Deep learning assisted physics-based modeling of aluminum extraction process

Haakon Robinson[a,1,*], Erlend Lundby[a,1], Adil Rasheed[a], Jan Tommy Gravdahl[a]

[a]*Department of Engineering Cybernetics, Norwegian University of Science and Technology, O. S. Bragstads plass 2, Trondheim, NO-7034, Norway*

Abstract

Modeling complex physical processes such as the extraction of aluminum is mainly done using pure physics-based models derived from first principles. However, the accuracy of these models can often suffer due to a partial understanding of the process, uncertainty in the input parameters, and numerous modeling assumptions. More recently, with the ever-increasing availability of data, there has been an explosion of interest in applying modern machine learning methods because of their ability to learn complex mappings directly from data. Unfortunately, these models tend to be black boxes, require an enormous amount of data, and do not utilize existing domain knowledge. In this work, we develop a novel approach combining physics-based and data-driven modeling approaches while eliminating some weaknesses. We use a data-driven model to correct a misspecified physics-based model of the Hall–Héroult process in an aluminum electrolysis cell using a corrective source term added to the set of governing ordinary differential equations. Our approach ensures that the existing knowledge is utilized to the maximum extent possible while relying on the data-driven models only to model those aspects which the physics-based model does not represent well. We compare this approach with an end-to-end learning approach and an ablated physics-based model, showing that the proposed hybrid method is more accurate, consistent, and stable for long-term predictions.

Keywords: Aluminum electrolysis, Sparse neural network (NN)s, Data-driven modeling, Nonlinear dynamics, Ordinary differential equations

1. Introduction

Many real-world phenomena can be modeled as differential equations, which allows us to predict the changes in the state of the system over time. These equations are often derived from first principles, and we refer to the resulting models as physics-based models (PBM). Through careful observation of physical phenomena, we can develop theories to describe and understand the underlying system. This understanding is condensed into mathematical equations, which can be solved to make predictions about the system.

In this work, we consider the case of aluminum electrolysis using a Hall-Héroult cell. In the process industry, state-of-the-art models are typically PBMs. For example, Gusberti et al. (2016) derive the model equations of an electrolysis cell from a mass and energy balance and a complete control volume analysis. Einarsrud et al. (2017) develop a multi-scale, multi-physics modeling framework based on three coupled models that predict the electromagnetic forces and metal pad profile, the bubble dynamics around a single anode, and the total cell bath flow, respectively. To reduce the computational requirements of such models, Johansen et al. (2022) propose a coarse-grained computational fluid dynamics (CFD) simulation that models the spatial dispersion rate of alumina.

PBMs have many inherent advantages. Due to their sound foundations from first principles, they are intuitive and explainable, they work well in operating conditions where the model assumptions are upheld, and there are mature theories that can be used to analyze their properties (e.g., stability and robustness to uncertainties and noise). However, accurately modeling many real-world systems comes at a high computational cost. Assumptions must be made to reduce the complexity of the model and minimize computational requirements. This is often necessary when developing control sys-

*Corresponding author

Email addresses: haakon.robinson@ntnu.no (Haakon Robinson), erlend.t.b.lundby@ntnu.no (Erlend Lundby), adil.rasheed@ntnu.no (Adil Rasheed), jan.tommy.gravdahl@ntnu.no (Jan Tommy Gravdahl)

[1]Equal contribution

tems or computing probabilistic state estimates for the system from noisy measurements (Pozna et al., 2010). We may also fail to describe aspects of the observations accurately. This can result in an incomplete, unfaithful, or overly simplified representation of the original system.

Data-driven modeling (DDM) is an alternative approach that does not base itself on an understanding of physics but instead attempts to approximate the underlying function directly from measurement data. Over the past decade, the rapid progress in machine learning has created a massive demand for data, with a supply to match. This has enabled the development of DDMs for a wide range of tasks, including the modeling and control of aluminum electrolysis processes. Meghlaoui et al. (1998) use a neural network to classify the internal state of an electrolysis cell and use this for feedback control. Chermont et al. (2016) apply a single hidden layer NN with more than 200 neurons to simulate the bath chemistry and temperature. de Souza et al. (2019) combine clustering methods and Deep Neural Network (DNN)s to create soft sensors of the bath temperature, aluminum fluoride, and metal level in the aluminum electrolysis. Bhattacharyay et al. (2017) model the effects of impurities in the carbon anode on the CO_2 reactivity of the anodes. Lundby et al. (2023) models important states of the aluminum electrolysis using DNNs, and studied the effect of sparsity promoting ℓ_1 regularization on model generalizability, interpretability, and stability. DDMs such as DNNs offer enormous flexibility, and it is often possible to achieve remarkable accuracy with relatively little computation, even when the underlying physics of a system is not fully understood. This reduces development costs and makes DDMs very attractive from an economic standpoint.

The downside of using DDMs is that they do not perform well in operating conditions that are not well represented by the training data, known as poor *generalization*. Many classes of DDMs also require unreasonably large amounts of data to reach sufficient accuracy and generalization. These drawbacks mean that when DDMs are used in practice, there is a preference for more transparent multivariate statistical models that can yield more insight into industrial processes. For example, Majid et al. (2011) use multivariate methods for state estimation of an aluminum smelting process, and Hedrea et al. (2021) propose the use of interpretable tensor-product methods to model nonlinear systems.

Combining PBM and DDM can help mitigate the disadvantages of both, as illustrated in Figure 1. This approach is referred to as hybrid analysis and modeling (HAM), although many other terms have been coined in the literature, such as Informed Machine Learning (von Rueden et al., 2023), Scientific Machine Learning (Rackauckas and Nie, 2017), and Structured Learning (Pineda et al., 2023). Interested readers are referred to (Rai and Sahu, 2020; von Rueden et al., 2020; Arias Chao et al., 2022; Bradley et al., 2022) for surveys of this field; some of the main approaches are reviewed below.

Structural methods are the most straightforward approach, where PBM is embedded into a differentiable framework such as PyTorch (Paszke et al., 2019). For example, Amos and Kolter (2017) insert a differentiable convex optimization solver into a NN, and Belbute-Peres et al. (2018) develop a differentiable physics simulator using a similar approach. Structural methods can serve as a powerful inductive bias for machine learning (ML) problems and as a way to incorporate constraints. A challenge is that the embedded PBMs are often iterative, which greatly increases the computational costs of training and inference. Other HAM approaches

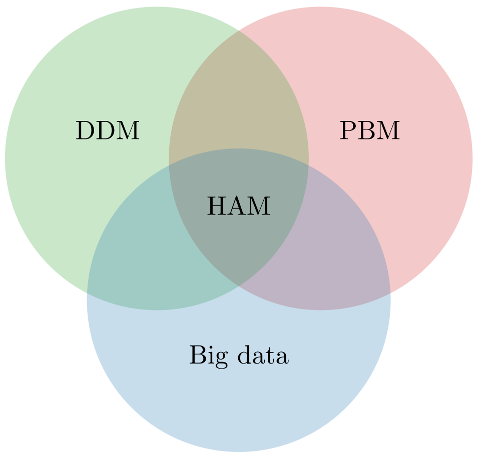


Figure 1: Hybrid analysis and modeling: working at the intersection of PBM, DDM and Big data.

introduce inductive biases into the training method instead of the model. For example, Raissi et al. (2018) propose the physics-informed neural network (PINN), where a NN acts as the solution to a partial differential equation (PDE) with specific boundary conditions. Each term in the PDE can be computed via automatic differentiation, allowing the specification of a loss function which can be introduced as a soft constraint for models that are additionally trained on measurement data. However, Krishnapriyan et al. (2021) show that optimizing such complex cost functions is challenging.

Sometimes the structure of the problem is unknown, but a solution can be constructed from a finite library of building block "primitives". Methods that find a useful combination of primitives are called search-based methods, as surveyed by Boussaïd et al. (2017). The family of methods that search for equations using a library of function primitives are known as *equation discovery* methods (Vaddireddy et al., 2020; Raviprakash et al., 2022). A notable work is SinDy, proposed by Brunton et al. (2016), which uses compressed sensing to obtain a sparse linear combination of functions. This line of research has inspired work by Bakarji and Tartakovsky (2021); Champion et al. (2019) that search for even sparser solutions. Udrescu et al. (2020) attempt to speed up the search by looking for symmetries in the data. Genetic programming or evolutionary approaches such as (Hachicha et al., 2011) are also popular approaches to this problem; see Zhong et al. (2017) for a more in-depth survey. However, equation discovery methods have only been demonstrated for relatively low dimensional examples and require significant computational time. Deep symbolic regression approaches proposed by Kim et al. (2021) and Xu et al. (2021) treat a NN itself as an expression tree, where the neurons in each layer have different activation functions representing the library of allowed functions. While this can quickly achieve good accuracy on higher dimensional data, the resulting expression trees are very dense and uninterpretable. A related concept, called physics-guided neural network (PGNN), addresses this by inserting promising features (i.e. that appear in existing PBMs) into the intermediate layers of a standard NN (Pawar et al., 2021a,b; Robinson et al., 2022). These additional features act as a store of prior knowledge that the network can utilize while still modeling the unknown physics as a black box and avoiding the overfitting that usually results from excessive feature engineering.

A common issue among the HAM methods presented above is that optimizing the models becomes significantly more expensive and demanding than a simpler DDM due to additional hyperparameters and the evaluation of PBMs during training. To avoid this, estimating or learning a *correction* to an existing PBM is possible. For example, Lundby et al. (2021) use compressed sensing methods to recover the model error from sparse measurements, which is used to improve state estimates for a Hall-Héroult cell. Blakseth et al. (2022b) propose the Corrective source term approach (CoSTA) approach, where a DDM is trained to correct the error of an existing PBM, which can be pre-computed for more efficient training. Blakseth et al. (2022b) show that this is sufficient to correct for various types of model error and apply CoSTA to simple one-dimensional heat transfer problems. Blakseth et al. (2022a) extend the work to 2D heat transfer problems and propose a mechanism to perform a "sanity check" of the CoSTA model. The additive nature of the CoSTA makes it easy to apply to any system where a PBM is available.

This work aims to study how an accurate model of a Hall-Héroult cell can be developed from an existing, but incorrect, PBM using a HAM approach. The incorrect PBM is constructed by modifying the "true" model and is corrected using the CoSTA method, where the corrective source term is parameterized as a NN. The corrector is trained on data sampled from the true system, and the results are compared with a pure DDM approach using a network with the same architecture. The contributions of this work are:

- Extending CoSTA to multidimensional problems: The previous works utilizing CoSTA were limited to modeling a single state temperature in either one or two-dimensional heat transfer.
- Successfully applying CoSTA to a system with external control inputs: None of the previous work involved any control inputs. In the current work, five inputs are used to excite the system.
- Investigating the predictive stability of CoSTA relative to end-to-end learning and showing that a hybrid approach can yield more trustworthy models.
- Demonstrating that CoSTA is applicable to a system with complex coupling between different states and inputs: the system considered here involves eight states and five inputs which form a set of eight ordinary equations which are highly coupled. The previous works involving heat transfer involved only one partial differential equation hence the potential of CoSTA to coupled problems was never evaluated earlier.

