ISBN: 978-628-95207-8-1. ISSN: 2414-6390. Digital Object Identifier: 10.18687/LACCEI2024.1.1.1795

Machine Learning Surrogate Dynamical System Model for Thermal Energy Storage

Erimar F. Diaz Sierra[1], HS, Felix M. Cruz De Jesus[2], HS, Jorge J. Jiménez José[3], HS, Andres J. Lebrón Santana[4], HS, Jeremy J. Nuñez Ríos[5], HS, Luis M. Traverso Aviles[6], PhD.
[1-6]Universidad Ana G. Méndez – Recinto de Gurabo, Puerto Rico, USA, ediaz341@email.uagm.edu, fcruz106@email.uagm.edu, jjimenez237@email.uagm.edu, alebron85@email.uagm.edu, jnunez146@email.uagm.edu, traversol1@uagm.edu

Abstract– ***A thermal energy storage (TES) can serve as a mean of minimizing energy losses when there is fluctuation of energy demand. A coupled fluid and conduction thermal model are performed to obtain the history of the temperature profile over 500 timesteps simulations. Results files are generated in text format and imported as numerical arrays in Python programing. These results are used to train a deep learning algorithm based on convolutional and dense layers. Two of these architectures are presented here. Under this architecture, results can match validation data for a certain number of cases with relatively low errors.***

Keywords—machine learning, surrogate model, dynamical system, thermal energy storage.

I. Introduction

TES systems can store thermal energy in the form of hot or cold mass to be used for heat transfer at a later time by maintain and controlling quality temperatures. They can help alleviate the thermal transients of power systems. When power availability is intermittent, energy can be stored in periods of abundance to later be consumed in periods of scarcity. TES are used to store solar heat using sensible or phase changing materials within dedicated devices [1] or implemented in civil infrastructure deigned for energy storage [2]. The rate at which energy is transferred during abundance and scarcity can impact the overall efficiency of a power system thus improving energy management and sustainability. TES offer a way to get returns on investments and it has been calculated that they can yield a payback period of between 0.61 to 1.13 years for latent and phase changing storages [3]. An exergy analysis assessment has demonstrated that energy savings can be optimized by properly selecting parameters such as size, material, and structure [4]. Like with many other numerical single physics calculations, TES models poses challenges that might affect the sensitivity of it's accuracy. The coupling of heat transfer modes such as conduction, convection, and radiation, introduce nonlinearities that might affect the results in volume boundaries. The complex transient nature of TES geometries benefits from improving its accuracy by increasing the number of nodes (building a finer mesh) and time steps. A proper surrogate model should take into consideration all these challenges.

Efficient and accurate simulations techniques can have great potential in real time applications, or simulators that could assist processes in industries such as power systems, aerospace, automotive engineering, ect. Testing incorporating hardware in the loop, and cyber-physical systems highlight the integration of real time simulators. The accuracy of a model unfortunately comes with a time constraint that impairs a model to be used in real time.

Surrogate models can significantly decrease the time it takes to produce an array of results in space and time for any given physics simulation.

This paper proposes to create a surrogate by training a machine learning (ML) algorithm with computer finite element method (FEM) and computational fluid dynamics (CFD) simulation of a TES. The proposed approach is for the model to function as a dynamical system based on a state and boundary condition input and evolved state as an output. The key contribution of this paper is to prove that the ML algorithm learns to predict the fundamental evolution of fully meshed physical state using only the state and boundary conditions of a given time step.

Traditional approaches to modeling TES performance includes zero-dimensional, quasi-one-dimensional and one-dimensional dynamic models [5] that must take into consideration the change of the state of the system with respect to time. Experimental data shows that one dimensional models are more accurate than zero and quasi dimensional [5]. During its early years before the proper computational threshold allowed it, the FEM method was already considered a state-of-the-art solution for relatively complex problems [6]. The FEM has widely been used for many problems including conduction in steady [7] and transient [8] heat transfer. Ever since its early years, CFD has been a precise tool for solving Navier Stokes equations, which contain non-linear convection effects [9], and it has been used for convective heat transfer modeling [10]. Combining FEM and CFD analysis results in conjugated models that incorporate both physics. These models have been accurately compared to experimental results[11]. Both methods increase their accuracy when the discretization space contains more nodes at the expense of increase computational times.

