Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2025 (43) 4** : 1–12.
ISSN-0037-8712
doi:10.5269/bspm.78566

Hyperparameter Optimization in Neural Networks Approach to Singular Matrix Differential Systems *

Praveen Kumar U M† and Venkata Sundaranand Putcha

ABSTRACT: Singular Matrix Differential Systems (SMDS) or, alternatively, semi-state, degenerate, descriptor, constrained, or differential-algebraic systems (DAEs) are key to modeling dynamic processes experiencing abrupt structural or parametric changes. It becomes difficult to solve initial value problems (IVPs) for these systems as classical methods are ineffective to apply owing to singularity and a lack of closed-form solutions. This paper introduces an adaptive neural network solution to linear singular matrix differential systems (LSMDS) with a semi-supervised learning framework. Singular systems, a commonplace within engineering models and constraint-laden physical models, are extremely computationally challenging with stiffness, index intricacy, and inconsistent initial conditions. Standard numerical solvers are afflicted by similar challenges, especially with singular matrices. To compensate, we propose a hybrid neural structure joining (i) a systematic search with activation functions and (ii) a two-stage optimizer sequence joining Adam's strengths with those of L-BFGS. The structure learns precise approximations without mesh-based discretizations. We build a task-specific loss function comprising differential-algebraic systems (DAEs) to guide optimizer training. We also undertake a detailed hyperparameter study, comprising network depth, width, activation function choosing, and optimizer switching plans, to establish suitable configurations. We evaluate our approach with a number of benchmark singular systems, achieving better accuracy, robustness, and generalization beyond standard solvers. This paper provides a flexible, data-efficient substitute to solving challenging constraint-laden systems, with significant applications to scientific computation and real-world modeling.

Key Words: Singular systems, differential equations, neural network, activation functions and optimization.

Contents

* The project is supported by the RUSA project letter No. RUSA-ANURU/Research Project-02/Sanction Order/2024 dt.01-06-2024

† Corresponding author.

2010 *Mathematics Subject Classification:* 34A09, 68T07, 65L08.

Submitted August 22, 2025. Published October 09, 2025

1 Introduction

The theory of differential equations provides a broad mathematical basis to understand the problems of arise in Social, Scientific, Biological, Engineering and Technological problems where dynamics of change are modeled. The main objective is to analyze the various observed evolution phenomena of nature by constructing suitable mathematical models. A large number of models have been developed to get an insight into the latent dynamics and complexities of the problems using dynamical systems in the form of either initial value problems or boundary value problems [27]. The differential equations of the type

$$E(t)X'(t) + B(X) = F(t) \tag{1.1}$$

Where $X(t)$ is a vector valued function of the real variable t, A is a real matrix, B is a vector valued function. When E is singular the above system is called a singular system. Singular systems are also called semi state, degenerate, descriptor, constrained, and differential algebraic systems. Sudden changes in the model parameters such as component failure or switching can be best described by Singular systems. Assume that switching occurs at $t = 0$ and for that $t > 0$ the physical system is modeled by the above equation. If $X(t)$ is the response of the system for $t < 0$ (not necessarily described by the above equation) and $X(t) \to X_0$ as $t \to 0$, then any $t < 0$ may be interpreted as an initial condition which, together with the system equation, determines the system behavior at the time of switching and for $t > 0$. Since no information is available regarding the system structure since there is no constraint on the initial value X_0. S.L Campbell [6], [5], [4], E L Yip and R F Sincovec [34] have solved certain class of singular systems according to the initial conditions [3]. S.L Campbell [6], G.C Verghese [32], P Van Dooren [13], [12] B Levy and T Kailath [32] have proposed certain distributions as solutions due to inconsistent solutions. In 1982 Danial Cobb [11] established the solutions of the singular systems by using singular perturbation theory. S.L Campbell [7] established the existence and uniqueness of a solution to the singular system imposing the regularity. Controllability and observability of Discrete and continuous systems were studied by [1], [2], [8], [9], [10] and analytical studies of various systems and their associated IVPs and BVPs were done by [25, 26, 28]. Many of the real world problems are singular in nature because of their ability to model unpredictable sudden changes. In most of the cases the sudden change scenarios are also modeled by approximated by nonsingular models since singular systems are difficult to handle and nonavailability of tools and theories to handle them. This really propelled many researchers [11] [13] [12] [15] [19] [20] [23] to analyze the structure of the linear singular systems by following algebraic and geometric approaches. In this paper we consider the Nonhomogeneous singular Matrix Differential systems(NHSMS)

$$E(t)X'(t) = A(t)X(t) + F(t) \tag{1.2}$$

where $A(t)$, $B(t)$ and $F(t)$ are square matrices of order n whose elements are real or complex functions defined on R (or C), $E(t)$ is a singular matrix for all t in the time interval of consideration and $X(t) \in R^{nn}$ (or C^{nn}). DAEs are great for describing systems where some variables don't change independently but are tied together through algebraic relations [14]. Now, solving these systems, especially when we are dealing with 1-index DAEs, is a real challenge for standard numerical methods [4, 5]. You run into issues like stiffness, the need for ridiculously small-time steps, loss of numerical accuracy (order reduction), and singular Jacobian that just refuse to cooperate [18]. Sure, we've tried speeding things up with parallel solvers and better preconditioner, but the core issue remains: small time steps are often unavoidable, which drives up the computational cost pretty fast [16, 31].

