Battery State-of-Charge Estimator Design based on The Least-Square Support Vector Machine

Luca Cavanini, Pawel Majecki, Mike J. Grimble, Gerrit M. van der Molen

***Abstract*— The design and development of a data-driven algorithm for battery State-of-Charge estimation is presented. The estimation of battery SoC is important in the development of Battery Management Systems. The proposed approach exploits the Least-Squares Support Vector Machine data-driven estimation paradigm and statistical methods. The algorithm's computational complexity is reduced by using a data pruning procedure. The approach is validated using a simulation model of the battery and an Estimator Design Tool in MATLAB software which provides a user-friendly interface for the different algorithms that may be used in the estimator design. The approach is applicable to a wide range of applications.**

***Keywords:* Battery Management System, Least-Squares Support Vector Machine, State-of-Charge Estimator.**

I. Introduction

Hybrid vehicles with electric power zero-emissions powertrains have been widely studied in the automotive systems. These vehicles involve battery energy storage supplying the vehicle powertrain (in pure battery electric vehicles) in conjunction with other power sources e.g., Fuel Cells (in some hybrid electric vehicles) [1]. The green revolution highlights the importance of batteries and related battery management systems (BMSs) in automotive and in other transport applications [2]. A modern BMS is has several control layers for different tasks, e.g. battery cell balancing [3].

The development of the estimators is a critical aspect of BMS design. These algorithms must compute the estimate of battery characteristics that are unmeasurable, e.g. for the battery equivalent capacity; the battery resistance, or the battery State-of-Charge (SoC) [4]. Several methods can be used for developing these estimators. Voltage-based and Current-based are the classical estimation methods used. Among advanced techniques, Kalman Filter (KF)-based methods have been widely used for BMS estimator design [5].

The limitations of these techniques are related to the dependency of the estimator performance on the ability of the user to specify the model structure and parameters for the actual battery system. Due to the large variety of battery technologies, architectures, and the uncertainties in the system parameters, the design of a BMS estimator is a complex and time-consuming task [6]. Several data-driven techniques and combined data-driven and model-based methods have been proposed in the last few years to solve the BMS SoC estimation problem. These papers considered Neural Network (NN)-based estimators, Fuzzy Logic inspired solutions, and KF-based algorithms, combining ML and models-based approaches [7].

In the following a purely data-driven design methodology is proposed to perform general battery estimation and the focus is on the battery State-of-Charge (SoC) estimation problem. It combines the modern data-driven Least Squares Support Vector Machine (LS-SVM) paradigm with a pruning procedure for more efficient computations. In addition, it uses the Particle Swarm Optimization (PSO) method for tuning the algorithm. The LS-SVM is used in a Supervised Learning ML framework for developing the algorithm [8]. The Pruning method selects the most important samples from the original training dataset. This reduces the computational complexity and the memory footprint of the algorithm whilst maintaining satisfactory estimation performance [9].

The dataset considered has been collected by emulating a real battery using an Enhanced Self-Correcting (ESC) battery simulation model. A baseline estimator has also been developed for comparison with a model-based approach. This is an Extended Kalman Filter (EKF) based on the battery ESC model. The estimators were compared in a simulation, in terms of performance and computational complexity, using a set of computed performance indices.

The paper is structured as follows. Section II describes the battery model and related dataset, Section III presents the baseline and data-driven estimation methods, Section IV reports test results, and Section V concludes the paper.

II. Battery Model and Dataset

The battery State-of-Charge model, and the dataset collected from the simulated system are now presented.

A. Battery State-of-Charge

The SoC of the *i*-th battery cell is defined as:

$$SoC_i(k) = \frac{\theta(k)-\theta_{0\%}}{\theta_{100\%}-\theta_{0\%}} \quad (1)$$

where $\theta(k)$ is the average lithium concentration stoichiometry at a discrete time k defined as:

$$\theta(k) = \frac{c_{s,avg,k}}{c_{s,max}} \quad (2)$$

This is intended to remain between 0% and 100%, although it is possible to violate these limits in an over-discharge or over-charge situation [11]. The issue here is that there is

Luca Cavanini is with Industrial Systems and Control Ltd, Glasgow, UK (e-mail: l.cavanini@isc-ltd.com) and Università Politecnica delle Marche, Ancona, Italy (corresponding author, phone: 0039-071-2204314; fax: 0039-071-2204315; e-mail: l.cavanini@univpm.it).

