A High Performance Simulation Framework for Battery Modular Multilevel Management Converter

Dominic Karnehm
Universität der Bundeswehr München
Neubiberg, Germany
dominic.karnehm@unibw.de

Nina Sorokina
Universität der Bundeswehr München
Neubiberg, Germany
nina.sorokina@unibw.de

Sebastian Pohlmann
Universität der Bundeswehr München
Neubiberg, Germany
sebastian.pohlmann@unibw.de

Ali Mashayekh
Universität der Bundeswehr München
Neubiberg, Germany
ali.mashayekh@unibw.de

Manuel Kuder
Universität der Bundeswehr München
Neubiberg, Germany
manuel.kuder@unibw.de

Antje Gieraths
Universität der Bundeswehr München
Neubiberg, Germany
antje.gieraths@unibw.de

***Abstract*—This paper introduces a model for a one-phase Battery Modular Multilevel Management (BM3) Converter System described by a system of differential equations . This allows to simulate BM3 Systems in a performant manner. The model is implemented in MATLAB and multiple Python extensions. Comparing the runtimes of the simulation on a central processing unit (CPU) and a graphics processing unit (GPU) cores has shown that the CPU implementation currently exceeds the GPU implementation performance-wise. The highest performance was achieved by using the Python compiler Numba. This allows to simulate 1 s uptime of a BM3 Converter with 10 modules in 0.33 s – a reduction by more than factor 130 compared to an existing Matlab/SIMULINK model [1].**

***Index Terms*—Battery Management Systems (BMS), Modular multilevel converter, Simulation, GPU, Matlab, Python**

I. Introduction

One main trend of the last decade in the automotive industry is the constant growth in demand for electric vehicles. Germany alone, with its coalition agreement, has set a target of at least 15 million electric cars on its roads by 2030 [2]. Another observed trend, directly related to the first one, is the increase of operating voltages of the battery-powered electric vehicles (BEVs) [3]. Also, more and more charging systems enable higher voltages [4]. This progression requires a change of the semiconductors or the inverter topology, and one of possible solutions for this is the usage of multilevel inverter topologies [3].

For the simulation of multilevel inverter topologies, mainly tools of Matlab/SIMULINK are used [5]–[7]. This type of simulation reaches its limits when, for example, driving cycles of electric vehicles are simulated. This is due to the high memory requirements of these simulations, the low possibilities of parallelization on central processing units (CPUs) and graphic processing units (GPUs), and the insufficient possibilities of optimization in the calculations with comparable accuracy. To make simulations like this possible and reduce computational effort, lookup tables are often used [7], [8]. However, this reduces the accuracy of such calculations.

On the other side, for proper performance, modular multilevel converter systems need a state-of-charge (SOC) balancing method for battery cells, i.e. based on an algorithmic method, which was introduced in [1]. To apply machine learning technologies to the whole system, a performance, and accuracy optimized model is necessary.

A multilevel system consists of numerous nodes or submodules, which makes it highly suitable for parallelization through GPU cores [9]. In this paper, a system of differential equations is introduced to model a multilevel inverter (Section II). Section III lists the different implementations, in section IV the simulation results are shown. The runtime of the different implementations are discussed in section V and section VI finishes with an outlook to further research VI.

This research is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr (Project MORE and Project ELAPSED), which we gratefully acknowledge.

II. Modular Multilevel Converter Model

In this article, the Battery Modular Multilevel Management (BM3) [10] is considered as a multilevel converter. Its fundamental difference from the classic two-level system commonly used in automotive applications today is the configuration of the battery pack. In a two-level system, up to several thousand batteries are connected in a fixed way and the battery pack always has a voltage of the same value. A half-bridge for each motor phase, which is controlled by a pulse-width modulation (PWM), is used to produce a necessary sinusoidal output voltage.

Multilevel inverters have configurable battery packs, which allow for the battery cells to be connected in parallel, in series or to be bypassed to produce an output voltage of the desired level. This is possible because the entire battery pack is divided into submodules, each controlled by multiple semiconductors. In the case of the BM3, each submodule has three MOSFET switches [10].

A. BM3 Converter System

Fig. 1 shows a converter system (in red) with 3 modules (in green). The number of modules in the system is represented by N, in Fig. 1 therefore $N = 3$. The behavior of an electric motor (blue) is modeled by the ohmic resistance R_{AC}, the electromagnetic coil L_{AC} and an ideal voltage source u_{AC}.

