

COMPACT GRADIENT-BASED NEURAL NETWORK FOR STOCHASTIC SUPPORT VECTOR REGRESSION WITH PROBABILISTIC CONSTRAINTS

NUTT TANANIMIT^{1,2}, THANATPORN GRACE¹, YIRGA ABEBE BELAY³, RABIAN WANGKEEREE^{4,*}

¹*Department of Mathematics Education, Walailak University, Nakhon Si Thammarat, Thailand*

²*Research Center for Intelligent Technology and Integration, Walailak University,
Nakhon Si Thammarat, Thailand*

³*Department of Mathematics, Aksum University, Aksum, Ethiopia*

⁴*Department of Mathematics, Naresuan University, Phitsanulok, Thailand*

Abstract. This paper introduces a neurodynamic system for solving stochastic support vector regression (SSVR) with probabilistic constraints. The proposed approach reformulates the SSVR problem into a deterministic quadratic programming problem. To address the nonsmoothness arising from the Karush–Kuhn–Tucker complementarity conditions, a smoothed Fischer–Burmeister function is employed, yielding a gradient-based neural network with enhanced numerical stability. The existence, uniqueness, and convergence of the network’s trajectories are rigorously analyzed, and the system is proven to be globally convergent to the optimal solution of the original SVR problem; Lyapunov stability is also established. Compared to existing methods, the proposed architecture achieves 25% reduction in network complexity leading to substantially faster training times. The efficiency and superiority of the model are validated through extensive numerical experiments on ten UCI benchmark datasets, where our method outperforms three existing SSVR baselines.

Keywords. Fischer-Burmeister function; Neural network; Probabilistic constraint; Quadratic optimization; Support vector regression.

1. INTRODUCTION

Support Vector Machine (SVM) is a well-established concept in statistics and computer science, encompassing a set of supervised learning methods used for classification and regression analysis across a wide range of scientific and engineering domains such as pattern recognition, text categorization, bioinformatics, computer vision, chemometrics, and many other real-world prediction tasks [7, 9, 18, 22, 41, 42]. Originally introduced by Vapnik [9, 47], the SVM framework was later extended to regression problems by Drucker et al., resulting in a method known as Support Vector Regression (SVR) [2, 11, 51]. As a result, SVR has become a widely used and effective tool for real-valued function estimation and has been successfully applied to time-series and financial forecasting, complex engineering modeling, multivariate calibration, and

*Corresponding author.

E-mail address: rabianw@nu.ac.th (R. Wangkeeree).

Received 27 April 2026; Accepted 14 June 2026; Published online 1 November 2026.

other noisy, and nonlinear regression problems [9, 18, 39]. Various enhancements were proposed, such as the penalized linear regression by Shivaswamy et al. [40], the Extreme Learning Machine (ELM) [7, 19], and the Twin Support Vector Regression (TSVR) [20, 38, 44].

SVR represents a powerful framework for regression tasks that seek to find a function that deviates from the observed responses by at most ε while maintaining maximum flatness [47]. The classical SVR formulation assumes deterministic input and output pairs, where the training data $\{(X_1, Y_1), \dots, (X_n, Y_n)\}$ consists of fixed vectors $x_i \in \mathbb{R}^m$ and scalar responses $y_i \in \mathbb{R}$.

The standard ε -insensitive SVR optimization problem is formulated as follows [9]:

$$\begin{aligned} & \text{minimize } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (\zeta_i + \zeta'_i) \\ & \text{subject to } \begin{cases} y_i - w^\top \phi(x_i) - b \leq \varepsilon + \zeta_i, \\ w^\top \phi(x_i) + b - y_i \leq \varepsilon + \zeta'_i, \\ \zeta_i, \zeta'_i \geq 0, \quad i = 1, \dots, n, \end{cases} \end{aligned}$$

where $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^{m_1}$ denotes a feature mapping to a higher-dimensional space, $C > 0$ is the regularization parameter controlling the trade-off between model complexity and training error, and ζ_i, ζ'_i represent slack variables that allow violations of the ε -tube constraints. The ε -insensitive function is defined as:

$$L_\varepsilon(d, y) = \begin{cases} |d - y| & \text{if } |d - y| \geq \varepsilon, \\ 0 & \text{otherwise.} \end{cases}$$

Despite its success, the standard SVR formulation typically assumes that training data are known exactly [41]. However, in many practical engineering and statistical applications, data cannot be observed precisely due to sampling, modeling, or measurement errors [6, 40, 45]. To address this uncertainty, recent studies have demonstrated that robust support vector regressions (RSVRs) often outperform standard SVRs [4, 5, 8, 46]. A particularly powerful approach to handling this uncertainty is the use of probabilistic constraints, where input and output data are modeled as random variables with known expectations [1, 16].

Consider the given training samples $\{(X_1, Y_1), \dots, (X_n, Y_n)\}$, where each input $X_i = [X_i^1, \dots, X_i^m]^\top$ is a random variable in $\mathbb{R}^m$ with expectation $E(X_i) = [E(X_i^1), \dots, E(X_i^m)]^\top$ and each output Y_i is a random variable in $\mathbb{R}$ with expectation $E(Y_i)$. To accommodate this stochastic setting, the SVR problem can be formulated using probabilistic constraints:

$$\begin{aligned} & \text{minimize } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (\zeta_i + \zeta'_i) \\ & \text{subject to } \begin{cases} \Pr(Y_i - w^\top \phi(X_i) - b \leq \varepsilon + \zeta_i) \geq \delta, \\ \Pr(w^\top \phi(X_i) + b - Y_i \leq \varepsilon + \zeta'_i) \geq \delta, \\ \zeta_i, \zeta'_i \geq 0, i = 1, \dots, n, \end{cases} \end{aligned}$$

where $\delta \in (0, 1)$ represents the desired confidence level for satisfying the regression constraints. The feature mapping $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^{m_1}$ with $m_1 \geq m$ transforms the input variables to a potentially higher-dimensional feature space.

Theorem 1.1. [1] *Let $\phi(X_i)$ and Y_i are random vectors with expectations $E(\phi(X))$ and $E(Y)$ respectively. Then, for every $i = 1, \dots, n$, the following implications hold:*

$$\begin{aligned} E(Y_i) - w^\top E(\phi(X_i)) - b \leq \zeta_i - 2a\delta - \varepsilon &\implies \Pr(Y_i - w^\top \phi(X_i) - b \leq \varepsilon + \zeta_i) \geq \delta, \\ w^\top E(\phi(X_i)) + b - E(Y_i) \leq \zeta'_i - 2a\delta - \varepsilon &\implies \Pr(w^\top \phi(X_i) + b - Y_i \leq \varepsilon + \zeta'_i) \geq \delta, \end{aligned}$$

where $a > \varepsilon$ is a positive constant.

This theorem transforms the above optimization problem into a deterministic equivalent (detailed in [1]):

$$\text{minimize } \frac{1}{2} w^\top w + C \sum_{i=1}^n (\zeta_i + \zeta'_i) \quad (1.1)$$

$$\text{subject to } \begin{cases} E(Y_i) - w^\top E(\phi(X_i)) - b \leq \zeta_i - 2a\delta - \varepsilon, \\ w^\top E(\phi(X_i)) + b - E(Y_i) \leq \zeta'_i - 2a\delta - \varepsilon, \\ \zeta_i, \zeta'_i \geq 0, i = 1, \dots, n. \end{cases} \quad (1.2)$$

Recently, authors proposed various methods for solving probabilistic support vector regression problems; see, e.g., [1, 26, 27]. Abaszade and Effati [1] proposed a novel model which solves the SVR by a means of solving quadratic optimization problem. Liu et al. [27] introduced a model which employs the Probabilistic Support Vector Regression for short term forecasting of power plant equipment. While effective, solving these stochastic support vector regression (SSVR) problems typically involves converting them into convex optimization problems, such as Quadratic Programming (QP) or Second-Order Cone Programming (SOCP) [14, 16, 28, 52]. Traditional numerical algorithms, such as interior-point methods, are often dependent on the problem's dimension and can be computationally expensive, making them unsuitable for real-time applications [14, 28, 30]. This limitation has driven a surge of interest in neural network-based optimization [14, 16, 28, 50, 52].

The history of neural networks for convex optimization has evolved significantly since the pioneering work of Tank and Hopfield in 1986 [43]. Early approaches, such as the gradient neural network by Kennedy and Chua [23], utilized penalty parameters to enforce constraints [23]. These methods only provided approximate solutions and required impractical parameter tuning [28, 50]. Subsequent research in the 1990s, such as the projection neural networks by Friesz et al. [17], eliminated penalty parameters but often required restrictive assumptions like monotonicity or symmetric gradients [17, 25, 49].

A major breakthrough occurred with the development of Lagrangian and KKT-based neurodynamic models. Nazemi [31] introduced a Lagrangian approach that achieved exact optimal solutions for strictly convex problems [31]. This was further generalized in 2013 to handle general convex problems without penalty parameters, guaranteeing global convergence to exact KKT points and found numerous applications [32, 33, 34]. More recent extensions have applied these frameworks to variational inequalities and robotics, demonstrating their versatility [35]. Parallel to these theoretical advancements, recent works have applied Recurrent Neural Networks (RNNs) to SVM learning specifically. For instance, recent studies have proposed Optimal RNNs for density-based SVMs, demonstrating superior time complexity over traditional kernel methods [29, 30]. Similarly, neurodynamic approaches have been successfully

extended to semi-supervised feature selection and fuzzy convex programming in recent years, highlighting the continued vitality of this field [15, 21, 48].

In 2022, Feizi et al. [16] proposed a recurrent neural network for SVR with probabilistic constraints that converts the stochastic problem into a deterministic quadratic program, derives dynamics from the KKT conditions, and achieves global asymptotic stability without penalty parameters using only $8n$ neurons. Motivated by their work and by smoothing techniques from related fields [37, 42], the present paper addresses two remaining challenges: the architectural dimension ($8n$ neurons) and the nonsmoothness inherent in the model. We propose a more efficient framework that reduces the neuron count to $6n$, lowering memory and computational cost, and incorporates a smoothed Fischer–Burmeister function to obtain a continuous-time dynamics with enhanced differentiability and stability properties.

In this paper, we make the following key contributions:

- (1) We propose a compact gradient-based neural network architecture that reduces the network complexity from $8n$ to $6n$ neurons, achieving a 25% reduction in network size while maintaining theoretical convergence guarantees.
- (2) We employ a smoothed Fischer–Burmeister function to handle the nonsmooth KKT complementarity conditions, resulting in a gradient-based neural network with enhanced numerical stability.
- (3) We provide rigorous analysis of the network’s existence, convergence, and Lyapunov stability properties, proving global convergence to optimal solutions.
- (4) We conduct extensive numerical experiments on both synthetic data and ten UCI benchmark datasets [3], demonstrating superior performance compared to three prior neurodynamic SSVR models: the Lagrangian network of Nazemi [31], the smoothed Fischer–Burmeister gradient network of Nazemi and Nazemi [34], and the projection-based recurrent network of Feizi et al. [16].

The remainder of this paper is organized as follows. Section 2 introduces the preliminary concepts of SSVR and introduces the proposed work. Section 3 provides the theoretical foundation for global convergence. Section 4 offers a detailed comparison of our proposed architecture with existing models, including the one by Feizi et al. Finally, Section 5 presents numerical experiments using both linear and nonlinear SVR kernels.

2. PROPOSED METHOD

This section presents the mathematical formulation of our proposed SSVR with probabilistic constraints. We begin with the reformulation of the deterministic optimization problem (1.1)–(1.2). Through Lagrangian duality, we obtain

$$\begin{aligned} \text{maximize } & \sum_{i=1}^n (\alpha_i - \alpha'_i) E(Y_i) + (2a\delta + \varepsilon) \sum_{i=1}^n (\alpha_i + \alpha'_i) \\ & - \frac{1}{2} \sum_{i,j=1}^n (\alpha_i - \alpha'_i)(\alpha_j - \alpha'_j) \left[E(\phi(X_i))^\top E(\phi(X_j)) \right], \end{aligned} \quad (2.1)$$

$$\text{subject to } \sum_{i=1}^n (\alpha_i - \alpha'_i) = 0, \quad 0 \leq \alpha_i, \alpha'_i \leq C, \quad i = 1, \dots, n. \quad (2.2)$$

To facilitate efficient implementation, we introduce compact matrix notation. Define the dual variable vector $\theta = [\alpha_1, \dots, \alpha_n, \alpha'_1, \dots, \alpha'_n]^\top \in \mathbb{R}^{2n}$, kernel matrix $K \in \mathbb{R}^{n \times n}$ with $K_{ij} = E(\phi(X_i))^\top E(\phi(X_j))$, and matrices:

$$\begin{aligned} Q &= \begin{bmatrix} K & -K \\ -K & K \end{bmatrix} \in \mathbb{R}^{2n \times 2n}, \\ d &= \begin{bmatrix} -E(Y) - (2a\delta + \varepsilon)\mathbf{1}_n \\ E(Y) - (2a\delta + \varepsilon)\mathbf{1}_n \end{bmatrix} \in \mathbb{R}^{2n}, \\ e &= [-\mathbf{1}_n^\top, \mathbf{1}_n^\top]^\top \in \mathbb{R}^{2n}, \\ A &= \begin{bmatrix} -I_{2n} \\ I_{2n} \end{bmatrix} \in \mathbb{R}^{4n \times 2n}, \quad b = [\mathbf{0}_{2n}^\top, C\mathbf{1}_{2n}^\top]^\top \in \mathbb{R}^{4n}, \end{aligned}$$

where $\mathbf{1}_n$ denotes an n -dimensional vector of ones and $\mathbf{0}_n$ denotes an n -dimensional vector of zeros. Using this notation, optimization problem (2.1)–(2.2) can be compactly stated as the following quadratic optimization problem:

$$\text{minimize } \frac{1}{2} \theta^\top Q \theta + d^\top \theta, \quad (2.3)$$

$$\text{subject to } A\theta - b \leq 0, \quad e^\top \theta = 0. \quad (2.4)$$

The Lagrangian function for quadratic optimization problem (2.3)–(2.4) is given as

$$\mathcal{L}(\theta, u, v) = \frac{1}{2} \theta^\top Q \theta + d^\top \theta + u^\top (A\theta - b) + v(e^\top \theta),$$

where $u \in \mathbb{R}^{4n}$, $u \geq 0$, and $v \in \mathbb{R}$.

