Explainable Neural Dynamics Models for Electric Motor Temperature Estimation

Xinyuan Liao¹, Shaowei Chen¹, Yunxiang Long¹, and Shuai Zhao²

¹Affiliation not available
²AAU Energy

March 30, 2024

Abstract

Accurate temperature estimation of permanent magnet synchronous motors serves as the fundamental basis for designing effective thermal management strategies. Model-based estimation methods exhibit superior real-time performance, but the intricate modeling process requires substantial expert knowledge and lacks versatility. Conversely, data-driven estimation methods, while offering flexibility, often lack physical implications in terms of system dynamics. This paper proposed a structured linear neural dynamics model for motor temperature estimation. This model is data-driven, with prior knowledge integrated into its structure, which preserves flexibility while guaranteeing system stability through the Perron-Frobenius theorem. Additionally, this paper achieves the decoupling of control input from state transitions and the embedded deployment of this model. The method is validated with a real dataset. The lightweight feature is demonstrated by the implementation of an STM32 Microcontroller with 1.808 KB and 27 mW. The paper is accompanied by the open source data and code at GitHub: https://github.com/ms140429/Explainable-Neural-Dynamics-Model
Abstract—Accurate temperature estimation of permanent magnet synchronous motors serves as the fundamental basis for designing effective thermal management strategies. Model-based estimation methods exhibit superior real-time performance, but the intricate modeling process requires substantial expert knowledge and lacks versatility. Conversely, data-driven estimation methods, while offering flexibility, often lack physical implications in terms of system dynamics. This paper proposed a structured linear neural dynamics model for motor temperature estimation. This model is data-driven, with prior knowledge integrated into its structure, which preserves flexibility while guaranteeing system stability through the Perron-Frobenius theorem. Additionally, this paper achieves the decoupling of control input from state transitions and the embedded deployment of this model. The method is validated with a real dataset. The lightweight feature is demonstrated by the implementation of an STM32 Microcontroller with 1.808 KB and 27 mW.

Index Terms—System Thermal Dynamics, Control-Oriented Modeling, State-Space Models, Neural Dynamics Models, Stability Constraints, STM32 Microcontroller.

I. INTRODUCTION

PERMANENT magnet synchronous motors (PMSMs) are extensively employed in a wide range of electric vehicles (EVs), hybrid electric vehicles (HEVs), etc [1]–[3] owing to their outstanding performance characteristics. In contrast to other competing products, PMSMs offer superior power efficiency, a simpler structure, and enhanced reliability [4]. Nevertheless, thermal management remains one of the primary factors restricting its power density. Furthermore, given that the thermal management system is key for the performance of many sub-systems (e.g., battery, motor, power electronics) in e-mobility, deploying efficient and precise temperature estimation can notably augment the driving range of EVs [5]. The stator winding insulation and permanent magnets are notably sensitive to temperatures. In cases of overload operation, the insulation of the stator winding may be at risk of melting, while the permanent magnets are prone to irreversible demagnetization. Therefore, effective thermal management serves as a crucial assurance for ensuring the long-term reliability of EVs as well. Model predictive control (MPC) is a popular technology in thermal management system design [6], and its main bottleneck lies in developing the motor thermal dynamics model. Developing a successful control-oriented thermal dynamics model necessitates achieving a delicate balance between estimation accuracy and real-time performance, a task that continues to pose significant challenges and has garnered considerable attention [7]–[9].

The motor thermal dynamics modeling methods can be broadly categorized into three groups: model-based, physics-informed, and data-driven. In general, as shown in Fig. 1, generalizability and model size will increase with reduced physics. At the same time, the interpretability of the model will be enhanced with more physics. Classical model-based methods [8], such as computational fluid dynamics (CFD) and finite element analysis (FEA), are challenging to apply due to their intricate modeling processes. The primary advantage of model-based methods lies in lightweight computing burden, making them well-suited for deployment on low-cost embedded hardware.