Surrogate models approximate and replace time-consuming models based on more traditional approaches. The selection of a proper model is based on the amount of information that needs to be computed, the required accuracy, and expected time of computation depending on the application. Framework for guidance on the fine tuning of these parameter have been presented to organize the design of surrogate models [12]. They can be used in dynamic models such as digital twins [13], and cyber-physical systems [14], [15]. Cyber-physical components have been incorporated in complex analysis that allows to control intelligently systems used to balance load distributions [16]. Surrogate models have the potential of allowing real time accurate modeling that could improve the performance of actuator controls. Surrogate models can advance the design optimization of new products increasing the exploration rate of the design variable space [17] which could impact positively in wide scale research and development of products.

Among the existing techniques to create surrogate models that have been used for thermal systems include kringing algorithm [18], k-nearest neighbor [19], and decision trees among [20] others. All the mentioned techniques were used to create surrogate model for case studies that included: heat sink design [18], prediction of building thermal loads [19], and battery pack thermal management among others [20]. The performance evaluations of these methods have been tested against experimental results, and against the prediction of classical approaches such as the response of state diagram controllers.

ML technics have without a doubt transformed most technologies in the planet. Ranging from consumers predictive behavior, object recognition, natural language processing, physics informed models and many other applications, they have served as the main method to capture patterns and behavior that are too complex to capture with simple equations. They have been used to assist TES optimization [21]. They have also been trained using density functional theories that allows to find materials with proper high heat capacity[22]. Recurrent neural networks have been used to predict thermal storage usage in terms of the time of the system and the demand of the consumer [23]. Even though they have been used to optimize a TES, the model themselves do not incorporate the entire state of the state in terms of its temperature profile [21],[22], [23] .

Integration of FEM and CFD into physics informed ML algorithm have the potential of achieving practical solutions to partial differential equations that govern different phenomena. This combined approach could yield benefits in the minimization of computational time while still maintaining the proper level of accuracy for conjugate problems. It also allows us to create dynamical systems model.

Dynamical systems govern the evolution of a state that could be mechanical, thermodynamical, electrical, quantical, or, chemical. All equations of physics could be considered dynamical systems. A state is capable of evolving using a partial differential equation as a rule that is a function of the state itself, its medium properties, and the environment surrounding an element or volume. TES and all thermodynamical transient systems are dynamical at the very core. Commercial software such as Ansys, COMSOL, Solid works, and many others gives us precise numerical representations of the state and evolution of a system. There is a growing and very important field of physics informed ML that aims to describe physics incorporating the collection of noisy data, roughness of frictional behavior and inexplicability of physical properties [24]. Within this area there are efforts to discover governing equations by sparce identification using ML that allows for a physical phenomenon to be describe with the fewest set of terms [25].

Even though there are many approaches to use ML to simulate a physical system and even create surrogate models, there are no attempts to construct a full dense mesh nodal state dynamical system model that allows to mimic an FEM and CFD conjugate analysis and create the proper surrogate model. Is it possible to construct a deep neural network that has a fully meshed state input and outputs the state of the same mesh at a later instant? Does the computational time of this model is compared to the time it takes to generate a training set made in FEM and CFD models? Answering this question can pave the way for creating full scale conjugate models that could be considered surrogates and takes a fraction of the computational time based solely on the state of the system and its boundary conditions. These models could be employed in cyber-physical systems and digital twins with great accuracy. Even though the learning presented in this paper is based on FEM and CFD, this approach could serve well using experimental observables retrieved from an array of sensors.

A methodology is presented in which the FEM and CFD models are described. Then a ML architecture is described with a description of parameters and hyper parameters. Results are described and the performance is compared in terms of nodal temperature error statistics and a brief description of the computational time comparison.