This paper is structured as follows. Section 2 presents the relevant theory behind the aluminum extraction process, NNs, and CoSTA. The data generation, training process, and evaluation are detailed in Section 3. The results are presented in Section 4, and the behavior of the system and the models is discussed. The main findings and future work are outlined in Section 5.

2. Theory

In the following section, we describe the underlying PBM for this system, the fundamentals of NNs, and the CoSTA approach to HAM.

2.1. Physics-based model for aluminum extraction

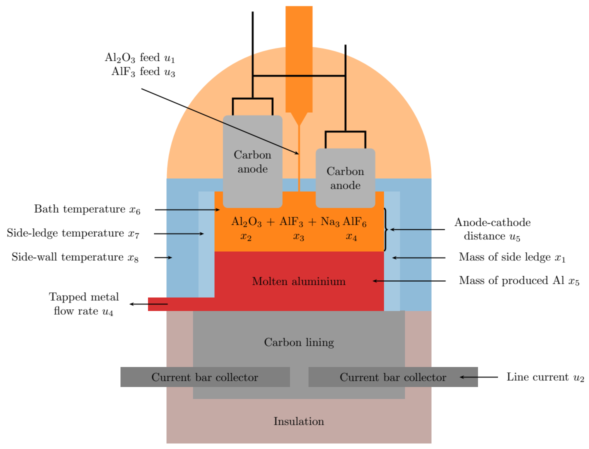


Figure 2: Schematic of the Hall-Héroult cell. Adapted with permission from (Lundby et al., 2023).

An overview of the physical plant is shown in Figure 2. A PBM of the plant can be derived from the mass/energy balance of the system. We omit this step and present the model directly. The internal dynamics of the aluminum electrolysis cell are described by a set of ordinary differential equations (ODE), with the general form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^8$ is the state vector, $\mathbf{u} \in \mathbb{R}^5$ are external inputs, and $f(\mathbf{x}, \mathbf{u})$ describes the nonlinear dynamics. Table 1 shows the names of the internal states and external inputs. The intrinsic properties of the Al_2O_3 + AlF_3 + $Na_3\,AlF_6$ mixture are determined by the mass ratios of x_2 (Al_2O_3) and x_3 (AlF_3), written as:

Table 1: Table of states and inputs

Variable	Physical meaning	Unit	Variable	Physical meaning	Unit
x_1	mass side ledge	kg	x_2	mass Al_2O_3	kg
x_3	mass AlF_3	kg	x_4	mass $Na_3\,AlF_6$	kg
x_5	mass metal	kg	x_6	temperature bath	$^\circ C$
x_7	temperature side ledge	$^\circ C$	x_8	temperature wall	$^\circ C$
u_1	Al_2O_3 feed	kg/s	u_2	Line current	kA
u_3	AlF_3 feed	kg/s	u_4	Metal tapping	kg/s
u_5	Anode-cathode distance	cm			

$$\begin{aligned} c_{x_2} &= x_2/(x_2 + x_3 + x_4) \\ c_{x_3} &= x_3/(x_2 + x_3 + x_4) \end{aligned} \tag{2}$$

We then define the following quantities:

$$\begin{aligned} g_1 &= 991.2 + 112c_{x_3} + 61c_{x_3}^{1.5} - 3265.5c_{x_3}^{2.2} \\ &\quad - \frac{793c_{x_2}}{-23c_{x_2}c_{x_3} - 17c_{x_3}^2 + 9.36c_{x_3} + 1} \end{aligned} \tag{3a}$$

$$g_2 = \exp\left(2.496 - \frac{2068.4}{273 + x_6} - 2.07c_{x_2}\right) \tag{3b}$$

$$\begin{aligned} g_3 &= 0.531 + 3.06 \cdot 10^{-18}u_1^3 - 2.51 \cdot 10^{-12}u_1^2 \\ &\quad + 6.96 \cdot 10^{-7}u_1 - \frac{14.37(c_{x_2} - c_{x_2,\mathrm{crit}}) - 0.431}{735.3(c_{x_2} - c_{x2,\mathrm{crit}}) + 1} \end{aligned} \tag{3c}$$

$$g_4 = \frac{0.5517 + 3.8168 \cdot 10^{-6}u_2}{1 + 8.271 \cdot 10^{-6}u_2} \tag{3d}$$

$$g_5 = \frac{3.8168 \cdot 10^{-6}g_3 g_4 u_2}{g_2(1 - g_3)} \tag{3e}$$

where g_1 is the liquidus temperature T_{liq}, g_2 is the electrical conductivity κ, g_3 is the bubble coverage , g_4 is the bubble thickness d_{bub} and g_5 is the bubble voltage drop U_{bub}. The critical mass ratio $c_{x_2,\mathrm{crit}}$ is given in Table 2.

The full PBM can now be written as a set of 8 ODEs:

$$\dot{x}_1 = \frac{k_1(g_1 - x_7)}{x_1 k_0} - k_2(x_6 - g_1) \tag{4a}$$

$$\dot{x}_2 = u_1 - k_3 u_2 \tag{4b}$$

$$\dot{x}_3 = u_3 - k_4 u_1 \tag{4c}$$

$$\dot{x}_4 = -\frac{k_1(g_1 - x_7)}{x_1 k_0} + k_2(x_6 - g_1) + k_5 u_1 \tag{4d}$$

$$\dot{x}_5 = k_6 u_2 - u_4 \tag{4e}$$

$$\begin{aligned} \dot{x}_6 &= \frac{\alpha}{x_2 + x_3 + x_4}\Bigg[u_2 g_5 + \frac{u_2^2 u_5}{2620 g_2} - k_7(x_6 - g_1)^2 \\ &\quad + k_8\frac{(x_6 - g_1)(g_1 - x_7)}{k_0 x_1} - k_9\frac{x_6 - x_7}{k_{10} + k_{11}k_0 x_1}\Bigg] \end{aligned} \tag{4f}$$

$$\begin{aligned} \dot{x}_7 &= \frac{\beta}{x_1}\Bigg[\frac{k_9(g_1 - x_7)}{k_{15}k_0 x_1} - k_{12}(x_6 - g_1)(g_1 - x_7) \\ &\quad + \frac{k_{13}(g_1 - x_7)^2}{k_0 x_1} - \frac{x_7 - x_8}{k_{14} + k_{15}k_0 x_1}\Bigg] \end{aligned} \tag{4g}$$

$$\dot{x}_8 = k_{17}k_9\left(\frac{x_7 - x_8}{k_{14} + k_{15}k_0 \cdot x_1} - \frac{x_8 - k_{16}}{k_{14} + k_{18}}\right) \tag{4h}$$

The constants (k_0, .., k_{18}, α, β) in Equation (4) are described and given numerical values in Table 2.

The model presented in Equation (4) makes some simplifications compared to the actual process of aluminum electrolysis. Firstly, only the heat transfer through the side walls is modeled, assuming that heat flow through the top and bottom of the plant is negligible in comparison. The model may thus overestimate the internal temperatures, and the required power input through the line current u_2 may be slightly lower than in practice. Secondly, the spatial variations of the state variables are not considered. Instead, only the average values of the states are computed, such as the side ledge temperature, or cumulative values like the mass of the side ledge x_1. Routine operations such as the alumina feeding and anode replacement disturb the local thermal balance and cause local thermal imbalances (Cheung et al., 2015). Modeling these local variations would require knowledge of the mass transfer inside the cell due to the flow patterns and velocity fields in the bath, current distribution, etc. These phenomena (corresponding to the orange ellipse of Figure 3) are challenging to model and measure and are therefore omitted to reduce complexity. Lundby et al. (2023) present a more detailed derivation of the model.

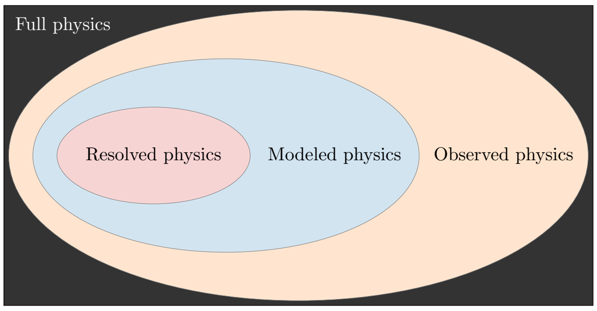


Figure 3: PBMs are limited by what we can observe and what we can compute. In other words, it is not always possible to observe the full physics of a system, the observations can be modeled with additional assumptions, and the model must be simplified and discretized to perform computations (i.e., "resolved"). Adapted with permission from (Blakseth et al., 2022a)

.

2.2. *Data-driven modeling using neural networks*

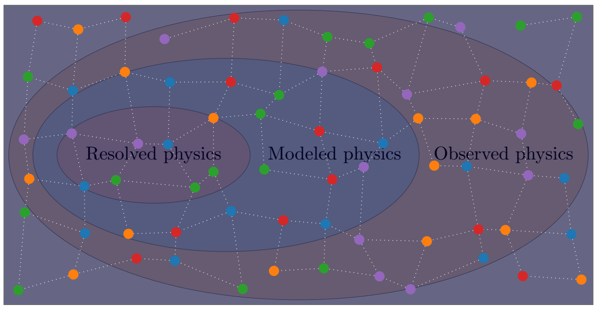


Figure 4: DDMs trained on the data will implicitly capture the complete physics of this process, assuming that the data is a realization of the underlying data-generating process. Adapted with permission from (Blakseth et al., 2022a)

.