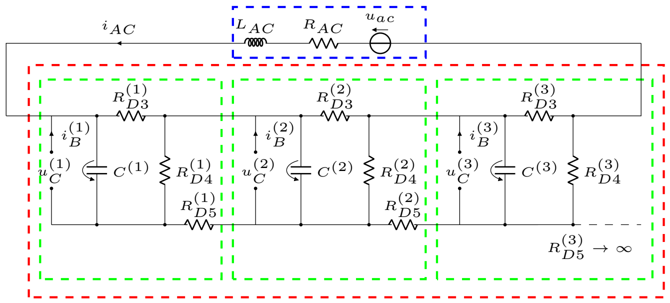


Fig. 1. Single-phase strand of three cascaded BM3 modules.

A BM3 module k consists of a battery, a capacitor, and three MOSFETs. Each module can be switched individually into three possible states: series, parallel, and bypass. These three states are described in Table I. The dynamic behaviour of the MOSFETs is not modeled, they are instead described as the resistors $R_{D3}^{(k)}$, $R_{D4}^{(k)}$ and $R_{D5}^{(k)}$. The two states of the switches are defined as follows: on-state with a fix resistance of $0.55\,\mathrm{m}\Omega$ and off-state as an infinite resistance.

TABLE I
SWITCH STATES OF MOSFETS, MODELED AS ELECTRICAL RESISTORS

	$R_{D3}^{(k)}$	$R_{D4}^{(k)}$	$R_{D5}^{(k)}$
Bypass	on	off	off
Parallel	on	off	on
Series	off	on	off

A system with N modules can be described by the state vector $\vec{x}$ as:

$$\vec{x} = \left(i_{AC} \left(SOC^{(1)} \;\; i_B^{(1)} \;\; u_C^{(1)}\right) \dots \left(SOC^{(N)} \;\; i_B^{(N)} \;\; u_C^{(N)}\right)\right)^T \tag{1}$$

where i_{AC} is the current of the converter system , $i_B^{(k)}$ is a battery current, $u_C^{(k)}$ is a capacitor voltage and $SOC^{(k)}$ represents the state-of-charge of battery cell in module k.

The nonlinear system behavior given in Eq. (1) can be approximated by numerical integration by the forward Euler method in time steps of equal duration. The current state $\vec{x}$ at t is considered to be known, and the state at the time step $t + \Delta t$ is to be determined.

The values of i_{AC} and $u_C^{(k)}$ at $t + \Delta t$ are described by the following algebraic system of $N + 1$ equations:

$$\begin{pmatrix} u_C^{(1)}(t+\Delta t) \\ u_C^{(2)}(t+\Delta t) \\ \vdots \\ u_C^{(N)}(t+\Delta t) \\ \hline i_{AC}(t+\Delta t) \end{pmatrix} = \mathbf{A}^{-1}\,\vec{v} \tag{2}$$

$$\overbrace{\left(\begin{array}{cccccc|c} a_{1,1} & a_{1,2} & 0 & \dots & & 0 & a_{1,N+1} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & & 0 & a_{2,N+1} \\ \vdots & \ddots & \ddots & \dots & & \vdots & \vdots \\ 0 & \dots & 0 & a_{N,N-1} & & a_{N,N} & a_{N,N+1} \\ \hline a_{N+1,1} & \dots & \dots & a_{N+1,N-1} & & a_{N+1,N} & a_{N+1,N+1} \end{array}\right)}^{\mathbf{A}} \tag{3}$$

The elements of matrix $\mathbf{A}$ are defined as:

$$\begin{aligned} a_{k,k-1} &= -\frac{\Delta t}{R_{D3}^{(k-1)} C^{(k-1)}} \frac{G_{D5}^{(k-1)}}{G^{(k-1)}} \\ a_{k,k} &= 1 + \frac{\Delta t}{R_{D3}^{(k)} C^{(k)}} \frac{G_{D4}^{(k)} + G_{D5}^{(k)}}{G^{(k)}} + \\ &\quad \underbrace{\frac{\Delta t}{C^{(k)} R_{D5}^{(k-1)}} \frac{G_{D3}^{(k-1)} + G_{D4}^{(k-1)}}{G^{(k-1)}}}_{\text{if } k \leq 1 \text{ then } 0} \\ a_{k,k+1} &= -\frac{\Delta t}{R_{D3}^{(k)} C^{(k)}} \frac{G_{D5}^{(k)}}{G^{(k)}} \\ a_{k,N+1} &= \frac{\Delta t}{C^{(k)}} \frac{G_{D5}^{(k)}}{G^{(k)}} \\ a_{N+1,k} &= \frac{\Delta t}{L_{AC}} \frac{G_{D5}^{(k)}}{G^{(k)}} \\ a_{N+1,N+1} &= 1 + \frac{\Delta t}{L_{AC}} \left(R_{AC} + \sum_{k=1}^{N} \frac{1}{G^{(k)}}\right) \end{aligned} \tag{4}$$

