This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TCAD.2016.2597224

Rapid Analysis of Active Cell Balancing Circuits

Matthias Kauer, *Student Member, IEEE*, Swaminathan Narayanaswamy, Sebastian Steinhorst, *Member, IEEE*, Samarjit Chakraborty, *Senior Member, IEEE*

***Abstract*—Active cell balancing improves the performance of a battery pack by transferring charge from one cell to another. Associated design questions require multiple simulations with 100 cells over several hours. Since the most efficient transfer methods switch between phases in the kilohertz range, these simulations require high computational effort or reduced accuracy.**

To enable detailed analysis on a large scale, this work includes state-of-the-art electrical battery models in active balancing simulation while keeping the computation effort for one transfer in the low millisecond range. This is achieved in three steps. First, we model the dynamics of each transfer phase using standard equivalent circuit abstraction. Next, we find closed form equations for the so-defined phase dynamics, yielding an iterative approach that saves computation time by replacing the numerical solver. Finally, we employ error control techniques to aggregate phases in that iteration, systematically reducing the millions of phase evaluations that would be necessary otherwise. Our experiments show that the speedup from equivalent circuit dynamics to error-controlled aggregation almost reaches 5 orders of magnitude while introducing virtually no additional error. This enables simulations of realistic balancing scenarios in less than a second and is hence suitable for design space exploration.

***Index Terms*—Batteries, Battery management systems, Differential equations, Charge equalization, Active cell balancing, Numerical simulation**

I. Introduction & Related Work

Context. Justified by their energy density, Lithium-Ion (Li-Ion) cells currently form the basis for a wide range of applications. These include smartphones, laptops, as well as electric vehicles. Since their cell chemistry limits voltage to about 4V, many cells are typically connected in series and/or parallel to provide the required power output. While cells in parallel connection are inherently balanced and can be treated as electrical unit, the charging or discharging of serially connected cells must stop as soon as the first cell reaches its limit. Imbalances between these cells, caused by parameter differences from production or by non-homogeneous cooling, hence reduce the effective capacity of the pack.

For applications that require high voltage, and thus many cells in series, imbalances are currently alleviated in two ways. Cells with similar properties are clustered at production time to minimize deviations in capacity and internal resistance of over 5% [1]. This is not sufficient, however, since cell properties evolve differently over time, even for identical cells under lab conditions [2]. Additionally, excess energy in individual cells is thus dissipated using switchable resistors [3]. This *passive balancing* is easy to implement, but not energy-efficient.

Alternatively, the excess charge can also be transferred to other cells, increasing effective capacity and performance. This approach, referred to as *Active Cell Balancing (ACB)*, is implemented by numerous circuit architectures [4]. The most efficient ones are built around temporary storage elements, like inductors or transformers, and actuated with switching signals in the kilohertz range. While we believe that the simulation principles are similar for others, we only investigate the family of inductor-based circuits in this work.

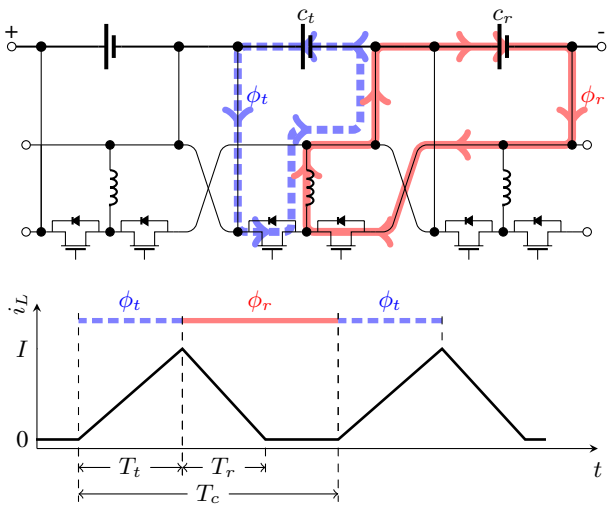

Fig. 1: *In the inductor-based architecture from [5], transistor switching drives the charge transfer. During transmitting phase ϕ_t, transmitting cell c_t lets inductor current i_L rise to peak current I. After T_t has elapsed, ϕ_r begins and the inductor discharges into the receiving cell. After the variable receiving period T_r, the current reaches $i_L = 0$ and discharge ends.*

Inductor-based charge transfer operation. Consider the balancing architecture in Fig. 1 transferring charge from cell c_t to cell c_r and the corresponding inductor current i_L. In the transmitting phase ϕ_t, the inductor is charged from cell c_t. After a period of T_t, the switch configuration changes and the inductor discharges into cell c_r for a period of T_r which is unknown a priori. The process restarts after cycle time T_c.