II. Methodology

FEM and CFD simulations were used as sources to gather the training data for our model. After gathering the data, deep learning libraries were used to create and train the model. FEM/CFD and ML procedures are presented in the next section.

A. FEM/CFD Simulations

The data used to train the ML algorithm was obtained from flow simulations with FEM and CFD, using the commercially available tool, SolidWorks, which is capable of coupling the two analyses. The most important data required for the algorithm is the history of the temperature profiles in the solid and fluid regions of the TES. First, we created the prototype of the TES in SolidWorks, for which some examples are presented in Figure 1. The initial conditions that we selected are the fluid inlet temperatures and mass flow rate, under fully developed conditions, the solid's material and its initial temperature. We also specified that the six walls are adiabatic, which is shown in Figure 1. Mesh and timesteps were defined. Different studies were carried out to ensure that we worked with the best model. We varied the number of flow pipes to check for symmetric temperature profiles, as well as to check for different charging and discharging capacities. Even though the primary goal of this study was not to optimize the TES, we wanted to have a better understanding of the response time of the system as well as the calculation time of the FEM and CFD coupled models. We applied heat transfer symmetry using adiabatic surfaces to simplify the model to one hole as shown in Figure 1. This model helps reduce simulation calculation time and storage space because the model has fewer nodes in the mesh. A mesh study was carried out to compare the results of different mesh qualities. Ultimately, we chose a mesh quality with a number of 6,359 nodes. Additionally, a time step study was carried out. The time step is the interval in time in which the model's equations are applied, and the state of the model highlighted. The longer the time step, the less accurate the results are, although longer time steps help reduce the simulation waiting time. Cases were simulated with time steps of 0.5 s, 1 s, and 5 s. A time step of 1 s was chosen because it considerably reduced the waiting time. After the selection of nodes number and time step, the final simulations were executed and the training data was collected.

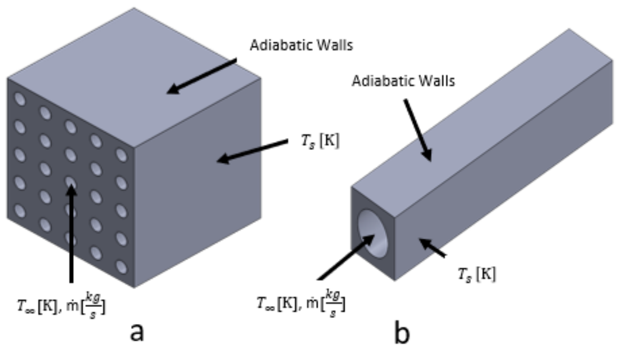
Adiabatic Walls
Adiabatic Walls
T_s [K]
T_s [K]
T_∞[K], $\dot{m}[\frac{kg}{s}]$
T_∞[K], $\dot{m}[\frac{kg}{s}]$
a
b
Fig. 1 Simulation models

Figure 1a shows the model with 25 flow pipes with its initial solid temperatures and boundary conditions. This model has more nodes; therefore, the calculation time is greater. Figure 1b shows the simplified model. Like the model in Figure 1a, it has 6 adiabatic walls, and the heat transfer is symmetrical. The simplified model has the same conditions as the 25 flow pipes model but since it has fewer nodes, the calculation time is shorter.

The FEM and CFD program solve a discrete version of the following equations .

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u}\cdot\nabla)\mathbf{u} = \nabla\cdot\left[\boldsymbol{\kappa} - p\mathbf{I}\right] + F \tag{1}$$

$$\boldsymbol{\kappa} = \mu\left[\nabla\mathbf{u} + (\nabla\mathbf{u})^{\mathrm{T}}\right] \tag{2}$$

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0 \tag{3}$$

Where, ρ is flow density, k is the thermal conductivity, **u** is fluid velocity vector, p is pressure, μ is dynamic viscosity of the fluid, F is volumetric force, ∇ is Laplacian of the velocity vector and t is the time. For the solid medium, the following equations are applied.

$$\rho c_p\frac{\partial T}{\partial t} + \rho c_p\mathbf{u}\cdot\nabla T + \nabla\cdot\mathbf{q} = Q \tag{6}$$

$$\mathbf{q} = -k\nabla T \tag{7}$$

Where, ρ is density, c_p is the specific heat capacity, T is temperature, t is the time, Q is rate of heat generation, ∇ is gradient operator and k is the thermal conductivity.