Despite a rich body of theoretical work on singular matrix differential systems, practical numerical solutions are still difficult to compute. Classical numerical solvers often face limitations when applied to SMDS, particularly when the index of the system is high or when the Jacobian becomes singular. These kinds of solvers might need very small time steps, which can make them very expensive to run and make them less accurate [16, 18]. Implicit methods can work well for stiff systems, but they are less effective for SMDS and can become unstable. Physics-Informed Neural Networks (PINNs) are of immense utility in solving differential equations, and also can be used for Differential equations with stiff or complex structures, in the last few years [17, 29]. These models directly include the laws of physics in the loss function and give approximations that do not use meshes. Extensions of PINNs have been proposed for differential-algebraic systems and higher-index DAEs, including hybrid schemes using Radau

IIA integration or attention mechanisms [22, 33]. However, the performance of PINNs depends on the selection of hyperparameters.

Much of the research until now has been focused on the design of PINNs, rather than exploring architecture depth, activation functions, optimizer settings, or training schedules. The strategies proposed by [21,40] work well for a class of problems but will not work for SMDS problems. It is observed that many authors come up with neural networks to solve the differential equations for a particular stiffness and constraint structure, and these cannot be applied to solve SMDS. If small changes in the initial condition causes erratic changes in the trajectories of the dynamical system of consideration, the validity of the computed solution is itself a major issue [22] to be addressed. Hence, it is of utmost importance to design and develop flexible and effective training methods for neural solvers in the SMDS framework. Therefore, it is required to design the structure of the network by selecting suitable optimizer for achieving stability, and rapid convergence [36, 37]. In this paper, we tried to fill this gap by constructing a semi-supervised PINN framework exclusively for linear SMDS.

In view of these challenges, this paper focuses on developing an adaptive neural network framework for solving linear singular matrix differential systems (LSMDS) using a semi-supervised learning approach. The objective of this paper is to construct a solution scheme that is not only mesh-free but also robust to stiffness and inconsistent initial conditions We investigate how various combinations of activation functions, optimizer scheduling, and network depth-width configurations can influence the quality and stability of solutions [30, 40]. We have made an attempt to improve the accuracy and convergence of solutions of SMDS by minimizing the loss function with a balanced approach of amalgamation of DAE residuals, initial conditions, and regularization

The following are the significant contributions of this study:

- We introduce a two-stage optimization framework that combines the global optimization power of Adam with the local fine-tuning capability of L-BFGS, exclusively designed for SMDS [35].
- Our approach is verified and validated by experiments and comparative study of several activation functions viz., Tanh, Sigmoid, ReLU, Swish with different widths and depths of the network [30,36].
- We develop a physics-informed loss that respects differential-algebraic constraints and initial conditions, even allowing for mesh-free training without the need for labeled solution data [22, 24, 29].
- Our investigations identified the architectural and training conditions that provide accurate and stable output for SMDS with reasonably good accuracy.

This paper is organized as follows: Section 2 presents the structure of solutions for homogeneous and nonhomogeneous singular matrix systems based on the fundamental matrix formulation. Section 3 describes the neural network architecture, activation functions, loss construction, and two-stage optimization process. Section 4 provides numerical examples that validate the proposed method. Section 5 concludes with observations and insights. Future directions are outlined in Section 6.

2 Singular Matrix Differential systems

This section describes the structure of solution of Singular Matrix System in terms of fundamental matrix solutions and also variation of parameters formula for the solution of the NHSMS (1.2).

Theorem 2.1. *If $\Phi(t, t_0)$ is the fundamental matrix solutions of $EX'(t) = A(t)X(t)$ then any solution $X(t)$ of the homogeneous Singular Matrix System*

$$E(t)X'(t) = A(t)X(t) \tag{2.1}$$

is of the form $\Phi(t, t_0)C$, where C is an arbitrary constant square matrix of order n.