The electrical conductance is defined as the multiplicative inverse of the electrical resistance, represented by G, and measured in siemens. $G_{D3/4/5}^{(k)}$, and $G^{(k)}$ are dependent on the switch state of the module k:

$$\begin{aligned} G_{D3/4/5}^{(k)} &= \frac{1}{R_{D3/4/5}^{(k)}} \\ G^{(k)} &= \left(G_{D3}^{(k)} + G_{D4}^{(k)} + G_{D5}^{(k)}\right) \end{aligned} \tag{5}$$

The vector $\vec{v}$ is defined as:

$$\vec{v} = \left(v_1 \quad v_2 \quad \dots \quad v_N \quad \middle| \quad v_{N+1}\right)^T \tag{6}$$

$$\begin{aligned} v_k &= u_C^{(k)}(t) + \frac{\Delta t}{C^{(k)}} \left(i_{AC}(t) \frac{G_{D4}^{(k)} + G_{D5}^{(k)}}{G^{(k)}} + i_B^{(k)}(t)\right) \\ v_{N+1} &= i_{AC}(t) + \frac{\Delta t}{L_{AC}} \left(\sum_{k=1}^{N-1} u_C^{(k+1)}(t) \frac{G_{D5}^{(k)}}{G^{(k)}} - u_{AC}(t)\right) \end{aligned} \tag{7}$$

B. Battery simulation

There are several ways to simulate the characteristics of the cell dynamics. It can be simulated by electro-thermal [11], electrochemical [12] or equivalent circuit models [13]. In this paper, a third order Thévenin model is used, see Fig. 2. The Thévenin model enables prediction of the battery behavior with low error under dynamic and non-dynamic current profiles [14]. It contains a SOC related voltage source, an internal ohmic resistance R_0 and three RC-pairs [1].

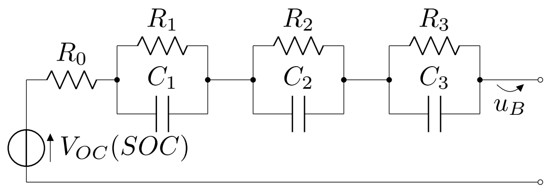


Fig. 2. Third-order Thévenin model of a battery cell [13]

In [15], the equation of the open-circuit voltage and the discharge voltage with the 2-RC model is introduced. This can be adapted to the 3-RC model. The modeling of the battery voltage is done within the inverter model to represent the influences of the individual models on each other. Here, only the current of the battery is modeled.

In [15], the discharging process is considered as a zero state response process and the terminal voltage u_B is described as:

$$u_B = V_{OC}(SOC) - i_B R_1 \left[1 - \exp\left(-\frac{T}{\tau_1}\right)\right] - i_B R_2 \left[1 - \exp\left(-\frac{T}{\tau_2}\right)\right] - i_B R_0, \tag{8}$$

where τ is the time constant, $\tau = RC$. The time T describes the discharge time of the battery. Through the circuit results $u_B = u_C$, where the calculations of u_C is introduced in the state vector $\vec{x}$ in Eq. (1).

Applied to the 3-RC model and rearranged to the battery current, this results for the time step $t + \Delta t$ in:

$$i_B(t+\Delta t) = \left[V_{OC}(SOC(t)) - u_B(t+\Delta t)\right] \left[R_1 \left[1 - \exp\left(-\frac{T}{\tau_1}\right)\right] + R_2 \left[1 - \exp\left(-\frac{T}{\tau_2}\right)\right] + R_3 \left[1 - \exp\left(-\frac{T}{\tau_3}\right)\right] + R_0\right]^{-1} \tag{9}$$

The SOC is modeled by coulomb counting and for the time step $t + \Delta t$ defined as:

$$SOC(t+\Delta t) = SOC(t) - \int_0^{\Delta t} \frac{i_B}{Q_0} dt, \tag{10}$$

where Q_0 indicates the rated capacity of the battery.