Data-driven methods based on AI do not require expert knowledge, which is its main advantage. It can model a wide range of systems if a substantial amount of measurement data is available for training. However, purely data-driven models, such as recurrent neural networks (RNNs) [10], convolutional neural networks (CNNs) [10], and temporal convolutional networks (TCNs) [11] often exhibit larger size, which may be less beneficial for real-time tasks. This limitation also presents challenges for the practical deployment of the model on microcontrollers. Furthermore, when employing data-driven methods to model complex systems, the inherent lack of interpretability limits the capability to offer stability guarantees or other trustworthy measures to the system, which is a critical concern in industrial applications.
Physics-informed methods leverage the strengths of both model-based and data-driven methods, and such a research paradigm is continuously expanding. The embedding of physics is divided into inductive bias embedding [12] and structure embedding [13]. Among them, methods with inductive bias [14], [15] regularize the surrogate model by incorporating physical regularization in the loss function, while methods with physical structure usually directly embed the physical model into the network structure. Obviously, physical structure embedding will provide more advantages in interpretability, and the subsequent physics-informed method refers to the method with physical structure. Lumped-parameter thermal network (LPTN) [13] is one of the exemplary cases of physics-informed methods, which integrated the physical models of the heat transfer process and empirical data-based model identification. LPTNs are informed by heat conduction formulas, enabling them to achieve high estimation accuracy while maintaining a smaller model size. Recently, the deep state-space model [16], [17] as a more concise physics-informed method achieved greater success in temporal modeling. The state-space model (SSM) dissects system behavior into state transitions, control inputs, and model outputs, which gives the capability to investigate system behavior from the underlying dynamics. Kirchgässner et al. [18] combined the SSM and heat conduction formula to achieve high estimation accuracy with fewer parameters. Ján et al. [19] utilized the property of neural networks for universal approximation to further simplify the process of state transition and control input. To summarize, physics-informed methods tend to provide higher accuracy than model-based methods, and they offer better interpretability and fewer parameters compared to purely data-driven methods.

It is challenging to apply data-driven methods to mission-critical industrial applications due to their limitations in real-time tasks and the stability guarantee lack. However, its generalizability has the advantage of reducing modeling difficulty and cost. Therefore, this paper proposed an explainable neural dynamics model, a physics-informed method combining data-driven methods and SSMs. According to the requirements for eXplainable Artificial Intelligence (XAI) [20], the pfNDM, based on SSM extensively applied in engineering, demonstrates the advantages in interpretability, explainability, and transparency. This explainable neural dynamics model retains the generalizability of data-driven methods and achieves superior performance with a compact model size. Furthermore, benefiting from the physical structure of the model, the Perron-Frobenius theorem can be utilized to guarantee the model’s stability. The new contributions are summarized as follows.

1) An explainable neural dynamics model is proposed for the PMSM temperature estimation, achieving state-of-the-art estimation accuracy with a relatively small model size.
2) From the perspective of dynamic systems, the Perron-Frobenius theorem is applied to constrain the spectral radius of the state matrix, ensuring the stability of the system model in a principled way.
3) The explainable neural dynamics model is lightweight and has been deployed on an STM32 microcontroller with a footprint of 1.808 KB and 27 mW. The software code and hardware implementation details accompanying the paper are open-sourced on GitHub1.

The remainder of this paper is structured as follows. Section II mainly introduces the methodology and the structure of the proposed framework. Section III details the dataset, data preprocessing, and hyperparameter selection, and provides an in-depth analysis of the temperature estimation performance, the eigenvalue distribution of the state matrix, and the decoupling characteristics. Section IV presents the process and results of model implementation on an STM32 microcontroller. Section V concludes the paper.

II. METHODOLOGY

To monitor a complex system’s condition, we often hope to obtain a series of system outputs through a series of system inputs. The SSM is a system modeling method that provides a concise form to reveal the relationship between the system’s internal state and external input and output, which have been widely applied in industry. For a discrete time-invariant system, its SSM can be formulated as

\[ x_t = Ax_{t-1} + Bu_t, \]
\[ y_t = Cx_t + Du_t, \]

where \( A \) is state matrix, \( B \) is input matrix, \( C \) is output matrix, and \( D \) is feedforward matrix. \( D \) represents the residual connection from input to output, which is often assumed as 0 for computational simplicity.