Proof: We desire a solution of (2.1) in the form $X(t) = \Phi(t, t0)K(t)$ where $K(t)$ is a square matrix of order n, whose elements are functions defined on R. Then $E(\Phi(t, t_0)K(t))' = A(t)\Phi(t, t_0)K(t)$ i.e., $E\Phi'((t, t_0)K(t) + E\Phi(t, t_0)K'(t) = A(t)\Phi((t, t_0)K(t))$ i.e., . $E\Phi(t, t_0)K'(t) = 0$ i.e., $K'(t) = 0$ i.e., $K(t) = C$. □

Theorem 2.2. *Every solution $X(t)$ of the NHSMS (1.2) is of the form $\Phi(t,t_0)C+\overline{X}(t)$ where $\overline{X}(t)$ is a particular solution of (1.2).*

Proof: By simple substitution one can verify that $\Phi(t,t_0)C+\overline{X}(t)$ is a solution of the NHSMS (1.2). Now we would like to show that every solution is of this form. Let X(t) be any solution of (2.1) and $\overline{X}(t)$ be a particular solution of (2.1). Then it is obvious that $X(t)-\overline{X}(t)$ is a solution of (2.1). From theorem 2.1 it follows that $X(t)-\overline{X}(t)=\Phi(t,t_0)C$ i.e., $X(t)=\Phi(t,t_0)C+\overline{X}(t)$. □

Theorem 2.3. *Every solution of the NHSMS (1.2) is of the form $\Phi(t,t_0)C+\overline{X}$ where $\overline{X}$ is a particular solution of (1.2) and is given by $\int_{t_0}^{t}\Phi(t,s)E^{+}F(s)ds$.*

Proof: The general solution of the homogeneous Singular Matrix System (1.2) is of the form $\Phi(t,t_0)C$. Let C be a function of t defined on R .Let us impose the condition that $X(t)$ satisfies NHSMS (1.2). Then we have $E(\Phi(t,t_0)C(t))^{'} = A(t)\Phi(t,t_0)C(t)+F(t)$ i.e., $E\Phi^{'}((t,t_0)C(t)+E\Phi^{(}(t,t_0)C^{'}(t)$ $=A(t)\Phi((t,t_0)C(t))+F(t)$ i.e. $E\Phi^{(}(t,t_0)C^{'}(t)=F(t)$. Consider $EY=F$ set of all square matrices of order n such that the columns of $F(t)$ is the column space of E. i.e., $\Re(E)$. With the assumption of solvability of NHSMS (1.2) the solution space of (1.2) is non empty. The set Ω is indeed guaranteed to be non empty if $\Re(E)\supseteq\Re(F)$. However, this condition is not necessary. Therefore $Y=E^{+}F$, where E^{+} is the pseudo inverse of E. Now $C^{'}(t)=\Phi^{(}(t_0,t)E^{+}F(t)$ i.e., $C(t)=C_1+\int_{t_0}^{t}\Phi(t_0,s)E^{+}F(s)ds$. Thus $X(t)=\Phi(t,t_0)[C_1+\int_{t_0}^{t}\Phi(t_0,s)E^{+}F(s)ds]$. Hence the theorem. □

Theorem 2.4. *Every solution $X(t)$ of the initial value problem associated with the NHSMS (1.2) with the initial condition $X(t_0)=X_0$ where X_0 is a given square matrix of order n is of the form*

$$X(t)=\Phi(t,t_0)X_0+\int_{t_0}^{t}\Phi(t,s)E^{+}F(s)ds \tag{2.2}$$

3 Neural Network Approach to NHSMS

The previous section 2 gives the theoretical base of the existence of solution of singular Matrix systems subject to the existence of the fundamental matrix solution for the homogeneous Singular Matrix System (2.1). Computation of fundamental matrix solution is always not straightforward and possible, there by bringing a limitation on the computation of closed from solution of the singular matrix system. To over come this difficulty we construct neural networks to compute the reasonably accurate solutions of singular matrix system by optimizing hyper parameters.

3.1 Architecture Design

Let the neural network be denoted as:

$$x_\theta(t)=\mathrm{NN}_\theta(t) \tag{3.1}$$

where θ represents all trainable parameters of the neural network.
The network consists of L layers and is defined as follows:

$$\begin{aligned}
h^{(0)} &= t \quad \text{(Input layer)}\\
h^{(l)} &= A\left((Wm)^{(l)}h^{(l-1)}+(bv)^{(l)}\right), \quad \text{for } l=1,2,\ldots,L-1\\
x_\theta(t) &= (Wm)^{(L)}h^{(L-1)}+(bv)^{(L)} \quad \text{(Output layer)}
\end{aligned}$$

where:

- $(Wm)^{(l)}$ and $(bv)^{(l)}$ are the weight matrix and bias vector for layer l,
- $A(\cdot)$ is a activation function,

- $h^{(l)} \in \mathbb{R}^{d_l}$ is the output of the l-th layer,
- The final output $x_\theta(t) \in \mathbb{R}^n$ approximates the true solution $x(t)$.