Pawel Majecki, Mike J. Grimble and Gerrit M. van der Molen are with with Industrial Systems and Control Ltd, Glasgow, UK (e-mail: {pawel, m.grimble, gerrit}@isc-ltd.com). Mike J. Grimble is also with University of Strathclyde, Glasgow, UK (e-mail: m.grimble@isc-ltd.com).

presently no direct way to measure the concentrations that would allow us to calculate the stoichiometries and from them the SoC. It is therefore necessary to infer or estimate SoC using only the measurements of cell terminal voltage, current, and temperature. Although the cell Open-Circuit Voltage (OCV) is closely related to the state of charge, the terminal voltage under load is a poor predictor of open-circuit voltage unless the cell is in electrochemical equilibrium (and hysteresis is negligible).

B. Battery Model

A high-fidelity model of a battery cell, based on reference [12], has been used to simulate a real battery and generate the dataset required for the data-driven algorithm training. This is the ESC model, referred to above. It represents the OCV as a function of SoC, linear polarization, diffusion voltage, SoC-varying hysteresis, and instantaneous hysteresis [13]. The ESC model shown in Fig.1 is described in the following.

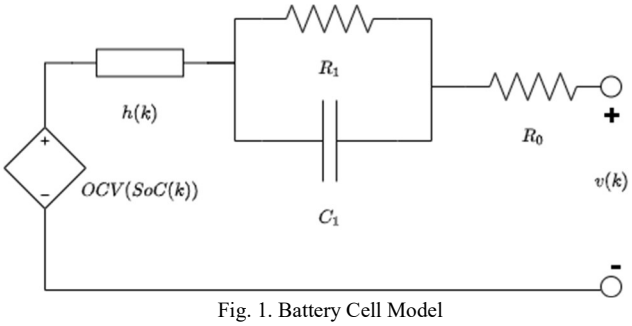

Fig. 1. Battery Cell Model

The battery SoC is modelled as:

$$SoC(k+1) = SoC(k) - \eta(k)i(k)\Delta t/Q \quad (3)$$

where $\eta(k)$ is the unitless cell coulombic efficiency at time k, $i(k)$ is the input current at time k, Δt is the sample period, and Q is the cell's total capacity. SoC is unitless, so e.g. if $i(k)$ is measured in amperes and Δt is measured in seconds, then Q must be expressed in ampere-seconds. The Diffusion-Resistor current is the current flowing through the resistor R_1 in the resistor-capacitor network and is modelled as:

$$i_{R_1}(k+1) = e^{\left(-\frac{\Delta t}{R_1 C_1}\right)} i_{R_1}(k) + \left(1 - e^{\left(-\frac{\Delta t}{R_1 C_1}\right)}\right) i(k) \quad (4)$$

This model captures the slow time constants of diffusion processes occurring within the cell. The hysteresis voltage is modelled as a hysteresis function h

$$h(k+1) = a_h(k)h(k) + (a_h(k) - 1)sgn\big(i(k)\big) \quad (5)$$

where $a_h(k) = e^{\left(-\left|\frac{\eta(k)i(k)\gamma\Delta t}{Q}\right|\right)}$. In this equation, the constant γ adjusts how quickly the hysteresis state changes with a change in cell *SoC* and $sgn\big(i(k)\big)$ is 1 if its input is positive, −1 if negative, and 0 otherwise. This ESC model describes the SoC evolution and all dynamic effects. The ESC output equation computes the voltage $v(k)$ at discrete-time index k as:

$$v(k) = OCV\big(SoC(k)\big) + Mh(k) + M_0 s(k) + \\ -\sum R_i i_{R_i}(k) - R_0 i(k) \quad (6)$$

where $OCV(SoC(k))$ is the OCV as a function of SoC, M is the maximum absolute analog hysteresis voltage at this temperature, M_0 is the instantaneous hysteresis voltage, and R_0 is the pure ohmic resistance of the cell. The battery parameters are given in Table 1 below [13].

C. Battery Dataset

The training data was generated from the ESC model and consisted of the battery cell voltage v, current i, and the SoC output signal. These signals are shown in Figure 2. The scenario represents a battery cell discharging from an initial value of $SoC(t_0) = 100\%$ to $SoC(t_f) = 0\%$. The dataset contains 3700 data points.