Lemma 2.1 (KKT Optimality Conditions). *If θ^* is an optimal solution of (2.3)–(2.4), then there exists a point (θ^*, u^*, v^*) such that $u^* \geq 0$ with*

$$\text{Stationarity :} \quad \nabla_{\theta} \mathcal{L}(\theta^*, u^*, v^*) = Q\theta^* + d + A^\top u^* + v^* e = 0, \quad (2.5)$$

$$\text{Primal feasibility :} \quad A\theta^* - b \leq 0, e^\top \theta^* = 0, \quad (2.6)$$

$$\text{Dual feasibility :} \quad u^* \geq 0, \quad (2.7)$$

$$\text{Complementarity :} \quad u^{*T} (A\theta^* - b) = 0. \quad (2.8)$$

Moreover, the converse is also true, i.e., for a feasible θ^* , if there exist multipliers u^*, v^* satisfying equations (2.5)–(2.8), then θ^* is an optimal solution of (2.3)–(2.4).

To handle the complementarity conditions numerically, we employ the smoothed Fischer-Burmeister (hereafter, FB) function [12, 37, 42], for $a, b \in \mathbb{R}$,

$$\phi_{\text{FB}}^{\varepsilon_{\text{FB}}}(a, b) = a + b - \sqrt{a^2 + b^2 + \varepsilon_{\text{FB}}}, \quad \varepsilon_{\text{FB}} > 0,$$

which is applied componentwise to vectors $x, y \in \mathbb{R}^p$, i.e.,

$$\phi_{\text{FB}}^{\varepsilon_{\text{FB}}}(x, y) = \left[x_1 + y_1 - \sqrt{x_1^2 + y_1^2 + \varepsilon_{\text{FB}}}, \dots, x_p + y_p - \sqrt{x_p^2 + y_p^2 + \varepsilon_{\text{FB}}} \right].$$

The equivalence

$$\phi_{\text{FB}}^{\varepsilon_{\text{FB}}}(a, b) = 0 \iff a > 0, b > 0, ab = \varepsilon_{\text{FB}}/2$$

provides a smooth approximation to the complementarity conditions for small ε .

Lemma 2.2 (Zero Set of Smoothed FB Mapping [13]). *For vectors $x, y \in \mathbb{R}^p$, $\phi_{\text{FB}}^{\varepsilon}(x, y) = 0$ if and only if $x \geq 0, y \geq 0, x \circ y = 0$, i.e., $x_i \cdot y_i = 0$ for every i .*

Define projection matrices as $\tilde{P} = e(e^\top e)^{-1}e^\top$ and $\tilde{Q} = e(e^\top e)^{-1}$. The projection matrices $\tilde{P}$ and $\tilde{Q}$ will play a crucial role in the subsequent residue system formulation and neural dynamics development. By enforcing constraints with a certain probability level (e.g., chance-constrained programming), the model becomes robust to stochastic variations [1, 16].

Lemma 2.3 (FB Residual System Formulation). *If θ^* is an optimal solution of (2.3)–(2.4), then there exists u^* such that*

$$(I - \tilde{P})(Q\theta^* + d + A^\top u^*) + \tilde{Q}(e^\top \theta^*) = 0, \quad (2.9)$$

$$\phi_{\text{FB}}^{\varepsilon}(u^*, b - A\theta^*) = 0 \quad (2.10)$$

where ε is a small positive real number. Moreover, the converse holds.

Proof. Let (θ^*, u^*, v^*) be the triple stated in Lemma 2.1. From equation (2.5), $Q\theta^* + d + A^\top u^* + v^*e = 0$. Left-multiplying it by $(I - \tilde{P})$ gives $(I - \tilde{P})(Q\theta^* + d + A^\top u^* + ev^*) = 0$. Since $(I - \tilde{P})e = e - e(e^\top e)^{-1}e^\top e = 0$ and $e^\top \theta^* = 0$, we have

$$(I - \tilde{P})(Q\theta^* + d + A^\top u^*) + \tilde{Q}(e^\top \theta^*) = 0.$$

Again from KKT, we have $u^* \geq 0, b - A\theta^* \geq 0$. Since $u^{*T}(A\theta - b) = 0$, $\phi_{\text{FB}}^{\varepsilon}(u^*, b - A\theta^*) = 0$ follows from Lemma 2.2.

Conversely, multiply (2.9) on the left by $e^\top$. Since $e^\top(I - \tilde{P}) = 0$ and $e^\top \tilde{Q} = 1$,

$$0 = e^\top(I - \tilde{P})(Q\theta^* + d + A^\top u^*) + e^\top \tilde{Q}(e^\top \theta^*) = e^\top \theta^*,$$

which gives primal feasibility ($e^\top \theta^* = 0$). Let us denote $Q\theta^* + d + A^\top u^*$ by w , so (2.9) reduces to $(I - \tilde{P})w = 0$, and $w = Pw = e(e^\top e)^{-1}e^\top w \in \text{range}(e)$. Taking $v^* = -(e^\top e)^{-1}e^\top w$ gives $w + ev^*$, i.e.,

$$Q\theta^* + d + A^\top u^* + ev^* = 0 \quad (\text{stationarity}).$$

Finally, (2.10) and Lemma 2.2 yield $A\theta^* - b \leq 0, u^* \geq 0$ (dual feasibility), and $u^{*T}(A\theta - b) = 0$ (complementarity slackness). $\square$

This residual formulation provides the foundation for our gradient-based neural network approach.

For small positive ε and $y = (\theta^\top, u^\top)^\top \in \mathbb{R}^{6n}$, let us define the residual mapping

$$\mathcal{R}_\varepsilon(y) = \begin{bmatrix} (I - \tilde{P})(Q\theta + d + A^\top u) + \tilde{Q}(e^\top \theta) \\ \phi_{\text{FB}}^{\varepsilon}(u, b - A\theta) \end{bmatrix}, \quad (2.11)$$

where $\tilde{P}$ and $\tilde{Q}$ are defined as in the previous section.

With the residual system formulation, we now proceed to analyze its fundamental properties and convergence behavior.

3. CONVERGENCE ANALYSIS

This section analyzes the convergence behavior of the proposed neurodynamic system by verifying that its equilibrium points correspond exactly to the KKT points of the SSVR problem and establishing global stability via Lyapunov theory.

Theorem 3.1. *For small positive ε and a point $y^* = (\theta^{*T}, u^{*T})^\top$, $\mathcal{R}_\varepsilon(y^*) = 0$ if and only if θ^* is a KKT point of (2.3)–(2.4).*

Proof. Suppose $\mathcal{R}_\varepsilon(y^*) = 0$. Then, by definition of the residual mapping, we have:

$$(I - \tilde{P})(Q\theta^* + d + A^\top u^*) + \tilde{Q}(e^\top \theta^*) = 0, \quad (3.1)$$

$$\phi_{\text{FB}}^{\varepsilon_{\text{FB}}}(u^*, b - A\theta^*) = 0. \quad (3.2)$$

Equation (3.2) and Lemma 2.2 imply that $u^* \geq 0$, $b - A\theta^* \geq 0$, and $u^{*T}(A\theta^* - b) = 0$. Equation (3.1) together with Lemma 2.3 implies that θ^* satisfies the KKT conditions. Therefore, θ^* is a KKT point of (2.3)–(2.4).

Conversely, assume that θ^* is a KKT point of (2.3)–(2.4). Then, by Lemma 2.1, there exists a Lagrange multiplier u^* such that the KKT conditions hold. Lemma 2.3 then implies that $\phi_{\text{FB}}^{\varepsilon_{\text{FB}}}(u^*, b - A\theta^*) = 0$. Moreover, the stationarity condition from the KKT system ensures that equation (3.1) is satisfied. Consequently, $\mathcal{R}_\varepsilon(y^*) = 0$. $\square$

We next establish the analytical smoothness of $\mathcal{R}_\varepsilon$ required for continuous-time optimization.

Theorem 3.2 (Smoothness and local Lipschitz continuity). *$\mathcal{R}_\varepsilon : \mathbb{R}^{6n} \rightarrow \mathbb{R}^{6n}$ is continuously differentiable, and therefore locally Lipschitz.*

Proof. To show that the function $\mathcal{R}_\varepsilon$ is continuously differentiable, we check its two components separately. The first component, $(I - \tilde{P})(Q\theta^* + d + A^\top u^*) + \tilde{Q}(e^\top \theta^*)$, is an affine function in $y = (\theta^\top, u^\top)^\top$, therefore continuously differentiable. The second component, $\phi_{\text{FB}}^{\varepsilon_{\text{FB}}}(u^*, b - A\theta^*)$, uses the smoothed FB function applied componentwise to each pair $(u_i, (b - A\theta)_i)$. The scalar function $\phi_{\text{FB}}^{\varepsilon_{\text{FB}}}(a, b) = a + b - \sqrt{a^2 + b^2 + \varepsilon}$ is continuously differentiable with gradient

$$\nabla \phi_{\text{FB}}^{\varepsilon_{\text{FB}}}(a, b) = \left[1 - \frac{a}{\sqrt{a^2 + b^2 + \varepsilon}}, 1 - \frac{b}{\sqrt{a^2 + b^2 + \varepsilon}} \right].$$

Since $\varepsilon > 0$, the denominator $\sqrt{a^2 + b^2 + \varepsilon}$ is always positive, so the gradient is well-defined and continuous. Therefore, the function $\phi_{\text{FB}}^{\varepsilon_{\text{FB}}}(u, b - A\theta)$, as a composition of continuously differentiable functions, is continuously differentiable on $\mathbb{R}^{6n}$. Since $\mathcal{R}_\varepsilon$ is continuously differentiable, by a standard analysis fact [24], it is locally Lipschitz continuous. $\square$

Building upon these analytical properties, we now introduce the neural network dynamics for solving the optimization problem. Let us consider the merit function

$$E_\varepsilon(y) = \frac{1}{2} \|\mathcal{R}_\varepsilon(y)\|^2, \quad (3.3)$$

and the Jacobian

$$J_\varepsilon(y) = \nabla \mathcal{R}_\varepsilon(y) \in \mathbb{R}^{6n \times 6n},$$

so that $\nabla E_\varepsilon(y) = J_\varepsilon(y)^\top \mathcal{R}_\varepsilon(y)$. Let $y(t) = (\theta(t), u(t))$ be a time-dependent variable. Our aim is to design a neural network that evolves to an equilibrium point of E_ε . Applying steepest descent to (3.3) gives the continuous-time dynamics for solving the convex problem (2.3)–(2.4):

$$\frac{dy(t)}{dt} = -\Gamma \nabla E_\varepsilon(y(t)) = -\Gamma J_\varepsilon(y(t))^\top \mathcal{R}_\varepsilon(y(t)), \quad (3.4)$$

with initial condition

$$y(0) = y_0, \quad (3.5)$$

where $\Gamma \succ 0$ is a positive definite scaling factor. A convenient choice is the block diagonal matrix

$$\Gamma = \begin{bmatrix} \gamma_\theta I_{2n \times 2n} & O_{2n \times 4n} \\ O_{4n \times 2n} & \gamma_u I_{4n \times 4n} \end{bmatrix}, \quad \gamma_\theta, \gamma_u > 0,$$

which reduces to the scalar step κI when $\gamma_\theta = \gamma_u = \kappa$.

A vector $y^* \in \mathbb{R}^{6n}$ is an equilibrium point of the following neural dynamics

$$\frac{dy(t)}{dt} = -\kappa \nabla E_\varepsilon(y(t)), \quad \kappa > 0, \quad E_\varepsilon(y) = \frac{1}{2} \|\mathcal{R}_\varepsilon(y)\|^2,$$

if the trajectory starting at y^* is constant, i.e.,

$$\frac{dy^*}{dt} = 0 \iff \nabla E_\varepsilon(y^*) = 0,$$

where $y = (\theta^\top, u^\top)^\top$.

We now establish the fundamental existence and uniqueness properties of the neural network dynamics.

Theorem 3.3. *The model (3.4) with any initial point $y(t_0) = (x(t_0)^\top, u(t_0)^\top)^\top$ has a unique solution $y(t) = (x(t)^\top, u(t)^\top)^\top$ on $[t_0, T)$.*

Proof. By Theorem 3.2, the mapping $\mathcal{R}_\varepsilon : \mathbb{R}^{6n} \rightarrow \mathbb{R}^{6n}$ is continuously differentiable and hence locally Lipschitz continuous. According to the Local Existence and Uniqueness Theorem, there exists a unique solution in some interval $[t_0, T)$ $\square$

Remark 3.1. Let $[t_0, T)$ denote the maximal interval of existence. We will establish in the proof of Theorem 3.6 that the trajectory $y(t)$ remains bounded on all $t \geq 0$, which implies the solution can be extended indefinitely, i.e., $T \rightarrow \infty$.

