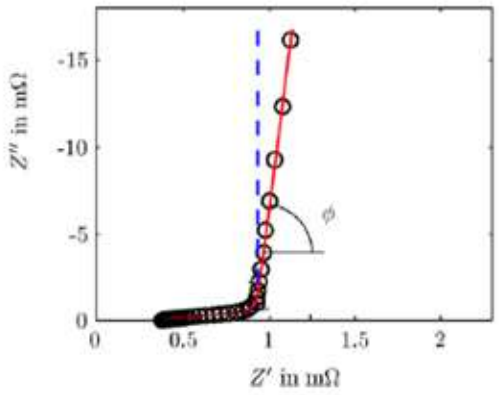


Figure 2.5: Nyquist plot for an Ideal capacitor and Non-ideal capacitor showing effect of pore size distribution [24]

At low frequency, (21) can be simplified to (23); the real part is equal to the resistance at DC or the ESR (24).

$$Z(i\omega) = R_s + \frac{r_{el}L}{3} + \frac{1}{i\omega^{\gamma}c_{dl}L} \tag{23}$$

$$R_{dc} = R_s + \frac{r_{el}L}{3} \tag{24}$$

The model used for this system is shown in figure 2.6; the transmission line parameters are included as they accurately represent the intrinsic dynamic behavior of the capacitor. The leakage resistance is included for

purposes of estimating the rate of self discharge of the capacitor in leakage tests. The dominant inductance at high frequency, the resistance and inductance after resonant frequency (transmission line model) as well as the resistance and capacitance for the delayed and long term branches (three branch RC model) are eliminated as these are observed at frequencies which exceed the frequency of the system i.e., the system's pulse duration is 60-90 seconds. The values of the parameters are obtained in Section 4.

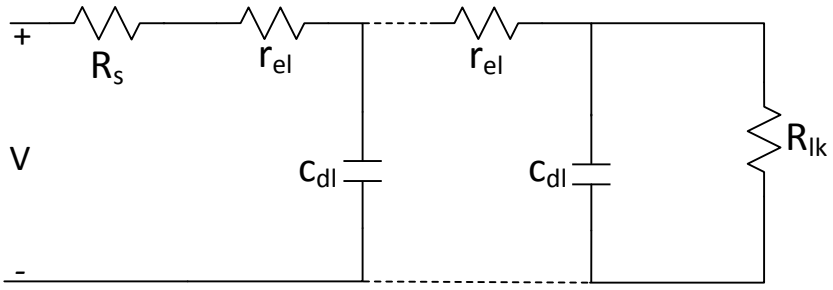


Figure 2.6: Supercapacitor model

3 SUPERCAPACITOR ENERGY STORAGE SYSTEM

3.1 In-lab Grid

Through funding received from the Bonneville Power Administration (BPA), WESRF built an in-lab grid shown in figure 3.1 for this research. The in-lab grid consists of emulated high-power grid sources and loads including a wind farm, energy storage system, a hydro resource and local loads. These are connected together at the 300 V bus via Modbus protocol and protected from system faults using Schweitzer Engineering Laboratories (SEL) 751A feeder protection relays. The in-lab grid is controlled using a dSPACE rapid prototyping machine which is used to test control algorithms in Matlab/Simulink and rapidly implement them in the laboratory. The wind farm is modeled using an Arbitrary Waveform Generator (AWG) which functions as a three-phase 120 kVA programmable source producing a power profile that follows the scaled down output of a real wind farm. The primary energy storage system is a 25 kW, 50 kWh Zinc-Bromine flow battery which delivers energy for up to 2 hours for medium time response energy demands. The supercapacitor energy storage system shown in figure 3.2 is the secondary storage system and supplies 25 kW for up to 90 seconds for fast time response energy demands. It is made up of 166 F/48.6 V modules each consisting of eighteen 3000 F/2.7 V Maxwell supercapacitors; figure 3.3 is a picture of such a module.

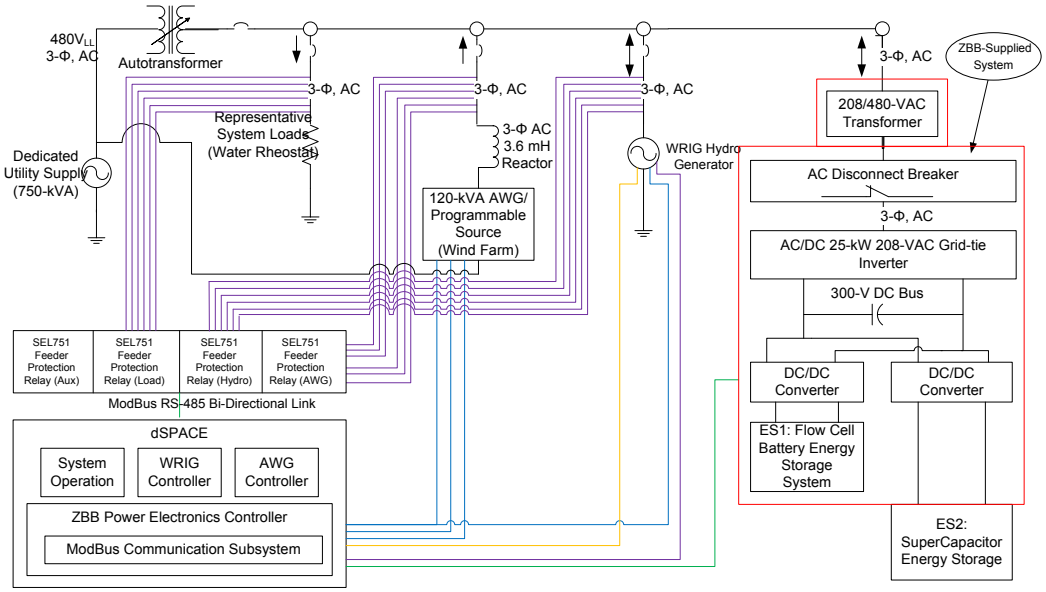


Figure 3.1: WESRF wind energy storage in-lab grid

Figure 3.2: Supercapacitor storage system

Figure 3.3: Module comprising Maxwell supercapacitors

3.2 Sizing of the Supercapacitor Storage System

The size of the storage system is dependent on the application; in this case the storage system is used in parallel with a battery storage system to supply peak power to the grid as explained in Section 3.1. The size is also dependent on the specifications of the DC/DC converter which interfaces the supercapacitor storage system to the grid. The first step in sizing the storage system is to set the specifications; table 1 summarizes the storage system's specifications.

Table 1: Supercapacitor storage system specifications

Parameter	Definition	Value
P	maximum power required from the system	25 kW
Δt	duration of a charge/discharge pulse	90 s
V_{max}	maximum operating voltage of the system, recommended to be less than bus voltage [28]	290 V
V_{min}	minimum operating voltage of the system, limited by converter's maximum current 165 A, for efficient use maximize V_{max}-V_{min}	145 V

Once the specifications are set the next step entails determining the total capacitance for the system. We use a methodology for a constant current application to derive an approximate capacitance. The average current of the system i_{avg} is used as the constant current and is determined using (25)-(26).We then use the average current in the basic equation which relates voltage and current for a capacitor given in (27).

$$i_{avg} = \frac{1}{2}(I_{max} + I_{min}) \tag{25}$$

$$I_{max} = \frac{P}{V_{min}}, \qquad I_{min} = \frac{P}{V_{max}} \tag{26}$$

$$V_{max} - V_{min} = i_{avg}\frac{\Delta t}{C} + i_{avg}R \tag{27}$$

R is the total ESR of the system and is determined using (28)-(31), where $R_{module} = 7.2\ m\Omega$ and $C_{module} = 166\ F$

$$R = \frac{\tau}{C} \tag{28}$$

$$R = R_{module}\frac{n_{series}}{n_{parallel}} \tag{29}$$

$$C = C_{module}\frac{n_{parallel}}{n_{series}} \tag{30}$$

$$\tau = R_{module}C_{module} = RC \tag{31}$$

Substitute for R in (27), gives

$$V_{max} - V_{min} = i_{avg}\frac{\Delta t}{C} + i_{avg}\frac{\tau}{C} \tag{32}$$

$$C = \frac{i_{avg}}{V_{max} - V_{min}}(\Delta t + \tau) \tag{33}$$

Using the total capacitance of the system, the number of modules in series and parallel can then be calculated in (34)-(35). $V_{max} = 2.68$ V, $V_{module} = 48.3$ V.

$$n_{series} = \frac{V_{max}}{V_{module}} \tag{34}$$

$$n_{parallel} = n_{series}\frac{C}{C_{module}} \tag{35}$$

The total capacitance calculated is 83 F, 18 modules (6 in parallel and 3 in series) were calculated to be sufficient to supply 25 kW for 90 seconds.

3.3 Testing of the Supercapacitor Storage System

Testing is carried out on a supercapacitor cell so as to gather data to use to predict the behavior of the supercapacitor system and ensure its efficient and safe use. The tests carried out can be characterized into: characterization tests, lifetime tests and system tests. Characterization tests are used to validate the supercapacitor model created and to determine the short term performance of the cell. They include EIS, constant current, and

leakage current measurement tests. Lifetime tests are used to determine the behavior of the cell over long periods of time, these include; cycle-life and calendar tests. System tests are used to test control algorithms on a simulation environment representing the in-lab grid in Matlab/Simulink.

The test benches used are shown in figures 3.4-3.5. Test bench 1 is used for the EIS tests; the cell is connected to an impedance analyzer which supplies the ac current for the tests through the current carrying wires (green and red) and the impedance analyzer is connected to a computer which collects the data measured through the voltage measuring wires (blue and white). The current carrying wires were separated from the voltage measuring wires to eliminate any errors due to a voltage drop in the voltage measuring wires and thus allowing for more accurate measurement [29]. The Reference 3000 Potentiostat (3 A/15 V) from Gamry Instruments capable of performing frequency sweep from 10 μHz to 1 MHz was used for impedance analysis. The rest of the tests are carried out using test bench 2; the cell is connected to a 50 A/40 V HP6269B DC power supply which delivers a constant current. The cell is also connected to a 300 W 60 A/80 V DC electronic load which is used to discharge the cell. The control of the tests and the capture of data are achieved using a dSPACE machine.

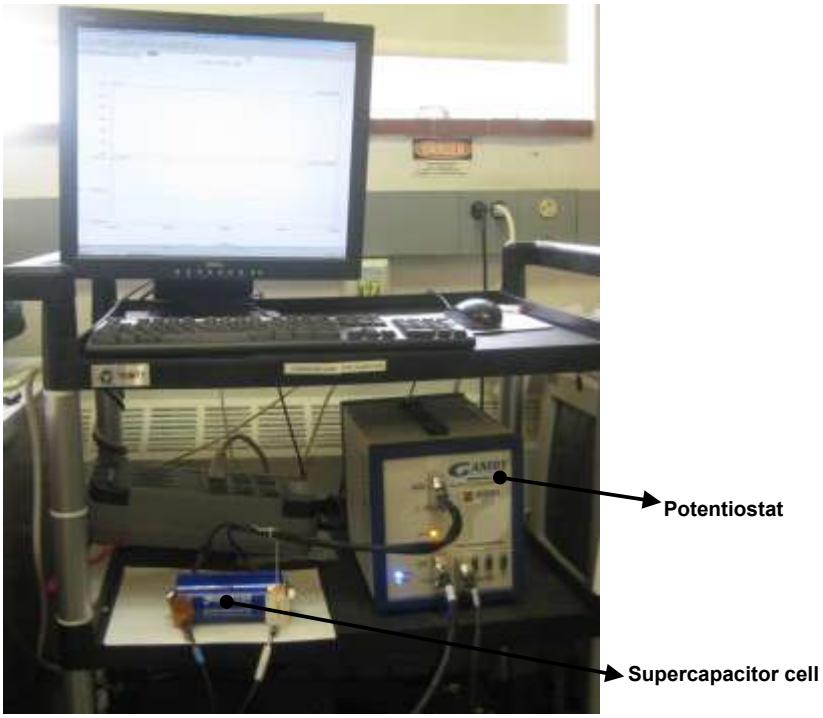


Figure 3.4: Test Bench 1 Setup

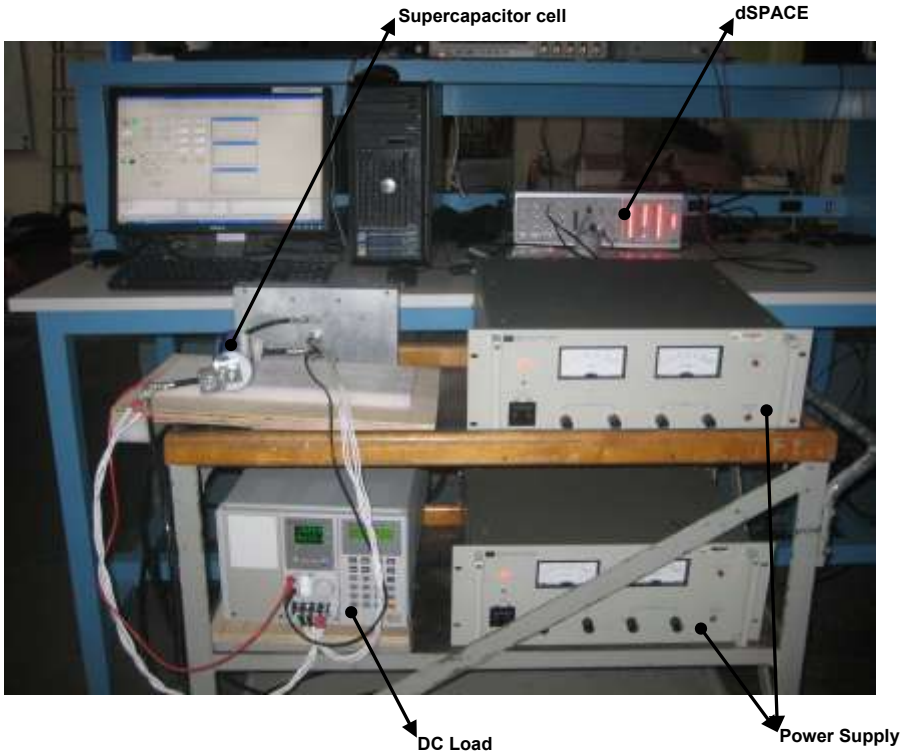


Figure 3.5: Test Bench 2 Setup

3.3.1 Electrochemical Impedance Spectroscopy Test

This test measures the electric and electrochemical properties of the electrode and electrolyte in the supercapacitor cell. It is carried out prior to any other tests to measure the transmission line parameters of the supercapacitor model. It is also carried out during cycling tests to measure the evolution of the parameters which is used in gauging aging of the cell.

The Galvanostatic mode is preferred over a Potentiostatic mode for measuring low impedance supercapacitors. A voltage AC signal is applied to the cell for a Potentiostatic mode test while a current AC signal is applied for a Galvanostatic mode test. In potentiostatic mode, a 1 mV error can create a large DC current in the cell which would change a cell's state of charge while a cell's state of change is unaffected in galvanostatic mode, since the potentiostat can control currents as small as a few milli-amperes [29].

In this test, a small sinusoidal signal with a frequency sweep is supplied to the cell that is already biased around a dc voltage. A small signal is used so as not to perturb the device's equilibrium; a 1 A ac current was used. The frequency range is selected according to the application's duration of a charge/discharge cycle; the frequency was varied from 10 mHz-10 kHz. The procedure for the test is as follows [30];

i. Charge the cell to the desired voltage using test bench 2.
ii. Transfer the cell to test bench 1
iii. Run a galvanostatic EIS

The voltage of the cell is measured and converted to complex impedance which is represented in a Nyquist plot from which the transmission line parameters of the supercapacitor model can be calculated, this is illustrated in Section 4.

3.3.2 Constant Current Test

This test measures the voltage and current of the cell during charge and discharge cycles under a constant current. The test may be carried out prior to and during cycling. The voltage and current measured are used to calculate the round trip efficiency of the cell. To calculate the round trip efficiency a group of cycles is used rather than data from one cycle in order to average out any errors. The voltage and current may also be used to measure the capacitance and ESR; however this test is an accurate way to measure the ESR.