Once the simulation is finished, the temperature profile in the solid and fluid regions of the TES is exported in text format for each time step in individual files containing node coordinates and temperatures on each line of the file.

It is expected that the accuracy of the ML model will increase when using more training samples or cases in which boundary and initial conditions vary. A set of 66 time steps were used to train. The variations were made in the initial temperature of the TES (solid) (T_s) and the fluid inlet temperature (T_{in}). T_s varied from 293 K to 600 K and T_{in} varied from 293 K to 1000 K. Cases in which the solid temperature is higher than the fluid are "charging" and the ones in which the solid is hotter were categorized are discharging the TES.

B. Machine learning

Python and TensorFlow were selected for their resourceful library of deep learning specific commands. The study was divided into three parts: preparation, training, and validation. Each phase was developed to obtain minimal errors in our predicted data. For preparation, data was selected and exported from a FEM analysis software into a readable text file (.txt) that contained the coordinates and temperatures for each of the nodes in the TES. Each timestep data was reshaped into a one-dimensional array to which the boundary conditions corresponding to inlet temperature, and mass flow rate were appended. The mesh of the model is kept constant to ensure the same input shape for each sample. Each sample is labeled with an output consisting of a later time step temperature profile.

Sequential and non-sequential model architectures were considered. The architecture with a sequential path of layers has an input layer, followed by dense and convolutional layers, and ending with the output. For both models, the output has the shape corresponding to a one-dimensional flattened temperature state at a later point in time. The non-sequential architecture divides the data into three inputs, these being the node coordinates, the temperatures, and the boundary conditions. Both the coordinates and the temperatures were processed through distinct sets of convolutional layers and then concatenated into a single layer. The concatenated layer is then processed through a dense layer, which concatenates with the boundary conditions input layer. The loss function used was mean square error and the optimizer was the adaptive moment estimation method [26]. Figures 2 and 3 show the flowcharts for both architectures Units refer to the number of neurons in fully connected layers, while filter and kernel sizes refers to the number of one-dimensional convolutional filters and their respective size.