Problem statement. Accurate ACB simulation is currently too slow to evaluate system-level scenarios with over 100 battery cells in an interactive fashion. The main challenge are the switching transitions that change the configuration and consequently the system dynamics in a non-differentiable fashion. To adequately capture these transitions, general purpose simulators must increase their resolution before the end of each phase, severely limiting the simulation speed. Balancing scenarios with many cells and hours of simulation time that consider details in the microsecond range hence require hours to be computed in this way. As such computation times are often unacceptable, balancing simulations typically either treat small scenarios [5] or simplify the underlying dynamics [6].

Related work. As first alternative to omitting details or limiting scenario size, the equivalent circuit abstraction of each transfer phase can be considered individually which yields an Ordinary Differential Equation (ODE) for the behavior within a phase. In some cases, the solution of this ODE is available in closed form and leads to a faster, iterative simulation approach where the system evolution is calculated phase-by-phase without numerical solver. Such approaches have been pursued previously, for instance in [7], but only simple battery models with one state have been considered.

Electrical battery models with multiple states that consist of a voltage source and several resistor-capacitor stages are considered state-of-the-art, however. It has been demonstrated that simpler models that

Matthias Kauer, Swaminathan Narayanaswamy, and Sebastian Steinhorst are with TUM CREATE Ltd., Singapore (e-mail: matthias.kauer@tum-create.edu.sg).

Samarjit Chakraborty is with the Institute for Real-Time Computer Systems, Technische Universität München, Germany (e-mail: samarjit@tum.de)

This work was financially supported in part by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

Copyright (c) 2016 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubs-permissions@ieee.org.

Copyright (c) 2016 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TCAD.2016.2597224

are commonly used in ACB do not track voltage well [8]. Although the error is negligible for small currents, it may reach 3% even for ordinary currents around 1 C[1]. When balancing immediately after regular operation, such as driving of an electric vehicle, where higher currents are typical, the error may be even larger. The inclusion of a more accurate battery model, as discussed in this work, is thus necessary to broaden the scope of ACB design.

Contributions & paper organization. This paper is concerned with accurate, but fast simulation of ACB. Considering the charge transfer dynamics in localized fashion via standard equivalent circuit abstraction, it compares three simulation approaches that build on one another. The first, straightforward approach (Section II-A) uses a numerical solver for the intra-phase dynamics. It represents a general purpose simulator and is included as a reference. The second approach (Section II-B) assumes constant voltages within a phase and derives closed-form equations for the intra-phase evolution. This leads to an iterative method that has been similarly derived in previous works like [9]. The paper at hand improves upon those results by (i) integrating the more sophisticated cell model from [8] into the formulation for a higher accuracy at equivalent computation speed. Instead of calculating the so-defined iteration phase by phase, the third approach, introduced here, aggregates phases by (ii) applying error control and adaptive step size techniques from the ODE domain (Section II-C).

Our experiments confirm that the equivalent circuit abstraction leads to less than 0.5% deviation from SPICE simulations (Section III-A). Although introducing virtually no error compared to the equivalent circuit dynamics, the phase aggregation technique is more than 2000 times faster than the iterative approach from previous works like [9] and more accurate thanks to the improved battery model (Section III-B).

II. Equivalent Circuit Charge Transfer Models

Fig. 1 shows a single transfer between cells c_t and c_r, occurring in two phases. There may be other, concurrent transfers with their own phases, but they cannot share connections. Please refer to, e.g., [7] for a more detailed discussion on the switching rules that enable such transfers. Each phase only involves a subset of the overall circuit and can be described using an equivalent circuit with aggregated resistances. For instance, in the architecture from Fig. 1, the serial resistances R_t and R_r of the respective phases summarize to

$$R_t = R_M + R_L + R_0 \qquad R_r = R_M + R_L + R_0. \tag{1}$$

Here, R_0 is the internal resistance of the battery and R_M, R_L refer to the resistance of transistor and inductor, respectively. Fig. 2 shows the equivalent circuits that correspond to the transfers in Fig. 1. Transmitting and receiving cell c_t and c_r have been expanded to the model from [8], with the serial resistance integrated into R_t, R_r. Many circuits from the family of inductor-based charge transfer circuits can be analyzed from this perspective, varying only their resistance formula.

