Supercap-Python: An Open-Source Python Based Super Capacitor Modelling Package

Venkatesh Pampana, Daniel Lavin, and Markus Duchon
Fortiss GmbH, Guerickestr. 25, 80805 Munich, Germany
Email: {pampana, lavin, duchon}@fortiss.org

Ankit Srivastava
Madan Mohan Malaviya University of Technology, Gorakhpur, India
Email: aksee@mmmut.ac.in

***Abstract*—Supercapacitors have attained high power density and exceptional durability with the recent advancement in terms of their materials and chemistries. The potential scientific and industrial applications of supercapacitors are being explored continually. This instigates the need for model-based analysis and synthesis tools, which can describe dynamic phenomena, support multiphysics problems, and allow for immediate use in design and advanced control analysis. For these aspects, modelling of supercapacitors would be beneficial. However, there are no open-source simulation tools on supercapacitors available for the scientific community to work with. This paper presents the development of an open-source supercapacitor modelling package in python language. The proposed package is evaluated by comparing the results with a standard MATLAB/Simulink supercapacitor model. The simulation results have shown that both models yielded similar envelope.**

***Index Terms*—Modelling, Super Capacitors, python, package, simulation**

I. Introduction

A supercapacitor, also known as ultra-capacitor or Electric Double-Layer Capacitor (EDLC) [1], is a type of capacitor based on the electrical double-layer phenomenon in which charge is stored at the interface between a solid electrode material and the liquid electrolyte in the micro-pores of the electrodes. Super capacitors have exceptionally high energy density when compared to other electrical energy storage devices, such as conventional electrolytic capacitors or batteries, as well as higher charge and discharge rates. They have found applications in a number of industrial fields, for example [2]: pulsed-current applications in mobile phones and electrical actuators, peak-power providing in power electronic applications for laptops and electric vehicles, regenerative braking, disturbance reduction in power quality systems, and providing back-up power for devices in remote areas such as Photovoltaic or wind power systems. They are also used along with solar cells to power smart electronic wearables like watches, portable gadgets etc. [3], [4].

Many potential scientific and industrial applications of super capacitors are being explored continually. For example, a Fast In-line Water Heating System has been developed using patent pending super capacitor assisted temperature modification apparatus, where the research team has simulated the system models before building the prototype [5]. This highlights the importance of model-based analysis and synthesis tools, which can describe dynamic phenomena and allow for immediate use in design and advanced control analysis.

A few super capacitor simulation models are available in modelling and simulation tools like MATLAB, PSpice and COMSOL. However, these software tools require a lot of processing capabilities and cannot be used with low cost single board computers such as Raspberry Pi, Tinker Board, etc. Furthermore, these software tools are closed source, meaning, they require a proprietary license to run. Licensed developer can only do the development of libraries and if someday the company, which owns it, go out of business, the software will become obsolete. On the other hand, using an open source software is advantageous over a closed source one. In an open-source software, one can always look under the hood to see how things have been implemented and can develop on top of it. Therefore, even if the original developers of python do not develop the libraries for it, the other researchers and developers can contribute in library development process.

Python is an interpreted and object-oriented programming language that has gained widespread popularity among scientific and engineering community in recent decade [6]. Unlike the procedure-oriented programming, where the main emphasis is on functions, the object-oriented programming stresses on the objects. An Object is simply a collection of data variables and methods (or functions) that act on those data. While the class is a blueprint for the object. Python is a high-level programming language, which means it has a more powerful syntax and more useful set of data structures than low-level languages, for example, COBOL or C. In a high-level language, complex tasks can be achieved with relatively few lines of readable code compared to low-level languages [7].

Manuscript received October 14, 2021; revised December 14, 2021.

doi: 10.18178/ijeee.9.4.93-99

The first ever photograph of black hole was taken recently. The picture was created using data processing capabilities of Python and its packages such as NumPy, SciPy, and pandas [8]. This illustrates the role that the Python ecosystem plays in supporting the scientific advancement through collaborative data analysis. Paul Romer, the 2018 Nobel Laureate in Economics, wrote an interesting article "Jupyter, Mathematica, and the Future of the Research Paper", about his experience with open-source software tool like Jupyter Notebook [9]. By his estimation, employing an open source software like Python brings greater accountability and integrity to research, as code can be used by anyone who wishes to work and develop on top of it.