The architecture we propose is a configurable, fully connected feedforward neural network. It's straightforward in structure but flexible in capacity. Each model consists of an input layer (taking in time values), several hidden layers, and an output layer that predicts the system's state variables at each time step. We can adjust the model complexity as required by varying the numbers of hidden layers (depth) and neurons per layer (width). Hidden layers are stacked using linear transformations followed by nonlinear activations, except for the final layer, which is purely linear. As a result, the network can estimate smooth time-dependent functions that describe differential system solutions [29]. The model can gradually acquire either coarse approximations or highly detailed behavior, depending on the width and depth. It is simpler to explore and determine which configuration performs best in terms of accuracy, convergence speed, and training stability thanks to its adaptable architecture.

3.2 Selection of Activation Functions

Activation functions are what give neural networks their nonlinearity, and let's be honest, without them, we'd just stack a bunch of linear functions and get nowhere. The choice of activation function can significantly impact how well and how fast the network learns [17]. In this study, we consider four types of activation functions.Tanh, Sigmoid, ReLU and Swish, [30] Each of these has its own pros and cons. For instance, Tanh works great with smaller networks and smoother functions, while ReLU can be better in deeper networks but sometimes causes dead neurons. So, we don't make assumptions, we test them all across various configurations. This kind of comparative exploration is necessary because the best activation can vary depending on the nature of the differential system and the architecture [36].

3.3 Two-Stage Optimization Strategy

Now, training the our model is where things get innovative. We do not stick with just single optimizer. Instead, we use a two-stage optimization strategy, To combine the strengths of different way approaches. Here's the idea: First, we use the Adam optimizer for a number of initial epochs. Adam is quick, adaptive, and great at jumping through the complex terrain of the loss landscape. Then, once the model reaches a decent neighborhood of the solution, we switch to L-BFGS, a quasi-Newton method that's more precise and ideal for fine-tuning [40]. This switching point is referred to as the changing point, and we experiment with several values (e.g., 10%, 20%, up to 50% of total epochs). The idea is that Adam gets us close, but L-BFGS cleans it up. The reason for doing this is pretty practical, Adam might converge faster early on but often stops improving before hitting a really low loss. L-BFGS, while slower at the beginning, is extremely effective at local optimization once we're near the minimum. So, we get the best of both worlds by combining them [37].

3.4 Loss Function Formulation

The total loss enforces the DAE dynamics, initialconditions, and regularization. Let $\hat{x}(t;\theta_x)$ and $\hat{z}(t;\theta_z)$ be the neural network predictions for differential and algebraic variables, respectively.

Composite Loss

$$\mathcal{L}(\theta_x, \theta_z) = \mathcal{L}_{\text{DAE}} + \mathcal{L}_{\text{IC}} + \mathcal{L}_{\text{Reg}} \tag{3.2}$$

DAE Residual Loss (Physics-Informed) Evaluated at Radau collocation points $\{t_k\}_{k=1}^5$:

$$\mathcal{L}_{\text{DAE}} = \frac{1}{N}\sum_{k=1}^{N}\left(\|\dot{\hat{x}}(t_k) - f(\hat{x}(t_k), t_k)\|^2 + \|g(\hat{x}(t_k), t_k)\|^2\right) \tag{3.3}$$

- **Automatic Differentiation**: Compute $\dot{\hat{x}}(t_k)$ via backpropagation.
- **Algebraic Constraint**: The term $\|g(\hat{x}, t)\|^2$ ensures manifold adherence.

Initial

$$\mathcal{L}_{\text{IC}} = \|\hat{x}(t_0) - x_0\|^2 \tag{3.4}$$

Regularization Loss

$$\mathcal{L}_{\text{Reg}} = \|\theta_x\|_2^2 + \|\theta_z\|_2^2 \quad \text{(L2 penalty)} \tag{3.5}$$

For the loss function, we stick with L2 penalty, but it's more than just a standard we took. In our method, L2 penalty is used to compare the PINNs predicted solution with a reference solution obtained from solving the SMD system using Neural Network. [38, 39]. We now present the design and implementation of neural network model to solve the singular matrix differential system.

3.5 Training Process

The model is trained in two stages. Initially, the Adam optimizer helps the network converge quickly. After a fixed number of epochs, training switches to L-BFGS to fine-tune the weights. This hybrid approach gave more stable and accurate results compared to using either optimizer alone.

3.6 Parameter Tuning

A grid search was used to test different combinations of model width, depth, activation functions, and optimizer switching points. Every setup was trained for a fixed number of epochs, and the performance was recorded based on loss values.

3.6.1 Changing Point We need to choose when to switch optimizers during training. To do this, we use a setting called Changing Point, which is a number between 0 and 1. It tells us what portion of the total training time will use the Adam optimizer. For example, if we train for 5000 epochs and the Changing Point is 0.4, that means we will use Adam for 2000 epochs (5000×0.4), and then switch to L-BFGS for the remaining 3000 epochs. Instead of testing every possible value, we only try five specific options: 0.1, 0.2, 0.3, 0.4, 0.5.