Table 2: Constants in the simulator

Constant	Physical meaning	Numeric value
k_0	$1/(\rho_{\text{sl}} A_{\text{sl}})$	$2 \cdot 10^{-5}$
k_1	$2k_{\text{sl}} A_{\text{sl}} / \Delta_{\text{fus}} H_{\text{cry}}$	$7.5 \cdot 10^{-4}$
k_2	$h_{\text{bath-sl}} A_{\text{sl}} / \Delta_{\text{fus}} H_{\text{cry}}$	0.18
k_3	$0.002 \frac{M_{Al_2O_3} \cdot CE}{z \cdot F}$	$1.7 \cdot 10^{-7}$
k_4	$C_{Na_2O} \frac{4M_{AlF_3}}{3M_{Na_2O}}$	0.036
k_5	$C_{Na_2O} \frac{2M_{\text{cry}}}{3M_{Na_2O}}$	0.03
k_6	$0.002 \frac{M_{\text{Al}} \cdot CE}{z \cdot F}$	$4.43 \cdot 10^{-8}$
k_7	$k_2 \cdot c_{p_{\text{cry, liq}}}$	338
k_8	$k_1 \cdot c_{p_{\text{cry, liq}}}$	1.41
k_9	A_{sl}	17.92
k_{10}	$1/h_{\text{bath-sl}}$	0.00083
k_{11}	$1/(2k_{\text{sl}})$	0.2
k_{12}	$k_2 \cdot c_{p_{\text{cry, s}}}$	237.5
k_{13}	$k_1 \cdot c_{p_{\text{cry, s}}}$	0.99
k_{14}	$x_{\text{wall}}/(2k_{\text{wall}})$	0.0077
k_{15}	$1/(2k_{\text{sl}})$	0.2
k_{16}	T_0	35
k_{17}	$1/(m_{\text{wall}}\, c_{\text{p, wall}})$	$5.8 \cdot 10^{-7}$
k_{18}	$1/h_{wall-0}$	0.04
α	$1/c_{p_{\text{bath, liq}}}$	$5.66 \cdot 10^{-4}$
β	$1/c_{p_{\text{cry, sol}}}$	$7.58 \cdot 10^{-4}$
$c_{x_{2,\text{crit}}}$		0.022

Instead of making assumptions and fitting our theories to the data, DDMs can learn to approximate the underlying process directly from data. Figure 4 shows this conceptually. In this work, we model Equation (4) using a NN. A NN can be seen as a general function approximator. We denote the trainable parameters of the model as $\boldsymbol{\theta} \in \mathbb{R}^p$ and denote the network as

$$\mathbf{y} = \hat{\mathbf{f}}(\mathbf{z}; \boldsymbol{\theta}), \tag{5}$$

where $\mathbf{z} \in \mathbb{R}^d, \mathbf{y} \in \mathbb{R}^s$ are the inputs and outputs to the model respectively. The network is composed of several layers. The jth layer operates on the output of the previous layer and produces its own output, which we call $\mathbf{Z}^j \in \mathbb{R}^{L_j}$. A *fully connected layer* can be seen as an affine transformation composed with a nonlinear activation function $\sigma : \mathbb{R}^n \mapsto \mathbb{R}^n$

$$\mathbf{Z}^j = \sigma(\mathbf{W}^j \mathbf{Z}^{j-1} + \mathbf{b}^j), \tag{6}$$

where $\mathbf{W}^j \in \mathbb{R}^{L_j \times L_{j-1}}$ is called the *weight or connection matrix*, and $\mathbf{b}^j \in \mathbb{R}^{L_j}$ is the *bias vector* of layer j. We denote the rows of $\mathbf{W}^j$ as $\mathbf{w}_i^{j+1}$, and the individual bias terms as b_i^j. The nonlinear activation function σ can, for example, be the `sigmoid` function, hyperbolic tangent function (`tanh`), or the binary step function, to mention a few. All of these operate element-wise over $\mathbf{Z}^j$, but there exist functions that operate on groups on elements, e.g., the `maxout` activation function. Recently, the most popular activation function is `ReLU` due to its computational simplicity, representational sparsity, and non-vanishing gradients. The `ReLU` activation function is given by:

$$\sigma(z) = \max\{0,\ z\}. \tag{7}$$

From the previous section, it can be seen that NNs are dense models with many parameters. The largest networks in use today often have more parameters than the amount of available data to train them on. For example, the widely publicized GPT-3 model has 175 billion parameters (Brown et al., 2020). Because of this, avoiding overfitting and getting deep learning models to generalize is a vital topic in deep learning, and methods that accomplish this are generically referred to as *regularization* (Goodfellow et al., 2016). Examples of such methods include weight decay (Krogh and Hertz, 1991), dropout (Srivastava et al., 2014), and batch normalization (Ioffe and Szegedy, 2015), all of which are essential tools in ensuring a low generalization error for these models. In recent years, more and more research has shifted towards sparse architectures with significantly fewer non-zero trainable parameters than their dense counterparts (Hoefler et al., 2021). There are many reasons for this. Firstly, sparser networks are much cheaper to store and evaluate, making it easy to deploy them on low-cost hardware (Sandler et al., 2018). Secondly, recent work shows a tantalizing hint that sparse models may generalize better than their dense counterparts. In their seminal work, Frankle and Carbin (2019) show with high probability that randomly initialized dense NNs contain subnetworks that can improve generalization compared to the dense networks.

Many regularization methods can be expressed as a penalty function $R(\mathbf{w})$ that operates on the parameters $\boldsymbol{\theta}$ of the network. The total loss function $C(\mathbf{z}_i, \mathbf{y}_i, \boldsymbol{\theta})$ used for training the network can then be written as

$$C(\mathbf{z}_i, \mathbf{y}_i, \boldsymbol{\theta}) = L(\mathbf{y}_i, \hat{\mathbf{f}}(\mathbf{z}_i; \boldsymbol{\theta})) + \lambda R(\mathbf{w}), \tag{8}$$

where the set $\mathcal{D} = \{(\mathbf{z}_i, \mathbf{y}_i)\}_{i=1}^N$ is the training dataset, $L(\cdot,\cdot)$ is the *loss function* and $\lambda \in \mathbb{R}^+$ serves to trade-off $L(\cdot,\cdot)$ and $R(\cdot)$.

The standard choice of loss function $L(\cdot,\cdot)$ for regression tasks is the mean squared error (MSE):

$$L(\mathbf{z}_i, \mathbf{y}_i) = (\mathbf{z}_i - \mathbf{y}_i)^2. \tag{9}$$

In the training process, the total cost function $C(\cdot,\cdot)$ is minimized to find optimal values of the parameters:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ \frac{1}{N} \sum_{i=1}^{N} C(\mathbf{z}_i, \mathbf{y}_i, \boldsymbol{\theta}) \right\}. \tag{10}$$

For NNs, this optimization problem is typically solved using stochastic gradient descent (SGD). This is preferred over higher-order methods due to the memory required to compute or approximate higher-order derivatives for many parameters $\boldsymbol{\theta}$. Training iterations are performed for randomly sampled subsets of the full dataset, known as *minibatches*. This further reduces computational and memory requirements and allows the training process to be run in parallel more easily. Wilson and Martinez (2003) demonstrated that minibatch methods might also improve the generalization error of NNs. When all examples in $\mathcal{D}$ have been processed once, it is said that a *training epoch* has been completed. Training is continued for multiple epochs until some stopping criterion is met. Refer to Goodfellow et al. (2016) for a more in-depth presentation of deep-learning fundamentals.

The most straightforward way to penalize non-sparse $\boldsymbol{\theta}$ is the ℓ_0 norm, often referred to as the sparsity norm:

$$R_{\ell_0}(\mathbf{w}) = \|\mathbf{w}\|_0 = \sum_i \begin{cases} 1 & w_i \neq 0, \\ 0 & w_i = 0. \end{cases} \tag{11}$$

It is clear that $\ell_0(\boldsymbol{\theta})$ returns the number of nonzero parameters. It has been shown that adding this regularization term can yield unique solutions for overdetermined linear systems, which is the basis of compressed sensing (Boche et al., 2015). However, $\ell_0(\boldsymbol{\theta})$ is non-differentiable, making it unsuitable for gradient descent optimization. In fact, Natarajan (1995) show that this optimization problem is NP-hard (Natarajan, 1995). Instead, we can utilize the ℓ_1 norm, which is a convex relaxation of the ℓ_0 norm and is given by:

$$R_{\ell_1}(\mathbf{w}) = \|\mathbf{w}\|_1 = \sum_i |w_i|. \tag{12}$$

The ℓ_1 norm sometimes does not reduce the weights to zero, but rather to very small magnitudes. In this case, we can apply a threshold to the weights and set all weights below this threshold to zero. This method is known as *magnitude pruning* and is the simplest of a family of pruning methods (Hoefler et al., 2021). Despite the simplicity of this method, it can reduce the computation complexity of a NN while maintaining the performance of the model (Gale et al., 2013).

2.3. Corrective source term approach (CoSTA)

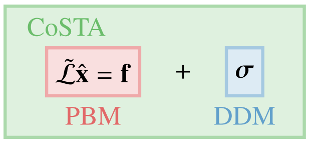


Figure 5: CoSTA combines PBM and DDM into a unified model by adding a NN-generated corrective source term to the governing equation of the PBM.

In this section we outline the CoSTA approach, illustrated in Figure 5. Consider the following general problem:

$$\mathcal{L}\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{13}$$

where $\mathcal{L}$ is a differential operator, $\mathbf{x}$ is the unknown state of the system that we wish to compute, and $\mathbf{f}(\cdot, \cdot)$ is a source term that depends on the state $\mathbf{x}$ and external inputs $\mathbf{u}(t)$.

Assume now that we have a PBM designed to predict $\mathbf{x}$, and let $\tilde{\mathbf{x}}$ denote the PBM's prediction of the true solution $\mathbf{x}$. If $\tilde{\mathbf{x}} \neq \mathbf{x}$, there is some error in the PBM, and this error must stem from at least one of the following misspecifications in the model:

1. Incorrect $\mathbf{f}$ in Equation (13), replaced by $\tilde{\mathbf{f}}$.
2. Incorrect $\mathcal{L}$ in Equation (13), replaced by $\tilde{\mathcal{L}}$.
3. A combination of the above.
4. Discretization of $\mathcal{L}$, replaced by $\mathcal{L}_{\mathcal{D}}$[2].

Note that case 4 is also mathematically equivalent to misspecifying $\mathcal{L}$. For example, $\frac{\partial}{\partial t}$ could be approximated using a finite forward difference. We can write this using the difference operator Δ_h, such that h is the time step and $\frac{1}{h}\Delta_h f(t) = (f(t+h) - f(t))/h$. We can therefore limit our discussion to Cases 1 and 2 without loss of generality.

Suppose now that the PBM-predicted solution $\tilde{\mathbf{x}}$ is given as the solution of the following system:

$$\tilde{\mathcal{L}}\tilde{\mathbf{x}} = \tilde{\mathbf{f}} \tag{14}$$

This formulation encompasses both Case 1 ($\tilde{\mathcal{L}} = \mathcal{L}$ and $\tilde{\mathbf{f}} \neq \mathbf{f}$), Case 2 ($\tilde{\mathcal{L}} \neq \mathcal{L}$ and $\tilde{\mathbf{f}} = \mathbf{f}$), and combinations thereof (for $\tilde{\mathcal{L}} \neq \mathcal{L}$ and $\tilde{\mathbf{f}} \neq \mathbf{f}$). Furthermore, suppose we modify the system above by adding a source term $\hat{\sigma}$ to Equation (14), and let the solution of the modified system be denoted $\hat{\tilde{\mathbf{x}}}$. Then, the modified system reads

$$\tilde{\mathcal{L}}\hat{\tilde{\mathbf{x}}} = \tilde{\mathbf{f}} + \hat{\sigma} \tag{15}$$

and the following theorem holds.

Theorem. Let $\hat{\tilde{\mathbf{x}}}$ be a solution of Equations (15), and let $\mathbf{x}$ be a solution of Equations (13). Then, for both operators $\tilde{\mathcal{L}}$, $\mathcal{L}$ and both functions $\mathbf{f}$, $\tilde{\mathbf{f}}$, such that $\hat{\tilde{\mathbf{x}}}$ and $\mathbf{x}$ are uniquely defined, there exists a function $\boldsymbol{\sigma}$ such that $\hat{\tilde{\mathbf{x}}} = \mathbf{x}$.