Due to the complexity of industrial systems, basic linear SSMs cannot accurately model complex cases. Embedding nonlinear modules into the SSM can enhance its expressive ability to achieve the goal of accurately modeling complex systems.

A. Structured Linear Neural Dynamics Model

The dynamic model modeled by the neural network can be referred to as the neural dynamics model [19], where the black-box form is expressed as

\[ x_0 = f_0 ([y_{-n}, \ldots, y_0]), \]
\[ x_t = f_x (x_{t-1}, u_t), \]
\[ y_t = f_y (x_t), \]

where \( f_0, f_x, \) and \( f_y \) are all nonlinear functions that can be facilitated by using neural networks. It is worth mentioning that this black-box representation is similar to the update formula of recurrent neural networks (RNNs), except for the \( f_0 \) that approximates the initial state.

Recently, some studies [16], [21], [22] have suggested that it is more advantageous to adopt a linear and additive update strategy between time steps. Therefore, structured neural dynamics models based on SSMs are introduced into the field.

of dynamical system modeling. Specifically, Eq. (3) can be extended to its gray box form [19] as
\[ x_t = f_x(x_{t-1}) + f_u(u_t), \]  
where \( f_u \) maps the control input into the state space. In order to make the structure of the gray-box model more transparent, we integrate the unique features of SSMs into the neural dynamics model. To decouple the control input from the state transition, \( f_u \) is set as a linear map. Moreover, to better analyze and constrain the stability of the dynamics model, \( f_x \) is set as a linear map as well. The structured linear neural dynamics model can be updated as
\[ x_0 = f_0([y_1, \ldots, y_0]), \]  
\[ x_t = Ax_{t-1} + f_u(u_t), \]  
\[ y_t = Cx_t, \]
where \( A \) is the state matrix, \( C \) is the output matrix, and \( f_u \) is the nonlinear form of the input matrix \( B \).

### B. Stability Guarantee

It is worth mentioning that the linear update strategy without the activation function of the structured neural dynamics model naturally has the potential for parallel computing. Specifically,
\[ x_1 = Ax_0 + f_u(u_1), \]  
\[ x_2 = A(Ax_0 + f_u(u_1)) + f_u(u_2), \]  
\[ = A^2x_0 + (Af_u(u_1) + f_u(u_2)), \]  
\[ x_k = A^kx_0 + \sum_{j=1}^{k} (A^{j-1}f_u(u_{k-j})), \]  
where \( S \) is the set of eigenvalues \( \lambda \in \mathbb{C}_n \) of the state matrix \( A \) as the source of instability in the data-driven approach. The spectral radius of a matrix is a measure of how large its elements are, and since \( \rho(A^k) \) is the lower bound of \( \|A^k\| \), the spectral radius is also a measure of the matrix size. A sufficient condition [16] to ensure the stability of the structured linear neural dynamics model is \( \rho(A) < 1 \). In addition, the eigenvalues of the state matrix \( A \) generally represent the importance of previous states in RNNs and are loosely related to the overall heat transfer coefficient of the system in thermodynamics [19].

**Perron-Frobenius theorem:** For an arbitrary matrix \( M \in \mathbb{R}^{n \times n} \), then
\[ \min_{1 \leq i \leq n} r_i \leq \rho(M) \leq \max_{1 \leq i \leq n} r_i, \]
where \( r_i \) is the summation of the \( i \)-th row of \( M \) and \( \rho(M) \) is the spectral radius of \( M \).

Guided by the Perron-Frobenius theorem, a new state matrix \( A' \) with a constrained spectral radius can be constructed by introducing a scaling matrix \( S \) to scale the row sum of the original state matrix \( A \). The constrained state matrix \( A' \) can be constructed as
\[ S = \lambda_{\text{max}} - (\lambda_{\text{max}} - \lambda_{\text{min}})\sigma(S'), \]  
\[ A' := \text{softmax}(A, \text{dim} = 1) \odot S, \]
where \( S' \in \mathbb{R}^{n \times n} \) is a trainable matrix, \( \sigma \) is the nonlinear activation function \( \text{sigmoid} \), and \( \odot \) is the Hadamard product.