Theorem 3.4 (Nonsingularity of J_ε). *Let $\varepsilon > 0$ and $y = (\theta^T, u^T)^\top$ be a feasible point of (2.3)–(2.4). Then $J_\varepsilon(y)$ is nonsingular.*

Proof. The Jacobian evaluated at y is the block matrix

$$J_\varepsilon(y) = \begin{bmatrix} (I - \tilde{P})Q + \tilde{Q}e^\top & (I - \tilde{P})A^\top \\ -D_1A & D_2 \end{bmatrix},$$

where $D_1, D_2 \in \mathbb{R}^{4n \times 4n}$ are diagonal matrices:

$$D_1 = \text{Diag} \left(1 - \frac{[b - A\theta]_i}{\sqrt{(u_i)^2 + ([b - A\theta]_i)^2 + \varepsilon}} \right), \quad D_2 = \text{Diag} \left(1 - \frac{u_i}{\sqrt{(u_i)^2 + ([b - A\theta]_i)^2 + \varepsilon}} \right).$$

Since $\varepsilon > 0$ and $u_i, [b - A\theta]_i \geq 0$ at a feasible point, we have $(D_1)_{i,i} > 0$, and $(D_2)_{i,i} > 0$ for all i . Now, consider the Schur complement of D_2 in $J_\varepsilon(y)$:

$$\begin{aligned} \mathcal{S} &= \left[(I - \tilde{P})Q + \tilde{Q}e^\top \right] - \left[(I - \tilde{P})A^\top \right] D_2^{-1} [-D_1A] \\ &= (I - \tilde{P})Q + \tilde{Q}e^\top + (I - \tilde{P})A^\top D_2^{-1} D_1A. \end{aligned}$$

In view of $\tilde{Q}e^\top = \tilde{P}$, we obtain $\mathcal{S} = \tilde{P} + (I - \tilde{P})(Q + A^\top D_2^{-1} D_1A)$. The matrix $Q + A^\top D_2^{-1} D_1A$ is positive semidefinite since Q is positive semidefinite (as a kernel matrix) and $A^\top D_2^{-1} D_1A$ is positive semidefinite (as D_1, D_2 are positive diagonal matrices). Moreover, $\tilde{P}$ is the projection

onto the range of e , and $I - \tilde{P}$ is the projection onto the orthogonal complement. The combined matrix $\mathcal{S}$ is therefore positive definite, making it invertible. $\square$

With these foundational results in place, we can now characterize the relationship between equilibrium points and optimal solutions.

Theorem 3.5. *If $y^* = (\theta^{*T}, u^{*T})^\top$ and θ^* be a KKT point of (2.3)–(2.4), then y^* is an equilibrium of the neural network (3.4)–(3.5). Conversely, if $y^* = (\theta^{*T}, u^{*T})^\top$ is an equilibrium point of (3.4)–(3.5), then θ^* is a KKT point of (2.3)–(2.4).*

Proof. Let θ^* be an optimal solution of (2.3)–(2.4). From Theorem 3.1, $\mathcal{R}_\varepsilon(y^*) = 0$, so $E_\varepsilon(y^*) = 0$. Consequently, $\nabla E_\varepsilon(y^*) = J_\varepsilon(y^*)^\top \mathcal{R}_\varepsilon(y^*) = 0$, i.e., y^* is an equilibrium point of neural network dynamics (3.4)–(3.5). Conversely, by Theorem 3.4, the Jacobian matrix $J_\varepsilon(y^*)$ is nonsingular, so the homogeneous system $0 = J_\varepsilon(y^*)^\top \mathcal{R}_\varepsilon(y^*)$ has only the trivial solution $\mathcal{R}_\varepsilon(y^*) = 0$. Therefore, θ^* is a KKT point of (2.3)–(2.4). $\square$

Finally, we establish the global stability of the system using a Lyapunov candidate.

Theorem 3.6. *Let y^* be an equilibrium point of the neural network (3.4)–(3.5). Then y^* is locally stable in the sense of Lyapunov. In fact, we can extend this result to y^* , globally stable in the sense of Lyapunov.*

Proof. Consider the exponential Lyapunov function $\tilde{E}(y) = \exp\left(\frac{1}{2}\|y - y^*\|^2\right) - 1$. Then $\tilde{E}(y) > 0$ for every $y \neq y^*$, $\tilde{E}(y) = 0$ if and only if $y = y^*$, and $\tilde{E}$ is radially unbounded. Moreover,

$$\nabla \tilde{E}(y) = \exp\left(\frac{1}{2}\|y - y^*\|^2\right) (y - y^*).$$

Along trajectories of (3.4),

$$\dot{\tilde{E}}(y) = \nabla \tilde{E}(y)^\top \dot{y} = -\exp\left(\frac{1}{2}\|y - y^*\|^2\right) (y - y^*)^\top \Gamma J_\varepsilon(y)^\top \mathcal{R}_\varepsilon(y) \quad (3.6)$$

where $\Gamma \succ 0$.

The rest of the proof will proceed in three steps: (1) locally expand the solution derivative, (2) show the negativity of the derivative near the equilibrium point to conclude that it is locally Lyapunov stable, and (3) extend the result to global stability.

Step 1: Local Expansion Around the Equilibrium. To prove that y^* is locally Lyapunov stable, we will first express the local expansion around the equilibrium point. By continuity of J_ε , nonsingularity of $J_\varepsilon(y^*)$ and the Inverse Function Theorem, the following Taylor expansion holds in a neighborhood of y^*

$$\mathcal{R}_\varepsilon(y) = \mathcal{R}_\varepsilon(y^*) + J_\varepsilon(y^*)(y - y^*) + o(\|y - y^*\|) = J_\varepsilon(y^*)(y - y^*) + o(\|y - y^*\|),$$

where $o(\cdot)$ is the little-o notation, and $J_\varepsilon(y) = J_\varepsilon(y^*) + o(1)$ as $y \rightarrow y^*$. Hence substituting these equations into 3.6 gives

$$\dot{\tilde{E}}(y) = -\exp\left(\frac{1}{2}\|y - y^*\|^2\right) \left[(y - y^*)^\top \Gamma J_\varepsilon(y)^\top J_\varepsilon(y^*)(y - y^*) + o(\|y - y^*\|^2) \right].$$

Step 2: Negativity of $\dot{\tilde{E}}$ near y^ .* Next, we proceed to show the negativity of the derivative $\dot{\tilde{E}}$ around y^* . $Q^{y^*}(y) := (y - y^*)^\top \Gamma J_\varepsilon(y)^\top J_\varepsilon(y^*)(y - y^*)$ is a quadratic form describing the local curvature of $\tilde{E}$. Because $\Gamma \succ 0$ and $J_\varepsilon(y^*)$ is nonsingular, we have $J_\varepsilon(y^*)^\top J_\varepsilon(y^*) \succ 0$, hence

$Q^{y^*}(y) > 0$ for all $y \neq y^*$. Let $\lambda_{\min} := \lambda_{\min}(\Gamma) > 0$ and $\sigma_{\min} := \sigma_{\min}(J_{\mathcal{E}}(y^*)) > 0$. Then, for all $v \in \mathbb{R}^{6n}$, $v^\top \Gamma J_{\mathcal{E}}(y^*)^\top J_{\mathcal{E}}(y^*) v \geq \lambda_{\min} \sigma_{\min}^2 \|v\|^2$. Letting $v = y - y^*$, we have

$$(y - y^*) \Gamma J_{\mathcal{E}}(y^*)^\top J_{\mathcal{E}}(y^*) (y - y^*) = Q^{y^*}(y) \geq \lambda_{\min} \sigma_{\min}^2 \|y - y^*\|^2.$$

Therefore, there exist $c \geq 0$ and a neighborhood Ω^* of y^* such that

$$(y - y^*) \Gamma J_{\mathcal{E}}(y^*)^\top \mathcal{R}_{\mathcal{E}}(y) \geq c \|y - y^*\|^2, \quad \forall y \in \Omega^*.$$

Substituting into (3.4) we obtain, for $y \in \Omega^* \setminus \{y^*\}$,

$$\dot{\tilde{E}}(y) \leq -c \exp\left(\frac{1}{2} \|y - y^*\|^2\right) \|y - y^*\|^2 \leq 0.$$

Thus $\tilde{E}$ is a Lyapunov function on Ω^* . By Lyapunov's direct method, y^* is Lyapunov stable. Moreover, if the equilibrium point, y^* , is unique, $\tilde{E}$ is a strict Lyapunov function, and y^* is locally asymptotically stable.

Step 3: Global Boundedness and Extendability. Moreover, the Lyapunov function $\tilde{E}(y)$ is nonincreasing along all trajectories of the system. Hence, for any $t_0 \leq t \leq T$, it follows that

$$\frac{1}{2} \|y(t) - y^*\|^2 \leq \tilde{E}(y(t)) \leq \tilde{E}(y(t_0)). \quad (3.7)$$

By the Maximal Interval of Existence Theorem, inequality (3.7) ensures the trajectory of the neural network (3.4) and (3.5) remains bounded for all $t \geq t_0$. Otherwise, there exists some T such that $\|\tilde{E}(y(t))\| \rightarrow \infty$ as $t \rightarrow T$ which contradicts its boundedness. Consequently, the solution is globally expandable, i.e., $T = +\infty$. $\square$

Theorem 3.7. *The solution trajectory of the neural network (3.4) and (3.5) is globally convergent to an optimal point of the optimization problem (2.3) and (2.4).*

Proof. The proof proceeds in four main steps: (1) establishing the existence of a convergent subsequence using a Lyapunov function, (2) applying LaSalle's invariance principle to characterize the limit set, (3) showing that this invariant set consists solely of equilibrium points, and (4) proving that the trajectory converges to a single equilibrium.

Step 1: Boundedness and Existence of a Limit Point. Given a solution trajectory of the neural network $y(t) = (\theta(t)^\top, u(t)^\top)^\top$ with initial condition $y(t_0)$ and an equilibrium point $y^* = (\theta^{*T}, u^{*T})$, consider the Lyapunov function candidate:

$$\tilde{E}(y) = \exp\left(\frac{1}{2} \|y - y^*\|^2\right) - 1.$$

The function is radially unbounded and positive definite with respect to the equilibrium point y^* . Its time derivative along the trajectories of the system is nonincreasing. Consequently, the level set

$$L(y_0) = \{y \in \mathbb{R}^{6n} | \tilde{E}(y) \leq \tilde{E}(y(t_0))\}$$

which constitutes a closed positively invariant set for the solution $y(t)$ is bounded, and therefore, compact [36]. Hence, the trajectory $y(t)$ remains in the bounded region $L(y_0)$ for all $t \geq t_0$. Therefore, by the boundedness of the trajectory, we can construct a convergent subsequence $(y(t_k))_{k=1}^\infty$, such that

$$\lim_{k \rightarrow \infty} y(t_k) = \bar{y} \quad (3.8)$$

for some limit point $\bar{y} = (\bar{\theta}^\top, \bar{u}^\top)^\top$.

Step 2: Application of LaSalle's Invariance Principle. Since $\dot{\tilde{E}}(y)$ is nonincreasing along trajectories and bounded below, we have $\dot{\tilde{E}}(y) \rightarrow 0$ as $t \rightarrow \infty$. Thus, the limit point $\bar{y}$ is an ω -limit point of $y(t)$. By LaSalle's invariance principle, the trajectory $y(t)$ converges to the largest invariant subset M contained in

$$\Lambda = \{y \in \mathbb{R}^{6n} | \dot{\tilde{E}}(y) = 0\},$$

and since the entire ω -limit set is contained within the largest invariant set M , $\bar{y} \in M$.

Step 3: Characterization of the Invariant Set. From (3.4) and (3.5),

$$\dot{y}(t) = -\Gamma J_{\varepsilon}(y(t))^{\top} \mathcal{R}_{\varepsilon}(y(t))$$

for some $\Gamma \succ 0$. Substituting this into the derivative of the Lyapunov function $\tilde{E}(y)$ yields

$$\dot{\tilde{E}}(y) = \exp\left(\frac{1}{2}\|y - y^*\|^2\right) (y - y^*)^{\top} \Gamma J_{\varepsilon}(y)^{\top} \mathcal{R}_{\varepsilon}(y).$$

Since the exponential term is always positive, $\dot{\tilde{E}}(y) = 0$ if and only if $(y - y^*)^{\top} \Gamma J_{\varepsilon}(y)^{\top} \mathcal{R}_{\varepsilon}(y) = 0$. Because $\Gamma \succ 0$, both $J_{\varepsilon}(y)$ and $\mathcal{R}_{\varepsilon}(y)$ are continuously differentiable and $J_{\varepsilon}(y)$ is nonsingular, the equality above only holds when $\mathcal{R}_{\varepsilon}(y) = 0$. Thus,

$$\mathcal{R}_{\varepsilon}(y) = 0 \iff \nabla E_{\varepsilon}(y) = 0 \iff \dot{y} = 0$$

since $\dot{y} = -\Gamma J_{\varepsilon}(y)^{\top} \mathcal{R}_{\varepsilon}(y)$. This implies that the invariant set $M \subseteq \Lambda$ consists entirely of equilibrium points, and therefore, $\bar{y}$ is an equilibrium point.