4 Examples

In this section three examples that demonstrates the developed optimized hyperparameter neural networks are presented.

Example 4.1. *Consider the NHSMS* (1.2) *with*

$$E = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix}, \quad F(t) = \begin{bmatrix} t & 1+t \\ 1 & t^2 \end{bmatrix}, X(t) = \begin{bmatrix} x_{11}(t) & x_{12}(t) \\ x_{21}(t) & x_{22}(t) \end{bmatrix},$$

satisfying the initial condition(IC) $X(0) = \begin{bmatrix} 0.5 & 1 \\ 0 & 1 \end{bmatrix}$.

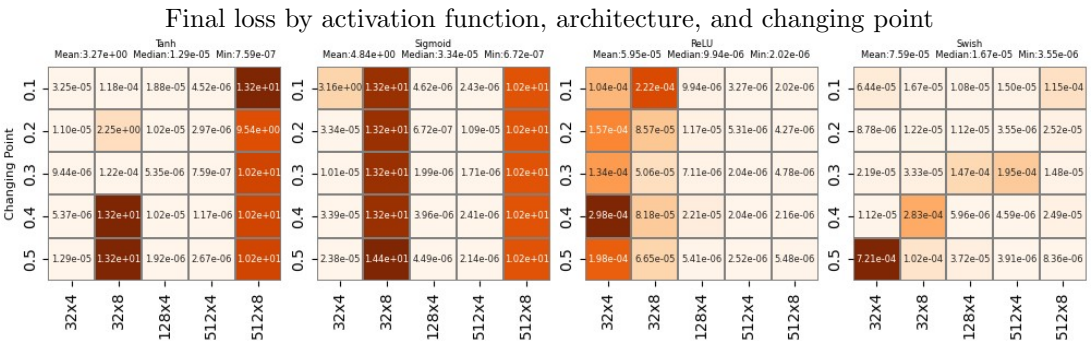

Figure 1: Example 4.1: Heat Map

- *Swish tends to perform better, especially at higher widths like 128 or 512, final losses stay really low.*

- *Tanh starts off well, but with depth 8, it gets unstable. The loss jumps a lot in some cases*

The comparison of the solution obtained by the numerical Radau method and PINN is shown in Figure 2.

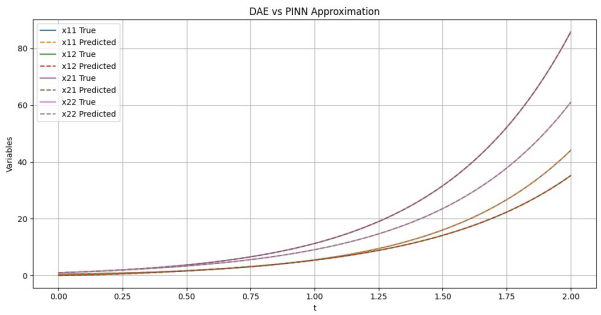

Figure 2: Example 4.1: Numerical method vs. PINN

Example 4.2. *Consider the NHSMS (1.2) with*

$$E = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad A(t) = \begin{bmatrix} \sin t & \cos t \\ \cos t & \sin t \end{bmatrix}, \quad F(t) = \begin{bmatrix} t & 1+t \\ 1 & t^2 \end{bmatrix}, X(t) = \begin{bmatrix} x_{11}(t) & x_{12}(t) \\ x_{21}(t) & x_{22}(t) \end{bmatrix},$$

satisfying the IC $X(0) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$.

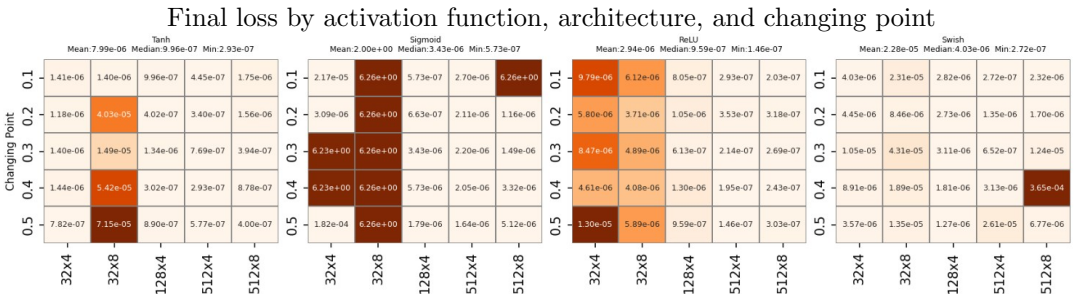

Figure 3: Example 4.2: Heat Map

- *activations show low losses across all configurations, but Sigmoid becomes unstable at higher depths.*

- *ReLU and Swish maintain consistently low loss values, especially with moderate changing points.*

The comparison of the solution obtained by the numerical Radau method and PINN is shown in Figure 4.