### C. Physics-Informed Loss Function

Incorporating prior physical knowledge can enhance data-driven performance. It is noted that the hidden state of a dynamics system temporal changes smoothly, implying that the difference in hidden states between consecutive moments is relatively small [19]. Building upon the aforementioned prior knowledge, the proposed method incorporates the difference
in states between consecutive time steps into the loss function. This new loss function can be expressed as

\[
\text{Loss} = \text{Loss}_{\text{inf}} + Q \cdot \text{Loss}_{\text{smth}}
\]

\[
= \text{Loss}(\hat{y}, y) + Q \cdot \frac{1}{n} \sum_{t=1}^{n} \text{Loss}(x_t, x_{t-1}),
\]

where \( \hat{y} \) is the estimated value, \( y \) is the real value, \( x_t \) is the hidden state of the system at time \( t \), \( x_{t-1} \) is the hidden state of the system at time \( t - 1 \), and \( Q \) is the weight coefficient of the smooth loss term.

D. Perron-Frobenius Neural Dynamics Model

Following the imposition of stability constraints, the resulting structured linear neural dynamics model is referred to as the Perron-Frobenius Neural Dynamics Model (pNDM). Fig. 2 provides details of the pNDM introduced in this paper. Note that this paper refers to the structured linear neural dynamics model corresponding to Eq. (7) as the Neural Dynamics Model (NDM). The difference between the pNDM and NDM lies in the state matrix. The state matrix of the NDM is an unconstrained matrix, while the state matrix of the pNDM is a stable matrix constrained by the Perron-Frobenius theorem.

III. EXPERIMENT VALIDATION

A. Dataset Introduction

The pNMD proposed in this paper is evaluated on an open-sourced electric motor temperature dataset [11]. The dataset includes sensor data obtained from a permanent magnet synchronous motor (PMSM) by Paderborn University. The dataset comprises 185 hours of multi-sensor time series data, sampled at a frequency of 2 Hz. Please refer to [11] for detailed test bench parameters. The data of each independent test are distinguished by profile_id. For comparison purposes, this paper uses data from profile_id=58 as the validation set and data from profile_ids 65 and 72 as the test set, according to [11]. Table I shows the details of the input data and target data of the dataset, where the derived inputs \( u_s = \sqrt{u_d^2 + u_q^2} \) and \( i_s = \sqrt{i_d^2 + i_q^2} \) are features constructed from the measured inputs.

B. Data Preprocessing

1) Downsampling: When the sampling frequency is excessively high, it leads to only slight variations between the continuous sensor data, resulting in an abundance of redundant information. This redundancy can significantly strain computational resources, making downsampling a resource-efficient choice. To strike a balance between computational resources and information retention, this paper applied a downsampling rate of 8:1, i.e., merging 8 sets of continuous sensor data into one. Specifically, we take the average of the data within the downsampling window as the newly generated data.

2) Normalization: Training the model with normalized data leads to faster convergence. This paper employs two distinct normalization methods, one for temperature data and the other for other sensor data. To facilitate the scaling of estimation results back to their original scale, all temperature data is divided by 100. For other sensor data, we divide it by the maximum value of the absolute value of the data by using

\[
\hat{d}_i = d_i / \max(\text{abs}(d)),
\]

where \( \hat{d}_i \) is the normalized sensor data.

C. Hyperparameter Selection

The training process adopted a strategy of an early stopping scheme and the decay of the learning rate. The initial learning rate is set to 1e-3, and the learning rate is divided by 5 if the loss of the validation set does not decrease after 100 epochs. Stop training when the validation loss does not decrease for the fourth time. In this paper, Smooth \( L_1 \) Loss [23] is selected as the loss function during the training process, defined as

\[
\text{smooth}_{L_1}(x) = \begin{cases} 
0.5x^2, & \text{if } |x| < 1.
\end{cases}
\]

In comparison to the \( L_1 \) Loss and \( L_2 \) Loss, Smooth \( L_1 \) Loss exhibits robustness to outliers, and derivable at the coordinate origin is applicable across various scenarios. For the sake of enhancing the robustness of the training process, this paper utilizes the Adan [24] optimizer.

