Applied Energy 371 (2024) 123526

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy

Discrete-time state-of-charge estimator for latent heat thermal energy storage units based on a recurrent neural network

Hector Bastida, Ivan De la Cruz-Loredo, Pranaynil Saikia, Carlos E. Ugalde-Loo *

School of Engineering, Cardiff University, Queen's Buildings, Cardiff, CF24 3AA, Wales, UK

ARTICLE INFO

Dataset link: http://doi.org/10.17035/d.2024.0322710137

Keywords:
Latent heat thermal energy storage
Recurrent neural network
State-of-charge estimator
Long short-term memory
Artificial intelligence

ABSTRACT

Energy storage systems enable balancing supply and demand and facilitate the integration of intermittent renewable energy sources. In particular, latent heat thermal energy storage units are attractive for deployment in thermal systems due to their high energy density. However, knowledge of the state-of-charge of a thermal store is crucial to effectively regulate its charging and discharging cycles. To achieve this, continuous-time non-linear observers may be employed to estimate the state-of-charge at the expense of a high computational cost. The emergence of artificial intelligence solutions may be helpful to reduce computational burden, but their adoption for thermal stores has been limited. This paper bridges this research gap by presenting a novel approach to predict the state-of-charge of a latent heat unit. It employs a discrete-time estimator based on a recurrent neural network, which is based on a long short-term memory structure and the regression method for estimation. The estimator offers a reduced computation time for state-of-charge estimation and allows flexible sampling adjustments without sacrificing accuracy. Additionally, the presented approach simplifies data collection by independently handling charging and discharging processes through internal state resets. The estimator, trained using MATLAB's deep learning toolbox, uses a dataset comprising various operating conditions obtained from simulations of a physics-based model. When compared against a more traditional state-of-charge estimation method using a discrete-time non-linear observer, the advantages of a recurrent neural network-based estimator are evidenced, highlighting its potential for practical applications. The presented method exhibited high accuracy with a maximum root mean square error under 0.73% and a mean absolute error below 0.41% with respect to direct state-of-charge calculation. Although the discrete-time non-linear observer exhibited a marginal higher accuracy, the recurrent neural network-based estimator achieved a significant improvement in computational efficiency. These findings make the proposed approach a robust tool facilitating enhanced control strategies, optimised energy management, and increased overall thermal system performance.

1. Introduction

The importance of energy storage systems has increased in recent times, proving crucial for the enhanced management of energy systems. Energy storage systems help addressing the mismatch between peaks in energy supply and demand. For instance, the intermittent nature of renewable energy sources such as wind and solar energy can be managed to optimise system performance through peak shaving and load shifting mechanisms [1,2].

Among the available thermal energy storage (TES) technologies, sensible heat TES (SHTES) and latent heat TES (LHTES) units are highly effective for heating and cooling systems [3]. Water is predominantly used as the storage medium in sensible heat thermal stores for heat provision. District heating and residential systems integrate water tanks to store hot water during periods of low-cost thermal energy production, such as when a co-generation system produces electricity to meet high loads in the absence of thermal energy demand [4]. The water utilised for energy storage retains its phase (liquid), thus employing only sensible heat to store the thermal energy by increasing the water temperature.

Phase change materials (PCMs) constitute the storage media for LHTES units. In these thermal stores, heat transfer during charging and discharging processes is facilitated by circulating a heat transfer fluid (HTF). A PCM stores energy during its transition between phases [5]. When energy is released, the PCM reverts to its nominal phase. Due to the large amount of energy released or absorbed during phase change, LHTES units in general possess a greater storage capacity compared to

* Corresponding author.
E-mail address: ugalde-looc@cardiff.ac.uk (C.E. Ugalde-Loo).

https://doi.org/10.1016/j.apenergy.2024.123526
Received 28 October 2023; Received in revised form 29 April 2024; Accepted 20 May 2024
Available online 12 June 2024

Nomenclature

Abbreviations			
AI	Artificial intelligence	$\dot{m}_f$	Mass flow rate
HTF	Heat transfer fluid	n	Number of inputs
LHTES	Latent heat thermal energy storage	o	Output gate
LSTM	Long short-term memory	tanh	Hyperbolic tangent
MAE	Mean absolute error	T_f	Temperature of HTF
ODE	Ordinary differential equation	T_p	Temperature of PCM
PCM	Phase change material	U	Weights for hidden state
ReLU	Rectified linear unit layer	W	Weights for input
RMSE	Root mean square error	x	Input vector
RNN	Recurrent neural network	$\mathbf{X}$	Input subset of training profiles
SHTES	Sensible heat thermal energy storage	y	Output vector
TES	Thermal energy storage	$\mathbf{Y}$	Output subset of training profiles
SoC	State-of-charge	$\mathbf{Z}$	Subset of training profiles
ZOH	Zero-order hold	$\odot$	Hadamard product
Greek Symbols		**Subscripts**	
σ	Sigmoid function	a	Ice tank tube 1
		b	Ice tank tube 2
Variables and mathematical functions		D	Dense layer
a	Output vector layer	f	Refers to forget input
b	Bias vector	g	Number of layers
c	Cell state vector	h	Refers to hidden state
$\tilde{c}$	Cell state candidate vector	i	Refers to input gate
f	Forget gate	in	Input
g	Activation functions	o	Refers to output gate
h	Hidden state vector	p	Refers to phase change material
i	Input gate	R	Refers to ReLU layer
k	Number of outputs	S	Refers to ice tank
m	Number of neurons	t	Time-step number

their sensible heat counterparts [6]. The storage capacity is determined by the specific latent heat value of the PCM. Thus, the size of the TES unit could be significantly reduced if an appropriately selected PCM is adopted. Additionally, the melting temperature of the PCM is a critical factor in determining the suitability of a TES unit for a specific application [7]. These characteristics have made the use of TES with PCM very popular in different engineering applications. Relevant examples include solar energy use through solar cookers [8] and cold-chain transportation [9] to name but a few. Other interesting applications are available in the construction sector, where concrete walls integrate PCM into modified concrete [10], wallboards store heat using PCM [11], and ceiling boards integrating PCM facilitate active cooling in renovated buildings [12].

Monitoring the state-of-charge (SoC) is paramount for the optimal and efficient operation of a TES unit regardless of its type. SoC is a dimensionless number defined by the ratio between the instantaneous available energy and the maximum energy stored in a TES unit. If it is accurately quantified in a timely manner, potential overcharging or incomplete discharging of the thermal store can be avoided [13]. For a thermally stratified SHTES tank, SoC is defined by the temperature gradient through its different levels, the specific heat of the storage medium, and its total mass [14]. However, for LHTES units, SoC quantifies the specific latent heat value released or absorbed during charging and discharging [15].

Although a two-dimensional modelling approach was employed in [16] to model an LHTES unit with a high accuracy and Kalman filters were adopted in [17] to effectively estimate its SoC, a decreased model complexity is desirable to facilitate the design of the estimator and reduce its implementation requirements. For instance, the SoC calculation method introduced in [15] relies on a suitable one-dimensional mathematical model of the LHTES unit and the use of a continuous-time non-linear state observer to estimate the temperature distribution of the PCM within the thermal store. A significant advantage of such an approach is the reduced number of measurements required for an effective SoC estimation. For instance, only the mass flow rate and the input and output temperatures of the HTF are needed. However, even for a continuous-time observer based on a simplified model, its implementation is constrained to a high-speed sampling which guarantees that the measurements of the inputs and outputs of the system emulate a continuous-time system [18]. This may require expensive instrumentation and a robust computational processing to solve the observer model. This is significant as the observer is described by non-linear differential equations.

The use of machine learning and artificial intelligence (AI) based algorithms, and in particular recurrent neural networks (RNNs), is an interesting approach for estimating SoC in energy storage systems due to their suitability for predicting time-series data. This capability is evident in highly non-linear dynamic systems such as LHTES units. For instance, an RNN with a feedback loop was adopted in [19] to model a compressor. Such an approach enabled to effectively capture the compressor's transient behaviour. This is a relevant example within the context of LHTES units as the compressor's outputs are its mass flow rates and pressures. The RNN's performance yielded a root mean square error (RMSE) of less than 4% when compared with physical measurements. Refs. [20,21] detail the modelling of pulse propagation in the ultrashort pulse evolution within fibre optical parametric amplifiers—a notably intricate non-linear system. Both models in the references exhibit a high efficacy, with normalised RMSE values of up to 0.12% and 0.026%.

In the field of energy systems, RNN-based architectures for estimating the SoC of electrical batteries have gained attention and have been investigated in the literature. For instance, Ref. [22] presents a detailed formulation for a time-delayed RNN to estimate the SoC of lithium-ion batteries. This method achieves an estimation error of less than

1%. In [23], a modified RNN structure termed 'clockwork RNN' was proposed. This architecture divides hidden layers into distinct modules with varying clock speeds, thereby addressing long-term dependencies and reducing the training and computation costs. The RMSE of the estimated SoC value was less than 1.29% in the reference.

Long short-term memory (LSTM) structures have been employed in RNNs with successful results for estimating SoC of electrical lithium-ion batteries [24,25]. In these references, the estimated SoC was compared using different prediction methods. When compared to an extended Kalman filter, the LSTM-based RNN exhibited a significant reduction of RMSE of approximately nine times under constant current conditions and around seven times under dynamic conditions [24]. The LSTM-based RNN also exhibited a reduced RMSE (1.02%) when compared to that obtained with a simple RNN (1.30%) in [25].

Despite the encouraging results in accurately estimating SoC in electrical batteries, the adoption of AI for predicting the behaviour of TES units has been restricted to a handful of studies. Pioneering work using a feed-forward back propagation artificial neural network (ANN) was presented in [26] to quantify the total energy stored in an LHTES unit during a charging process. The ANN directly estimates the total energy stored in the unit without intermediate calculations from temperatures, thus reducing the complexity of the estimation. However, system parameters and critical values in the calculation of heat transfer such as the heat transfer area and Reynolds number are included as inputs of the ANN. The need for these preliminary calculations before deploying the ANN may result in an inefficient implementation. Moreover, a reliable SoC quantification is not achievable as the method estimates the stored energy at 30-min intervals only. Ref. [27] is of relevance, as the PCM temperature for a heat exchanger-based LHTES unit was successfully predicted using an LSTM-based RNN. However, the approach does not consider calculation of SoC.

More recent works have successfully employed LSTM-based structures for LHTES units. In [28] an LSTM-back propagation neural network to predict the transient melting process in an LHTES tank is presented. The liquid fraction and the average temperature of the PCM, a paraffin wax, are predicted considering the velocity and input temperature of the HTF as inputs while achieving reduced values of RMSE and mean absolute error (MAE) (0.0050% and 0.0042%). Although the results are promising, the use of this LSTM-based structure, as in [27], was restricted to temperature monitoring only.

This paper presents a novel SoC estimator for LHTES units underpinned by AI methodologies. An in-depth examination of the requirements and characteristics of the approach is carried out by addressing the following research questions:

- Is it feasible to estimate SoC of an LHTES unit using an LSTM-based RNN architecture and what are requirements of this approach in terms of training datasets and structure?
- What is the effect of sampling time on the accuracy, training times, and computational cost of the LSTM-based RNN SoC estimator?
- What are the advantages of using an LSTM-based RNN architecture for SoC estimation of an LHTES unit over a conventional state observer?

In the presented approach, the output of the SoC estimator serves as feedback for the subsequent time-step prediction. Then, the previous estimation value is used to generate the subsequent estimation. These outputs are generated at a predefined time (depending on the sampling time), making the proposed method a discrete-time estimator. The presented LSTM-based RNN architecture relieves the need for highly demanding computational resources and small sampling times required by continuous-time observers as it relies on basic matrix operations and moderate sampling times. SoC is directly estimated by the RNN, which further reduces complexity by avoiding any intermediate calculations from PCM temperatures to determine the energy stored by the LHTES unit. This facilitates an easy implementation into a basic microcontroller. Training of the neural network was conducted with the machine learning toolbox available in MATLAB R2023a. To this end, simulation output data of the validated physics-based model of the LHTES unit presented in [29] were adopted.

Via simulations conducted in MATLAB/Simulink, the performance of the RNN estimator was compared against that of a discrete-time non-linear observer suitable for practical implementation. Specifically, the RNN estimator exhibited a good accuracy, with RMSEs kept under 0.73%, and MAEs below 0.41% with respect to direct SoC calculation. The RNN estimator significantly outperformed the discrete-time observer in terms of computational resources, as the computation time for estimation observed a significant drop, from 284 μs to 12 μs, marking a nearly 24-fold acceleration. The estimation efficiency enabled amplifying the sampling interval from 0.5 s to 300 s (600-fold) without compromising accuracy. In turn, due to such higher sampling time adopted, the RNN estimator required only 12 estimations per hour of simulation—a considerable reduction compared to the 7200 estimations per hour performed by the observer. These key results underscore the RNN estimator's robustness and its potential for improving the operation of LHTES units.

The key contributions of the paper to the state-of-the art are summarised as follows:

- The introduction of a novel AI-based methodology for directly estimating SoC of an LHTES unit through the use of an RNN based on LSTM layers. This approach notably decreases the computation time compared to continuous-time non-linear observers reported in the literature. This effect stems from the primary use of matrix operations instead of ordinary differential equations (ODEs). To the best of the authors' knowledge, the use of such an AI architecture for estimating SoC of LHTES units has not been previously reported in the literature.
- As opposed to continuous-time non-linear observers, the sampling time to estimate SoC using the proposed RNN estimator can be increased without compromising accuracy. Although the discrete-time version of a non-linear observer significantly reduces the complexity of implementation by avoiding the use of an ODE engine solver, it requires concurrently solving several algebraic equations at the same sampling time to ensure convergence. This in turn demands significant computational resources.
- The use of the internal states reset (for hidden and cell states) of LSTM layers to improve the performance of the prediction. This novel approach for SoC estimation enables simplifying the collection of training data by considering charging and discharging processes independently. The trained RNN estimator is applied for charging–discharging cycles, with a reset of these states being the only requirement to achieve a good performance.

2. Description of the LHTES unit under study

2.1. Ice tank for cooling provision

The LHTES unit adopted for this work is the commercial 1098C ice tank from CALMAC [30]. A schematic of the unit is shown in Fig. 1. It has a nominal storage capacity of 350 kWh and its storage medium (i.e. PCM) is water. The internal structure of the tank comprises 68 spiralled polyethylene tubes grouped into 34 pairs to resemble counter-flow heat exchangers. The tubes are submerged in water and a 34% water–glycol mixture flowing through the tubes acts as the HTF. To store cooling energy, the system injects cold HTF (e.g. at −6 °C) into the tank, thereby freezing the water around the tubes into ice. Conversely, the discharging process involves injecting a warmer HTF (e.g. at 12 °C) to extract the cooling energy during the melting process of the ice.

The operation of the ice tank is governed by two key variables: mass flow rate ($\dot{m}_f$) and input temperature of the HTF ($T_{f,\mathrm{in}}$). The

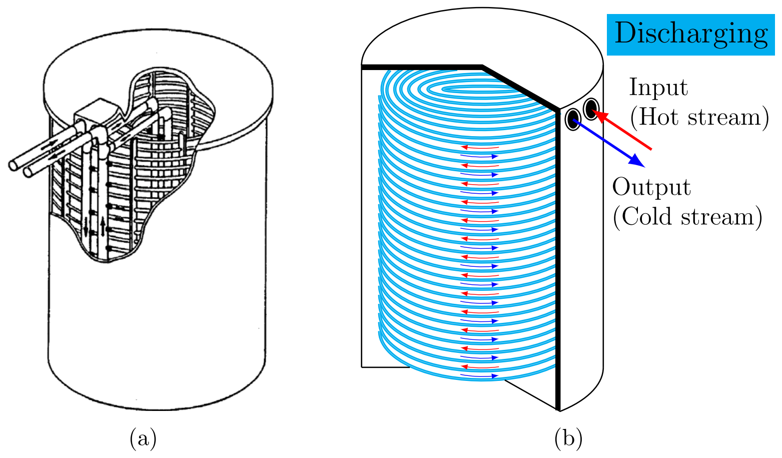


Fig. 1. (a) Internal structure of the ice tank and its external connections [31]. (b) Schematic of the internal counter-flow heat exchanger structures formed by the pair of tubes and their HTF flows [29].

Table 1
Ranges of input temperature and mass flow rate of the HTF.

Variable	Charging	Discharging
$\dot{m}_f$	$[18 \quad 22]$ kg/s	$[5 \quad 6]$ kg/s
$T_{f,\text{in}}$	−6 °C	$[10 \quad 14]$ °C

operating conditions for charging and discharging adopted for this work are shown in Table 1. These were assumed based on experimental data presented in [32,33], as the tank manufacturer does not provide specific information. According with the experiments reported in the references, $T_{f,\text{in}}$ ranges from 10 °C to 14 °C during the discharging process. Although these temperature limits may be exceeded in practice, in this paper discharging was restricted within these boundaries to maintain simplicity. Conversely, $T_{f,\text{in}}$ ranges from −3.7 °C to −6 °C during charging due to the limited capabilities of the chiller used in the experiment. However, a constant $T_{f,\text{in}} = -6$ °C was here considered, assuming a chiller with a higher capacity.

The duration of experiments was 500 min (8.33 h) for discharging and 1400 min (23.33 h) for charging. These durations were consequence of the HTF mass flow rates for discharging and charging ($\dot{m}_f = 1.027$ kg/s = 3689 kg/h and $\dot{m}_f = 0.68$ kg/s = 2454 kg/h) and a special setup of the thermal store employed to reduce the storage capacity from 350 kWh to 172 kWh [33]. While the discharging and charging times under the same operating conditions may double when the ice tank operates at full capacity (i.e. 16 h for discharging and 46 h for charging), a full charging cycle is expected to last between 6 to 12 h in a practical application [34]. Thus, higher values of $\dot{m}_f$ were here adopted to keep the discharging and charging times within this time range according to [15]. To meet cooling demand, $\dot{m}_f$ is maintained between 5 kg/s and 6 kg/s for discharging operations. In contrast, the charging process requires a considerable increase in mass flow rate, with $\dot{m}_f$ ranging between 18 kg/s and 22 kg/s. This increment is required as the temperature difference between $T_{f,\text{in}} = -6$ °C and the melting temperature of the PCM (i.e. 0 °C) is significantly reduced when compared to a discharging process as $T_{f,\text{in}}$ ranges from 10 °C to 14 °C.

2.2. On the mathematical model of the ice tank and its continuous-time observer for SoC estimation

The choice of an ice tank as the LHTES unit is motivated by its widespread use in cooling systems, such as food and pharmaceutical cold-chains and large district cooling systems, and its cost-effectiveness arising from the convenient melting temperature of the PCM (i.e. water) and considerable latent heat absorbed or released during phase transition. Detailed data on the internal structure of the tank and thermophysical properties of the HTF are provided in [30,35].

A mathematical model of the tank is provided in [29], which is based on principles of energy balance and heat transfer theory. A one-dimensional spatial discretisation method is employed to model the thermal stratification within the tank. This approach divides the tank's volume into a number of discrete volumes or nodes to capture the temperature gradient of water within the tank (either in liquid or crystallised form) and the HTF circulating through the tubes.

The temperature distribution of each node in the tank model is described by [29]

$$\begin{bmatrix} \dot{T}_{f,i,a} \\ \dot{T}_{w,i,a} \\ \dot{T}_{f,i,b} \\ \dot{T}_{w,i,b} \end{bmatrix} = \begin{bmatrix} \dfrac{\dot{m}_f c_{p,f,1,a}\left[T_{f,i-1,a}-T_{f,i,a}\right]+U\left(A_{tr}/N\right)\left[T_{w,i,a}-T_{f,i,a}\right]}{\rho_{f,i,a}c_{p,f,i,a}\left(V_f/N\right)} \\ \dfrac{U\left(A_{\text{tr}}/N\right)\left[T_{f,i,a}-T_{w,i,a}\right]+U_w\left(A_e/N\right)\left[T_{w,i,b}-T_{w,i,a}\right]+U_l\left(A_e/N\right)\left[0-T_{w,i,a}\right]}{\rho_{w,i,a}c_{p,w,i,a}\left(V_w/N\right)} \\ \dfrac{\dot{m}_f c_{p,f,1,b}\left[T_{f,i+1,b}-T_{f,i,b}\right]+U\left(A_{tr}/N\right)\left[T_{w,i,b}-T_{f,i,b}\right]}{\rho_{f,i,b}c_{p,f,i,b}\left(V_f/N\right)} \\ \dfrac{U\left(A_{tr}/N\right)\left[T_{f,i,b}-T_{w,i,b}\right]+U_w\left(A_e/N\right)\left[T_{w,i,a}-T_{w,i,b}\right]+U_l\left(A_e/N\right)\left[0-T_{w,i,b}\right]}{\rho_{w,i,b}c_{p,w,i,b}\left(V_w/N\right)} \end{bmatrix} \tag{1}$$

where T [°C], c_p [J/(kg °C)] and ρ [kg/m^3] refer to the temperature, specific heat coefficient and density, subscripts 'f' and 'w' refer to HTF and water, and subscripts 'a' and 'b' refer to each individual tube within the tank. Subscript 'i' refers to the node modelled, while '$i-1$' and '$i+1$' refer to nodes immediately before and after node i. V_f and V_w [m^3] refer to the volume fractions of HTF and water, A_{tr} [m^2] and U [W/(m^2 °C)] are the heat transfer area and heat transfer coefficient between the water in the tank and the HTF in the tubes, A_e [m^2] is the external area of the node, and U_w and U_l [W/(m^2 °C)] are the conduction heat transfer coefficients of water and between the node and the environment.

To model the phase change of water, specific heat curves were defined for $c_{p,w}$ to characterise the charging and discharging processes. These are given by the probability density function (PDF) [29]

$$c_{p,w} = 1000\left[a_0 + a_1(\varphi(T) - a_2)\right], \tag{2}$$

where

$$\varphi(T) = \begin{cases} \dfrac{\exp\left[-\left(\left(\ln\left[-T/\gamma\right]\right)^2/\left(2\sigma^2\right)\right)\right]}{-T\sigma\sqrt{2\pi}}, & T < 0 \\ 0, & T \geq 0 \end{cases}, \tag{3}$$

where a_0, a_1, a_2, γ and σ are dimensionless parameters heuristically adjusted to replicate experimental data.

The tank's model was validated in [29] by comparison with experimental data available in [32,33], with further information on the modelling considerations also provided. Results in [29] demonstrate high modelling accuracy, with values of mean square error of 0.23 °C for the HTF's output temperature in charging processes and of 0.04 °C for discharging with respect to experimental data. To avoid duplicating published work, no further discussion on the model validation is here provided. Interested readers are referred instead to [29].

A continuous-time non-linear observer based on the experimentally verified model of the ice tank presented in [29] was developed and assessed in [15] for the estimation of SoC with successful results. The observer determines SoC by estimating the PCM's temperature and employing a calculation methodology that considers the specific heat–temperature curve of the PCM and its latent heat value. The observer's performance to estimate the input and output temperatures of the HTF and the temperature gradient of the PCM was successful, with the estimation error converging to zero in all the operating conditions under examination.

An overview of the continuous-time non-linear observer is provided in [15], with the method to calculate SoC included in Appendix B for completeness. A methodology to obtain a discrete-time representation of the continuous-time observer is presented in Appendix A, and this is further discussed in Section 5. To prevent duplication of published work, no further discussion on the validation of the ice tank model or the continuous-time non-linear observer is here provided and, instead, interested readers are referred to [15,29].

2.3. Simulations of the ice tank model

The charging and discharging processes of the ice tank were simulated in MATLAB/Simulink using the mathematical model presented in [29]. The model was spatially discretised into 20 nodes. As demonstrated in the same reference, increasing the number of discrete nodes for modelling the ice tank does not produce a significant improvement in the accuracy of the simulation results. Simulations were run for the operating conditions defined in Table 1, with results shown in Fig. 2. Subscripts '5', '6', '18' and '22' in the figure's legend stand for the HTF's mass flow rate in kg/s employed in a given simulation. The input temperature of the HTF ($T_{f,\mathrm{in}}$) is shown with a green trace. The temperature of the final node of the PCM (i.e. water, T_p, shown with blue traces) was included alongside the HTF output temperature ($T_{f,\mathrm{o}}$, black traces) to visualise the exact moment when this node reaches a solid or liquid state depending on whether the tank was being charged or discharged. The maximum and minimum mass flow rates defined in Table 1 were used for charging simulations: $\dot{m}_f = 18$ kg/s and $\dot{m}_f = 22$ kg/s with a similar $T_{f,\mathrm{in}} = -6$ °C for both. As it can be observed, there is a reduction in the rate of change of the PCM temperature during the release of specific latent heat between 1 h and 7.5 h into the simulation for charging (see Fig. 2(a)), whereas such a reduction is exhibited from the beginning of the simulation to 2.3 h (with $\dot{m}_f = 5$ kg/s and $T_{f,\mathrm{in}} = 10$ °C) and 3.5 h (with $\dot{m}_f = 6$ kg/s and $T_{f,\mathrm{in}} = 14$ °C) during the absorption of the specific latent heat for discharging (see Fig. 2(b)). After the specific latent heat is fully released or absorbed, the temperature gradient increases until the temperature of the PCM and the output temperature of the HTF reach the input temperature.