Proof: Define the residual $\boldsymbol{\sigma}$ of the PBM's governing equation (14) as[3]

$$\boldsymbol{\sigma} = \tilde{\mathcal{L}}\mathbf{x} - \tilde{\mathbf{f}}. \tag{16}$$

If we set $\hat{\boldsymbol{\sigma}} = \boldsymbol{\sigma}$ in Equation (15), we then obtain

$$\begin{aligned}
\tilde{\mathcal{L}}\hat{\tilde{\mathbf{x}}} &= \tilde{\mathbf{f}} + \hat{\sigma} && (17)\\
&= \tilde{\mathbf{f}} + \tilde{\mathcal{L}}\mathbf{x} - \tilde{\mathbf{f}} && (18)\\
&= \tilde{\mathcal{L}}\mathbf{x} && (19)\\
\implies \quad \hat{\tilde{\mathbf{x}}} &= \mathbf{x} + \mathbf{c} && (20)
\end{aligned}$$

where $\mathbf{c}$ is a function of independent variables. We can eliminate $\mathbf{c}$ by setting appropriate boundary conditions. ■

The theorem shows that we can always find a corrective source term $\hat{\boldsymbol{\sigma}}$ that compensates for any error in the PBM's governing equation (14) such that the solution $\hat{\tilde{\mathbf{x}}}$ of the modified governing equation (15) is equal to the true solution $\mathbf{x}$. This observation is the principal theoretical justification of CoSTA. As illustrated by Figure 6, the CoSTA approach should be applicable to many physical problems that can be described using differential equations.

[2]Derived using, for example, finite differences. This is necessary when Equation (13) lacks analytical solutions, which is almost always the case.

[3]Instead of defining the residual in terms of the approximate solution (e.g., as is done in truncation error analysis (LeVeque, 2002, chapter 8)), we define $\boldsymbol{\sigma}$ by inserting the true solution into Equation (13). Our proof is simpler and fits well with systems where state measurements are more readily available than the true governing equations.

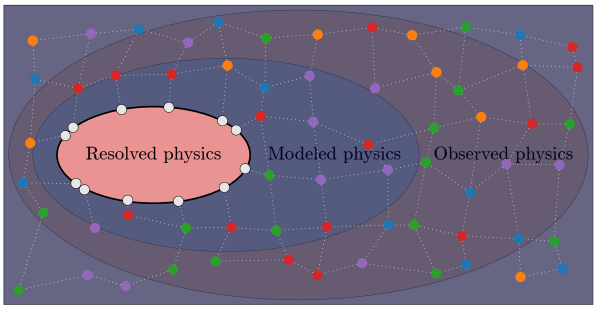


Figure 6: CoSTA maximizes the utilization of existing PBMs while correcting for the unknown using DDM. The PBM represents the resolved physics, a set of differential equations describing the system's state with unmodeled errors.

The data-driven corrective source terms capture all the unaccounted physics and unintentional numerical discretization errors.Adapted with permission from (Blakseth et al., 2022a).

3. Method and experimental setup

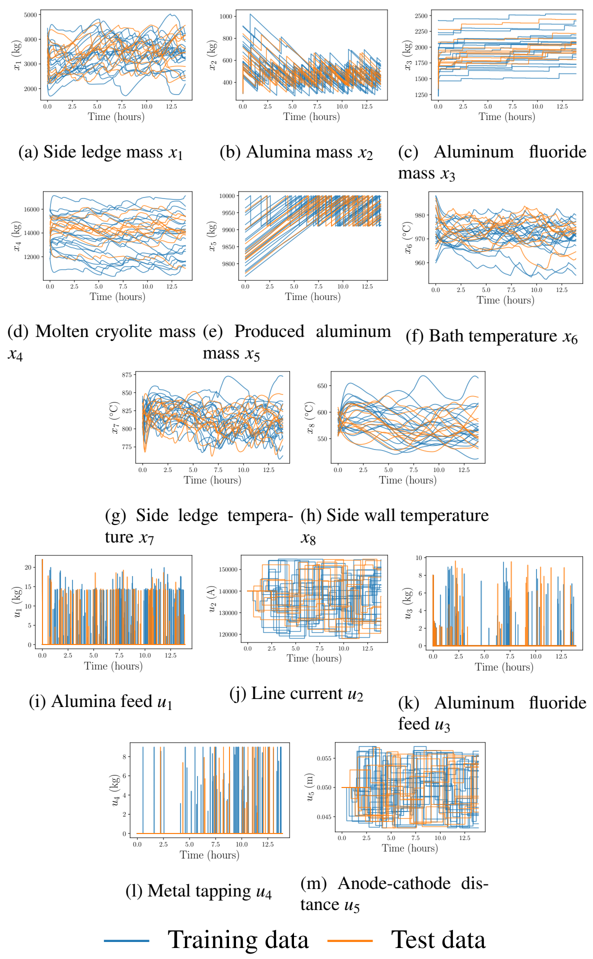
This section explains how data is generated, how modeling errors are induced in the PBM, the modeling approaches applied and compared in the case study, and the performance metrics used to evaluate the models used in the case study.

3.1. Data generation and preprocessing

The dynamical system data is generated by integrating the set of non-linear ODE's in Eq.(4) representing the system dynamics using the fourth-order numerical integrator Runge-Kutta 4 (RK4) with a fixed timestep $\Delta T = 10s$. One time-series simulation starts at an initial time t_0 with a set of initial conditions $\mathbf{x}(t_0)$, and last until a final time $T = 5000 \times \Delta T$. For the slow dynamics of the aluminum process, a sampling time of $10s$ turns out to be sufficiently fast with negligible integration errors. Higher sampling frequencies would lead to unnecessary high computational time and large amounts of simulation data. The initial conditions for each trajectory were uniformly sampled from the ranges shown in Table 3. Each simulation generates a set of trajectories with 8 states and 5 inputs. The training set consists of 40 simulated trajectories, and 100 simulated trajectories are used as the test set. This relatively large number of test cases was chosen to allow us to explore the statistics of how each model performs.

Table 3: Initial conditions for system variables. For x_2 and x_3, concentrations c_{x_2} and c_{x_3} are given.

Variable	Initial condition interval
x_1	$[2060,\ 4460]$
c_{x_2}	$[0.02,\ 0.05]$
c_{x_3}	$[0.09,\ 0.13]$
x_4	$[11500,\ 16000]$
x_5	$[9550,\ 10600]$
x_6	$[940,\ 990]$
x_7	$[790,\ 850]$
x_8	$[555,\ 610]$



(a) Side ledge mass x_1 (b) Alumina mass x_2 (c) Aluminum fluoride mass x_3

(d) Molten cryolite mass x_4 (e) Produced aluminum mass x_5 (f) Bath temperature x_6

(g) Side ledge temperature x_7 (h) Side wall temperature x_8

(i) Alumina feed u_1 (j) Line current u_2 (k) Aluminum fluoride feed u_3

(l) Metal tapping u_4 (m) Anode-cathode distance u_5

Figure 7: Training and test set trajectories of the system states. Only 10 random sample test trajectories are shown here to clarify the figures.

Figure 7 shows the complete training and test sets. The training set trajectories are blue, while the test set trajectories are orange. The figures show that the training set's range covers the test set's range, indicating that models are evaluated on interpolation cases in the test set.

3.1.1. *Estimation of the regression variable $\dot{\mathbf{x}}$*

The ODEs in Equation (4) are time-invariant. This means that at time $k+1$, $\dot{\mathbf{x}}_{k+1}$ in general only depends on the current state and input $(\mathbf{x}_k, \mathbf{u}_k)$ at time k. In other words, the system has the Markov property. The time derivatives at time k are estimated as the forward difference:

$$\dot{\mathbf{x}}_k = \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{h}, \tag{21}$$

where h is the time step. In this work, we use $h = 10$s. This numerical derivative induces a discretization error. However, since the dynamics of the aluminum electrolysis are slow, this error is considered negligible. In a pure DDM approach, the datasets are listed in pairs as follows

$$\mathcal{D}_{\text{DDM}} = \{((\mathbf{x}_k, \mathbf{u}_k), \dot{\mathbf{x}}_k)\}_1^N, \tag{22}$$

with N training pairs. In other words, the DDM aims to map $(\mathbf{x}_k, \mathbf{u}_k)$ to $\dot{\mathbf{x}}_k$. For the corrective source term model presented in Equation (15), the target variable is the error of the PBM model:

$$\mathcal{D}_{\text{CoSTA}} = \{(\mathbf{x}_k, \mathbf{y}_{k,\text{CoSTA}})\} = \{(\mathbf{x}_k, \mathbf{u}_k), (\dot{\mathbf{x}}_k - \hat{\dot{\mathbf{x}}}_{k,\text{PBM}})\}_1^N. \tag{23}$$

3.1.2. *Input signal generation*

While machine learning models are excellent for function approximation and interpolating data, they naturally only sometimes extrapolate correctly and are highly dependent on the quality and variety of the training data. Due to this, the training data must adequately cover the intended operational space of the system, i.e., the region of the state space where the system typically operates. Furthermore, the data should capture the different nonlinear trends of the system. For systems *without exogenous inputs*, variation can only be induced by simulating the system with different initial conditions $\mathbf{x}(t_0)$. For systems *with exogenous inputs*, the initial conditions are generated similarly. Moreover, the input vector $\mathbf{u}$ will excite the system dynamics. The aluminum process has a feedback controller that ensures safe and prescribed operation. However, operational data from a controlled, stable process is generally characterized by a low degree of variation which is insufficient for effective system identification. A well-known convergence criterion for identifying linear time-invariant systems is persistency of excitation (PE). A signal $\mathbf{x}(t_k)$ is PE of order L if all sub-sequences $[\mathbf{x}(t_k), \dots, \mathbf{x}(t_k + L)]$ span the space of all possible sub-sequences of length L that the system is capable of generating. While the PE criterion is not directly applicable to nonlinear systems, sufficient coverage of the dynamics is required for successful system identification (Ljung, 1998; Nelles, 2020). To this end, we add random perturbations to the control inputs to push the system out of its standard operating conditions. In general, each control input i is given by:

$$u_i = \text{Deterministic term} + \text{Random term}. \tag{24}$$

The control inputs u_1, u_3 and u_4 are impulses. The random term is zero for these control inputs when the deterministic term is zero. The deterministic term is a proportional controller. The control inputs u_2 and u_5 are always nonzero. These control inputs have constant deterministic and random terms that change periodically. The random term stays constant for a certain period ΔT_{rand} before changing to a new randomly determined constant. Different objectives must be considered when choosing the period ΔT_{rand}.

On the one hand, it is desirable to choose a large period ΔT_{rand} so that the system can stabilize and evolve under the given conditions to reveal the system dynamics under the given conditions. On the other hand, the systems must be tested under many different operational conditions. By empirically testing different periods ΔT_{rand}, and seeing how the dynamics evolve in simulation, it turns out that setting $\Delta T_{\text{rand}} = 30\Delta T$ is a fair compromise between the two. In this study, we generate the random disturbances using the Amplitude-modulated Pseudo-Random Binary Signal (APRBS) method (Winter and Breitsamter, 2018).