With a growing list of free software packages and an active community, makes Python ecosystem suitable for just about any type of scientific analysis. In contrast, MATLAB is closed and proprietary and even the algorithms are proprietary in some cases [10]. With the appropriate open source scientific packages like pandas, NumPy, SciPy, Matplotlib, etc., Python provides the functionality that is similar to MATLAB, R, and other numerical computing environments.

In recent decade, any research has multidisciplinary aspects. For example, incorporation of Machine learning techniques can be seen in the research works of nearly every discipline. Though the popularity and usage of Python has grown tremendously during recent years, there exists no python based packages for super capacitors. Hence, developing a Python based super capacitor library would be advantageous not just for scientific community but also for industrial applications. This would open up new application possibilities in future. Especially, when Python based electrical packages will become more versatile. This leads to performing various simulations with multidisciplinary tools very conveniently in Python and the hardware platforms that support this language.

The approach presented in this paper is the implantation of SuperCap-Python package, an open-source toolset for super capacitor modelling and simulation consisting of Python packages written by the authors. The paper is organized as follows; Section II presents the literature review; super capacitor technical details and calculations is discussed in Section III; Section IV describes the key implementation aspects of proposed modelling package; a usage example is provided in Section V and lastly, the paper is concluded with summary of work and future directions.

II. Motivation

Recently, several projects have commenced that harness the power and flexibility of modelling and computational tools, especially in electrical and energy domain. For example, "PV_LIB" Python package could help in modelling of Photovoltaic systems [11]. It is a Python ported version of the original PVLIB MATLAB toolbox to perform advanced data analysis and research into the performance modelling and operations of PV assets. Similarly, an open-source project called Python for Power System Analysis (PyPSA) was developed in Germany [12]. PyPSA is a free software python-based toolbox for simulating and optimizing modern electrical power systems over multiple periods.

Researchers from the United States Geological Survey have proposed an alternative approach for developing, running, and post- processing groundwater models [7]. They have developed Python scripts using FloPy package to process input files, read and plot simulation results for MODFLOW-based models. In a study on "Ultracapacitors Electrobus for Urban Transport", the authors investigate application of super capacitors in an electric bus using Matlab Simulink [13]. The proposed SuperCap-Python package in this paper could potentially be used to simulate and validate similar usecases and applications.

Supercapacitor modelling and simulation often entail the characterization of the physical device, which is done by proposing an equivalent circuit that behaves in the same way a supercapacitor would in practice. Indeed, once the circuit is proposed, a supercapacitor is tested using equipment such as potentiostats or Electrochemical Impedance Spectrometers (EIS), for empirically estimating the values of the resistive, capacitive and inductive elements of the circuit. Afterwards, the circuit is simulated using software, and the results are compared against the experimental measurements to assess its fidelity. In [14], a Maxwell Boostcap 3000F supercapacitor was subjected to EIS testing and the characterized circuit was implemented in MATLAB/Simulink. In [15], a Scribner potentiostat was used to characterize a supercapacitor. While experimental results were compared against simulations, the software employed was not mentioned. In [16] an experimental testbed was used to characterize a PScap350 and the equivalent circuit was implemented in PSpice. The components of the testbed are however not mentioned.

The researchers in [17] had proposed a control method on wind turbine that could maximize the harvested energy from a wind turbine using ultra-capacitor energy storage in the microgrid condition. The software models developed in such ways could be integrated into the simulations of more complex systems. In [18], a supercapacitor model was utilized to simulate the storage of energy in a regenerative braking system in MATLAB/Simulink. Even though the paper focuses more on the electrical parameters of the DC motor of the vehicle, it serves to demonstrate one of several potential applications of software tools for supercapacitor simulation.

Nonetheless, a few problems are evident in the implementations presented above:

- The models are not open for contribution and/or are not openly distributable.
- The models are developed using software tools that require licenses that can be expensive or unavailable altogether.
- Beyond the papers in which they are presented, no other documentation for the implementation is available.

Therefore, an alternative approach for supercapacitor modelling using python package toolset called "SuperCap-Python" is proposed in this paper.

SuperCap-Python is released as free software under the MIT License [19]. This means that the user is free to inspect, use and modify the code, and also permitted to put the code in proprietary software on the condition that the license is given with that software. The library will be further expanded by involving the scientific community and academia through collaboration on Github repository.