For completeness, SoC during charging and discharging was also included in Fig. 2 and is shown with solid and dashed red traces. In this calculation, SoC only considers the specific latent heat of water as defined in [15] (see Appendix B). This approach is adopted to exclude the sensible heat, whose value depends on the HTF's input temperature used to discharge the unit. Thus, during charging, shown in Fig. 2(a), SoC starts to increase 1 h into the process as the sensible heat is firstly released. Once the specific latent heat has been released through the whole PCM volume, SoC reaches a value of 100% and the unit is considered as fully charged. On the other hand, during discharging (Fig. 2(b)), SoC immediately decreases as the specific latent heat is absorbed by the start of the process. This concludes after 2 h for a mass flow rate $\dot{m}_f = 5$ kg/s (solid red trace) and after 3 h when $\dot{m}_f = 6$ kg/s (dashed red trace), when, in either case, SoC drops to a value of 0% and the ice tank is fully discharged. The behaviour of the traces for SoC aligns with the variations in the rate of change of the PCM's temperature.

3. Recurrent neural networks

Machine learning algorithms such as support vector regression, random forest, and the widely used feed-forward neural network do not feature memory [36,37]. This means they do not retain any state from one time-step to the next one and thus require an entire time-series or data sequence to make a prediction. In contrast, an RNN incorporates an internal loop that retains a state, thereby preserving historical information from previous time-steps throughout the operations [38, 39].

The mathematical formulation of a basic RNN structure is defined as follows [40]:

$$h_t = g\left(W_h x_t + U_h h_{t-1} + b_h\right), \tag{4}$$

$$y_t = g\left(W_y h_t + b_y\right), \tag{5}$$

where x is the input vector, h is the hidden layer vector, y is the output vector, W is the weight matrix for the inputs, U is the weight matrix for the hidden state, b is the bias vector, and g is the activation function. Subscript 't' stands for the time-step number, whereas 'h' and 'y' stand for matrices corresponding to hidden state and output calculations.

Activation functions such as the hyperbolic tangent and sigmoid are critical for neuron activation and determining the relevance of the inputs to network predictions. Both are derived from the exponential function and guide the hidden state in recognising significant information from the cell state. Mathematically, the sigmoid and hyperbolic tangent functions are expressed as [38,41,42]

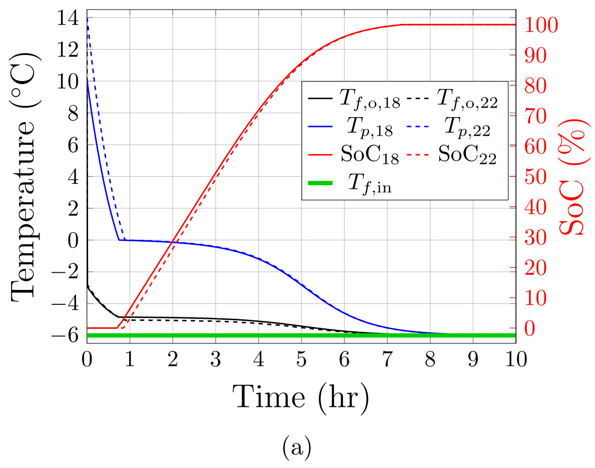
$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{6}$$

$$\tanh(\mathrm{x}) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{7}$$

3.1. Overview of long short-term memory structures

Due to the gradient-based learning nature of conventional RNNs, a vanishing gradient issue can be experienced during training in which the error function (i.e. the gradient) decays exponentially [43]. Such a behaviour is not desired as a vanishingly small gradient may prevent the RNN weights from being updated and this effect could prematurely stop the training process. Ref. [38] introduced a state memory cell to tackle this issue. The memory cell is included in an LSTM structure to relay information across time-steps. This ensures a constant error signal overflow during generation of the network weights throughout the training process.

Fig. 3 shows the internal structure of an LSTM cell including its three main components, termed forget, input, and output gates. The forget gate, indicated with a light red shading, employs a sigmoid activation function to decide which information must be kept or discarded. The input gate, illustrated with a purple shading, feeds the current state x_t and the previous hidden state h_{t-1} into a second sigmoid function to generate the input data i_t. This function facilitates the transformation of values within the range of 1 (indicating significance) to 0 (indicating insignificance). In parallel, h_{t-1} and x_t are fed to a hyperbolic tangent activation function. The purpose of using this function is to establish control over values flowing through the network to prevent information from fading. Thus, the output of this operator is used to generate the cell state candidate, $\tilde{c}_t$, with potential values ranging from −1 to 1. Then, a proportion of i_t is incorporated into the cell state, c_t, based



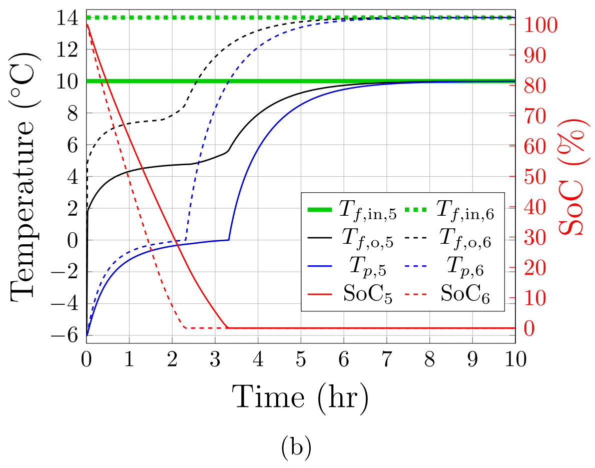


Fig. 2. Simulation results of the ice tank with input and output temperatures of the HTF ($T_{f,\mathrm{in}}$ and $T_{f,o}$), PCM temperature of the last node of a 20-nodes spatially discretised model (T_p), and SoC: (a) charging; (b) discharging. The additional numerical subscript in the variables stands for the mass flow rate in kg/s employed for charging and discharging.

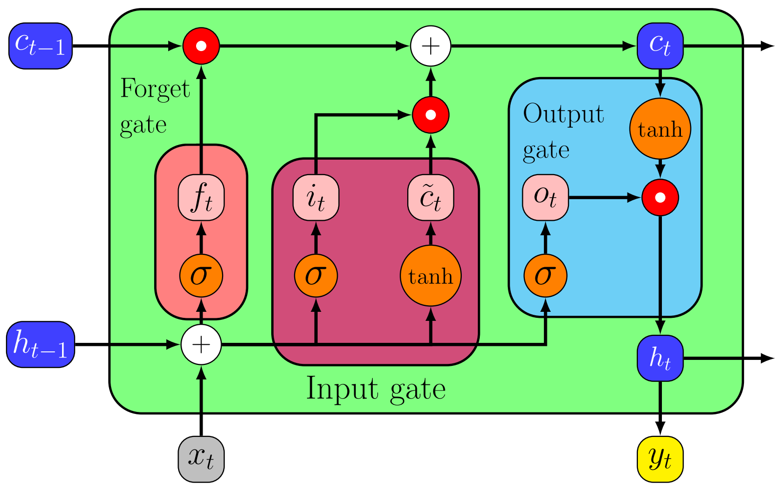


Fig. 3. LSTM architecture with the forget, input, and output gates highlighted in light red, purple, and light blue. Input, output, hidden, and cell states are shown as inputs for the cell.

on a Hadamard product with $\tilde{c}_t$. The Hadamard product operator, also known as the element-wise product, is indicated in the figure by the red circle with a white dot.

The output gate determines the hidden state, h_t, which encodes information from prior inputs. It involves processing x_t and state h_{t-1} through a third sigmoid function to generate the output o_t in addition to processing c_t through a hyperbolic tangent activation function. Both outputs from the activation functions are multiplied using a Hadamard product to determine the relevant information for the hidden state used in the prediction. h_t and c_t are then propagated to the subsequent time-step. The mathematical formulation for all the described internal calculations within an LSTM layer is as follows [38,41]:

$$i_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right), \tag{8}$$

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right), \tag{9}$$

$$\tilde{c}_t = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right), \tag{10}$$

$$o_t = \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right), \tag{11}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{12}$$

$$h_t = o_t \odot \tanh\left(c_t\right), \tag{13}$$

where subscripts 'c', 'f', 'i', and 'o', stand for the corresponding weights of the candidate cell state, forget gate, input gate, and output gate and $\odot$ stands for the Hadamard product.

Fig. 4 shows the temporal unfolding of an LSTM cell showing both the hidden and cell states. It is important to emphasise the need for initial conditions (h_i and c_i) both for the training process and subsequent computation during implementation of the LSTM-based network. They are critical for ensuring an accurate prediction. Fig. 5 provides an extended diagram showing the calculations for the first three time-steps to highlight the internal structure of the LSTM cell and the interconnections between hidden and cell states.

3.2. Proposed network architectures

The architectures proposed in this work comprise either five or six layers in total considering three distinct types of layers. Dense layers are used at the beginning and end of the network to align the number of inputs and outputs with the neuron count in the hidden layers. A rectified linear unit (ReLU) layer is placed after the input layer and before the output layer (both dense layers) to execute a threshold operation retaining only positive values. Thus, by introducing such a non-linearity the issue of vanishing gradients during the training process is mitigated [44]. Finally, one or two LSTM layers are integrated in the middle of the structure, leading to the two architectures shown in

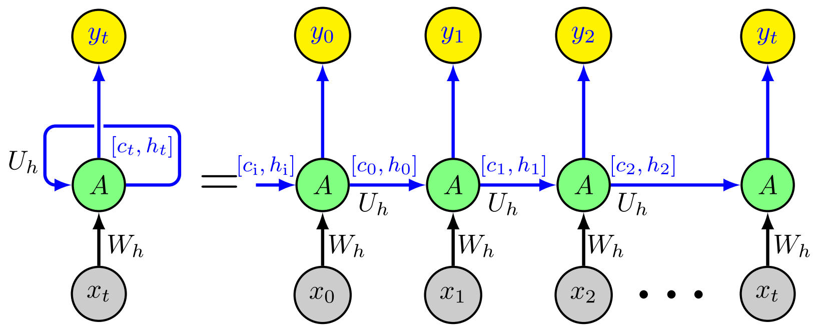


Fig. 4. Architecture of the LSTM unfolding through the time-steps.

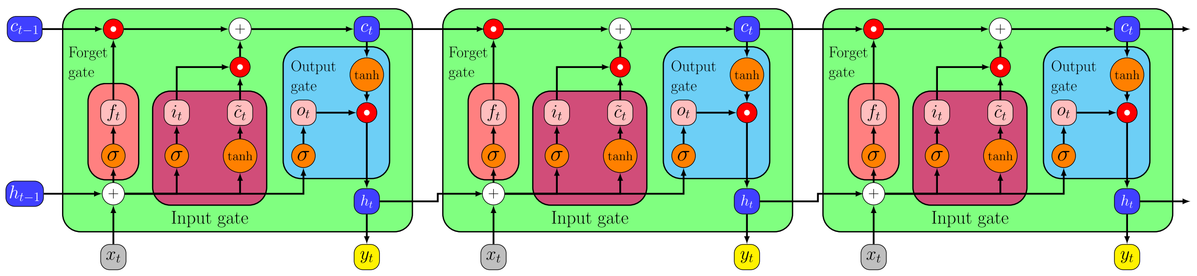
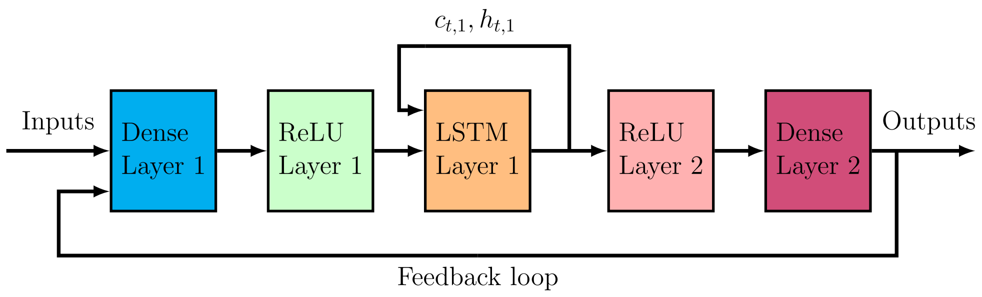
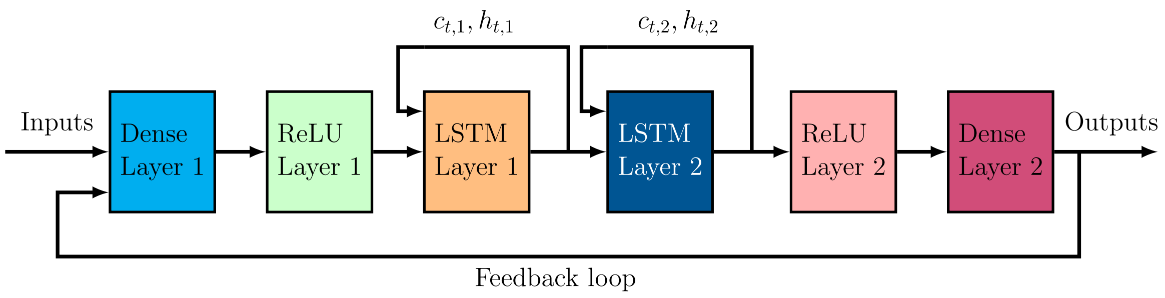


Fig. 5. LSTM cell unfolded during three time-steps including its internal structure.



(a)



(b)

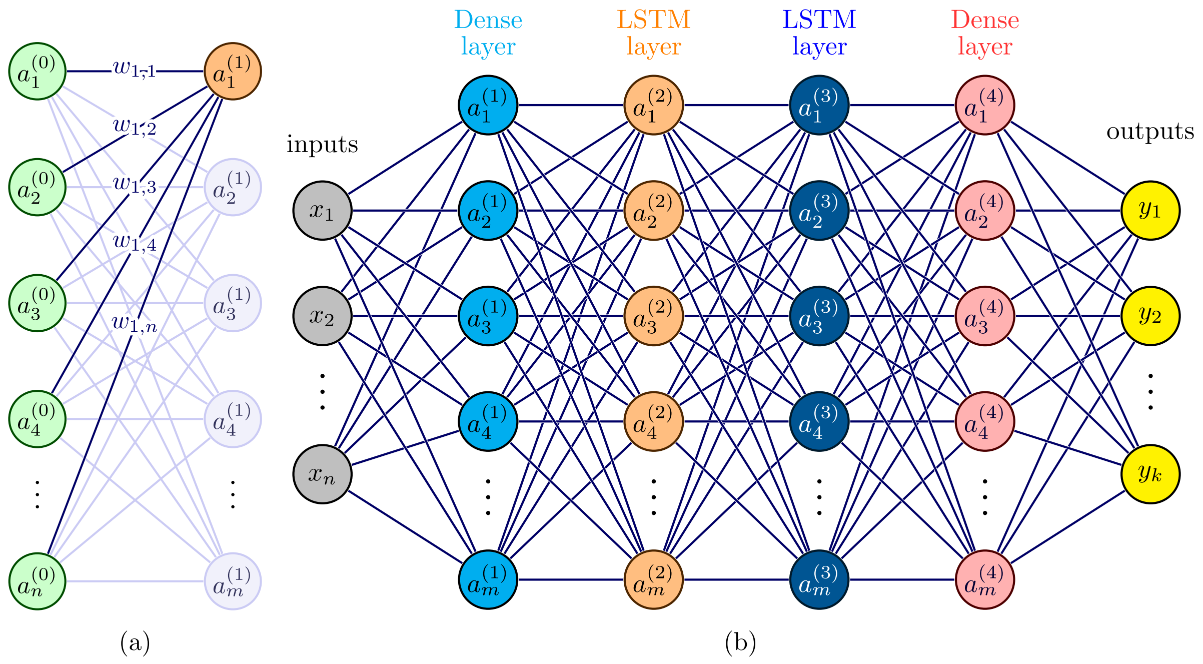
Fig. 6. Architecture of the proposed RNN for the SoC prediction of an LHTES unit: (a) one LSTM hidden layer, (b) two LSTM hidden layers.

Fig. 6. The LSTM layers provide memory feedback through their hidden and cell states. Both architectures consider an overall feedback loop, which represents the use of data regression in the predictive process. The predictive process will be discussed later in Section 4.2.

A mathematical description of the architectures shown in Fig. 6 is provided next. The dense layers at the network's extremities are deeply interconnected with their preceding layer (or inputs) by linking each neuron, given by the weight matrix W and bias vector b, to reinforce the connection. This is mathematically expressed, in compact form, as [42]

$$y_D = W_D x_t + b_D, \tag{14}$$

where x represents the input vector (with n elements) and y the output vector (with k entries) of the dense layer. Subscript 'D' is used to denote the dense layer. The dimensions of W and b must be consistent with



$w_{1,1}$
$w_{1,2}$
$w_{1,3}$
$w_{1,4}$
$w_{1,n}$
$a_1^{(0)}$ $a_2^{(0)}$ $a_3^{(0)}$ $a_4^{(0)}$ $a_n^{(0)}$
$a_1^{(1)}$ $a_2^{(1)}$ $a_3^{(1)}$ $a_4^{(1)}$ $a_m^{(1)}$
(a)
Dense layer
LSTM layer
LSTM layer
Dense layer
inputs
outputs
x_1 x_2 x_n
$a_1^{(2)}$ $a_2^{(2)}$ $a_3^{(2)}$ $a_4^{(2)}$ $a_m^{(2)}$
$a_1^{(3)}$ $a_2^{(3)}$ $a_3^{(3)}$ $a_4^{(3)}$ $a_m^{(3)}$
$a_1^{(4)}$ $a_2^{(4)}$ $a_3^{(4)}$ $a_4^{(4)}$ $a_m^{(4)}$
y_1 y_2 y_k
(b)

Fig. 7. (a) Interconnection of the inputs from a previous layer to the first neuron of a dense layer through its matrix of weights. (b) Schematic of the RNN with two LSTM layers specifying the number of inputs (n), the number of neurons (m), and the number of outputs (k). ReLu layers are not shown for clarity.

the layer's number of inputs n and outputs k, and the size of vector b, which is equal to the number of neurons m.

As a way of an example, let the outputs of each layer be represented by vector a (to distinguish it from the adopted notation in (14) using y as the output). Superscripts within parentheses indicate the layer number and subscripts stand for the number of the vector element. This way, the outputs of layer 0 are the inputs of layer 1 (dense layer). Calculation of $a_1^{(1)}$, as shown in Fig. 7(a), is given by

$$a_1^{(1)} = w_{1,1}a_1^{(0)} + w_{1,2}a_2^{(0)} + \cdots + w_{1,n}a_n^{(0)} + b_1^{(1)} = \sum_{i=1}^{n} w_{1,i}a_i^{(0)} + b_1^{(0)}. \tag{15}$$

Expanding (15) to consider all the elements of the output vector of the dense layer leads, in matrix form, to:

$$\begin{bmatrix} a_1^{(1)} \\ a_2^{(1)} \\ \vdots \\ a_m^{(1)} \end{bmatrix}_{[m\times1]} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix}_{[m\times n]} \begin{bmatrix} a_1^{(0)} \\ a_2^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix}_{[n\times1]} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_m^{(1)} \end{bmatrix}_{[m\times1]} \tag{16}$$

where the dimensions of the matrices and vectors are explicitly given as subscripts within square brackets. For a dense layer the number of neurons is equal to its number of outputs, so $k = m$.

The ReLU layer employs a piecewise linear function. It outputs the input values directly when these are positive and zero otherwise. This is mathematically described as [42]

$$y_R = \begin{cases} x_t, & x_t \geq 0 \\ 0, & x_t < 0 \end{cases} \tag{17}$$

where subscript 'R' stands for ReLU layer. For this type of layer, the number of inputs and outputs is equal, so $n = k$.

The mathematical description of the LSTM layers aligns with the formulation described in Section 3.1. However, this is simplified here by merging the weight matrices of the gates and the cell state into a single matrix before the activation functions. This enables the independent calculation of the outputs of the gates (i_t, f_t, $\tilde{c}_t$, and o_t) as defined by (8)–(11). In compact form, this is expressed by

$$\begin{bmatrix} i_t \\ f_t \\ \tilde{c}_t \\ o_t \end{bmatrix}_{[4\ m\times1]} = \begin{bmatrix} W_i \\ W_f \\ W_c \\ W_o \end{bmatrix}_{[4\ m\times n]} x_{t[n\times1]} + \begin{bmatrix} U_i \\ U_f \\ U_c \\ U_o \end{bmatrix}_{[4\ m\times n]} h_{t-1[n\times1]} + \begin{bmatrix} b_i \\ b_f \\ b_c \\ b_o \end{bmatrix}_{[4\ m\times1]} \tag{18}$$

The dimensions of matrices in (18) shown with subscripts within square brackets are defined by the number of inputs n of the LSTM layer and the number of neurons m. As for the dense layer, m is equal to the number of outputs k (i.e. $k = m$).

Then, as shown in Fig. 3, the two outputs of the LSTM cell c_t and h_t are calculated as

$$c_t = \sigma\left(f_t\right) \odot c_{t-1} + \sigma\left(i_t\right) \odot \tanh\left(\tilde{c}_t\right), \tag{19}$$

and the hidden state vector h_t is obtained with

$$h_t = \sigma\left(o_t\right) \odot \tanh\left(c_t\right). \tag{20}$$

The dimensions of the cell and hidden vectors in the LSTM layer are consistent with the number of neurons.

Fig. 7(b) shows a detailed schematic of the proposed RNN. The relationship between each layer is shown through lines representing the weight values that multiply the output of each neuron for the subsequent layer connection. The ReLU layers were excluded for simplicity as they do not feature any weight matrix or bias vector. The layer number is indicated by the superscript in the neuron label.

The simplification made by merging the weight matrices of the gates and the cell state into a single matrix before the activation functions, which in turn provides an output in column form, is relevant as it facilitates an implementation in Simulink. This is consistent with the outputs following RNN training using the deep learning toolbox in MATLAB, where the weight matrices and their corresponding bias vectors for the calculation of i_t, f_t, $\tilde{c}_t$, and o_t are returned in two single matrices ordered in column form.

4. RNN SoC estimator: problem formulation, datasets and training

In this section, a detailed description of the RNN estimator based on LSTM layers introduced in Section 3.2 and its training are presented.

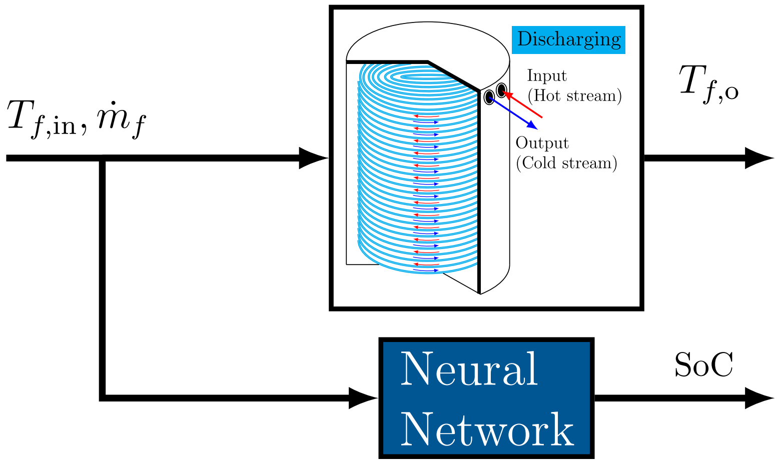


Fig. 8. General architecture of the RNN-based SoC estimator.

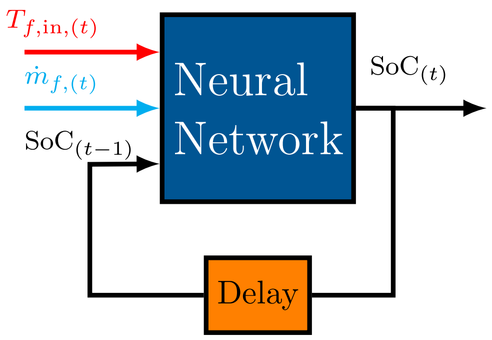


Fig. 9. Delay implementation over the feedback of the output of the RNN-based SoC estimator.

SoC prediction of the LHTES unit is formulated as a time-series forecasting problem—an approach borrowed from the field of statistics [45]. This approach guides the definition of the dataset employed during the training stage and is presented first. Then, the training of the RNN estimator is described. Since the objective of adopting a neural network is not to conduct a time-series analysis but rather to implement the SoC estimator, the trained RNN serves as a reduced-order model that may substitute a more intricate model (in this case, a non-linear continuous-time observer). This facilitates the dynamic SoC estimation using real-time measurements of key system variables (i.e. mass flow rate and input temperature of the HTF).