Table 4: Equations used to control the aluminum process

Input	Deterministic term	Random term interval	ΔT_{rand}
u_1	$3 \cdot 10^4(0.023 - c_{x_2})$	$[-2.0, 2.0]$	ΔT
u_2	$1.4 \cdot 10^4$	$[-7 \cdot 10^3, 7 \cdot 10^3]$	$30 \cdot \Delta T$
u_3	$1.3 \cdot 10^4(0.105 - c_{x_3})$	$[-0.5,\ 0.5]$	ΔT
u_4	$2(x_5 - 10^4)$	$[-2.0,\ 2.0]$	ΔT
u_5	0.05	$[-0.015,\ 0.015]$	$30 \cdot \Delta T$

Table 4 gives the numerical values of the deterministic term of the control input, the interval of values for the random terms, and the duration ΔT_{rand} of how long the random term is constant before either becoming zero (u_1, u_3, u_4) or changing to a new randomly chosen value (u_2, u_5).

3.2. *Modeling approaches*

The case study compares three different modeling approaches, namely a PBM approach using an ablated PBM, the hybrid modeling approach CoSTA - combining the ablated PBM with a DNN, and a purely DDM approach - modeling the entire set of ODEs with DNNs.

Comparing the CoSTA approach with a PBM and a DDM approach will reveal the effect of the CoSTA approach.

3.2.1. Ablated PBM

As previously discussed, we are interested in modeling scenarios where the PBM does not capture the full underlying physics of the system. For this case study, the PBM is described by Equation (4), with modifications to induce modeling errors so that the PBM deviates from the simulation model. That is, Equation (3a) describing the liquidus temperature g_1 was ignored, and g_1 was set to a constant.

$$g_{1,\text{PBM}} = 968^{\circ}C. \tag{25}$$

The resulting model, which ignores the dynamics of the liquidus temperature, is referred to as the *ablated PBM*. This variable was chosen because the model is particularly sensitive to errors in g_1. Inspecting Equation (4) shows that the ablated PBM will incorrectly predict the evolution of $[x_1, x_4, x_6, x_7, x_8]$. As we will see later in Section 4 and Figure 10, this can lead to errors of roughly 5°C in g_1, and 500kg in the side ledge mass x_1 (a relative error of 10%). The PBM used in the case study is a set of ODEs on the general form:

$$\hat{\dot{\mathbf{x}}}_k = \mathbf{f}_{\text{PBM}}(\mathbf{x}_k,\ \mathbf{u}_k), \tag{26}$$

where $\hat{\dot{\mathbf{x}}}_k$ is the PBM estimate of the time derivative at timestep k, $f_{\text{PBM}}(\cdot,\ \cdot)$ is the ablated PBM, and $\mathbf{x}_k$ and $\mathbf{u}_k$ are state variables and control inputs at timestep k.

3.2.2. DDM

The DDM approach models the time derivative of the state, and has the following form:

$$\hat{\dot{\mathbf{x}}}_k = \mathbf{f}_{\text{DDM}}(\mathbf{x}_k,\ \mathbf{u}_k). \tag{27}$$

In the case study, the DDM $\mathbf{f}_{\text{DDM}}(\cdot,\ \cdot)$ is a DNN. The DNN is trained on the training set $\mathcal{D}$ in Equation (22).

3.2.3. CoSTA

The case study aims to develop a DDM to correct the ablated PBM using measurement data sampled from the true model. The resulting hybrid model CoSTA presented on a general form in Equation (15) consists of the PBM in Equation (26) and a DDM that is meant to correct the misspecified PBM. The CoSTA in this case study used to model the set of ODEs in Equation (4) is given by:

$$\hat{\dot{x}}_k = \mathbf{f}_{\text{CoSTA}}(\mathbf{x}_k,\ \mathbf{u}_k) = \mathbf{f}_{\text{PBM}}(\mathbf{x}_k,\ \mathbf{u}_k) + \mathbf{f}_{\text{corr}}(\mathbf{x}_k,\ \mathbf{u}_k). \tag{28}$$

$\mathbf{f}_{\text{corr}}$ is a DNN that aims to correct the errors in the PBM. The parameters of the corrective source term $\mathbf{f}_{\text{corr}}$ are learned through training, using the manipulated training set in Equation (23).

3.3. DNN Training

The PBM in CoSTA will typically reduce the complexity of the learning problem compared to the learning problem of a pure DDM approach. In light of this, it is interesting to see the effects of sparsity promoting ℓ_1 regularization, which is known to lead to sparser models and better generalization with less available data, while maintaining model accuracy. The case study includes two versions of the DNN in the CoSTA, namely:

- Dense $\mathbf{f}_{\text{corr}}$
- Sparse $\mathbf{f}_{\text{corr}}$

where $\mathbf{f}_{\text{corr}}$ is the corrective source term in Equation (28). In order to compare the results with a purely DDM approach, the same two versions of complexity are used in the DDM approach in Equation (27), that is:

- Dense $\mathbf{f}_{\text{DDM}}$
- Sparse $\mathbf{f}_{\text{DDM}}$

The architecture of all networks, both in CoSTA and the DDM approach, was $[13, 20, 20, 20, 20, 8]$ (13 inputs, 8 outputs, 4 hidden layers with 20 neurons each). The `ReLU` activation function was used for all layers except the output layer, which had no activation function. The same architecture was used for all networks for a fairer comparison. The models were trained on the training set using the total-loss function shown in Equation (8), where the loss function $L(\cdot, \cdot)$ is the MSE as shown in Equation (9).

The ADAM optimizer, a popular SGD method with adaptive learning rates proposed by Kingma and Ba (2017), was used with the following default parameters: Initial learning rate $\eta = 10^{-3}$, Gradient forgetting factor $\beta_1 = 0.9$, and Gradient second-moment forgetting factor $\beta_2 = 0.999$. The dense networks were trained with $\lambda = 0$, and sparse networks with $\lambda = 10^{-4}$. All models were trained for 100 epochs (an epoch is defined as one full pass over the dataset). This value was chosen empirically during initial training attempts by inspecting training loss curves and selecting a rough average of optimal training times. An alternative would be *early stopping*, where training is terminated when performance on an additional validation dataset begins to drop. However, Sjöberg and Ljung (1995) and Bishop

(1995) showed that early stopping could have an additional regularizing effect by constraining the parameter space. Therefore, early stopping was not used to maintain similar training conditions over all experiments.

3.4. Performance metrics

This work focuses on long-term forecast error as a performance measure. Starting with the initial condition $\mathbf{x}(t_0)$, the next states in the trajectory are estimated iteratively $\{\hat{\mathbf{x}}(t_1), \ ..., \ \hat{\mathbf{x}}(t_n)\}$. The predicted trajectory is called a *rolling forecast*. The model estimates the time derivatives of the states $d\hat{\mathbf{x}}_\mathbf{i}/dt$ based on the current state $\mathbf{x}(t_i)$ and control inputs $\mathbf{u}(t_i)$ and initial conditions $\mathbf{x}_0 = \mathbf{x}(t_0)$, or the estimate of the current state variables $\hat{\mathbf{x}}(t_i)$ if $t > t_0$:

$$\frac{d\hat{\mathbf{x}}(t_i)}{dt} = \begin{cases} \hat{\mathbf{f}}(\hat{\mathbf{x}}(t_i), \ \mathbf{u}(t_i)), & \text{if } t_i > t_0 \\ \hat{\mathbf{f}}(\mathbf{x}_0(t_i), \ \mathbf{u}(t_i)), & \text{if } t_i = t_0 \end{cases} \tag{29}$$

Then, the next state estimate $\mathbf{x}(t_{i+1})$ is calculated as

$$\hat{\mathbf{x}}(t_{i+1}) = \hat{\mathbf{x}}(t_i) + \frac{d\,\hat{\mathbf{x}}(t_i)}{dt} \cdot \Delta T. \tag{30}$$

3.4.1. Model accuracy measure

The rolling forecast can be computed for each of the states x_i for one set of test trajectories $\mathcal{S}_{\text{test}}$. However, presenting the rolling forecast of multiple test sets would render the interpretation difficult. By introducing a measure called Average Normalized Rolling Forecast Mean Squared Error (AN-RFMSE) that compresses the information about model performance, the models can be evaluated quickly on many test trajectories. The AN-RFMSE is a scalar defined as:

$$\text{AN-RFMSE} = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{n}\sum_{j=1}^{n}\left(\frac{\hat{x}_i(t_j) - x_i(t_j)}{\text{std}(x_i)}\right)^2, \tag{31}$$

where $\hat{x}_i(t_j)$ is the model estimate of the simulated state variable x_i at time step t_j, $std(x_i)$ is the standard deviation of variable x_i from the training set $\mathcal{S}_{\text{train}}$, $p = 8$ is the number of state variables, and n is the number of time steps the normalized rolling forecast MSE is averaged over. Hence, for every model $\hat{\mathbf{f}}_j$ and every test set time series $\mathcal{S}_{\text{test}}(i)$, there is a corresponding AN-RFMSE.

3.4.2. Model stability measure

When performing a rolling forecast, the predicted state may reach a region of the state space where the model is unstable. This may arise because there is no data in that region of the state space or because the true model is unstable. At this point, the predicted trajectory will diverge, and the error will grow exponentially. This phenomenon is referred to as a *blow-up*. The open loop instability of the model can be quantified by counting the number of blow-ups that occur within a finite time horizon. In this work, a blow-up is defined using the following criterion:

$$\max_{j<n}\left[\frac{1}{p}\sum_{i=1}^{p}\left(\frac{\left|\hat{x}_i(t_j) - x_i(t_j)\right|}{\text{std}(x_i)}\right)\right] > 3 \tag{32}$$

where $p = 8$ is again the number of state variables and n is the number of time steps to consider. In other words, a blow-up is said to occur when the mean absolute error for all states and timesteps exceeds three standard deviations of the test set. Although this estimate is conservative, the number of blow-ups is not underestimated due to the exponential growth of the error.

4. Results and discussion

As described in Section 3.1, the test set consists of 100 simulated trajectories, each with a length of 5000 timesteps, with a timestep of $\Delta T = 10sec$. Each model is used to perform a rolling forecast given the initial conditions and input signal of each test trajectory. The experiments were repeated 10 times with different random initializations of the trainable parameters to increase the statistical significance of the results. The 4 model types evaluated in the case study that consists of DNNs are the dense and sparse DNN model structures in the CoSTA(”CoSTA dense” and ”CoSTA sparse” in Figure 8 and Figure 9), and dense and sparse DNN model structures in the DDM approach (”DDM dense” and ”DDM sparse” in Figure 8 and Figure 9). These model structures are described in Section 3.3. The ablated PBM for the CoSTA method has no trainable parameters, and remained the same in all experiments.