III. Implementation

To ensure a significant evaluation of the runtime the simulation was implemented using MATLAB, and Python libraries, NumPy [16], and CuPy [17], and the Python compiler Numba [18].

Numba - Just-in-Time compilation in Python
Numba is a LLVM-based just-in-time compiler for Python. The focus is in scientific and array-oriented computing [18]. This allows developers to implement Python code which can reach the execution time of C or FORTRAN. LLVM [19] is an open-source project that provides a collection of modular and reusable compiler and tool chain technologies.

CuPy - GPU-accelerated Computing with Python
CuPy is a library that allows to use the high performance parallelization of matrix operations on GPUs by using NumPy syntax or implementing NVIDIA CUDA Kernels in Python [17]. An illustration of the implemented CUDA Kernels to realize parallel execution is shown in Fig. 3.

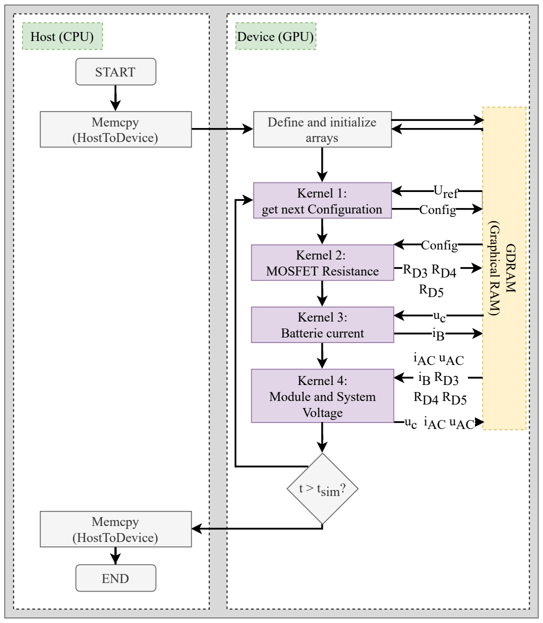
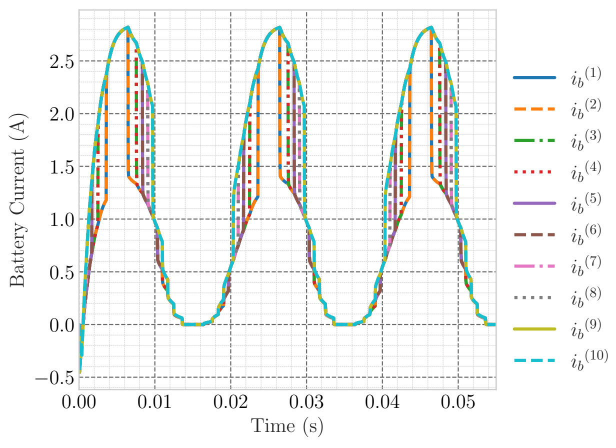

Host (CPU)
Device (GPU)
START
Memcpy (HostToDevice)
Define and initialize arrays
Kernel 1: get next Configuration
Kernel 2: MOSFET Resistance
Kernel 3: Batterie current
Kernel 4: Module and System Voltage
$t > t_{sim}$?
Memcpy (HostToDevice)
END
GDRAM (Graphical RAM)
U_{ref}
Config
Config
R_{D3} R_{D4} R_{D5}
u_c
i_B
i_{AC} u_{AC} i_B R_{D3} R_{D4} R_{D5}
u_c i_{AC} u_{AC}

Fig. 3. Architecture for parallel computing with CUDA Kernels on NIVIDIA GPU implemented in CuPy

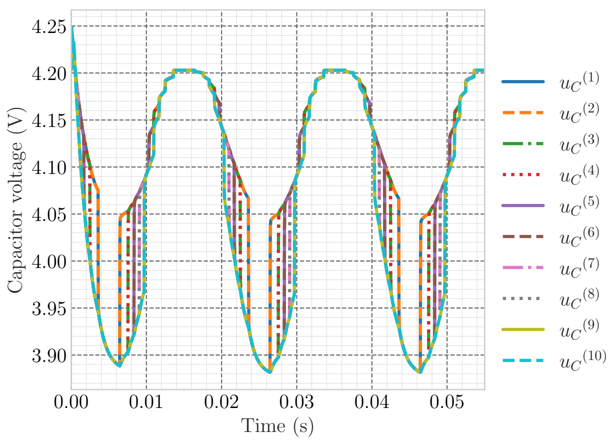
IV. Simulation Results

Fig. 4 shows the battery currents $i_B^{(k)}$ for each module, its capacitor voltages $u_C^{(k)}$, the state of the charge of the batteries and the inverter voltage u_{conv}, resulting current i_{AC}. As a battery cell for modelling, a KeepPower P1834J [20] was taken. The figure shows a $55\,\text{ms}$ simulation of a BM3 System with 10 modules with a step size of $\Delta t = 10\,\mu\text{s}$.