Fig. 8 shows a high-level schematic of the SoC estimation process of the ice tank using an RNN. Measurements of $\dot{m}_f$ and $T_{f,\text{in}}$ are taken to activate the neural network as these variables are typically accessible in practical LHTES units. This way, the RNN predicts SoC based on prior training. (Details on the training process are provided in Section 4.4.) However, predicting SoC requires considering the previous estimation value, which is similar to a linear regression problem. Thus, the RNN inputs comprise the actual HTF values ($\dot{m}_f$, $T_{f,\text{in}}$) and the previously estimated SoC. This earlier value is obtained by including a unit delay to the SoC prediction, as shown in Fig. 9. Unlike the estimators presented in [46], there is no pre-established data window to encapsulate several measurements and then estimate the corresponding number of outputs as used in time-series analysis. The RNN estimator generates a single SoC value only every time the previous input values are fed.

4.1. Dataset for training

The primary objective of the estimator is to predict the SoC of the TES unit upon variations in system inputs (e.g. changes in mass flow rate or temperature of the HTF). Through simulations of the mathematical model of the ice tank (briefly discussed in Section 2.2), SoC was obtained using the temperature of the PCM and the calculation method presented in [15] (see Appendix B). The interested reader is referred to [29] for a comprehensive description of the model of the ice tank used to obtain the training dataset for this paper. The dataset was determined by simulating charging and discharging operations of the tank considering the operating limits in Table 1. A range of temperature and mass flow rate inputs of the HTF ($T_{f,\text{in}}$ and $\dot{m}_f$) was adopted to generate the possible combinations of the operating conditions. A particular condition considered is the initial temperatures of the PCM and HTF during charging. This condition was determined by the inlet temperature of the HTF $T_{f,\text{in}}$ previously used for a discharging process. In contrast, variations for a discharging process consider $\dot{m}_f$ and $T_{f,\text{in}}$, as the initial temperatures of the PCM and HTF are always −6 °C (i.e. for full charge).

Table 2 shows the range of the variables to obtain SoC for charging and discharging. The set of inputs was generated varying the value of $\dot{m}_f$ and $T_{f,\text{in}}$ in steps of 0.1 for each variable. Thus, 41 different mass flow rate inputs and 41 different initial conditions of the HTF and PCM temperatures were used for charging, leading in turn to a set of 1681 SoC profiles. In contrast, the smaller range of mass flow rate for discharging produced only 11 different inputs, which led to a set of 451 SoC profiles considering the 41 different input temperatures of the HTF. A total of 2132 SoC profiles was thus generated and these profiles are graphically shown in Fig. 10.

The simulation time for each charging and discharging operation was 10 h. Although discharging is considerably shorter than charging, by establishing the same duration the number of data elements for the training process was homogenised. This prevented issues during training arising from an element number mismatch in the dataset.

4.2. Regression problem

The underpinning concept of the regression problem for estimation involves leveraging preceding values to forecast the subsequent step. Thus, the simulation-generated data must be adjusted to be suitable for training an RNN which will use this approach for prediction. This is illustrated in Table 3 for a sequence of the initial four time-steps. In each time-step, output y is directly generated by input x. The output generated is subsequently converted into the input of the next step.

Using previous values of the output as an input simplifies the dataset adjustment process [46]. Thus, by having the dataset of the system variable that will be predicted, the input and output of the system for training are defined by truncating the final elements of the dataset arrays and shifting these arrays. This process is shown in Fig. 11 for

Table 2
Combinations of the input variables and initial conditions for generating the training dataset.

Charging Constant input temperature of the HTF: $T_{f,\text{in}} = -6$ °C		**Discharging** Constant initial conditions of the PCM and HTF at -6 °C	
Variable	Range	Variable	Range
Mass flow rate [kg/s]	$[18 \quad 18.1 \quad \cdots \quad 22]$	Mass flow rate [kg/s]	$[5 \quad 5.1 \quad \cdots \quad 6]$
Initial conditions [°C]	$[10 \quad 10.1 \quad \cdots \quad 14]$	Input temperature [°C]	$[10 \quad 10.1 \quad \cdots \quad 14]$

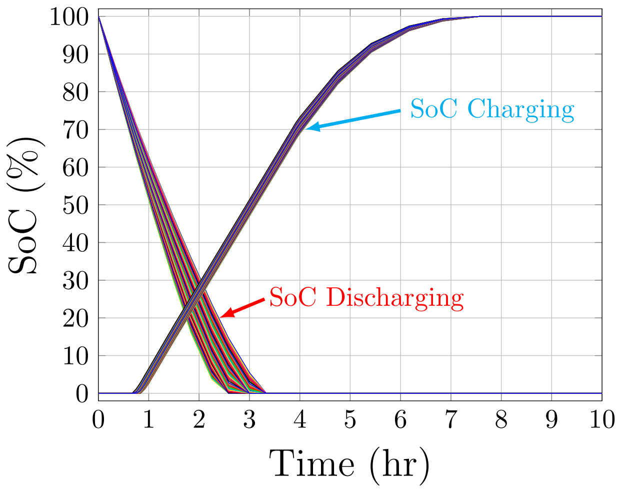


Fig. 10. SoC profiles (2132 in total) for all possible combinations of the operating conditions in Table 2.

Table 3
Sequence of inputs and outputs produced during the implementation of a regression method.

Step	Input and output relationship
t_0	$x(t_0) \rightarrow y(t_0)$
t_1	$y(t_0) = x(t_1) \rightarrow y(t_1)$
t_2	$y(t_1) = x(t_2) \rightarrow y(t_3)$
t_3	$y(t_2) = x(t_2) \rightarrow y(t_3)$
$\vdots$	$\vdots$

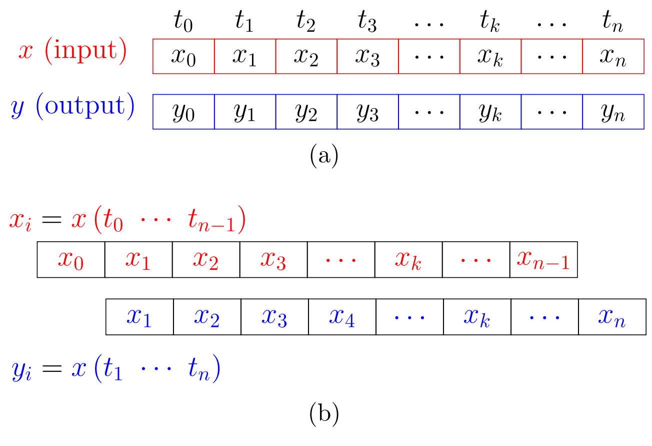

x (input)
y (output)
t_0 t_1 t_2 t_3 $\cdots$ t_k $\cdots$ t_n
x_0 x_1 x_2 x_3 $\cdots$ x_k $\cdots$ x_n
y_0 y_1 y_2 y_3 $\cdots$ y_k $\cdots$ y_n
(a)
$x_i = x(t_0 \ \cdots \ t_{n-1})$
x_0 x_1 x_2 x_3 $\cdots$ x_k $\cdots$ x_{n-1}
x_1 x_2 x_3 x_4 $\cdots$ x_k $\cdots$ x_n
$y_i = x(t_1 \ \cdots \ t_n)$
(b)

Fig. 11. (a) Input and output arrays. (b) New arrangement after a one-element shift is applied.

a one-element shift. In time-series analysis, such a shift procedure is implemented using a sliding window over dataset arrays which include more than one element [46].

In this paper, since SoC estimation is performed at each time-step, shifting is done by moving the dataset array one element to the right to generate the output array. On the other hand, the input array is arranged by eliminating the last element.

4.3. Sampling time

The training dataset as discussed in Section 4.1 was obtained from simulations conducted in MATLAB/Simulink. A time-step of 0.5 s was originally adopted to achieve good accuracy [29]. Such a small time-step is necessary to ensure convergence in the implementation of the discrete-time non-linear observer (see Appendix A). This resulted in 72,000 elements for the array corresponding to each variable when considering a 10-h simulation period (as discussed in Section 4.1). Given three variables are under consideration ($\dot{m}_f$, $T_{f,\text{in}}$, and SoC), the resulting dataset is large. Such a considerable number of elements may affect the RNN's training when multiplied by the number of profiles encompassing all possible operating modes of the ice tank.

To prevent issues such as a vanishing gradient, long periods of training, or convergence to suboptimal results, it is essential to reduce the number of elements of the dataset by downsampling. This is achieved by increasing the sampling time. Using a sampling time of 5 min instead of 0.5 s, the number of elements was brought down to 120 elements per array. The rationale behind adopting a longer sampling time compared to that of the discrete-time observer is that knowing SoC every second is not required in a practical application of the ice tank.

Fig. 12 shows an example of downsampling of the datasets. A profile for charging and one for discharging as in Fig. 2 were sampled at 5-min intervals. Subscripts 's' and '5' denote the results derived from the simulation with a time-step of 0.5 s and the re-sampled results every 5 min. Notably, larger sampling times can be adopted as the SoC does not exhibit abrupt changes during charging or discharging—thereby eliminating the need for a shorter sampling time. Hence, considering the slow thermal dynamics of the ice tank, a 5-min sampling time is deemed suitable for a practical SoC estimation.

4.4. Training

A subset **Z** of 132 randomly selected profiles from the total 2132 profiles described in Section 4.1 was adopted for training. The subset considers 26 charging profiles and 106 discharging profiles. The rationale for the reduced subset was to expedite the training process. Given that each profile consists of 121 samples, a significant reduction was deemed necessary. In addition, a considerable impact on the accuracy of the estimator was not expected as the SoC profiles do not exhibit significant variations between them for either charging or discharging (see Fig. 10). Each selected profile incorporates three variables: $\dot{m}_f$, $T_{f,\text{in}}$, and the corresponding SoC output. Then, the downsampling process, the elimination of the last element of the arrays to obtain the input dataset **X**, and the shifting to obtain the output dataset **Y** were implemented to subset **Z** as discussed in Sections 4.2 and 4.3. All this is schematically illustrated in Fig. 13.

MATLAB's machine learning toolbox, a comprehensive suite of algorithms and tools for constructing, training, and validating machine learning models, was used to train the RNN-based SoC estimator. The toolbox enabled specifying the neural network structure, determining hyperparameters, and launching the training regime. The Adam optimisation algorithm was employed to iteratively update the network weights based on training data. This algorithm, widely utilised in computer vision and natural language processing applications, was adopted due to its straightforward implementation, computation efficiency, and

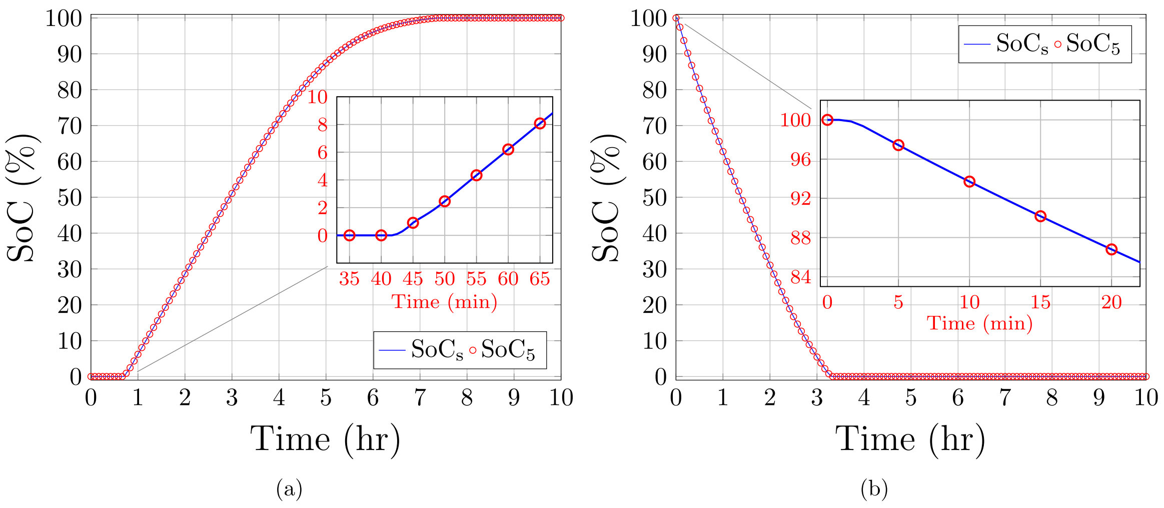


Fig. 12. Downsampling process of the SoC profiles. (a) Charging. (b) Discharging.

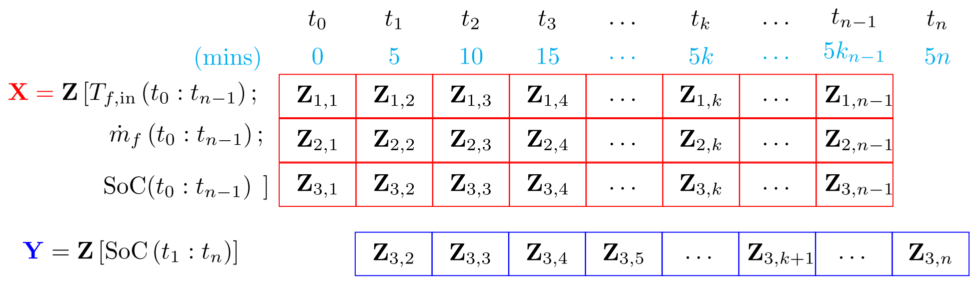

t_0 t_1 t_2 t_3 $\dots$ t_k $\dots$ t_{n-1} t_n
(mins) 0 5 10 15 $\dots$ $5k$ $\dots$ $5k_{n-1}$ $5n$
$\mathbf{X} = \mathbf{Z}\,[T_{f,\mathrm{in}}(t_0 : t_{n-1})\,;$
$\dot{m}_f\,(t_0 : t_{n-1})\,;$
$\mathrm{SoC}(t_0 : t_{n-1})\ ]$
$\mathbf{Z}_{1,1}$ $\mathbf{Z}_{1,2}$ $\mathbf{Z}_{1,3}$ $\mathbf{Z}_{1,4}$ $\dots$ $\mathbf{Z}_{1,k}$ $\dots$ $\mathbf{Z}_{1,n-1}$
$\mathbf{Z}_{2,1}$ $\mathbf{Z}_{2,2}$ $\mathbf{Z}_{2,3}$ $\mathbf{Z}_{2,4}$ $\dots$ $\mathbf{Z}_{2,k}$ $\dots$ $\mathbf{Z}_{2,n-1}$
$\mathbf{Z}_{3,1}$ $\mathbf{Z}_{3,2}$ $\mathbf{Z}_{3,3}$ $\mathbf{Z}_{3,4}$ $\dots$ $\mathbf{Z}_{3,k}$ $\dots$ $\mathbf{Z}_{3,n-1}$
$\mathbf{Y} = \mathbf{Z}\,[\mathrm{SoC}\,(t_1 : t_n)]$
$\mathbf{Z}_{3,2}$ $\mathbf{Z}_{3,3}$ $\mathbf{Z}_{3,4}$ $\mathbf{Z}_{3,5}$ $\dots$ $\mathbf{Z}_{3,k+1}$ $\dots$ $\mathbf{Z}_{3,n}$

Fig. 13. Illustrative dataset shifting.

the use of hyperparameters that are intuitively interpretable and require minimal tuning [47]. These hyperparameters include the learning rate (driven by the estimated error, it determines the magnitude of adjustments made to the model parameters when updating the weight matrices), the learning drop factor (which modifies the learning rate after a specified number of epochs have elapsed), and the learning drop period (the number of epochs after which the drop factor is applied). Further information on these hyperparameters is available in [48].

The RNN training process was conducted using MATLAB R2023a and an AMD Ryzen 7 7730U CPU @ 2.00 GHz. It was started with an initial learning rate of 0.005. The learning rate drop factor was set at 0.6, adjusting every 10,000 epochs (learning rate drop period). To achieve an optimal learning rate, the number of epochs was set to 30,000. As a way of an example, Fig. 14 shows the outcome of the training process for the architecture comprising 2 LSTM layers and 20 neurons as obtained using MATLAB's machine learning toolbox, which presents an approximate duration of ∼34 min for training completion. Even when the validation data were excluded from training to speed-up the process (and thus the validation RMSE field is not here applicable), a consistent learning rate of 0.0018 was achieved, which is a value deemed acceptable [49,50]. Such learning rate was exhibited by all the RNN architectures (not shown). This is further evaluated in the next section.

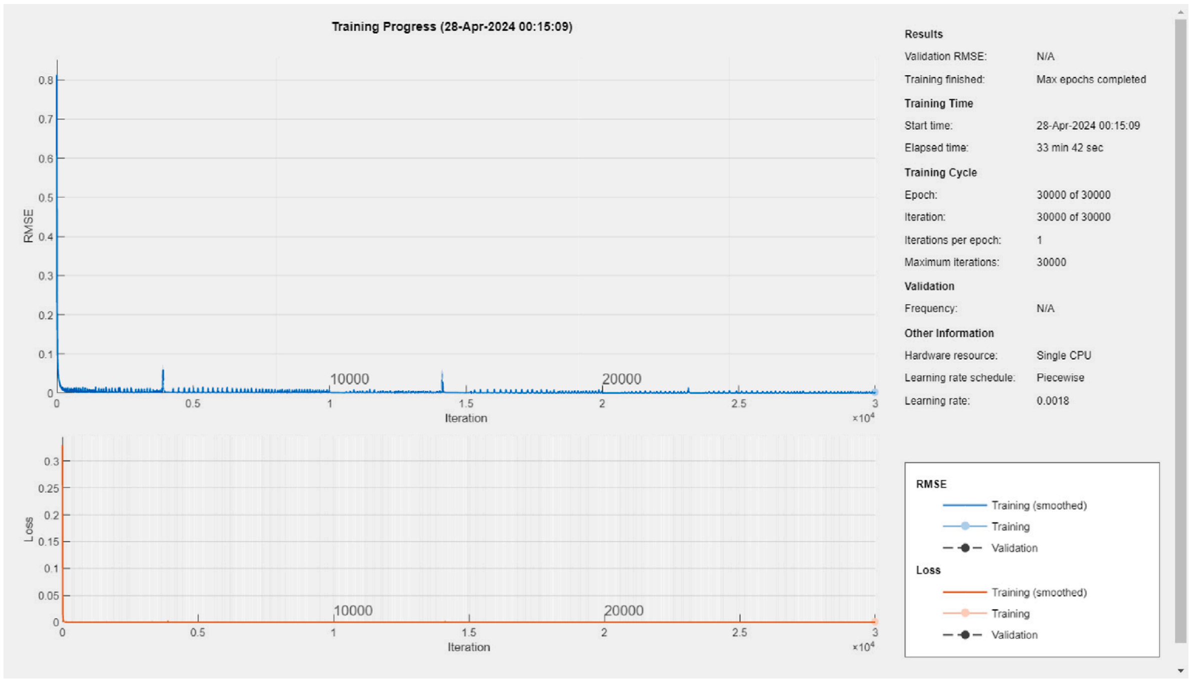
Fig. 15 schematically shows the training process discussed in this section. While some papers in the literature adopt systematic hyperparameter methods to fine-tune the optimal number of neurons of an RNN architecture, this was determined manually in this paper as in [51,52]. The rationale behind this was the relatively simple architecture of the proposed RNN. Eight different configurations were assessed in total. One and two LSTM hidden layers were adopted to prevent potential overfitting issues [52], as shown in Fig. 6. A minimum number of 5 neurons and a maximum of 20 neurons were considered in the LSTM layers. This led to four different configurations with one LSTM hidden layer and 5, 10, 15, and 20 neurons and four additional configurations with two LSTM hidden layers and 5, 10, 15, and 20 neurons. The minimum and maximum number of neurons per layer and the steps of 5 neurons were defined heuristically.

5. Implementation and results

This section presents the implementation process of the RNN-based SoC estimator. It uses MATLAB functions within the MATLAB/Simulink environment where the code was integrated. A detailed assessment of the performance of the estimator was carried out and a comparison was made against a discrete-time non-linear observer. This observer is the discretised version of the continuous-time domain non-linear observer discussed in Section 2.2 and originally presented in [15]. Discretisation is based on signal sampling concepts and was achieved using the mathematical representation of a data-hold circuit, a sampler, and the two-point backward difference formula. The method enables defining a discrete-time observer represented by a set of algebraic equations—circumventing the need for an ODE engine solver. For further details, interested readers are referred to Appendix A, which provides an overview of the discretisation process.

5.1. Discrete-time non-linear observer

The observer's performance was evaluated considering the ice tank model adopted from [29], the observer model included in Appendix A, and the SoC calculation methodology relying on the observer's estimated temperatures of the PCM shown in Appendix B. The ice tank model and the observer structure consider a (thermal) discretisation of the TES unit into 20 nodes. As discussed in Section 2, given that there are two tubes per control volume (denoted 'a' and 'b') and



Results

Validation RMSE:	N/A
Training finished:	Max epochs completed

Training Time

Start time:	28-Apr-2024 00:15:09
Elapsed time:	33 min 42 sec

Training Cycle

Epoch:	30000 of 30000
Iteration:	30000 of 30000
Iterations per epoch:	1
Maximum iterations:	30000

Validation

Frequency:	N/A

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Piecewise
Learning rate:	0.0018

Fig. 14. Training results for an RNN with 2 LSTM layers and 20 neurons.

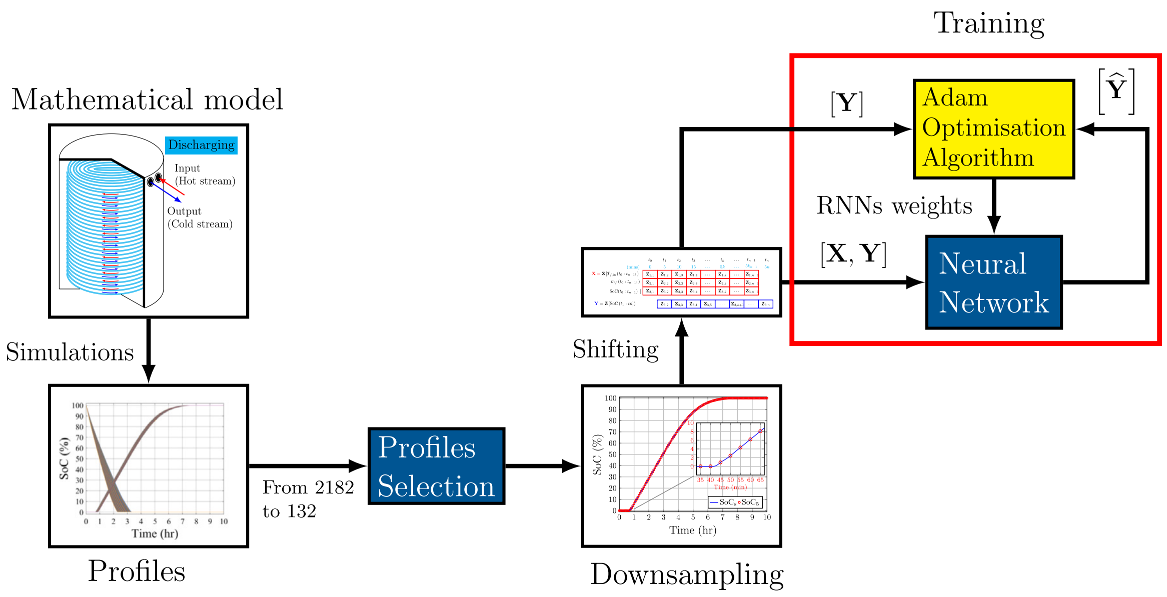


Fig. 15. Illustration of the complete RNN training process including dataset processing. In the training block within the red rectangle, $\hat{\mathbf{Y}}$ is a subset of $\mathbf{Y}$ used to verify the training process.

considering the HTF and PCM elements, this leads to an ice tank model of 80 ODEs and 80 algebraic functions for the discrete-time non-linear observer model. The observer requires sampled values of mass flow rate and input and output temperatures of the HTF for each tube. It also requires the estimated temperatures from the previous time-step. To achieve this, a delay was included in a feedback loop as illustrated in Fig. 16. Fig. 17 shows screenshots of the discrete-time observer configuration as implemented in MATLAB/Simulink. (**Note:** when alluding to a discrete-time observer, this refers to a dynamic structure operating in a discrete-time domain in a control engineering sense and not to a thermally discretised (or stratified) model of the ice tank.)