A. System dynamics within a transfer phase

This section introduces the dynamics of the ACB process. As such, it serves first of all as stepping stone for the following sections. Moreover, the straightforward application of a numerical solver to these dynamics is close to the computation performed by a circuit simulator. For this reason, we also use the approach from this section as reference solution with the highest accuracy, but longest computation time.

[1]A current of 1 C is defined to fully charge or discharge a cell in 1 hour.

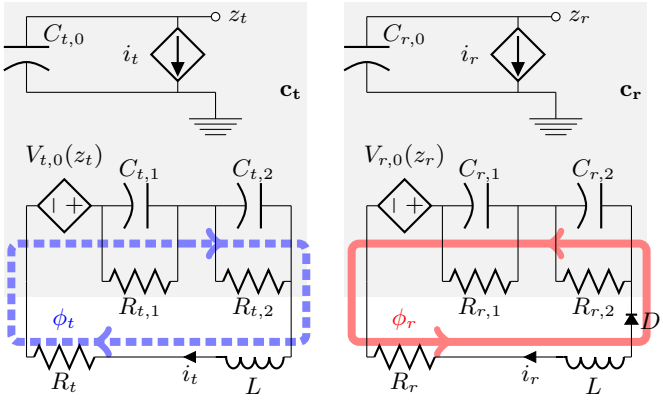

Fig. 2: *By aggregating all resistances on the current path, we can describe the charge transfer in two equivalent circuits. The current in the inductor rises during ϕ_t until the transistors switch over and the inductor discharges during ϕ_r. Cells c_t and c_r have been expanded to the accurate model from [8]. Diode D is not necessary in some circuit variants; we then set $V_d = 0$.*

Consider the equivalent circuits from Fig. 2. They show the charge transfer from Fig. 1 separated into two phases, ϕ_t and ϕ_r. An electrical model with two resistor-capacitor stages, as in [8], has been substituted for the battery cells. Such models are currently state-of-the-art; two stages are the most common choice although "in many studies, using one RC pair has shown accurate enough performance" as Rahimi-Eichi et al. summarize in their overview paper [10]. Note that the approach from the paper at hand can accommodate additional stages in a straightforward fashion. For parameterization details, please refer to the respective literature.

The dynamics of the transmitting and receiving cells are identical except for their parameters and current direction. The current direction follows the convention that positive currents discharge and negative currents charge a cell. The charge differences q, being integrals of the currents, adopt the same signs. In this way, we can always add q without distinguishing cases and the Resistor-Capacitor (RC) voltages receive the correct sign automatically.

The system dynamics have four states per cell and 8 states in total: i_t, i_r are the respective cell currents and $Q_{t,0}$, …, $Q_{r,2}$ are the charges in the respective capacitors from Fig. 2. Q_0, a short notation for both $Q_{t,0}$ and $Q_{r,0}$, models the charge of the cell itself. It is often expressed in relation to C_0 rather than as absolute value. This ratio $z = Q_0/C_0$ is referred to as the State-of-Charge (SoC). Absolute changes in SoC are denominated in percentage (pp) or basis points (bp). $V_0(Q_0)$, the Open Circuit Voltage (OCV), describes the relation between a cell's charge and its voltage, without external influences. The OCV is obtained from measurements and typically expressed as piecewise linear function or via another form of curve-fitting. The voltages of the RC stages, modeling parasitic voltage effects, are given by $V_j(Q_j) = Q_j/C_j$ for $j = 1, 2$. The voltages from the various stages then yield the following total voltages for the respective phases:

$$V_t = \sum_{j=0}^{2} V_{t,j}(Q_{t,j}) \qquad V_r = \sum_{j=0}^{2} V_{r,j}(Q_{r,j}) + V_d \tag{2}$$

V_d models additional voltage from the diode preventing undesirable discharge of c_r during ϕ_r. This is not required during ϕ_t and elaborate switching signals allow forgoing the diode entirely, modeled by setting $V_d = 0$, to increase efficiency.

Copyright (c) 2016 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TCAD.2016.2597224

With the voltage definitions from (2), the dynamics of the main mesh in Fig. 2, can be described as follows.

$$\frac{\mathrm{d}}{\mathrm{d}t}Q_0 = -i \qquad \frac{\mathrm{d}}{\mathrm{d}t}i = \frac{1}{L}[V - iR] \tag{3}$$

In addition to Q_0 in (3), the RC stages must also be updated.

$$\frac{\mathrm{d}}{\mathrm{d}t}Q_j = -i - \frac{V_j(Q_j)}{R_j} \quad j = 1, 2 \tag{4}$$

When $i_t = 0$ or $i_r = 0$, because the corresponding transistor is not conducting, the respective cell is *relaxing*. In this case, we must still update the resistor-capacitor stages using (4).