Figure 8 shows violin plots of the AN-RFMSE values defined in Equation (31) for all model types. The AN-RFMSE violin plots are shown at three different prediction horizons to demonstrate the models’ short-term, medium-term, and long-term performance. The AN-RFMSE values included in constructing a single violin plot for a given horizon and model type are based on the forecasts of the test-set trajectories up until the given forecasting horizon, and blow-ups are omitted as outliers. A possible issue here is that excluding these AN-RFMSE values favors the models that have many blow-ups. However, the results also show high blow-up rates correlate with high AN-RFMSE. Figure 9 shows

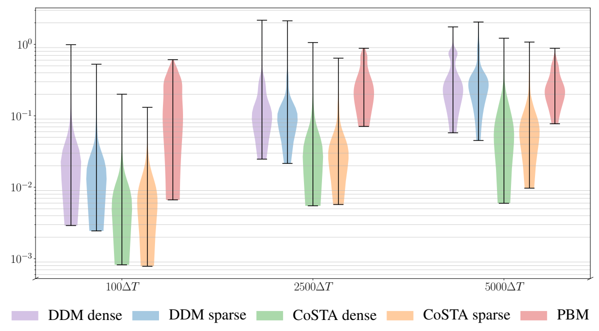


Figure 8: Violin-plot of the AN-RFMSE for all model types for 100 different initial conditions and inputs signals. The width of the bar reflects the distribution of the data points, and the error bars represent the range of the data. The error is shown at three different cutoffs to compare the short, medium, and long-term performance. We trained 10 different instances for each model type for statistical significance. The results show that CoSTA improves the predictive accuracy over the whole trajectory. Introducing sparse regularization appears to improve performance for DDM. However, it only appears to affect CoSTA models in the long term, where sparse CoSTA models appear to have less variance.

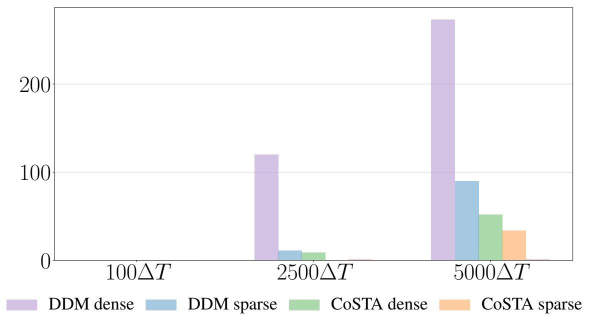


Figure 9: Bar chart of the number of times model estimates blow up and diverges. The plot contains 100 different initial conditions and input signals for all model types. The number of blow-ups was counted at three different cutoffs to compare the short, medium, and long-term performance. We trained 10 different instances for each model type for statistical significance. It can be seen that CoSTA increases the predictive stability in the long term, i.e., the number of blow-ups for CoSTA models is far less than the number of blow-ups for DDM. However, PBM does not suffer from significantly fewer blow-ups than CoSTA.

the frequency of blow-ups for each model type, based on the blow-up measure defined in Equation (32).

These results show that, on average, all DDM and CoSTA models have a lower AN-RFMSE than the ablated PBM in the short and medium term. However, all DDM and CoSTA models experience some blow-ups in the long term, which the PBM model does not. The dense DDM fared the worst, with 27.3% of long-term forecasts blowing up. The sparse DDM marginally improves on the AN-RFMSE, but we found that the blow-up rate was significantly reduced in the long term compared to the dense DDM. Both dense and sparse CoSTA models were significantly more accurate than the DDM models. The sparse CoSTA had similar accuracy to the dense CoSTA models in the short and medium term. However, the sparse CoSTA model had no blow-ups in the short and medium term and had half the blow-up rate of the Sparse DDM in the long term. These experiments demonstrate that CoSTA can reliably correct misspecified PBMs and significantly improves predictive stability compared to end-to-end learning. The base PBM does not exhibit any blow-up issues, suggesting that the blow-ups can be attributed to the NNs used in this work. If long-term forecasts are required ($>$ 3000 timesteps), we recommend combining the CoSTA approach with a sanity check mechanism to detect potential blow-ups.

Figure 8 and Figure 9 summarize the main results of the case study as they include forecasts of all test-set trajectories. To illustrate and elaborate on the effect of the results when forecasting without feedback from measurements, we have included plots of forecasts of a single, representative test trajectory. Figure 10 shows the mean predictions for each model type for the representative test trajectory, along with a 99.7% confidence interval to show the spread of the predictions from the 10 instances of each model type. Only the DDM and CoSTA models trained with ℓ_1 regularization are shown for clarity. Before discussing the differences between the models, we will describe the system's dynamics and how the incorrect PBM behaves in comparison. First, note that all variables are non-negative, reflecting different physical quantities in the system, i.e., mass, temperature, and current. Inspecting Equation (4), we see that the states x_2, x_3, and x_5 are linearly dependent on u_1, u_2, u_3, and u_4. We refer to these as the *linear states*, and the rest as the *nonlinear states*.

Liquidus temperature g_1:. Figure 10i shows the true liquidus temperature g_1 (in black) and the constant PBM estimate of the liquidus temperature (in red dotted line). The liquidus temperature g_1, which is the temperature at which the bath solidifies, is determined by the chemical composition of the bath. That is, g_1 is determined by the mass ratios between x_2, x_3, and x_4. The fact that PBM assumes g_1 to be constant induces modeling errors for the PBM.

Mass of side ledge x_1:. Figure 10a shows the mass of frozen cryolite ($Na_3\,Al\,F_6$), or side ledge. The solidification rate $\dot{x}_1$ is proportional to the heat transfer $Q_{\text{liq-sl}}$ through the side ledge ($Q_{\text{liq-sl}} \sim \left(\frac{g_1 - x_7}{x_1}\right)$) minus the heat transfer $Q_{\text{bath-liq}}$ between the side ledge and the bath ($Q_{\text{bath-liq}} \sim (x_6 - g_1)$). The solidification rate $\dot{x}_1$ is dependent on the value of g_1, and therefore the PBM incorrectly predicts the mass rate $\dot{x}_1$. In Figure 10a, we see that the PBM modeling error for x_1 starts to increase after approximately one hour as the true liquidus temperature g_1 drifts away from the constant PBM estimate of g_1. Figure 10i shows that the PBM overestimates g_1. Therefore, the PBM also overestimates the heat transfer out of the side ledge, leading to a higher estimate of frozen cryolite and side ledge mass. However, this modeling error is limited by the effect that an increased side ledge mass (and therefore increased side ledge thickness) leads to better isolation. Thus, the PBM estimate of the heat transfer through the side ledge $Q_{\text{liq-sl}}$ is inversely proportional to the x_1 estimate, and the modeling error of x_1 reaches a steady state for a constant modeling error in g_1. In addition to modeling errors due to errors in the g_1 estimate, modeling errors of x_6 and x_7 propagate to $\dot{x}_1$.

Both the mean of DDM and the mean of CoSTA models appear to predict the response of x_1 correctly. The variance of both model classes grows over time, with the DDM models growing roughly twice as fast as the CoSTA models. Furthermore, CoSTA and DDM show some cases where the error bound becomes significantly large, indicating that one or more models blow up. For the DDM models, these cases appear more frequently, and the errors are more significant than for the CoSTA models. Figures 10f and 10g show that these error peaks often coincide with the peaks in the bath temperature x_6 and the side ledge temperature x_7.

Mass of alumina x_2:. Figure 10b shows the mass of aluminum in the bath. Equation (4) shows that $\dot{x}_2$ (mass rate of Al_2O_3) is proportional to u_1 (Al_2O_3 feed), and negatively proportional to u_2. Figure 10b shows that this yields a saw-tooth response that rises as u_1 spikes and decays with a rate determined by u_2. This state has no dependence on g_1 nor on other states that depend on g_1. Therefore the PBM (and CoSTA) predict this state with no error. On the other hand, the spread of the DDM models grows over time, with the mean error eventually becoming significant.

Mass of aluminum fluoride x_3:. The x_3 state (mass of AlF_3) acts as an accumulator, rising when AlF_3 is added to the process (u_3 spikes), and falling when Al_2O_3 is added to the process (u_1 spikes). The latter is caused by impurities (Na_2O) in the Alumina (Al_2O_3) reacting with AlF_3, generating cryolite (Na_3AlF_6). As shown in Figure 10c, the latter effect is relatively small. Despite this, the DDM appears to model these decreases correctly. However, the DDM models become less and less accurate as time passes. The PBM and CoSTA model x_3 with no error.

Mass of molten cryolite x_4:. This state represents the mass of molten cryolite in the bath, where $\dot{x}_4 = k_5 u_1 - \dot{x}_1$. The first term represents additional cryolite generated by reactions between impurities in the added alumina (u_1) and AlF_3 (x_3). The second term describes how the cryolite can freeze (x_1) on the side ledge, which can melt again as the side-ledge temperature x_7 increases. As seen in Figure 10d, the response of x_4, therefore, mirrors that of x_1, with relatively small upturns when alumina is added (u_1). Inspecting Figure 10a, we see that the models behave similarly. Incorrectly estimating x_4 causes some issues. The mass ratio c_{x_2} (see Equation (2)) is important in terms of determining the cell voltage U_{cell}. A forecasting error of x_4 will propagate as a forecasting error of c_{x_2}, leading to inaccurate estimates of the cell voltage U_{cell}.

Mass of produced metal x_5:. This linear state also has a saw-tooth characteristic, growing at a rate proportional to the line current (u_2) and falling when metal is tapped (u_4 spikes). Looking at Figure 10e, the DDM models have similar error dynamics to the other linear states, while the PBM and CoSTA models have virtually no error.

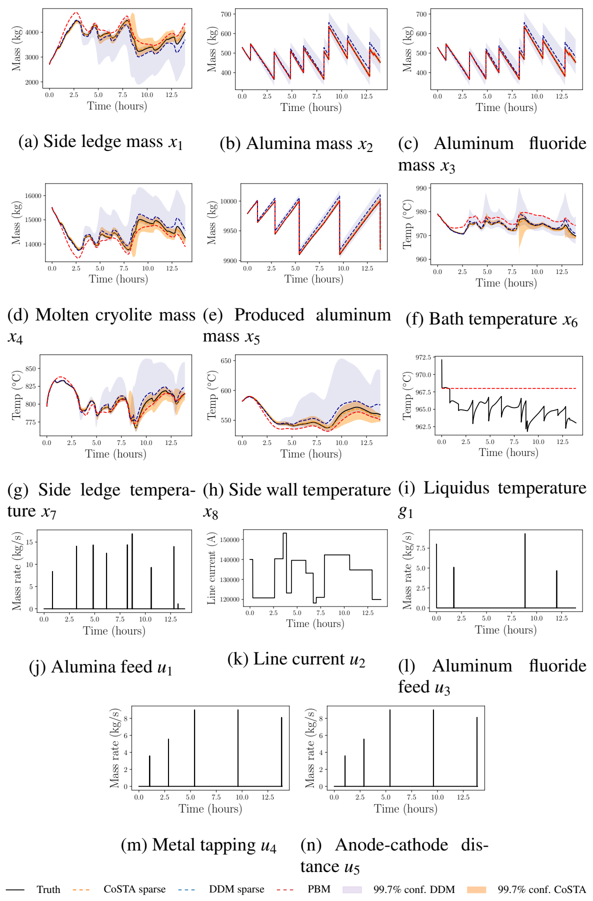
Temperature in the bath x_6:. There are several possible sources of PBM modeling errors of the bath temperature x_6. As discussed earlier, since the PBM overestimates the side ledge thickness due to a modeling error of g_1. It follows that the PBM overestimates the thermal insulation of the side ledge, leading to an overestimation of the bath temperature and an underestimation of the heat transfer out of the bath. In Figure 10f, we see this overestimate of x_6 provided by the PBM after approximately one hour, simultaneously as the PBM starts to overestimate the side ledge mass x_1.