Fig. 18 shows the response of the discrete-time observer during a charging process. To better appreciate its performance, initial conditions were set at 0 °C and constant values for the input temperature and mass flow rate of the HTF were adopted ($T_{f,\mathrm{in}} = -6$ °C and $\dot{m}_f = 22$ kg/s). The system states being tracked, denoted with variable x and shown with dashed traces, correspond to the temperatures of the HTF and PCM at the different nodes, that is $T_{f,i,a} = x_{4i-3}$ and $T_{w,1} = x_{4i-2}$, where subscript 'f' stands for the HTF, 'w' for the PCM (water), and 'i' for the node number. A hat notation is used to identify the estimated states, shown with solid traces. Only temperatures at nodes 1, 11, and 20 of tube 'a' are explicitly provided for simplicity. These correspond to the following states: $T_{f,1,a} = x_1$, $T_{w,1,a} = x_2$, $T_{f,11,a} = x_{41}$, $T_{w,11,a} = x_{42}$, $T_{f,20,1} = x_{77}$ and $T_{w,20,1} = x_{78}$. As shown in

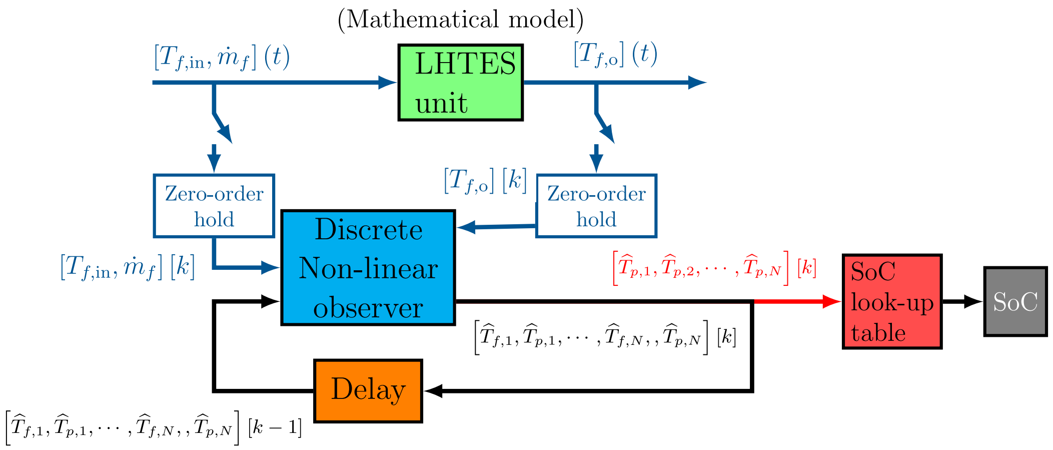

(Mathematical model)
$[T_{f,\mathrm{in}}, \dot{m}_f]\,(t)$
LHTES unit
$[T_{f,\mathrm{o}}]\,(t)$
Zero-order hold
Zero-order hold
$[T_{f,\mathrm{o}}]\,[k]$
Discrete Non-linear observer
$[T_{f,\mathrm{in}}, \dot{m}_f]\,[k]$
$[\widehat{T}_{p,1}, \widehat{T}_{p,2}, \cdots, \widehat{T}_{p,N}]\,[k]$
SoC look-up table
SoC
$[\widehat{T}_{f,1}, \widehat{T}_{p,1}, \cdots, \widehat{T}_{f,N}, \widehat{T}_{p,N}]\,[k]$
Delay
$[\widehat{T}_{f,1}, \widehat{T}_{p,1}, \cdots, \widehat{T}_{f,N}, \widehat{T}_{p,N}]\,[k-1]$

Fig. 16. Block diagram of the ice tank model, discrete-time non-linear observer, and SoC calculation using a look-up table. The estimation values from the previous time-step required for the calculation of the algebraic functions is obtained using a feedback loop with a delay over the observer output.

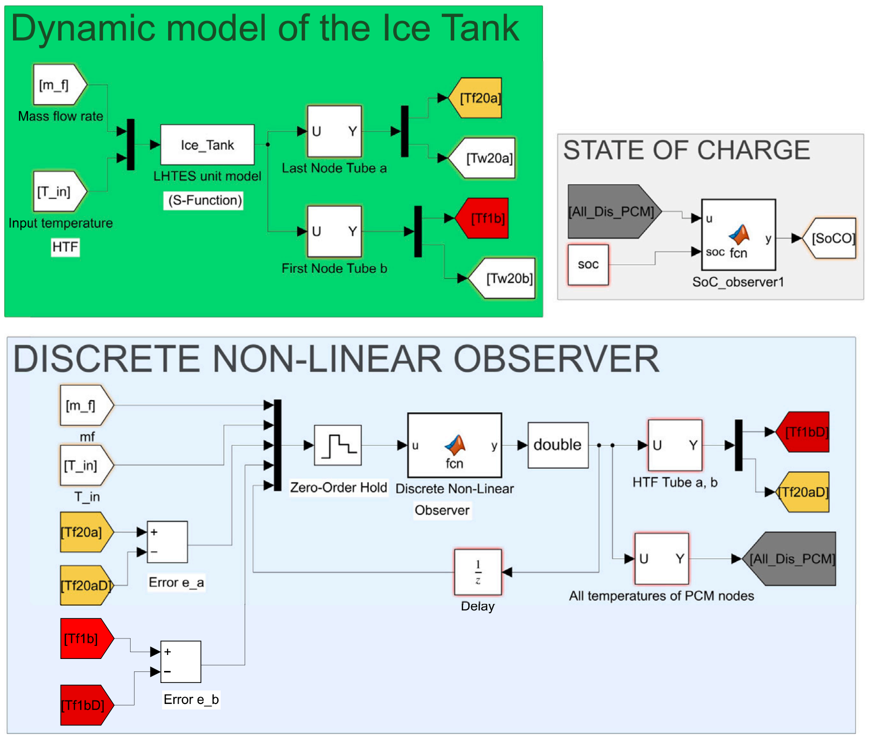


Fig. 17. Screenshot of the discrete-time observer implementation in MATLAB/Simulink: S-function of the one-dimensional model of the ice tank, MATLAB function with the code of the discrete-time non-linear observer, and SoC calculation using a look-up table as presented in [15].

Fig. 18(a), the observer tracks the system states accurately—mirroring the findings reported in [15] with a continuous-time observer. By the beginning of the simulation, the temperatures of the HTF exhibit oscillations. The temperatures of the PCM, in contrast, indicate a marginally slower response with smaller oscillations. This is shown more clearly in the zoomed-in graph within the figure. Fig. 18(b) demonstrates the observer's precision in estimating SoC ($\mathrm{SoC_{Do}}$, red trace) compared to that calculated directly from the system temperatures ($\mathrm{SoC_S}$, dashed blue trace). Both traces match well.

Fig. 18(c) and (d) show the system's estimation errors for all the discretised nodes. The observer estimates the states of the dynamic model of the ice tank, which are the temperatures of the control volumes. The estimation error shown in the figures is the deviation between these states and the estimations from the observer. As it can be seen, the observer exhibits a convergence to zero within about one min into the simulation for the node temperatures of the HTF and within 2.5 min for the temperatures of the PCM. For further clarity, Fig. 18(e) and (f) show the behaviour of the estimation errors for nodes 1 and 20 only during the beginning of the simulation. Given the slow thermal dynamics intrinsic to the ice tank, these convergence times are deemed acceptable. Like the continuous-time observer performance, a quicker error convergence to zero is achieved in HTF nodes near the HTF inlet, whereas PCM temperature estimation presents a slower but more uniform response.

As evidenced by Fig. 18(c)–(f), the estimation errors exhibit a fluctuating behaviour. To explain this, it is worth recalling that the observer aims to accurately estimate the states of a dynamic system represented by non-linear differential equations. The observer is also

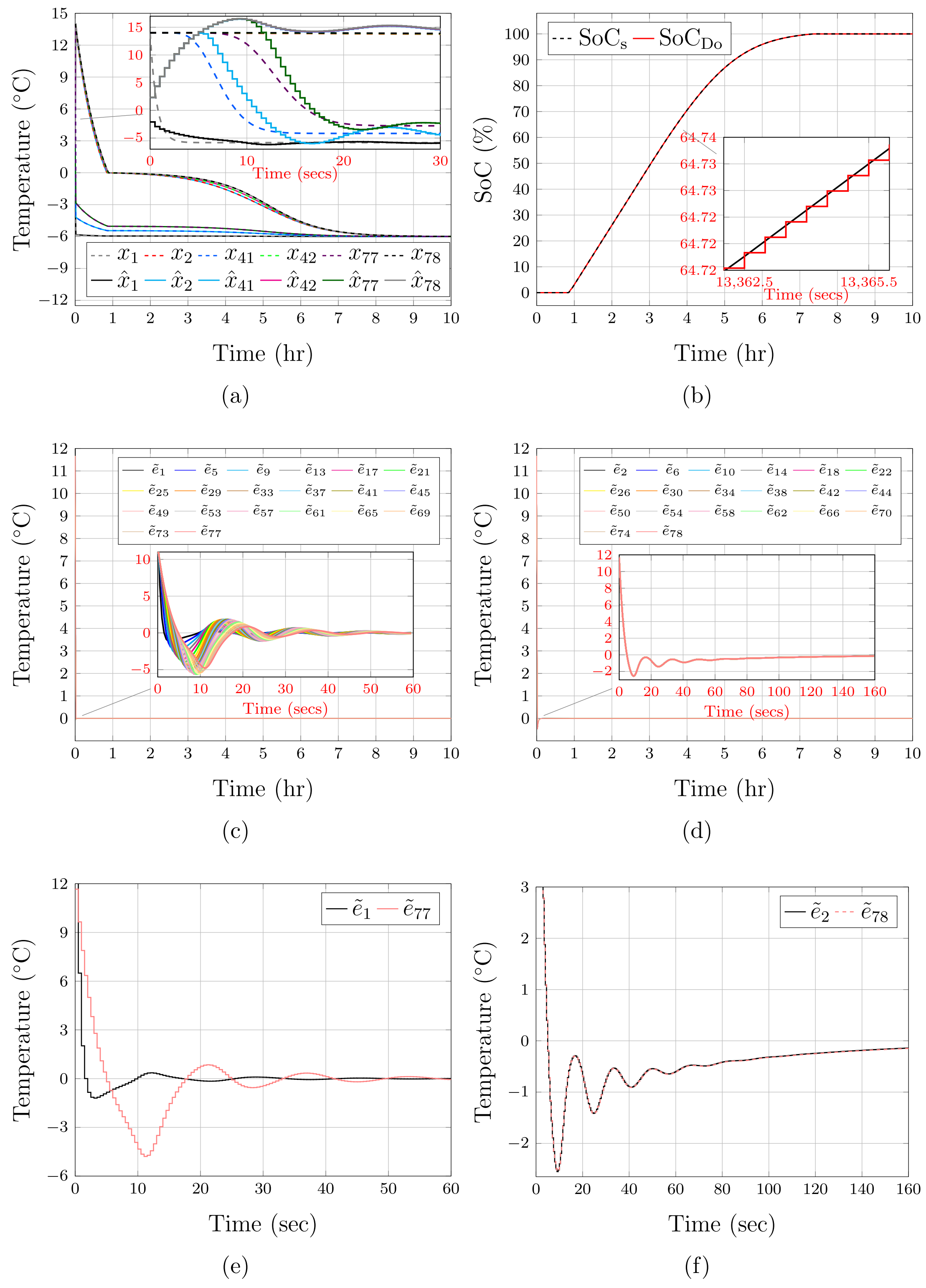


Fig. 18. Performance of the discrete-time non-linear observer during a charging process for constant operating conditions of the HTF ($\dot{m}_f$ = 22 kg/s and $T_{f,\text{in}}$ = −6 °C). (a) System states and estimation by the discrete-time non-linear observer (HTF and PCM temperatures are shown for nodes 1, 11 and 20). (b) Calculated SoC of the ice tank model (SoC$_S$) and using the discrete-time non-linear observer (SoC$_{Do}$). Estimation errors for all node temperatures (c) of the HTF and (d) the PCM. For the sake of clarity, the estimation errors for nodes 1 and 20 are presented in (e) for the HTF and in (f) for the PCM during the beginning of the simulations.

a non-linear system and both the ice tank model and the observer require initial conditions to initialise the simulations. For the observer states these were set to 0 °C, while the initial conditions for a charging process were set to 14 °C. Due to this discrepancy, the observer requires time to converge on a zero-estimation error, as shown by the results and, during the transitory period, fluctuations dictated by the assigned dynamics of the observer are exhibited.

Fig. 19 presents the performance of the discrete-time non-linear observer during a discharging process. In this operation mode, constant values of $\dot{m}_f$ = 5 kg/s and $T_{f,\text{in}}$ = 12 °C were used. The initial

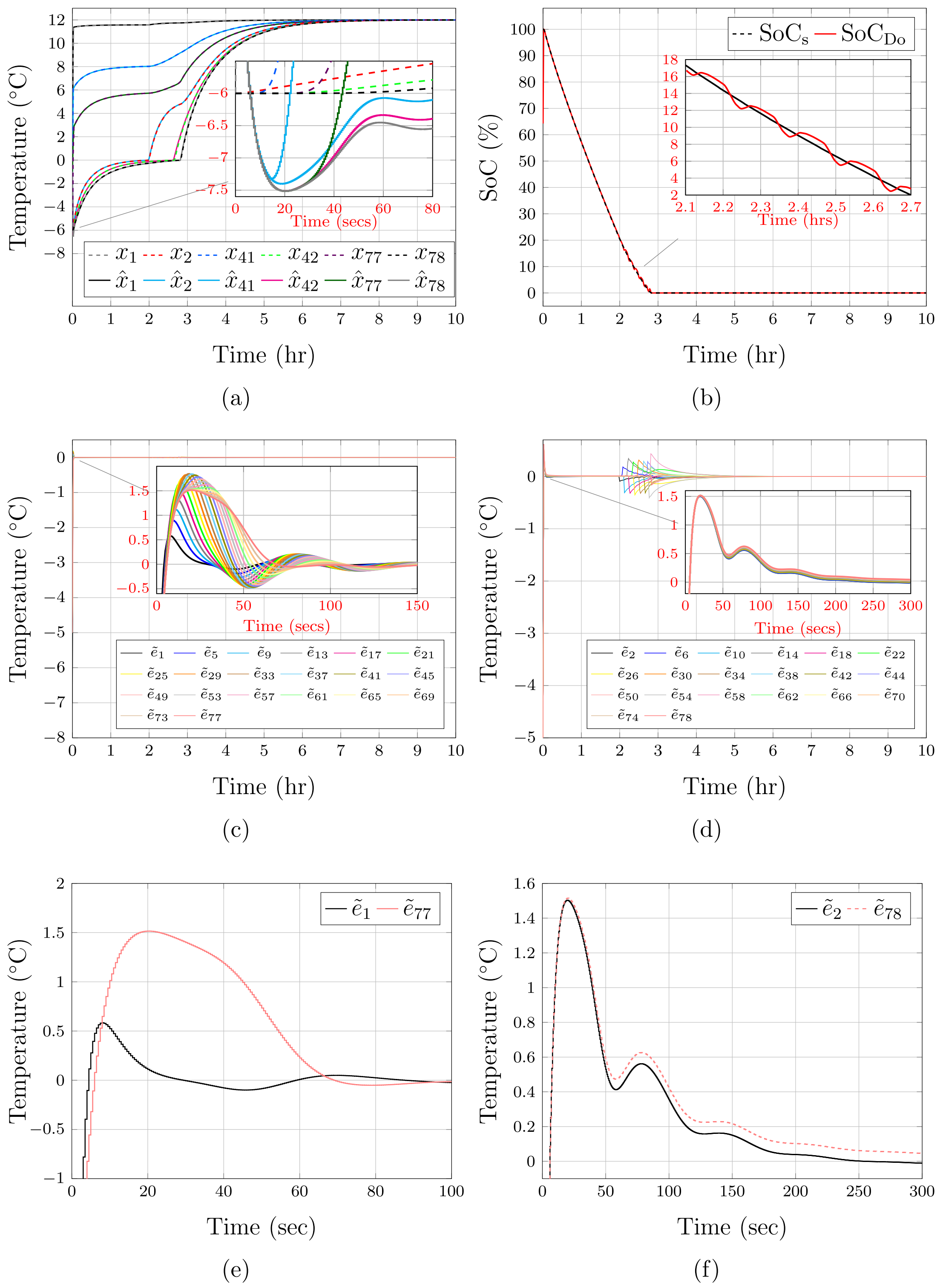


Fig. 19. Performance of the discrete-time non-linear observer during a discharging process for constant operating conditions of the HTF ($\dot{m}_f = 5$ kg/s and $T_{f,\text{in}} = 12$ °C). (a) System states and estimation by the discrete-time non-linear observer (HTF and PCM temperatures are shown for nodes 1, 11 and 20). (b) Calculated SoC of the ice tank model (SoC_S) and using the discrete-time non-linear observer (SoC_Do). Estimation errors for all node temperatures (c) of the HTF and (d) the PCM. For the sake of clarity, the estimation errors for nodes 1 and 20 are presented in (e) for the HTF and in (f) for the PCM during the beginning of the simulation.

conditions of the observer were also set as 0 °C, while for the ice tank these were defined as −6 °C. The temperatures of nodes 1, 11, and 20 are shown in Fig. 19(a), demonstrating the capabilities of the discrete-time observer to accurately estimate the node temperatures of the HTF and PCM. This is further evidenced by Fig. 19(c) and (d), where the estimation errors for all nodes converge to zero following minor oscillations albeit with a slightly slower response compared to the charging process: approximately 2.5 min for the HTF and around 5 min for the PCM. This is further evidenced by Fig. 19(e) and (f), which present the behaviour of the estimation errors for nodes 1 and 20 only by the start of the discharging process. As shown by Fig. 19(b), the estimated SoC (SoC_Do, red trace) is comparable to that calculated

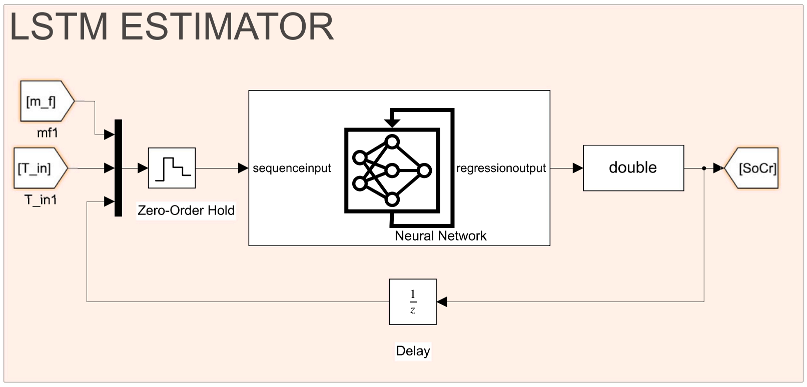


Fig. 20. Screenshot of the implementation in MATLAB/Simulink of the trained RNN of the estimation of SoC.

directly from the system temperatures (SoC_S, blue trace). Therefore, it can be concluded that the observer maintains a consistently good performance for both charging and discharging processes.

Note: To achieve convergence for the discrete-time non-linear observer when solving the 80 algebraic equations and accurately estimate SoC, the sampling time was established at 0.5 s. Sampling times exceeding this value presented convergence problems.

5.2. RNN-based SoC estimator

The trained RNN was stored in a *mat* file within MATLAB for subsequent deployment using commands of the deep learning toolbox. This *mat* file considers the weights and biases for all the layers of the RNNs. The MATLAB/Simulink function '*Stateful predict*' from the toolbox was used to call the trained RNN. The delay in the RNN's output was afforded with a delay block, while a zero-order hold (ZOH) is applied to the three inputs of the RNN estimator as shown in Fig. 9. The sample time for both functions was set to 5 min. Fig. 20 provides a screenshot of this implementation.

Simulations of the 8 different RNN configurations as discussed by the end of Section 4.4 were conducted in parallel with the mathematical model of the ice tank. To assess their performance 20 profiles were used: 10 for charging and 10 for discharging. The profiles were chosen randomly and were different from those employed for training. To clearly identify the 8 RNNs, the subscript notation 'g, m' was adopted, where 'g' denotes the number of LSTM layers and 'm' the number of neurons.

Simulation results for the different charging and discharging processes are shown in Fig. 21, where the SoC derived from system temperatures is presented alongside the responses of the RNN configurations. An error analysis was undertaken to numerically quantify the agreement between the mathematical model and RNN estimator. This analysis entailed the calculation of the RMSE and the MAE for the estimated SoC (SoC_r) relative to the SoC determined by the system temperatures (SoC_S).

Table 4 shows the mean values of the errors ($RMSE_{ave}$ and MAE_{ave}) from the ten simulations for charging and the ten simulations for discharging. Overall, seven RNN structures exhibit an excellent performance, with their RMSE and MAE values falling below 1% for charging. For discharging the performance is comparable, with the RMSE always less than 2% and the MAE less than 1.04%. However, $RNN_{[2,10]}$ is an exception, as observed by the brown trace in Fig. 21. This RNN architecture inadequately estimates the SoC for both charging and discharging and particularly leads to a larger estimation error for the charging process.

In general, a slight increase in both error values for the SoC estimation of a charging process is observed. Nonetheless, $RNN_{[1,10]}$ and $RNN_{[2,15]}$ exhibit a superior accuracy for both processes with reduced RMSE and MAE values. Owing to their superior estimation performance, these two RNN architectures were selected for comparison with the discrete-time non-linear observer in Section 5.3.

5.3. Comparison between the discrete-time observer and RNN-based SoC estimators

Both types of SoC estimators were assessed during charging and discharging cycles of the ice tank upon variations of $\dot{m}_f$ and $T_{f,\text{in}}$ as shown by Fig. 22(a) and (b). The discharging processes take place by the beginning and the end of the simulations. For the first discharging operation, $\dot{m}_f = 5$ kg/s and $T_{f,\text{in}} = 10$ °C, whereas $\dot{m}_f = 5.5$ kg/s and $T_{f,\text{in}} = 13$ °C for the second discharging run. The charging process at the middle of the simulations was run with $\dot{m}_f = 18.5$ kg/s and $T_{f,\text{in}} = -6$ °C. All simulations were carried out using MATLAB R2023a and a central processing unit with an AMD Ryzen 7 7730U CPU @ 2.00 GHz.

Fig. 23 shows the simulation results for the conditions shown in Fig. 22. The direct SoC calculation using the PCM temperatures of the ice tank model (SoC_S, see the cyan trace with the circular marker) is plotted alongside the estimations obtained with the RNNs ($SoC_{1,10}$ with black trace, and $SoC_{2,15}$ with red trace) using the MATLAB/Simulink function block '*Stateful predict*'. As discussed towards the end of Section 5.2, only two RNN architectures are considered for this comparison. The estimation of SoC obtained by the discrete-time non-linear observer (SoC_o, blue trace) is also included.

As shown in Fig. 23, the use of the '*Stateful predict*' block, which relies solely on a *mat* file with weight values for all layers of the RNN, results in significant errors in SoC estimation following the completion of the first discharging process (see the black and red traces). The estimation provided by $RNN_{[1,10]}$, $SoC_{1,10}$ with the black trace, exhibits a substantial discrepancy compared to the direct SoC calculation (SoC_S) at the onset of the charging process approximately 10 h into the simulation. This leads to values of SoC approximately 10% higher than for the rest of the estimations. Similarly, the estimation by $RNN_{[2,15]}$, $SoC_{2,15}$ with the red trace, exhibits an erratic performance during the second discharging process. This includes a high peak by the start of the process at around 22 h into the simulation as shown in the zoomed-in graph. This is followed by a wrong SoC estimation with values 10% higher than the system estimation SoC_S.

The previous results deserve additional discussion. Initially the RNN estimations using the '*Stateful predict*' block provide accurate results during the first discharging operation, but accuracy dramatically reduces for the subsequent processes. This behaviour is attributed to residual information within the hidden and cell states (h, c) given that the RNNs were trained independently for charging and discharging (see Section 5.2). Therefore, the information retained in these states is not needed when a new process begins, requiring a reset of the states. However, the use of the '*Stateful predict*' block function does not permit access to this information when running a simulation. This limitation was circumvented by an additional RNN implementation based on operation matrices and activation functions explicitly using the equations in Section 3.

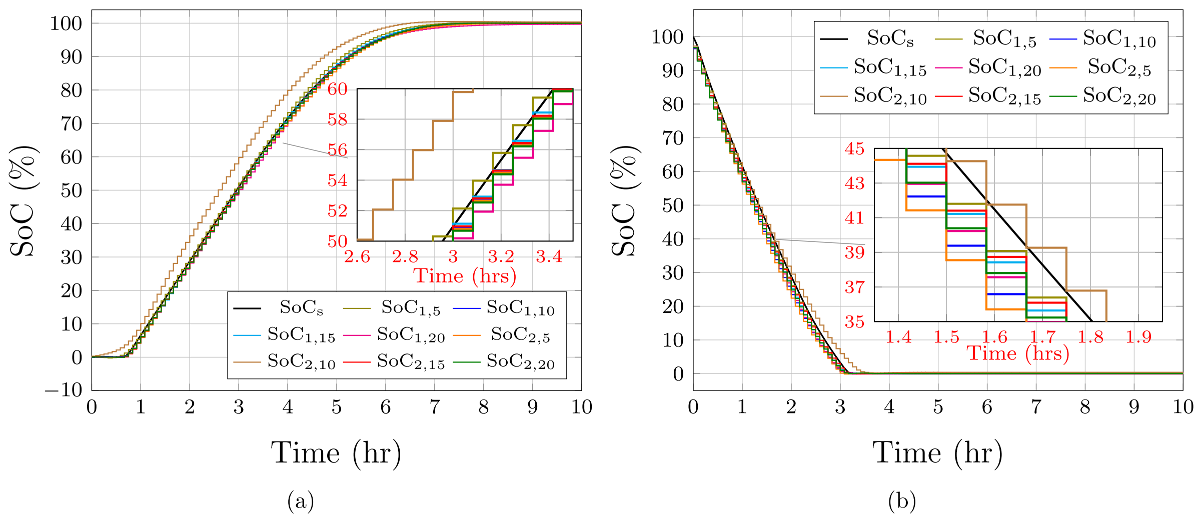


Fig. 21. Comparison of the SoC calculated from the ice tank model (SoC$_S$) and the estimated values provided by the eight different configurations of the RNN estimators (SoC$_{g,m}$) for: (a) charging and (b) discharging.