Given timing parameters T_t, T_c, the system can now be simulated as follows.

- ϕ_t Solve ODE system (3 & 4) with $i_t(0) = 0$ and end time T_t to update transmitting cell c_t. This updates states $\begin{bmatrix} Q_{t,0} & Q_{t,1} & Q_{t,2} \end{bmatrix}$ and yields peak current I. Simultaneously, relax receiving cell c_r by solving ODE (4). This updates states $\begin{bmatrix} Q_{r,1} & Q_{r,2} \end{bmatrix}$.
- ϕ_r After the circuit switches over, relax transmitting cell c_t with ODE (4), updating $\begin{bmatrix} Q_{t,1} & Q_{t,2} \end{bmatrix}$. Solve ODE system (3 & 4) with $i_r(T_t) = -i_t(T_t)$ and end time T_c for c_r. This updates $\begin{bmatrix} Q_{r,0} & Q_{r,1} & Q_{r,2} \end{bmatrix}$.
- * Repeat until K, the desired number of cycles, is reached.

Note that during ϕ_r, receiving time T_r is not known a priori. Its calculation can be circumvented by setting $i_r(t) := \max(0, i_r(t))$ before updating charges. Alternatively, one may use (8) from Section II-B to calculate the corresponding timing directly or perform a binary search over the time domain if the ODE solver permits. These alternative approaches also handle requirements like constant peak current I, implying T_t adjustments at runtime and immediate switching back to ϕ_t when $i = 0$ and T_r has elapsed to ensure $T_c = T_t + T_r$.

B. Closed form equation for intra-phase behavior

All capacitors from the cell model in Fig. 2 are large, modeling effects in the domain of seconds and minutes. Assuming their voltages remain constant during individual cycles hence does not introduce noticeable errors, but it lets us derive closed-form solutions for current and charge movement. The capacitors are then updated after each cycle. This reformulation has been used previously to speed up simulation, e.g., in [9]. Whereas those results relied on a simple battery model with a single state, the paper at hand includes a state-of-the-art model for higher accuracy. Nevertheless, the computational effort remains in the same range and the formulation from this section hence also represents previous computation speeds in comparison with the approach from Section II-C.

With the assumption of a constant voltage V, ODE (3) has a unique solution:

$$i(t, V, i_0, R) = \frac{V}{R} - \frac{V - i_0 R}{R} \exp\left(\frac{-R}{L}t\right) \tag{5}$$

The ODE parameters V, $i_0 = i(0)$, R need to be adjusted according to the individual phase as described in the previous section. To update charges and thus voltages in the model, we first obtain the main update term q by integrating i.

$$\begin{aligned} q(T, V, i_0, R) &= \int_0^T -i(t, V, i_0, R)\,\mathrm{d}t \\ &= \frac{V}{R}T + \frac{L(V - i_0 R)}{R^2}\left[\exp\left(\frac{-R}{L}T\right) - 1\right] \end{aligned} \tag{6}$$

Here, the integration constant is chosen such that $q(0) = 0$.

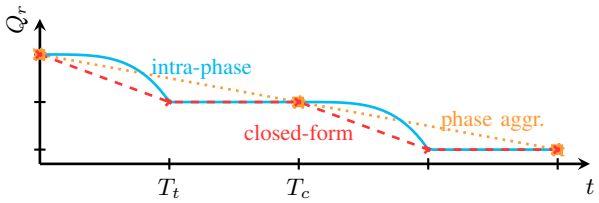

Fig. 3: *Modeling intra-phase behavior (——, Section II-A) provides the most accurate charge information at all times. When solving phases in closed form (- - -, Section II-B), we obtain accurate charge information only at the end of phases. In order to avoid kinks, the model underlying phase aggregation (·····, Section II-C) is content being accurate only at the end of cycles.*

The updates of the RC stages must consider their self-discharge in addition to the effects of balancing current i. Integrating that self-discharge term in (4), we obtain

$$q_j(T, Q_j) = -\frac{V_j(Q_j)}{R_j}T. \tag{7}$$

For time calculations, the current from (5) is also invertible. More concretely, given desired current i_d, the time T_d, fulfilling $i(T_d) = i_d$ can be calculated as

$$T_d(i_d, V, i_0, R) = \frac{-L}{R}\log\left(\frac{V - i_d R}{V - i_0 R}\right). \tag{8}$$

This equation is helpful in several situations. If peak current I is specified, it calculates on-the-fly timing. During ϕ_r, it is required even if timing parameters are specified. As shown in Fig. 1, the inductor current reaches $i_L = 0$ before T_c. To calculate the transferred charge, the discharge period $T_r = T_d(0, V_r, -I, R_r)$ must be known.