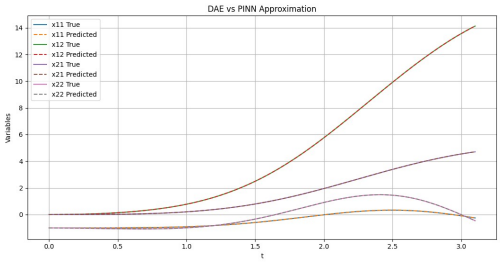

Figure 4: Example 4.2: Numerical method vs. PINN

Example 4.3. *Consider the NHSMS:*

$$E = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad A(t) = \begin{bmatrix} e^{-t} & 0 \\ e^{t} & e^{-2t} \end{bmatrix}, \quad F(t) = \begin{bmatrix} \sin(t) & t \\ 0 & \cos(t) \end{bmatrix}, X(t) = \begin{bmatrix} x_{11}(t) & x_{12}(t) \\ x_{21}(t) & x_{22}(t) \end{bmatrix},$$

satisfying the IC $X(0) = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$.

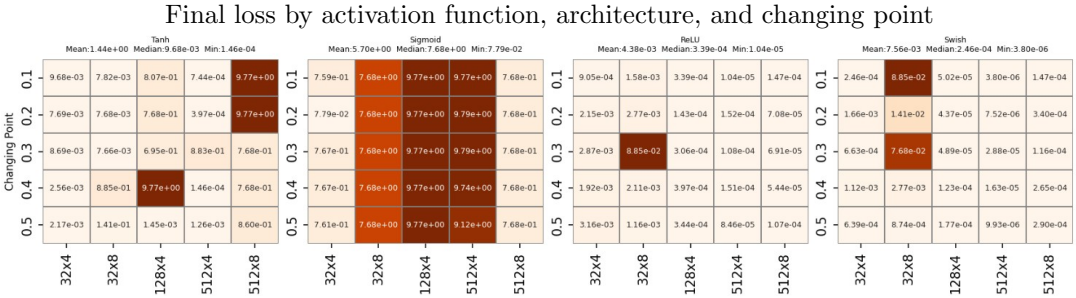

Figure 5: Example 4.3: Heat Map

- *ReLU and Swish show strong results, very low losses at wider settings, especially 512.*

- *Sigmoid doesn't improve much. The loss hangs around 9.76 most of the time, barely changes, even with different setups.*

The comparison of the solution obtained by the numerical Radau method and PINN is shown in Figure 6.

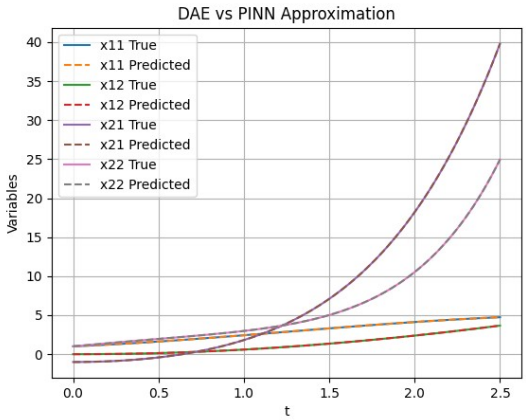

Figure 6: Example 4.3: Numerical method vs. PINN

4.1 Heatmap Analysis

Based on the Heat Maps corresponding to the above three examples the following are the observations are made:

- **ReLU and Swish give the lowest losses across all three problems**
  These activations work best, especially with wider networks like width = 128 or 512. Their performance stays steady even as the changing point increases, showing they are reliable across settings. Tanh and Sigmoid, on the other hand, are more affected by their non-linear nature.

- **Sigmoid often causes high or even exploding losses, especially in deeper networks:**
  For example, with width = 128 or 512 and depth = 4 or 8, the loss often goes above 9.76. This points to possible numerical issues or poor gradient flow when using Sigmoid.

- **Swish activation shines with larger widths and shallow depths, especially for Problem 3:**
  Notably, for width 128, depth 4, Swish at changing point 0.1 or 0.2 loss values as low as 5.02×10^{-5}. This suggests Swish's capacity for better generalization in nonlinear systems.

- **Tanh performs well in limited conditions small depth and moderate width but degrades with scale:**
  the non-linear Tanh performs best when depth = 4 and width = 32, such as at shifting Adam to L-BFGS and 0.4 or 0.5. However, when scaling up to depth 8 and width 128 or 512, it either diverges or delivers high final losses.

- **ReLU shows remarkably consistent behavior across problems, even in large models:**
  For both depth 4 and 8, and width up to 512, ReLU is low and stable high losses. No sudden spikes. Clear behavior.

- **Some Tanh is seem unstable only at some changing point values:** For example, width 512, depth 8, changing point 0.1 or 0.2 causes loss is nearly 9.7 for Problem 1. Suggests a tipping point where gradient flow breaks down.