Table 4
RMSE and MAE of the direct calculation of SoC and estimated SoC given by the eight configurations of the RNN-based estimator.

RNN configurations (RNN$_{[g,m]}$)																
Error	Charging								Discharging							
	RNN [1, 5]	RNN [1, 10]	RNN [1, 15]	RNN [1, 20]	RNN [2, 5]	RNN [2, 10]	RNN [2, 15]	RNN [2, 20]	RNN [1, 5]	RNN [1, 10]	RNN [1, 15]	RNN [1, 20]	RNN [2, 5]	RNN [2, 10]	RNN [2, 15]	RNN [2, 20]
RMSE$_{ave}$	0.88	0.45	0.53	0.43	0.46	2.41	0.47	0.52	1.72	1.85	1.68	1.95	1.97	1.18	1.68	1.79
MAE$_{ave}$	0.66	0.32	0.40	0.34	0.34	1.86	0.34	0.38	0.90	0.97	0.87	1.03	1.02	0.78	0.87	0.90

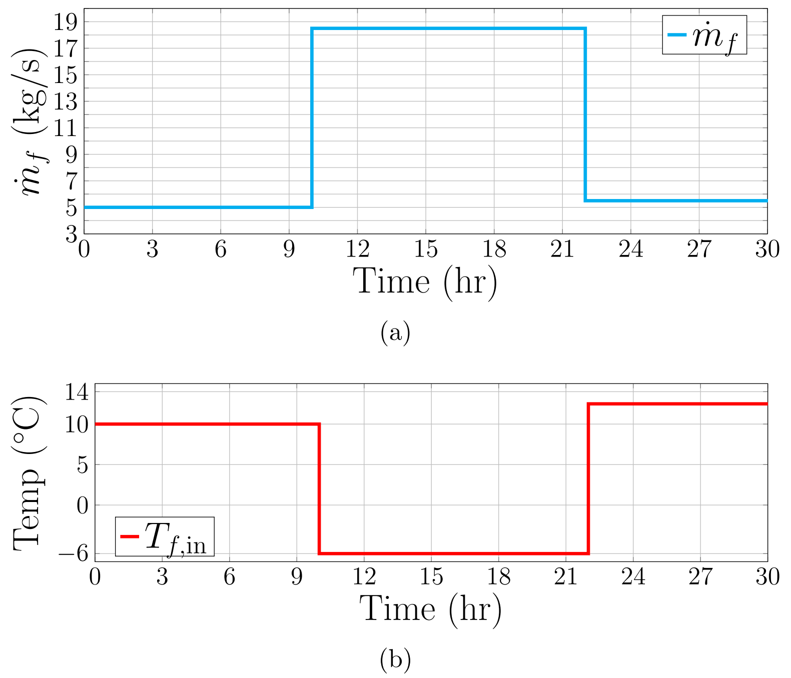


Fig. 22. Simulation conditions to compare the performance of the SoC estimators: (a) Mass flow rate of the HTF; (b) input temperature of the HTF. These consider two discharging processes and one charging process of the ice tank.

For the alternative RNN implementation just described, the weights and biases derived from the RNN estimator training were used. The MATLAB deep learning toolbox provides all this information for each layer of the network. For further insight into these data, Table 5 shows the dimensions of the matrices for each layer of the investigated architectures shown in Fig. 6. The dimensions vary according to the number of neurons adopted. For instance, in the LSTM layer of architecture RNN$_{[2,15]}$, the hidden state weight matrix has a dimension $[60 \times 15]$. Here, 15 is the number of neurons and 60 represents the sum of rows of the weight matrix used to calculate the input and forget gates, the candidate cell gate, and the output gate.

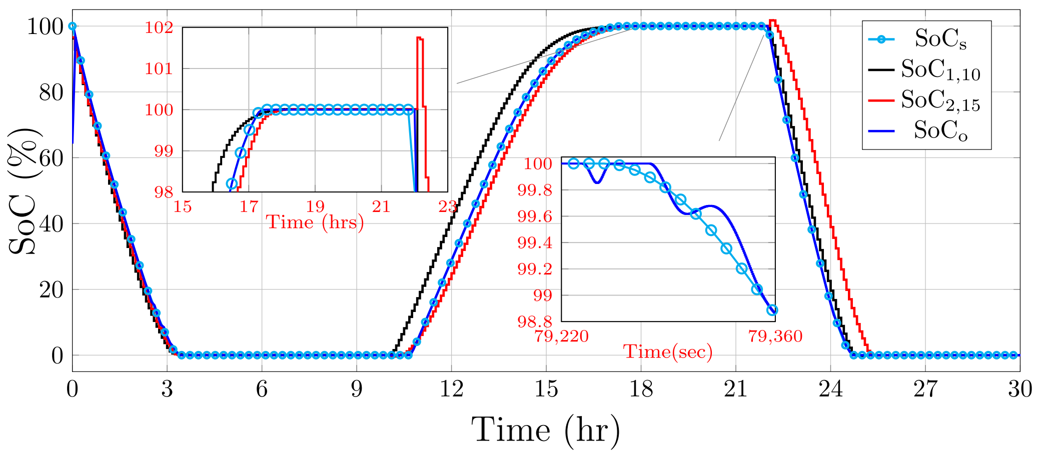


Fig. 23. Comparison between the SoC calculated with the continuous-time non-linear model of the ice tank (SoC$_S$) and the estimated SoCs given by the RNNs (SoC$_{g,m}$) and the discrete-time non-linear observer (SoC$_o$).

Table 5
Dimensions of weights and bias of all layers for RNN$_{[1,10]}$ and RNN$_{[2,15]}$.

Layer	Parameter	Dimensions	
		RNN$_{[1,10]}$	RNN$_{[2,15]}$
1st fully connected layer	W_2, b_2	$[10 \times 3], [10 \times 1]$	$[15 \times 3], [15 \times 1]$
1st LSTM layer	$W_4 = [W_{i,4}; W_{f,4}; W_{c,4}; W_{o,4}]$	$[40 \times 10]$	$[60 \times 15]$
	$U_4 = [U_{i,4}; U_{f,4}; U_{c,4}; U_{o,4}]$	$[40 \times 10]$	$[60 \times 15]$
	$b_4 = [b_{i,4}; b_{f,4}; b_{c,4}; b_{o,4}]$	$[40 \times 1]$	$[60 \times 1]$
2nd LSTM layer	$W_5 = [W_{i,5}; W_{f,5}; W_{c,5}; W_{o,5}]$	–	$[60 \times 15]$
	$U_5 = [U_{i,5}; U_{f,5}; U_{c,5}; U_{o,5}]$		$[60 \times 15]$
	$b_5 = [b_{i,5}; b_{f,5}; b_{c,5}; b_{o,5}]$		$[60 \times 1]$
2nd fully connected layer	W_7, b_7	$[1 \times 10], [1 \times 1]$	$[1 \times 15], [1 \times 1]$

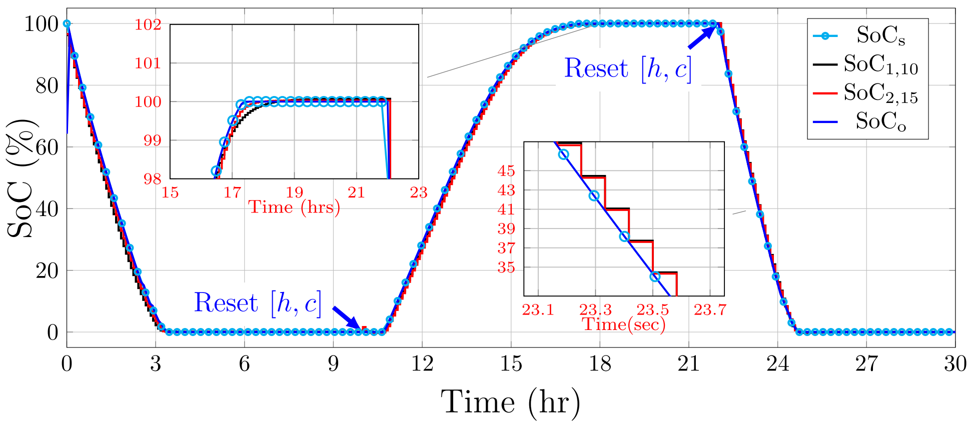


Fig. 24. Comparison between the SoC calculated with the continuous-time non-linear model of the ice tank (SoC$_S$) and the estimated SoCs given by the RNNs (SoC$_{g,m}$) employing matrix operations.

Table 6 shows the pseudo-code for an RNN estimator function built in MATLAB/Simulink. The matrix operations and activation functions are embedded within a '*for*' loop which, in turn, is executed every five minutes. This is consistent with the dataset used for training. To prevent the issues arising with the '*Stateful predict*' block, a reset is implemented at the onset of each charging and discharging process. Fig. 24 shows the results of this implementation. The state resets are indicated by navy blue arrows for clarity. As opposed to the results shown in Fig. 23, there is a notable improvement in the SoC estimation for both configurations, namely RNN$_{[1,10]}$ (SoC$_{1,10}$, black trace) and RNN$_{[2,15]}$ (SoC$_{2,15}$, red trace).

The performance of the discrete-time non-linear observer is noteworthy for its high accuracy in SoC estimation. Results are also shown in Figs. 23 and 24 for a direct comparison with the RNN architectures. At the beginning of the simulation, the discrete-time observer exhibits a minor divergence due to the initial conditions of its states (set at 0 °C). The estimation process requires ~150 s to reach a zero estimation error. This is further appreciated in the zoomed-in graph within Fig. 23.

Table 7 shows a comparison of the computation time required for the execution of each step in the SoC estimation process. These values were also recorded using an AMD Ryzen 7 7730U CPU @ 2.00 GHz and do not consider the time required to solve the non-linear differential equations representing the dynamic model of the ice tank. (**Note**: a comprehensive analysis on the run time for solving the ice tank model used in this paper is already available in [29] and is not here accounted for to restrict the discussion to SoC estimation only.) Each

Table 6
Pseudo-code of the RNN estimator implemented in MATLAB/Simulink.

```
Pseudo-code of RNN estimator

1: Input: $\mathbf{Z} = \{T_{f,\mathrm{in},t-1}, \dot{m}_{f,t-1}, \mathrm{SoC}_{\mathrm{RNN},t-1}\}$
2: Given parameters: $W_2, b_2, W_4, U_4, b_4, W_5, U_5, b_5, W_7, b_7$
3: Initialise $h_4, c_4, h_5, c_5 = 0$
4: for $t = 1, \dots, t_{tot}$ do
5:        Check if a new process will start to reset $h_4, c_4, h_5, c_5 = 0$
6:        otherwise the values remain as $h_4 = Z_4, c_4 = c_{t,4}, h_5 = Z_5, c_5 = c_{t,5}$
7:        Normalise inputs $Z$ (rescale 0 to 1)
8:        Calculate the first full connection $Z_2 = W_2 Z + b_2$
9:        Calculate first ReLU layer $Z_3 = Z_2(\mathrm{id}) = 0$ with $\mathrm{id} = Z_2 < 0$ for first LSTM layer
10:       Calculate $i_{t,4}$ (Eq. (8)), $f_{t,4}$ (Eq. (9)), $\tilde{c}_{t,4}$ (Eq. (10)), with $x_4 = Z_3$
11:       Calculate $o_{t,4}$ (Eq. (11))
12:       Update cell state $c_{t,4}$ (Eq. (12))
13:       Calculate $h_4$ (Eq. (13))
14:       Update hidden and cell state $c_4 = c_{t,4}, Z_4 = h_4$ for second LSTM layer
15:       Calculate $i_{t,5}$ (Eq. (8)), $f_{t,5}$ (Eq. (9)), $\tilde{c}_{t,5}$ (Eq. (10)), with $x_5 = Z_4$
16:       Calculate $o_{t,5}$ (Eq. (11))
17:       Update cell state $c_{t,5}$ (Eq. (12))
18:       Calculate $h_5$ (Eq. (13))
19:       Update hidden and cell state $c_5 = c_{t,5}, Z_5 = h_5$
20:       Calculate second ReLU layer $Z_6 = Z_5(\mathrm{id}) = 0$ with $\mathrm{id} = Z_5 < 0$
21:       Calculate the second full connection $Z_t = W_7 Z_6 + b_7$
22: end for
23: Output: SoC= $[Z_1, Z_2, \dots, Z_{tot}]$
```

Table 7
Run time in μs employed by the RNNs and discrete-time non-linear observer to compute each time-step for SoC estimation.

Average run time for each step (μs)		
Discrete-time observer	$\mathrm{RNN}_{[1,10]}$	$\mathrm{RNN}_{[2,15]}$
∼284	∼12	∼21

solution step for the 80 algebraic equations describing the discrete-time non-linear observer demands an average run time of ∼284 μs when a sampling time of 0.5 s is used. In contrast, the matrix operations and execution of activation functions only require ∼12 μs for $\mathrm{RNN}_{[1,10]}$ and ∼21 μs for $\mathrm{RNN}_{[2,15]}$ (with a sampling time of 5 min for either RNN structure). Thus, the reduced run time and larger sampling time afforded by the RNN estimators may be critical parameters for practical implementation when the computation resources are constrained—such as with basic microcontrollers.

For further assessment of the SoC estimators, a cycle of five charging–discharging operations was simulated. Fig. 25 shows the conditions for the HTF ($\dot{m}_f$ and $T_{f,\mathrm{in}}$). During the charging processes, $\dot{m}_f$ ranged from 18.5 kg/s to 22 kg/s and $T_{f,\mathrm{in}}$ remained constant at −6 °C. For the discharging processes, $\dot{m}_f$ varied between 5 kg/s and 6 kg/s and $T_{f,\mathrm{in}}$ between 10 °C and 14 °C. Fig. 26 presents the simulation results. As observed, both the RNNs (black and red traces with circle markers) and the discrete-time non-linear observer (navy blue trace with diamond marker) exhibit a good estimation performance when compared to the SoC calculation obtained from the ice tank model (cyan trace with the circle marker). It is evident that by resetting the states of the RNNs their performance is significantly enhanced.

To assess the accuracy of both the discrete-time non-linear observer and the RNN estimator, an error analysis was conducted by quantifying the RMSEs and MAEs of the estimated values against those directly obtained with the ice tank model. This exercise was conducted for the complete simulation of charging and discharging cycles. A summary of this error quantification is shown in Table 8. The discrete-time non-linear observer exhibits the lowest values of RMSE (0.7159%) and MAE (0.0434%). This was expected as this control structure achieves a high accuracy at the expense of an increased computational cost, performing 864,000 estimations during the simulated 120 h. In contrast, larger errors were exhibited by the RNN estimators, as shown in the table. These are however deemed acceptable considering the significantly faster

Table 8
RMSE and MAE of estimated SoC by RNNs and non-linear observer with respect to the SoC directly calculated from the simulated values of the PCM temperatures.

	Discrete-time observer	$\mathrm{RNN}_{[1,10]}$	$\mathrm{RNN}_{[2,15]}$
RMSE (%)	0.7159	1.1160	0.9955
MAE (%)	0.0434	0.4357	0.3015

Table 9
RMSE and MAE values for the estimated SoC using the discrete-time non-linear observer when compared to the directly calculated SoC over 25 charging–discharging cycles.

	Discrete-time observer	$\mathrm{RNN}_{[1,10]}$	$\mathrm{RNN}_{[2,15]}$
RMSE (%)	0.0844	0.7294	0.5335
MAE (%)	0.0146	0.4032	0.2610

computational processing (see Table 7), with $\mathrm{RNN}_{[1,10]}$ and $\mathrm{RNN}_{[2,15]}$ performing a total of 1440 estimations each in the 120-h simulation. Despite these minor discrepancies, the performance of all estimators is deemed successful as their RMSE and MAE values fall below 1.12%.

For completeness, 25 additional charging–discharging cycles were simulated to provide further evidence on the accuracy afforded by the RNN estimators. These simulations include variations in the initial conditions for charging, ranging from 10 °C to 14 °C in increments of 1 °C for the input temperature of the HTF, and from 18 to 22 kg/s in increments of 1 kg/s for the mass flow rate. In contrast, step increments of 1 °C from 10 °C to 14 °C for the HTF input temperature and of 0.25 kg/s from 5 kg/s to 6 kg/s for the mass flow rate were used for discharging. Results for these extended simulations are shown in Fig. 27. An error analysis of the SoC estimation using the discrete-time observer and the RNNs with respect to the directly calculated SoC is summarised in Table 9.

Fig. 27(a) shows the mass flow rate of the HTF for the 25 additional charging–discharging cycles, while Fig. 27(b) shows the input temperature conditions of the HTF. As it can be observed, as in the simulation results presented in Fig. 26, a good SoC estimation performance is achieved by the discrete-time observer and the RNNs throughout the simulation. As shown by Table 9, the observer yields the most accurate SoC estimation and exhibits the lowest RMSE and MAE, with a total of 3,960,000 estimations over the 550-h simulation. Among the RNN estimators, $\mathrm{RNN}_{[2,15]}$ exhibits slightly lower RMSE and MAE values than $\mathrm{RNN}_{[1,10]}$, with each conducting a total of 6600 estimations.

5.4. Influence of the sampling time-step in training, performance accuracy, and computational cost of RNN-based SoC estimators

Sections 4.3 and 4.4 illustrate the practical implications of adopting a specific sampling time in the training process of the RNN-based estimators. In turn, the sampling time also impacts the training time and the performance accuracy of the RNNs. A rigorous analysis of the optimal sampling time is out of the scope of this work. However, a comparative analysis is carried out in this section to illustrate the impact of using different sampling times in the duration of training, accuracy, and computational cost of the studied RNN-based estimators.

Both SoC estimators (i.e. $\mathrm{RNN}_{[1,10]}$ and $\mathrm{RNN}_{[2,15]}$) were trained using sampling times of 120 s and 600 s (2 min and 10 min) in addition to the 300 s (5 min) previously used in Sections 5.2 and 5.3. Compared to the 120 data points in the training profiles required with a 300-s sampling time, the adjustments result in an increment to 300 data points for the 120-s sampling time and a decrement to 60 data points for the 600-s sampling time.

Table 10 shows the training times for the RNN estimators under different sampling times. Using the 300-s sampling time as a reference, the training time for RNNs with a 120-s sampling time increased to 34 min (from 16 min) for $\mathrm{RNN}_{[1,10]}$ and 71 min (from 27 min) for $\mathrm{RNN}_{[2,15]}$. The training time also decreased to 10 min for $\mathrm{RNN}_{[1,10]}$ and 16 min for $\mathrm{RNN}_{[2,15]}$ when the sampling time was 600 s.

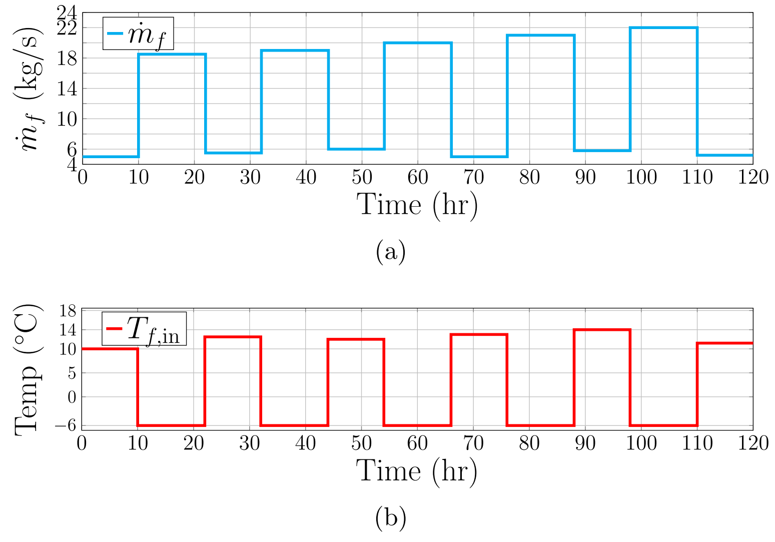


Fig. 25. (a) Mass flow rate and (b) input temperature profiles of the HTF for a series of five discharging and charging cycles of the ice tank.

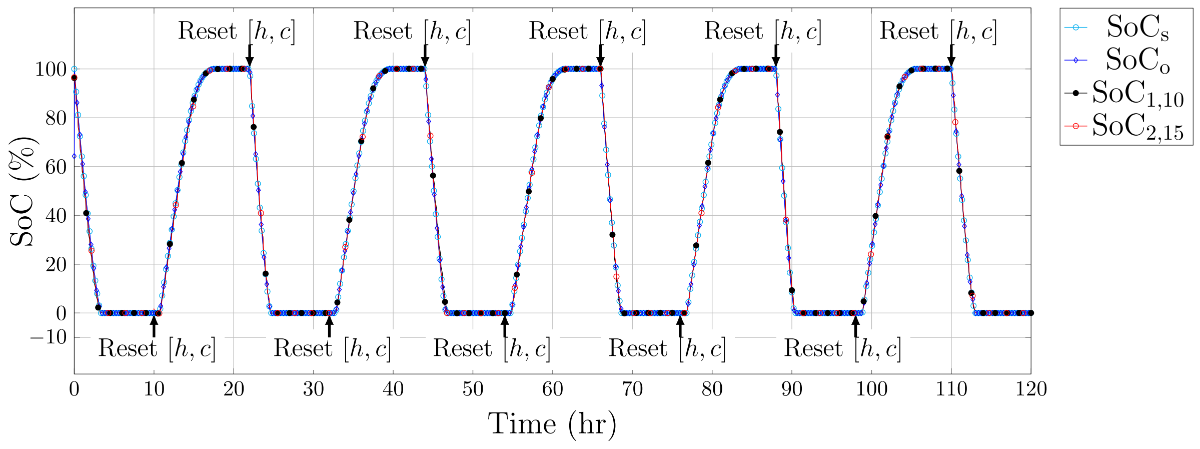


Fig. 26. Comparison between the SoC calculated from the temperature of the ice tank and the estimated SoCs given by the RNNs and the discrete-time non-linear observer for five discharging–charging cycles.

Table 10
Time required for the training using different sampling times (T_s).

	T_s = 120 s	T_s = 300 s	T_s = 600 s
$\mathrm{RNN}_{[1,10]}$	34 min	16 min	10 min
$\mathrm{RNN}_{[2,15]}$	71 min	27 min	16 min

The RNN-based estimators trained under the additional sampling times were simulated and results compared against those obtained with estimators with a 300-s sampling time (presented in Section 5.3) to evaluate their performance accuracy. To this end, the 30-h discharging–charging–discharging cycle shown in Fig. 22 was adopted. The comparison of simulation results is shown in Fig. 28, where subscript ‘A’ was used for results obtained with a sampling time of 120 s and subscript ‘B’ for a sampling time of 600 s.

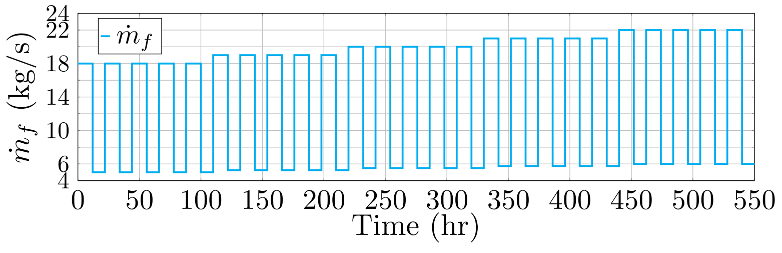
RNNs trained with a 120-s sampling time (see black and orange traces) show an enhanced accuracy, while RNNs trained with a 600-s sampling time (red and teal traces) exhibit a decreased accuracy. To support these observations, Table 11 shows the specific RMSEs and MAEs of the direct calculation of SoC and estimated SoC. It is evident that the accuracy afforded by both $\mathrm{RNN}_{[1,10]}$ and $\mathrm{RNN}_{[2,15]}$ increases as the sampling time is reduced—at the expense of longer training times as shown in Table 10.