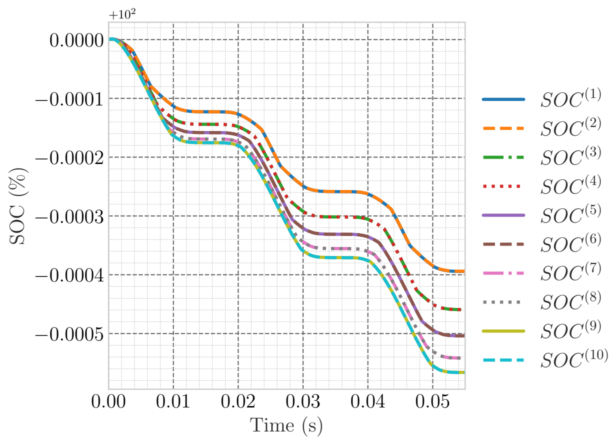
Fig. 4d shows that in the simulation, the ideal reference sinusoidal control signal u_{ref} is approximated by the multi-level system in a step by step manner. In total, 11 levels can be identified, which correspond to the connection of 1 up to



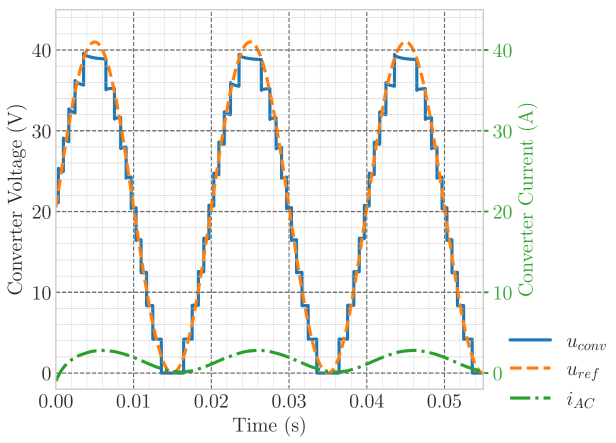
(a) Currents $i_B^{(1)}$ - $i_B^{(10)}$ of batteries



(b) Voltage $u_C^{(1)}$ - $u_C^{(10)}$ of batteries/capacitors



(c) State-of-Charge $SOC^{(1)}$ - $SOC^{(10)}$ of batteries



(d) Estimated voltage u_{ref}, BM3 converter Voltage u_{conv} and current i_{ac} of inverter system

Fig. 4. Simulation of System with $N = 10$, $t = 55\,\mathrm{ms}$, $\Delta t = 10\,\mu\mathrm{s}$

$N = 10$ modules in series, plus another one equal to 0 V, when all modules are not involved in the voltage generation and are in bypass mode.

In addition, the plot of the resulting voltage u_{conv} illustrates the influence of the capacitance of each module, especially in the areas corresponding to a voltage of 20 V or more. The resulting current i_{AC} has a smooth sinusoidal shape and takes a steady-state value after 5 ms, due to the presence of inductance L_{AC} in the system, it delays the voltage by 34.6 degrees.

Fig. 4a shows that each of the modules of the system during the time corresponding to one u_{conv} stage has a current, that differs in value from the current of the other modules. This is due to the different type of connection of the modules at different times of operation. The maximum current value is reached at the stage which corresponds to the maximum possible voltage value of 40 V. In order to obtain this voltage, all modules are switched in series with each other.

Fig. 4c demonstrates that the batteries in the different modules are not discharging evenly. This is due to the lack of a suitable algorithm for balancing the batteries in this program. Such an algorithm should take into account the state of the charge of each battery cell in the system and use the ones with the highest charge first.

TABLE II
HARD- AND SOFTWARE SPECIFICATIONS OF USED SERVER

Server	Dell PowerEdge R750xa
Operating System	Ubuntu 20.04 LTS
CPU	Intel Xeon Gold 6338
GPU	NVIDIA A40
RAM	512 GB

V. Discussion

A. Time measurement of different implementations

The hard- and software specification of the measurement's used server is shown in Table II.