The prediction length is the time length of the data input to \( f_0 \), and the estimation length is the time length for temperature estimation. The selection of the prediction length and estimation length is consistent with [11]. The selection principle of the loss coefficient \( Q \) is to ensure that the smooth loss term is smaller than but close to the inference loss so that the smooth loss plays a regularizing role without disturbing the main training path of the model, e.g., if \( \text{Loss}_{\text{inf}} > \text{Loss}_{\text{smth}} \) then set \( Q > 1 \), if \( \text{Loss}_{\text{inf}} < \text{Loss}_{\text{smth}} \) then set \( Q < 1 \), and specific value of \( Q \) is optimized by grid search methodology. The constraint method for the state matrix is a hard constraint method, i.e., the spectral radius of the constrained matrix must be within the range of \([\lambda_{\text{min}}, \lambda_{\text{max}}]\). Therefore, the
The selection principle of the $\lambda_{\text{min}}$ and $\lambda_{\text{max}}$ is to give the model a wider parameter space for optimization while ensuring stability. In this context, $[\lambda_{\text{min}}, \lambda_{\text{max}}]$ is set as $[0.1, 0.999]$. More investigations on the selection of $\lambda_{\text{min}}$ on the pfNDM can be found in Section III-D. The size of the hidden state is the dimension of the state matrix, the principle of selecting the hidden state size is to select the most suitable value for the data set through grid search. In this paper, the hidden state size is set as 48, and the effect of the hidden state size on estimation accuracy will be discussed in Section III-D.

The $f_0$ and $f_u$ are modules that can be flexibly structured, and different network structures can be selected according to different tasks. In this paper, a one-dimensional convolutional neural network is used as $f_0$ to approximate the initial state, to facilitate the processing of multivariate time series data. Average pooling layers are embedded between the convolutional layers. After the data are flattened, the initial state is obtained through a fully connected layer. $f_u$ is a feed-forward neural network that nonlinearly maps the control input, and GELU [25] is utilized as the activation function. The other hyperparameters are given in Table II, which are optimized by using the grid search method.

**D. Performance Analysis**

The model is trained using time series data, where the length of each training data is the prediction length plus the estimation length. The temperature measurement values at the first prediction length moments are used to approximate the initial state, and the control inputs at the last estimation length moments are used to update the hidden state. During testing, a sliding-window approach is used to estimate the temperature throughout the monitoring period. The sliding window moves a distance equal to the estimation length with each step. Fig. 3 shows the temperature estimation by the pfNDM during the entire monitoring period for the test set with $\text{profile\_id}=65$. Based on the estimation results, it is evident that the pfNDM exhibits a high level of accuracy in its estimations.

1) Impact of Hidden state Size: The hidden state is a high-dimensional abstraction of the internal state of the system. Increasing the size of the hidden state can improve the representation ability of the model and can also enhance its flexibility. However, larger hidden states also place higher demands on training data, as there is an increased risk of overfitting if these requirements are not met. Fig. 4 illustrates the trend of model estimation performance as the size of the hidden state increases. It can be seen from the average value of model estimation accuracy under different hidden state sizes that both the pfNDM and NDM achieve the best performance when the hidden state size is 48. Continuing to increase the size of the hidden state will only increase the computational burden. Therefore, 48 was chosen as the value of the hidden state size. In addition, it can be found that the pfNDM that imposes constraints on the state matrix outperforms the unconstrained NDM in terms of estimation accuracy. The pfNDM has better performance under each size of the hidden state.