III. Design and Modelling

A few equivalent circuit models have been proposed in [14], [20], [21]. However, the model presented in this paper is chosen due to its simplistic nature, ability to represent the chemical or electrical characteristics of super capacitors and considering their application in medium or large-scale energy storage projects from an electrical point of view [22], [23]. The super capacitor's equivalent circuit model that is considered in this paper is presented in Fig. 1.

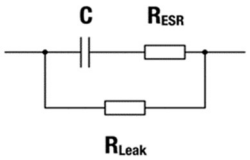

Figure 1. Supercapacitor representation circuit.

The elements of the circuit are explained as follows:

- The capacitance C represents the total capacitance related to the charge capacity of the supercapacitor.
- The resistance R_{ESR} is known as the equivalent series resistance (esr) and is associated with the energy losses and with the maximum power output of the supercapacitor.
- The parallel resistance R_{LEAK} is associated with the self-discharge behaviour of the supercapacitor. It is mainly represented by the leak current I_{LEAK} found normally in supercapacitor datasheets.

The following basic parameters are considered in the model, based on equivalent circuit mentioned above:

- The maximum or rated voltage of the supercapacitor, V_R
- The maximum or rated current that can flow in or out of the supercapacitor, I_R
- The capacitor's initial voltage, V_0
- A minimum voltage, V_{MIN}, can optionally be defined for the supercapacitor. If it is not set, a value of zero shall be assumed.

The model calculates the maximum charge and discharge powers of the super capacitor, P_{CH_MAX} and P_{DCH_MAX}. These are used for further computation of charge and discharge cycles of super capacitor operation, and are given through the following equations:

$$P_{CH\_MAX} = V_R \times I_R \tag{1}$$

$$P_{DCH\_MAX} = \frac{V_R^2}{4 \times R_{ESR}} \tag{2}$$

The equations presented in this paper are derived based on previous studies on super capacitor modelling [15], [24] and technical guides from a couple of super capacitors manufacturers (Wuerth Electronik and Tecategroup) [23], [25].

A. Supercapacitor Charging

Super capacitors can be charged using three different methods:

1. Constant power
2. Constant current
3. Constant voltage

The equations describing the behavior of the voltage are given in the below.

1) Constant power mode:

$$V_1 = \sqrt{V_0^2 + \frac{2 \times P_{CH} \times \Delta t}{C}} \tag{3}$$

where:

- V_0, V_1 are the initial and final voltages
- P_{CH} is the charging power
- Δ t is the duration of the timestep in seconds

2) Constant current mode:

$$V_1 = V_0 + \frac{I_{CH} \times \Delta t}{C} \tag{4}$$

where:

- V_0, V_1 are the initial and final voltages
- I_{CH} is the charging current
- Δt is the duration of the timestep in seconds

B. Supercapacitor Discharging

Super capacitors can be discharged using two different methods:

1. Constant power
2. Constant current

The model implements both methods. The equations describing the behaviour of the voltage are given below.

1) Constant power mode:

$$V_1 = \sqrt{(V_0^2 - (\frac{2 \times P_{DCH} \times \Delta t}{C})} \tag{5}$$

where:

- V_0, V_1 are the initial and final voltages
- P_{DCH} is the discharging power
- Δ t is the duration of the timestep in seconds

2) Constant current mode:

$$V_1 = V_O - \frac{I_{DCH} \times \Delta t}{C} \tag{6}$$

where:

- V_0, V_1 are the initial and final voltages
- I_{DCH} is the discharging current
- Δ t is the duration of the timestep in seconds

C. Self-Discharge of the Supercapacitor

The self-discharge behaviour is entirely defined by the self-discharge current, and is simulated as a constant

current discharge process, where $I_{DCH} = I_{LEAK}$. The same considerations are taken here as in the discharge process.

IV. Implementation

This section explains the implementation approach and inner workings of SuperCap-Python, the super capacitor modelling python package, based on information presented in previous section. SuperCap-Python is written in the Python programming language and followed the conventional programming patterns and guidelines set by the python community. The model equations are written in the python code using Dictionary objects, constructor methods, functions and dataframes in object oriented manner. This coding pattern would increase the modularity and makes it easy to update the existing model or introduce new super capacitor models into the modelling package.