All formulas of this section are only valid as long as the system configuration does not switch. Over one cycle of charge and discharge the dynamics move through different equivalent circuits with different parameters and initial conditions. For simulations of multiple cycles, the following iterative approach is hence required.

1) Obtain cell voltages from (2).
2) Calculate peak current I given transmitting time T_t with (5). Alternatively, calculate T_t given I using (8).
3) Update charges for phase ϕ_t, using partially specialized $q_t(T) := q(T, V_t, 0, R_t)$ from (6) and $q_{t,j}$, $q_{r,j}$ from (7).

$$\begin{aligned} Q_{t,0}(t + T_t) &= Q_{t,0}(t) + q_t(T_t) \\ Q_{t,j}(t + T_t) &= Q_{t,j}(t) + q_t(T_t) + q_{t,j}(T_t, Q_{t,j}(t)) \\ Q_{r,0}(t + T_t) &= Q_{r,0}(t) \\ Q_{r,j}(t + T_t) &= Q_{r,j}(t) + q_{r,j}(T_t, Q_{r,j}(t)) \end{aligned} \tag{9}$$

4) From (8), calculate discharging time T_r until $i = 0$ and discharge of c_r ends. Ensure $T_r + T_t \leq T_c$ if T_c has been specified, then analogously update charges for phase ϕ_r, using $q_r(T_r) := q(T, V_r, -I, R_r)$ from (6).
5) Relax all RC stages for $T_c - (T_r + T_t)$ with q_j from (7).
6) Repeat, until K, the desired number of cycles, is reached.

C. Error-controlled, adaptive phase aggregation

In this section, adaptive techniques from the ODE domain are applied to the recurrence relation in Section II-B. This method requires knowledge of embedded Runge-Kutta methods as well as reformulations and challenges from the ACB domain, like the variable timing discussed in the next paragraph. Consequently, most ACB simulations are currently still performed by directly applying a

Copyright (c) 2016 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TCAD.2016.2597224

numerical solver, as in Section II-A, or with a recurrence relation, as in Section II-B.

The previous abstraction levels calculate system dynamics over time. Here, we consider the evolution over cycles k. Each cycle has a duration of T_c that varies slightly if a constant peak current is maintained as voltages change. It is thus necessary to track time as a separate state variable in this formulation. Tracking time in this way is simpler than scaling results to account for time variations. This perspective also makes it easier to reason about the accuracy loss outside cycle ends that we accept to alleviate kinks (see Fig. 3).

In the following, we adopt the formulations from Section II-B to sum up the state differences of a single cycle:

$$\begin{aligned}
\begin{bmatrix} V_t & V_r \end{bmatrix} &= \begin{bmatrix} \sum_{j=0}^{2} V_{t,j}(Q_{t,j}) & \sum_{j=0}^{2} V_{r,j}(Q_{r,j}) + V_d \end{bmatrix} \\
\begin{bmatrix} I & T_r \end{bmatrix} &= \begin{bmatrix} i(T_t, V_t, 0, R_t) & T_d(0, V_r, -I, R_r) \end{bmatrix} \\
\frac{\Delta t}{\Delta k} &= T_c \\
\frac{\Delta Q_t}{\Delta k} &= q(T_t, V_t, 0, R_t) \\
\frac{\Delta Q_{t,j}}{\Delta k} &= q(T_t, V_t, 0, R_t) + q_{t,j}(T_c, Q_{t,j}) \quad , j = 1, 2 \\
\frac{\Delta Q_r}{\Delta k} &= q(T_r, V_r, -I, R_r) \\
\frac{\Delta Q_{r,j}}{\Delta k} &= q(T_r, V_r, -I, R_r) + q_{r,j}(T_c, Q_{r,j}) \quad , j = 1, 2
\end{aligned} \tag{10}$$

This is the formulation for fixed timing parameters T_t, T_c. Constant peak current can be achieved by calculating T_t, T_c on the fly with (8) in addition to T_r.