The maximum test current (I_{test}) used is selected based on the maximum current of the system (I_{max}). I_{max} is 160 A and there are three parallel branches in the system therefore the maximum current through the cell is$160/3=53$ A, I_{test} is set to 50 A. Other test current values used include $0.75I_{test}$, $0.5I_{test}$, $0.25I_{test}$ and $0.125I_{test}$. The procedure for the test outlined in [30] is as follows:

i. Open circuit the cell for 10 seconds, record time t_1, voltage V_1, current I_1

i. Charge the cell to half its rated voltage at specified test current, record time t_2, voltage V_2, current I_2

ii. Open circuit the cell for 5 seconds, record time t_3, voltage V_3, current I_3

iii. Open circuit the cell for 10 seconds, record time t_4, voltage V_4, current I_4

iv. Discharge the cell to its rated voltage at specified test current, record time t_5, voltage V_5, current I_5

v. Open circuit the cell for 5 seconds, record time t_6, voltage V_6, current I_6

3.3.3 Leakage Current Measurement Test

This test measures the losses in the cell when it is inactive. The losses can be represented in various ways; in terms of leakage current, resistance and energy. The leakage resistance is the resistor in parallel with the transmission line in the supercapacitor model. The results of this test can also be used to measure the self discharge rate prior to carrying out more tests and also during cycling tests. The procedure for the test is as follows:

i. Charge cell to its rated voltage

ii. Open circuit the cell for at least 72 hours, record the voltage

3.3.4 Lifetime Estimation Test

This test aims to determine the effect of cycling at conditions similar to normal operation to quantify ageing. Other research on ageing has been carried out at accelerated ageing [23]-[24], [32]-[33]. Ageing in supercapacitors is seen as an increase in the equivalent series resistance, a decrease in capacitance and an increase in the self discharge rate. The main ageing factors in supercapacitors are voltage and temperature. High temperatures increase the rate of reactivity of chemical components [34]-[35]. Higher voltages contribute to the decomposition of the electrolyte; a process that creates impurities and by-product such as gases and water which block the electrode and separator pores. Therefore conductivity deteriorates resulting in an increased resistance and decreased capacitance [34]-[35].

Authors have used various ageing models to quantify the effect of voltage and temperature on ageing of EDLCS. In [24] a heuristic model is used and it assumes an exponential rate of degradation with temperature and voltage and a linear change in the cell's parameters under a constant voltage and temperature. The model defines the change of a parameter a(t, T, V) over an ageing time t_{eq} using (36)-(37). T_0 and V_0 can be set to the rated values of the cell and the ageing constants c_a, c_T and c_V for the transmission line parameters R_s, r_{el}, c_{dl} and γ have been determined experimentally.

$$a(t, T, V) = a_{initial}(1 + c_a t_{eq}) \tag{36}$$

$$t_{eq} = t\, c_T^{(T - T_0)/\Delta T}\, c_V^{(V - V_0)/\Delta V} \quad (37)$$

The Arrhenius law has been used by the author [35] to quantify ageing of EDLCS due to the effect of temperature. For a given voltage, the reaction rate v_A is given by (38) where A is an exponential factor and E_A is the activation energy. The lifetime of the cell is then quantified using the assumption that the life is inversely proportional to the reaction rate thus giving (39).

$$v_A = Ae^{-\frac{E_A}{kT_i}} \quad (38)$$

$$t_i = Be^{\frac{E_A}{kT_i}} \quad (39)$$

The cycling test involves running several cycles of the constant current test interrupted at specified number of cycles to run an EIS test. The procedure is as follows;

i. Charge the cell to its rated voltage using test bench 2
ii. Transfer the cell to test bench 1
iii. Run a galvanostatic EIS test
iv. Return the cell to test bench 2
v. Run the constant current test continuously for the desired number of cycles
vi. Repeat the procedure

3.3.5 In-lab Grid System Test

Both energy storage systems are controlled by an Automatic Generation Control (AGC) method. The AGC controller is run in a simulated environment which contains models for all devices in the in-lab grid and the supercapacitor control gain is tuned to buffer rapid changes in energy demand from the flow cell battery. The controller is tuned through a genetic algorithm that uses the mean absolute error and mean squared error in a cost function. The goal of the AGC controller is to meet the power demand with +/-4% per unit error. The simulations are performed in Matlab/Simulink; figure 3.6 is a Simulink model of the in-lab grid and figure 7 is a component of this model which represents the supercapacitor.

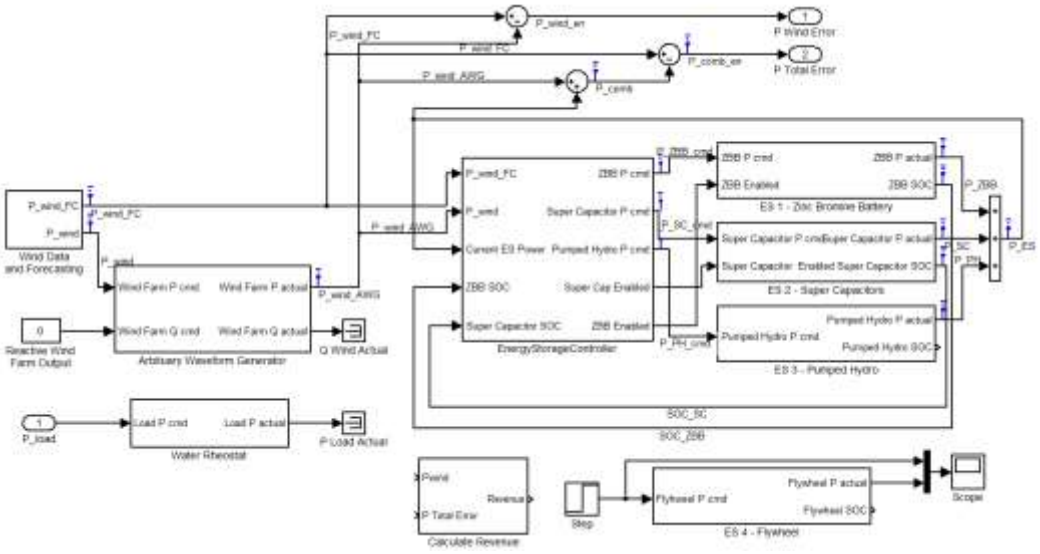


Figure 3.6: In-lab grid control model

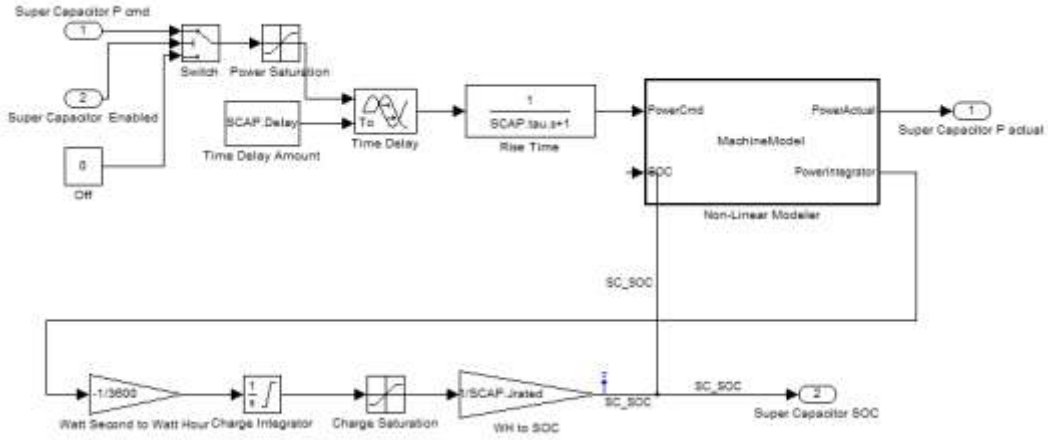


Figure 3.7: Supercapacitor control model

4 SIMULATION RESULTS

4.1 Electrochemical Impedance Spectroscopy Test

The first impedance spectroscopy results obtained from testing the cell prior to carrying out any other tests are presented in figures 4.1- 4.2. The cell was biased at 2.5 V DC and a 1 A AC signal was supplied to the cell while the frequency was swept from 10 mHz to 10 kHz to perform a galvanostatic spectroscopy test. The results are used to derive the values of the transmission line parameters in the supercapacitor model; the calculations are summarized in table 2.

The behavior of the cell as the frequency is varied can be divided into two frequency regions. A dominant capacitive behavior exists at low frequencies. This is seen in figure 4.1, where the highest impedance is obtained at the lowest frequency (this is because a capacitor appears as an open circuit at low frequency). The capacitive behavior is also seen in figure 4.2 where the magnitude decreases as the frequency increases with a slope of -20dB/decade (capacitance varies inversely with frequency). In addition, the phase when the capacitance dominates in a circuit is close to -90°, in figure 4.2 the phase is seen to increase from -90° at low frequencies. The non-ideality in the capacitor discussed in Section 2 is also observed at lower frequencies. The angle of the complex impedance curve with x-axis in figure 4.1 is less than -90° which means that $\gamma < 1$. Figure 4.3 shows the capacitance versus frequency. It is derived from the complex impedance using (36). From this figure the double

layer capacitance can be calculated using the value C indicated in the figure at ω=1radian.

$$C(\omega) = \frac{1}{i\omega Z} \tag{36}$$

At medium frequencies, there is a minimum resistance which is observed in figure 4.2 when the magnitude of the impedance remains constant with a change in frequency. This minimum resistance is also the value of real impedance where the complex impedance curve crosses the x-axis in figure 4.1. At these frequencies, the series resistance and the electrolyte resistance can be obtained using the values R_1 and R_2 read off from figure 4.1.

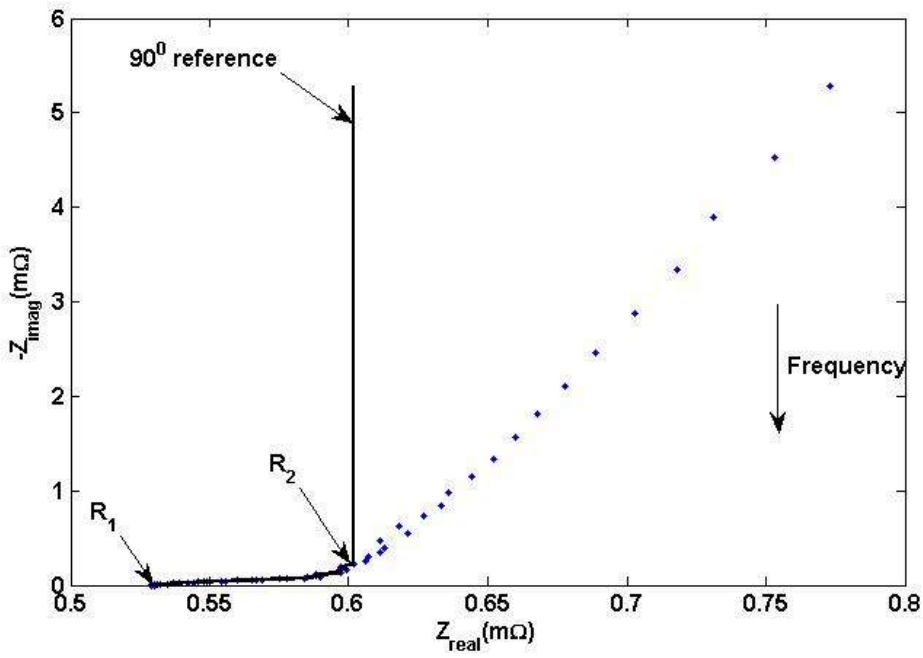


Figure 4.1: Nyquist Plot for a cell biased at 2.5 V

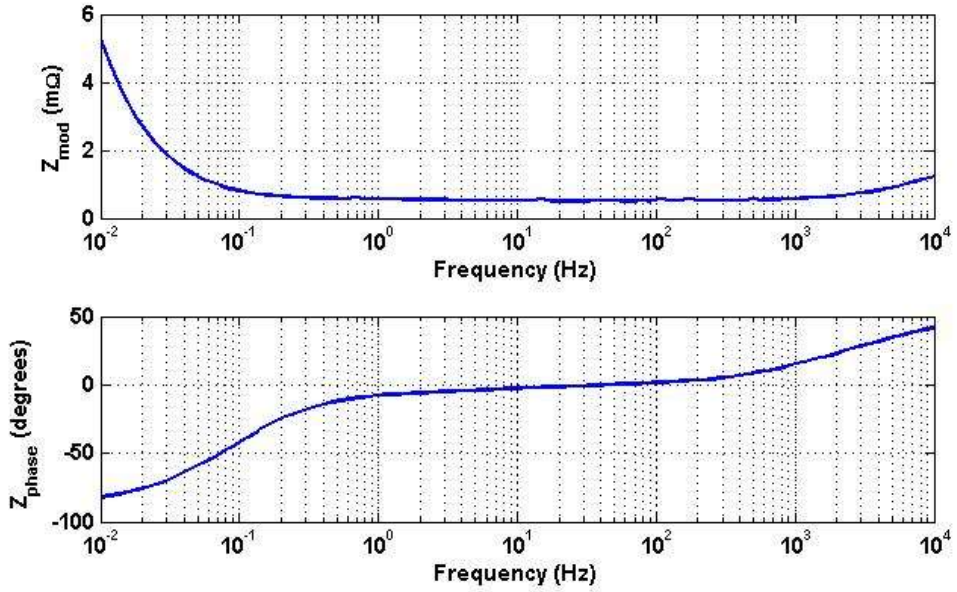


Figure 4.2: Bode Plot for a cell biased at 2.5 V

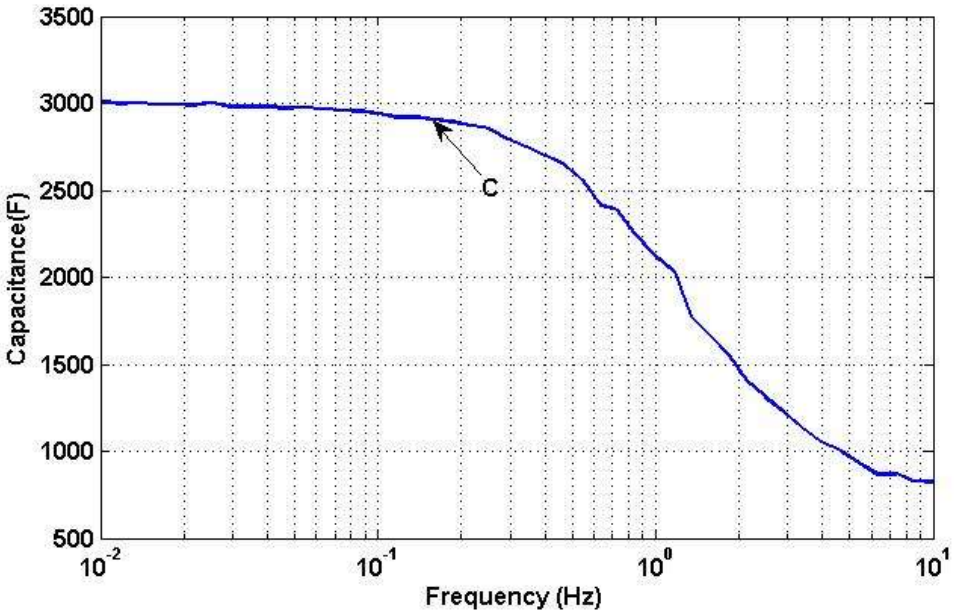


Figure 4.3: Capacitance vs. frequency for a cell biased at 2.5 V

Table 2: Transmission Line Parameters

Parameter	Equation	Values	Standard Deviation
R_s	$R_s = R_1$	0.529 mΩ	$1.41e^{-5}$
r_{el}	$r_{el} = 3\frac{(R_2 - R_1)}{L}$	0.183 Ωm^{-1}	0.016
c_{dl}	$c_{dl} = \frac{C}{L}$	$2.91e^{6}$ Fm^{-1}	8983
R_{dc}	$R_{dc} = R_s + \frac{r_{el}L}{3}$	0.59 mΩ	$8.94e^{-6}$
γ	$\gamma = -\frac{\phi}{90°}$	0.978	0.0018

To validate the supercapacitor model generated, the values in table 2 were used in a Bisquert Open model available in the impedance analyzer's software tools to fit a model to experimental results. The Bisquert Open model can be used to represent the transmission line behavior of the supercapacitor model [36]. From Figures 4.4-4.5 obtained, it is clear that the theoretical model results closely follow those of the test.