Step 4: Convergence to an Equilibrium. To prove the trajectory's convergence toward the equilibrium point $\bar{y}$ from Step 1, one considers the Lyapunov function centered at $\bar{y}$:

$$\tilde{\tilde{E}}(y) = \exp\left(\frac{1}{2}\|y - \bar{y}\|^2\right) - 1.$$

Clearly, this function is also radially unbounded and positive definite with respect to $\bar{y}$. Its time derivative along the system trajectories is

$$\dot{\tilde{\tilde{E}}}(y) = \dot{\tilde{E}}(y)^{\top} \dot{y} = \exp\left(\frac{1}{2}\|y - \bar{y}\|^2\right) (y - \bar{y})^{\top} \Gamma J_{\varepsilon}(y)^{\top} \mathcal{R}_{\varepsilon}(y).$$

This derivative is continuous and, by the same logic above, is nonpositive. Hence $\dot{\tilde{\tilde{E}}}(y) \leq 0$ for all $t \geq 0$. From (3.8), $\lim_{k \rightarrow \infty} y(t_k) = \bar{y}$ implies $\lim_{k \rightarrow \infty} \tilde{\tilde{E}}(y(t_k)) = 0$. Thus, for any $\varepsilon > 0$, there exists $t_q > 0$ such that, for all $t \geq t_q$,

$$\tilde{\tilde{E}}(y(t)) < \varepsilon \implies \frac{1}{2}\|y(t) - \bar{y}\|^2 \leq \tilde{\tilde{E}}(y(t)) \leq \varepsilon.$$

Therefore, $\lim_{t \rightarrow \infty} \|y(t) - \bar{y}\| = 0$, i.e., $y(t) \rightarrow \bar{y}$. Hence, the solution trajectory $y(t)$ of the neural network (3.4) and (3.5) is globally convergent to the equilibrium point $\bar{y}$, and the corresponding $\bar{\theta}$ is an optimal solution to the optimization problem (2.3) and (2.4). $\square$

Corollary 3.1. *Assume that equilibrium point y^* is unique. Then y^* is globally asymptotically stable, and the trajectory satisfies $y(t) \rightarrow y^*$ as $t \rightarrow \infty$.*

Proof. From Theorem 3.7, we see that every trajectory $y(t)$ of the neural network globally converges to some equilibrium point. Under the assumption that the equilibrium point y^* is unique, the equilibrium point must be y^* . Therefore,

$$\lim_{t \rightarrow \infty} \|y(t) - y^*\| = 0, \text{ i.e., } y(t) \rightarrow y^*.$$

Global asymptotic stability follows directly from this convergence property. From the Lyapunov analysis in the proof of the Theorem 3.7, the function

$$\tilde{E}(y) = \exp\left(\frac{1}{2}\|y - y^*\|^2\right) - 1$$

is radially unbounded and positive definite with respect to y^* , and its derivative along the system trajectories, $\dot{\tilde{E}}(y)$, is nonpositive. By Lyapunov's direct method and LaSalle's invariance principle, we conclude that the equilibrium y^* is globally asymptotically stable. $\square$

This completes the convergence analysis, establishing that the proposed neural network dynamics not only converge to optimal solutions but do so with strong stability guarantees.

4. COMPARISON WITH SOME EXISTING NEURAL NETWORK DYNAMICS

Two neurodynamic ancestors of the probabilistic SVR problem are particularly relevant to the present work, both reviewed in [16]. The *Lagrangian-type* network of Nazemi [31] treats the probabilistic QP as a saddle-point problem and integrates the primal-dual flow

$$\begin{cases} \frac{d\theta}{dt} = -\left(Q\theta + d + \frac{1}{2}A^\top(u \odot u) + ev\right), \\ \frac{du}{dt} = u \odot (A\theta - b), \quad \frac{dv}{dt} = e^\top \theta, \end{cases} \quad (4.1)$$

with state $(\theta, u, v) \in \mathbb{R}^{4n} \times \mathbb{R}^{4n} \times \mathbb{R}$ (total $8n + 1$ neurons), where $u \odot u$ acts as a squared slack to enforce $u \geq 0$ implicitly. In a complementary direction, the *smoothed Fischer–Burmeister gradient* network of Nazemi and Nazemi [34] casts the KKT system into a residual $\rho(y) = 0$ and performs steepest descent on $E_2(y) = \frac{1}{2}\|\rho(y)\|^2$, where

$$\rho(y) = \begin{bmatrix} Q\theta + d + ev + A^\top u \\ -e^\top \theta \\ \phi_{\text{FB}}^\varepsilon(u, b - A\theta) \end{bmatrix}, \quad \phi_{\text{FB}}^\varepsilon(a, b) = (a + b) - \sqrt{a^2 + b^2 + \varepsilon}, \quad (4.2)$$

again requiring $8n + 1$ neurons.

Building on these, Feizi et al. [16] proposed a recurrent neural network (RNN) that achieves a slight reduction to $8n$ neurons by omitting the scalar multiplier v :

$$\begin{cases} \frac{d\theta}{dt} = \eta \left\{ -(I - \tilde{P})[Q\theta + d + A^\top(u + A\theta - b)^+] - \tilde{Q}(e^\top \theta) \right\}, \\ \frac{du}{dt} = \eta \left\{ -u + (u + A\theta - b)^+ \right\}. \end{cases} \quad (4.3)$$

All three prior models solve the SSVR problem through the same high-dimensional matrices (Q, d, e, A, b) ,

$$\begin{aligned}\theta &= [\alpha_1 - \alpha'_1, \dots, \alpha_n - \alpha'_n, \alpha_1 + \alpha'_1, \dots, \alpha_n + \alpha'_n, \\ &\quad \alpha_1 - \alpha'_1, \dots, \alpha_n - \alpha'_n, \alpha_1 + \alpha'_1, \dots, \alpha_n + \alpha'_n]^\top \in \mathbb{R}^{4n}, \\ K_{i,j} &= E(\phi(X_i))^\top E(\phi(X_j)), \\ Q &= \begin{bmatrix} K & O_{n \times 3n} \\ O_{3n \times n} & O_{3n \times 3n} \end{bmatrix} \in \mathbb{R}^{4n \times 4n}, \\ d &= [-E(Y)^\top, (-2a\delta - \varepsilon)\mathbf{1}_n^\top, \mathbf{0}_{2n}^\top]^\top \in \mathbb{R}^{4n}, \\ e &= [\mathbf{1}_n^\top, \mathbf{0}_{3n}^\top]^\top \in \mathbb{R}^{4n}, \\ A &= \begin{bmatrix} I_{2n \times 2n} & O_{2n \times 2n} \\ O_{2n \times 2n} & -I_{2n \times 2n} \end{bmatrix} \in \mathbb{R}^{4n \times 4n}, \\ b &= [C\mathbf{1}_n^\top, 2C\mathbf{1}_n^\top, C\mathbf{1}_n^\top, \mathbf{0}_n^\top]^\top \in \mathbb{R}^{4n}.\end{aligned}$$

While effective, these architectures leave two concrete gaps: (i) the state-space dimensionality is unnecessarily large ($8n+1$ or $8n$ neurons), and (ii) the dynamics either involve a non-smooth projection $(\cdot)^+$ [16] or embed the smoothing into larger Lagrangian systems [31, 34] carrying redundant equality-multiplier variables. The proposed $6n$ neurodynamic architecture addresses these gaps across three key aspects.

- (1) **Reduced State-Space Complexity:** A primary contribution of this work is the significant reduction in network dimensionality. By reformulating the SSVR problem without the redundant $4n$ structure seen in prior matrices, our architecture achieves a 25% reduction over [16] and a nearly 26% reduction over [31, 34], requiring only $6n$ neurons ($\theta \in \mathbb{R}^{2n}, u \in \mathbb{R}^{4n}$). This is a critical advancement for large-scale applications where memory and hardware constraints are significant.
- (2) **Regularity and Differentiability:** A fundamental distinction lies in the regularity of the residual mapping. The dynamics in [16] rely on a non-differentiable projection operator, $(x)^+ = \max\{0, x\}$, which can induce numerical oscillations. In contrast, our model utilizes the smoothed Fischer–Burmeister function $\phi_{\text{FB}}^{\text{EFB}}$, ensuring the residual mapping is C^1 -continuous. This provides enhanced numerical stability and allows for analysis as a gradient-based descent.
- (3) **Convergence and Stability:** Our approach explicitly defines a merit function $E_\varepsilon(y) = \frac{1}{2} \|\mathcal{R}_\varepsilon(y)\|^2$, following a steepest descent trajectory. This formulation ensures that equilibrium points correspond precisely to the KKT points of the original SSVR problem, avoiding the penalty parameters of earlier Lagrangian methods [23] or the restrictive monotonicity assumptions of projection-based models [17].

5. NUMERICAL SIMULATIONS

This section provides empirical validation of the proposed neurodynamic framework. We evaluate the system’s convergence properties on synthetic data and its predictive accuracy on UCI benchmark datasets using both linear and RBF kernels.

5.1. Convergence and Stability. To verify the global convergence and Lyapunov stability established by Theorem 3.6 and Theorem 3.7, we simulate the proposed dynamics on two synthetic regression tasks designed to stress-test different aspects of the SSVR formulation. Throughout, we fix the hyperparameters $C = 1.0$, $\varepsilon = 0.1$, $a = 0.2$, $\delta = 0.5$, $\gamma_\theta = \gamma_u = 400$ and $\varepsilon_{\text{FB}} = 10^{-3}$, and integrate the dynamics with the stiff BDF solver. The synthetic convergence experiments use an extended horizon $t \in [0, 0.3]$ so that the stability plateau is clearly visible after the initial transient; the UCI benchmarks in Section 5.2 retain the shorter horizon $t \in [0, 0.1]$ used by all baselines for a fair comparison.

Case A – linear kernel with stochastic inputs. The observed feature is $x_{\text{obs}} = x_{\text{true}} + \delta_{\text{data}}\xi$ with $\xi \sim \mathcal{N}(0, 1)$ and $\delta_{\text{data}} = 0.05$, while the response is $y = 2x_{\text{true}} + 1 + 0.15\eta$, $\eta \sim \mathcal{N}(0, 1)$; $n = 50$ training points are drawn on $x_{\text{true}} \in [-1, 1]$. Four random initial conditions are used. Figure 1 traces the merit function and the primal trajectories $(w(t), b(t))$; all four initial conditions drive the merit E_ε down from $\sim 10^2$ by four orders of magnitude to $\approx 1.3 \times 10^{-2}$ and, crucially, *plateau*: the primal pair $(w(t), b(t))$ stops moving once the merit stabilises, confirming the Lyapunov-stable equilibrium predicted by Theorem 3.6. All four trajectories reach numerically identical final values $(w, b) \approx (1.80, 0.98)$, confirming the initialisation-independence predicted by Theorem 3.7. The small residual bias with respect to the ground truth $(w^*, b^*) = (2, 1)$ is the characteristic shrinkage of the ε -insensitive loss and is not a symptom of incomplete convergence.

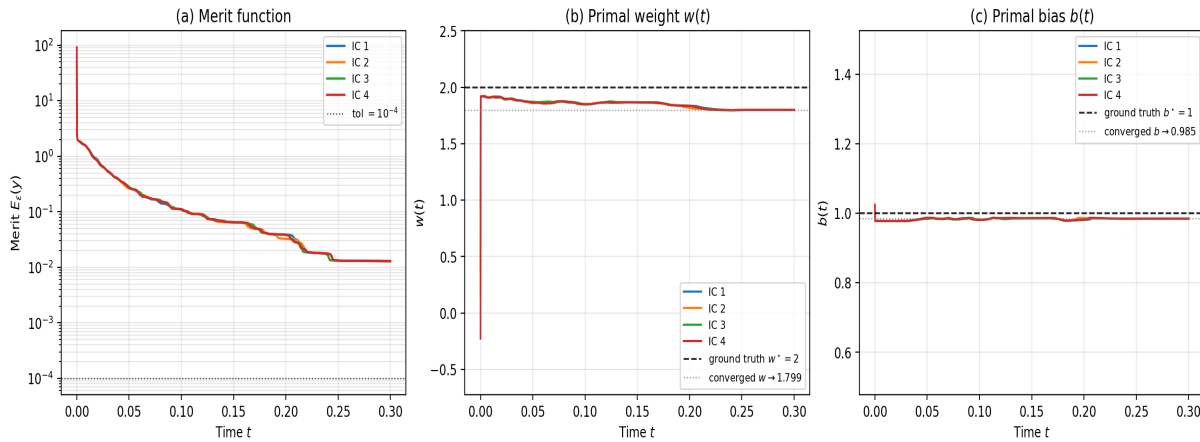


FIGURE 1. Case A – convergence of the primal variables from four random initial conditions. (a) Merit function $E_\varepsilon(y)$ on a logarithmic scale; the dotted line marks the target tolerance 10^{-4} , and the four curves plateau near 1.3×10^{-2} , the residual level set by the finite Fischer–Burmeister smoothing. (b)–(c) Primal weight $w(t)$ and bias $b(t)$ on a linear y-axis covering the full transient range. The ground-truth values $w^* = 2$ and $b^* = 1$ are shown as black dashed lines and the shared converged values are shown as grey dotted lines; the small gap between them is the expected ε -insensitive shrinkage. All four initial conditions settle at indistinguishable final values well before $t = 0.3$, demonstrating the Lyapunov-stable plateau predicted by Theorem 3.6.

Figure 2 visualises the internal structure of the converged solution. Panels (a)–(b) show the time evolution of the dual coefficients $\alpha_j(t)$ and $\alpha'_j(t)$ for the first initial condition, with each component coloured by its *final* magnitude: true support vectors emerge as bright yellow curves driven to the upper bound $C = 1$, while non-support components fade into dark purple near zero. Panel (c) overlays the converged linear fit, the ε -insensitive tube $[f(x) - \varepsilon, f(x) + \varepsilon]$, and the training data with support vectors highlighted in red. The geometric support-vector count (points whose residual lies on or outside the tube) agrees with the dual-level activation seen in panels (a)–(b), as the KKT conditions demand.