The most important methods in SuperCap-Python are explained below:

- Charge_ctP(pow) – charges the supercapacitor at constant power, "pow", for the duration of one timestep
- Charge_ctI(cur) - charges the supercapacitor at constant current, "cur", for the duration of one timestep
- Discharge_ctP(pow) and Discharge_ctI(cur) – discharge counterparts of the previous two functions. These will first verify that there is enough energy left in the supercapacitor to provide the requested power, if not, the functions will return the actual value at which the supercapacitor discharged.
- Selfdischarge() - discharges the super capacitor at constant leak current for the duration of one timestep.

Beside the equations that are presented in previous section, following additional considerations are taken in order to simulate a realistic charging and discharging process of the model.

- For both charging and discharging methods, the operating current is calculated to ensure that it does not exceed the rated current. If this is the case, then the super capacitor charges at either maximum power or maximum current.
- After the charging process is calculated for the provided duration of the timestep, the resulting final voltage of the super capacitor is checked to ensure it does not exceed the rated value V_R. If this is the case, then the voltage of the super capacitor is set to its rated value.
- While the super capacitor is switching from one mode to the other, a sudden voltage drop or spike will occur, i.e discharging, self-discharging or fully discharged before it is set to charging mode and vice versa. This behavior is modelled using the voltage across the equivalent series resistance with equation: $\Delta V = I_{CH} \times R_{ESR}$

The current version of python package implements the constant power and constant current modes. The constant voltage mode is not implemented due to time constraints and is planned for implementation in future releases of the package. SuperCap-Python is available online in the Python Package Index (PyPI), on GitHub [26]. Additionally, it is archived on Zenodo [27]. PyPI is a public repository of software packages for the Python programming language. The Documentation and examples are available on repository website [26]. The model package can be downloaded and imported into any computer using "pip install supercap" command. It has been tested with 3.5 version of Python. This package is compatible with the Open Energy Modelling Framework (oemof), a popular open source modelling python package [28].

V. Results and Evaluation

A. Simulation Results with SuperCap-Python Package

The charging and discharging behavior of super capacitors is simulated using the proposed python-based model. The parameters of chosen super capacitor is presented in Table I.

TABLE I. Parameters of Supercapacitor Used for Simulation

Parameter	Value	Unit
Capacitance	300	farad
Rated/Maximum current	30	Ampere
Rated voltage	70	Volt
Initial voltage	0	Volt
ESR	1.65	Ohm
Leak current	1500x10-4	Ampere

The model is subjected to various charging and discharging conditions as given below for the simulation:

Step-1: Charge at 100 W for 1000 seconds.

Step-2: Rest (self-discharge) for 300 seconds.

Step-3: Discharge at 100 W for 1200 seconds.

Step-4: Charge in constant current mode at 2A for 800 seconds.

Step-5: Rest (self-discharge) for 200 seconds.

Step-6: Discharge at 3A for 1000 seconds.

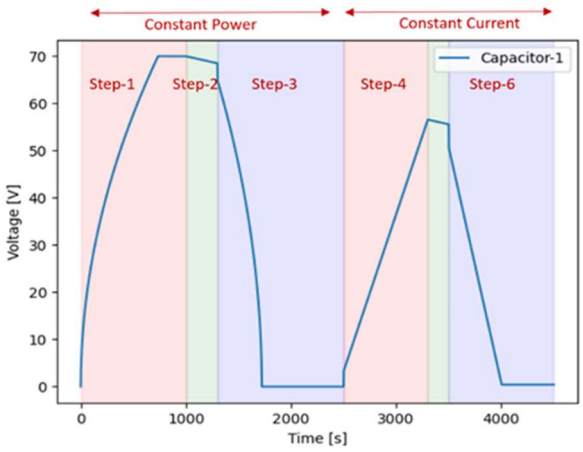

Figure 2. Voltage vs Time simulation output for charging and discharging of super capacitor.

The voltage time evolution at the super capacitor's terminals is plotted in Fig. 2. The aforementioned steps have been marked in the figure for better representation. Charging process is represented in red color, self-discharging in green and discharging in violet.

When charging at 100 watts, the supercapacitor became fully charged at the 723 second mark. During rest period of 300 seconds, it self-discharged to 68.5 volts. Under constant power discharging of 150 watts, it becomes fully discharged at around 1721 seconds. Similarly, when it charged at 2A, after 800 seconds, the voltage reaches to 57 volts. Finally, when it is discharged at constant current of 3A, the complete discharge happened at 4000-second mark.