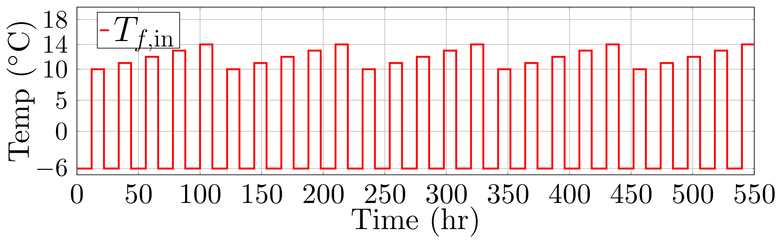
The average run time was also analysed for both RNN structures under different sampling times, with results provided in Table 12. However, as expected, no variations were observed in run time for a given RNN as the same activation functions and matrix operations are used to estimate SoC. This calculation procedure is detailed in Section 5.3 and it is not affected by sampling time. The only differences between RNNs with the same structure and different sampling times are the weights of the layers.

The relatively short durations of the training processes, along with the unchanged efficiency of calculations of the RNN estimators due to sampling time, indicate that selecting the sampling time involves a trade-off only between the available computational resources and desired accuracy. For simplicity, a sampling time of 300 s was chosen as suitable in this paper because it ensures a reasonable training duration without compromising accuracy, facilitating rapid evaluation of different structures (see Section 5.2 and Table 4).

6. Discussion

As shown in Section 5.3, the comparative analysis of the performance of SoC estimators for an ice tank based on either a discrete-time non-linear observer or an RNN indicates a high estimation accuracy for both approaches. RMSE and MAE values below 1% were obtained for both. However, the computational efficiency of these methods varies


(a)


(b)

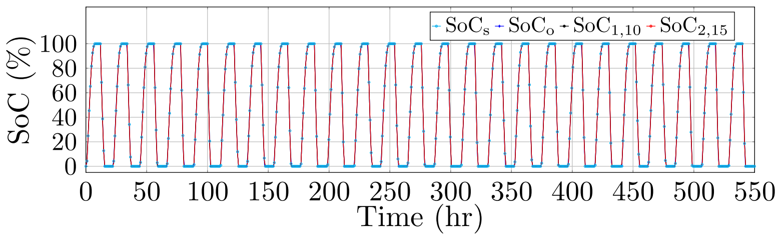

(c)

Fig. 27. (a) Mass flow rate profile of HTF ($\dot{m}_f$) with variations from 10 kg/s to 22 kg/s in steps of 1 kg/s for charging and from 5 kg/s to 6 kg/s in steps of 0.25 kg/s for discharging. (b) Input temperature profile ($T_{f,\mathrm{in}}$) with variations from 10 °C to 14 °C in steps of 1 °C for discharging a constant value of −6 °C for charging. (c) Comparison of the SoC obtained from the temperature of the ice tank (light blue trace, $\mathrm{SoC_S}$), the discrete-time non-linear observer (dark blue, $\mathrm{SoC_O}$), and the RNN estimators (black, $\mathrm{SoC}_{[1,10]}$, and red, $\mathrm{SoC}_{[2,15]}$) under different operating conditions which include variations in the initial conditions, input temperature, and mass flow rate of the HTF.

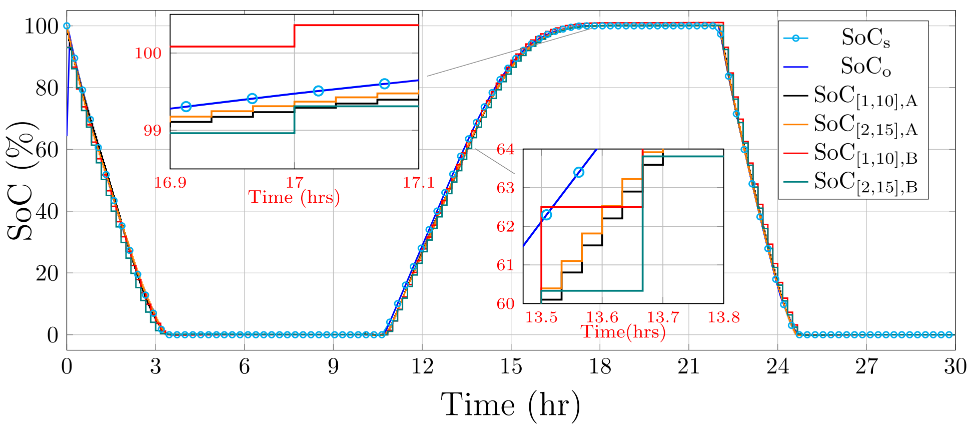


Fig. 28. Comparison between the SoC calculated with the continuous-time non-linear model of the ice tank (SoCs) and the estimated SoCs given by the RNNs with different sampling times. Subscripts ‘A’ and ‘B’ respectively stand for a sampling time of 120 s and 600 s.

Table 11
RMSE and MAE of the direct calculation of SoC and estimated SoC given by $\mathrm{RNN}_{[1,10]}$ and $\mathrm{RNN}_{[2,15]}$ with sampling times (T_s) of 120 s, 300 s and 600 s.

	$\mathrm{RNN}_{[1,10]}$			$\mathrm{RNN}_{[2,15]}$		
	T_s = 120 s	T_s = 300 s	T_s = 600 s	T_s = 120 s	T_s = 300 s	T_s = 600 s
RMSE (%)	0.73	1.71	1.79	0.75	1.29	2.20
RMSE (%)	0.41	0.85	0.98	0.38	0.67	1.02

Table 12
Average run time in μs to compute SoC estimation of $RNN_{[1,10]}$ and $RNN_{[2,15]}$ with sampling times (T_s) of 120 s, 300 s and 600 s.

Average run time for each step (μs)					
$RNN_{[1,10]}$			$RNN_{[2,15]}$		
T_s = 120 s	T_s = 300 s	T_s = 600 s	T_s = 120 s	T_s = 300 s	T_s = 600 s
~12	~12	~12	~21	~21	~21

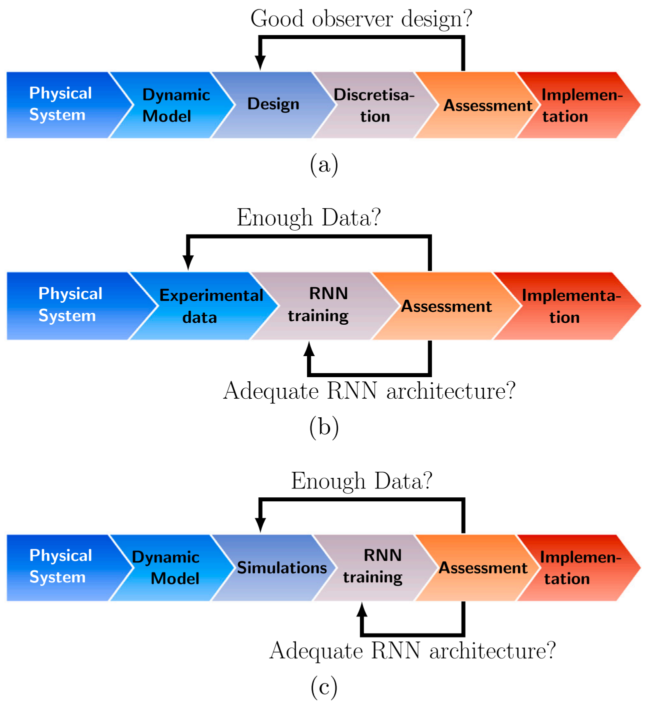


Fig. 29. Development diagram for the (a) discrete-time non-linear observer, (b) RNN estimator using experimental data, and (c) mathematical model for simulations.

substantially. While the discrete-time non-linear observer takes 284 μs to produce a SoC estimation and requires a sampling time of 0.5 s to ensure convergence, the examined RNN estimators take 12 and 21 μs to estimate SoC every 300 s. Considering this may thus aid in their selection and implementation in embedded systems.

The discrete-time non-linear observer requires a verified mathematical model of the LHTES unit under consideration—in this case a thermally discretised ice tank. Additionally, a comprehensive understanding of the thermophysical properties of the HTF, PCM, and the tube materials where the HTF circulates is essential. Assuming this information is readily available, the design and implementation of the discrete-time observer may be conducted by following the steps presented in this paper (see Appendix A and [15] for further information). In contrast, the RNN estimator requires experimental data from a practical system or simulation data derived from an accurate (in this case verified) mathematical model. Once the architecture of the RNN is established, the training process can be completed within minutes using MATLAB's machine learning toolbox. A summary of the development process for both estimators is presented in Fig. 29.

Although the development of both estimators can be achieved as long as sufficient information is available and the methodologies presented in this paper are followed, the critical aspect lies in their implementation. The RNN-based estimator offers a distinct advantage over its discrete-time non-linear observer counterpart. For the RNN architecture, SoC estimation relies solely on matrix operations and the execution of basic activation functions. In contrast, the discrete-time non-linear observer requires solving numerous algebraic operations. Despite a reduction in complexity when compared with the continuous-time version of the non-linear observer presented in [15], which requires an ODE engine solver, the discrete-time version still has a high computational demand as all operations must be completed within a sampling time of 0.5 s. Notably, the RNN implementation with a sampling time of 5 min yields highly accurate estimation results.

The most important constraint for the design of an RNN estimator is the availability of mathematical models (or experimental data) and the different profiles required for training. Additionally, the operating conditions of the TES unit may influence the scope of the use of the estimator. In other words, a larger number of profiles for training is required if the operating conditions span through a large range of temperatures and mass flow rates of the HTF. However, the ice tank investigated in this paper has a unique characteristic: the charging process was conducted at temperatures near −6°C, thereby limiting the operating mode to a (nearly) constant value. For discharging, the operating conditions were restricted to a narrow range of mass flow rate (within 5 kg/s to 6 kg/s). These specific characteristics enable a reasonably limited number of combinations for the dataset adopted in the paper. As discussed in Section 4.1, this comprised a total of 2132 SoC profiles, with 1681 used for charging but only 451 for discharging, with 132 random profiles of the dataset used for training purposes. Nevertheless, conducting additional simulations or experimental runs considering a broader operating range could enhance the reliability of the RNN estimators.

As highlighted in Section 3.1, RNN architectures are susceptible to a vanishing gradient, which inhibits their ability to capture long-term dependencies. To alleviate issues with gradient dispersion, LSTM layers were incorporated into the RNN structure, where the provision of a state memory enables maintaining a constant error signal flow during the generation of network weights throughout the training process [38]. RNN structures may also exhibit stability and robustness issues when dealing with long time-series data. To prevent this, downsampling was conducted. This enabled reducing the number of elements per profile from 72,000 elements for a time-step of 0.5 s for a 10-h simulation to 120 elements only using a sampling time of 5 min (i.e. 300 s) instead. This number of elements may vary while still yielding favourable outcomes, as demonstrated in Section 5.4, where training was conducted adopting 60 and 300 elements per profile.

The comparison of three different sampling times—2 min, 5 min, and 10 min (i.e. 120 s, 300 s, and 600 s)—in Section 5.4 revealed an improvement in accuracy as time was shortened and a reduced accuracy as it was increased. Regarding training duration, a longer training period was observed with a 120-s sampling time (300 elements) compared to the durations required for 300 s and 600 s. Nevertheless, all training times are reasonably short considering the number of elements and profiles used. Moreover, training time is not a parameter related to the calculation cost of SoC. In fact, the SoC computation time remains consistent regardless of the choice of sampling time. It is concluded that an optimal sampling time may be reached through a direct trade-off between implementation requirements from users and desired performance accuracy.

Based on the previous discussion, an LSTM-based RNN architecture may be suitable for other engineering applications that require managing long process data. Although in this paper long charging and discharging processes have been employed for training, the results presented in Section 5 do not exhibit instability or convergence issues in SoC estimation.

By considering an increase in sampling time, the RNN estimator is able to produce similar results to those obtained by the discrete-time non-linear observer faster and with a significantly lower number of estimations. This has a meaningful implication for practical engineering applications: namely, the selection of the data acquisition system, which for the RNN estimator would require, for example, a lower storage memory, reduced power consumption, and reduced susceptibility to noise.

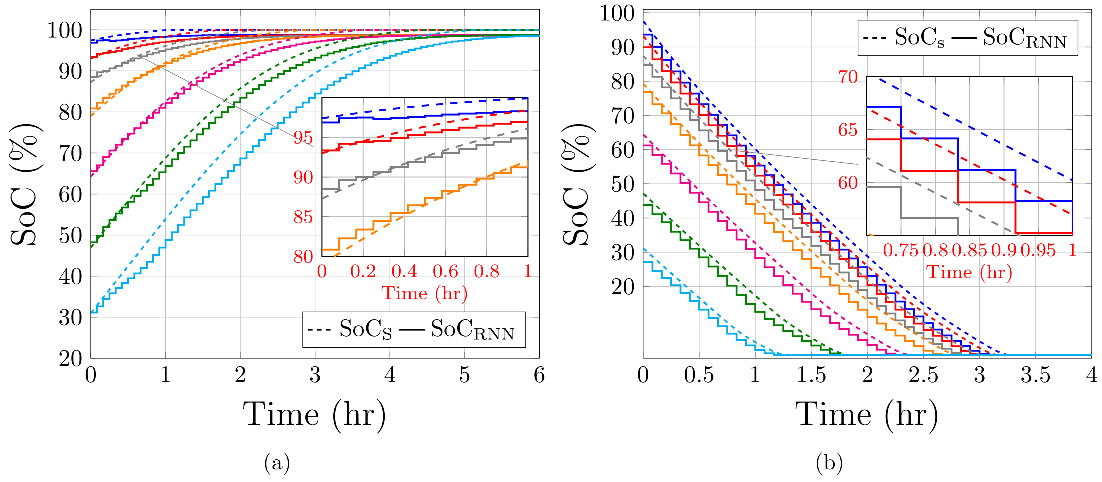


Fig. 30. RNN-LSTM performance under different initial conditions of SoC: (a) charging; (b) discharging.

7. On the limitations of the presented SoC estimator and its adoption for other systems

The work presented in this paper has been centred around thermal stores for cooling applications. Due to the promising results afforded by the presented RNN architecture for SoC estimation, extending its use to other energy storage systems may be of interest.

The proposed RNN estimator requires SoC data from the LHTES unit under varying operating conditions for training purposes. These conditions, arising from changes in HTF parameters, determine the behaviour of the thermal store during charging and discharging processes. While the method can be easily adapted for LHTES units for heating applications by establishing the ranges of the HTF conditions (namely mass flow rate and input temperature), its application for electric batteries would require to identify the distinct operating conditions and the variables that affect the battery performance. For instance, the state-of-health of an electric battery also influences its performance, adding further complexity for SoC estimation. As a result, adapting the proposed method to electric batteries, although promising and of arguably a high research value, would require further analysis which falls out of the scope of this paper.

On the same note, a discrete-time non-linear observer could be also adopted for SoC estimation in thermal stores for heating applications. As it has been explained in Section 5.1, the continuous-time non-linear observer for the ice tank is essentially a dynamic model described by ODEs. Its design is based on the mathematical model of the ice tank. The observer equations consider the thermophysical properties of elements in the tank involved in heat transfer and parameters such as volumetric capacity and heat transfer area. A dynamic model and its corresponding continuous-time non-linear observer were developed for an LHTES unit for heating applications in [15] following a similar methodology as the one adopted for the ice tank. In that reference, a comparison with experimental data showed an accurate performance of both the dynamic model of the TES unit and its observer, demonstrating the suitability of the modelling approach and the observer design for different types of TES technologies and for different applications. The interested reader is referred to [15] for a detailed description on the modelling implications for a heating system.

It is worth highlighting that a continuous-time observer for either an ice tank or an LHTES unit for heating applications would exhibit a similar mathematical structure. However, differences may arise when considering the internal structure of the storage unit and the properties of the HTF and PCM (including the PCM's specific heat–temperature curve, which is a key modelling parameter). Despite these modelling differences, a similar discretisation method (from a continuous-time to a discrete-time domain) as the one used for the ice tank can be easily adopted for non-linear observers for heating storage units. Designing a discrete-time non-linear observer for a heating application, however, falls out of the scope of this paper.

While the RNN-based estimator presented in this paper offers an accurate SoC quantification for an ice tank, it is necessary to recognise its limitations. Its performance depends on the continuous-time monitoring of its inputs—namely, the input temperature and mass flow rate of the HTF—and precise initial SoC conditions. In contrast, a well-designed observer or a Kalman filter has the capacity to reduce the estimation error to zero even when SoC initial conditions do not match the ice tank's actual conditions. Thus, the need for consistent input measurements limits the RNN's implementation. Arguably, a simpler SoC estimation method relying on the cumulatively calculation of SoC using the temperature difference (input–output) of the HTF and its mass flow rate would be sufficient. However, such an approach would require three measurements to be effective. The proposed RNN-based estimator however needs only two sensors. This attribute may be significant when assessing the operation of multiple ice tanks within a single system—as it is common in large district cooling systems, where a large number of variables may be monitored.

The results presented in this paper so far only consider initial conditions for a fully charged or a fully discharged ice tank. The rationale behind this selection is that the use of initial conditions with a SoC ranging from 0% to 100% would considerably increase the number of profiles to be used for training, thereby increasing the duration of the process. However, the RNN-LSTM architecture presented in the paper could be re-trained to consider random initial conditions for SoC and still exhibit an acceptable SoC estimation performance. To support this idea, 210 additional SoC profiles were considered alongside those previously discussed in Section 4.4 to train an RNN with two LSTM layers with 15 neurons in each layer. The training process for this RNN, which considered a total of 342 profiles, had a duration of 4 h.

Fig. 30 shows the performance of the re-trained RNN. Charging and discharging processes were simulated for HTF conditions of $T_{f,\mathrm{in}} = -6^{\circ}\mathrm{C}$ and $\dot{m}_f = 18$ kg/s for charging and with $T_{f,\mathrm{in}} = 10^{\circ}\mathrm{C}$ and $\dot{m}_f = 5$ kg/s for discharging. In the figure, subscript 's' stands for the SoC calculated directly from the mathematical model (dashed traces) and 'RNN' for the SoC obtained with the RNN estimator (solid traces). It can be observed that the estimation accuracy afforded by the estimator declined slightly.

Fig. 31 shows the simulation results for three consecutive charging–discharging cycles. The reduced accuracy observed in Fig. 30 is also appreciated in these results, as an increased estimation error is exhibited (notably during charging).

Even when the estimation accuracy of the re-trained RNN was reduced when compared to the results presented in Section 5, the

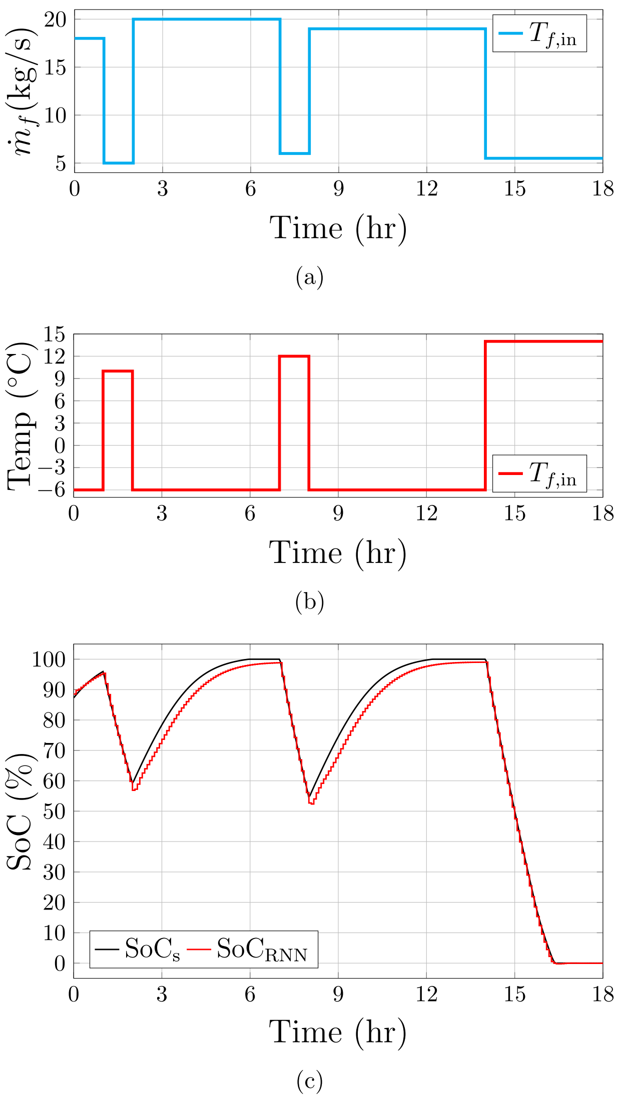


Fig. 31. Simulation conditions to assess the performance of a RNN estimator trained with SoC profiles that include variations in the initial conditions: (a) Mass flow rate of HTF; (b) input temperature of the HTF; and (c) comparison of the SoC calculated with the mathematical model of the ice tank (SoC_S) and the re-trained RNN estimator (SoC_RNN).

example presented in this section demonstrates that the proposed SoC estimation methodology enables adopting new arbitrary initial conditions not limited to SoCs of 0% and 100%. The estimation performance could be further improved by expanding the dataset for training. During this process, some profiles could be excluded from the dataset to restrict it to a manageable size while improving training efficiency and achieving an optimal estimation performance following deployment. However, this exercise falls out of the scope of this paper.

Although the results presented in the paper demonstrate a good SoC estimation performance by an RNN estimator incorporating LSTM layers into its architecture, increasing the overall data deviation would provide further confidence on the suitability of the presented methodology. Owing to the constraints imposed by the discrete-time non-linear observer's convergence issues and efficiency of RNN training, such a comprehensive analysis falls out of the scope of this work.

It is important to clarify that this paper adopted well-established neural network architectures to address a well-defined practical engineering challenge rather than to advance AI theory or to enhance existing LSTM structures and RNNs. To the best of the authors' knowledge, these neural network architectures have not been previously used to estimate the SoC of thermal stores for cooling applications.

8. Conclusions

Effective management of energy systems incorporating thermal stores requires the accurate knowledge of the state of the TES units to achieve effective control strategies and operational optimisation. However, this may require significant investment in instrumentation which, in turn, can lead to high implementation costs. To address these challenges, an RNN-based SoC estimator for LHTES units was presented in this paper. The RNN-based estimator eliminates the need for internal temperature sensors and only requires measurements of mass flow rate and input temperature of the HTF for its operation. It also enables a simpler implementation compared to other alternatives by adopting LSTM layers within its architecture.

A key advantage of the RNN-based SoC estimator lies in its computational efficiency. It operates with a reduced set of matrix operations and activation functions, resulting in a decreased computation time (24 times faster) when compared to a discrete-time non-linear observer-based SoC estimator. This is possible as the sampling time afforded by the RNN structure may be substantially increased without significantly compromising estimation precision. The training methodology can be suitably tailored for charging–discharging cycles with varied time durations by resetting the hidden and cell states of LSTM layers. This adjustment in the internal parameters of the RNN architecture considerably streamlines training, enabling an independent evaluation of charging and discharging and ensuring a precise SoC estimation across these processes.

When compared with direct calculations using the mathematical model of the ice tank, RNN estimators registered RMSE and MAE values below 0.73% and 0.41% over a long range of charging–discharging cycles under varying operating conditions. In contrast, a discrete-time non-linear observer produced RMSE and MAE values of 0.08% and 0.015%. While the observer exhibited marginally enhanced accuracy metrics, its drawbacks concerning sampling and computation times render the RNN-based SoC estimator a better alternative towards practical implementation.

CRediT authorship contribution statement

Hector Bastida: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Ivan De la Cruz-Loredo:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization. **Pranaynil Saikia:** Methodology, Investigation, Formal analysis, Conceptualization. **Carlos E. Ugalde-Loo:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Relevant datasets produced for this paper have been made available as supplementary material. These are available in the Cardiff University data catalogue at http://doi.org/10.17035/d.2024.0322710137.

Acknowledgements

The work presented in this paper was supported by the Engineering and Physical Sciences Research Council (EPSRC), United Kingdom, UK Research and Innovation (UKRI), through the project ‘Flexibility from Cooling and Storage (Flex-Cool-Store)’, grant EP/V042505/1.

Appendix A. Discretisation of continuous-time non-linear observer

The continuous-time non-linear observer presented in [15] was discretised (to a discrete-time domain in a control engineering sense) for comparison with the RNN-based SoC estimator presented in this paper. Discretisation of the set of ODE describing the non-linear observer is based on signal sampling concepts, resulting in a simple and accurate discrete-time observer.

The basic concept of system discretisation is to convert a continuous-time signal to discrete-time values determined by a periodic sampling. Thus, the signal can be rebuilt continuously from a discrete-time sequence using data hold circuits. The most common and simplest hold circuit is the ZOH, mathematically expressed as:

$$h\left(jT_s + t\right) = x\left(jT_s\right), \qquad \text{for} \qquad 0 \le T_s \tag{A.1}$$

where j is the number of the sample and T_s is the sampling time. Thus, the values from a sampling instant are retained and kept constant until the next sampling instant.

A sampler and a ZOH are required to generate the discretised signal. This is illustrated in Fig. A.32, where a signal is sampled at specific times. Then the signal values are retained by the ZOH until the next sample occurs.