Table. 1. Battery Cell Model Specifications

Parameter	*Value*	*Unit*
C_1	38	kF
R_1	0.0158	Ω
R_0	0.0082	Ω

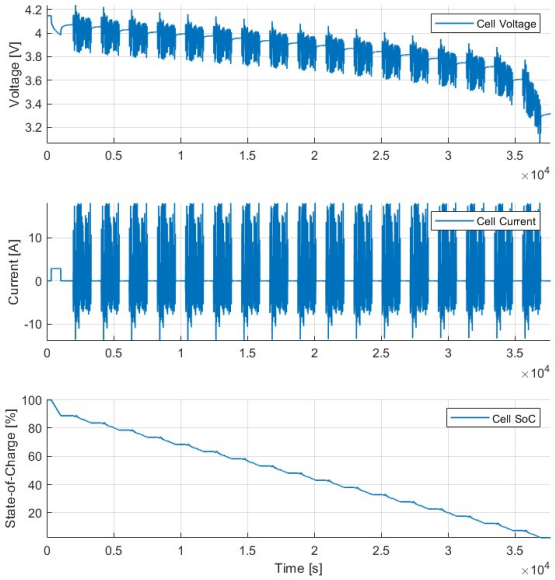

Fig. 2. Dataset signals: battery cell voltage (top), battery current (center), and SoC (bottom)

III. Battery State-of-Charge Estimators

The proposed data driven SoC estimator design approach is presented below and the baseline estimation algorithm for comparison purposes is described.

A. Least-Squares Support Vector Machine

The LS-SVM is derived from a standard SVM and is often used for optimal control of nonlinear Karsh-Kuhn-Tucker (KKT) systems for classification and regression. Given a set of data $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$, with $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$, the LS-SVM finds a nonlinear regression function:

$$y(x) = w^T \phi(x) + b \quad (7)$$

by solving the optimization problem:

$$\min J(w, \xi)_{w,\xi} = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{k=1}^{N} \xi_k{}^2 \quad (8)$$

such that

$$y(x_i) = w^T \phi(x_i) + b + \xi_i. \quad (9)$$

This formulation of the problem consists of equality constraints instead of inequality constraints, such that the related Lagrange function is defined as:

$$L(w, b, \xi, \alpha) = J(w, b, \xi) - \sum_{i=1}^{N} \alpha_i \{w^T \phi(x_i) + b - y_i + \xi_i\} \quad (10)$$

where the Lagrange multipliers α_i represent the solution of the dual problem and can be computed by solving the following system of equations:

$$\frac{\partial L}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{N} \alpha_i \phi(x_i) \quad (11)$$

$$\frac{\partial L}{\partial b} = 0 \rightarrow \sum_{i=1}^{N} \alpha_i = 0 \quad (12)$$

$$\frac{\partial L}{\partial \xi_i} = 0 \rightarrow \gamma - \alpha_i = 0 \quad (13)$$

$$\frac{\partial L}{\partial \alpha_i} = 0 \rightarrow w^T \phi(x_i) + b - y_i + \xi_i = 0 \quad (14)$$

The solution of the system as in Eq. (11) - Eq. (14) is given by the following matrix equation:

$$\left(\Omega + \frac{1}{\gamma} \mathrm{I}\right) \alpha = Y \quad (15)$$

with $Y = [y_1, \ldots, y_N]'$, $\alpha = [\alpha_1, \ldots, \alpha_N]'$, and where the kernel matrix Ω entries are computed by

$$\Omega_{i,j} = y_i y_j \phi(x_i)^T \phi(x_j) = y_i y_j K(x_i, x_j) \quad (16)$$

The fitting function, representing the LS-SVM regression output is then:

$$y(x) = \sum_{i=1}^{N} \alpha_i K(x_i, x) \quad (17)$$

The Lagrangian multipliers α_i are the solution of the linear system Eq. (17), and $K(x_i, x)$ is the selected kernel function.

B. Pruning Procedure

One of the main issues in LS-SVM identification is the size of the training dataset. The estimate is computed iteratively by comparing the training dataset information with the measurements from the real system. There are two issues: (i) The computational burden increases with the dataset size and when the training dataset is large, the iterative computation of the estimate could be prohibitive in real-time or even for simulation purposes [8]; (ii) To use the LS-SVM on real systems, the training data must be stored in memory, reducing the possibility of porting the algorithm onto hardware with limited resources [9]. Because the LS-SVM is a kernel method, the most significant features of the training dataset cannot be selected a priori, but can be only evaluated after the training of the algorithm. A possible solution for reducing the training dataset size, reducing computational burden and data storage memory, is given by the so-called *pruning method.* This method involves iteratively performing a LS-SVM identification, reducing the size of the training dataset at each iteration, by gradually omitting the training data related to the less significant Lagrangian multipliers. This method allows one to define a priori the maximum size of the data subset to consider, or equivalently, the acceptable value of the identification performance degradation [14].

The pruning procedure is performed by the following steps:

(1). Considering the original dataset of size N, train the LS-SVM.

(2). Remove small number of points (e.g., $\Delta N = 5\%$ of N) corresponding to the smallest values in the $|\alpha_k|$ spectrum.