B. Test Simulation Results Comparison with MATLAB/Simulink

Matlab/Simulink provides a supercapacitor block in Simscape Electrical library. The charging, discharging and self-discharging behavior of developed python model is simulated using the supercapacitor block in Simulink for Supercapacitor with same parameters as mentioned in Table I.

1) Charging mode simulation:

The charging of Supercapacitor is simulated with in constant current charging at 2A for 250 seconds in both MATLAB and Python. The results for both of them are plotted in Fig. 3. From the plot, it can be observed that both MATLAB and Python model presented similar charging curves with very minute variation in voltage values in the order of 0.1 volts.

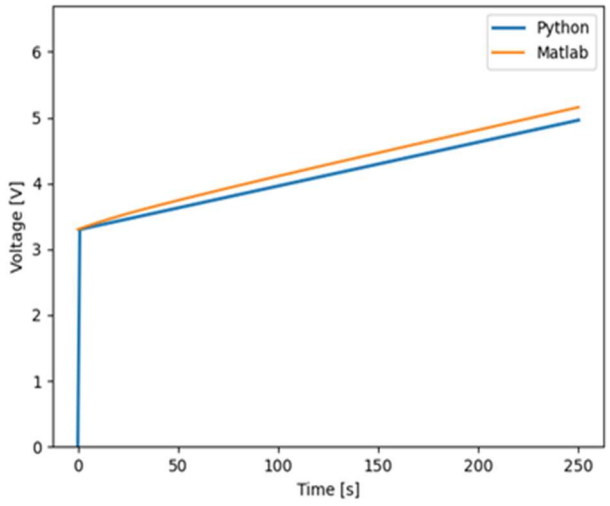

Figure 3. Charging mode simulation results comparison between MATLAB and Python.

2) Discharging mode simulation:

Supercapacitor is discharged at 3A under constant current for 200 seconds with initial voltage of 15V. Both simulations yielded very similar results with no significant variation and the discharging curves are almost overlapped. The comparison of results is presented in Fig. 4.

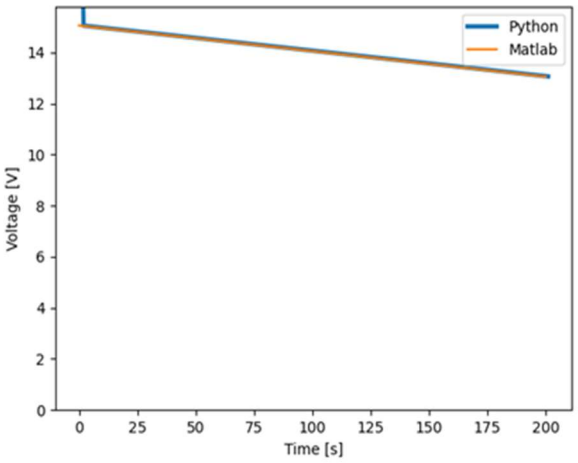

Figure 4. Discharging mode simulation results comparison between MATLAB and Python.

3) Self-discharging mode simulation

The self-discharging behaviour at 10V is simulated and shown in Fig. 5. Both the MATLAB and Python yielded same values with overlapping curves.

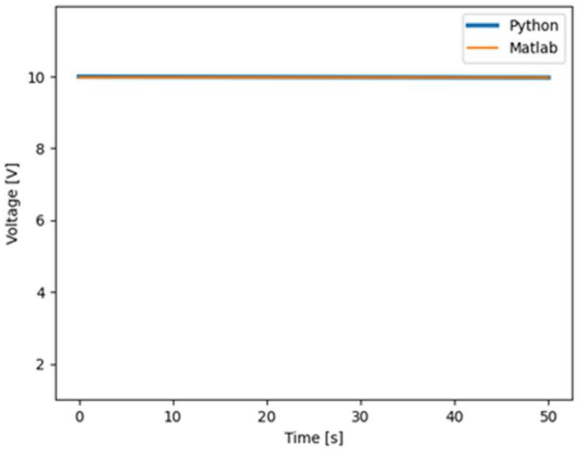

Figure 5. Self-discharging mode simulation results comparison between MATLAB and Python.