(10) is not inherently an ODE because $k, \Delta k \in \{1, 2, \dots K\}$ are discrete variables. Nevertheless, we are interested in applying ODE techniques with error control and adaptive step size. The typical choice for these requirements are solvers from the class of embedded Runge-Kutta methods. An unadjusted solver from that class may also evaluate the right-hand side for non-integral cycle counts and thus at instants where charge evolution is incorrectly interpolated. We will first quantify the magnitude of this issue and then propose a remedy for it. Consider the following worst case calculation. With an average balancing current equivalent to a C rate of 1 and using a low frequency with $T_t = 10\mathrm{ms}$, the SoC changes only by

$$\Delta z = \frac{\Delta Q}{C_0} = \frac{\frac{1 \cdot C_0}{3600\mathrm{s}} \cdot 10\mathrm{ms}}{C_0} \approx 2.7\text{e-}4\mathrm{pp} = 0.027\mathrm{bp}.$$

Even assuming a region where the OCV increases quickly by $100\frac{\mathrm{mV}}{\mathrm{pp}}$, Δz corresponds to only $\Delta V = 0.027\mathrm{mV}$. With a minimum cell voltage of $2.5\mathrm{V}$, the worst case error thus remains in the order of 1e-5 and is typically far smaller. As cell models come with inherent modeling errors in the order of 1e-3 [8], one may accept this error and resort to readily available methods directly.

Alternatively, ensuring that embedded Runge-Kutta solvers only evaluate integral cycle counts is also possible. These solvers have fixed nodes, proportional to the current step size Δk, at which a function is evaluated. For instance, the Bogacki-Shampine method [11], combining orders 2 and 3, evaluates at $0\Delta k$, $\frac{1}{2}\Delta k$, $\frac{3}{4}\Delta k$ and $1\Delta k$. Restricting steps Δk to multiples of their common denominator, here 4, ensures that all evaluation points are integral. Other popular embedded methods with order 4/5, like Dormand-Prince [12] which is the default solver in MATLAB and Cash-Karp [13] similarly require multiples of 90 and 40, respectively.

III. Experimental Results

We have implemented the approaches described in Section II, using the C++ library boost::odeint [14]. This forms the back end of our implementation that we drive from Python.

All battery parameters correspond to the $850\mathrm{mA}$ Li-Ion polymer cell which has been characterized in [8] for various SoC levels. We only replace their analytic OCV relation with a piecewise linear formulation because it interacts better with the rest of our framework. This model comes with a good SoC (or runtime) error of 0.4% and a voltage deviation of $15\mathrm{mV}$.

We evaluate accuracy and speedup of the proposed techniques in two stages. First, we compare the intra-phase model from Section II-A to a circuit simulator to quantify the errors introduced by the equivalent circuit abstraction. Next, we compare all abstraction levels from Section II to each other, using longer simulation times of several minutes. Although impossible for the circuit simulator, such times are still small for ACB strategies.

A. Equivalent circuit model accuracy & speedup

We investigate a transfer between a transmitting cell at 60% SoC and a receiving cell at 40%, setting the cell parameters accordingly. In addition, we modeled an inductor resistance $R_L = 1\mathrm{m}\Omega$ and a diode voltage $V_d = 0.7\mathrm{V}$. This transfer is simulated in SPICE and with the intra-phase ODE from Section II-A. We perform the transfer over a variety of Pulse Width Modulation (PWM) frequencies, keeping the peak current approximately constant by suitably decreasing the inductance with shorter timing. All transfers are executed over 10 cycles of fixed timing. Fig. 4 shows two sets of this experiment. The nominal peak currents, assuming lossless dynamics, were $400\mathrm{mA}$ and $2\mathrm{A}$. Although we do not see a discernible trend, the relative error in SoC remained below 0.5% over the investigated frequencies. We have selected these frequencies to span more than the most relevant range between $1\mathrm{kHz}$ and $100\mathrm{kHz}$. The SoC evolution depends on all voltages and is, as such, the most error-sensitive value in the system. Relative errors in the various voltages all remained even smaller. We attribute these errors to several aspects that are only modeled in the circuit simulator, like diodes and switches or imperfect, but realistic current edges. Since the errors do not exceed inherent modeling errors of the cell model, the equivalent circuit abstraction is well justified.

The speedup in this experiment is significant. All SPICE simulations required over $5\mathrm{s}$ to calculate 10 cycles at the desired accuracy. With the equivalent circuit abstraction, the intra-phase model solves the same task in the millisecond range. Although the simulation time varies widely, the higher frequency and accuracy requirement in the shorter experiments keep the computation effort roughly constant.