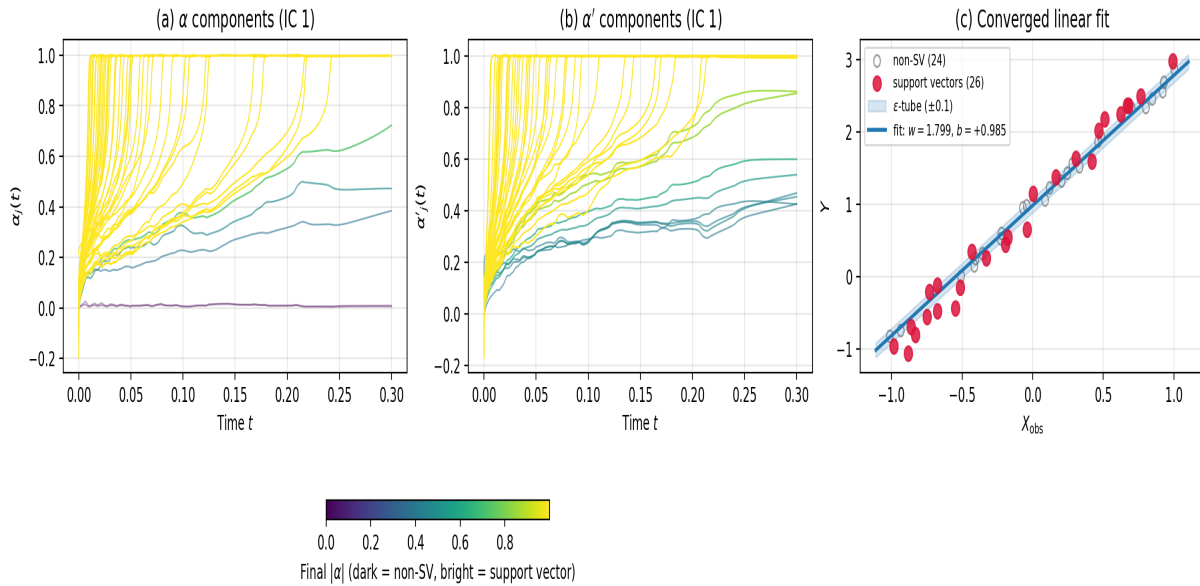


FIGURE 2. Case A – support-vector activation and converged fit. (a)–(b) Time evolution of the dual coefficients $\alpha_j(t)$ and $\alpha'_j(t)$ for the first initial condition, colour-coded by final magnitude $|\alpha_j|$ (viridis). Bright-yellow components are driven to the upper box constraint $C = 1$ and correspond to bound support vectors; dark-purple components stay near zero and are inactive. (c) Converged linear fit, ε -tube (shaded), non-support training points (grey hollow) and support vectors (red filled). The ε -insensitive zone captures roughly half of the training set, matching the $\varepsilon/\sigma_y \approx 0.67$ ratio designed into the data generator.

Case B – RBF kernel, nonlinear target with outliers. To test the dynamics on a nonlinear kernel and in the presence of gross outliers, we generate $n = 80$ training points from $y = \sin(1.5x) + 0.1\eta$ on $x \in [-3, 3]$ and corrupt 5% of the responses with ± 2 shocks. An RBF kernel with bandwidth $\gamma = 1/d$ is used. Figure 3(a) shows that the merit function decreases monotonically by three orders of magnitude (from $\sim 10^1$ to $\approx 1.1 \times 10^{-2}$) and reaches a clear plateau; it does not converge to the 10^{-4} tolerance because the outliers produce an irreducible residual in the smoothed Fischer–Burmeister term, but the primal solution stabilises well before the end of the integration horizon. Figure 3(b) shows the converged RBF fit: the four outliers (orange crosses) sit well outside the ε -tube and are effectively down-weighted by the ε -insensitive loss, while the fit tracks the true $\sin(1.5x)$ curve to within the ε -tube on the clean

points. This confirms empirically that the proposed dynamics inherit the classical robustness of SVR against sparse gross outliers, while retaining the convergence guarantees of Theorem 3.7.

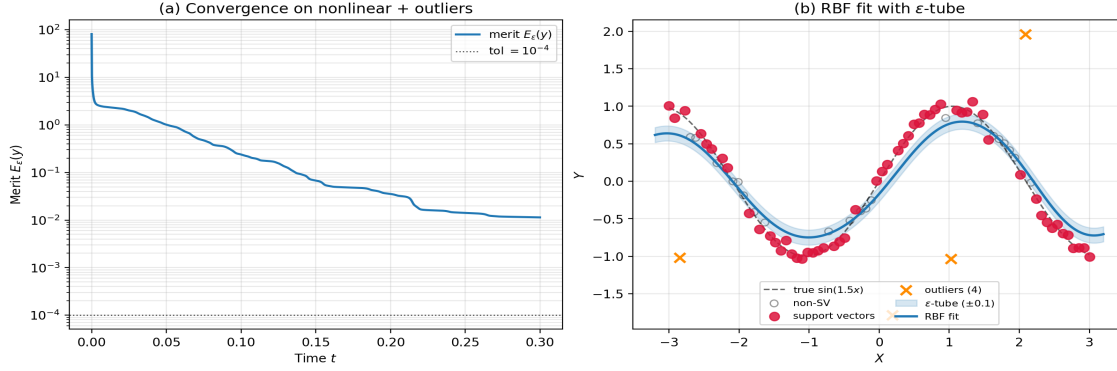


FIGURE 3. **Case B – RBF kernel on $y = \sin(1.5x)$ with 5% gross outliers.** (a) Merit function $E_\varepsilon(y)$ on a log scale; the flow decreases monotonically and stabilises even though four outlier responses create an irreducible residual. (b) Converged RBF fit (solid blue), true $\sin(1.5x)$ curve (dashed black), ε -tube (shaded), and the training data: non-support points in grey, support vectors in red, outliers in orange. The ε -insensitive loss isolates the outliers without distorting the fit on the clean data.

5.2. Predictive Performance.

Datasets. The predictive efficacy of the proposed SSVR is evaluated on ten regression benchmarks from the UCI Machine Learning Repository [3] – Boston Housing (506 samples, 13 features), Real Estate Valuation (414 samples, 6 features), a subset of California Housing (400 random samples, 8 features), Concrete Compressive Strength (1,030 samples, 8 features), Energy Efficiency (768 samples, 8 features, target Y_1), and Auto MPG (392 samples after NA drop, 7 features), NASA Airfoil Self-Noise (1,503 samples, 5 features; acoustic measurement noise), Yacht Hydrodynamics (308 samples, 6 features; physical simulation), Wine Quality Red (1,599 samples, 11 physicochemical features, sensory target with human-scoring noise) and Abalone (4,177 samples, 10 features after one-hot encoding of “sex”; biometric ring-count target); we abbreviate them as “Boston”, “REV”, “CA”, “CCS”, “EE”, “AMPG”, “Airfoil”, “Yacht”, “WineRed” and “Abalone” respectively.

Models compared. We compare the proposed **Ours** model against the three neurodynamic SSVR baselines reviewed in Section 1: the projection recurrent network of Feizi et al. [16] ($8n$ neurons, equation (4.3)), the Lagrangian-type network of Nazemi [31] ($8n+1$ neurons, equation (4.1)), and the smoothed Fischer–Burmeister gradient network of Nazemi & Nazemi [34] ($8n+1$ neurons, equation (4.2)). All four models solve the same SSVR reformulation using the same matrices (Q, d, e, A, b) ; only the neurodynamic map on top differs.

Experimental protocol. For every (dataset, model) pair we repeat the following procedure and report the mean and standard deviation of each metric:

- (1) *Split.* An 80/20 train/test partition is drawn independently for each seed.

- (2) *Scaling*. Features are standardized on the training split only. The target is likewise centered and scaled so that the probabilistic margin ε is applied on comparable units across datasets.
- (3) *Training*. Every model is fitted on the standardized training set by integrating its ODE with the stiff BDF solver of `scipy.integrate.solve_ivp` on the time horizon $[0, 0.1]$, 1000 reporting points, $\text{rtol} = 10^{-3}$, $\text{atol} = 10^{-6}$.
- (4) *Evaluation*. Predictions are inverted to the original target scale before computing the metrics listed in Table 1.

TABLE 1. Main evaluation metrics reported in Tables 3–4. Here $\hat{y}_i$ denotes the model prediction on the i -th test sample and $\bar{y}$ is the test-set mean. Test-set metrics are reported throughout the paper; train-set R^2 /RMSE are also reported to flag overfitting. All statistics are computed after inverting the target scaling.

Metric	Formula	Interpretation
R^2	$1 - \sum_i (y_i - \hat{y}_i)^2 / \sum_i (y_i - \bar{y})^2$	explained variance (higher is better)
RMSE	$\sqrt{\frac{1}{n} \sum_i (y_i - \hat{y}_i)^2}$	squared-loss error (lower is better)
MAE	$\frac{1}{n} \sum_i y_i - \hat{y}_i $	absolute error, robust to outliers (lower is better)
#SV	$ \{i : \alpha_i - \alpha'_i > 10^{-3}\} $	active support vectors at the terminal state
CPU	wall-clock fit time	end-to-end training cost in seconds

TABLE 2. Hyperparameter settings shared across all four neurodynamic SSVR models for the UCI experiments (Tables 3–4). The same values are applied across all datasets for a fair comparison.

Symbol	Value	Role / consumed by
C	1.0	slack / regularization trade-off (all four models)
ε	0.1	ε -insensitive tube (all four models)
a	0.2	probabilistic margin multiplier (all four models)
δ	0.5	noise-covariance bound (all four models)
ε_{FB}	10^{-3}	Fischer–Burmeister smoothing (Ours; Nazemi & Nazemi)
γ_θ, γ_u	200 or 400	step-size scaling of θ - and u -dynamics (Ours)
γ_θ	200 or 400	step-size scaling of θ -dynamics (Feizi, Nazemi, Nazemi & Nazemi)
$[t_0, t_f]$	$[0, 0.1]$	integration horizon
reporting points	1000	t_{eval} grid density
ODE solver	BDF (<code>scipy</code>)	stiff variable-order backward-differentiation
rtol, atol	$10^{-3}, 10^{-6}$	relative / absolute ODE tolerances
Split	80%/20%	train/test partition
Seeds	5 seeds	independent repetitions per (dataset, model)
Max samples	$n_{\text{max}} = 500$	uniform subsampling of datasets larger than 500

Results. Tables 3 (linear kernel) and 4 (RBF kernel, bandwidth $\gamma = 1/d$ after standardisation) report test R^2 , RMSE, MAE, the number of active support vectors #SV, and the CPU time

required to fit each model across all ten UCI benchmarks. The proposed SSVR achieves the best test R^2 and the lowest test RMSE/MAE on the majority of benchmarks, and does so at a fraction of the CPU cost of the $8n+1$ Lagrangian network [31]. The gap with the projection-based network of Feizi et al. [16] exceeds 15 percentage points of explained variance on CA and CCS. On the noisier Tier-A datasets (REV, WineRed, AMPG) the smoothed-FB gradient network of Nazemi and Nazemi [34] comes within one standard deviation of the proposed model on the test metrics, confirming empirically that the Fischer–Burmeister smoothing is the dominant ingredient and that the main novelty of the proposed method is the compact $6n$ state layout on top of it – a design choice that translates directly into the $2\text{--}4\times$ CPU-time advantage reported in Tables 3 and 4.

5.3. Statistical Analysis. The raw mean- $\pm$ -std numbers in Tables 3–4 describe central tendency but do not address whether the observed ranking of the four models is statistically significant. We therefore follow the non-parametric protocol recommended by Demšar [10] for comparing multiple learners across multiple datasets, which makes no assumption about the distribution of generalisation errors.

For every metric we pool the ten UCI benchmarks under both kernels into $N = 20$ independent problems, and rank the four models on each problem (rank 1 = best, ties handled by the average-rank rule). The Friedman test statistic

$$\chi_F^2 = \frac{12N}{k(k+1)} \left(\sum_{j=1}^k R_j^2 - \frac{k(k+1)^2}{4} \right)$$

with $k = 4$ models and R_j the mean rank of the j -th model is asymptotically χ_{k-1}^2 under H_0 (all models equivalent). When the omnibus test rejects H_0 , we apply the Holm step-down procedure to the pairwise z -statistics

$$z_j = (R_j - R_{\text{Ours}}) / \sqrt{k(k+1)/(6N)}$$

with the proposed compact $6n$ network designated as the control. For visual inspection we additionally plot the Nemenyi critical-difference (CD) diagram of Demšar, using the CD value

$$\text{CD} = q_\alpha \sqrt{k(k+1)/(6N)}$$

with $q_{0.05} = 2.569$ for $k = 4$, giving $\text{CD} = 1.05$ for $N = 20$.

Table 5 summarises the test statistics. For every metric the omnibus χ_F^2 strongly rejects H_0 (all $p_F < 10^{-7}$), so the mean-rank differences are genuine signal rather than sampling noise. The Holm-adjusted p -values then confirm that *Ours* is significantly better than the $8n$ projection network of Feizi et al. [16] and the $8n+1$ Lagrangian network of Nazemi [31] on every metric (all $p_{\text{Holm}} < 0.01$). Against the smoothed-FB predecessor of Nazemi and Nazemi [34]—which shares our complementarity formulation but carries $8n+1$ state variables—the improvement on pure accuracy (R^2 and MAE) narrowly fails to reach the $\alpha = 0.05$ threshold ($p_{\text{Holm}} = 0.066$ and 0.27), but becomes significant as soon as the criterion penalises either error dispersion (Test RMSE, $p_{\text{Holm}} = 0.037$) or computational cost (CPU time, $p_{\text{Holm}} = 0.027$). In other words, the two FB-based models are statistically indistinguishable in accuracy, and the proposed compact $6n$ dynamics translate directly into a significant cost advantage without sacrificing that accuracy.

TABLE 3. Predictive performance with a **linear kernel**, ten UCI datasets (subsampled to $n_{\max} = 500$), five independent 80/20 splits. Rows: the proposed model (**Ours**, $6n$), Feizi et al. [16] (**Feizi**, $8n$), the Lagrangian network of Nazemi [31] (**Nazemi**, $8n+1$), and the smoothed-FB gradient network of Nazemi and Nazemi [34] (**Nazemi-Nazemi**, $8n+1$). Best value per column within each dataset block is in bold; values are mean $\pm$ std.