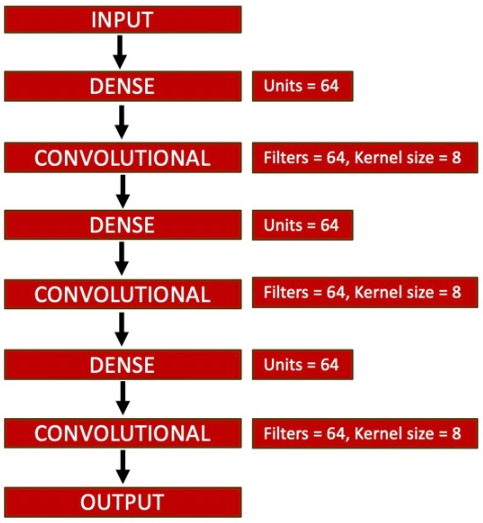

Figure 2. Sequential deep learning architecture

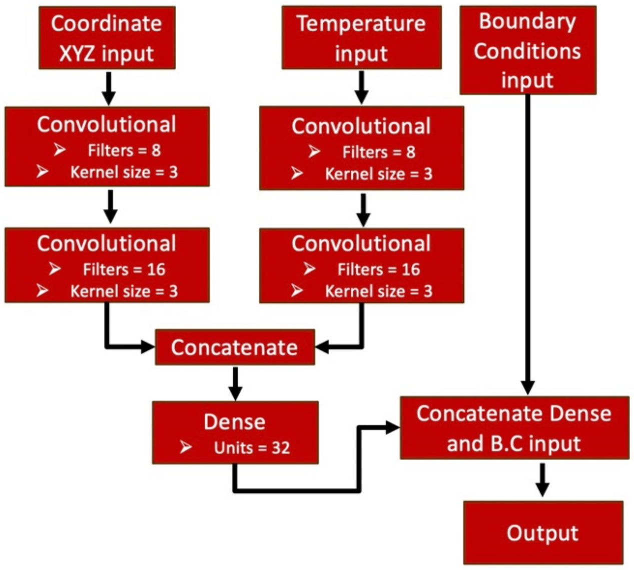

Figure 3. Non-sequential deep learning architecture

The training phase requires the tuning of hyper parameters that includes number of epochs, learning rate, and batch size. Weights are saved after each training and before new data is imported into the algorithm. When previous weights are loaded, the model trains with the new data and readjusts the previously learned parameters, calculating an updated set of weights. This phase was repeated for 24 different TES simulation cases. The training was validated with different samples consisting of temperature profiles at different times. The metric evaluation for the validation was based on mean square error. For nodal temperature comparison, error histograms were plotted to visualize the frequencies of different ranges of errors. We finalized this procedure and compared the accuracy that each model iteration produced in the predicted data, by observing the difference between their predictions and the real values of each temperature profile.

III. Results

The following shows the results of training 66 different cases. As new cases were used to train the model, their results were being saved and compared throughout the model's evolution. The analytical data used for this comparison included loss function graphs, root mean squared error (RMSE) graphs, and error histograms. The loss function graph records the change in the loss function over each epoch, showing how well the model trained. The RMSE is graphed VS each epoch and it shows how accurate the model can calculate a prediction. The error histograms plot the frequency for the nodal errors for different error ranges. Since the range of errors varied across each temperature profile, we calculated the average, mode, and maximum error that occurred for each case. These were our frames of reference throughout the training, indicating the model's performance.

A. Architecture

Both architectures produced different outcomes in the training of the model and its predictions. To compare them, we implemented model summaries that indicated the structure of the model, the shape of the input received in each layer, and the number of parameters that each layer contained. Salient factors about the performance of each architecture include the number of parameters calculated, the average training time required, and the average predicting time taken. Using the data of the simplified TES model, the sequential architecture had 4,872,556 parameters, required an average training time of 1.39 hours, and took an average prediction time of 0.62 seconds. Alternatively, using the non-sequential architecture resulted in lowering the parameters by approximately 31.35%, the training duration by 24.45%, and the prediction time by 20.96%. The calculated parameters were 1,527,792, while the average training and prediction times were 20.4 minutes and 0.13 seconds, respectively. Further testing proved that the non-sequential architecture was more accurate in its predictions by having calculation errors that ranged on a smaller scale than the sequential architecture errors. For these reasons, we chose the non-sequential architecture as the most optimal architecture to train our model.

B. Loss function and RMSE graphs

For most of the training, we noticed that the loss decreased as mode data was used for training. When the last case was used to train our model, the loss function graph reached a loss that ranged from 10^{-7} and 10^{-6} units, for both the training and validation data. The largest value recorded out of all the loss function graphs is 0.0425, which was recorded in the first training, while the lesser value recorded was 2.1382×10^{-8}. In the latest version of the model, the last values calculated by the loss function are 8.0864×10^{-7}, for the training data, and 4.6863×10^{-7} for the validation data. Moreover, throughout each case, the RMSE would also vary in a large range. In the last training of our model, the RMSE graph reached values that ranged from 10^{-4} and 10^{-3} units, which applied for both the training and validation data. The largest value recorded out of all the RMSE graphs was 0.2061 and the lesser valua was 1.4623 x 10^{-4}. These values were also recorded in the eleventh training of the model. During the last training, the last values calculated by the RMSE metric are 8.9925 x 10^{-4}, for the training data, and 6.8457 x 10^{-4} for the validation data. Figures 4 and 5 represent an example of how these graphs tend to behave for each training.