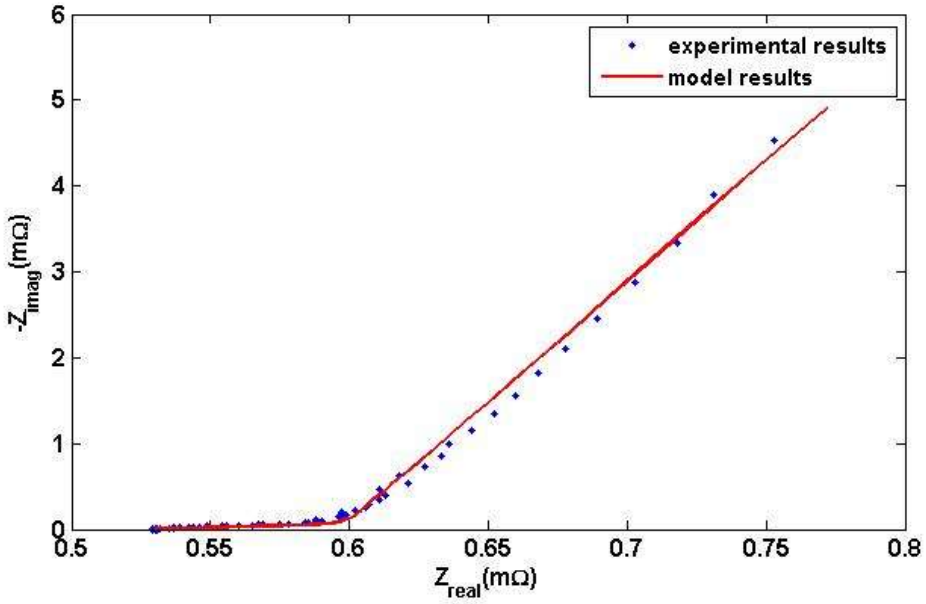


Figure 4.4: Nyquist Plot comparing experimental and model results

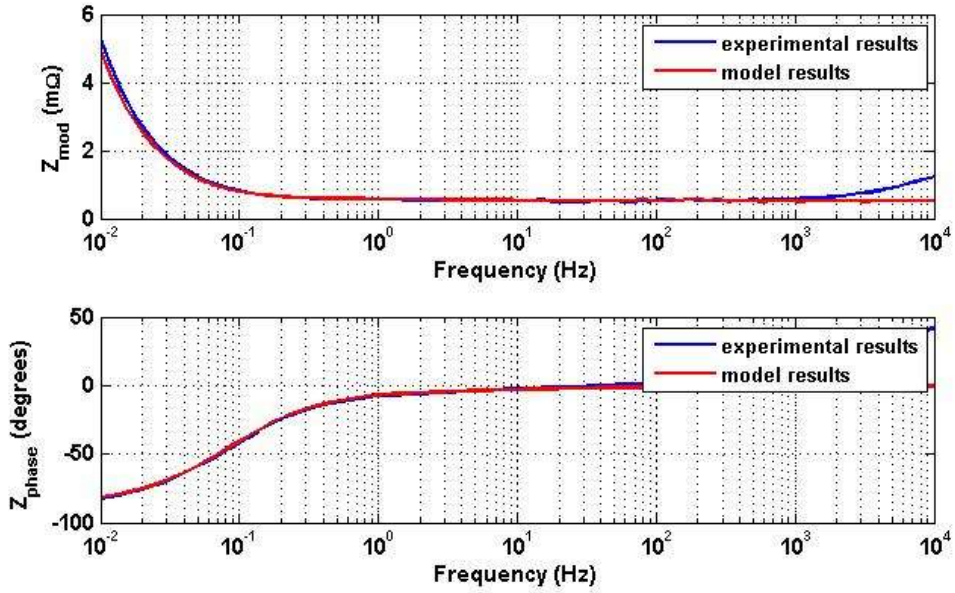


Figure 4.5: Bode Plot comparing experimental and model results

4.2 Constant Current Test

Figures 4.6-4.9 shows the voltage and current of the cell during constant current test at different test currents.

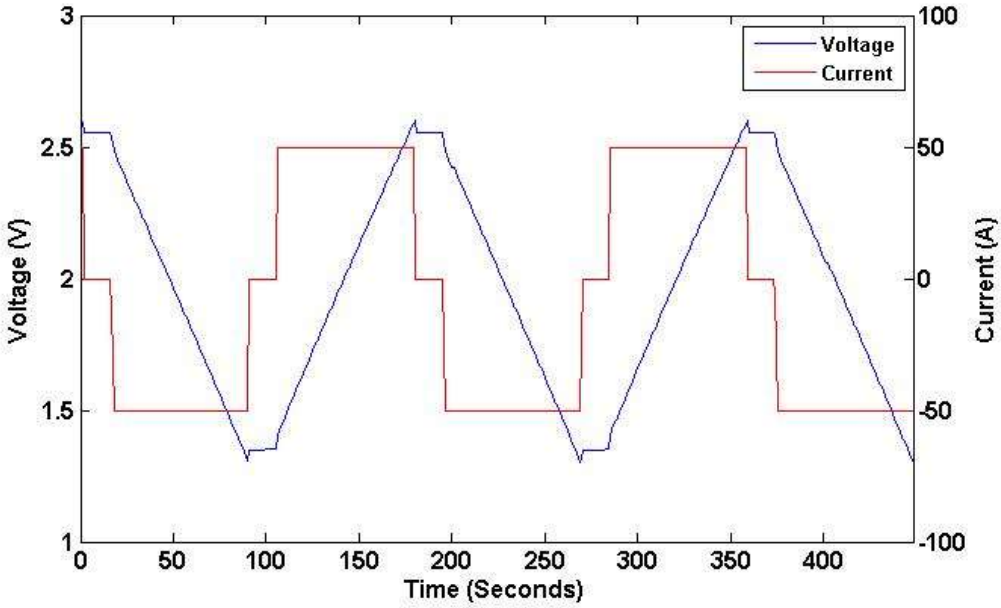


Figure 4.6: Constant Current Test Profile at I_{test}

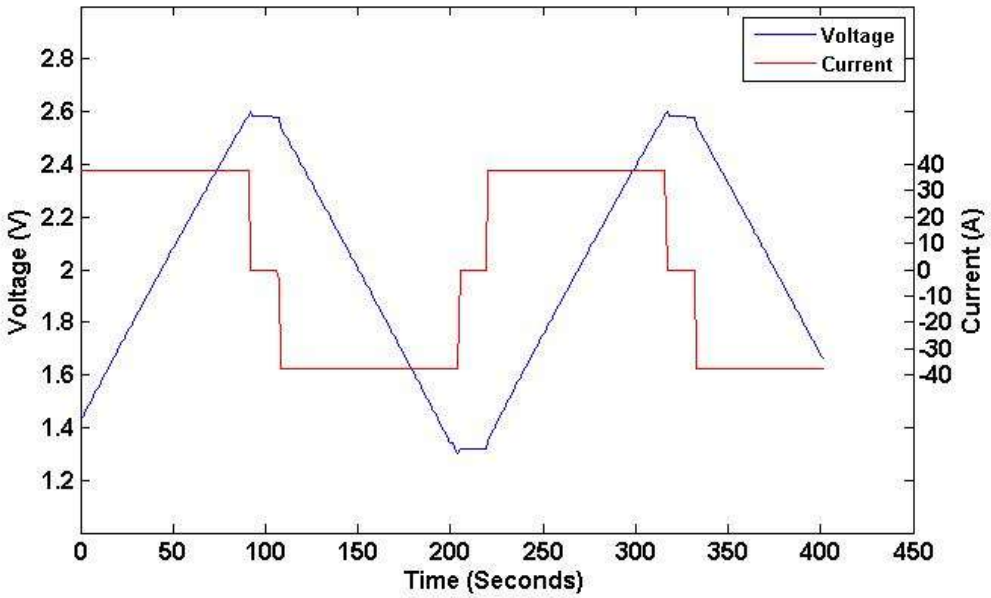


Figure 4.7: Current and Voltage for $0.75I_{test}$

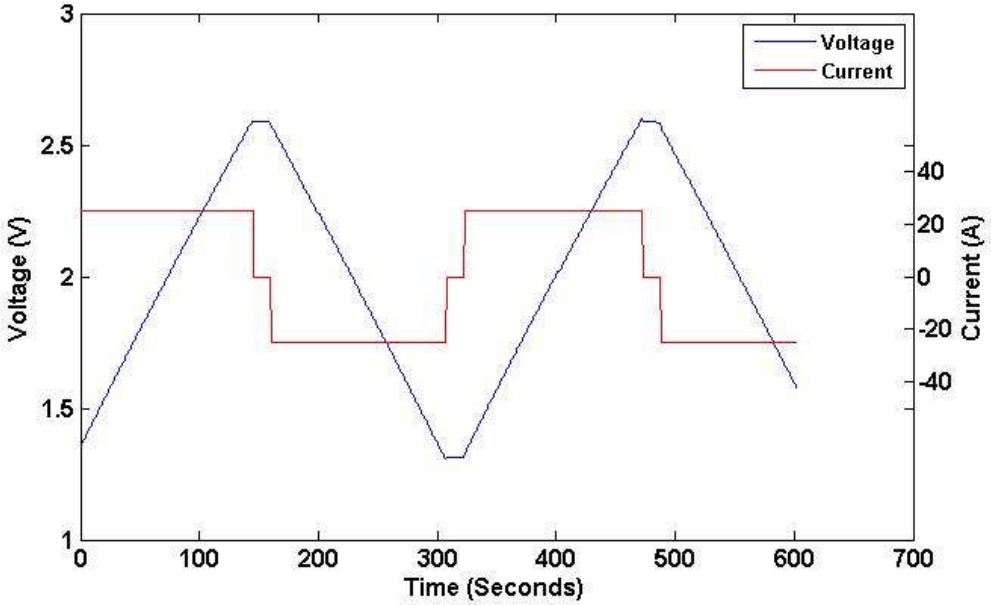


Figure 4.8: Current and Voltage for $0.5I_{test}$

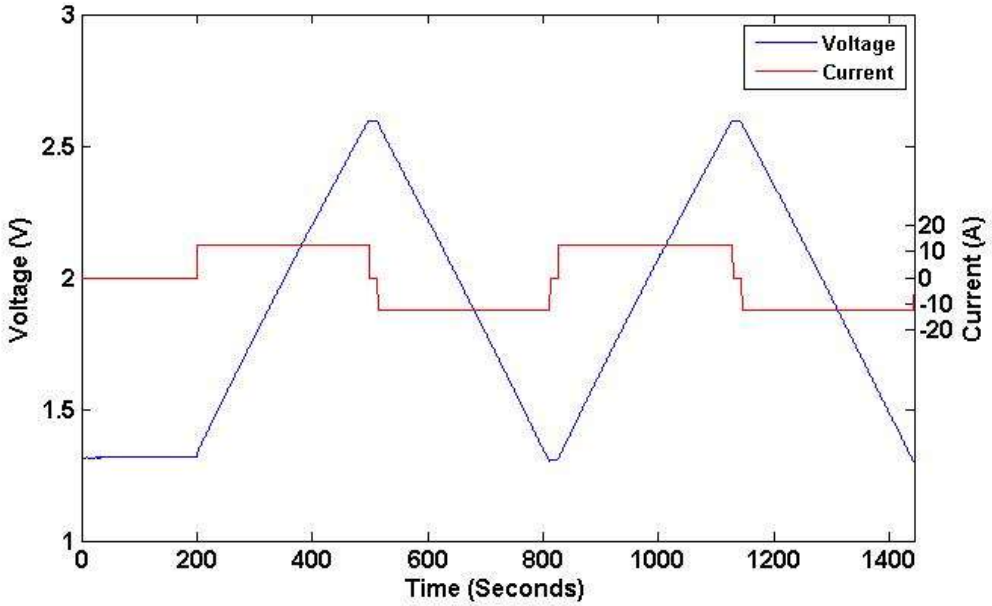


Figure 4.9: Current and Voltage for $0.25I_{test}$

The current and voltage are used to calculate the power for the charge and discharge intervals from which the round trip efficiency is calculated using (38)-(39). Figure 4.10 shows that the efficiency is independent of the test current.

$$\eta = \frac{\text{Energy}_{\text{discharge}}}{\text{Energy}_{\text{charge}}} \% \tag{38}$$

$$\eta = \frac{\text{Average Power}_{\text{discharge}} \text{x Time}_{\text{discharge}}}{\text{Average Power}_{\text{charge}} \text{x Time}_{\text{charge}}} \% \tag{39}$$

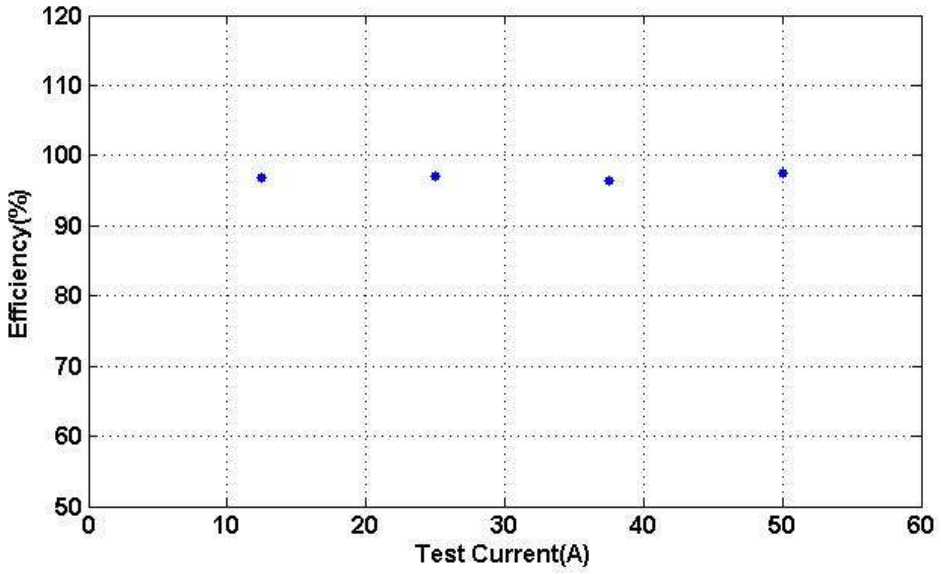


Figure 4.10: Efficiency for different test currents

The capacitance and ESR are also calculated from the voltage and current measurements from this test; the equations and values are given in table 3.