(3). Train the LS-SVM with the new reduced dataset. Go to point 2 until the identification performance degradation threshold is exceeded.

C. Particle Swarm Optimization

The Particle Swarm Optimization (PSO) is a global optimization algorithm, inspired by birds' flocking or fish schooling for the solution of nonlinear, nonconvex, or combinatorial optimization problems [10]. The PSO optimization uses evolutionary techniques for finding the global minimum of a function. The solution to the optimization problem is obtained through a random search equipped with swarm intelligence. The initial ideas on particle swarms of Kennedy and Eberhart were aimed at using computational intelligence by exploiting simple analogues of social interaction, rather than purely individual cognitive abilities [10]. In this problem PSO is used to optimize the tuning parameters that calibrate the LS-SVM trained on the selected dataset. In this approach several simple entities (the particles), are placed in the search space of some problem or function, and each evaluates the objective function at its current location. Every particle in a swarm has three N-dimensional vectors, where N is the dimensionality of the search space. These are the current position x_i, the previous best position p_i, and the velocity v_i. The current position $x_i$can be considered as a set of coordinates describing a point in space. On each iteration of the algorithm, the current position is evaluated as a problem solution. If that position is better than any that has been found so far, then the coordinates are stored in the second vector, p_i. The value of the best function result so far is stored in a variable that can be called $p_{pb,i}$, for comparison on later iterations. The goal is to keep finding better positions and updating p_i and $p_{pb,i}$. New points are chosen by adding v_i coordinates to x_i, and the algorithm operates by adjusting v_i, which can effectively be seen as a step size. The structure of the PSO algorithm is described by the following steps:

(1). The population is initialized with random initial positions and velocities on the research space of dimension N.

(2). The desired fitness function in N variables is evaluated for each particle.

(3). Compare the fitness function values of each particle in the swarm to find the best particle.

(4). Identify the particles in the neighborhood of the best particle by indexing the swarm.

(5). Change the velocity and position of each particle according to the updated law.

(6). Check the stopping criteria: this stops the algorithm if satisfied or iterates from step 2 if not satisfied.

The basic PSO described above has a small number of parameters that need to be fixed. One parameter is the size of the population. This is often set empirically based on the dimensionality and the perceived difficulty of a problem. The role of the PSO is to automatically compute the best set of calibration parameters for the LS-SVM to minimize the estimated SoC tracking error.

D. Data-driven Estimator Design Procedure

The data-driven estimator design is based on ML and statistical methods. To limit the computational complexity, the following design procedure was defined:

- Phase 1: Perform a first LS-SVM training based on PSO and the original training dataset to compute estimator calibration parameters.
- Phase 2: Perform the Pruning Analysis with the calibration parameters computed in Phase 1 and select the size of the pruned dataset to be considered for the final estimator training.
- Phase 3: Perform the final LS-SVM training.

The proposed design approach has been undertaken using design software presented in the following section.

E. Data-driven Estimator Design Tool

The design approach proposed in the previous section is implemented in a Data-driven Estimator Design Tool developed by Industrial Systems and Control Ltd., to simplify the time-consuming development process. The software generates the data-driven estimator by exploiting the capabilities of ML techniques. The Graphical User Interface of the software is shown in Figure 3. The Design Tool allows the adjustment of the tuning parameters for the LS-SVM training and the pruning procedure. The GUI provides feedback to the user in terms of estimation/pruning graphical results and identification of performance statistics. As shown in the GUI figure, the Design Tool includes a set of common data manipulation methods e.g., data normalization or Entropy-based Analysis.

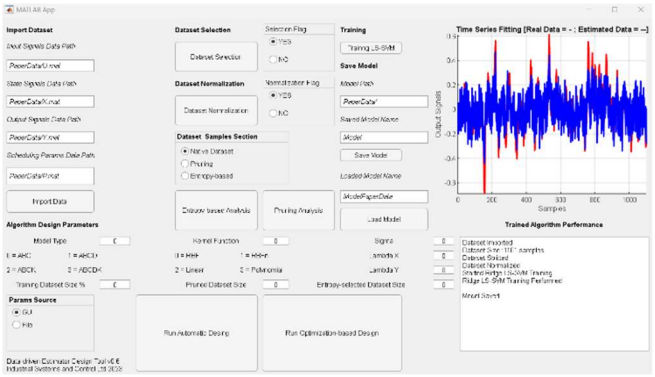

Fig. 3. Data-Driven Estimator Design Tool GUI

F. Baseline Estimator

An Extended Kalman Filter (EKF) is the baseline estimator to be used for comparison purposes. The first step for developing an EKF for battery SoC estimation is to define the state-space matrices locally describing the model over instantaneous operating conditions. Suppose that the process noise represents the current-sensor measurement error and that the true cell current is $i_k + w_k$, but that we measure i_k only. Also, assume we can simplify the model with coulombic efficiency $\eta_k = 1$, and allow the adaptivity of the EKF to handle the small error introduced by this assumption. Using the notation $SoC_k = SoC(k)$, the SoC can be written as:

$$SoC_{k+1} = SoC_k - \frac{\Delta t}{Q}(i_k + w_k) \tag{18}$$

and the two derivatives needed are:

$$\frac{\partial SoC_{k+1}}{\partial SoC_k}\Big|_{SoC_k=SoC_k^*} = 1; \quad \frac{\partial So_{\ k+1}}{\partial w_k}\Big|_{SoC_k=SoC_k^*} = -\frac{\Delta t}{Q} \tag{19}$$

By defining $\tau_j = e^{\left(\frac{\Delta t}{R_j C_j}\right)}$ then:

$$i_{R,k+1} = \begin{bmatrix} \tau_1 & 0 & 0 \\ 0 & \tau_{\ 2} & 0 \\ 0 & 0 & \dots \end{bmatrix} i_{R,k} + \begin{bmatrix} 1-\tau_1 \\ 1-\tau_2 \\ \dots \end{bmatrix}(i_k + w_k) \tag{20}$$

Then the state equation matrices are computed as:

$$\frac{\partial i_{R,k+1}}{\partial R_k}\Big|_{R_k=R_k^*} = A_{RC} \quad ; \quad \frac{\partial i_{R,k+1}}{\partial w_k}\Big|_{w_k=w_k^*} = B_{RC} \tag{21}$$

By defining $A_{H,k} = e^{\left(-\frac{(i_k+w_k)\gamma\Delta t}{Q}\right)}$ then the hysteresis state equation can be written as:

$$h_{k+1} = A_{H,k}h_k + (A_{H,k} - 1)sgn(i_k + w_k). \tag{22}$$

By taking the partial derivative with respect to the state and evaluating it at the setpoint p_k

$$\frac{\partial h_{k+1}}{\partial h_k}\Big|_{h_k=h_k^*} = e^{\left(-\frac{(i_k+w_k)\gamma\Delta t}{Q}\right)} = \bar{A}_{H,k} \tag{23}$$

We find $\frac{\partial h_{k+1}}{\partial w_k}$ as follows:

- If $(i_k + w_k) > 0$ then

$$\frac{\partial h_{k+1}}{\partial w_k}\Big|_{h_k=h_k^*} = -\left|\frac{\gamma\Delta t}{Q}\right| e^{\left(-\left|\frac{\gamma\Delta t}{Q}\right||(i_k+w_k)|\right)}(1 + h_k) \tag{24}$$

- If $(i_k + w_k) < 0$ then:

$$\frac{\partial h_{k+1}}{\partial w_k}\Big|_{h_k=h_k^*} = -\left|\frac{\gamma\Delta t}{Q}\right| e^{\left(-\left|\frac{\gamma\Delta t}{Q}\right||(i_k+w_k)|\right)}(1 - h_k) \tag{25}$$

While evaluating the Taylor-series linearization at the setpoint it may be assumed that the following generalization is reasonable for all $(i_k + w_k)$

$$\frac{\partial h_{k+1}}{\partial w_k}\Big|_{\substack{h_k=h_k^* \\ w_k=w_k^*}} = -\left|\frac{\gamma\Delta t}{Q}\right| \bar{A}_{H,k}(1 + sgn(i_k + w^*)\hat{h}_k^+) \tag{26}$$

The zero-state hysteresis equation is defined as:

- If $|i_k + w_k| > 0$ then $s_{k+1} = sgn(i_k + w_k)$
- Else $s_{k+1} = s_k$.

If we consider $i_k + w_k = 0$ to be a zero-probability event, then $\frac{\partial s_{k+1}}{\partial s_k} = 0$ and $\frac{\partial s_{k+1}}{\partial w_k} = 0$. The ESC-model output is:

$$y_k = OCV(SoC_k) + Mh_k + M_0 s_k + \sum_j R_j i_{j,k} + R_0 i_k + v_k \tag{27}$$

We can neglect the noise w_k previously added to i_k because this would add correlation between process noise and the overall noise present in the measurement, which violates an assumption made when deriving the Kalman filter. Then,

$$\frac{\partial y_k}{\partial s_k} = M_0 \, ; \frac{\partial y_k}{\partial h_k} = M \, ; \; \frac{\partial y_k}{\partial i_{j,k}} = -R_j \, ; \frac{\partial y_k}{\partial v_k} = 1 \tag{28}$$

and

$$\frac{\partial y_k}{\partial z_k}\Big|_{z_k=z_k^*} = \frac{\partial OCV(z_k)}{\partial z_k}\Big|_{z_k=z_k^*} \tag{29}$$

which can be approximated from OCV data.