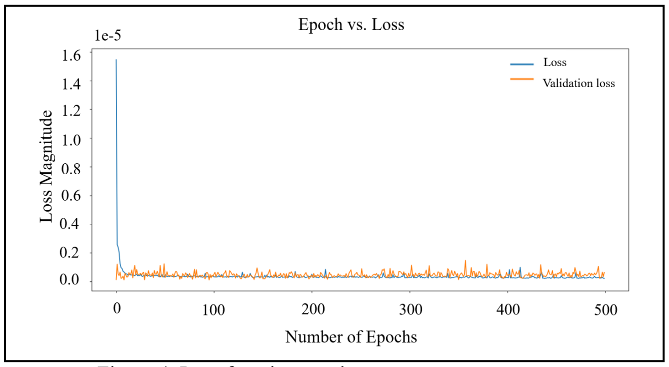

Figure 4. Loss function graph

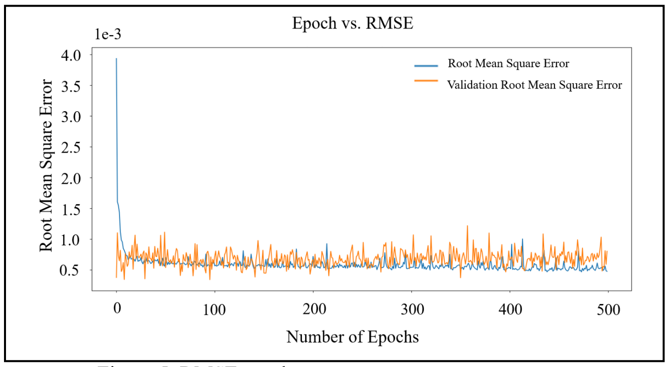

Figure 5. RMSE graph

C. *Errors*

The errors decreased and increased in range throughout the training. The first predicted temperature profile for each case contained the largest error. When the model predicted temperature profiles at later time steps of each case, the errors decreased. The prediction with the highest error was found in case 5, having a maximum error of approximately 74.88 K. In contrast, the prediction with the lowest error was found in case 23, having a maximum error of 1.42 K. The average error for these cases were 0.63 K and 0.05 K, respectively. Even though case 5 contained the highest maximum error in its prediction, compared to the rest of the data, case 11 had the highest average error out of all the cases, which was of 1.14 K. Parallel to its maximum error, case 23 also had the lowest average error. This makes case 23's predicted temperature profiles the most accurate data that the model has produced throughout its learning evolution. Figures 6 and 7 show the error statistics for each case, plotting the model's prediction evolution. A notable occurrence throughout the training is that when the model was presented with new trainable data containing different boundary conditions to those of the already trained datasets, the range of error increased. A possible reason for this occurrence is the model's inability to differentiate between cases that contain different boundary conditions. Nevertheless, further training resulted in better predictions. Another notable and expected detail was that the range of errors in the validation data was greater than the range of errors in the training data. However, the average errors remained on a small scale throughout the training of the model. This proves that the model can be rigorously trained to achieve a permissible range of error, for it to be considered accurate. Unfortunately, the model only predicts results with small errors using the data of the last case it was trained with. In its current state, the model predicts data with long ranges of error if it uses past datasets as its input data, even though it was trained with those datasets. These events should be investigated more thoroughly, to optimize our training process to its full capability. Future implementations that could better these results are the incorporation of different types of cases, and a provision of more variability in the boundary conditions. In conclusion, it has been determined that the model's current composition and architecture produces low average errors in its predicted data, and that the predictions improve with more training.