The continuous-time non-linear observer presented in [15] is converted to a discrete-time form such that obtaining a solution to this dynamic system does not involve an ODE engine solver. In turn, this reduces the computational resources required. The mathematical procedure for this is illustrated in Fig. A.33 with a block diagram. The two-point backward difference formula [53] was used to represent the ODEs as a set of derivatives considering a specific time-step. This is also defined as the sampling time of the implementation. This method establishes that the derivative of a function $f'(t)$ in a specific point is given by the difference between the function evaluated at an immediately subsequent point and the function evaluated at the current point (i.e $f\left(x+h_x\right) - f(x)$) divided by the difference h_x between the points. This is expressed mathematically as

$$f'(x) = \frac{f\left(x+h_x\right) - f(x)}{h_x}. \tag{A.2}$$

Given that a state–space representation is a set of derivative functions as in (A.2), the described method is employed to solve them using

$$f\left(x+h_x\right) = h_x f'(x) + f(x), \tag{A.3}$$

where h_x is the increment Δx to define the next point to solve the function.

The continuous-time non-linear observer is described by Eq. (A.4) given in Box I.

By replacing $f'(x)$ with the function of the estimated state variable $\dot{\hat{x}}$ as $f'(x) = \dot{\hat{x}}$, and defining the evaluation of the function in $(x+h_x)$ as the solution of the state variable for two instances of times (t_0 and t_1) as $f(x) = x\left(t_0\right)$ and $f\left(x+h_x\right) = x\left(t_1\right)$, the solution of the state variables, represented as vector $\hat{\mathbf{x}}$, for time t_1, is defined by

$$\begin{aligned}\hat{\mathbf{x}}\left(t_1\right) = \Delta t\dot{\hat{\mathbf{x}}}\left(t_0\right) + \hat{\mathbf{x}}\left(t_0\right) = \Delta t\left[f\left(\hat{\mathbf{x}}\left(t_0\right), \mathbf{u}\left(t_0\right)\right) + \mathbf{J}\left(\mathbf{y}\left(t_0\right) - \mathbf{C}\left(\hat{\mathbf{x}}\left(t_0\right)\right)\right)\right] \\ + \hat{\mathbf{x}}\left(t_0\right),\end{aligned} \tag{A.5}$$

where $\Delta t = t_1 - t_0$. By adopting a discrete-time notation in (A.5), the general expression for the solution of the state variables of the non-linear observer is expressed as

$$\begin{aligned}\hat{\mathbf{x}}\left((j+1)T_s\right) = T_s\left[f\left(\hat{\mathbf{x}}\left(jT_s\right), \mathbf{u}\left(jT_s\right)\right) + \mathbf{J}\left(\mathbf{y}\left(jT_s\right) - \mathbf{C}\hat{\mathbf{x}}\left(jT_s\right)\right)\right] \\ + \hat{\mathbf{x}}\left(jT_s\right),\end{aligned} \tag{A.6}$$

where T_s is the time-step defined for the discrete-time solution of the equation, $\mathbf{J}$ is the observer gain matrix, $\mathbf{u}$ is the input vector of the state–space representation of the ice tank, and $\mathbf{C}$ is the output matrix.

To illustrate the use of the discretisation method described previously to solve the ODEs of the non-linear observer, the first two equations of the set in (A.4) are used. These equations describe the temperature of the HTF and PCM for the first node, with the state variables defined as $T_{f,1,a} = x_1$ and $T_{w,1,a} = x_2$. Assuming a (thermal)

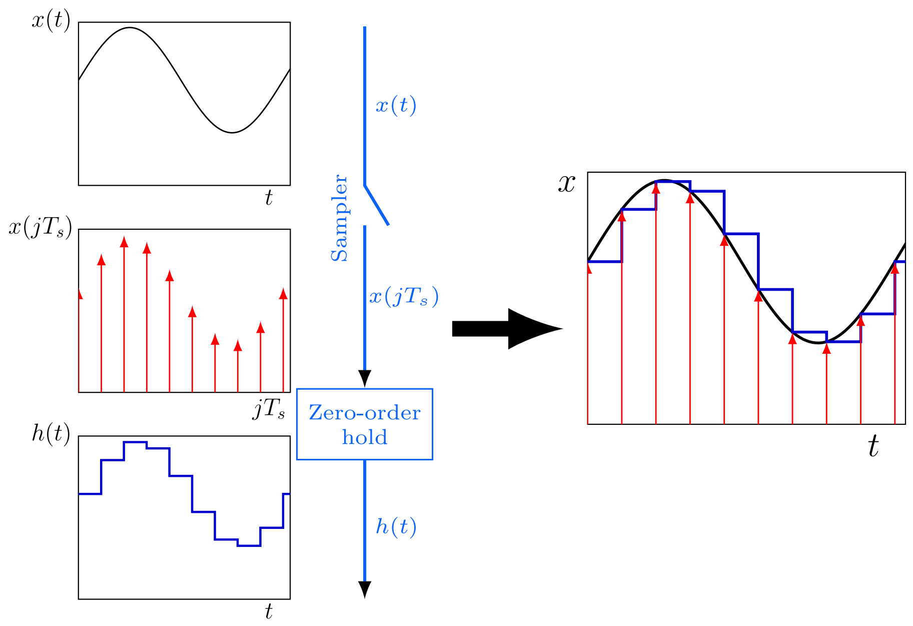


Fig. A.32. Illustrative sequence of the sampling process of a signal using a sampler and a ZOH.

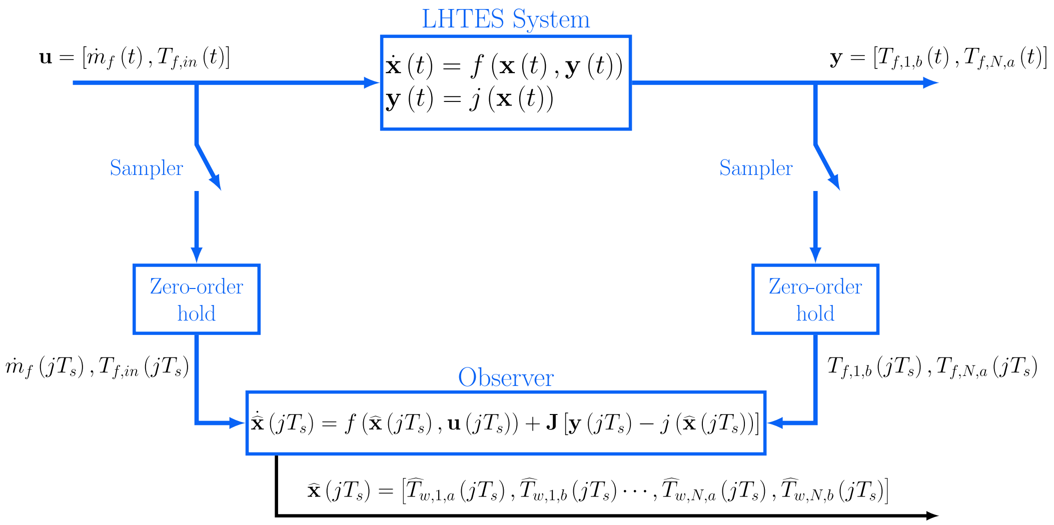

LHTES System
$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{y}(t))$
$\mathbf{y}(t) = j(\mathbf{x}(t))$
$\mathbf{u} = [\dot{m}_f(t), T_{f,in}(t)]$
$\mathbf{y} = [T_{f,1,b}(t), T_{f,N,a}(t)]$
Sampler
Sampler
Zero-order hold
Zero-order hold
$\dot{m}_f(jT_s), T_{f,in}(jT_s)$
$T_{f,1,b}(jT_s), T_{f,N,a}(jT_s)$
Observer
$\dot{\hat{\mathbf{x}}}(jT_s) = f(\hat{\mathbf{x}}(jT_s), \mathbf{u}(jT_s)) + \mathbf{J}[\mathbf{y}(jT_s) - j(\hat{\mathbf{x}}(jT_s))]$
$\hat{\mathbf{x}}(jT_s) = [\widehat{T}_{w,1,a}(jT_s), \widehat{T}_{w,1,b}(jT_s) \cdots, \widehat{T}_{w,N,a}(jT_s), \widehat{T}_{w,N,b}(jT_s)]$

Fig. A.33. Implementation of discretised non-linear state observer for the ice tank.

$$\begin{bmatrix} \dot{\hat{T}}_{f,1,a} \\ \dot{\hat{T}}_{w,1,a} \\ \dot{\hat{T}}_{f,1,b} \\ \dot{\hat{T}}_{w,1,b} \\ \vdots \\ \dot{\hat{T}}_{f,i,a} \\ \dot{\hat{T}}_{w,i,a} \\ \dot{\hat{T}}_{f,i,b} \\ \dot{\hat{T}}_{w,i,b} \\ \vdots \\ \dot{\hat{T}}_{f,N,a} \\ \dot{\hat{T}}_{w,N,a} \\ \dot{\hat{T}}_{f,N,b} \\ \dot{\hat{T}}_{w,N,b} \end{bmatrix} = \begin{bmatrix} \dot{\hat{x}}_1 \\ \dot{\hat{x}}_2 \\ \dot{\hat{x}}_3 \\ \dot{\hat{x}}_4 \\ \vdots \\ \dot{\hat{x}}_{l+1} \\ \dot{\hat{x}}_{l+2} \\ \dot{\hat{x}}_{l+3} \\ \dot{\hat{x}}_{l+4} \\ \vdots \\ \dot{\hat{x}}_{4N-3} \\ \dot{\hat{x}}_{4N-2} \\ \dot{\hat{x}}_{4N-1} \\ \dot{\hat{x}}_{4N} \end{bmatrix} = \begin{bmatrix} \left[\frac{\dot{m}_f c_{p,f,1}(T_{f,\text{in}}-\hat{x}_1)+U(A_{tr}/N)(\hat{x}_2-\hat{x}_1)}{\rho_{f,1}c_{p,f,1}(V_f/N)}\right] + [J_1 e_a + J_1 e_b] \\ \left[\frac{U(A_{tr}/N)(\hat{x}_1-\hat{x}_2)+U_w(A_{ex}/N)(\hat{x}_4-\hat{x}_2)+U_{ls}(A_{ex}/N)(0-\hat{x}_2)}{\rho_{w,2}c_{p,w,2}(V_w/N)}\right] + [J_2 e_a + J_2 e_b] \\ \left[\frac{\dot{m}_f c_{p,f,3}(\hat{x}_7-\hat{x}_3)+U(A_{tr}/N)(\hat{x}_4-\hat{x}_3)}{\rho_{f,3}c_{p,f,3}(V_f/N)}\right] + [J_3 e_a + J_3 e_b] \\ \left[\frac{U(A_{tr}/N)(\hat{x}_3-\hat{x}_4)+U_w(A_{ex}/N)(\hat{x}_2-\hat{x}_4)+U_{ls}(A_{ex}/N)(0-\hat{x}_4)}{\rho_{w,4}c_{p,w,4}(V_w/N)}\right] + [J_4 e_a + J_4 e_b] \\ \vdots \\ \left[\frac{\dot{m}_f c_{p,f,l}(\hat{x}_{l-4}-\hat{x}_l)+U(A_{tr}/N)(\hat{x}_{l+1}-\hat{x}_l)}{\rho_{f,l}c_{p,f,l}(V_f/N)}\right] + [J_l e_a + J_l e_b] \\ \left[\frac{U(A_{tr}/N)(\hat{x}_l-\hat{x}_{l+1})+U_w(A_{ex}/N)(\hat{x}_{l+3}-\hat{x}_{l+1})+U_{ls}(A_{ex}/N)(0-\hat{x}_{l+1})}{\rho_{w,l+1}c_{p,w,l+1}(V_w/N)}\right] + [J_{l+1} e_a + J_{l+1} e_b] \\ \left[\frac{\dot{m}_f c_{p,f,l+2}(\hat{x}_{l+6}-\hat{x}_{l+2})+U(A_{tr}/N)(\hat{x}_{l+3}-\hat{x}_{l+2})}{\rho_{f,l+2}c_{p,f,l+2}(V_f/N)}\right] + [J_{l+2} e_a + J_{l+2} e_b] \\ \left[\frac{U(A_{tr}/N)(\hat{x}_{l+2}-\hat{x}_{l+3})+U_w(A_{ex}/N)(\hat{x}_{l+1}-\hat{x}_{l+3})+U_{ls}(A_{ex}/N)(0-\hat{x}_{l+3})}{\rho_{w,l+3}c_{p,w,l+3}(V_w/N)}\right] + [J_{l+3} e_a + J_{l+3} e_b] \\ \vdots \\ \left[\frac{\dot{m}_f c_{p,f,4N-3}(\hat{x}_{4N-7}-\hat{x}_{4N-3})+U(A_{tr}/N)(\hat{x}_{4N-2}-\hat{x}_{4N-3})}{\rho_{f,4N-3}c_{p,f,4N-3}(V_f/N)}\right] + [J_{4N-3} e_a + J_{4N-3} e_b] \\ \left[\frac{U(A_{tr}/N)(\hat{x}_{4N-3}-\hat{x}_{4N-2})+U_w(A_{ex}/N)(\hat{x}_{4N}-\hat{x}_{4N-2})+U_{ls}(A_{ex}/N)(0-\hat{x}_{4N-2})}{\rho_{w,4N-2}c_{p,w,4N-2}(V_w/N)}\right] + [J_{4N-2} e_a + J_{4N-2} e_b] \\ \left[\frac{\dot{m}_f c_{p,f,4N-1}(T_{f,\text{in}}-\hat{x}_{4N-1})+U(A_{tr}/N)(\hat{x}_{4N}-\hat{x}_{4N-1})}{\rho_{f,4N-1}c_{p,f,4N-1}(V_f/N)}\right] + [J_{4N-1} e_a + J_{4N-1} e_b] \\ \left[\frac{U(A_{tr}/N)(\hat{x}_{4N-1}-\hat{x}_{4N})+U_w(A_{ex}/N)(\hat{x}_{4N-2}-\hat{x}_{4N})+U_{ls}(A_{ex}/N)(0-\hat{x}_{4N})}{\rho_{w,4N}c_{p,w,4N}(V_w/N)}\right] + [J_{4N} e_a + J_{4N} e_b] \end{bmatrix} \tag{A.4}$$

Box I.

$$\hat{x}_2[j+1] = \hat{x}_2[j] + \left\{ T_s \left[\frac{U\left(A_{tr}/N\right)\left(\hat{x}_2[j]-\hat{x}_1[j]\right) + U_w\left(A_{ex}/N\right)\left(\hat{x}_4[j]-\hat{x}_2[j]\right) + +U_{ls}\left(A_{ex}/N\right)\left(0-\hat{x}_2[j]\right)}{\rho_{f,1}c_{p,f,1}\left(V_f/N\right)} \right] + \left[J_1 e_a[j] + J_1 e_b[j] \right] \right\}, \tag{A.10}$$

Box II.

discretisation model of the ice tank of 20 nodes, the ODEs for these state variables are defined as

$$\dot{\hat{x}}_1 = \left[\frac{\dot{m}_f c_{p,f,1}\left(T_{f,\text{in}}-\hat{x}_1\right) + U\left(A_{tr}/N\right)\left(\hat{x}_2-\hat{x}_1\right)}{\rho_{f,1}c_{p,f,1}\left(V_f/N\right)} \right] + \left[J_1 e_a + J_1 e_b\right] \tag{A.7}$$

and

$$\dot{\hat{x}}_2 = \left[\frac{U\left(A_{tr}/N\right)\left(\hat{x}_2-\hat{x}_1\right) + U_w\left(A_{ex}/N\right)\left(\hat{x}_4-\hat{x}_2\right) + U_{ls}\left(A_{ex}/N\right)\left(0-\hat{x}_2\right)}{\rho_{w,2}c_{p,w,2}\left(V_f/N\right)} \right] + \left[J_2 e_a + J_2 e_b\right], \tag{A.8}$$

where the errors are $e_a = T_{f,\text{o}} - T_{f,N,a}$ and $e_b = T_{f,\text{o}} - T_{f,1,b}$, c_p [J/kg °C] is the specific heat, U [W/(m^2 °C)] is the overall heat transfer coefficient between the HTF and PCM, A_{tr} [m^2] is the heat transfer area between the HTF and PCM, N is the number of nodes, ρ [kg/m^3] refers to density, V [m^3] refers to volume, U_w [W/(m^2 °C)] is the conduction heat transfer coefficient between the control volumes of water within the tubes, U_{ls} [W/(m^2 °C)] is the conduction heat transfer coefficient given by the heat lost due to the portion of water whose phase change is incomplete, and A_{ex} [m^2] is the external surface area of the control volume.

Following discretisation, (A.7) and (A.8) are expressed as

$$\hat{x}_1[j+1] = \hat{x}_1[j] + \left\{ T_s \left[\frac{\dot{m}_f[j] c_{p,f,1}\left(T_{f,\text{in}}[j]-\hat{x}_1[j]\right) + U\left(A_{tr}/N\right)\left(\hat{x}_2[j]-\hat{x}_1[j]\right)}{\rho_{f,1}c_{p,f,1}\left(V_f/N\right)} \right] + \left[J_1 e_a[j] + J_1 e_b[j] \right] \right\}, \tag{A.9}$$

and Eq. (A.10) given in Box II, where the discrete-time form of the errors is given by $e_a[j] = T_{f,\text{o}}[j] - T_{f,N,a}[j]$ and $e_b[j] = T_{f,\text{o}}[j] - T_{f,1,b}[j]$.

As for its continuous-time counterpart, the calculation of the overall heat transfer coefficient is also needed for the discrete-time non-linear observer. This requirement entails additional computational time as the thermophysical properties of the HTF and PCM must be incorporated and these are dependent on temperature. Thus, the mathematical model of the discrete-time non-linear observer is described by a set of algebraic functions that requires the previous state values to calculate the solution of all states at the current sampling time. Then, the model of the discrete-time non-linear observer is defined by Eq. (A.11) given in Box III, where $a_1 = \dot{m}_f c_{p,f,l}$, $a_2 = U\left(A_{tr}/N\right)$, $a_3 = U_w\left(A_{ex}/N\right)$, and $a_4 = U_{ls}\left(A_{ex}/N\right)$.

For an easy implementation in software, the pseudo-code corresponding to the discrete-time observer is shown as Eq. (A.12) in Box IV.

The discretisation method for the continuous-time non-linear observer is summarised by the flowchart in Fig. A.34.

It is important to note that while the continuous-time non-linear observer presented in [29] was developed based on the non-linear model of the LHTES unit, the discrete-time version presented in this section was derived by applying a discretisation method. Such an approach follows standard methodologies available in control theory which enable obtaining a discrete-time representation of a continuous-time system which is suitable for practical implementation. More specifically, by adopting a method based on the backward difference formula, the discrete-time observer is represented by a set of algebraic equations and requires storing previous estimation values for estimation in subsequent

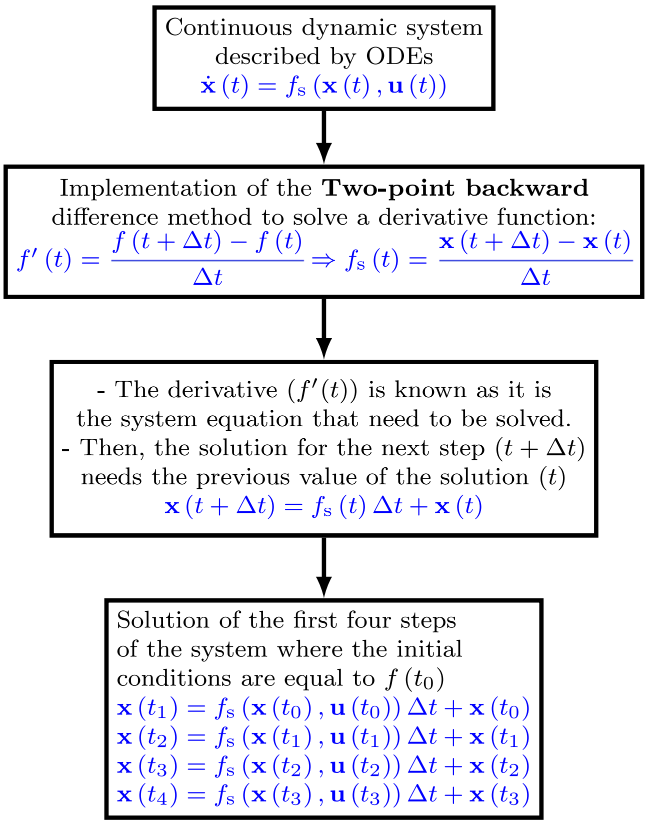

Continuous dynamic system described by ODEs
$\dot{\mathbf{x}}(t) = f_s(\mathbf{x}(t), \mathbf{u}(t))$
Implementation of the **Two-point backward** difference method to solve a derivative function:
$f'(t) = \frac{f(t+\Delta t) - f(t)}{\Delta t} \Rightarrow f_s(t) = \frac{\mathbf{x}(t+\Delta t) - \mathbf{x}(t)}{\Delta t}$
- The derivative $(f'(t))$ is known as it is the system equation that need to be solved.
- Then, the solution for the next step $(t+\Delta t)$ needs the previous value of the solution (t)
$\mathbf{x}(t+\Delta t) = f_s(t)\Delta t + \mathbf{x}(t)$
Solution of the first four steps of the system where the initial conditions are equal to $f(t_0)$
$\mathbf{x}(t_1) = f_s(\mathbf{x}(t_0), \mathbf{u}(t_0))\Delta t + \mathbf{x}(t_0)$
$\mathbf{x}(t_2) = f_s(\mathbf{x}(t_1), \mathbf{u}(t_1))\Delta t + \mathbf{x}(t_1)$
$\mathbf{x}(t_3) = f_s(\mathbf{x}(t_2), \mathbf{u}(t_2))\Delta t + \mathbf{x}(t_2)$
$\mathbf{x}(t_4) = f_s(\mathbf{x}(t_3), \mathbf{u}(t_3))\Delta t + \mathbf{x}(t_3)$

Fig. A.34. Flowchart describing the procedure to discretise a continuous-time dynamic system.