Dataset	Model	R^2		RMSE		MAE (test)	#SV	CPU (s)
		Train	Test	Train	Test	Test		
Boston	Ours	0.7295 $\pm$ 0.0060	0.6468 $\pm$ 0.0363	4.7439 $\pm$ 0.0163	4.4393 $\pm$ 0.2073	2.8523 $\pm$ 0.0875	240.0000 $\pm$ 0.0000	1.1377 $\pm$ 0.0607
Boston	Feizi	0.6460 $\pm$ 0.0175	0.5952 $\pm$ 0.0265	5.4257 $\pm$ 0.1337	4.7609 $\pm$ 0.2392	3.2433 $\pm$ 0.1845	240.0000 $\pm$ 0.0000	4.9939 $\pm$ 0.6230
Boston	Nazemi	0.4786 $\pm$ 0.0913	0.4040 $\pm$ 0.0752	6.5620 $\pm$ 0.6244	5.7760 $\pm$ 0.5484	4.1370 $\pm$ 0.7494	240.0000 $\pm$ 0.0000	7.1992 $\pm$ 0.4980
Boston	Nazemi-Nazemi	0.7054 $\pm$ 0.0076	0.6396 $\pm$ 0.0206	4.9504 $\pm$ 0.0255	4.4938 $\pm$ 0.2407	2.9800 $\pm$ 0.1230	240.0000 $\pm$ 0.0000	2.6716 $\pm$ 0.1303
REV	Ours	0.5443 $\pm$ 0.0224	0.4951 $\pm$ 0.0685	9.0946 $\pm$ 0.5705	10.3023 $\pm$ 2.0383	7.3086 $\pm$ 0.3936	240.0000 $\pm$ 0.0000	0.9842 $\pm$ 0.1144
REV	Feizi	0.4768 $\pm$ 0.0340	0.4155 $\pm$ 0.0970	9.7442 $\pm$ 0.6635	11.0737 $\pm$ 2.2713	8.0450 $\pm$ 0.7783	240.0000 $\pm$ 0.0000	5.4648 $\pm$ 0.3587
REV	Nazemi	0.3635 $\pm$ 0.1420	0.3141 $\pm$ 0.2069	10.7024 $\pm$ 1.4303	11.7828 $\pm$ 2.1210	8.6261 $\pm$ 1.3001	240.0000 $\pm$ 0.0000	8.9886 $\pm$ 0.2800
REV	Nazemi-Nazemi	0.5280 $\pm$ 0.0267	0.4796 $\pm$ 0.0821	9.2560 $\pm$ 0.6150	10.4657 $\pm$ 2.1940	7.4101 $\pm$ 0.5305	240.0000 $\pm$ 0.0000	2.6371 $\pm$ 0.2037
CA	Ours	0.6757 $\pm$ 0.0056	0.6085 $\pm$ 0.0485	0.6859 $\pm$ 0.0111	0.6994 $\pm$ 0.0823	0.5389 $\pm$ 0.0499	240.0000 $\pm$ 0.0000	0.9796 $\pm$ 0.1040
CA	Feizi	0.5036 $\pm$ 0.0342	0.4910 $\pm$ 0.0181	0.8485 $\pm$ 0.0413	0.8010 $\pm$ 0.0961	0.6078 $\pm$ 0.0538	240.0000 $\pm$ 0.0000	5.2533 $\pm$ 0.5070
CA	Nazemi	0.1664 $\pm$ 0.1937	0.1549 $\pm$ 0.0744	1.0939 $\pm$ 0.1482	1.0308 $\pm$ 0.1281	0.7889 $\pm$ 0.1119	240.0000 $\pm$ 0.0000	8.2424 $\pm$ 0.6168
CA	Nazemi-Nazemi	0.6278 $\pm$ 0.0062	0.5855 $\pm$ 0.0351	0.7349 $\pm$ 0.0176	0.7220 $\pm$ 0.0899	0.5452 $\pm$ 0.0541	239.6667 $\pm$ 0.4714	2.6047 $\pm$ 0.3934
CCS	Ours	0.6110 $\pm$ 0.0260	0.4771 $\pm$ 0.1109	10.1663 $\pm$ 0.3017	11.8045 $\pm$ 1.1126	9.4602 $\pm$ 0.9858	240.0000 $\pm$ 0.0000	1.4549 $\pm$ 0.2911
CCS	Feizi	0.5295 $\pm$ 0.0167	0.4308 $\pm$ 0.0825	11.1884 $\pm$ 0.2654	12.3549 $\pm$ 0.7251	10.0349 $\pm$ 0.8645	240.0000 $\pm$ 0.0000	16.7553 $\pm$ 3.5768
CCS	Nazemi	-0.5654 $\pm$ 1.2932	-0.7085 $\pm$ 1.5129	18.8050 $\pm$ 8.0043	19.3491 $\pm$ 8.9907	14.6069 $\pm$ 6.3944	240.0000 $\pm$ 0.0000	19.0664 $\pm$ 0.7829
CCS	Nazemi-Nazemi	0.5936 $\pm$ 0.0290	0.4508 $\pm$ 0.1141	10.3912 $\pm$ 0.3302	12.1044 $\pm$ 1.1184	9.7853 $\pm$ 1.1084	240.0000 $\pm$ 0.0000	4.5410 $\pm$ 0.1351
EE	Ours	0.9094 $\pm$ 0.0016	0.9020 $\pm$ 0.0133	3.0893 $\pm$ 0.0287	3.3106 $\pm$ 0.1899	2.2748 $\pm$ 0.1551	240.0000 $\pm$ 0.0000	1.1616 $\pm$ 0.0891
EE	Feizi	0.8732 $\pm$ 0.0047	0.8612 $\pm$ 0.0257	3.6529 $\pm$ 0.0503	3.9443 $\pm$ 0.4447	2.8476 $\pm$ 0.3240	240.0000 $\pm$ 0.0000	4.0644 $\pm$ 0.5005
EE	Nazemi	0.8380 $\pm$ 0.0320	0.8299 $\pm$ 0.0311	4.1149 $\pm$ 0.4393	4.3508 $\pm$ 0.3371	3.2184 $\pm$ 0.1650	240.0000 $\pm$ 0.0000	7.2587 $\pm$ 0.5994
EE	Nazemi-Nazemi	0.8958 $\pm$ 0.0022	0.8830 $\pm$ 0.0184	3.3130 $\pm$ 0.0372	3.6219 $\pm$ 0.3408	2.5174 $\pm$ 0.1939	240.0000 $\pm$ 0.0000	2.6250 $\pm$ 0.1749
AMPG	Ours	0.8213 $\pm$ 0.0093	0.8003 $\pm$ 0.0394	3.2952 $\pm$ 0.1317	3.5741 $\pm$ 0.5505	2.6454 $\pm$ 0.3210	240.0000 $\pm$ 0.0000	1.1079 $\pm$ 0.0707
AMPG	Feizi	0.7707 $\pm$ 0.0260	0.7506 $\pm$ 0.0348	3.7282 $\pm$ 0.2492	4.0001 $\pm$ 0.5049	2.8437 $\pm$ 0.3143	240.0000 $\pm$ 0.0000	4.5965 $\pm$ 0.3923
AMPG	Nazemi	0.7226 $\pm$ 0.0326	0.7041 $\pm$ 0.0110	4.1015 $\pm$ 0.2954	4.3477 $\pm$ 0.1990	3.2701 $\pm$ 0.1948	240.0000 $\pm$ 0.0000	8.0936 $\pm$ 0.8902
AMPG	Nazemi-Nazemi	0.8166 $\pm$ 0.0102	0.7911 $\pm$ 0.0449	3.3379 $\pm$ 0.1387	3.6542 $\pm$ 0.5983	2.6717 $\pm$ 0.3435	240.0000 $\pm$ 0.0000	2.5841 $\pm$ 0.1806
Airfoil	Ours	0.5741 $\pm$ 0.0037	0.4866 $\pm$ 0.0218	4.6053 $\pm$ 0.0483	5.0824 $\pm$ 0.1986	3.8828 $\pm$ 0.0884	240.0000 $\pm$ 0.0000	2.2712 $\pm$ 0.6650
Airfoil	Feizi	0.4239 $\pm$ 0.0069	0.2699 $\pm$ 0.1055	5.3567 $\pm$ 0.0945	6.0284 $\pm$ 0.0795	4.9602 $\pm$ 0.1575	240.0000 $\pm$ 0.0000	14.0421 $\pm$ 1.2683
Airfoil	Nazemi	0.1342 $\pm$ 0.2329	-0.0879 $\pm$ 0.4360	6.4973 $\pm$ 0.8647	7.2851 $\pm$ 1.5266	5.7603 $\pm$ 0.9257	240.0000 $\pm$ 0.0000	18.5969 $\pm$ 0.2127
Airfoil	Nazemi-Nazemi	0.5314 $\pm$ 0.0045	0.4304 $\pm$ 0.0624	4.8309 $\pm$ 0.0614	5.3366 $\pm$ 0.0710	4.3095 $\pm$ 0.0117	240.0000 $\pm$ 0.0000	5.2469 $\pm$ 0.1788
Yacht	Ours	0.6090 $\pm$ 0.0108	0.5468 $\pm$ 0.0220	9.4387 $\pm$ 0.2416	10.3295 $\pm$ 1.8836	7.0801 $\pm$ 1.0607	240.0000 $\pm$ 0.0000	1.0152 $\pm$ 0.1021
Yacht	Feizi	0.3054 $\pm$ 0.0318	0.2699 $\pm$ 0.0938	12.5865 $\pm$ 0.5827	13.1727 $\pm$ 2.7986	6.7968 $\pm$ 1.9047	240.0000 $\pm$ 0.0000	4.1678 $\pm$ 0.3436
Yacht	Nazemi	0.3165 $\pm$ 0.1730	0.2490 $\pm$ 0.1594	12.4280 $\pm$ 2.0447	12.9999 $\pm$ 1.4816	8.4011 $\pm$ 1.0027	240.0000 $\pm$ 0.0000	7.8837 $\pm$ 0.9267
Yacht	Nazemi-Nazemi	0.4692 $\pm$ 0.0155	0.4248 $\pm$ 0.0479	11.0028 $\pm$ 0.4404	11.6871 $\pm$ 2.3704	6.6850 $\pm$ 1.5063	240.0000 $\pm$ 0.0000	2.1818 $\pm$ 0.5429
WineRed	Ours	0.3866 $\pm$ 0.0052	0.4289 $\pm$ 0.0245	0.6618 $\pm$ 0.0022	0.6788 $\pm$ 0.0124	0.5229 $\pm$ 0.0181	240.0000 $\pm$ 0.0000	2.4917 $\pm$ 0.9734
WineRed	Feizi	0.1565 $\pm$ 0.1501	0.1438 $\pm$ 0.1424	0.7727 $\pm$ 0.0619	0.8280 $\pm$ 0.0672	0.6572 $\pm$ 0.0565	240.0000 $\pm$ 0.0000	14.9152 $\pm$ 0.9404
WineRed	Nazemi	-0.5552 $\pm$ 0.2521	-0.3432 $\pm$ 0.0680	1.0496 $\pm$ 0.0822	1.0408 $\pm$ 0.0159	0.8181 $\pm$ 0.0298	240.0000 $\pm$ 0.0000	20.0936 $\pm$ 0.7506
WineRed	Nazemi-Nazemi	0.3409 $\pm$ 0.0108	0.3224 $\pm$ 0.1190	0.6860 $\pm$ 0.0018	0.7349 $\pm$ 0.0341	0.5732 $\pm$ 0.0279	240.0000 $\pm$ 0.0000	4.9239 $\pm$ 0.1521
Abalone	Ours	0.5185 $\pm$ 0.0144	0.4398 $\pm$ 0.0555	2.3008 $\pm$ 0.0100	2.2898 $\pm$ 0.0544	1.7013 $\pm$ 0.0617	240.0000 $\pm$ 0.0000	8.6044 $\pm$ 3.8582
Abalone	Feizi	0.4258 $\pm$ 0.0196	0.4256 $\pm$ 0.0438	2.5123 $\pm$ 0.0072	2.3302 $\pm$ 0.2167	1.6893 $\pm$ 0.1373	240.0000 $\pm$ 0.0000	12.7554 $\pm$ 2.0242
Abalone	Nazemi	0.2338 $\pm$ 0.1574	0.2902 $\pm$ 0.1635	2.8840 $\pm$ 0.2689	2.5847 $\pm$ 0.4561	2.0117 $\pm$ 0.3920	240.0000 $\pm$ 0.0000	17.1849 $\pm$ 1.2299
Abalone	Nazemi-Nazemi	0.4421 $\pm$ 0.0049	0.4433 $\pm$ 0.0174	2.4772 $\pm$ 0.0349	2.2929 $\pm$ 0.1650	1.6303 $\pm$ 0.1038	240.0000 $\pm$ 0.0000	5.2444 $\pm$ 0.1462

The critical-difference diagram in Figure 4 visualises the same information for Test R^2 : the four methods align on the mean-rank axis, and the horizontal bar below the axis connects methods whose mean ranks differ by less than the critical difference $CD = 1.05$ and are therefore *not* significantly different under the Nemenyi post-hoc test. Ours occupies the best average rank (1.20) and lies in a clique with Nazemi and Nazemi [34] (mean rank 1.95), while Feizi et al. [16] (2.95) and Nazemi [31] (3.90) are strictly separated from the control.

TABLE 4. Predictive performance with the **RBF kernel**, ten UCI datasets (sub-sampled to $n_{\max} = 500$), five independent 80/20 splits. Same four models and metrics as in Table 3. Bandwidth $\gamma = 1/d$ (applied after feature standardization). Best value per column within each dataset block is in bold.