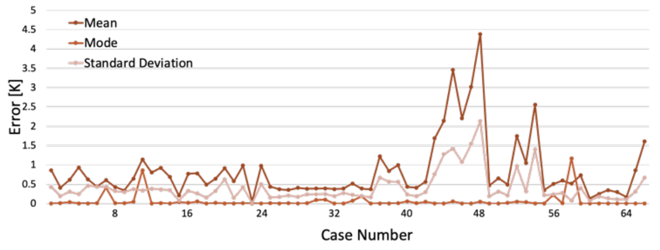

Figure 6. Error Statistics vs. Case Number

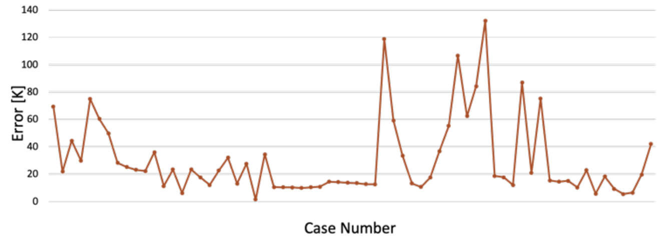

Figure 7. Maximum Errors vs. Case Number

IV Conclusion

ML can be used to capture patterns or behavior of simulated FEM models. In this case a thermal energy storage can be simulated and trained using FEM results for a transient analysis. This opens the possibility of using simulation results of other types of physics that apply similar numerical methods for deep neural networks. The time it takes to infer results are comparable to FEM solving times and can be used to avoid overhead times related to FEM analysis. For this study, we can conclude that non-sequential architecture yields result with lower error. Many architectures remained unexplored which brings many potential future studies that could help increase accuracy and decrease solving times. This is of course at the expense of requiring large sets of training data for proper performance, which is a common ML drawback. Another possibility for future improvement could include the generalization and application of different types of physics that could ultimately provide many other surrogate models to engineering analysis.

Acknowledgment

The authors of this paper would like to acknowledge the Consortium for Hybrid Resilient Energy Systems (CHRES) for their mentoring and contribution in this project.

References

[1] I. Sarbu and C. Sebarchievici, "A Comprehensive Review of Thermal Energy Storage," Sustainability, vol. 10, no. 1, 2018, doi: 10.3390/su10010191.

[2] S. Barbhuiya, B. B. Das, and M. Idrees, "Thermal energy storage in concrete: A comprehensive review on fundamentals, technology and sustainability," Journal of Building Engineering, vol. 82, p. 108302, Apr. 2024, doi: 10.1016/j.jobe.2023.108302.

[3] A. A. Godarzi, M. Jalilian, J. Samimi, A. Jokar, and M. A. Vesaghi, "Design of a PCM storage system for a solar absorption chiller based on exergoeconomic analysis and genetic algorithm," International Journal of Refrigeration, vol. 36, no. 1, pp. 88–101, Jan. 2013, doi: 10.1016/j.ijrefrig.2012.08.028.

[4] I. Dincer, "On thermal energy storage systems and applications in buildings," Energy and Buildings, vol. 34, no. 4, pp. 377–388, May 2002, doi: 10.1016/S0378-7788(01)00126-8.

[5] J. Raccanello, S. Rech, and A. Lazzaretto, "Simplified dynamic modeling of single-tank thermal energy storage systems," Energy, vol. 182, pp. 1154–1172, Sep. 2019, doi: 10.1016/j.energy.2019.06.088.

[6] R. W. Clough, "The finite element method after twenty-five years: A personal view," Computers & Structures, vol. 12, no. 4, pp. 361–370, Oct. 1980, doi: 10.1016/0045-7949(80)90113-3.

[7] G. R. Liu and S. S. Quek, "Chapter 12 - FEM for Heat Transfer Problems," in The Finite Element Method (Second Edition), G. R. Liu and S. S. Quek, Eds., Oxford: Butterworth-Heinemann, 2014, pp. 347–396. doi: 10.1016/B978-0-08-098356-1.00012-6.

[8] G. Warzee, "Finite element analysis of transient heat conduction application of the weighted residual process," Computer Methods in Applied Mechanics and Engineering, vol. 3, no. 2, pp. 255–268, Mar. 1974, doi: 10.1016/0045-7825(74)90028-0.

[9] F. H. Harlow and J. E. Welch, "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface," The Physics of Fluids, vol. 8, no. 12, pp. 2182–2189, Dec. 1965, doi: 10.1063/1.1761178.

[10]G. Maragkos and T. Beji, "Review of Convective Heat Transfer Modelling in CFD Simulations of Fire-Driven Flows," Applied Sciences, vol. 11, no. 11, 2021, doi: 10.3390/app11115240.