5 Conclusions

In this study, we explored adaptive Physics Informed neural network approach to tackle Linear Singular Matrix Differential Systems (LSMDS) from a semi-supervised point of view. The solutions obtained by our developed approach defined by Optimization algorithms, network structure, activation, and training strategies, could combat computational challenges posed by singular systems. The three examples of

our consideration are divergent in nature and our experiments with various combinations of activation functions, depths, widths, and optimizer change points (optimizer switching) clearly demonstrated the improvement of model accuracy.

The result, across trials, was consistent, with lowest final loss achieved by ReLU and Swish activation, especially with broader architectures. Tanh activation represented inconsistent patterns at times reasonable but with a general tendency to be unstable with deeper configurations. Sigmoid was very likely to converge, especially with higher width, and depth.From the observations, it is established that wider and deeper models are not necessarily better. In fact, our experiments established that a depth 4 and width 128 network (i.e., 4 hidden layers with 128 neurons per layer) achieved a balanced trade-off between generalization and model complexity suitable for LSMDS. Another important observation is that changing point values between 0.4 to 0.5 during the two-stage optimization protocol (Adam followed by L-BFGS) yielded a stable and accurate convergence across problems.The primary contribution of the our research lies in integrating adaptive architecture selection and hybrid optimization methods, neural solvers developed exclusively for singular systems. The proposed loss function successfully enforces the DAE dynamics and initial constraints without mesh-based methods.

These findings provide practical design methodologies to build effective neural solvers within a constrained dynamical system framework. They also provide opportunities for future work towards automated hyperparameter optimization, parallel computation structures, and extensions to large-index DAEs and large-index physical models.

6 Future Work

Characterization and classification of efficient neural networks with corresponding bounds on hyper parameters is to be explored. We try to explore the tradeoff of the hyper parameters on the parallel computation setup and HPC setup. We are planning to investigate the time complexity of the developed methodology and try to arrive at an effective and efficient hyperparameter portfolio.

Acknowledgments

This work is supported by the RUSA project letter No. RUSA-ANURU/Research Project-02/Sanction Order/2024 dt01-06-2024 and the authors would like to thank RUSA.

References

1. P.V.S.Anand and K.N.Murty, Controllability and Observability of Liapunov type matrix difference system, *In Proceedings of* 50*th Congress of ISTAM (An International Meet) IIT Kharagpur*, 125–132, 2005.
2. P.V.S.Anand, Controllability and Observability of the Matrix Lyapunov Systems, *In Proceedings of the international conference on Recent Advances in Mathematical Science and Applications (RAMSA) held at Vizag*, 117–131, 2009.
3. K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, 1996.
4. S. L. Campbell, C. D. Meyer, and N. J. Rose, "Applications of Drazin Inverse to Linear System of Differential Equations with Singular Coefficients," *SIAM J. Appl. Math.*, vol. 3, pp. 411–425, 1976.
5. S. L. Campbell, "Linear System of Differential Equations with Singular Coefficients," *SIAM J. Math. Anal.*, vol. 6, pp. 1057–1066, 1977.
6. S. L. Campbell, *Singular Systems of Differential Equations*, Pitman, Boston, 1980.
7. S. L. Campbell, "High-index DAEs in applications," *Appl. Numer. Math.*, vol. 5, 1989.
8. L.N.Charyulu, Rompicharla, Venkata Sundaranand Putcha, G.V.S.R.Deekshithulu, Controllability and observability of fuzzy matrix discrete dynamical systems, *Journal of Nonlinear Sciences and Applications*, **12**, 816–828, 2019.
9. L.N.Charyulu, Rompicharla, Venkata Sundaranand Putcha, G.V.S.R.Deekshithulu, Existence of $(\phi \otimes \psi)$ bounded Solutions For Linear First Order Kronecker Product Systems, *In International Journal of Research Scientific Journal*, **11**, Issue 06(E), 39047–39053, 2020.
10. L.N.Charyulu, Rompicharla, Venkata Sundaranand Putcha, G.V.S.R.Deekshithulu, Kronecker product Three point boundary value problems Existence and Uniqueness, *In International Research Journal of Engineering and Technology (IRJET)*, **8**, Issue:02, 2021.
11. D. Cobb, "On the Solution of Linear Differential Equations with Singular Coefficients," *J. Differential Equations*, vol. 46, no. 3, pp. 310–323, 1982.

12. P. Van Dooren, "The Generalized Eigenstructure Problem in Linear Systems Theory," *IEEE Trans. Automat. Control*, vol. 26, pp. 111–129, 1981.

13. P. Van Dooren, "The Eigenstructure of an Arbitrary Polynomial Matrix: Computational Aspects," *Linear Algebra Appl.*, vol. 50, pp. 545–579, 1983.