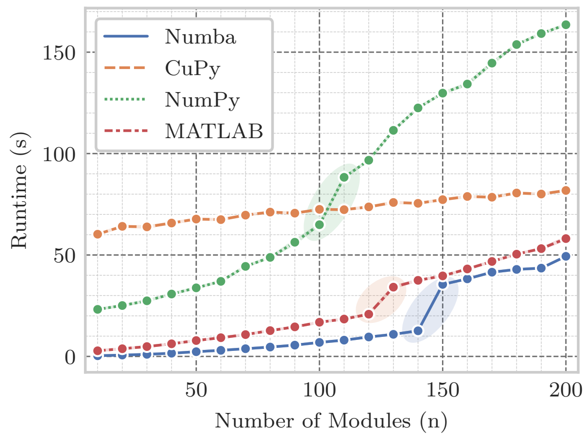


Fig. 5. Runtime of implementations by the number of modules. Simulated time 1 s with Δt =10 µs. The Numba (blue) implementation out-performs the other solutions. Runtime leap points are highlighted by colored ellipses.

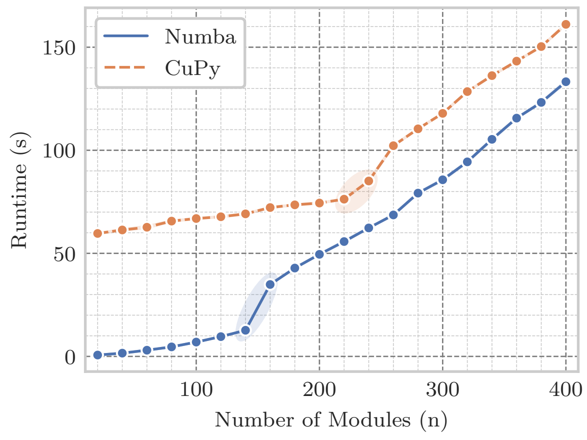


Fig. 6. Runtime comparison of CPU (Numba) and GPU (CuPy) solutions up to 400 modules

Fig. 5 shows the runtime by the number of modules N to simulate 1 s uptime of the system with Δt =10 µs. Ten experiments were performed and represented per implementation and number of modules. This figure shows the results of experiments from 20 modules up to 200 modules. The largest absolute increase of runtime is observed from the NumPy (green) implementation. The smallest relative increase can be observed from the GPU solution implemented with CuPy (orange). The best runtimes are to be seen at MATLAB (red) and specially Numba (blue). The experiments performed here do not show any advantages when run on the GPU compared to the results with Numba or MATLAB. Only worse runtimes can be determined. The measurements suggest that the CPU runtime exceeds the GPU, with a linear increase of this for about 300 modules.

To verify this prediction of a linear rise, a series of experiments up to 400 modules were conducted . The Numba and

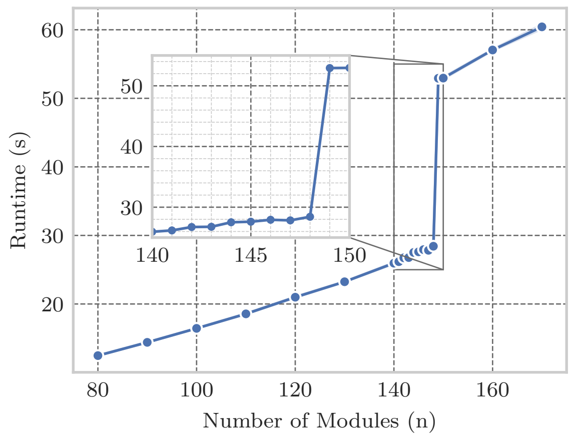


Fig. 7. Runtime by solving Eq. (2) with random matrix and vector implemented in NumPy

CuPy implementations are compared against each other. Figure 6 shows that the postulated intersection can not be determined. Between 220 and 240 modules, a leap point at the CuPy implementation can be detected. From this point, the runtime increases similarly for each of the two different solutions. Therefore, an increase in runtime performance cannot be determined with the current setup for the GPU solution.