[11]P. Renze and K. Akermann, "Simulation of Conjugate Heat Transfer in Thermal Processes with Open Source CFD," ChemEngineering, vol. 3, no. 2, 2019, doi: 10.3390/chemengineering3020059.

[12]R. Alizadeh, J. K. Allen, and F. Mistree, "Managing computational complexity using surrogate models: a critical review," Research in Engineering Design, vol. 31, no. 3, pp. 275–298, Jul. 2020, doi: 10.1007/s00163-020-00336-7.

[13]S. Chakraborty, S. Adhikari, and R. Ganguli, "The role of surrogate models in the development of digital twins of dynamic systems," Applied Mathematical Modelling, vol. 90, pp. 662–681, Feb. 2021, doi: 10.1016/j.apm.2020.09.037.

[14]H. B. Adeyemo, R. Bahsoon, and P. Tiño, "Surrogate-based Digital Twin for Predictive Fault Modelling and Testing of Cyber Physical Systems," in 2022 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT), Dec. 2022, pp. 166–169. doi: 10.1109/BDCAT56447.2022.00028.

[15]Á. Bárkányi, T. Chován, S. Németh, and J. Abonyi, "Modelling for Digital Twins—Potential Role of Surrogate Models," Processes, vol. 9, no. 3, 2021, doi: 10.3390/pr9030476.

[16]D. Tucker, P. Pezzini, and K. Bryden, Cyber-Physical Systems: A New Paradigm for Energy Technology Development. 2018. doi: 10.1115/POWER2018-7315.

[17]P. S. Palar, R. P. Liem, L. R. Zuhal, and K. Shimoyama, "On the use of surrogate models in engineering design optimization and exploration: the key issues," in Proceedings of the Genetic and Evolutionary Computation Conference Companion, in GECCO '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1592–1602. doi: 10.1145/3319619.3326813.

[18]G. Qiao, W. Cao, Y. Hu, J. Li, L. Sun, and C. Hu, "Surrogate Model-Based Heat Sink Design for Energy Storage Converters," Energies, vol. 16, no. 3, 2023, doi: 10.3390/en16031075.

[19]Y. Liang, Y. Pan, X. Yuan, W. Jia, and Z. Huang, "Surrogate modeling for long-term and high-resolution prediction of building thermal load with a metric-optimized KNN algorithm," Energy and Built Environment, vol. 4, no. 6, pp. 709–724, Dec. 2023, doi: 10.1016/j.enbenv.2022.06.008.

[20]M. Arrinda, G. Vertiz, D. Sanchéz, A. Makibar, and H. Macicior, "Surrogate Model of the Optimum Global Battery Pack Thermal Management System Control," Energies, vol. 15, no. 5, 2022, doi: 10.3390/en15051695.

[21]W. Jin et al., "Machine-learning-assisted high-temperature reservoir thermal energy storage optimization," Renewable Energy, vol. 197, pp. 384–397, Sep. 2022, doi: 10.1016/j.renene.2022.07.118.

[22]J. Ojih, U. Onyekpe, A. Rodriguez, J. Hu, C. Peng, and M. Hu, "Machine Learning Accelerated Discovery of Promising Thermal Energy Storage Materials with High Heat Capacity," ACS Appl. Mater. Interfaces, vol. 14, no. 38, pp. 43277–43289, Sep. 2022, doi: 10.1021/acsami.2c11350.

[23]W. Zi-hao, W. Jing, Z. Ling, and J. Shu-juan, "A Thermal Energy Usage Prediction Method for Electric Thermal Storage Heaters Based on Deep Learning," in 2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Apr. 2019, pp. 149–154. doi: 10.1109/ICCCBDA.2019.8725757.

[24]G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," Nature Reviews Physics, vol. 3, no. 6, pp. 422–440, Jun. 2021, doi: 10.1038/s42254-021-00314-5.

[25]S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," Proceedings of the National Academy of Sciences, vol. 113, no. 15, pp. 3932–3937, Apr. 2016, doi: 10.1073/pnas.1517384113.

[26]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.