G. Performance Criteria

The following performance indices were considered to compare the different estimators:

- **Mean Absolute Error** (MAE) computed as:

$$MAE = \sum_{i=1\dots n} \frac{|x(k) - \hat{x}(k)|}{n} \tag{30}$$

where $x(k)$ is the real value to estimate, $\hat{x}(k)$ is the estimated signal samples provided by the estimator and n is the number of samples.

- **Root Mean Square Error** (RMSE) computed as:

$$RMSE = \sqrt{\frac{\sum_{k=1}^{n}(x(k)-\hat{x}(k))^2}{n}} \quad (31)$$

where $x(k)$ is the real value to estimate, $\hat{x}(k)$ is the estimated signal samples provided by the estimator.

- **Execution Time** (ET) is the time needed for performing the computation of the estimation algorithm in seconds.

IV. Simulation Results

In this section, the estimation algorithms are compared by evaluating the performance criteria for the scenario presented in Section II.*B*. The parameters of the baseline EKF are included in Table 2 whereas the data-driven policy design is further described in the following sections. Initially, the original dataset was used and 7200 samples were selected for estimator training. The original dataset, with 625 minutes of measurements, was then used to evaluate the performance of the algorithms in terms of MAE and RMSE.

Table. 2. Extended Kalman Filter Parameters

Parameter	***Value***
σ_X	$[10^{-6}, 10^{-8}, 2 \times 10^{-4}]$
σ_W	0.2
σ_V	0.2

A. Data-driven estimator design: Phase 1

The first step of the ML-based estimator design is the selection of the initial parameters determining the algorithm. The first SoC estimator is designed by considering 75% of this initial dataset, composed of 7200 features, as the training set and the remaining 25% as the validation dataset. The SVM uses Radial Basis activation functions (RBFs) K to define the Kernel matrix entries $\Omega_{i,j}$ as in Eq. (16) such that

$$K^i(p(k), p(i)) = e^{\left(-\frac{||p(k)-p(j)||_2^2}{\sigma_i^2}\right)} \quad (32)$$

$$\Omega_{i,j} = y_i y_j K\left(x_i, x_j\right) = y_i y_j e^{\left(-\frac{||x(i)-x(j)||_2^2}{\sigma_i^2}\right)} \quad (33)$$

and the values of the tuning parameters σ_i for the identification algorithm are selected by the PSO approach. The goal of this design step is to compute the set of tuning parameters permitting to optimize the estimation performance while considering the full original dataset.

Table. 3. Data-driven estimator design: Phase 1 Optimized Parameters

Parameter	***Value***
Sigma Values	[1306.7763, 8218.1091, 7447.8777, 17.7607, 4626.9248]
LambdaX Values	[5237.4766, 1009.6543, 8388.2786, 152.8805]
LambdaY Values	[7185.4658, 152.8805]

The set of parameters in Table 3 shows the settings obtained by using the PSO for tuning the LS-SVM estimator. The performance achieved by the data-driven estimator in Table 4, shows the estimation indices found by testing the algorithm on the initial complete dataset presented in Section II together with the averaged execution time evaluated during this test.

Table. 4. Data-driven estimator design: Phase 1 Estimation Performance

Performance Index	***Value***
RMSE	4.432×10^{-3}
MAE	2.18×10^{-3}
ET [s]	6.05×10^{-5}

B. Data-driven estimator design: Phase 2

In the second stage of the design, the pruning procedure was applied to the dataset using the optimized settings computed in Phase 1. The result shown in Figure 4 reveals the average fitting error with respect to size of the selected dataset, together with the normalized averaged execution time.

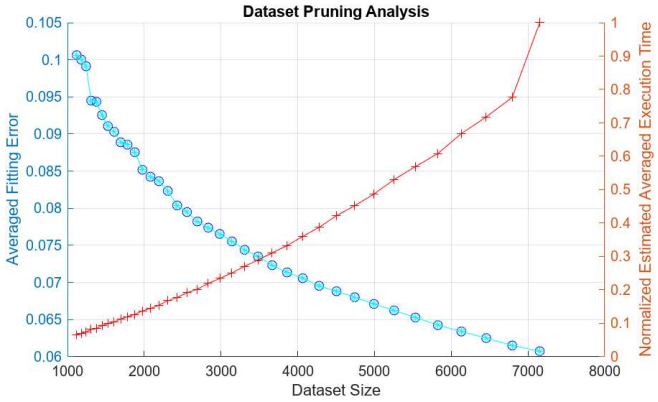

Fig. 4. Data-driven estimator design: Phase 2 Pruning Analysis

This analysis of results enables one to evaluate the trade-off between fitting error increase and related execution time reduction. From this analysis, the size of the pruned dataset to be used in the following Phase 3 is selected to be composed of 5000 samples. This should reduce the computational burden by 50%, with an expected 16% increase of the fitting error.