Table 3: Constant Current Test Parameters

Parameter	Equation	Value
C_{charge}	$C_{charge} = \frac{I_2(t_2 - t_1)}{V_2 - V_1}$	2986 F
$C_{discharge}$	$C_{discharge} = \frac{I_5(t_5 - t_4)}{V_5 - V_4}$	2899 F
ESR_{charge}	$ESR_{charge} = \frac{(V_2 - V_3)}{I_2}$	0.727 mΩ
$ESR_{discharge}$	$ESR_{discharge} = \frac{(V_5 - V_6)}{I_5}$	0.983 mΩ

4.3 Leakage Current Measurement Test

Figure 4.11 shows the voltage recorded during the leakage current measurement test. From this figure the slope is calculated and used in (40) to calculate the leakage current; 5.77 mA was obtained as the leakage current after 72 hours. The leakage resistance is calculated to be 450 Ω using (41) where $V_{initial}$ is the initial voltage of the cell. The energy loss due to leakage is also calculated using (42) and is shown in figure 4.12.

$$I_{lk} = C\frac{dv}{dt} \tag{40}$$

$$R_{lk} = \frac{V_{initial}}{I_{lk}} \tag{41}$$

$$E_{lk} = \int_0^t v(t)\, I_{lk} dt \tag{42}$$

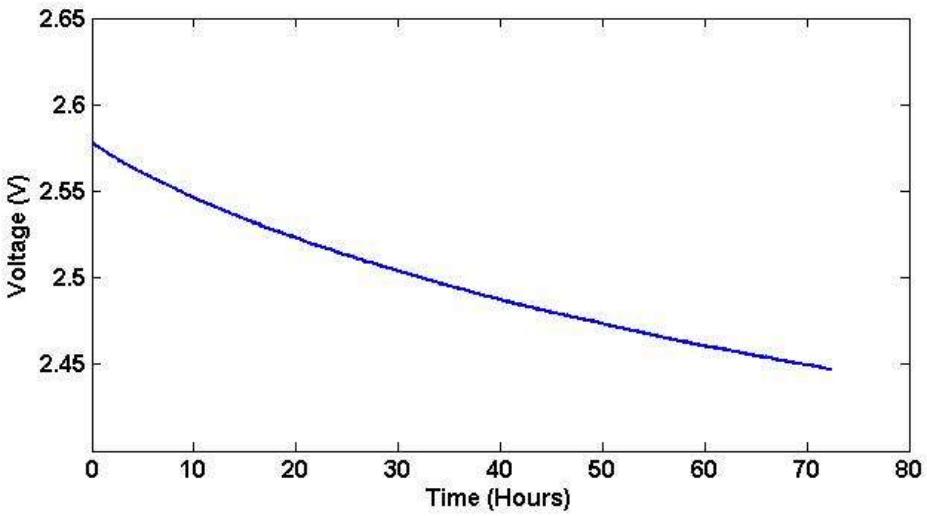


Figure 4.11: Voltage drop due to leakage

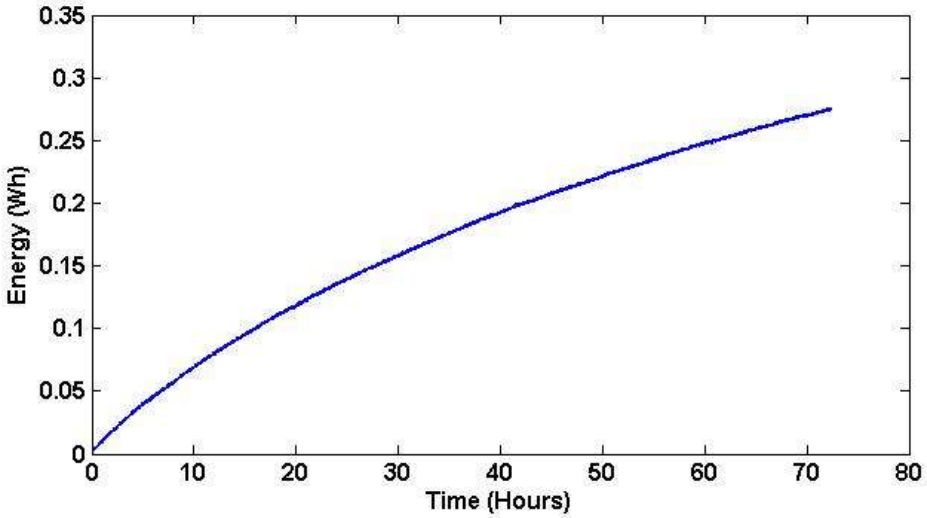


Figure 4.12: Energy lost due to leakage

The self discharge can be represented as a fraction of the total energy lost over time using (43) or as a fraction of energy over the operating voltage range lost over time using (44) [31]. Figure 4.13 shows the initial self discharge of the cell.

$$SD(t) = \frac{V_{initial}{}^2 - v(t)^2}{V_{intial}{}^2} \text{ X } 100 \tag{43}$$

$$SD(t) = 100 \text{ X } \frac{(V_{initial}{}^2 - V_{low}{}^2) - (v(t)^2 - V_{low}{}^2)}{(V_{initial}{}^2 - V_{low}{}^2)} \text{ X } 100 \tag{44}$$

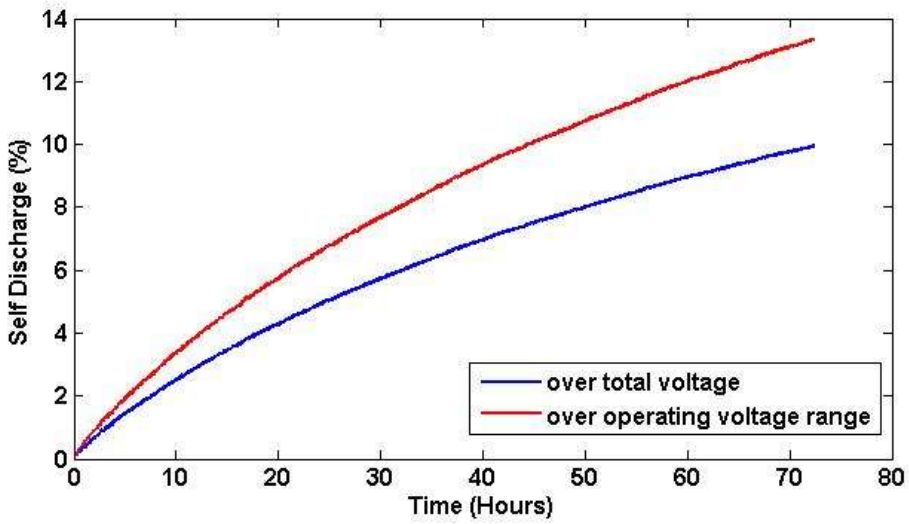


Figure 4.13: Initial Self Discharge due to leakage

4.4 Lifetime Estimation Test

10,000 cycles were completed for the cycling test and the results are shown in figures 4.14-4. 17. A recovery phenomenon is observed, this is where the degradation of the cell parameters deviates from the expected curve because the cell is allowed to rest between the constant current test and EIS test [38].

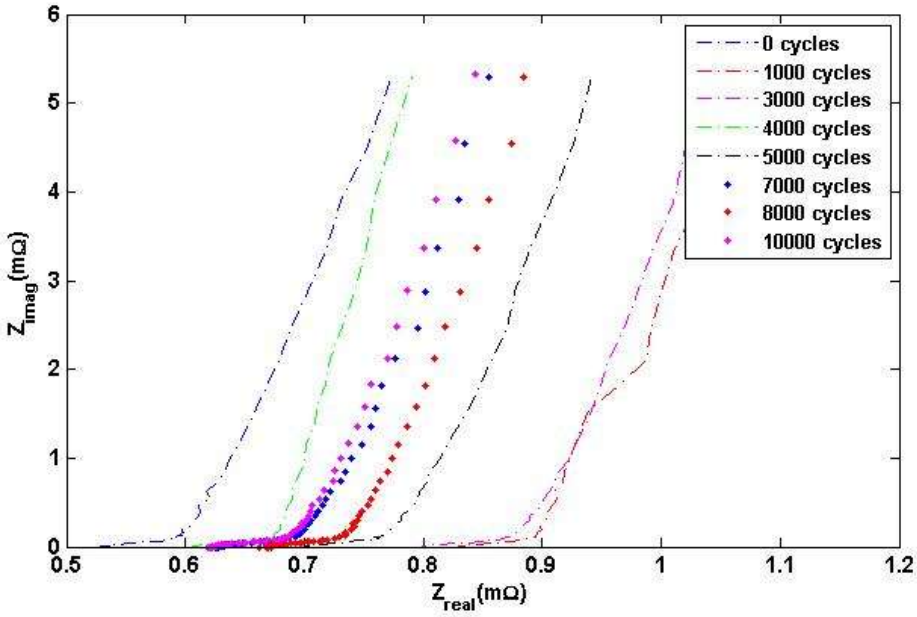


Figure 4.14: Nyquist plot for a cell under cycling testing

The shifting of the complex impedance curve along the axis shows an increase in the series resistance and electrolyte resistance.

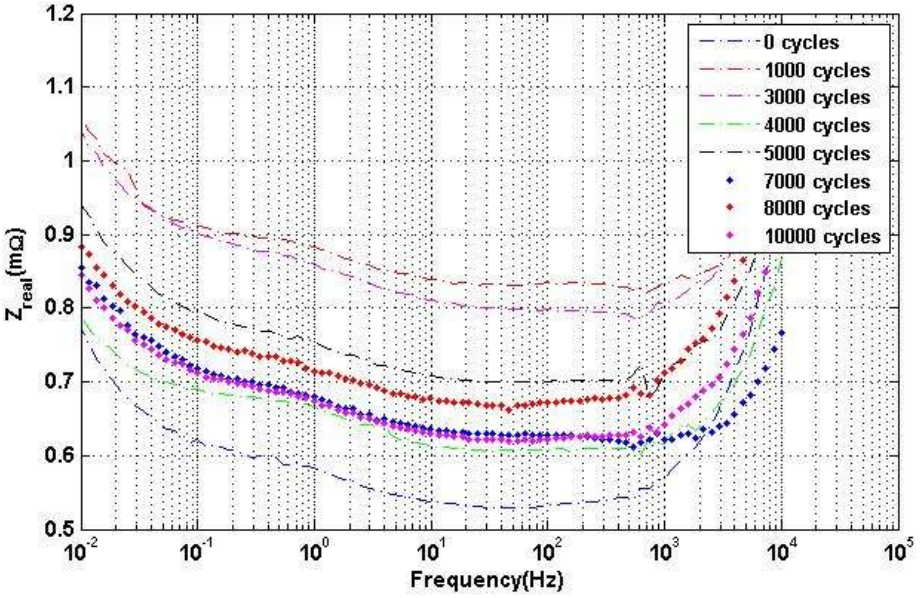


Figure 4.15: Z_{real} vs frequency plot for a cell under cycling testing

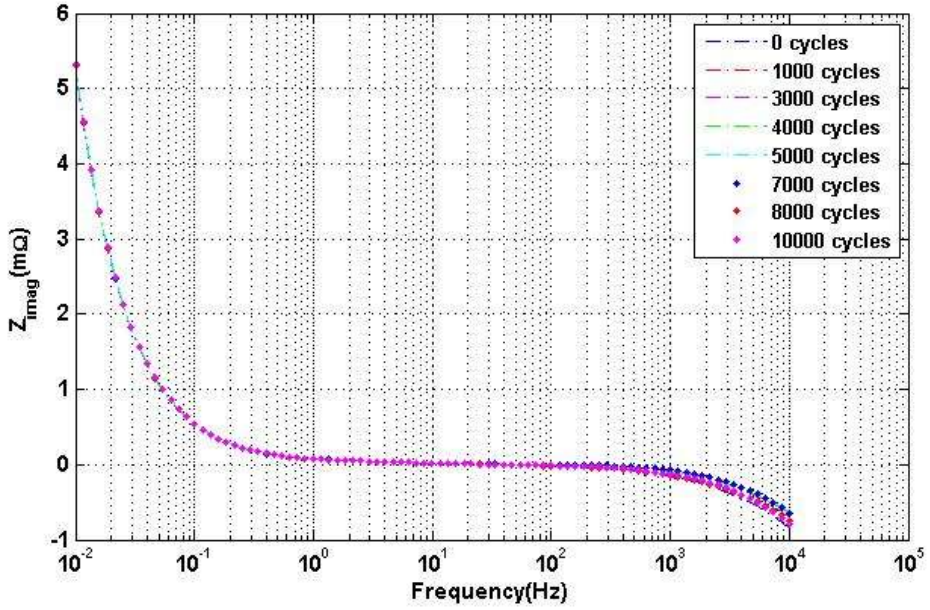


Figure 4.16: Z_{imag} vs frequency plot for a cell under cycling testing

There is a larger deviation in the real part of the cell's impedance than in the imaginary part of the cell's impedance; therefore we expect the resistances to show a faster rate of ageing than the capacitance of the cell.

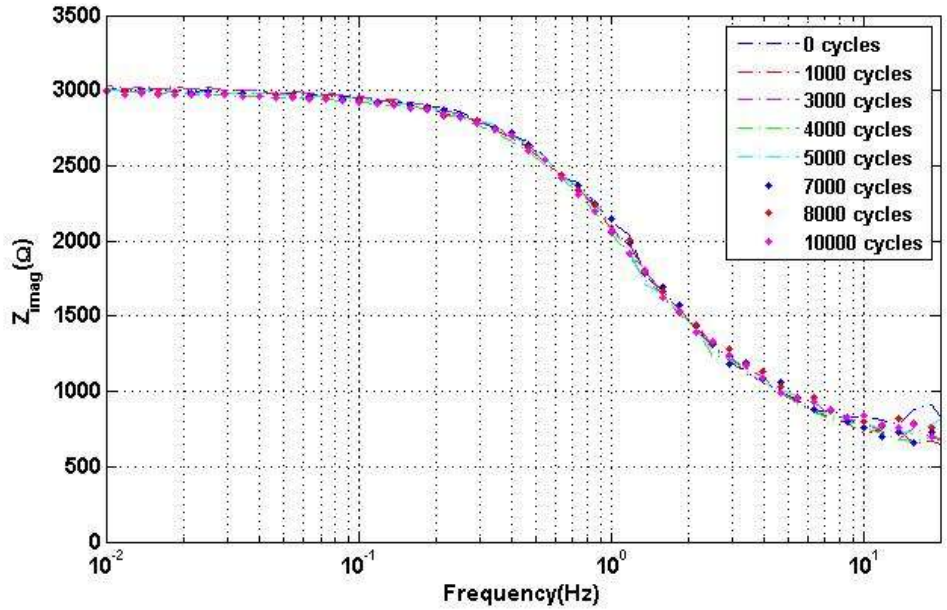


Figure 4.17: Capacitance vs. frequency plot for a cell under cycling testing

A leakage test was also carried out on the cell after the 10,000 cycles of the constant current test were completed; the results are shown in figure 4.18. Both self discharge parameters decrease, the self discharge measured over the operating voltage range decays faster by at least 1.5times than that over the total voltage range.

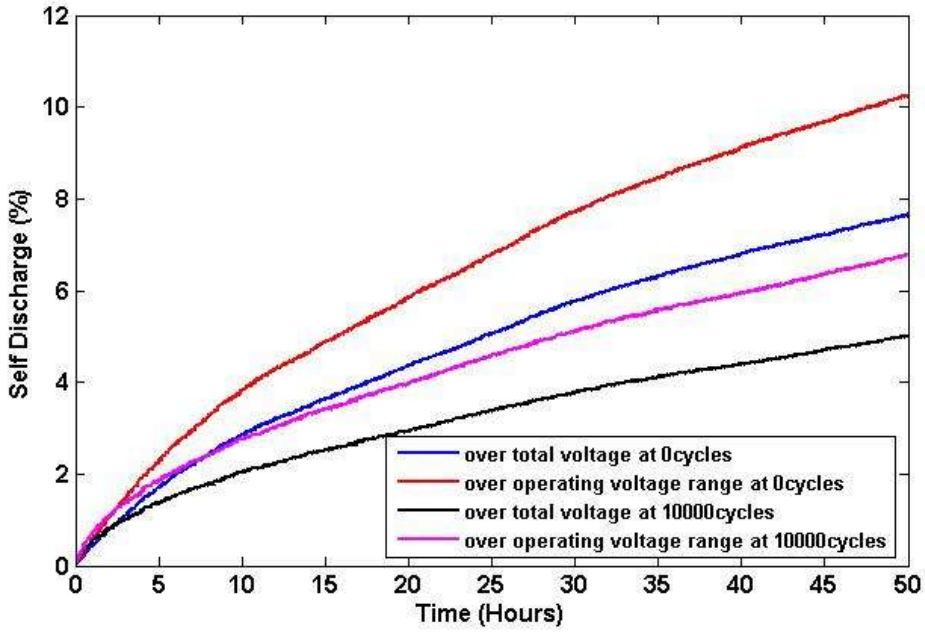


Figure 4.18: Self-discharge change for 10,000 cycles

The degradation of the supercapacitor model parameters plotted in figures 4.19-4.24 is extracted from the cycling test results above, any data showing recovery is eliminated.