B. Abstraction level accuracy & speedup

The previous section mainly evaluated inaccuracies introduced by the equivalent circuit abstraction. This section investigates the techniques from Section II building on that abstraction. For this purpose, we consider the same cell pair with 60 and 40% SoC

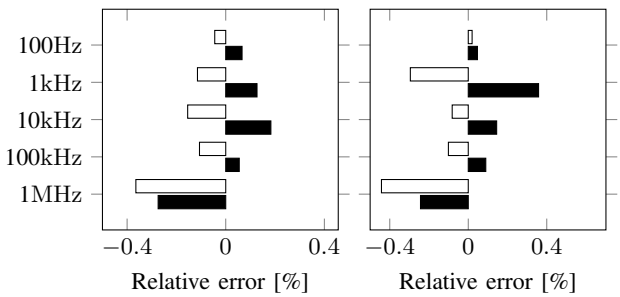

Fig. 4: *For both nominal currents of* $400\mathrm{mA}$ *(left) and* $2\mathrm{A}$ *(right), the relative error in SoC of transmitting (black) and receiving (white) cell remains small compared to SPICE over a wide frequency range.*

Copyright (c) 2016 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TCAD.2016.2597224

over longer simulation times with randomized transfer parameters. We draw resistances $R = 10^x\Omega, x \in [-3, 0]$ and peak currents $I \in [0.1\text{A}, 2.5\text{A}]$, according to a uniform distribution. Timing parameters were fixed to $\begin{bmatrix} T_t & T_c \end{bmatrix} = \begin{bmatrix} 200\mu\text{s} & 420\mu\text{s} \end{bmatrix}$. The inductance $L = \frac{4.0\text{V}}{I} T_t$ is scaled to approximately achieve the desired peak current. After drawing the random parameters, we perform $K = 10^6$ cycles with the respective settings.

Fig. 5 shows the results from 50 such random transfers. Although the closed form approach from Section II-B and the phase aggregation technique from Section II-C introduce *virtually no additional error*, they achieve a *remarkable speedup*. The closed form approach is roughly 35 times faster than the intra-phase ODE solver, and phase aggregation is another 2500 times faster. Compared to intra-phase ODE, phase aggregation hence achieves a speedup of about 90000. These measurements were made on a workstation with 3.4GHz Intel i7-3770 CPU and 16GB RAM. All implementations are single-threaded. Note that the computation time of each approach does not vary a lot over the experiment set, indicating that the required effort depends on the number of simulation cycles and not the total simulation time. With higher frequency, the proposed methods therefore become even more beneficial.

Extrapolating the computation times from Fig. 5 involving two cells and a simulation time of about 7 minutes, we consider a battery pack that performs 10 parallel transfers, using a frequency of 20kHz (10 times higher) and a simulation time of 7 hours. For this 6000 times larger scenario, a general purpose solver requires at least a full day. Closed-form iteration reduces that time, but still needs more than 40 minutes. Only the phase aggregation approach calculates results in less than 2 seconds and remains fast enough to interactively evaluate balancing strategies with respect to balancing time and efficiency.

Fig. 6 shows an example time series plot from one of the random transfers that we performed, demonstrating why phase aggregation is so much faster. Instead of calculating millions of cycles one by one, it requires only dozens of evaluations to provide the desired result. This plot also demonstrates the importance of the RC stages in the model, as the voltage they contribute on the receiver side exceeds 40mV.

IV. Concluding Remarks

This paper improves the design flow of ACB circuits and strategies with a simulation technique that calculates detailed transfer dynamics responsively, even for large scenarios. The numerical error has been carefully controlled in all stages of abstraction and never exceeds the inherent modeling error. The achieved speedup of over 2000 compared to previous techniques is crucial, in particular for interactive applications and ACB routing design where a parameterized strategy has to be evaluated over a large set of scenarios.

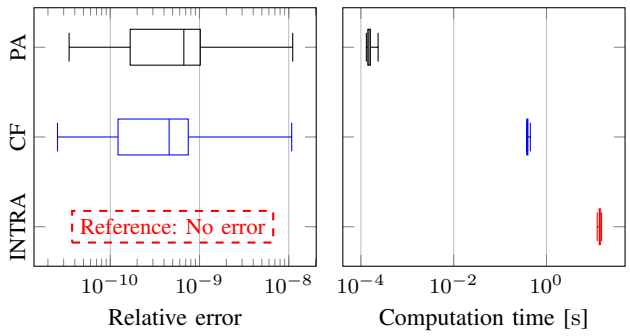

Fig. 5: *Box plot showing percentiles 0/25/50/75/100: Neither closed form (CF, relerr* 2.6e-11 *to* 1.1e-8*, mean* 7.0e-10*) nor phase aggregation (PA, relerr* 3.5e-11 *to* 1.1e-8*, mean* 8.7e-10*) method introduce noticeable errors into the simulation over a direct solution of the equivalent circuit dynamics (INTRA) (all methods in Section II). However, speedups of several orders of magnitude are achieved.*