B. Leap point in runtime

By analyzing the runtime of the simulation, shown in Fig. 5, the leap points in the NumPy (green , 100 - 110 modules), MATLAB (red , 120 - 130 modules) and Numba (blue , 140 - 150 modules) implementations can be observed . In the figure these leap points are highlighted. The same can be observed for CuPy between 220 and 240 modules in Fig. 6. An analysis of the runtime showed that the rise originates from how the algebraic system described in Eq. (2) is solved .

It is necessary to determine if this is a general problem or a phenomenon within the model. The problem is abstracted to further investigate this. The vector $\vec{v}$, and the matrix $\mathbf{A}$, defined in Eq. (6), and (3), are filled with random values. They are used to solve the algebraic system, as defined in Eq. (2).

Fig. 7 shows the measured runtimes by solving 10^5 of these random algebraic systems. From 148 to 149 modules an increase of 24.49 s can be observed. This is a rise of 86.1%, with the remaining increase being 5.94%.

A probable explanation could be the increased amount in needed memory. From a certain size of the matrix, it is necessary to use the Random-Access Memory (RAM) of the server instead of the CPU cache only . This leads to higher memory access times, more cache misses and thus to higher runtimes. Another assumption could be the solver implementation. Standard libraries are used, and it cannot be ruled out that from a certain size of the matrix another solver method is implemented. However, this is less likely, as we see the effect in all used programming languages and libraries. It’s more likely to find the cause for the leap points in a caching or memory configuration or size as we see the effect on different hardware setups with different matrix sizes.

C. Comparison with MATLAB/Simulink

A runtime comparison with the in [1] used MATLAB/Simulink model showed a significant improvement. To simulate a system with 10 modules, the Matlab model needed $42.10\,\mathrm{s} \pm 0.73\,\mathrm{s}$. This simulation uses the same simulation time and step size as configured during the time measurements depicted in Fig. 5. To simulate this number of batteries, the best implementation based on Pyton needed $0.33\,\mathrm{s}$ without significant differences in the simulation results. This reduces the runtime by at least factor 130. The here presented solution allows to simulate between 140 and 150 modules while the original Matlab/SIMULINK model can only simulate 10 modules at the same time.

VI. Conclusion

In this paper, a new framework was introduced which allows to simulate a BM3 converter system in one phase with a variable number of modules. Four different implementations were compared using MATLAB and the Python libraries NumPy, and CuPy, and the Python compiler Numba. Beside the observation of the programming languages MATLAB and Python, a comparison of the runtime performance of GPU and CPU was taken. It has been shown that there were no performance rises by using a GPU. The best runtime results were $0.33\,\mathrm{s} \pm 0.7\,\mathrm{ms}$ at 10 BM3 modules up to $133.25\,\mathrm{s} \pm 0.5386\,\mathrm{s}$ at 400 BM3 modules to simulate $1\,\mathrm{s}$ uptime of the system. These results were measured by using the Python compiler Numba.

Two variants were shown with the MATLAB and Numba implementation, which represent a high-performance alternative to the Matlab/SIMULINK implementation. A performance gain of more than factor 130 could be achieved at $1\,\mathrm{s}$ simulation time with 10 modules.

VII. Outlook

In modelling the next step, this introduced framework will be extended by a three-phase electrical motor. Among other things, this extension should make it possible to gain insights into balancing methods or the evaluation of losses of the converter system . This will make it possible to simulate driving cycles such as the Worldwide harmonized Light vehicles Test Procedure (WLTP) without resorting to time consuming methods such as lookup tables.

References

[1] A. Mashayekh, A. Kersten, M. Kuder, J. Estaller, M. Khorasani, J. Buberger, R. Eckerle, and T. Weyh, “Proactive SoC Balancing Strategy for Battery Modular Multilevel Management (BM3) Converter Systems and Reconfigurable Batteries,” in *2021 23rd European Conference on Power Electronics and Applications (EPE’21 ECCE Europe)*. IEEE, Oct. 2021.

[2] “KOALITIONSVERTRAG ZWISCHEN SPD, BÜNDNIS 90/DIE GRÜNEN UND FDP,” p. 178, Dec. 2021.

[3] A. Poorfakhraei, M. Narimani, and A. Emadi, “A Review of Multilevel Inverter Topologies in Electric Vehicles: Current Status and Future Trends,” *IEEE Open Journal of Power Electronics*, vol. 2, pp. 155–170, 2021.

[4] I. Aretxabaleta, I. M. De Alegría, J. Andreu, I. Kortabarria, and E. Robles, “High-voltage stations for electric vehicle fast-charging: trends, standards, charging modes and comparison of unity power-factor rectifiers,” *IEEE Access*, 2021.