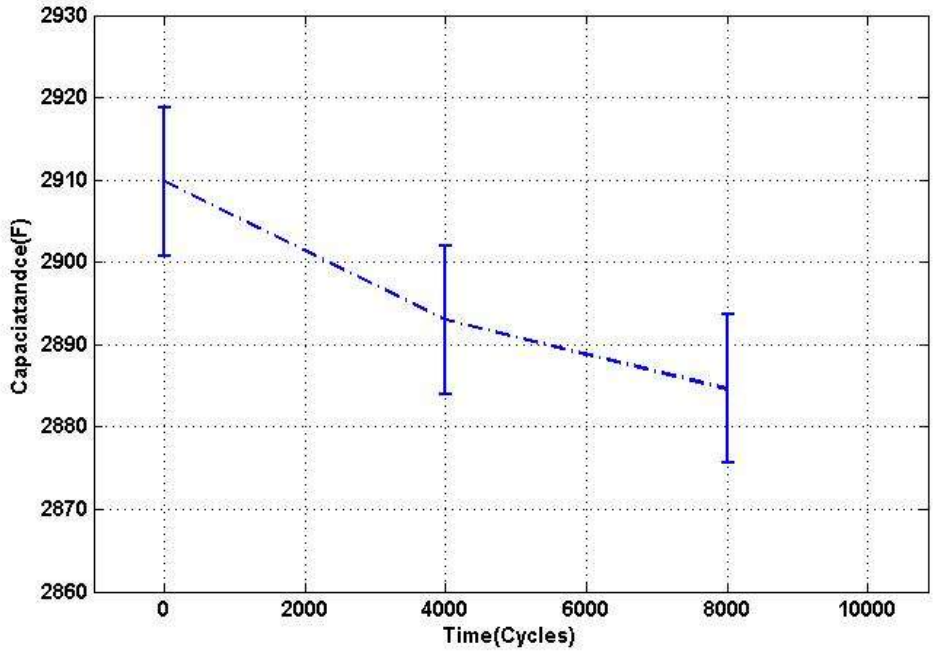


Figure 4.19: Capacitance change for 10,000 cycles

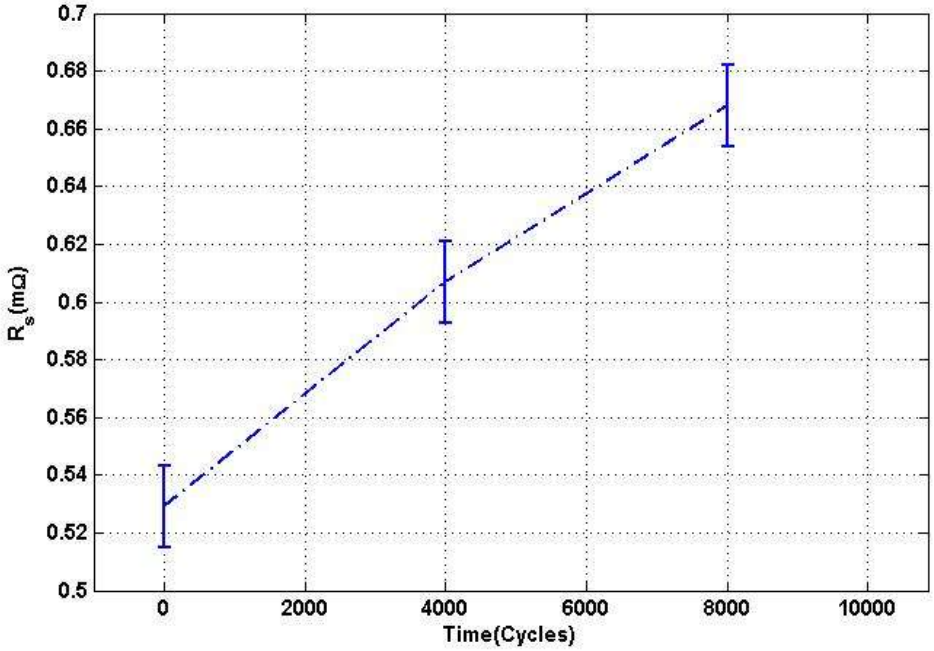


Figure 4.20: R_s change for 10,000 cycles

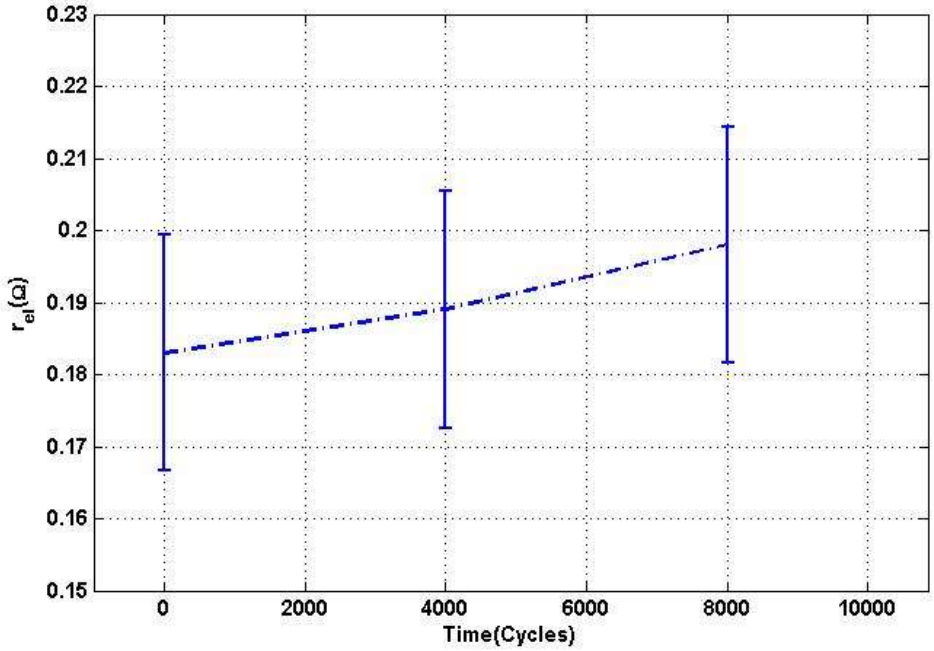


Figure 4.21: r_{el} change for 10,000 cycles

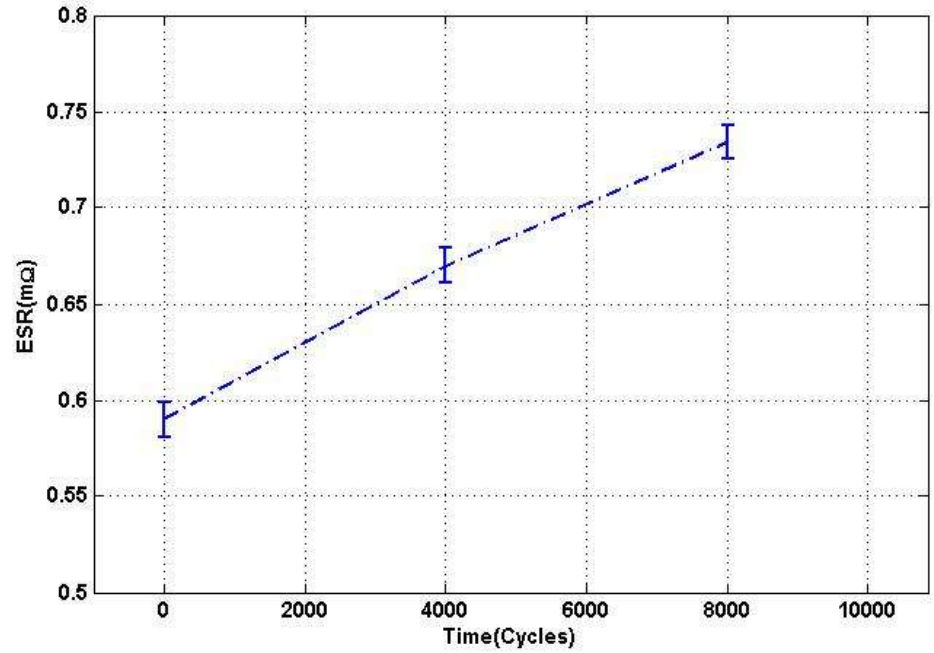


Figure 4.22: R_{dc} change for 10,000 cycles

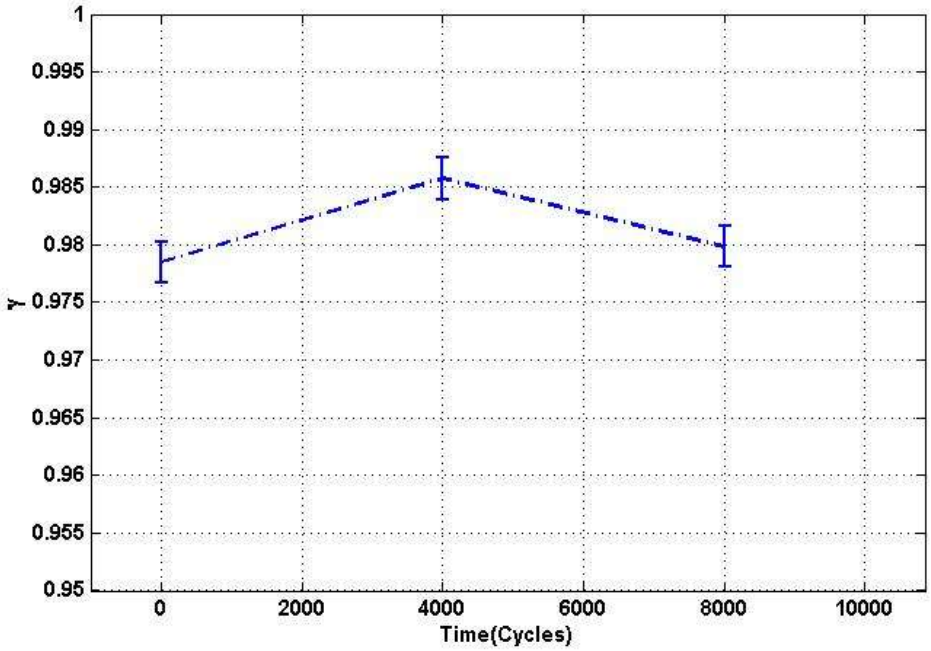


Figure 4.23: ϒ change for 10,000 cycles

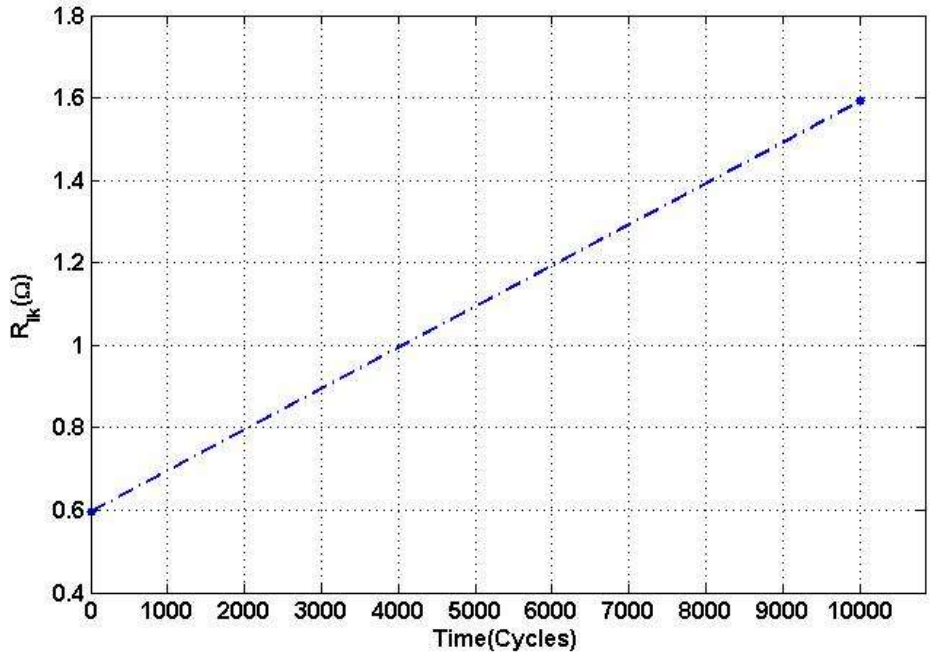


Figure 4.24: R_{lk} change for 10,000 cycles

The capacitance decreases over time; a Maxwell supercapacitor's ageing period is divided into; an initial exponential decrease followed by a linear decrease and lastly a slow exponential decrease [37]. Another observation made is the slower rate of ageing in capacitance than in the resistances by a factor of at least 20. Also the electrolyte resistance does not degrade at the same rate as the series resistance; the rates of degradation differ slightly. The pore size distribution parameter does not show significant change over time. The leakage resistance shown in figure 4.24 is measured after 1 hour to fit the application, figure 4.24 shows that the leakage resistance has the most degradation over time of all the supercapacitor model parameters.

4.5 In-lab Grid System Test

Simulation using the AGC scheme was run for 24 hours. During this run several parameters were tracked and they are plotted in figures 4.25-4.28. We observe that power error is reduced to less than +/-0.2 per unit indicating that the supercapacitors are able to buffer rapid energy demands from the battery.

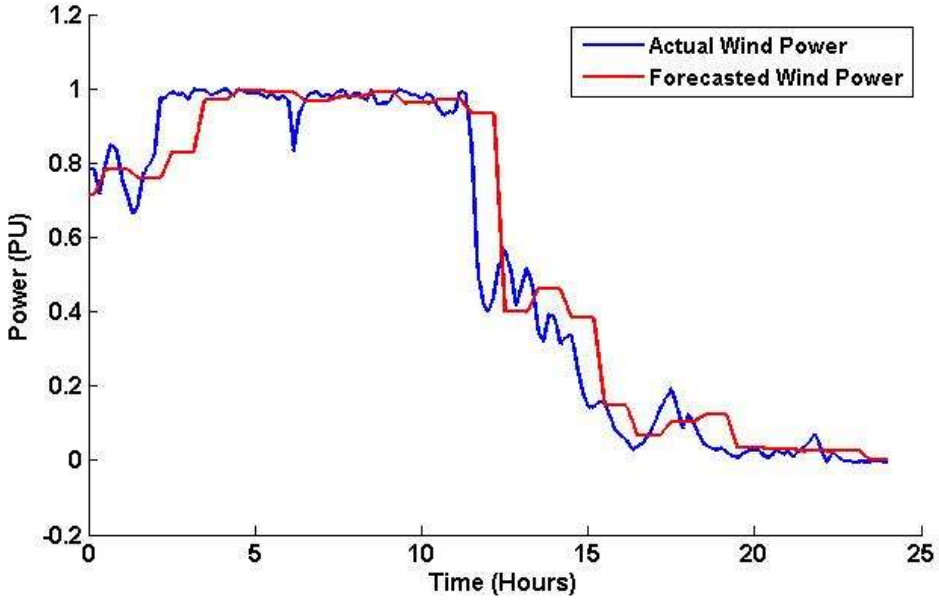


Figure 4.25: Wind farm output power vs. Forecasted wind power

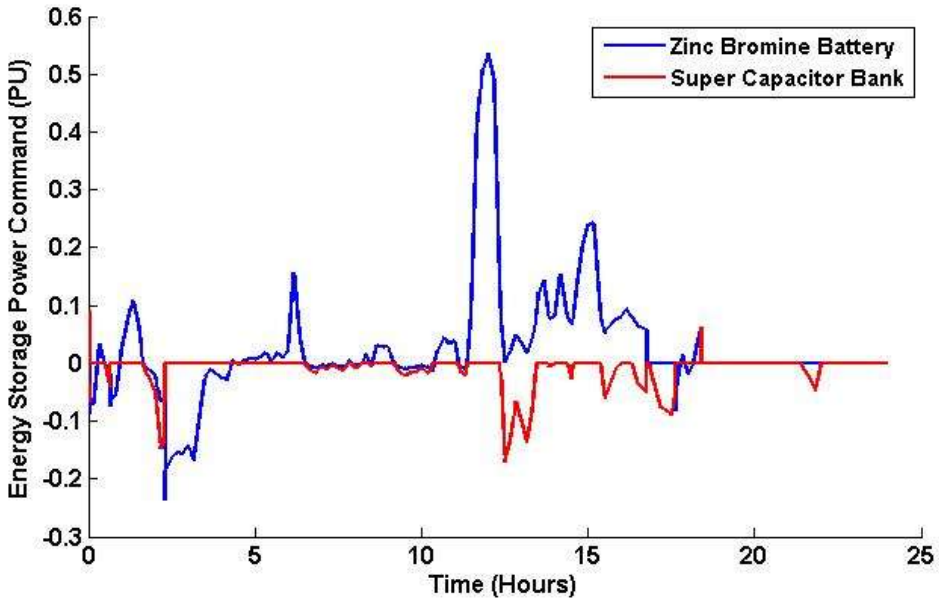


Figure 4.26: Power command to energy storage systems

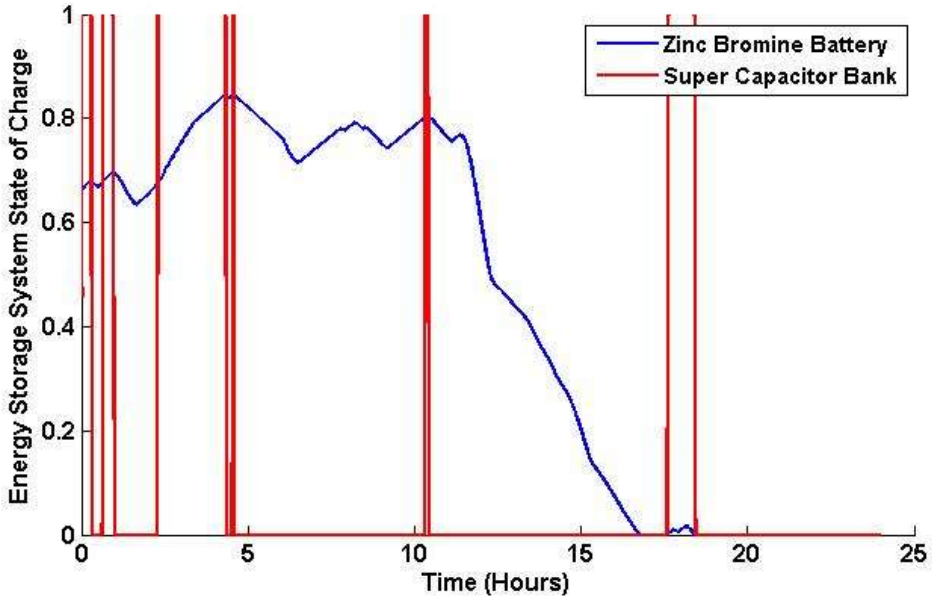


Figure 4.27: State of charge (SOC)