Furthermore, the change in bath temperature $\dot{x}_6$ is determined by the energy balance in the bath. The energy balance in the bath consists of several components, namely the electrochemical power P_{el} which adds energy to the system, the heat transfer from the bath to the side ledge $Q_{\text{bath-sl}}$ which transports energy out of the bath, and the energy $E_{tc,\text{liq}}$ required to break interparticle forces in the frozen cryolite liquidus temperature. The electrochemical power $P_{\text{el}} = U_{\text{cell}} \cdot u_2$ is the product of the cell voltage U_{cell} and the line current u_2. The cell voltage is given by $U_{\text{cell}} = \left(g_5 + \frac{u_2 u_5}{2620 g_2}\right)$, where

g_5 is the bubble voltage drop, and $\frac{u_2 u_5}{2620 g_2}$ is the voltage drop due to electrical resistance in the bath. The bubble voltage drop g_5 increases exponentially as the mass ratio of alumina - c_{x_2} approaches the critical mass ratio of alumina $c_{x_2,\text{crit}} \sim 2$. This process upset is known as an *anode effect*. This can explain the peaks in the x_6 predictions, which are most present for the DDM models. As shown in Figure 10f, the peaks of the error band for the DDM coincide with overestimates of x_4 (see Figure 10d), indicating that the DDM incorrectly predicts anode effects in these cases. Moreover, the voltage drop due to electrical resistance is given by $\frac{u_2 u_5}{2620 g_2}$, where u_2 is the line current, u_5 is the Anode-Cathode Distance (ACD), $2620[m^2]$ is the total surface of the anodes and g_2 is the electrical conductivity. Within reasonable operational conditions, $\frac{1}{g_2}$ can be approximated as a function that increases linearly with the increasing mass ratio of alumina c_{x_2}. The modeling error in x_4 can therefore propagate to x_6. After approximately eight hours, the error bound of CoSTA models shows that at least one of the CoSTA models calculates an instantaneous overestimate of x_6, followed by an immediate underestimate of x_6. A possible explanation is that the CoSTA model first erroneously predicts the anode effect. The underestimate of x_6 that instantaneously follows can be caused by an underestimate of c_{x_2} that is lower than $c_{x_2,\text{crit}}$ which leads to negative P_{el} values in the model.

Temperature in the side ledge x_7:. The change of temperature in the side ledge $\dot{x}_7$ is determined by the heat balance in the side ledge. This includes the heat transfer from the bath to the side ledge $Q_{\text{liq-sl}}$, the heat transfer from the side ledge to the side wall $Q_{\text{sl-wall}}$, and the energy $E_{tc,\text{sol}}$ required to heat frozen side ledge to liquidus temperature from side ledge temperature. The change in the side ledge temperature depends on the side ledge thickness x_1, the bath temperature x_6, the side ledge temperature x_7, the wall temperature x_8 and the liquidus temperature g_1. As argued above, for the PBM modeling errors in x_1, x_6, x_7, x_8, and g_1 will propagate as modeling errors in the side ledge temperature change $\dot{x}_7$. For the DDM and CoSTA models, the error in x_7 shown in Figure 10g mainly grows as the error in x_6 spikes, presumably caused by erroneously predicted anode effects, as explained above.

Temperature in the wall x_8:. Figure 10h shows that The temperature of the side wall x_8 changes according to the heat transfer from the side ledge to the wall $Q_{\text{sl-wall}}$, and the heat transfer from the wall to the ambient Q_{wall-0}. Changes in the wall temperature $\dot{x}_8$ depend on the side ledge temperature x_7, the wall temperature x_8, and the side ledge thickness x_1. PBM modeling errors of these states at time k propagate as modeling errors in the side wall temperature x_8 in the next time step, $k+1$. Hence, with correct inputs, the PBM will always model the correct $\dot{x}_8$ since the PBM model of $\dot{x}_8$ is equal to the simulator.



(a) Side ledge mass x_1 (b) Alumina mass x_2 (c) Aluminum fluoride mass x_3

(d) Molten cryolite mass x_4 (e) Produced aluminum mass x_5 (f) Bath temperature x_6

(g) Side ledge temperature x_7 (h) Side wall temperature x_8 (i) Liquidus temperature g_1

(j) Alumina feed u_1 (k) Line current u_2 (l) Aluminum fluoride feed u_3

(m) Metal tapping u_4 (n) Anode-cathode distance u_5

Figure 10: Rolling forecast of a representative test trajectory. 10 CoSTA models with sparse corrective NNs, 10 DDMs consisting of sparse NN models, as well as a PBM, are predicting the test set trajectories given the initial conditions and the input vector at any given time.

5. Conclusions and future work

In this work, we presented a recently developed approach in modeling called the Corrective Source Term Approach (CoSTA). CoSTA belongs to a family of hybrid analysis and modeling (HAM) tools where physical-based models (PBM) and data-driven models (DDM) are combined to exploit the best of both approaches while eliminating their weaknesses. The method was applied to model an aluminum extraction process governed by complex physics. First, a ground truth dataset was generated using a detailed high-fidelity simulator. Then, an ablated model was created by setting an internal variable of the simulator to a constant. Finally, the ablated model was supplemented with a corrective source term modeled using a NN that compensated for the ignored physics. The main conclusions from the study are as follows:

- CoSTA, in all the scenarios investigated, could correct for the ignored physics and was consistently more accurate in predicting all eight states of the process than the PBM and DDM over a reasonably long time horizon.
- Both end-to-end learning and CoSTA captured the complex coupling between states and inputs.
- CoSTA consistently yields more stable predictions when compared to pure DDM, despite the numerous input signals being very sparse and discontinuous.
- Regularizing the networks using ℓ_1 weight decay was effective in improving model stability in both DDM and CoSTA.

One significant benefit of the CoSTA method is that it can utilize domain knowledge, only relying on black-box DDMs to model unknown or poorly understood physics. A limitation of this work is the assumption that full-state measurements are available, which only sometimes holds in practice. In the case of unobservable variables and noisy measurements, the full state trajectories can be estimated using a PBM, e.g., by solving a moving horizon estimate problem. Then a corrective source term can be trained to correct the PBM, given the estimated trajectories. If hidden states exist, obtaining enough information to reconstruct the system's dynamics is still possible by augmenting the measurements with additional information, i.e., lookback states from previous time steps. Takens' Theorem gives an upper bound on the number of necessary lookback states (Takens, 1981).

Although it remains to be investigated in future work, it can be expected that much simpler models will be sufficient for modeling the corrective source terms. The properties of these source terms can then be analyzed to achieve additional insight into the system. Even when it is not possible to interpret the source terms, it should still be possible to place bounds on their outputs using domain knowledge. These bounds can then serve as an inbuilt sanity check mechanism in the system. For example, since the amount of energy put into the system is known, the source terms for the energy equation will be bounded, so any NN-generated source term violating this bound can be confidently rejected, making the models more suitable for safety-critical applications like the one considered here. Another topic worth investigating is the robustness of the method to noise.

Acknowledgments

This work was supported by the project TAPI: Towards Autonomy in Process Industries (grant no. 294544), and EXAIGON: Explainable AI systems for gradual industry adoption (grant no. 304843)

Author contributions

Haakon Robinson: Methodology, Investigation, Software, Visualization, Writing (Original Draft) ***Erlend Lundby***: Methodology, Investigation, Software, Visualization, Writing (Original Draft) ***Adil Rasheed***: Conceptualization, Writing (Review & Editing), Supervision, Funding acquisition ***Jan Tommy Gravdahl***: Supervision, Writing (Review & Editing), Funding acquisition. First and second authors made equal contributions.

References

Amos, B., Kolter, J.Z., 2017. OptNet: Differentiable optimization as a layer in neural networks, in: Proceedings of the 34th International Conference on Machine Learning - Volume 70, JMLR.org, Sydney, NSW, Australia. pp. 136–145.

Arias Chao, M., Kulkarni, C., Goebel, K., Fink, O., 2022. Fusing physics-based and deep learning models for prognostics. Reliability Engineering & System Safety 217, 107961. doi:https://doi.org/10.1016/j.ress.2021.107961.

Bakarji, J., Tartakovsky, D.M., 2021. Data-driven discovery of coarse-grained equations. Journal of Computational Physics 434, 110219. doi:https://doi.org/10.1016/j.jcp.2021.110219.

Belbute-Peres, F.d.A., Smith, K.A., Allen, K.R., Tenenbaum, J.B., Kolter, J.Z., 2018. End-to-end differentiable physics for learning and control, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA. p. 7178–7189.

Bhattacharyay, D., Kocaefe, D., Kocaefe, Y., Morais, B., 2017. An artificial neural network model for predicting the CO2 reactivity of carbon anodes used in the primary aluminum production. Neural Computing and Applications 28, 553–563. doi:10.1007/s00521-015-2093-7.

Bishop, C.M., 1995. Regularization and complexity control in feed-forward networks, in: International Conference on Artificial Neural Networks, pp. 141–148. URL: https://publications.aston.ac.uk/id/eprint/524/.

Blakseth, S.S., Rasheed, A., Kvamsdal, T., San, O., 2022a. Combining physics-based and data-driven techniques for reliable hybrid analysis and modeling using the corrective source term approach. Applied Soft Computing 128, 109533. doi:10.1016/j.asoc.2022.109533.

Blakseth, S.S., Rasheed, A., Kvamsdal, T., San, O., 2022b. Deep Neural Network Enabled Corrective Source Term Approach to Hybrid Analysis and Modeling. Neural Netw. 146, 181–199. doi:10.1016/j.neunet.2021.11.021.

Boche, H., Calderbank, R., Kutyniok, G., Vybíral, J. (Eds.), 2015. Compressed Sensing and Its Applications: MATHEON Workshop 2013. Applied and Numerical Harmonic Analysis, Springer International Publishing, Cham. doi:10.1007/978-3-319-16042-9.

Boussaïd, I., Siarry, P., Ahmed-Nacer, M., 2017. A survey on search-based model-driven engineering. Automated Software Engineering 24, 233–294. URL: https://doi.org/10.1007/s10515-017-0215-4, doi:10.1007/s10515-017-0215-4.

Bradley, W., Kim, J., Kilwein, Z., Blakely, L., Eydenberg, M., Jalvin, J., Laird, C., Boukouvala, F., 2022. Perspectives on the integration between first-principles and data-driven modeling. Computers & Chemical Engineering , 107898doi:https://doi.org/10.1016/j.compchemeng.2022.107898.

Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D., 2020. Language Models are Few-Shot Learners. doi:10.48550/arXiv.2005.14165.

Brunton, S.L., Proctor, J.L., Kutz, J.N., 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences 113, 3932–3937. doi:10.1073/pnas.1517384113, arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.1517384113.