Dataset	Model	R^2		RMSE		MAE	#SV	CPU (s)
		Train	Test	Train	Test	Test		
Boston	Ours	0.8619 $\pm$ 0.0060	0.7606 $\pm$ 0.0363	3.3899 $\pm$ 0.0854	3.6711 $\pm$ 0.4668	2.3252 $\pm$ 0.1638	240.0000 $\pm$ 0.0000	0.8783 $\pm$ 0.0516
Boston	Feizi	0.8244 $\pm$ 0.0048	0.7422 $\pm$ 0.0424	3.8221 $\pm$ 0.0035	3.8135 $\pm$ 0.5382	2.3124 $\pm$ 0.1127	240.0000 $\pm$ 0.0000	3.3976 $\pm$ 0.3146
Boston	Nazemi	0.8220 $\pm$ 0.0015	0.7331 $\pm$ 0.0515	3.8493 $\pm$ 0.0714	3.8797 $\pm$ 0.6054	2.4790 $\pm$ 0.2064	240.0000 $\pm$ 0.0000	6.3439 $\pm$ 0.5244
Boston	Nazemi-Nazemi	0.8491 $\pm$ 0.0016	0.7585 $\pm$ 0.0527	3.5435 $\pm$ 0.0321	3.6875 $\pm$ 0.6124	2.3141 $\pm$ 0.1647	240.0000 $\pm$ 0.0000	2.1664 $\pm$ 0.1783
REV	Ours	0.7027 $\pm$ 0.0281	0.6538 $\pm$ 0.0806	7.3467 $\pm$ 0.6277	8.5470 $\pm$ 2.1188	5.6583 $\pm$ 0.2651	240.0000 $\pm$ 0.0000	0.8758 $\pm$ 0.0453
REV	Feizi	0.5603 $\pm$ 0.0403	0.5430 $\pm$ 0.1252	8.9339 $\pm$ 0.7473	9.8167 $\pm$ 2.6318	6.6858 $\pm$ 0.6414	240.0000 $\pm$ 0.0000	3.9060 $\pm$ 0.7777
REV	Nazemi	0.4306 $\pm$ 0.1021	0.3772 $\pm$ 0.2100	10.1075 $\pm$ 0.9478	11.3878 $\pm$ 3.3199	8.0570 $\pm$ 1.5284	240.0000 $\pm$ 0.0000	8.0841 $\pm$ 0.1909
REV	Nazemi-Nazemi	0.6915 $\pm$ 0.0291	0.6344 $\pm$ 0.0888	7.4838 $\pm$ 0.6388	8.7850 $\pm$ 2.2286	5.9735 $\pm$ 0.3733	240.0000 $\pm$ 0.0000	2.1049 $\pm$ 0.4159
CA	Ours	0.7721 $\pm$ 0.0059	0.6417 $\pm$ 0.0496	0.5749 $\pm$ 0.0062	0.6648 $\pm$ 0.0317	0.5124 $\pm$ 0.0157	240.0000 $\pm$ 0.0000	0.9755 $\pm$ 0.0543
CA	Feizi	0.6574 $\pm$ 0.0136	0.5152 $\pm$ 0.0297	0.7048 $\pm$ 0.0120	0.7809 $\pm$ 0.0914	0.5779 $\pm$ 0.0385	240.0000 $\pm$ 0.0000	4.1303 $\pm$ 1.3588
CA	Nazemi	0.6292 $\pm$ 0.0598	0.4519 $\pm$ 0.0265	0.7309 $\pm$ 0.0552	0.8294 $\pm$ 0.0872	0.6078 $\pm$ 0.0420	240.0000 $\pm$ 0.0000	6.5434 $\pm$ 0.2987
CA	Nazemi-Nazemi	0.7263 $\pm$ 0.0050	0.5987 $\pm$ 0.0159	0.6301 $\pm$ 0.0058	0.7098 $\pm$ 0.0738	0.5348 $\pm$ 0.0428	240.0000 $\pm$ 0.0000	1.7403 $\pm$ 0.1635
CCS	Ours	0.8320 $\pm$ 0.0053	0.6484 $\pm$ 0.0637	6.6848 $\pm$ 0.1016	9.7095 $\pm$ 0.8799	7.5041 $\pm$ 0.7470	240.0000 $\pm$ 0.0000	1.5501 $\pm$ 0.0616
CCS	Feizi	0.7987 $\pm$ 0.0038	0.6590 $\pm$ 0.0510	7.3188 $\pm$ 0.0689	9.5815 $\pm$ 0.8653	7.1014 $\pm$ 0.5102	240.0000 $\pm$ 0.0000	11.0307 $\pm$ 0.3575
CCS	Nazemi	0.7990 $\pm$ 0.0059	0.6558 $\pm$ 0.0557	7.3113 $\pm$ 0.0616	9.6187 $\pm$ 0.8613	7.3552 $\pm$ 0.4924	240.0000 $\pm$ 0.0000	12.0764 $\pm$ 0.3718
CCS	Nazemi-Nazemi	0.8243 $\pm$ 0.0014	0.6580 $\pm$ 0.0389	6.8380 $\pm$ 0.0185	9.6032 $\pm$ 0.6449	7.2162 $\pm$ 0.4621	240.0000 $\pm$ 0.0000	3.9093 $\pm$ 0.2160
EE	Ours	0.9559 $\pm$ 0.0015	0.9437 $\pm$ 0.0089	2.1551 $\pm$ 0.0480	2.5176 $\pm$ 0.2793	1.8370 $\pm$ 0.2779	240.0000 $\pm$ 0.0000	1.0122 $\pm$ 0.1077
EE	Feizi	0.8994 $\pm$ 0.0014	0.8771 $\pm$ 0.0405	3.2561 $\pm$ 0.0494	3.6866 $\pm$ 0.6819	2.1774 $\pm$ 0.4434	240.0000 $\pm$ 0.0000	4.1830 $\pm$ 0.6700
EE	Nazemi	0.8822 $\pm$ 0.0317	0.8467 $\pm$ 0.0862	3.4886 $\pm$ 0.4223	4.0168 $\pm$ 1.2290	2.5782 $\pm$ 0.6367	240.0000 $\pm$ 0.0000	5.4532 $\pm$ 0.7126
EE	Nazemi-Nazemi	0.9529 $\pm$ 0.0011	0.9306 $\pm$ 0.0146	2.2268 $\pm$ 0.0332	2.7911 $\pm$ 0.3801	2.0746 $\pm$ 0.2941	240.0000 $\pm$ 0.0000	2.0763 $\pm$ 0.1551
AMPG	Ours	0.9020 $\pm$ 0.0024	0.8703 $\pm$ 0.0098	2.4394 $\pm$ 0.0085	2.8854 $\pm$ 0.2749	2.0561 $\pm$ 0.1778	240.0000 $\pm$ 0.0000	0.9871 $\pm$ 0.0598
AMPG	Feizi	0.8769 $\pm$ 0.0048	0.8382 $\pm$ 0.0270	2.7329 $\pm$ 0.0271	3.2236 $\pm$ 0.4655	2.3234 $\pm$ 0.2823	240.0000 $\pm$ 0.0000	4.0615 $\pm$ 0.8005
AMPG	Nazemi	0.8621 $\pm$ 0.0049	0.8303 $\pm$ 0.0234	2.8933 $\pm$ 0.0056	3.3020 $\pm$ 0.4281	2.3547 $\pm$ 0.3109	240.0000 $\pm$ 0.0000	5.4526 $\pm$ 0.2165
AMPG	Nazemi-Nazemi	0.8970 $\pm$ 0.0024	0.8630 $\pm$ 0.0098	2.5005 $\pm$ 0.0166	2.9663 $\pm$ 0.2855	2.1248 $\pm$ 0.2155	240.0000 $\pm$ 0.0000	2.3177 $\pm$ 0.2851
Airfoil	Ours	0.7893 $\pm$ 0.0091	0.7185 $\pm$ 0.0428	3.2376 $\pm$ 0.0415	3.7458 $\pm$ 0.1804	2.8452 $\pm$ 0.1661	240.0000 $\pm$ 0.0000	1.1765 $\pm$ 0.1670
Airfoil	Feizi	0.6935 $\pm$ 0.0146	0.5473 $\pm$ 0.1278	3.9050 $\pm$ 0.0367	4.7066 $\pm$ 0.4745	3.6615 $\pm$ 0.4492	240.0000 $\pm$ 0.0000	12.3516 $\pm$ 0.7417
Airfoil	Nazemi	0.6679 $\pm$ 0.0942	0.5914 $\pm$ 0.0760	4.0215 $\pm$ 0.4980	4.5244 $\pm$ 0.5651	3.5291 $\pm$ 0.5206	240.0000 $\pm$ 0.0000	13.5857 $\pm$ 0.8161
Airfoil	Nazemi-Nazemi	0.7697 $\pm$ 0.0057	0.6539 $\pm$ 0.0839	3.3858 $\pm$ 0.0146	4.1285 $\pm$ 0.3345	3.2639 $\pm$ 0.4125	240.0000 $\pm$ 0.0000	3.7381 $\pm$ 0.3395
Yacht	Ours	0.8847 $\pm$ 0.0082	0.8084 $\pm$ 0.0114	5.1206 $\pm$ 0.1131	6.7197 $\pm$ 1.2568	4.0576 $\pm$ 0.9144	240.0000 $\pm$ 0.0000	0.8466 $\pm$ 0.0589
Yacht	Feizi	0.7622 $\pm$ 0.0233	0.7066 $\pm$ 0.0157	7.3527 $\pm$ 0.3866	8.3366 $\pm$ 1.6205	3.7831 $\pm$ 1.1924	240.0000 $\pm$ 0.0000	3.6923 $\pm$ 0.8876
Yacht	Nazemi	0.7619 $\pm$ 0.0339	0.6670 $\pm$ 0.0337	7.3451 $\pm$ 0.4966	8.8563 $\pm$ 1.6421	4.4608 $\pm$ 1.1647	240.0000 $\pm$ 0.0000	5.9503 $\pm$ 0.1437
Yacht	Nazemi-Nazemi	0.8235 $\pm$ 0.0158	0.7394 $\pm$ 0.0149	6.3330 $\pm$ 0.2638	7.8569 $\pm$ 1.5442	4.4405 $\pm$ 1.1434	240.0000 $\pm$ 0.0000	2.1158 $\pm$ 0.1056
WineRed	Ours	0.6023 $\pm$ 0.0176	0.2652 $\pm$ 0.0815	0.5327 $\pm$ 0.0100	0.7685 $\pm$ 0.0363	0.5756 $\pm$ 0.0169	240.0000 $\pm$ 0.0000	1.3503 $\pm$ 0.1247
WineRed	Feizi	0.5167 $\pm$ 0.0230	0.0349 $\pm$ 0.1443	0.5873 $\pm$ 0.0117	0.8783 $\pm$ 0.0331	0.6694 $\pm$ 0.0329	240.0000 $\pm$ 0.0000	7.2336 $\pm$ 1.3711
WineRed	Nazemi	0.5158 $\pm$ 0.0174	0.0330 $\pm$ 0.1579	0.5880 $\pm$ 0.0125	0.8785 $\pm$ 0.0360	0.6716 $\pm$ 0.0420	240.0000 $\pm$ 0.0000	13.8882 $\pm$ 1.7920
WineRed	Nazemi-Nazemi	0.5853 $\pm$ 0.0215	0.1044 $\pm$ 0.1535	0.5440 $\pm$ 0.0147	0.8449 $\pm$ 0.0474	0.6325 $\pm$ 0.0411	240.0000 $\pm$ 0.0000	3.5945 $\pm$ 0.5245
Abalone	Ours	0.5729 $\pm$ 0.0056	0.4730 $\pm$ 0.0178	2.1673 $\pm$ 0.0209	2.2310 $\pm$ 0.1614	1.6642 $\pm$ 0.0905	240.0000 $\pm$ 0.0000	1.3543 $\pm$ 0.0734
Abalone	Feizi	0.5262 $\pm$ 0.0147	0.3966 $\pm$ 0.0287	2.2829 $\pm$ 0.0599	2.3847 $\pm$ 0.1428	1.7779 $\pm$ 0.0962	240.0000 $\pm$ 0.0000	12.8412 $\pm$ 2.1437
Abalone	Nazemi	0.4915 $\pm$ 0.0442	0.3782 $\pm$ 0.0456	2.3628 $\pm$ 0.1053	2.4251 $\pm$ 0.2256	1.8227 $\pm$ 0.1725	240.0000 $\pm$ 0.0000	14.2244 $\pm$ 0.8957
Abalone	Nazemi-Nazemi	0.5424 $\pm$ 0.0024	0.4477 $\pm$ 0.0196	2.2438 $\pm$ 0.0376	2.2866 $\pm$ 0.1999	1.6904 $\pm$ 0.1443	240.0000 $\pm$ 0.0000	4.0260 $\pm$ 0.6398

TABLE 5. Friedman test and Holm post-hoc analysis pooled over ten UCI benchmarks and two kernels ($N = 20$ problems, $k = 4$ models, control = Ours, $\alpha = 0.05$). The column “NN” abbreviates the smoothed-FB network of Nazemi and Nazemi [34]. Lower mean rank is better for every metric.