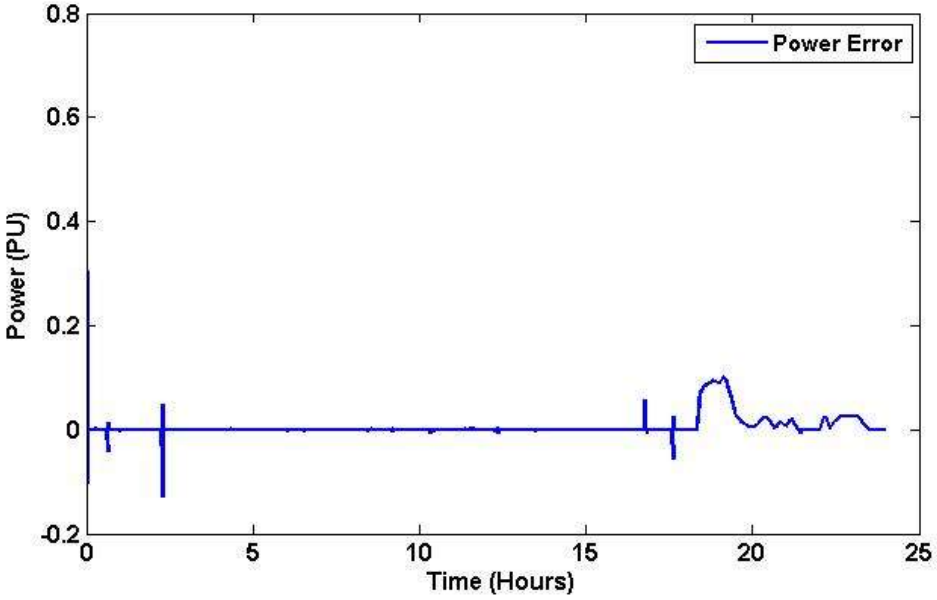


Figure 4.28: Total power error (including energy storage devices)

5 CONCLUSION

5.1 Summary

This work presented the implementation of a supercapacitor storage system in an in-lab grid that is utilized in research on improving wind energy integration. A procedure for sizing a supercapacitor storage system for this application was presented. A model was developed to represent the system and utilized in characterization tests. It was also used in a control algorithm to run simulation tests on the system. The results generated from the characterization tests validate the supercapacitor model. Lifetime estimation testing was completed for over 10,000 cycles; the results are used to observe the change in the supercapacitor model parameters. The leakage resistance is shown to degrade the fastest; it is nearly 50% of its original value after 10,000 cycles. The double layer capacitance degrades the slowest and it is shown to be only close to 99% of its original value on utilizing the cell for 10,000 cycles. A control algorithm is designed for the supercapacitors to meet fast response energy demands while the battery meets medium response energy demands. The control algorithm simulation results show that the designed system including energy storage meets the forecasted power with less than +/- 4% per unit error 90% of the time.

5.2 Future Work

Further work on this research could entail a continuation of the lifetime estimation testing. According to cycle test data provided by the cell's manufacturer [37], the cell is deficient when its capacitance is equal to or less than 80% of its original value and the ESR is equal to or more than twice its original value. This information was obtained for different test conditions than this application; it will be useful to estimate the ageing behavior for the given application. To find the point where the cell is deficient more cycling test data is required from which the capacitance and ESR degradation can be extrapolated. From the cycling test data gathered so far, the recovery behavior is significant therefore during further testing care should be taken to minimize further the any rest periods in between cycling. If there is a possibility to use the same application at different temperature and voltage conditions (the operating temperature range for a Maxwell supercapacitor is -40°C to +65°C) then the temperature effect and voltage effect on cycling may be investigated; various cells set to different temperatures and voltages can be tested from which ageing parameters described in section 3.3.4 for temperature and voltage for this particular cell can be determined.

BIBLIOGRAPHY

[1] GTM Research, Grid Scale Energy Storage: Technologies and Forecasts Through 2015, http://www.greentechmedia.com/research/report/grid-scale-energy-storage-technologies-and-forecasts-through-2015.

[2] AWEA, 20% Wind Energy by 2030, http://www.awea.org/documents/factsheets/20percent_Windfactsheet.pdf.

[3] Brekken T.K.A., Yokochi A., von Jouanne A., Yen Z., Hapke H., Halamay D., "Optimal energy storage sizing and control for wind power applications," IEEE Transactions on Sustainable Energy, vol. 2, pp. 69, Aug. 2010.

[4] Andrews J., Jelley N., Energy science- principles, technologies and impacts, Oxford: New York, 2007, pp.283-292.

[5] Brekken T.K.A., Contemporary Energy Application Class Notes, Spring Term 2011.

[6] Guerrero M.A., Romero E., Barrero F., Milanés M.I., González E., "Overview of medium scale energy storage systems," Compatibility and Power Electronics, 2009, pp. 93-100.

[7] Spahi E., Balzer G., Hellmich B., Münch W., "Wind energy storages - possibilities," PowerTech, 2007, pp. 615-620.

[8] Schainker R.B., "Executive overview: energy storage options for a sustainable energy future," Power Engineering Society General Meeting, IEEE, vol. 2, pp. 2309-2314, June 2004.

[9] Liyan Q., Wei Q., "Constant power control of DFIG wind turbines with supercapacitor energy storage," IEEE Transactions on Industry Applications, vol. 47, pp. 359-367, Nov. 2010.

[10] Zhenhua J., Xunwei Y., "Modeling and control of an integrated wind power generation and energy storage system," Power and Energy General Society Meeting, IEEE, 2009, pp. 1-8.

[11] Shuang Y., Mays T., Dunn R., "A new methodology for designing hydrogen energy storage in wind power systems to balance generation and demand," International Conference on Sustainable Power Generation and Supply, 2009, pp.1-6.

[12] Voller S., Al-Awaad A., Verstege J., "Wind farms with energy storage integrated at control power market," IEEE Symposium on Integration of Wide-Scale Renewable Resources into the Power Delivery System, 2009, pp. 1-13.

[13] Ummels B.C., Pelgrum E., Kling W.L., "Integration of large-scale wind power and use of energy storage in the Netherlands electricity supply," Renewable Power Generation, IET, vol. 2, pp. 34-36, Mar. 2008.

[14] Abbey C., Joos G., "Supercapacitor energy storage for wind energy applications," IEEE Transactions on Industry Applications, vol.43, pp. 769-776, May-June 2007.

[15] Srithorn P., Summer M., Yao L., Parashar R., "The control of a STATCOM with supercapacitor energy storage for improved power quality," CIRED Seminar on Smart Grids for Distribution, 2008, pp. 1-4.

[16] Kuava R., Ramos R.A., Bretas N.G., "Control design of a STATCOM with energy storage system for stability and power quality improvements," IEEE International Conference on Industrial Technology, 2009, pp. 1-6.

[17] Zhengping X., Parkhideh B., Bhattacharya S., "Improving distribution system performance with integrated STATCOM and supercapacitor energy storage system," IEEE Power Electronics Specialists Conference, 2008, pp. 1390-1395.

[18] Molina M.G., Mercado P.E., "Power flow stabilization and control of microgrid with wind generation by superconducting magnetic energy storage," IEEE Transactions on Power Electronics, vol. 26, pp.910-922, Dec. 2010.

[19] Sander M., Gehring R., "LIQHYSMES-a novel energy storage concept for variable renewable energy sources using hydrgen and SMES," vol. 21, pp. 1362, Nov. 2010.

[20] Ise T., Kita M., Taguchi A., "A hybrid energy storage with a SMES and secondary battery," vol. 15, pp. 1915, June 2005.

[21] Belhachemi F., Rael S., Davat B., "A physically based model of power electric double layer capacitors," Industry Applications Conference, IEEE, vol. 5, pp. 3069-3076, Oct. 2000.

[22] Zubieta L., Bonert R., "Characterization of double-layer capacitors (DLCs) for power electronics applications," Industry Applications Conference, IEEE, vol. 2, pp. 1149-1154, Oct. 1998.

[23] Rizoug N., Bartholomeus P., Le Moigne P., "Modeling and characterizing supercaapcitors using an online method," IEEE Transactions on Inustrial Electronics, vol. 57, pp. 3980-3990, Feb. 2010.

[24] Bohlem O., Kowal J., Uwe Sauer D., "Ageing behavior of electrochemical double layer capacitors Part I. experimental study and ageing model," Journal of Power Sources, vol. 172, pp.468-475, 2007.

[25] El-Brouji E.L.H., Briat O., Vinassa J.-M., Bertrand N., Woirgard E., "Impact of calendar life and cycling ageing on supercapacitor perfomance," IEEE Transactions on Vehicular Technology, vol. 58, pp. 3917, July 2009.

[26] Lajinef W., Vinassa J.-M., Briat O., Azzopardi S., Woirgard E., "Characterization methods and modelling of ultracapacitors for use as peak power sources," Journal of Power Sources, vol. 168, pp. 553-560, 2007.

[27] Song H.-K., Hwang H.-Y, Lee K.-H, Dao L.H., "The effect of pore size distribution on the dispersion of porous electrodes," Electrochimica Acta, vol. 45, pp. 2241-2257, 2000.

[28] Arnet B.J., Haines L.P., "High power DC-to-DC converter for supercapacitors," Electric Drives and Machines Conference, IEEE International, 2001, pp. 985-990.

[29] Gamry Instruments, Reference 3000 Potentiostat Operator's Manual.

[30] Maxwell Technologies, Test Procedures for Capacitance, ESR, Leakage Current and Self-Discharge Characterizations for Ultracapacitors, http://www.maxwell.com/products/ultracapacitors/docs/APPLICATIONNOTE_MAXWELLTESTPROCEDURES.PDF

[31] DOE, FreedomCAR Ultracapacitor Test Manual, http://avt.inel.gov/battery/pdf/FreedomCAR_Capacitor_Test_Manual_Sept_2004.pdf

[32] Paul K., Christian M., Pascal V., Guy C., Gerard R., Younes Z., "Constant power cycling for accelerated ageing of supercapacitors," 13th European Conference on Power Electronics and Applications, 2009.

[33] Hammar A., Venet P., Lallemand R., Coquery G. and Rojat G., "Study of accelerated aging of supercapacitors for transport applications," IEEE Transactions on Industrial Electronics, vol. 57, pp. 3972, April 2010.

[34] Azais P. et al., "Causes of supercapacitors ageing in organic electrolyte," Journal of Power Sources, vol. 171, pp. 1046-1053, 2007.

[35] Alcicek G., Gualous H., Venet P., Gallay R., Miraoui A., "Experimental study of temperature effect on ultracapacitor ageing," European Conference on Power Electronics and Applications, pp. 1, Sept. 2007.

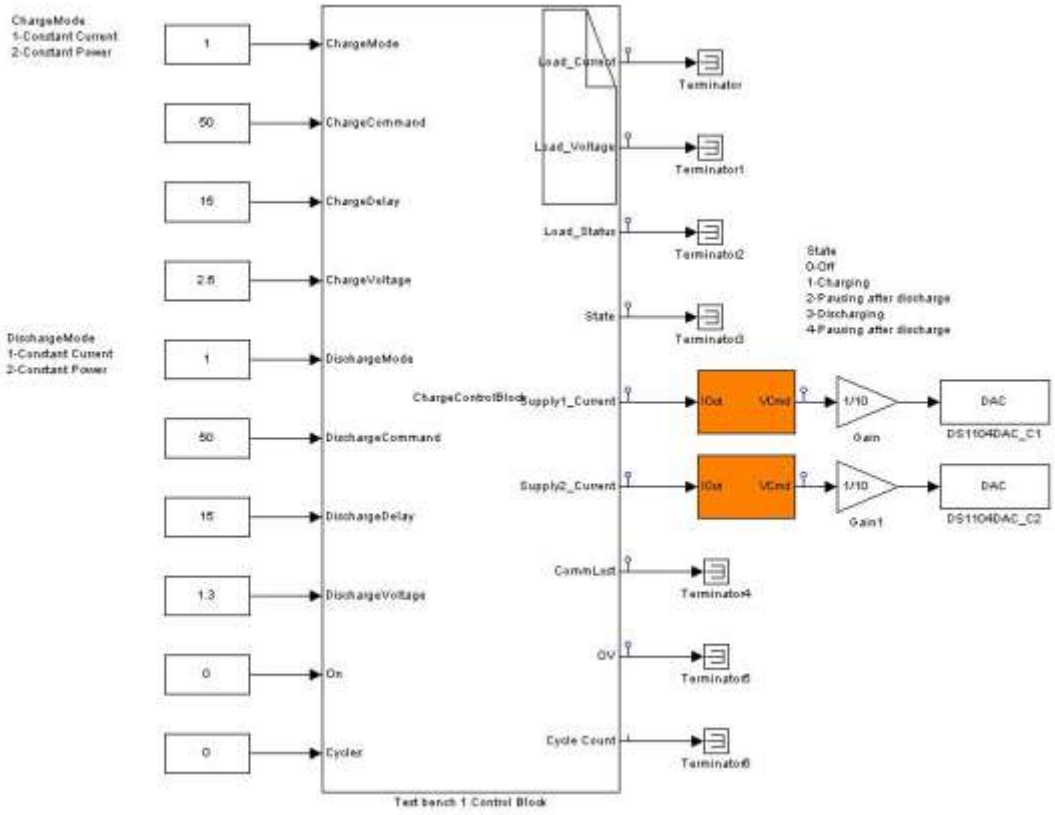
[36] Bisquert J., "Theory of the impedance of electron diffusion and recombination in a thin layer," Journal of Physical Chemistry B, vol. 106, pp. 325-333, 2002.

[37] Maxwell Technologies, Boostcap Energy Storage Modules Life Duration Estimation, http://www.maxwell.com/products/ultracapacitors/docs/APPLICATIONNOTE1012839_1.PDF

[38] El-Brouji E.L.H., Vinassa J.-M., Briat O., Bertrand N., Deletage J.-Y., Woirgard E., "Ageing assessment of supercapacitors during calendar life and power cycling tests," IEEE Energy Conversion Congress and Exposition, pp. 1791, Sept 2009.