2) Impact of Range of Spectral Radius: The spectral radius of the state matrix is, in fact, a parameter that adheres to physical principles. In building thermodynamics, the spectral radius exhibits a loose correlation with the building’s overall heat transfer coefficient. However, the precise definition of the spectral radius range requires extensive expert knowledge. A general choice for the spectral radius of the state matrix is to use a large amount of data to learn the most appropriate state matrix based on a parameter space with a larger spectral radius range while ensuring stability. Fig. 5 serves to corroborate this perspective, the training performance in terms of different spectral radii is investigated. When a model is unconstrained, it has a faster convergence speed and a more evident fluctuation in loss values. When the spectral radius is confined to a
narrower interval, it results in reduced fluctuations in the loss value. Furthermore, when the spectral radius interval is set to [0.1, 0.999], the loss value exhibits a smooth convergence to its minimum point. However, when the spectral radius interval is narrowed to [0.5, 0.999], there is a slight plateau phase in the decline of the loss value, which slows down the convergence speed of the model, followed by convergence to the minimum point. In cases where the spectral radius interval is further constrained to [0.9, 0.999], the loss value ceases to decrease during the plateau period and eventually converges to a local minimum point. In summary, when a substantial reservoir of expert knowledge is not readily accessible, it is advisable to provide a broader optimization space for the spectral radius of the state matrix. Based on the above considerations, we finally set $[\lambda_{\min}, \lambda_{\max}]$ to [0.1, 0.999].

3) Comparisons with existing Methods: The method is validated with comparisons with state-of-the-art methods, focusing on estimation performance and model size. Table III shows the details of the comparison, in which both the NDM and pfNDM are superior in estimation performance. LPTN [13] and TNN [18], being models that incorporate more physics, holds an advantage in terms of model size. However, when compared to deep learning-based models like RNN [10] and TCN [11], the approach outlined in this paper boasts a notable advantage in terms of model size. Overall, the NDM and pfNDM achieve the best performance and have smaller model sizes than other deep learning-based models. Meanwhile, the proposed structured data-driven pipeline can be easily extended to other complex systems.

![Fig. 5. Training performance under different constraints.](image)

**TABLE III**

<table>
<thead>
<tr>
<th>Method</th>
<th>MSE ($K^2$)</th>
<th>Model size</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTN (Wallscheid et al., 2015) [13]</td>
<td>3.64</td>
<td>34</td>
</tr>
<tr>
<td>CNN (Kirchgässner et al., 2019) [10]</td>
<td>1.52</td>
<td>67k</td>
</tr>
<tr>
<td>RNN (Kirchgässner et al., 2019) [10]</td>
<td>3.02 (&gt;850k)</td>
<td></td>
</tr>
<tr>
<td>TCN (Kirchgässner et al., 2021) [11]</td>
<td>1.72</td>
<td>&gt;320k</td>
</tr>
<tr>
<td>ET (Kirchgässner et al., 2021) [7]</td>
<td>6.51</td>
<td>5.5M</td>
</tr>
<tr>
<td>TNN (Kirchgässner et al., 2023) [18]</td>
<td>2.87</td>
<td>1525</td>
</tr>
<tr>
<td>NDM (Proposed)</td>
<td>1.43</td>
<td>12.9k</td>
</tr>
<tr>
<td>pfNDM (Proposed)</td>
<td><strong>0.903</strong></td>
<td><strong>15.2k</strong></td>
</tr>
</tbody>
</table>

![Fig. 6. The eigenvalues of the state matrix on the complex plane. (a) eigenvalues of the state matrix of the NDM; and (b) eigenvalues of the state matrix of the pfNDM.](image)

**E. Eigenvalue Analysis**

This section compares and analyzes the distribution of eigenvalues of the state matrices of the pfNDM and NDM on the complex plane, and discusses the consistency of the pfNDM trained with varying random seeds.