VI. CONCLUSION

In this paper a novel open source software package has been presented for simulating and modelling of the supercapacitor. SuperCap-Python can be used to perform high-fidelity simulations of charge, discharge and self-discharge processes of super capacitors as well as other supercapacitor related modelling and simulation projects. As it is Python based package, it is easily distributable, and due to its simplicity and absence of third party package-specific datatypes, it can be readily integrated into other projects to provide quick results. The implemented package is demonstrated using a usecase example and validated with MATLAB/Simulink model. As an open source library, the code of SuperCap-Python can easily be inspected and extended by contributors or users, thereby contributing to further research and transparency in super capacitor modelling.

As a future work, the model itself can be further improved by optimizing the charge and discharge functions without compromising reliability. Additionally, other charge and discharge methods can be further implemented, such as charging at constant voltage. Even more, some other equivalent circuits can be implemented into the model, and the usage of one or another could be set as a variable in the model. Finally, more practical and specific use cases could be exemplified, e.g. integrating the supercapacitor model with a power-feeding simulation tool, and adding optimization scenarios as well as embedding financial calculations for feasibility in including super capacitors for specific applications.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Venkatesh and Daniel developed the python package. Ankit did some experiments for evaluation of the developed model and Dr. Markus has put forward some innovative points.

ACKNOWLEDGMENT

This work is being carried out for on-going research project called ECO-WET, under the flagship of IGSTC (Indo-German Science and Technology Centre). The Authors would like to thank Federal Ministry of Education and Research (BMBF, Germany) and Department of Science and Technology (DST, India) for funding the research and development activities of the project.