Champion, K., Lusch, B., Kutz, J.N., Brunton, S.L., 2019. Data-driven discovery of coordinates and governing equations. Proceedings of the National Academy of Sciences 116, 22445–22451. doi:10.1073/pnas.1906995116, arXiv:https://www.pnas.org/content/116/45/22445.full.pdf.

Chermont, P.R.S., Soares, F.M., de Oliveira, R.C.L., 2016. Simulations on the Bath Chemistry Variables Using Neural Networks. Springer International Publishing, Cham. pp. 607–612. URL: https://doi.org/10.1007/978-3-319-48251-4_102, doi:10.1007/978-3-319-48251-4_102.

Cheung, C.Y., Menictas, C., Bao, J., Skyllas-Kazacos, M., Welch, B.J., 2015. Spatial thermal condition in aluminum reduction cells under influences of electrolyte flow. Chemical Engineering Research and Design 100, 1–14. doi:10.1016/j.cherd.2015.04.034.

Einarsrud, K.E., Eick, I., Bai, W., Feng, Y., Hua, J., Witt, P.J., 2017. Towards a coupled multi-scale, multi-physics simulation framework for aluminium electrolysis. Applied Mathematical Modelling 44, 3–24. doi:10.1016/j.apm.2016.11.011.

Frankle, J., Carbin, M., 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. doi:10.48550/arXiv.1803.03635, arXiv:arXiv:1803.03635.

Gale, S., Vestheim, S., Gravdahl, J.T., Fjerdingen, S., Schjølberg, I., 2013. RBF network pruning techniques for adaptive learning controllers, in: 9th International Workshop on Robot Motion and Control, pp. 246–251. doi:10.1109/RoMoCo.2013.6614616.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press. URL: http://www.deeplearningbook.org.

Gusberti, V., Severo, D.S., Welch, B.J., Skyllas-Kazacos, M., 2016. Modeling the Mass and Energy Balance of Different Aluminium Smelting Cell Technologies, in: Suarez, C.E. (Ed.), Light Metals 2012. Springer International Publishing, Cham, pp. 929–934. doi:10.1007/978-3-319-48179-1_161.

Hachicha, N., Jarboui, B., Siarry, P., 2011. A fuzzy logic control using a differential evolution algorithm aimed at modelling the financial market dynamics. Information Sciences 181, 79–91. URL: https://www.sciencedirect.com/science/article/pii/S002002551000438X, doi:10.1016/j.ins.2010.09.010.

Hedrea, E.L., Precup, R.E., Roman, R.C., Petriu, E.M., 2021. Tensor product-based model transformation approach to tower crane systems modeling. Asian Journal of Control 23, 1313–1323. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/asjc.2494, doi:10.1002/asjc.2494.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., Peste, A., 2021. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. Journal of Machine Learning Research 22, 1–124.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Bach, F., Blei, D. (Eds.), Proceedings of the 32nd International Conference on Machine Learning, PMLR, Lille, France. pp. 448–456. URL: https://proceedings.mlr.press/v37/ioffe15.html.

Johansen, S.T., Einarsrud, K.E., Solheim, A., Vachaparambil, K.J., 2022. A pragmatic model for alumina feeding, in: Light Metals 2022, Springer International Publishing, Cham. pp. 503–511.

Kim, S., Lu, P.Y., Mukherjee, S., Gilbert, M., Jing, L., Ceperic, V., Soljacic, M., 2021. Integration of neural network-based symbolic regression in deep learning for scientific discovery. IEEE Transactions on Neural Networks and Learning Systems 32, 4166–4177. doi:10.1109/TNNLS.2020.3017010.

Kingma, D.P., Ba, J., 2017. Adam: A Method for Stochastic Optimization. doi:10.48550/arXiv.1412.6980.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., Mahoney, M.W., 2021. Characterizing possible failure modes in physics-informed neural networks. Advances in Neural Information Processing Systems 34, 26548–26560.

Krogh, A., Hertz, J.A., 1991. A simple weight decay can improve generalization, in: Proceedings of the 4th International Conference on Neural Information Processing Systems, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. p. 950–957.

LeVeque, R.J., 2002. Finite Volume Methods for Hyperbolic Problems. Cambridge Texts in Applied Mathematics, Cambridge University Press, Cambridge. doi:10.1017/CBO9780511791253.

Ljung, L., 1998. System Identification: Theory for the User. 2nd ed., Pearson.

Lundby, E.T.B., Rasheed, A., Gravdahl, J.T., Halvorsen, I.J., 2021. A novel hybrid analysis and modeling approach applied to aluminum electrolysis process. Journal of Process Control 105, 62–77. doi:https://doi.org/10.1016/j.jprocont.2021.06.005.

Lundby, E.T.B., Rasheed, A., Gravdahl, J.T., Halvorsen, I.J., 2023. Sparse deep neural networks for modeling aluminum electrolysis dynamics. Applied Soft Computing 134, 109989. doi:10.1016/j.asoc.2023.109989.

Majid, N.A.A., Taylor, M.P., Chen, J.J.J., Young, B.R., 2011. Multivariate statistical monitoring of the aluminium smelting process. Journal of Computers & Chemical Engineering 35, 2457–2468. doi:10.1016/j.compchemeng.2011.03.001.

Meghlaoui, A., Thibault, J., Bui, R., Tikasz, L., Santerre, R., 1998. Neural networks for the identification of the aluminium electrolysis process. Computers & Chemical Engineering 22, 1419–1428. doi:https://doi.org/10.1016/S0098-1354(98)00223-3.

Natarajan, B.K., 1995. Sparse Approximate Solutions to Linear Systems. SIAM Journal on Computing 24, 227–234. doi:10.1137/S0097539792240406.

Nelles, O., 2020. Nonlinear system identification: from classical approaches to neural networks, fuzzy models, and gaussian processes. Springer Nature.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv:1912.01703 [cs, stat] URL: http://arxiv.org/abs/1912.01703. arXiv: 1912.01703.

Pawar, S., San, O., Aksoylu, B., Rasheed, A., Kvamsdal, T., 2021a. Physics guided machine learning using simplified theories. Physics of Fluids 33, 011701. URL: https://aip.scitation.org/doi/10.1063/5.0038929, doi:10.1063/5.0038929.

Pawar, S., San, O., Nair, A., Rasheed, A., Kvamsdal, T., 2021b. Model fusion with physics-guided machine learning: Projection-based reduced-order modeling. Physics of Fluids 33, 067123. URL: https://aip.scitation.org/doi/abs/10.1063/5.0053349, doi:10.1063/5.0053349.

Pineda, L., Fan, T., Monge, M., Venkataraman, S., Sodhi, P., Chen, R.T.Q., Ortiz, J., DeTone, D., Wang, A., Anderson, S., Dong, J., Amos, B., Mukadam, M., 2023. Theseus: A Library for Differentiable Nonlinear Optimization. doi:10.48550/arXiv.2207.09442, arXiv:arXiv:2207.09442.

Pozna, C., Precup, R.E., Tar, J.K., Škrjanc, I., Preitl, S., 2010. New results in modelling derived from Bayesian filtering. Knowledge-Based Systems 23, 182–194. URL: https://www.sciencedirect.com/science/article/pii/S0950705109001579, doi:10.1016/j.knosys.2009.11.015.

Rackauckas, C., Nie, Q., 2017. Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia. Journal of Open Research Software 5, 15.

Rai, R., Sahu, C.K., 2020. Driven by data or derived through physics? a review of hybrid physics guided machine learning techniques with cyber-physical system (cps) focus. IEEE Access 8, 71050–71073. doi:10.1109/ACCESS.2020.2987324.

Raissi, M., Perdikaris, P., Karniadakis, G.E., 2018. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics 378. URL: https://www.osti.gov/pages/biblio/1595805, doi:10.1016/j.jcp.2018.10.045.

Raviprakash, K., Huang, B., Prasad, V., 2022. A hybrid modelling approach to model process dynamics by the discovery of a system of partial differential equations. Computers & Chemical Engineering 164, 107862. doi:10.1016/j.compchemeng.2022.107862.

Robinson, H., Pawar, S., Rasheed, A., San, O., 2022. Physics guided neural networks for modelling of non-linear dynamics. Neural Networks 154, 333–345. doi:https://doi.org/10.1016/j.neunet.2022.07.023.

von Rueden, L., Mayer, S., Sifa, R., Bauckhage, C., Garcke, J., 2020. Combining machine learning and simulation to a hybrid modelling approach: Current and future directions, in: Berthold, M.R., Feelders, A., Krempl, G. (Eds.), Advances in Intelligent Data Analysis XVIII, Springer International Publishing, Cham. pp. 548–560.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C., 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4510–4520. doi:10.1109/CVPR.2018.00474.

Sjöberg, J., Ljung, L., 1995. Overtraining, regularization and searching for a minimum, with application to neural networks. International Journal of Control 62, 1391–1407. doi:10.1080/00207179508921605.

de Souza, A.M.F., Soares, F.M., de Castro, M.A.G., Nagem, N.F., Bitencourt, A.H.d.J., Affonso, C.d.M., de Oliveira, R.C.L., 2019. Soft Sensors in the Primary Aluminum Production Process Based on Neural Networks Using Clustering Methods. Sensors 19, 5255. doi:10.3390/s19235255.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15, 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

Takens, F., 1981. Detecting strange attractors in turbulence, in: Rand, D., Young, L.S. (Eds.), Dynamical Systems and Turbulence, Warwick 1980, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 366–381.

Udrescu, S.M., Tan, A., Feng, J., Neto, O., Wu, T., Tegmark, M., 2020. Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. Advances in Neural Information Processing Systems 33, 4860–4871. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/33a854e247155d590883b93bca53848a-Paper.pdf.

Vaddireddy, H., Rasheed, A., Staples, A.E., San, O., 2020. Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensors. Physics of Fluids, Editor's pick 32, 015113. doi:https://doi.org/10.1063/1.5136351.

von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Pfrommer, J., Pick, A., Ramamurthy, R., Walczak, M., Garcke, J., Bauckhage, C., Schuecker, J., 2023. Informed Machine Learning – A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems. IEEE Transactions on Knowledge and Data Engineering 35, 614–633. doi:10.1109/TKDE.2021.3079836.

Wilson, D.R., Martinez, T.R., 2003. The general inefficiency of batch training for gradient descent learning. Neural Networks 16, 1429–1451. doi:10.1016/S0893-6080(03)00138-2.

Winter, M., Breitsamter, C., 2018. Nonlinear identification via connected neural networks for unsteady aerodynamic analysis. Aerospace Science and Technology 77, 802–818. doi:10.1016/j.ast.2018.03.034.

Xu, H., Zhang, D., Wang, N., 2021. Deep-learning based discovery of partial differential equations in integral form from sparse and noisy data. Journal of Computational Physics 445, 110592. doi:https://doi.org/10.1016/j.jcp.2021.110592.

Zhong, J., Feng, L., Ong, Y.S., 2017. Gene Expression Programming: A Survey [Review Article]. IEEE Computational Intelligence Magazine 12, 54–72. doi:10.1109/MCI.2017.2708618.