Fig. 6 shows the details of the state matrix eigenvalues of the NDM and pfNDM. Due to the PMSM tested being a stable system, the spectral radius of the unconstrained NDM after training is less than 1 as well. However, it is crucial to recognize that there is no theoretical guarantee of the model stability, and the model could potentially exhibit instability, particularly when influenced by noise within the training data. Utilizing an unstable model to represent a stable system, apparently will cause unexpected consequences. In this paper, the Perron-Frobenius theorem is applied to ensure the stability of the pfNDM in a principled way. The relationship between eigenvalues and stability can be understood more intuitively from a geometric perspective. Since the eigenvectors of a diagonalizable matrix form a set of basis, and up to an arbitrarily small perturbation of the entries, any matrix is diagonalizable in the complex domain [22]. Any system state can be expressed as a linear combination of state matrix eigenvectors. Therefore, if the maximum absolute value of the eigenvalue is below 1, the state will persistently contract throughout the transition process until it converges to the attractor. This stability characterized by shrinkage also leads to greater robustness. In addition, it can be found that the
eigenvalues of pfNDM predominantly cluster around two concentration points 0 and 1. From a dynamics perspective, this signifies the presence of dynamic modes with varying levels of importance [26]. This observation opens up the possibility of achieving a low-dimensional representation for high-dimensional dynamics models [19]. Simultaneously, it is important to note that the imaginary part of the eigenvalue often corresponds to the periodic oscillation frequency of the system [19], [22]. Considering the physical context, the motor’s temperature is primarily influenced by operation conditions and the cooling system, so it typically does not exhibit a periodic fluctuation pattern. Hence, the smaller imaginary parts of the eigenvalues align more closely with the physical reality of the system.

Note that the transition matrix $A$ determines the underlying system state evolution. From an interpretability perspective, once the hidden state size is determined, the state matrices of models trained with varying random seeds should be similar. In other words, the behavior of state transitions should be primarily determined by the underlying system dynamics. To validate this hypothesis, this paper employs cosine similarity as the metric to quantify the similarity between state matrices for trained models with different random seeds.

The cosine similarity is defined as the measure of similarity between two non-zero vectors defined in an inner product space. Specifically, the cosine similarity of non-zero vectors $A$ and $B$ is defined as

$$S_C(A, B) := \frac{A \cdot B}{\|A\|\|B\|}.$$  

The physical interpretation of the cosine similarity corresponds to the cosine of the angle formed between two vectors. When the two vectors share the same direction, the cosine similarity value is equal to 1, while if they possess opposing directions, the cosine similarity value becomes -1. In this study, five distinct models were trained using five different random seeds. The mean cosine similarity of the state matrix’s eigenvalues vectors between these models was found to be 0.934, with a variance of 1.07e-3. As a result, it can be seen that the proposed pfNDM is able to characterize the state evolution deterministically.

Fig. 8 shows the determinism of the pfNDM in a more intuitive way. The eigenvalues of three distinct state matrices are depicted within the same complex plane. Observations reveal that the majority of these 144 eigenvalues are concentrated within two red ellipses, with only an eigenvalue lying outside these regions. This observation implies that the
different models have similar dynamic modes.

**F. Decoupling Analysis**

According to the distributive law of matrix multiplication, Eq. (8) can be decoupled as

\[ C(Ax_{t-1} + f_u(t_1)) = CAx_{t-1} + Cf_u(t_1). \]  

(14)

As a result, the structured linear neural dynamics model can decouple control inputs from state transitions. Here, \( CAx_{t-1} \) and \( Cf_u(t_1) \) are the temperatures that correspond to the hidden state after the transition and the control input, respectively. Fig. 7 visualized the temperatures corresponding to the hidden state and the control input during the entire monitoring period. It can be found that state transition plays a more pivotal role in model estimation, while the temperature associated with the control input remains relatively stable. This discovery contradicts the common understanding that control inputs predominantly dictate system evolution, even when considering that \( x_t \) contains information from previous control inputs. In the structured linear neural dynamics model, where state transitions dominate, the determined direction of state transitions becomes even more critical. This new perspective could potentially offer valuable inspiration for the field of data-driven control.

Although the decoupling characteristics of the NDM and pfNDM exhibit remarkable similarities, some distinctions can still be discerned. The temperature associated with the control input in the pfNDM is notably higher than that in the NDM, except for the permanent magnet, indicating that the disparity in hidden states before and after the transition is more evident in the pfNDM. This phenomenon may be attributed to the eigenvalues of the state matrix in the pfNDM having smaller imaginary parts, which require further investigation in the future. A smaller imaginary part would imply that the pfNDM places greater emphasis on the impact of the control input as opposed to its inherent periodic variations.