[5] K. Thakre, K. B. Mohanty, and A. Chatterjee, “Modelling and Simulation of an Asymmetrical Modular Multilevel Inverter with Less Number of Components,” *EPE Journal*, vol. 30, no. 2, pp. 69–79, Apr. 2020.

[6] R. Niraimathi and R. Seyezhai, “Analysis, simulation and implementation of a novel dual bridge asymmetric cascaded multi level inverter using MGWO-PI-PWM controller,” *Microprocessors and Microsystems*, vol. 77, Sep. 2020.

[7] N. Sorokina, J. Estaller, A. Kersten, J. Buberger, M. Kuder, T. Thiringer, R. Eckerle, and T. Weyh, “Inverter and Battery Drive Cycle Efficiency Comparisons of Multilevel and Two-Level Traction Inverters for Battery Electric Vehicles,” in *2021 IEEE International Conference on Environment and Electrical Engineering and 2021 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I CPS Europe)*, Sep. 2021.

[8] A. Kersten, M. Kuder, E. Grunditz, Z. Geng, E. Wikner, T. Thiringer, T. Weyh, and R. Eckerle, “Inverter and battery drive cycle efficiency comparisons of CHB and MMSP traction inverters for electric vehicles,” in *2019 21st European Conference on Power Electronics and Applications (EPE ’19 ECCE Europe)*. IEEE, 2019.

[9] R. Zhu, N. Lin, V. Dinavahi, and G. Liang, “An Accurate and Fast Method for Conducted EMI Modeling and Simulation of MMC-Based HVdc Converter Station,” *IEEE Transactions on Power Electronics*, vol. 35, no. 5, pp. 4689–4702, May 2020.

[10] M. Kuder, J. Schneider, and A. Kersten, “Battery Modular Multilevel Management (BM3) Converter applied at Battery Cell Level for Electric Vehicles and Energy Storages,” in *PCIM Europe digital days 2020; International Exhibition and Conference for Power Electronics, Intelligent Motion, Renewable Energy and Energy Management*. VDE, Jul. 2020.

[11] M. Schmid, U. Vögele, and C. Endisch, “A novel matrix-vector-based framework for modeling and simulation of electric vehicle battery packs,” *Journal of Energy Storage*, vol. 32, Dec. 2020.

[12] V. Sulzer, S. G. Marquis, R. Timms, M. Robinson, and S. J. Chapman, “Python Battery Mathematical Modelling (PyBaMM),” *Journal of Open Research Software*, vol. 9, no. 1, Jun. 2021, number: 1 Publisher: Ubiquity Press.

[13] X. Ding, D. Zhang, J. Cheng, B. Wang, and P. C. K. Luk, “An improved Thevenin model of lithium-ion battery with high accuracy for electric vehicles,” *Applied Energy*, vol. 254, Nov. 2019.

[14] M.-K. Tran, A. DaCosta, A. Mevawalla, S. Panchal, and M. Fowler, “Comparative study of equivalent circuit models performance in four common lithium-ion batteries: Lfp, nmc, lmo, nca,” *Batteries*, vol. 7, no. 3, p. 51, 2021.

[15] D. Liu, X. Wang, M. Zhang, and M. Gong, “SOC Estimation of Lithium Battery Based on N-2RC Model in Electric Vehicle,” in *2019 Chinese Control And Decision Conference (CCDC)*, Jun. 2019, pp. 2916–2921, iSSN: 1948-9447.

[16] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020.

[17] R. Okuta, Y. Unno, D. Nishino, S. Hido, and C. Loomis, “CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations,” in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, 2017.

[18] S. K. Lam, A. Pitrou, and S. Seibert, “Numba: a LLVM-based Python JIT compiler,” in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, ser. LLVM ’15. New York, NY, USA: Association for Computing Machinery, Nov. 2015.

[19] “The LLVM Compiler Infrastructure Project.” [Online]. Available: https://llvm.org/

[20] J. Estaller, A. Kersten, M. Kuder, A. Mashayekh, J. Buberger, T. Thiringer, R. Eckerle, and T. Weyh, “Battery Impedance Modeling and Comprehensive Comparisons of State-Of-The-Art Cylindrical 18650 Battery Cells considering Cells’ Price, Impedance, Specific Energy and C-Rate,” in *2021 IEEE International Conference on Environment and Electrical Engineering and 2021 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I CPS Europe)*, Sep. 2021.