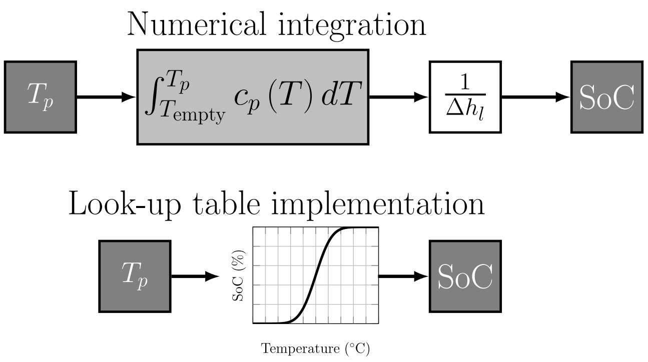


Fig. A.35. Block diagrams of the implementation of the SoC calculation method using numerical integration and a look-up table [15].

$$\begin{bmatrix} \hat{x}_1[j+1] \\ \hat{x}_2[j+1] \\ \hat{x}_3[j+1] \\ \hat{x}_4[j+1] \\ \vdots \\ \hat{x}_{l+1}[j+1] \\ \hat{x}_{l+2}[j+1] \\ \hat{x}_{l+3}[j+1] \\ \hat{x}_{l+4}[j+1] \\ \vdots \\ \hat{x}_{4N-3}[j+1] \\ \hat{x}_{4N-2}[j+1] \\ \hat{x}_{4N-1}[j+1] \\ \hat{x}_{4N}[j+1] \end{bmatrix} = \begin{bmatrix} \hat{x}_1[j] + T_s\left\{ \frac{a_1(T_{f,\mathrm{in}}[j]-\hat{x}_1[j])+a_2(\hat{x}_2[j]-\hat{x}_1[j])}{\rho_{f,1}c_{p,f,1}(V_f/N)} + \left[J_1 e_a[j] + J_1 e_b[j]\right] \right\} \\ \hat{x}_2[j] + T_s\left\{ \frac{a_2(\hat{x}_1[j]-\hat{x}_2[j])+a_3(\hat{x}_4[j]-\hat{x}_2[j])+a_4(0-\hat{x}_2[j])}{\rho_{w,2}c_{p,w,2}(V_w/N)} + \left[J_2 e_a[j] + J_2 e_b[j]\right] \right\} \\ \hat{x}_3[j] + T_s\left\{ \frac{a_1(\hat{x}_7[j]-\hat{x}_3[j])+a_2(\hat{x}_4[j]-\hat{x}_3[j])}{\rho_{f,3}c_{p,f,3}(V_f/N)} + \left[J_3 e_a[j] + J_3 e_b[j]\right] \right\} \\ \hat{x}_4[j] + T_s\left\{ \frac{a_2(\hat{x}_3[j]-\hat{x}_4[j])+a_3(\hat{x}_2[j]-\hat{x}_4[j])+a_4(0-\hat{x}_4[j])}{\rho_{w,4}c_{p,w,4}(V_w/N)} + \left[J_4 e_a[j] + J_4 e_b[j]\right] \right\} \\ \vdots \\ \hat{x}_l[j] + T_s\left\{ \frac{a_1(\hat{x}_{l-4}[j]-\hat{x}_l[j])+a_2(\hat{x}_{l+1}[j]-\hat{x}_l[j])}{\rho_{f,l}c_{p,f,l}(V_f/N)} + \left[J_l e_a[j] + J_l e_b[j]\right] \right\} \\ \hat{x}_{l+1}[j] + T_s\left\{ \frac{a_2(\hat{x}_l[j]-\hat{x}_{l+1}[j])+a_3(\hat{x}_{l+3}[j]-\hat{x}_{l+1}[j])+a_4(0-\hat{x}_{l+1}[j])}{\rho_{w,l+1}c_{p,w,l+1}(V_w/N)} + \left[J_{l+1} e_a[j] + J_{l+1} e_b[j]\right] \right\} \\ \hat{x}_{l+2}[j] + T_s\left\{ \frac{a_1(\hat{x}_{l+6}[j]-\hat{x}_{l+2}[j])+a_2(\hat{x}_{l+3}[j]-\hat{x}_{l+2}[j])}{\rho_{f,l+2}c_{p,f,l+2}(V_f/N)} + \left[J_{l+2} e_a[j] + J_{l+2} e_b[j]\right] \right\} \\ \hat{x}_{l+3}[j] + T_s\left\{ \frac{a_2(\hat{x}_{l+2}[j]-\hat{x}_{l+3}[j])+a_3(\hat{x}_{l+1}[j]-\hat{x}_{l+3}[j])+a_4(0-\hat{x}_{l+3}[j])}{\rho_{w,l+3}c_{p,w,l+3}(V_w/N)} + \left[J_{l+3} e_a[j] + J_{l+3} e_b[j]\right] \right\} \\ \vdots \\ \hat{x}_{4N-3}[j] + T_s\left\{ \frac{a_1(\hat{x}_{4N-7}[j]-\hat{x}_{4N-3}[j])+a_2(\hat{x}_{4N-2}[j]-\hat{x}_{4N-3}[j])}{\rho_{f,4N-3}c_{p,f,4N-3}(V_f/N)} + \left[J_{4N-3} e_a[j] + J_{4N-3} e_b[j]\right] \right\} \\ \hat{x}_{4N-2}[j] + T_s\left\{ \frac{a_2(\hat{x}_{4N-3}[j]-\hat{x}_{4N-2}[j])+a_3(\hat{x}_{4N}[j]-\hat{x}_{4N-2}[j])+a_4(0-\hat{x}_{4N-2}[j])}{\rho_{w,4N-2}c_{p,w,4N-2}(V_w/N)} + \left[J_{4N-2} e_a[j] + J_{4N-2} e_b[j]\right] \right\} \\ \hat{x}_{4N-1}[j] + T_s\left\{ \frac{a_1(T_{f,\mathrm{in}}[j]-\hat{x}_{4N-1}[j])+a_2(\hat{x}_{4N}[j]-\hat{x}_{4N-1}[j])}{\rho_{f,4N-1}c_{p,f,4N-1}(V_f/N)} + \left[J_{4N-1} e_a[j] + J_{4N-1} e_b[j]\right] \right\} \\ \hat{x}_{4N}[j] + T_s\left\{ \frac{a_2(\hat{x}_{4N-1}[j]-\hat{x}_{4N}[j])+a_3(\hat{x}_{4N-2}[j]-\hat{x}_{4N}[j])+a_4(0-\hat{x}_{4N}[j])}{\rho_{w,4N}c_{p,w,4N}(V_w/N)} + \left[J_{4N} e_a[j] + J_{4N} e_b[j]\right] \right\} \end{bmatrix} \tag{A.11}$$

Box III.

$$\begin{bmatrix} T_{f,i,a} \\ T_{w,i,a} \\ T_{f,i,b} \\ T_{w,i,b} \end{bmatrix} = \begin{bmatrix} \hat{x}_{j+1}(i) \\ \hat{x}_{j+1}(i+1) \\ \hat{x}_{j+1}(i+2) \\ \hat{x}_{j+1}(i+3) \end{bmatrix} = \left[\begin{matrix} \hat{x}_j(i) \\ \hat{x}_j(i+1) \\ \hat{x}_j(i+2) \\ \hat{x}_j(i+3) \\ +T_s\left\{ \frac{\dot{m}_f c_{p,i}\left[\hat{x}_j(i-4)-\hat{x}_j(i)\right]+U(c)(A_{tr}/N)\left[\hat{x}_j(i+1)-\hat{x}_j(i)\right]}{\rho_{f,i}c_{p,f,i}(V_f/N)} + \left[J_i e_{a,j} + J_i e_{b,j}\right] \right\} \\ +T_s\left\{ \frac{U(c)(A_{tr}/N)\left[\hat{x}_j(i)-\hat{x}_j(i+1)\right]+U_w(A_{ex}/N)\left[\hat{x}_j(i+3)-\hat{x}_j(i+1)\right]+U_{ls}(A_{ex}/N)\left[0-\hat{x}_j(i+1)\right]}{\rho_{f,i+1}c_{p,f,i+1}(V_w/N)} + \left[J_{i+1} e_{a,j} + J_{i+1} e_{b,j}\right] \right\} \\ +T_s\left\{ \frac{\dot{m}_f c_{p,i+2}\left[\hat{x}_j(i+6)-\hat{x}_j(i+2)\right]+U(d)(A_{tr}/N)\left[\hat{x}_j(i+3)-\hat{x}_j(i+2)\right]}{\rho_{f,i+2}c_{p,f,i+2}(V_f/N)} + \left[J_{i+2} e_{a,j} + J_{i+2} e_{b,j}\right] \right\} \\ +T_s\left\{ \frac{U(d)(A_{tr}/N)\left[\hat{x}_j(i+2)-\hat{x}_j(i+3)\right]+U_w(A_{ex}/N)\left[\hat{x}_j(i+1)-\hat{x}_j(i+3)\right]+U_{ls}(A_{ex}/N)\left[0-\hat{x}_j(i+3)\right]}{\rho_{f,i+3}c_{p,f,i+3}(V_w/N)} + \left[J_{i+3} e_{a,j} + J_{i+3} e_{b,j}\right] \right\} \end{matrix} \right] \tag{A.12}$$

Box IV.

time-steps but prevents the use of an ODE solver—which is an essential requirement for a continuous-time observer based on ODEs.

Appendix B. State-of-charge calculation method

The specific latent heat Δh_l represents the energy required per unit mass to produce a phase change in a PCM. Δh_l can be determined using the specific heat–temperature curve of the PCM and the temperature boundaries of the phase change transition zone (T_{empty} and T_{full}). As proposed in [15], integrating the curve from T_{full} to the current temperature of the PCM T_p divided by Δh_l quantifies the remaining latent heat stored by the PCM. In turn, this indicates a completely discharged ice tank when the melting PCM temperature (T_{empty}) has been exceeded. This is expressed mathematically as

$$\mathrm{SoC}_T(T) = \begin{cases} 0 & T_p > T_{\mathrm{empty}} \\ 100 - \left[\dfrac{\int_{T_{\mathrm{full}}}^{T_p} c_p(T)\,dT}{\Delta h_l} \times 100 \right] & T_{\mathrm{full}} \leq T_p \leq T_{\mathrm{empty}} \\ 100 & T_p < T_{\mathrm{full}}. \end{cases} \tag{B.1}$$

For the ice tank model, the specific latent heat ($\Delta h_l = 334$ kJ/kg K) is limited by $T_{\mathrm{full}} = -5.7$ °C and $T_{\mathrm{empty}} = 0$ °C. If (B.1) is applied to all the estimated PCM temperatures of the nodes N given by the non-linear

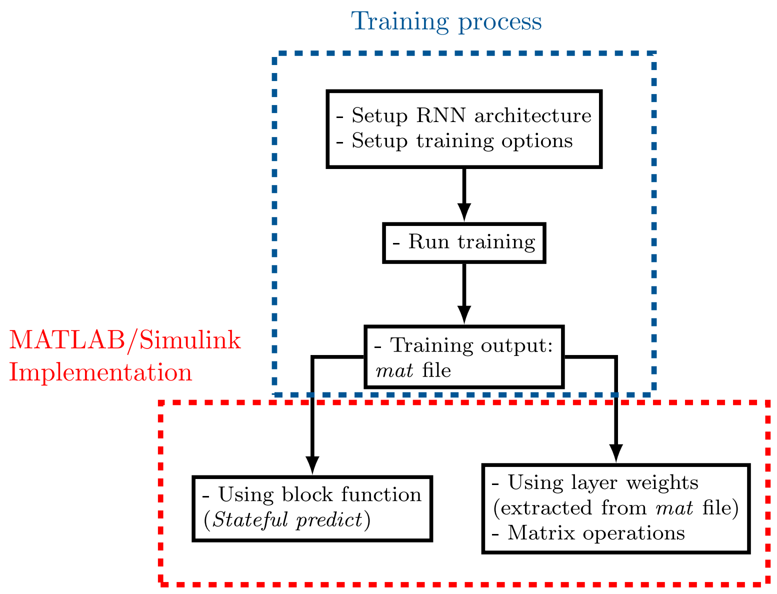

Training process
- Setup RNN architecture
- Setup training options
- Run training
- Training output: *mat* file
MATLAB/Simulink Implementation
- Using block function (*Stateful predict*)
- Using layer weights (extracted from *mat* file)
- Matrix operations

Fig. C.36. Schematic showing how training and implementation of the RNN are carried out.

observer, the SoC of the total volume of the PCM is determined by

$$\mathrm{SoC} = \frac{\sum_{i=1}^{N} \mathrm{SoC}_{T_i}}{N}, \tag{B.2}$$

where T_i is the temperature at node i.

A schematic for the implementation of SoC calculation is shown in Fig. A.35, where the integration of a well-defined specific heat–temperature curve is replaced by a look-up table to expedite the calculation process, thereby eliminating the need for numerical integration during the estimation time. The interested readers are directed to [15] for further details on the method.

Appendix C. MATLAB commands and flow chart for training and implementation of the RNN-based estimator

Commands available within MATLAB's machine learning toolbox enable creating the RNN architectures presented in Section 3.2 and undertaking its training process as described in Section 4.4. As a way of an example, the code to create the RNN with 2 LSTM layers and 15 neurons is provided next.

```
% Number of neurons
numNeurons = 15;
% Number of inputs and outputs
numInp = 3;
numOut = 1;
% Architecture of the RNN defined in Fig. 6(a)
layers = [
    % Normalization of the inputs from 0 to 1
    sequenceInputLayer(numInp,Normalization="rescale-zero-one")
    % Dense layer
    fullyConnectedLayer(numNeurons)
    % ReLU layer
    reluLayer
    % LSTM layer
    lstmLayer(numNeurons)
    % ReLU layer
    reluLayer
    % Dense layer
    fullyConnectedLayer(numOut)
    % Regression layer to computes the half-mean-squared-error
    % Loss during training process (this layer is only used during
    % training and it is not part of the RNN architecture)
regressionLayer];
```

The training options are comprehensively detailed in the third paragraph of Section 4.4 and shown below.

```
    options = trainingOptions("adam", ...
    MaxEpochs=Epochs2, ...
    GradientThreshold=1, ...
    InitialLearnRate=5e-3, ...
    LearnRateSchedule="piecewise", ...
    LearnRateDropPeriod=1e4, ...
    LearnRateDropFactor=0.6, ...
    Verbose=0, ...
    Plots="training-progress");
```

Finally the command to start the training process is

```
    trainNetwork(Xtrain, Ttrain, layers, options);
```

where Xtrain and Ttrain correspond to the inputs and targets profiles for training. The RNN is called 'layers' and the training options are defined with the parameter 'option'.

The implementation in MATLAB/Simulink can be executed either by employing the predefined block function ('Stateful predict') or by using the layer weights and biases. These values are extracted from the 'mat' file generated by training. Fig. C.36 illustrates the link between the training process and the implementation of the RNN.

References

[1] Pachori H, Choudhary T, Sheorey T. Significance of thermal energy storage material in solar air heaters. Mater Today: Proc 2022;56:126–34. http://dx.doi.org/10.1016/j.matpr.2021.12.516.
[2] Fan Y, Zhang C, Jiang L, Zhang X, Qiu L. Exploration on two-stage latent thermal energy storage for heat recovery in cryogenic air separation purification system. Energy 2022;239:122111. http://dx.doi.org/10.1016/j.energy.2021.122111.
[3] Cabeza LF, de Gracia A, Zsembinszki G, Borri E. Perspectives on thermal energy storage research. Energy 2021;231:120943. http://dx.doi.org/10.1016/j.energy.2021.120943.
[4] Hou J, Li H, Nord N, Huang G. Model predictive control for a university heat prosumer with data centre waste heat and thermal energy storage. Energy 2023;267:126579. http://dx.doi.org/10.1016/j.energy.2022.126579.
[5] Lizana J, Chacartegui R, Barrios-Padura A, Valverde JM. Advances in thermal energy storage materials and their applications towards zero energy buildings: A critical review. Appl Energy 2017;203:219–39. http://dx.doi.org/10.1016/j.apenergy.2017.06.008.
[6] Bruno F, Belusko M, Liu M, Tay NHS. Using solid-liquid phase change materials (PCMs) in thermal energy storage systems. Woodhead Publishing; 2015, p. 201–46. http://dx.doi.org/10.1533/9781782420965.2.201.

[7] Reddy KS, Mudgal V, Mallick TK. Review of latent heat thermal energy storage for improved material stability and effective load management. J Energy Storage 2018;15:205–27. http://dx.doi.org/10.1016/j.est.2017.11.005.

[8] Domanski R, El-Sebaii AA, Jaworski M. Cooking during off-sunshine hours using PCMs as storage media. Energy 1995;20(7):607–16. http://dx.doi.org/10.1016/0360-5442(95)00012-6.

[9] Liu G, Li Q, Wu J, Xie R, Zou Y, Marson A, Scipioni A, Manzardo A. Improving system performance of the refrigeration unit using phase change material (PCM) for transport refrigerated vehicles: An experimental investigation in South China. J Energy Storage 2022;51:104435. http://dx.doi.org/10.1016/j.est.2022.104435.

[10] Cabeza LF, Castellón C, Nogués M, Medrano M, Leppers R, Zubillaga O. Use of microencapsulated PCM in concrete walls for energy savings. Energy Build 2007;39(2):113–9. http://dx.doi.org/10.1016/j.enbuild.2006.03.030.

[11] Athienitis AK, Liu C, Hawes D, Banu D, Feldman D. Investigation of the thermal performance of a passive solar test-room with wall latent heat storage. Build Environ 1997;32(5):405–10. http://dx.doi.org/10.1016/S0360-1323(97)00009-7.

[12] Koschenz M, Lehmann B. Development of a thermally activated ceiling panel with PCM for application in lightweight and retrofitted buildings. Energy Build 2004;36(6):567–78. http://dx.doi.org/10.1016/j.enbuild.2004.01.029.

[13] Shanks M, Jain N. Control of a hybrid thermal management system: A heuristic strategy for charging and discharging a latent thermal energy storage device. In: 2022 21st IEEE intersociety conference on thermal and thermomechanical phenomena in electronic systems. iTherm, 2022, p. 1–10. http://dx.doi.org/10.1109/iTherm54085.2022.9899546.

[14] De la Cruz-Loredo I, Zinsmeister D, Licklederer T, Ugalde-Loo CE, Morales DA, Bastida H, Perić VS, Saleem A. Experimental validation of a hybrid 1-D multi-node model of a hot water thermal energy storage tank. Appl Energy 2023;332:120556. http://dx.doi.org/10.1016/j.apenergy.2022.120556.

[15] Bastida H, De la Cruz-Loredo I, Ugalde-Loo CE. Effective estimation of the state-of-charge of latent heat thermal energy storage for heating and cooling systems using non-linear state observers. Appl Energy 2023;331:120448. http://dx.doi.org/10.1016/j.apenergy.2022.120448.

[16] Barz T, Seliger D, Marx K, Sommer A, Walter SF, Bock HG, Körkel S. State and state of charge estimation for a latent heat storage. Control Eng Pract 2018;72:151–66. http://dx.doi.org/10.1016/j.conengprac.2017.11.006.

[17] Pernsteiner D, Schirrer A, Kasper L, Hofmann R, Jakubek S. State estimation concept for a nonlinear melting/solidification problem of a latent heat thermal energy storage. Comput Chem Eng 2021;153:107444. http://dx.doi.org/10.1016/j.compchemeng.2021.107444.

[18] Katayama H. Digital implementation of continuous-time observers for nonlinear networked control systems. SICE J Control Meas Syst Integr 2021;14(1):213–22. http://dx.doi.org/10.1080/18824889.2021.1956405.

[19] Venturini M. Simulation of compressor transient behavior through recurrent neural network models. J Turbomach 2005;128(3):444–54. http://dx.doi.org/10.1115/1.2183315.

[20] Salmela L, Tsipinakis N, Foi A, Billet C, Dudley JM, Genty G. Predicting ultrafast nonlinear dynamics in fibre optics with a recurrent neural network. Nat Mach Intell 2021;3:344–54. http://dx.doi.org/10.1038/s42256-021-00297-z.

[21] Sui H, Zhu H, Wu J, Luo B, Taccheo S, Zou X. Modeling pulse propagation in fiber optical parametric amplifier by a long short-term memory network. Optik 2022;260:169125. http://dx.doi.org/10.1016/j.ijleo.2022.169125.

[22] Xi Z, Wang R, Fu Y, Mi C. Accurate and reliable state of charge estimation of lithium ion batteries using time-delayed recurrent neural networks through the identification of overexcited neurons. Appl Energy 2022;305:117962. http://dx.doi.org/10.1016/j.apenergy.2021.117962.

[23] Feng X, Chen J, Zhang Z, Miao S, Zhu Q. State-of-charge estimation of lithium-ion battery based on clockwork recurrent neural network. Energy 2021;236:121360. http://dx.doi.org/10.1016/j.energy.2021.121360.

[24] Ren X, Liu S, Yu X, Dong X. A method for state-of-charge estimation of lithium-ion batteries based on PSO-LSTM. Energy 2021;234:121236. http://dx.doi.org/10.1016/j.energy.2021.121236.

[25] Ma L, Hu C, Cheng F. State of charge and state of energy estimation for lithium-ion batteries based on a long short-term memory neural network. J Energy Storage 2021;37:102440. http://dx.doi.org/10.1016/j.est.2021.102440.

[26] Ermis K, Erek A, Dincer I. Heat transfer analysis of phase change process in a finned-tube thermal energy storage system using artificial neural network. Int J Heat Mass Transfer 2007;50(15):3163–75. http://dx.doi.org/10.1016/j.ijheatmasstransfer.2006.12.017.

[27] Benzaama MH, Menhoudj S, Mokhtari AM, Lachi M. Comparative study of the thermal performance of an earth air heat exchanger and seasonal storage systems: Experimental validation of Artificial Neural Networks model. J Energy Storage 2022;53:105177. http://dx.doi.org/10.1016/j.est.2022.105177.

[28] Xiao T, Liu Z, Lu L, Han H, Huang X, Song X, Yang X, Meng X. LSTM-BP neural network analysis on solid-liquid phase change in a multi-channel thermal storage tank. Eng Anal Bound Elem 2023;146:226–40. http://dx.doi.org/10.1016/j.enganabound.2022.10.014.

[29] Bastida H, Ugalde-Loo CE, Abeysekera M, Jenkins N. Dynamic modelling of ice-based thermal energy storage for cooling applications. IET Energy Syst Integr 2022;4(3):317–34. http://dx.doi.org/10.1049/esi2.12061.

[30] CALMAC - Ice Bank energy storage tanks. CALMAC, N. J.. 2022, https://www.calmac.com/. [Accessed 03 October 2022].

[31] Drees KH, Braun JE. Modeling of area-constrained ice storage tanks. HVAC&R Res 1995;1(2):143–58. http://dx.doi.org/10.1080/10789669.1995.10391315.

[32] Lopez-Navarro A, Biosca-Taronger J, Torregrosa-Jaime B, Corberan JM, Bote-Garcia JL, Paya J. Experimental investigations on the influence of ice floating in an internal melt ice-on-coil tank. Energy Build 2013;57:20–5. http://dx.doi.org/10.1016/j.enbuild.2012.10.040.

[33] Lopez-Navarro A, Biosca-Taronger J, Torregrosa-Jaime B, Martinez-Galvan I, Corberan JM, Esteban-Matias JC, Paya J. Experimental investigation of the temperatures and performance of a commercial ice-storage tank. Int J Refrig 2013;36(4):1310–8. http://dx.doi.org/10.1016/j.ijrefrig.2012.09.008.

[34] CALMAC - Ice bank energy storage model C tank. CALMAC, NJ. 2024, https://www.calmac.com/icebank-energy-storage-model-c. [Accessed 02 February 2024].

[35] Barz T, Zauner C, Lager D, López Cárdenas DC, Hengstberger F, Cruz Bournazou MN, Marx K. Experimental analysis and numerical modeling of a shell and tube heat storage unit with phase change materials. Ind Eng Chem Res 2016;55(29):8154–64. http://dx.doi.org/10.1021/acs.iecr.6b01080.

[36] Khairudin NBM, Mustapha NB, Aris TNBM, Zolkepli MB. Comparison of machine learning models for rainfall forecasting. In: 2020 International conference on computer science and its application in agriculture. ICOSICA, 2020, p. 1–5. http://dx.doi.org/10.1109/ICOSICA49951.2020.9243275.

[37] Cordeiro-Costas M, Villanueva D, Eguía-Oller P, Martínez-Comesaña M, Ramos S. Load forecasting with machine learning and deep learning methods. Appl Sci 2023;13(13):7933. http://dx.doi.org/10.3390/app13137933.

[38] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[39] Jung M, da Costa Mendes PR, Önnheim M, Gustavsson E. Model predictive control when utilizing LSTM as dynamic models. Eng Appl Artif Intell 2023;123:106226. http://dx.doi.org/10.1016/j.engappai.2023.106226.

[40] Elman JL. Finding structure in time. Cogn Sci 1990;14(3):179–211. http://dx.doi.org/10.1207/s15516709cog1402-1.

[41] Gers FA, Schmidhuber J, Cummins F. Learning to forget: continual prediction with LSTM. In: 1999 Ninth international conference on artificial neural networks. ICANN 99, 1999, p. 850–5. http://dx.doi.org/10.1049/cp:19991218.

[42] Geron A. Hands-on machine learning with scikit-learn, keras, and tensorflow: Concepts, tools, and techniques to build intelligent systems. 2nd ed.. Sebastopol, United States: O'Reilly Media; 2022.

[43] Wang Y. A new concept using LSTM Neural Networks for dynamic system identification. In: 2017 American control conference. ACC, 2017, p. 5324–9. http://dx.doi.org/10.23919/ACC.2017.7963782.

[44] Detect vanishing gradients in deep neural networks by plotting gradient distributions - MATLAB & Simulink - MathWorks United Kingdom. 2023, https://uk.mathworks.com/help/deeplearning/ug/detect-vanishing-gradients-in-deep-neural-networks.html. [Accessed 19 June 2023].

[45] Mariani M, Tweneboah O, Beccar-Varela M. Data science in theory and practice: Techniques for big data analytics and complex data sets. Wiley; 2021, p. 370.

[46] Llerena Caña JP, García Herrero J, Molina López JM. An approach to forecasting and filtering noise in dynamic systems using LSTM architectures. Neurocomputing 2022;500:637–48. http://dx.doi.org/10.1016/j.neucom.2021.08.162.

[47] Kingma DP, Ba J. Adam: A method for stochastic optimization. In: 3rd International conference for learning representations. 2015, p. 1–15. http://dx.doi.org/10.48550/arXiv.1412.6980.

[48] Beale MH, Hagan MT, Demuth HB. Deep learning toolbox user's guide. The MathWorks, Inc; 2023, p. 1–4872.

[49] Bengio Y. Practical recommendations for gradient-based training of deep architectures. In: Neural networks: Tricks of the trade. Lecture notes in computer science, vol. 7700, Berlin, Germany: Springer; 2012, http://dx.doi.org/10.1007/978-3-642-35289-8-26.

[50] Goodfellow I, Bengio Y, Courville A, Bach F. Deep Learning. Cambridge, Mass: MIT Press; 2017.

[51] Chen J, Zhang Y, Wu J, Cheng W, Zhu Q. SOC estimation for lithium-ion battery using the LSTM-RNN with extended input and constrained output. Energy 2023;262:125375. http://dx.doi.org/10.1016/j.energy.2022.125375.

[52] Shafi J, Ghalambaz M, Fteiti M, Ismael M, Ghalambaz M. Computational modeling of latent heat thermal energy storage in a shell-tube unit: Using neural networks and anisotropic metal foam. Mathematics 2022;10(24):4774. http://dx.doi.org/10.3390/math10244774.

[53] Ogata K. Discrete-time control systems. 2nd ed.. Englewood Cliffs, N.J: Pearson; 1995.