**IV. Deployment on Microcontroller**

The lightweight feature of the method is demonstrated by the embedded implementation of pfNDM. Due to its significantly smaller model size compared to other AI-based methods, pfNDM offers a lower deployment complexity.

STM32 is a microcontroller series launched by STMicroelectronics that is widely used in industrial control, smart home, automotive electronics, medical equipment, etc [27]. It has the characteristics of high performance, low power consumption, and strong reliability. Therefore, embedded deployment on STM32 is of great significance to industrial deployments. The model of the microcontroller utilized in the deployment is STM32H743IIT6, which is an ARM Cortex-M7 MCU with 480 MHz CPU, 2MB FLASH memory, and 1MB RAM. In addition, there is also a 512MB NAND FLASH and a 32MB SDRAM used.

Due to the complex structure of pfNDM, this paper split it into \( f_0, f_x, f_u, f_y \) to facilitate transplantation according to Section II. Note that since the CubeAI does not support the deployment of the ONNX model of one-dimensional convolutional neural networks directly, it needs to convert the \( f_0 \) under the PyTorch framework to a Keras model. Furthermore, due to parameter dimension inconsistencies between the PyTorch and Keras models, it is essential to introduce a transpose layer before the flatten layer when converting the PyTorch model to the Keras model. To facilitate deployment, \( f_0 \) is split into two components \( f_{01} \) and \( f_{02} \) to deployment, respectively. The \( f_{01} \) comprises two convolution layers with activation layers and pooling layers, while the \( f_{02} \) consists of a flatten layer and a fully connected layer. Finally, the logic of the entire

<table>
<thead>
<tr>
<th>Model</th>
<th>RAM</th>
<th>FLASH</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{01} )</td>
<td>0.416 KB</td>
<td>0.416 KB</td>
</tr>
<tr>
<td>( f_{02} )</td>
<td>0.416 KB</td>
<td>10.69 KB</td>
</tr>
<tr>
<td>( f_u )</td>
<td>0.384 KB</td>
<td>29.66 KB</td>
</tr>
<tr>
<td>( f_x )</td>
<td>0.384 KB</td>
<td>9.19 KB</td>
</tr>
<tr>
<td>( f_y )</td>
<td>0.208 KB</td>
<td>0.768 KB</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1.808 KB</strong></td>
<td><strong>50.724 KB</strong></td>
</tr>
</tbody>
</table>

Fig. 8. The structure and results demonstration of the condition monitor deployed on the STM32 microcontroller.
inference processing is implemented with C language. Fig. 9 shows the detailed process of embedding pNDM into the microcontroller. Besides, the memory usage of each converted model is shown in Table IV.

After completing the model deployment, a real-time condition monitor is implemented on the STM32, and the monitoring results are displayed on the LCD screen. Fig. 10 shows the hardware structure and result demonstration. The LCD screen outputs the temperature estimation values of the permanent magnets, stator yoke, stator teeth, and stator windings, displayed in chronological order. At the same time, to more intuitively verify the estimation accuracy of the deployed model, the LCD screen also outputs the actual temperature and estimation error (MSE) at each moment. Moreover, the power of algorithm execution on the microcontroller was measured as 27 mW. More details can be found in the accompanied illustrative video.

V. CONCLUSION

This paper proposes an explainable neural dynamics model for estimating PMSM temperature. By using a state-space framework, the model combines data-driven models with constraints on the underlying dynamics, achieving state-of-the-art estimation performance on the motor temperature dataset. We ensured the stability of the neural dynamics model by utilizing the Perron-Frobenius theorem to impose constraints on the state matrix. Furthermore, it is found the directions of state transition are determined by underlying dynamics, as observed from the cosine similarity of different state matrices’ eigenvalues. Leveraging the SSM decoupling property, this paper also offers a new perspective on the interplay between state transitions and system control inputs. This lightweight method has been successfully implemented on an STM32 microcontroller.

REFERENCES