REFERENCES

[1] A. Burke, "Ultracapacitors: Why, how, and where is the technology," *J. Power Sources*, vol. 91, no. 1, pp. 37-50, 2000.
[2] A. Namisnyk and J. Zhu, "A survey of electrochemical super-capacitor technology," presented at Australian Universities Power Engineering Conference, University of Canterbury, New Zealand, 2003.
[3] J. Varma, K. S. Kumar, S. Seal, S. Rajaraman, and J. Thomas, "Fiber-Type solar cells, nanogenerators, batteries, and supercapacitors for wearable applications," *Advanced Science*, vol. 5, no. 9, p. 1800340, 2018.
[4] P. Du, X. Hu, C. Yi, H. C. Liu, P. Liu, H. L. Zhang, and X. Gong "Self-powered electronics by integration of flexible solid-state graphene-based supercapacitors with high performance perovskite hybrid solar cells," *Advanced Functional Materials*, vol. 25, no. 16, pp. 2420-2427, 2015.
[5] N. Gurusinghe, N. Kularatna, S. A. Charleston, and J. Fernando, "System implementation aspects of supercapacitor based fast in-line water heating system," in *Proc. 2015 IEEE 24th International Symposium on Industrial Electronics (ISIE)*, Buzios, 2015, pp. 1313-1317.
[6] F. Perez, B. E. Granger, and J. D. Hunter, "Python: An ecosystem for scientific computing," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 13-21, 2010.
[7] M. Bakker, *et al.*, "Scripting MODFLOW model development using Python and FloPy," *Groundwater*, vol. 54, no. 5, pp. 733-739, 2016.
[8] Numpy.org. Case study: The first image of a black hole. [Online]. Available: https://numpy.org/case-studies/blackhole-image/
[9] P. Romer. *Jupyter, Mathematica, and the Future of the Research Paper*. [Online]. Available: https://paulromer.net/jupyter-mathematica-and-the-future-of-the-research-paper/ (accessed Oct. 22, 2020)
[10] M. Feldman. *Eight Advantages of Python over Matlab*. [Online]. Available: http://phillipmfeldman.org/Python/Advantages_of_Python_Over_Matlab.html
[11] R. W. Andrews, J. S. Stein, C. Hansen, and D. Riley, "Introduction to the open source PV LIB for python photovoltaic system modelling package," in *Proc. 2014 IEEE 40th Photovoltaic Specialist Conference (PVSC)*, 2014, pp. 170-174.
[12] T. Brown, J. Hörsch, and D. Schlachtberger, "PyPSA: Python for power system analysis," *arXiv Prepr.*, arXiv1707.09913, 2017.
[13] A. Hnatov, S. Arhun, K. Tarasov, H. Hnatova, V. Mygal, and A. Patlins, "Researching the model of electric propulsion system for bus using Matlab Simulink," in *Proc. IEEE 60th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON)*, 2019, pp. 1-6.
[14] L. Gauchia, S. Castaño, and J. Sanz, "New approach to supercapacitor testing and dynamic modelling," in *Proc. 2010 IEEE Vehicle Power and Propulsion Conference*, 2010, pp. 1-5.
[15] S. D. Fenol, F. S. Caluyo, and J. L. Lorenzo, "Simulation and modeling of charging and discharging of supercapacitors," in *Proc. 2017 International Conference on Circuits, System and Simulation (ICCSS)*, 2017, pp. 14-17.
[16] I. Ciocan, C. Farcăş, A. Grama, and A. Tulbure, "An improved method for the electrical parameters identification of a simplified PSpice supercapacitor model," in *Proc. 2016 IEEE 22nd International Symposium for Design and Technology in Electronic Packaging (SIITME)*, 2016, pp. 171-174.
[17] A. Aranizadeh, A. Zaboli, O. A. Gashteroodkhani, and B. Vahidi, "Wind turbine and ultra-capacitor harvested energy increasing in microgrid using wind speed forecasting," *Eng. Sci. Technol. an Int. J.*, vol. 22, no. 5, pp. 1161-1167, 2019.
[18] A. Adib and R. Dhaouadi, "Modeling and analysis of a regenerative braking system with a battery-supercapacitor energy storage," in *Proc. 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, 2017, pp. 1-6.
[19] Tldrlegal. *MIT License (Expat) Explained in Plain English-TLDRLegal*. [Online]. Available: https://tldrlegal.com/license/mit-license
[20] Z. M. Salameh, "Modeling, evaluation and simulation of a supercapacitor module for energy storage application," in *Proc. International Conference on Computer Information Systems and Industrial Applications*, Atlantis Press, 2015, pp. 876-882.
[21] N. Kularatna, D. A. Steyn-Ross, and K. Milani, "A designer's view on non-traditional supercapacitor techniques for sustainable energy applications," in *Proc. IEEE 25th International Symposium on Industrial Electronics (ISIE)*, 2016, pp. 962-967.
[22] B. E. Conway, *Electrochemical Supercapacitors, Scientific Fundamentals and Technological Applications*, New York: Kluwer Academic/Plenum Publishers, 1999.
[23] R. Kalbitz and F. Puhane. Supercapacitor—A guide for the design-in process. Wuerth Elektronik. [Online]. Available: https://www.we-online.de/web/en/index.php/download/media/07_electronic_components/download_center_1/application_notes_berichte/anp077_/ANP077a_EN.pdf
[24] N. Kularatna, *Energy Storage Devices for Electronic Systems: Rechargeable Batteries and Supercapacitors*, Academic Press, 2014.
[25] Tecategroup. *Ultracapacitor Technical Guide*. [Online]. Available: www.tecategroup.com/downloads/Ultracap_Tech/Ultracapacitor Technical Guide.pdf
[26] D. Lavin and V. Pampana. *Supercapacitor Modelling package for Python (SuperCap-Python)*. [Online]. Available: https://github.com/venkat0249/SuperCap-Python
[27] V. Pampana. venkat0249/SuperCap-Python: zendo_release2. [Online]. Available: https://pypi.org/project/supercap/
[28] H. Simon, *et al.*, "The Open Energy Modelling Framework (oemof)-A new approach to facilitate open science in energy system modelling," *Energy Strategy Reviews*, vol. 22, pp. 16-25, 2018.

Venkatesh Pampana received his B.S. degree in Electrical and Electronics Engineering from Jawaharlal Nehru Technological University, Kakinada, Republic of India in 2012 and an M.S. Degree in Renewable Energy Management from the Technical University of Cologne, Germany. Besides scientific research, he is an enthusiastic Product Developer, and a certified Product Owner. His research interests include renewable energy, electric vehicles, energy modelling, Internet of Things, energy storage, remote sensing and hardware design.

Ankit Srivastava is currently persuing his Ph.D from Madan Mohan Malaviya University of Technology, Gorakhpur, India. He also did his M-Tech from the same University in area of Power Electronic and Drives. His area of research includes Power Quality, Harmonic, and Interharmonics.