14. C. W. Gear, "Numerical Initial Value Problems in Differential-Algebraic Equations," *IEEE Trans.*, vol. 36, 1988.

15. T. Geerts, "Invariant Subspaces and Invertibility Properties for Singular Systems: The General Case," *Linear Algebra Appl.*, vol. 183, pp. 61–68, 1993.

16. A. Griewank, "On solving singular systems numerically," *Math. Comp.*, vol. 42, pp. 123–132, 1984.

17. G. E. Karniadakis et al., "Physics-informed machine learning," *Nature Rev. Phys.*, vol. 3, pp. 422–440, 2021.

18. P. Kunkel and V. Mehrmann, *Differential-Algebraic Equations: Analysis and Numerical Solution*, EMS, 2006.

19. P. Lancaster, "Explicit Solutions of Linear Matrix Equations," *SIAM Review*, vol. 12, no. 4, pp. 544–566, 1970.

20. V. Lakshmikantham and D. Trigiante, *Theory of Difference Equations – Theory, Methods and Applications*, Academic Press, 1988.

21. J. Lee and A. Ryou, "Hyperparameters in PINNs," *Mach. Learn. Sci. Technol.*, vol. 3, no. 1, 2022.

22. D. Li et al., "DAE-PINN: Neural approach for constrained dynamics," *IEEE Trans. Neural Netw.*, vol. 34, no. 2, 2023.

23. J. J. Loiseau, "Some Geometric Considerations about the Kronecker Normal Form," *Internat. J. Control*, vol. 42, pp. 1411–1413, 1985.

24. Y. Lu et al., "Deep learning of dynamics with constraints," *Neural Comput.*, vol. 33, no. 5, 2021.

25. K.N.Murty, K.R.Prasad, and P.V.S.Anand, Two-point boundary value problems associated with Lyapunov type matrix difference system, *Dynamic Systems and Applications, USA*, **4(2)**, 205–213, 1995.

26. K.N.Murty, P.V.S.Anand, and V.Lakshmi Prasannam, First Order Difference System Existence and Uniqueness, *Proceedings of the American Mathematical Society*, **125(12)**, 3533–3539, 1997.

27. L. Petzold, "Differential/algebraic equations are not ODEs," *SIAM J. Sci. Stat. Comput.*, vol. 3, pp. 367–384, 1982.

28. Putcha, V. S. (2014). Discrete linear Sylvester repetitive process. *Nonlinear Studies*, 21(2), 205–218.

29. M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems," *J. Comput. Phys.*, vol. 378, pp. 686–707, 2019.

30. P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," *arXiv preprint arXiv:1710.05941*, 2017.

31. Venkata Sundaranand Putcha, D.P.R.V.Subba Rao, and Ramu Malladi, Existence of ψ-Bounded Solutions for First Order Matrix Difference System on Z, *In International journal of Embedded Systems and Emerging Technologies*, **5(1)**, 6–16, 2019.

32. G. C. Verghese, B. Levy, and T. Kailath, "A Generalized State Space for Singular Systems," *IEEE Trans. Automat. Control*, vol. 26, pp. 811–831, 1981.

33. J. Chen, J. Tang, M. Yan, S. Lai, K. Liang, J. Lu, and W. Yang, "Physical information neural networks for solving high-index differential-algebraic equation systems based on Radau methods," *arXiv preprint arXiv:2310.12846*, 2023. Available: https://arxiv.org/abs/2310.12846

34. E. L. Yip and R. F. Sincovec, "Solvability, Controllability, and Observability of Continuous Descriptor Systems," *IEEE Trans. Automat. Control*, vol. 26, pp. 702–707, 1981.

35. X. Wu, D. Zhang, M. Zhang, C. Guo, S. Zhao, Y. Zhang, H. Wang, and B. Yang, "AutoPINN: When AutoML meets physics-informed neural networks," *arXiv preprint arXiv:2212.04058*, 2022. Available: https://arxiv.org/abs/2212.04058

36. A. D. Jagtap and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," *J. Comput. Phys.*, vol. 404, p. 109136, 2020.

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

38. R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

39. C. Rackauckas et al., "Universal differential equations for scientific machine learning," *arXiv preprint arXiv:2001.04385*, 2020.

40. Y. Wang, N. Patel, Z. Lu, L. Sun, and R. Yu, "Auto-PINN: Understanding and optimizing physics-informed neural architecture," *arXiv preprint arXiv:2205.13748*, 2022.

Venkata Sundaranand Putcha,
Professor
Department of Mathematics,
Rayalasema University,Kurnool-518007(A.P) India.
E-mail address: putcha@yahoo.com

and

Praveen Kumar U M,
Research Scholar
Department of Mathematics,
Rayalasema University,Kurnool-518007(A.P) India.
E-mail address: praveenumkumar5@gmail.com