C. Data-driven estimator design: Phase 3

In the third and final stage of development of the data-driven estimator the selected subset of data was considered for performing a refinement of the calibration parameters of the LS-SVM by the PSO approach. By adjusting the estimator calibration parameters, it is possible to improve the performance of the algorithm with respect to the expected performance estimated in the previous Phase 2 during the pruning analysis. On the other hand, because only the dataset size affects the computational complexity of the algorithm (and related memory footprint), the expected complexity of the developed algorithm is the same as in Phase 2.

The optimized parameters are given in Table 5 and the performance indices of this refined data-driven estimator evaluated with respect to the initial dataset in Section II are given in Table 6.

Table. 5. Data-driven estimator design: Phase 3 Optimized Parameters

Parameter	*Value*
Sigma Values	[8356. 1163, 8832.2549, 8305.5998, 267.1314, 2053.1127]
LambdaX Values	9792.1244
LambdaY Values	9638.2447

After the application of the pruning procedure and the PSO-based refinement, the data-driven estimator reduces the ET by 58%, in line with the pruning analysis. In terms of performance, this refined estimator shows a slight reduction of the estimation performance by 11% in terms of RMSE and by 17% for the MAE index.

Table. 6. Data-driven estimator design: Phase 3 Estimation Performance

Performance Index	*Value*
RMSE	4.7922×10^{-3}
MAE	2.9855×10^{-3}
ET [s]	2.55×10^{-5}

D. Performance Comparison

In this section, the estimation performance of the data-driven estimator is compared against that of the baseline EKF. Figure 5 and Figure 6 show the estimators' performance for the scenario considered (with the SoC varying from 100% to 50%, and from 50% to 0%, respectively), and Table 5 collects the related performance indices. Both images refer to the same test, split into two different figures to increase the readability of the results.

As shown in Figure 5, in the first half of the test the data-driven estimator provides superior performance whereas the EKF tends to perform better in the second part of the scenario, as shown in Figure 6. The performance provided over the complete scenario given in Table 7 shows how the data-driven estimator gives an improvement of 7% in terms of RMSE and 39% for the MAE, compared with the EKF. This is possible due to the capability of data driven methods to learn and reflect real-world system behavior overcoming the need of an accurate model of the plant.

V. CONCLUSIONS

The problem of battery State-of-Charge (SoC) estimation was considered, and an estimation method developed exploiting the capabilities of data-driven Machine Learning (ML) techniques. The proposed approach can be used for more general applications. It combines a Least-Squares Support Vector Machine (LS-SVM) identification technique with a Pruning Dataset Selection procedure and uses the meta-heuristic Particle Swarm Optimization (PSO) method. An automatic procedure that involves combining these techniques has been developed and implemented within the Data-driven Estimator Design Tool. This was developed to reduce effort in the battery estimation studies. The estimator approach achieves good performance with limited computational complexity. The performance of the algorithm was compared against a baseline EKF estimator. The results demonstrate the ability of the approach to develop an estimator able to overcome some of the limitations of model-based methods whilst considering the computational burden. Future research will consider the integration of an online adaptation procedure within the LS-SVM policy, and application trials of the estimator and design tool.