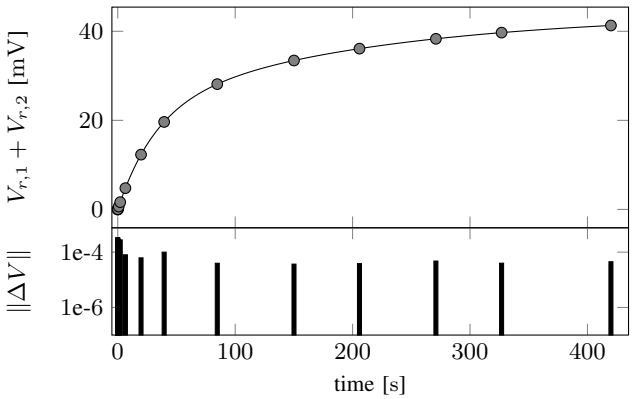

Fig. 6: *The phase aggregation approach (circles) tracks the intra-phase ODE solution (line) perfectly, calculating only dozens, not millions, of cycles. In the RC voltage of the receiver, the absolute error remains around* 1e-4mV.

References

[1] D. Shin, M. Poncino, E. Macii, and N. Chang, "A Statistical Model-Based Cell-to-Cell Variability Management of Li-ion Battery Pack," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 252–265, 2015.

[2] T. Baumhöfer, M. Brühl, S. Rothgang, and D. U. Sauer, "Production caused variation in capacity aging trend and correlation to initial cell performance," *Journal of Power Sources*, vol. 247, pp. 332–338, 2014.

[3] M. Isaacson, R. Hollandsworth, P. Giampaoli, F. Linkowsky, A. Salim, and V. Teofilo, "Advanced Lithium Ion Battery Charger," in *Proc. of BCAA*, 2000.

[4] J. Cao, N. Schofield, and A. Emadi, "Battery Balancing Methods: A Comprehensive Review," in *Proc. of VPPC*, 2008.

[5] N. H. Kutkut, "A Modular Nondissipative Current Diverter for EV Battery Charge Equalization," in *Proc. of APEC*, 1998.

[6] F. Baronti, R. Roncella, and R. Saletti, "Performance Comparison of Active Balancing Techniques for Lithium-ion Batteries," *Journal of Power Sources*, vol. 267, pp. 603–609, 2014.

[7] M. Kauer, S. Naranayaswami, S. Steinhorst, M. Lukasiewycz, S. Chakraborty, and L. Hedrich, "Modular System-Level Architecture for Concurrent Cell Balancing," in *Proc. of DAC*, 2013.

[8] M. Chen and G. A. Rincon-Mora, "Accurate Electrical Battery Model Capable of Predicting Runtime and IV Performance," *IEEE Transactions on Energy Conversion*, vol. 21, no. 2, pp. 504–511, 2006.

[9] M. Kauer, S. Narayanaswamy, S. Steinhorst, M. Lukasiewycz, and S. Chakraborty, "Many-to-Many Active Cell Balancing Strategy Design," in *Proc. of ASP-DAC*, 2015.

[10] H. Rahimi-Eichi, U. Ojha, F. Baronti, and M. Chow, "Battery Management System: An Overview of Its Application in the Smart Grid and Electric Vehicles," *IEEE Industrial Electronics Magazine*, vol. 7, no. 2, pp. 4–16, 2013.

[11] P. Bogacki and L. F. Shampine, "A 3(2) pair of Runge - Kutta Formulas," *Applied Mathematics Letters*, vol. 2, no. 4, pp. 321–325, 1989.

[12] J. R. Dormand and P. J. Prince, "A Family of Embedded Runge-Kutta Formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[13] J. R. Cash and A. H. Karp, "A Variable Order Runge-Kutta Method for Initial Value Problems with Rapidly Varying Right-Hand Sides," *ACM Transactions on Mathematical Software*, vol. 16, no. 3, pp. 201–222, 1990.

[14] K. Ahnert and M. Mulansky, "Odeint – Solving Ordinary Differential Equations in C++," in *Proc. of AIP*, 2011.

Copyright (c) 2016 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.