APPENDICES

Appendix A: Test Bench 1 Simulink Model



Appendix B: Test Bench 1 Code

```
// SuperCapCyc.c
//
// Written by David Naviaux
// Oregon State University PhD candidate
// 3/29/2011
//
// The following code implements the bulk of a Matlab s-function
block for Eunice Naswali.  The
// s-function block allows cycle testing of a 3000F, 2.7V
supercapacitor for Eunice's research
// and thesis.
//
// A QuadTech 44020 (with a 44003 load module) is connected to the
dSPACE DS1104 (or DS1103)
```

```
// RS232 port and up to two HP 6269B power supplies configured for
remote current control are
// connected to the dSPACE DAC channel 1 and 2 outputs.  If only
one supply is used, it must
// be connected to DAC channel 1.
//

// only     compile     this file if we   are   actually generating
      code
#ifndef     MATLAB_MEX_FILE
#include <dstypes.h>
#include <simstruc.h>
#include <dsserdef.h>
#include <dsser.h>
#include <dsts.h>
#include <dsmsg.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "ChargeControl.h"

#define     SIZEOF_RXQ 2048               /* size of our local
RS232 receive buffer */
#define     SIZEOF_RXFIFO 1024            /* size of the dSPACE
serial receive (and transmit) buffer */
#define SIZEOF_TXRXFRAME 256  /* size of our local RS232 command
and response buffers */
#define RSP_TIMEOUT 0.2             /* [s] maximum time to receive
a response our last command */
#define MAX_CHARGE_CURRENT 50.0     /* [A] maximum charge current
*/
#define MAX_ILOWRANGE 6.0           /* [A] maximum current when the
load is in the low current range */
#define MAX_PLOWRANGE 30.0          /* [W] maximum power when the
load is in the low power range */
#define MAX_RLOWRANGE 100.0         /* [OHMS] maximum resistance
when in the load low resistance range */
#define MAX_VSUPERCAP 2.70          /* [V] maximum voltage allowed
while charging the supercap */
#define HP6269B_OFF -0.02           /* [A] output that will stop
the current from the HP power supply */
#define HP1_MAXCURRENT 50.0         /* [A] maximum current that can
be supplied by first HP6269B */
#define HP2_MAXCURRENT 50.0         /* [A] maximum current that can
be supplied by second HP6269B */

// the following holds a pointer used for most of the dSPACE serial
port related functions
static dsserChannel* serCh;

// the stucture holds a command that will or has been sent to the
QuadTech44020
static struct _tx_ {
```

```
      uint8_T           Buf[SIZEOF_TXRXFRAME + 1];
      uint16_T    uBufLen;
} tx;

// holds a command response from the QuadTech 44020 that is
currently being constructed
static struct _rx_ {
      uint8_T           Buf[SIZEOF_TXRXFRAME + 1];
      uint16_T    uBufLen;
} rx;

SCC   scc;


////////////////////////////////////////////////////////////////////
////////////////////////////////
//
// This     structure holds   serial receive data     locally.  We
double      buffer the data   this way so that
// we can minimize calls to   the   dsser_receive()   function
(which      has   a lot of overhead)
//
// This     structure is only updated by the UpdateRxq() function.
//
////////////////////////////////////////////////////////////////////
////////////////////////////////
static struct {
      uint16_T    uNextRead;
      uint16_T    uNextWrite;
      uint16_T    uBytesInQ;
      uint8_T           q[SIZEOF_RXQ];
} RxQ;



////////////////////////////////////////////////////////////////////
////////////////////////////////
//
// static UInt8 UpdateRxQ(void)
//
// According to   the   dSPACE documentation, the dsser_receive()
function has significant
// overhead so we want to minimize calls to     this function. To
do that, we use a local
// queue to double buffer the data received from the serial port.
//
// This     function reads as much received     data from the
dSpace receive FIFO as possible.    We only
// read     data from the dSpace buffer   if our local buffer
      is empty.
//
////////////////////////////////////////////////////////////////////
////////////////////////////////
```

```
static UInt8 UpdateRxQ(void)
{
      uint32_T ulCount1;
      uint32_T ulCount2;
      int         i1;
      int         i2;

      /* Read     as much     data from the dSpace receive FIFO as
we    can   */

      /* with     each call to this function,   we will     call
dsser_receive() at      most two times as */
      /* required to fill     our   local queue. */

      /* determine bytes available at     the   end   and   the
      begining of our   local receive queue     */
      if (RxQ.uNextRead <     RxQ.uNextWrite)
      {
            i1 = sizeof(RxQ.q) - RxQ.uNextWrite;      /* bytes at
      end   of queue */
            i2 = RxQ.uNextRead;
      /* bytes at beginning of queue */
      }
      else if     (RxQ.uNextRead > RxQ.uNextWrite)
      {
            i1 = RxQ.uNextRead - RxQ.uNextWrite;      /* bytes at
      end   of queue */
            i2 = 0;
      /* bytes at beginning of queue */
      }
      else if     (!RxQ.uBytesInQ)
      {
            /* uNextRead ==   uNextWrite, this can only happen if
      the   buffer is full or empty */
            /* in this case, it     is empty */
            i1 = sizeof(RxQ.q) - RxQ.uNextWrite;      /* bytes at
      end   of queue */
            i2 = RxQ.uNextWrite;                            /*
bytes at    beginning of queue */
      }
      else
      {
            /* buffer is full.      This should never happen since
we are the master and no one should */
            /* be sending anything without us asking for it. But
you know how that goes... */
            i1 = 0;
            i2 = 0;
      }

      // we can only read     SIZEOF_RXFIFO bytes     at most
      if (i1 > SIZEOF_RXFIFO)
      {
```

```
            i1 = SIZEOF_RXFIFO;
            i2 = 0;
      }
      else if     (i1   + i2 >=     SIZEOF_RXFIFO)
      {
            i2 = i1     + i2 - SIZEOF_RXFIFO;
      }

      /* no data copied yet */
      ulCount1 = ulCount2     = 0;

      /* copy     as much     of the dSPACE buffered data   to our
local queue as we can */
      if (i1)
      {

            /* read     the   first block of data     */
            if ( dsser_receive(serCh, i1, &RxQ.q[RxQ.uNextWrite],
&ulCount1) ==     DSSER_NO_ERROR )
            {
                  /* read     was   successful */

                  /* adjust the index     to the next byte to
      write */
                  if ( (RxQ.uNextWrite += ulCount1) >=
sizeof(RxQ.q))
                        RxQ.uNextWrite = 0;

                  /* adjust the number of bytes in the queue */
                  RxQ.uBytesInQ += ulCount1;

                  /* read     the   next block, if needed */
                  if (i2 && i1==ulCount1)
                  {
                        // read     the   second block of   data
                        if ( dsser_receive(serCh, i2,
&RxQ.q[RxQ.uNextWrite], &ulCount2) ==     DSSER_NO_ERROR )
                        {
                              /* read     was   successful */

                              /* adjust the index     to the next
      byte to     write */
                              if ( (RxQ.uNextWrite += ulCount2) >=
sizeof(RxQ.q))
                                    RxQ.uNextWrite = 0;

                              /* adjust the number of bytes in the
queue */
                              RxQ.uBytesInQ += ulCount2;
                        }
                  }
            }
      }
```

```
      // return true if we copied   some data from the dSpace
receive buffer.
      return(i1 != 0);
}


////////////////////////////////////////////////////////////////////
////////////////////////////////
// int16_T RxChar(void)
//
// Reads a character from our local serial receive queue.  If our
local queue is empty,   the   dSpace
// serial receive buffer will be read into our local buffer.
//
// In:      None
//
// Out:     received character,     -1 if not data available
//
////////////////////////////////////////////////////////////////////
////////////////////////////////
int16_T     RxChar(void)
{
      int16_T     ch;

      // read     as much     of the dSPACE receive buffer as
      we can to our local     buffer,     only if
      // our local buffer     is completely empty
      if (!RxQ.uBytesInQ)
            UpdateRxQ();

      // return a character if one is     available
      if (RxQ.uBytesInQ)
      {
            ch = RxQ.q[RxQ.uNextRead++];
            if (RxQ.uNextRead >= sizeof(RxQ.q))
                  RxQ.uNextRead =   0;

            // one less character available
            RxQ.uBytesInQ--;

            // return the character
            return(ch);
      }
      else
            return(-1);
}


////////////////////////////////////////////////////////////////////
////////////////////////////////
//
// static void FlushLocalRxQ(void)
```

```
//
// Flushes all characters currently in our local serial receive
buffer.
//
////////////////////////////////////////////////////////////////////
//////////////////////////////////
static void FlushLocalRxQ(void)
{
      RxQ.uBytesInQ =   RxQ.uNextRead =   RxQ.uNextWrite = 0;
}


////////////////////////////////////////////////////////////////////
//////////////////////////////////
//
// static void    FlushAllRxQ(void)
//
// Flushes all characters in our local serial receive buffer and
the dSPACE receive buffer.
//
////////////////////////////////////////////////////////////////////
//////////////////////////////////
static void FlushAllRxQ(void)
{
      FlushLocalRxQ();
      RxChar();
      FlushLocalRxQ();
}



////////////////////////////////////////////////////////////////////
//////////////////////////////////
//
// void SendTxBuf(void)
//
// Transmits the command string contained in our transmit command
buffer (tx) out of the dSPACE
// serial port to the QuadTech 44020.
//
////////////////////////////////////////////////////////////////////
//////////////////////////////////
void  SendTxBuf(void )
{
      uint32_T ulCount;
      int16_T iRet;

      // don't execute if the serial port initialization failed
      if (!serCh)
            return;

      // send the command string
      iRet = dsser_transmit(serCh, tx.uBufLen, tx.Buf, &ulCount);
```

```
	switch (iRet)
	{
	case DSSER_NO_ERROR:
		// all bytes were successfully buffered, mark that we
are now waiting for it to get sent
		scc.bTransmitting = YES;
		break;

	case DSSER_FIFO_OVERFLOW:
		/* FIFO overflowed, for now, ignore it */
		msg_printf(MSG_MC_ERROR, MSG_DLG_OKCANCEL, MSG_SM_USER,
iRet, "dsser_transmit() - DSSER_FIFO_OVERFLOW");
		break;

	case DSSER_COMMUNICATION_FAILED:
		msg_printf(MSG_MC_ERROR, MSG_DLG_OKCANCEL, MSG_SM_USER,
iRet, "dsser_transmit() - DSSER_COMMUNICATION_FAILED");
		break;

	default:
		msg_printf(MSG_MC_ERROR, MSG_DLG_OKCANCEL, MSG_SM_USER,
iRet, "dsser_transmit() - Unknown error code: %d", iRet);
		break;
	}
}


////////////////////////////////////////////////////////////////////
//////////////////////////////////
////////////////////////////////////////////////////////////////////
//////////////////////////////////
static void SetChargeCurrent(double dAmps)
{
	if (dAmps < 0.0)
	{
		// turn off the supplies since a negative current was
specified
		scc.outputs.dHP1IChgCmd = scc.outputs.dHP2IChgCmd =
dAmps;
	}
	else if (dAmps <= HP1_MAXCURRENT)
	{
		// all current can be supplied by the first supply, so
do it

		scc.outputs.dHP1IChgCmd = dAmps;
		scc.outputs.dHP2IChgCmd = 0.0;
	}
	else
	{
		// we must use both power supplies to get the required
current
```

```
            // set the first at its maximum
            scc.outputs.dHP1IChgCmd = HP1_MAXCURRENT;

            // set the second as required
            if (dAmps >= HP1_MAXCURRENT + HP2_MAXCURRENT)
                  scc.outputs.dHP2IChgCmd = HP2_MAXCURRENT;
            else
                  scc.outputs.dHP2IChgCmd = dAmps - HP1_MAXCURRENT;
      }
}


////////////////////////////////////////////////////////////////////
////////////////////////////////
//
// static void ChargeCurrentTask(void)
//
// This function handles the charge current outputs (HP1IChgCmd and
HP2IChgCmd) of the s-function
// block.  These outputs are only changed from within this block.
//
////////////////////////////////////////////////////////////////////
////////////////////////////////
static void ChargeCurrentTask(void)
{
      double dI;

      // make sure that the maximum supercapitor voltage has not
been exceeded
      if (!scc.outputs.bCapOV)
      {
            if (scc.outputs.dLdVmeas >= MAX_VSUPERCAP)
            {
                  // mark that we have a supercapacitor overvoltage
condition
                  scc.outputs.bCapOV = YES;
            }
      }

      // turn off the current source if we are not charging,
communication has been lost, or if the
      // maximum supercapacitor voltage has been exceeded
      if (scc.ChgState != ST_CHARGING || scc.outputs.bCommLost ||
scc.outputs.bCapOV)
      {
            // we use a negative value to indicate that we want to
stop the 6269B current
            SetChargeCurrent(HP6269B_OFF);
            return;
      }
```

```
	// at this point, we know that we are charging the
supercapacitor and its maximum safe voltage
	// has not been exceeded

	// Adjust the commanded current
	switch(scc.inputs.ChgMode)
	{
	case CM_POWER:
		// When we are to charge the supercapacitor with a
constant power, we need to adjust
		// the charge current periodically depending upon the
current supercapacitor voltage.
		// ( I = P / V )
		if (scc.outputs.dLdVmeas <= 0.0)
			dI = MAX_CHARGE_CURRENT;
		else
		{
			dI = scc.inputs.dChgCmd / scc.outputs.dLdVmeas;
		}

		// set the charge current
		SetChargeCurrent(dI);
		break;

	case CM_CURRENT:
	default:
		// When we are to charge the supercapacitor with a
constant current, we simply pass
		// through the current from the s-function block input
pin.
		SetChargeCurrent(scc.inputs.dChgCmd);
		break;
	}

}



////////////////////////////////////////////////////////////////////
////////////////////////////////
//
// static void TransmitInitialization(void)
//
// Sends a command string to initialize the QuadTech 44020 and load
module
//
////////////////////////////////////////////////////////////////////
////////////////////////////////
static void TransmitInitialization(void)
{
	tx.uBufLen = sprintf( (char*) tx.Buf, "conf:rem off;:load
off;:fetc:volt?;curr?;stat?\n" );
```

```
	// send the command string
	SendTxBuf();
}



////////////////////////////////////////////////////////////////////
////////////////////////////////
//
// static void TransmitPoll(void)
//
// Send a command string to read the current voltage, current, and
status from the QuadTech 44020.
//
////////////////////////////////////////////////////////////////////
////////////////////////////////
static void TransmitPoll(void)
{
	tx.uBufLen = sprintf((char*) tx.Buf,
"fetc:volt?;curr?;stat?\n");

	// send the command string
	SendTxBuf();
}



////////////////////////////////////////////////////////////////////
////////////////////////////////
//
// static void TransmitLoadOff(boolean_T bRemoteOn)
//
// In:	boolean_T bRemoteOn - non-zero if are to keep the
QuadTech 44020 in remote mode.
//
////////////////////////////////////////////////////////////////////
////////////////////////////////
static void TransmitLoadOff(boolean_T bRemoteOn)
{
	if (bRemoteOn)
		tx.uBufLen = sprintf((char*) tx.Buf, "conf:rem on;:load
off;:fetc:volt?;curr?;stat?\n");
	else
		tx.uBufLen = sprintf((char*) tx.Buf, "conf:rem
off;:load off;:fetc:volt?;curr?;stat?\n");

	// send the command string
	SendTxBuf();
}
```

```
////////////////////////////////////////////////////////////////////
////////////////////////////////
//
// static void TransmitLoadCurrent(double dAmps)
//
// Sends the command string to turn on the QuadTech 44020 into a
constant current load of the
// specified current.
//
// In:      dAmps - amps to set the load to
//
////////////////////////////////////////////////////////////////////
////////////////////////////////
static void TransmitLoadCurrent(double dAmps)
{
      // Debug - Output an error message to control desk
      msg_printf(MSG_MC_INFO , MSG_DLG_NONE , MSG_SM_USER , 0,
"Constant current load [A]: %lf", dAmps);

      // build the command string
      if (dAmps <= MAX_ILOWRANGE)
      {
            // build the command string for the low current mode of
operation
            tx.uBufLen = sprintf( (char*)tx.Buf, "conf:rem on;:mode
ccl;:curr:stat:l1 %.6lf;:load on;:fetc:volt?;curr?;stat?\n", dAmps
);
      }
      else
      {
            // build the command string for the high current mode
of operation
            tx.uBufLen = sprintf( (char*)tx.Buf, "conf:rem on;:mode
cch;:curr:stat:l1 %.6lf;:load on;:fetc:volt?;curr?;stat?\n", dAmps
);
      }

      // send the command string
      SendTxBuf();
}


////////////////////////////////////////////////////////////////////
////////////////////////////////
//
// static void TransmitLoadPower(double dWatts)
//
// Sends the command string to turn on the QuadTech 44020 into a
constant power load of the
// specified power.
//
// In:      dWatts - desired power
```

```
//
////////////////////////////////////////////////////////////////////
////////////////////////////////
static void TransmitLoadPower(double dWatts)
{
	// Debug - Output an error message to control desk
	msg_printf(MSG_MC_INFO , MSG_DLG_NONE , MSG_SM_USER , 0,
"Constant power load [W]: %lf", dWatts);

	// build the command string
	if (dWatts <= MAX_PLOWRANGE)
	{
		// build the command string for the low power mode of
operation
		tx.uBufLen = sprintf((char*)tx.Buf, "conf:rem on;:mode
cpl;:pow:stat:l1 %.6lf;:load on;:fetc:volt?;curr?;stat?\n", dWatts
);
	}
	else
	{
		// build the command string for the high power mode of
operation
		tx.uBufLen = sprintf((char*)tx.Buf, "conf:rem on;:mode
cph;:pow:stat:l1 %.6lf;:load on;:fetc:volt?;curr?;stat?\n", dWatts
);
	}

	// send the command string
	SendTxBuf();
}



////////////////////////////////////////////////////////////////////
////////////////////////////////
//
// static void TransmitLoadResistance(double dOhms)
//
// Sends the command string to turn on the QuadTech 44020 into a
constant resistance load of the
// specified resistance.
//
// In:	dOhms - desired resistance
//
////////////////////////////////////////////////////////////////////
////////////////////////////////
static void TransmitLoadResistance(double dOhms)
{
	// Debug - Output an error message to control desk
	msg_printf(MSG_MC_INFO , MSG_DLG_NONE , MSG_SM_USER , 0,
"Constant resistance load [Ohms]: %lf", dOhms);

	// build the command string
```

```
	if (dOhms <= MAX_RLOWRANGE)
	{
		// build the command string for the low resistance mode
of operation
		tx.uBufLen = sprintf( (char*)tx.Buf, "conf:rem on;:mode
crl;:res:l1 %.6lf;:load on;:fetc:volt?;curr?;stat?\n", dOhms );
	}
	else
	{
		// build the command string for the high resistance
mode of operation
		tx.uBufLen = sprintf( (char*)tx.Buf, "conf:rem on;:mode
crh;:res:l1 %.6lf;:load on;:fetc:volt?;curr?;stat?\n", dOhms );
	}

	// send the command string
	SendTxBuf();
}



////////////////////////////////////////////////////////////////////
////////////////////////////////
// static void SendNextCommand(void)
//
// This function will send the next appropriate command string to
the QuadTech Programmable load
// depending on its current state of operation.
//
// On entry, the QuadTech programmable load is ready to receive a
command string (i.e. it has
// completed processing the last command string and returned the
appropriate response).
//
// This function is only executed when the communication state is
CS_POLLING.
//
////////////////////////////////////////////////////////////////////
////////////////////////////////
static void SendNextCommand(void)
{
	// If a change of charge state is not requested, then simply
poll the QuadTech for its
	// current voltage, current, and status

	if (scc.ChgState == scc.NewChgState)
	{
		// no charging state change requested, so just poll the
load
		TransmitPoll();
	}
	else
	{
```

```
            // a change of state has been requested, so send the
appropriate command string
            // to the QuadTech 44020

            // Transmit the appropriate command to change to the
new state
            // based on the desired state
            switch(scc.NewChgState)
            {
            case ST_DISCHARGING:
                  // we are to start the discharge state

                  // the command string will depend on the selected
discharge mode
                  switch(scc.inputs.DisMode)
                  {
                  case DM_CURRENT:
                        // we are to discharge the supercap with a
constant current
                        TransmitLoadCurrent(scc.inputs.dDisCmd);
                        break;

                  case DM_POWER:
                        // we are to discharge the supercap with a
constant power
                        TransmitLoadPower(scc.inputs.dDisCmd);
                        break;

                  case DM_RESISTANCE:
                        // we are to discharge the supercap with a
constant resistance
                        TransmitLoadResistance(scc.inputs.dDisCmd);
                        break;
                  } /* switch(scc.inputs.DisMode) */
                  break;

            // all other states simply turn off the programmable
load
            case ST_PAUSINGAFTERCHARGE:
            case ST_PAUSINGAFTERDISCHARGE:
            case ST_CHARGING:
                  // turn the load off, but keep it in remote mode
                  TransmitLoadOff(YES);
                  break;

            case ST_OFF:
                  // turn the load off and stop the remote mode
                  TransmitLoadOff(NO);
                  break;
            } /* switch(scc.NewChgState) */

            // we are now waiting for a response to the new state
command
```

```
            scc.LastTxChgState = scc.NewChgState;
      }
}


////////////////////////////////////////////////////////////////////
////////////////////////////////
//
// static void SerialCommTask(void)
//
// This function handles the serial communications with the
QuadTech 44020 programmable load.
//
// Since the various states of the programmable load are set by
sending commands to the RS232
// remote port on the programmable load, any state changes that
require sending commands to the
// load must be handled only after the response to the last
transmitted command has been
// received.
//
////////////////////////////////////////////////////////////////////
////////////////////////////////
static void SerialCommTask(void)
{
      int32_T     lBytes;
      int         ch;
      double      dV;
      double      dI;
      int   iStatus;
      char  *pV;
      char  *pI;
      char  *pStatus;
      char  debug[50];

      // don't execute if     the   serial port was   not   been
successfully opened
      if (!serCh)
            return;

      // wait until the command has been transmitted before
continuing
      if (scc.bTransmitting)
      {
            // exit if the command has not been completely sent
(i.e. bytes remaining in the tx fifo)
            if ( dsser_transmit_fifo_level(serCh) )
                  // command not completely sent yet, so return
                  return;

            // the command string has now been sent
            scc.bTransmitting = NO;
```

```
            // setup to wait for the response
            scc.dCommTime = ts_time_read() + RSP_TIMEOUT;
            rx.uBufLen = 0;
      }

      // process all charactters that are currently in the receive
buffer
      while ( (ch=RxChar()) != EOF )
      {
            // we have a character to process

            // check if this is the last character of the response
(all responses from the QuadTech
            // end in a newline character)
            if (ch=='\n')
            {
                  // we have now received a complete response to
our last command

                  // All command sequences that we send will result
in the same response format
                  // consisting of the current load voltage, load
current, and load status delimited
                  // by semicolons. (i.e.
"voltage;current;status\n")
                  // example: "2.3456;0.1501;16\n" = 2.3456V,
0.1501A, status of 16 (0x10)

                  // copy the response before we break it up with
strtok() for debugging
                  strncpy(debug, (char*)rx.Buf, sizeof(debug));
                  debug[sizeof(debug)-1] = 0;

                  // check if the response was too long
                  if (rx.uBufLen == sizeof(rx.Buf) )
                  {
                        // Output an error message to control desk
                        msg_printf(MSG_MC_INFO , MSG_DLG_NONE ,
MSG_SM_USER , 0, "QuadTech 44020 was too long: \"%s\"", debug);

                        // flush the response queue
                        FlushAllRxQ();

                        // resend the last command
                        SendTxBuf();

                        // and exit
                        return;
                  }


                  // tokenize the received data
                  pV = strtok((char*)rx.Buf, "; \t\n");
```

```
                  pI = strtok(NULL, "; \t\n");
                  pStatus = strtok(NULL, "; \t\n");

                  // make sure that we received all 3 tokens
                  if (pV==NULL || pI==NULL || pStatus==NULL )
                  {
                        // we didn't get all of the tokens that we
were expecting
                        msg_printf(MSG_MC_ERROR, MSG_DLG_OKCANCEL,
MSG_SM_USER, 0, "Invalid QuadTech 44020 response - \"%s\"", debug
);

                        // flush the response queue
                        FlushAllRxQ();

                        // and retransmit the last command
                        SendTxBuf();

                        // we sent something, so exit
                        return;
                  }
                  else
                  {
                        // we received three tokens, so process the
response

                        // convert the tokens to their equivalent
values and update the s-function
                        // block output holding registers
                        scc.outputs.dLdVmeas = strtod(pV, NULL);
                        scc.outputs.dLdImeas = -strtod(pI, NULL);
      /* load current is negative */
                        scc.outputs.ucLdStatus = (uint8_T)
strtoul(pStatus, NULL, 10);

                        // check if this received data was in
response to a command string to change the
                        // current charge state.
                        if (scc.ChgState != scc.LastTxChgState)
                        {
                              // this was a response to a command
string to change the charge state

                              // now we can update the state.
                              scc.ChgState = scc.LastTxChgState;
                        }

                        // if this is the first response after we
had a loss of communications or if this,
                        // is the very first response, start the
initialization state
                        if (scc.outputs.bCommLost || scc.CommState
== CS_DISCONNECTED)
```

```
                        {
                              // serial communications with the
QuadTech has been established or restored
                              scc.CommState = CS_INITIALIZING;

                              // switch to the off state
                              scc.LastTxChgState = scc.ChgState =
scc.NewChgState = ST_OFF;

                              // mark that communications is OK
                              scc.outputs.bCommLost = NO;

                              // send the initialization command
string
                              TransmitInitialization();

                              // and return
                              return;
                        }
                        else if (scc.CommState == CS_INITIALIZING)
                        {
                              // we have now initialized the
QuadTech 44020, so switch to polling
                              scc.CommState = CS_POLLING;
                        }

                        // only gets here if we have received a
response to our last command string and
                        // our current communicatoin state is now
CS_POLLING

                        // The QuadTech 44020 is now ready to
receive a command string, so send the
                        // next command.
                        SendNextCommand();
                  }

            } /* if (ch=='\n') */
            else
            {
                  // a character was received, but it is not the
end of the response

                  // add the character to the response being
constructed
                  if (rx.uBufLen < sizeof(rx.Buf))
                  {
                        rx.Buf[rx.uBufLen++] = ch;
                  }
            } /* if (ch!='\n') */
      } /* while ( (ch=RxChar()) != EOF ) */
```