Metric	χ_F^2	p_F	Ours rank	Feizi rank	Nazemi rank	NN rank	p_{Holm} vs Feizi	p_{Holm} vs Nazemi	p_{Holm} vs NN
Test R^2	49.86	$< 10^{-4}$	1.20	2.95	3.90	1.95	$< 10^{-4}$	$< 10^{-4}$	0.0662
Test RMSE	49.74	$< 10^{-4}$	1.15	3.00	3.85	2.00	$< 10^{-4}$	$< 10^{-4}$	0.0373
Test MAE	39.42	$< 10^{-4}$	1.50	2.65	3.90	1.95	0.0097	$< 10^{-4}$	0.2703
CPU time	58.86	$< 10^{-4}$	1.05	3.00	4.00	1.95	$< 10^{-4}$	$< 10^{-4}$	0.0275

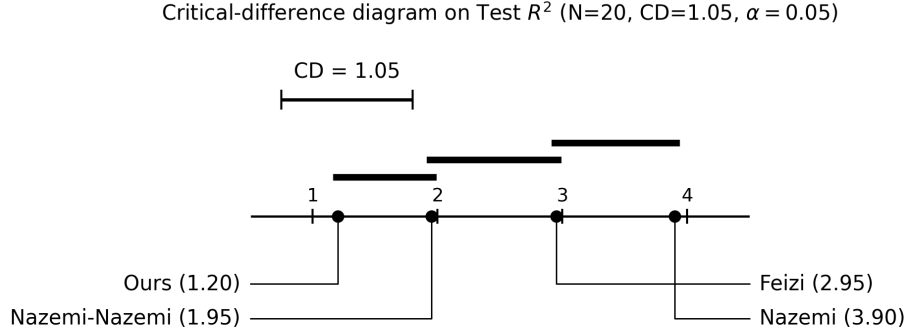


FIGURE 4. Critical-difference (Nemenyi) diagram for Test R^2 over $N = 20$ problems (ten UCI datasets $\times$ two kernels). Methods connected by a solid horizontal bar differ by less than the critical difference $CD = 1.05$ and are *not* significantly different at $\alpha = 0.05$. The proposed compact $6n$ model attains the best mean rank and separates the two remaining baselines (Feizi et al. [16] and Nazemi [31]) from the leading clique.

5.4. Reproducibility. All four neurodynamic models and every experimental protocol described above are implemented in a single open-source Python package released with this paper. The four networks — the proposed compact $6n$ architecture, the $8n$ projection network of Feizi et al. [16], the $8n+1$ Lagrangian network of Nazemi [31], and the $8n+1$ smoothed Fischer–Burmeister gradient network of Nazemi and Nazemi [34] — share the same kernel layer, the same dataset pipeline, the same standardization rule (zero-mean, unit-variance features, fit on the training split only) and the same uncertainty model $\tilde{X} = X + \delta \mathbf{1}$ with $\delta = 0.1$, so that any differences reported in Tables 3–4 can be attributed solely to the underlying dynamics.

The benchmark protocol is fixed across all tables: ten independent 80/20 splits with a deterministic seed schedule, feature standardization applied after splitting, and the hyperparameters $C = 1$, $\varepsilon = 0.1$, $\delta = 0.5$, $a = 0.2$ and FB smoothing $\eta = 10^{-3}$ used for every model. The continuous-time dynamics are integrated on the horizon $t \in [0, 0.1]$ using the `solve_ivp` routine from SciPy 1.16 (BDF method) with relative tolerance 10^{-3} and absolute tolerance 10^{-6} ; the step-size scaling γ_θ (and γ_u for the proposed model) is set to 400 on the largest dataset (CCS) and 200 elsewhere, reflecting the spectral scaling of the kernel matrix. Convergence is monitored via the merit function E_ε of equation (3.3), and reported values are mean $\pm$ standard deviation over the ten seeds.

All numerical simulations were implemented in Python 3.13 and executed on an Apple MacBook Pro equipped with an Apple M4 Max chip (16-core CPU: 12 performance cores and 4 efficiency cores) and 48 GB of unified memory, running macOS 26.1. Numerical arrays and linear algebra operations were handled with NumPy 2.3. Visualization was produced with Matplotlib 3.10. No GPU acceleration was employed; all computations ran on the CPU.

6. CONCLUSIONS

This paper presented a novel, compact neurodynamic architecture designed to solve the stochastic support vector regression (SSVR) problem under probabilistic constraints. By leveraging a smoothed Fischer-Burmeister function, we transformed the non-smooth Karush-Kuhn-Tucker (KKT) complementarity conditions into a differentiable residual system, enabling an efficient gradient-based descent approach. A significant contribution of this work is the reduction of network complexity from $8n$ to $6n$ neurons, representing a 25% decrease in state-space dimensionality while maintaining theoretical integrity. We established the existence, uniqueness, and global asymptotic stability of the neural trajectories using Lyapunov stability theory, rigorously proving that the system converges to the optimal solution of the original SSVR problem. Extensive numerical experiments on synthetic data and ten UCI benchmark datasets validated the superiority of our approach against three representative neurodynamic SSVR baselines: the Lagrangian network of Nazemi [31], the smoothed Fischer–Burmeister gradient network of Nazemi and Nazemi [34], and the projection recurrent network of Feizi et al. [16]. The results demonstrate that the proposed model achieves generally higher predictive accuracy (higher R^2 , lower RMSE and MAE) and lower CPU time than all three baselines under both linear and RBF kernels. Future research may explore further computationally optimized implementations of this compact architecture for more complex datasets or its integration into large-scale deep learning frameworks.

Acknowledgements

This work was financially supported by Walailak University under the New Researcher Development scheme (Contract N. WU69244). This work was also partially supported by the Frontier Research and Innovation Cluster Fund, Naresuan University (Grant No. R2569C008)

REFERENCES

- [1] M. Abaszade, S. Effati, Stochastic support vector regression with probabilistic constraints, *Applied Intelligence*, 48 (2018), 243–256,
- [2] M. Awad, R.I. Khanna, Support vector regression, In: *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, pp. 67–80, Apress, 2015,
- [3] K. Bache, M. Lichman, *UCI Machine Learning Repository*, 2013,
- [4] A. Ben-Tal, S. Bhadra, C. Bhattacharyya, J.S. Nath, Chance constrained uncertain classification via robust optimization, *Mathematical Programming*, 127 (2011), 145–173.
- [5] C. Bhattacharyya, P.K. Shivaswamy, A.J. Smola, A second order cone programming formulation for classifying missing data, In: *NIPS’04: Proceedings of the 18th International Conference on Neural Information Processing Systems*, pp. 153–16, 2004.
- [6] E. Carrizosa, J. Gordillo, F. Plastra, Kernel support vector regression with imprecise output, Technical report, Vrije Universiteit Brussel, 2008.
- [7] J. Cervantes, F. Garcia-Lamont, L. Rodriguez-Mazahua, A. Lopez, A comprehensive survey on support vector machine classification: Applications, challenges and trends, *Neurocomputing*, 408 (2020), 189–215.
- [8] C.C. Chuang, S.F. Su, J.T. Jeng, C.C. Hsiao, Robust support vector regression networks for function approximation with outliers, *IEEE Transactions on Neural Networks*, 13 (2022), 1322–1330.
- [9] C. Cortes, V.N. Vapnik, Support-vector networks, *Machine Learning*, 20 (1995), 273–297.
- [10] J. Demšar, Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research*, 7 (2006), 1–30.
- [11] H. Drucker, C.J.C. Burges, L. Kaufman, A.J. Smola, V.N. Vapnik, Support vector regression machines, *Advances in Neural Information Processing Systems*, 1996.

- [12] F. Facchinei, H. Jiang, L. Qi, A smoothing method for mathematical programs with equilibrium constraints, *Mathematical Programming*, 85 (1995), 107–134.
- [13] F. Facchinei, J.S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems*, Volume I Springer Series in Operations Research, Springer-Verlag, 2003.
- [14] A. Feizi, A. Nazemi, An application of a practical neural network model for solving support vector regression problems, *Intelligent Data Analysis*, 21 (2017), 1443–1461.
- [15] A. Feizi, A. Nazemi, Classifying random variables based on support vector machine and a neural network scheme, *Journal of Experimental & Theoretical Artificial Intelligence*, 36 (2024), 679–702.
- [16] A. Feizi, A. Nazemi, M. Rabiei, Solving the stochastic support vector regression with probabilistic constraints by a high-performance neural network model, *Engineering with Computers*, 38 (2021), 1005–1020.
- [17] T.L. Friesz, D. Bernstein, N.J. Mehta, R.L. Tobin, S. Ganjalizadeh, Day-to-day dynamic network disequilibria and idealized traveler information systems, *Operations Research*, 42 (1994), 1120–1136.
- [18] S.R. Gunn, Support vector machines for classification and regression, University of Southampton, Faculty of Engineering, Institute for Sound and Vibration Research, vol. 14, *Image Speech and Intelligent Systems Technical Report*, pp. 1–54, 1998.
- [19] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, *Neurocomputing*, 70 (2006), 489–501.
- [20] H. Huang, X. Wei, Y. Zhou, An overview on twin support vector regression, *Neurocomputing*, 490 (2022), 80–92.
- [21] M. Jahangiri, A. Nazemi, An efficient rnn based algorithm for solving fuzzy nonlinear constrained programming problems with numerical experiments, *Journal of Computational and Applied Mathematics*, 463 (2025), 116448.
- [22] F. Jarre, A rank-one-update method for the training of support vector machines, *Journal of Nonlinear and Variational Analysis* 10 (2026), 363–379.
- [23] M.P. Kennedy, L.O. Chua, Neural networks for nonlinear programming, *IEEE Transactions on Circuits and Systems*, 35 (1988), 554–562.
- [24] H.K. Khalil, *Nonlinear Systems*, Prentice Hall, 3 edition, 2002.
- [25] W.E. Lillo, M.H. Loh, S. Hui, S.H. Zak, On solving constrained optimization problems with neural networks: A penalty method approach, *IEEE Transactions on Neural Networks*, 4 (1993), 931–940.
- [26] C.J. Lin, g et al., Simple probabilistic predictions for support vector regression, National Taiwan University, Taipei, 2004.
- [27] J. Liu, R. Seraoui, V. Vitelli, E. Zio, Probabilistic support vector regression for short-term prediction of power plants equipment, In: *Prognostics and System Health Management Conference*, 2013.
- [28] Q. Liu, Y. Zhao, A continuous-time recurrent neural network for real-time support vector regression, In: *2013 IEEE Symposium on Computational Intelligence in Control and Automation (CICA)*, 2013,
- [29] K.E. Moutaouakil, A.E. Ouissari, Opt-RNN-DBFSVM: Optimal recurrent neural network density based fuzzy support vector machine, *RAIRO - Operations Research*, 57 (2023), 2493–2517.
- [30] K.E. Moutaouakil, A.E. Ouissari, A. Olaru, V. Palade, M. Ciorei, OPT-RNN-DBSVM: Optimal recurrent neural network and density-based support vector machine, *Mathematics*, 11 (2023), 3555.
- [31] A. Nazemi, A dynamic system model for solving convex nonlinear optimization problems, *Communications in Nonlinear Science and Numerical Simulation*, 17 (2012), 1696–1705.
- [32] A. Nazemi, Solving general convex nonlinear optimization problems by an efficient neurodynamic model, *Engineering Applications of Artificial Intelligence*, 26 (2013), 685–696.
- [33] A. Nazemi, A neural network model for solving convex quadratic programming problems with some applications, *Engineering Applications of Artificial Intelligence*, 32 (2014), 54–62.
- [34] A. Nazemi, M. Nazemi, A gradient-based neural network method for solving strictly convex quadratic programming problems, *Cognitive Computation*, 6 (2014), 484–495.
- [35] A. Nazemi, A. Sabeghi, A novel gradient-based neural network for solving convex second-order cone constrained variational inequality problems, *Journal of Computational and Applied Mathematics*, 347 (2019), 343–356.
- [36] J.M. Ortega, W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Society for Industrial and Applied Mathematics, 2000.

- [37] S. Pan, J.S. Chen, A semismooth newton method for SOCCPs based on a one-parametric class of SOC complementarity functions, *Computational Optimization and Applications*, 45 (2010), 59–88.
- [38] X. Peng, TSVR: An efficient twin support vector machine for regression, *Neural Networks*, 23 (2010), 365–372.
- [39] A. Roy, S. Chakraborty, Support vector machine in structural reliability analysis: A review, *Reliability Engineering & System Safety*, 233 (2023), 109126.
- [40] P.K. Shivaswamy, C. Bhattacharyya, A.J. Smola, Second order cone programming approaches for handling missing and uncertain data, *Journal of Machine Learning Research*, 7 (2006), 1283–1314.
- [41] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, *Statistics and Computing*, 14 (2004), 199–222.
- [42] D. Sun, J. Sun, Strong semismoothness of the fischer-burmeister SDC and SOC complementarity functions, *Mathematical Programming*, 103 (2005), 575–581.
- [43] D.W. Tank, J.J. Hopfield, Simple ‘neural’ optimization networks: An a/d converter, signal decision circuit, and a linear programming circuit, *IEEE Transactions on Circuits and Systems*, 33 (1986), 533–541.
- [44] M. Tanveer, T. Rajani, R. Rastogi, Y.-H. Shao, Comprehensive review on twin support vector machines, *Annals of Operations Research*, 339 (2021), 1223–1268.
- [45] T.B. Trafalis, S.A. Alwazzi, Support vector regression with noisy data: a second order cone programming approach, *International Journal of General Systems*, 36 (2007), 237–250.
- [46] T.B. Trafalis, R.C. Gilbert, Robust classification and regression using support vector machines, *European Journal of Operational Research*, 173 (2006), 893–909.
- [47] V.N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 2013.
- [48] Y. Wang, J. Wang, Neurodynamics-driven holistic approaches to semi-supervised feature selection, *Neural Networks*, 157 (2023), 377–386.
- [49] Y. Xia, A new neural network for solving linear and quadratic programming problems, *IEEE Transactions on Neural Networks*, 7 (1996), 1544–1548.
- [50] Y. Xia, J. Wang, A one-layer recurrent neural network for support vector machine learning, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34 (2004), 1261–1269.
- [51] F. Zhang, L.J. O’Donnell, Support vector regression, In: *Machine Learning: Methods and Applications to Brain Disorders*, pp. 121–139, Academic Press, 2020.
- [52] Y. Zhao, Q. Liu, Generalized recurrent neural network for ϵ -insensitive support vector regression, *Mathematics and Computers in Simulation*, 86 (2012), 2–9.