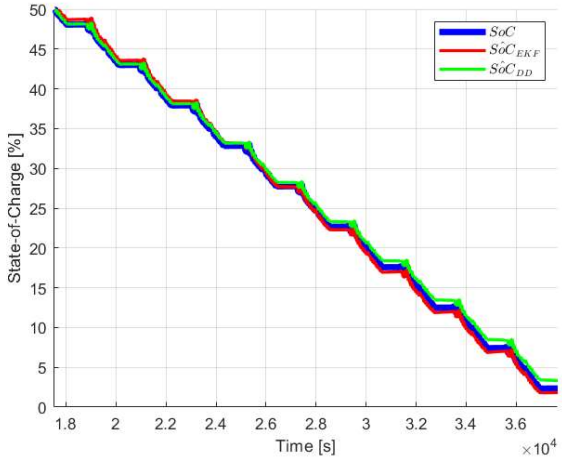

Fig. 5. Estimators' performance comparison: SoC interval [0, 50] %

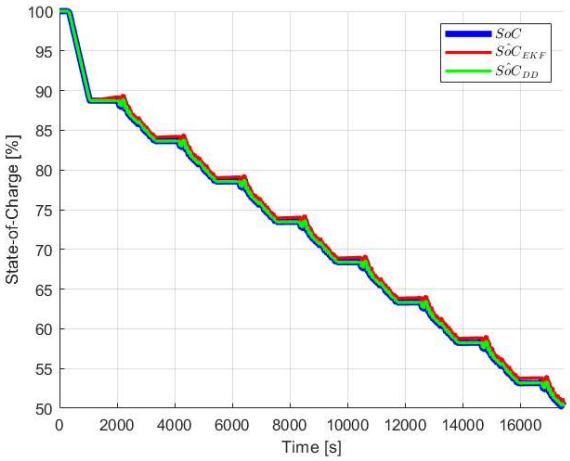

Fig. 6. Estimators' performance comparison: SoC interval [50, 100] %

Table. 7. Estimation Performance Comparison

Estimator	*RMSE*	*MAE*
EKF	5.1049×10^{-3}	4.8496×10^{-3}
LS-SVM	4.7922×10^{-3}	2.9855×10^{-3}

REFERENCES

[1] Tran, D. D., Vafaeipour, M., El Baghdadi, M., Barrero, R., Van Mierlo, J., & Hegazy, O. , "Thorough state-of-the-art analysis of electric and hybrid vehicle powertrains: Topologies and integrated energy management strategies.," *Renewable and Sustainable Energy Reviews,* 2020.

[2] Wang, Y., Tian, J., Sun, Z., Wang, L., Xu, R., Li, M., & Chen, Z., "A comprehensive review of battery modeling and state estimation approaches for advanced battery management systems.," *Renewable and Sustainable Energy Reviews,* 2020.

[3] Gabbar, H. A., Othman, A. M., & Abdussami, M. R., "Review of battery management systems (BMS) development and industrial standards," *Technologies,* 2021.

[4] Ali, M. U., Zafar, A., Nengroo, S. H., Hussain, S., Junaid Alvi, M., & Kim, H. J. , “owards a smarter battery management system for electric vehicle applications: A critical review of lithium-ion battery state of charge estimation,” *Energies,* 2019.

[5] Shrivastava, P., Soon, T. K., Idris, M. Y. I. B., & Mekhilef, S., “Overview of model-based online state-of-charge estimation using Kalman filter family for lithium-ion batteries,” *Renewable and Sustainable Energy Reviews,* 2019.

[6] Lipu, M. H., Hannan, M. A., Hussain, A., Ayob, A., Saad, M. H., Karim, T. F., & How, D. N. , “ Data-driven state of charge estimation of lithium-ion batteries,” *Algorithms, implementation factors, limitations and future trends,* 2020.

[7] Hu, X., Feng, F., Liu, K., Zhang, L., Xie, J., & Liu, B. , “State estimation for advanced battery management: Key challenges and future trends,” *Renewable and Sustainable Energy Reviews,* 2019.

[8] Cavanini, L., Ferracuti, F., Longhi, S., Marchegiani, E., & Monteriù, A., “Sparse approximation of ls-svm for lpv-arx model identification: Application to a powertrain subsystem,” in *IEEE AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)*, 2020.

[9] Cavanini, L., Ferracuti, F., Longhi, S., & Monteriù, A. , “ Ls-svm for lpv-arx identification: Efficient online update by low-rank matrix approximation,” in *IEEE International conference on unmanned aircraft systems (ICUAS)*, 2020 .

[10] Poli, R., Kennedy, J., & Blackwell, T., “Particle swarm optimization,” *Swarm intelligence,* vol. 1, pp. 33-57, 2007.

[11] Plett, Gregory L., Battery management systems, Volume II: Equivalent-circuit methods, Artech House, 2015.

[12] Plett, Gregory L., “Battery management system algorithms for HEV battery state-of-charge and state-of-health estimation.,” *Advanced materials and methods for lithium-ion batteries,* pp. 1-25, 2007.

[13] Plett, G. L., Battery management systems, Volume I: Battery modeling, Artech House, 2015.

[14] Cavanini, L., Ciabattoni, L., Ferracuti, F., Marchegiani, E., & Monteriù, A., “A comparative study of driver torque demand prediction methods,” *IET Intelligent Transport Systems,* 2022.

Acknowledgements: Industrial Systems and Control Ltd., Glasgow is grateful for the support of the Scottish Enterprise on the Data-driven Estimation for Battery Systems project, and the University of Strathclyde for cooperation.