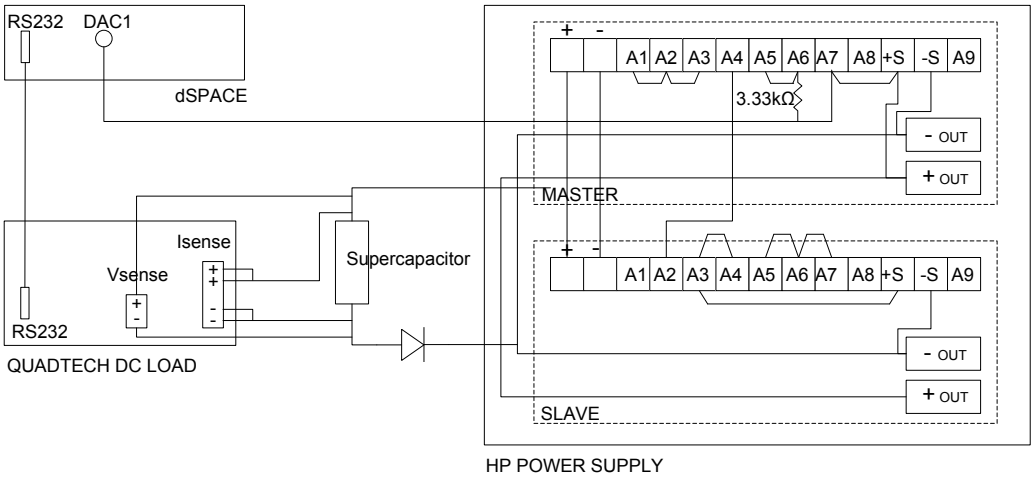
```
      // only gets here if we have processed all characters in the
receive buffer

      // timeout waiting for the response?
      if ( ts_time_read() >= scc.dCommTime )
      {
            // mark communication with the programmable load lost
            scc.outputs.bCommLost = YES;

            // since we have lost communication, change
communication state to disconnected
            scc.CommState = CS_DISCONNECTED;

            // ... and retransmit our last command
            SendTxBuf();
      }
}

////////////////////////////////////////////////////////////////////
////////////////////////////////
//
// void SccInit(void)
//
// Initializes the s-function block
//
////////////////////////////////////////////////////////////////////
////////////////////////////////
void SccInit(void)
{
      // only     execute     if we have not already been
      executed
      if (scc.bInitialized)
            return;

      // initialize the data structures
      memset(&scc, 0, sizeof(scc));

      /* initialize the serial port */
      serCh =     dsser_init(DSSER_ONBOARD, 0, SIZEOF_RXFIFO);
      // RS232 Serial   3

      /* the serial port has been   initialized, now configure the
port  */
      if (serCh)
      {
            // configure the port as as required by the QuadTech
44020 programmable load
            // ( RS232, 9600 BAUD, 8-bits, no Parity, and 1 stop
bit )
            dsser_config(
                  serCh,
                  /* point to the   serial channel structure */
```

```
                  DSSER_FIFO_MODE_BLOCKED,
      /* ignore new data if the receive fifo is full */
                  9600L,
                  /* 9600     BAUD */
                  8,
                  /* 8 data   bits */
                  DSSER_1_STOPBIT,
      /* 1 stop bit */
                  DSSER_NO_PARITY,
      /* no parity */
                  DSSER_8_BYTE_TRIGGER_LEVEL,
      /* 8 byte UART fifo     trigger     level */
                  DSSER_TRIGGER_LEVEL_DISABLE,
      /* no receive subinterrupt handling */
                  DSSER_RS232 | DSSER_AUTOFLOW_DISABLE );
      /* RS232 without flow control */
      }
      else
      {
            // couldn't initialize the serial port
            msg_printf(MSG_MC_ERROR, MSG_DLG_OKCANCEL, MSG_SM_USER,
0, "dsser_init() - could not initialize RS232 port");
      }

      // mark     that we     have now started the serial
      communications
      scc.bInitialized = YES;

      // set the initial state to initializing
      scc.ChgState = scc.LastTxChgState = scc.NewChgState = ST_OFF;

      // Initially we are logically disconnected from the
programmable load
      scc.CommState = CS_DISCONNECTED;

      // transmit a poll command to start the serial communications
      TransmitPoll();
}


////////////////////////////////////////////////////////////////////
//////////////////////////////
//
// This function is called every Simulink sample interval.  The
function is the main task
// for the s-function block.
//
////////////////////////////////////////////////////////////////////
//////////////////////////////
void  SccTask(void)
{
      // handle the serial communications to and from the QuadTech
44020 programmable load
```

```
	SerialCommTask();

	// handle the current command to the HP6269B power supply
used to charge the supercap
	ChargeCurrentTask();

	// handle switching the block to off
	if (!scc.inputs.bOn)
	{
		// set the new state to ST_OFF
		// (this will have no effect if the current state is
already ST_OFF)
		scc.NewChgState = ST_OFF;

		// reset the overvoltage condition
		scc.outputs.bCapOV = NO;
	}
	else if (scc.ChgState == scc.NewChgState)
	{
		// at this point, we know that there are no pending
charge state transitions and the block
		// On input is set

		// check for charge state transitions depending on the
current states
		switch(scc.ChgState)
		{
		case ST_OFF:
			// start charging now that we have switched to on
			scc.NewChgState = ST_CHARGING;

			// clear the cycle counter
			scc.outputs.ulCycle = 0;
			break;

		case ST_CHARGING:
			// check if we are done charging
			if (scc.outputs.dLdVmeas >= scc.inputs.dChgVth)
			{
				scc.NewChgState = ST_PAUSINGAFTERCHARGE;

				// set the delay timer
				scc.dPauseTime = ts_time_read() +
scc.inputs.dChgDelay;
			}
			break;

		case ST_DISCHARGING:
			// check if we are done discharging
			if (scc.outputs.dLdVmeas <= scc.inputs.dDisVth)
			{
				scc.NewChgState = ST_PAUSINGAFTERDISCHARGE;
```

```
                        // set the delay timer
                        scc.dPauseTime = ts_time_read() +
scc.inputs.dDisDelay;
                  }
                  break;

            case ST_PAUSINGAFTERCHARGE:
                  // check if we are done pausing
                  if (ts_time_read() >= scc.dPauseTime)
                  {
                        scc.NewChgState = ST_DISCHARGING;
                  }
                  break;

            case ST_PAUSINGAFTERDISCHARGE:
                  // check if we are done pausing
                  if (ts_time_read() >= scc.dPauseTime)
                  {
                        // we have completed a charge/discharge
cycle
                        scc.outputs.ulCycle++;

                        // check if it is time to stop
                        if (scc.inputs.ulCycles &&
scc.outputs.ulCycle >= scc.inputs.ulCycles)
                        {
                              // we have reached the required
number of cycles, so stop
                              scc.NewChgState = ST_OFF;

                              // this will stop further charge /
discharge cycles
                              scc.inputs.bOn = NO;
                        }
                        else
                              // start the charge cycles
                              scc.NewChgState = ST_CHARGING;
                  }
                  break;
            } /* switch(scc.ChgState) */
      } /* if (scc.ChgState == scc.NewChgState) */
}

#endif // #ifndef MATLAB_MEX_FILE
```

Appendix C: Test Bench 1 Hardware Configuration


RS232
DAC1
dSPACE
Isense
Vsense
Supercapacitor
RS232
QUADTECH DC LOAD
+ -
A1 A2 A3 A4 A5 A6 A7 A8 +S -S A9
3.33kΩ
- OUT
+ OUT
MASTER
+ -
A1 A2 A3 A4 A5 A6 A7 A8 +S -S A9
- OUT
+ OUT
SLAVE
HP POWER SUPPLY