ENGINEERING Information Technology & Electronic Engineering
https://www.academax.com/EITEE; https://www.engineering.org.cn
https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=11344065
E-mail: eng.itee@zju.edu.cn

Supplementary materials for

Yanpeng ZHENG, Guijie ZHANG, Xiaoyu JIANG, Zhaolin JIANG, Sung-Kwun OH, 2026. A neural network algorithm for superstructure quadrilateral dynamic resistor networks on hammock surfaces with application to artificial intelligence. *ENGINEERING Information Technology & Electronic Engineering*, 27(6):260055.
https://doi.org/10.1631/ENG.ITEE.2026.0055

1 Activation functions used in FZNNHRN

1.1 Monotonically increasing odd activation functions

This class includes functions that preserve symmetry about the origin and grow consistently with the input magnitude.

Linear activation function (LAF) (Wu and Zheng, 2020). The simplest identity mapping is defined as

$$f(u) = u. \tag{S1}$$

Bipolar sigmoid activation function (BSAF) (Wu and Zheng, 2020): This function introduces saturation at large values:

$$f(u) = \frac{1-\exp(-\varepsilon u)}{1+\exp(-\varepsilon u)}, \tag{S2}$$

where $\varepsilon \geq 1$ controls the slope.

Power sigmoid activation function (PSAF) (Wu and Zheng, 2020): This function combines a power term and a bounded sigmoid as

$$f(u) = \begin{cases} u^a, & \text{if } |u| \geq 1, \\ \frac{1+\exp(-q)}{1-\exp(-q)} \cdot \frac{1-\exp(-qu)}{1+\exp(-qu)}, & \text{if } |u| < 1, \end{cases} \tag{S3}$$

with $q > 2$ and $a = 2s+1$, $s = 1, 2, \ldots$.

Smooth power sigmoid function(SPSF) (Wu and Zheng, 2020): A smooth combination of power and sigmoid terms:

$$f(u) = \left(\frac{1}{2}u\right)^a + \frac{1+\exp(-q)}{1-\exp(-q)} \tanh\left(\frac{qu}{2}\right), \tag{S4}$$

where $q > 2$ and $a = 2s+1$, $s = 1, 2, \ldots$.

Sign-bi-power activation function (SBPAF) (Jia et al., 2020; Stanimirović et al., 2020): Incorporates symmetry and power variation:

$$f(u) = \frac{1}{2}\left(|u|^\alpha + |u|^{1/\alpha}\right) \cdot \operatorname{sign}(u), \tag{S5}$$

with $\alpha > 0$.

1.2 Multiplicative-type activation functions

These functions scale the input x by a smooth, positive modulation term $\Psi(x)$ such that $f(x) = x \cdot \Psi(x)$.
Hyperbolic tangent (Tanh) (Jentzen et al., 2023): The output is smoothed via the hyperbolic tangent:

$$f(u) = \tanh(u). \tag{S6}$$

Swish (Jentzen et al., 2023) introduces a sigmoid-based gating mechanism:

$$f(u) = u \cdot \text{sigmoid}(u). \tag{S7}$$

Gaussian error linear unit (GELU) (Jentzen et al., 2023): The input is scaled by the standard normal CDF:

$$f(u) = \frac{u}{2}\left[1 + \text{erf}\left(\frac{u}{\sqrt{2}}\right)\right], \tag{S8}$$

where $f(\cdot)$ denotes the error function.

Mish (Jentzen et al., 2023) employs a soft approximation to identity:

$$f(u) = u \cdot \frac{(1+e^u)^2 - 1}{(1+e^u)^2 + 1}. \tag{S9}$$

2 Individual neuron dynamics and circuit schematics

The FZNNHRN belongs to the class of recurrent neural networks. According to Eq. (16), the dynamic behavior of the i^{th} neuron (for $i = 1, 2, \ldots, mn$) is characterized as follows:

$$\dot{\boldsymbol{V}}_i = \sum_{j=1}^{mn} p_{i,j}\dot{\boldsymbol{V}}_i - \gamma \sum_{j=1}^{mn} \beta_{i,j}\boldsymbol{f}_j - \sum_{j=1}^{mn} \mu_{i,j}\boldsymbol{V}_j, \tag{S10}$$

where

$$\boldsymbol{f}_j = f\left(\sum_{k=1}^{mn} \delta_{j,k}\boldsymbol{\nu}_k - \boldsymbol{\mathcal{I}}_j\right). \tag{S11}$$

In this context, $\boldsymbol{V}_i$ refers to the i^{th} component of the neural state vector $\boldsymbol{V}(t)$, corresponding to the i^{th} neuron in model (12). The variables $p_{i,j}$, $\beta_{i,j}$, and $\mu_{i,j}$ denote the $(i,j)^{\text{th}}$ entries of the matrices $\boldsymbol{E}_{mn} - \boldsymbol{L}_{mn}^{\text{hammock}}(t)$, $\left[\left(\boldsymbol{L}_{mn}^{\text{hammock}}(t)\right)^{\text{T}}\boldsymbol{L}_{mn}^{\text{hammock}}(t) + \lambda\boldsymbol{E}_{mn}\right]^k$, and $\dot{\boldsymbol{L}}_{mn}^{\text{hammock}}(t)$, respectively. The symbol $\delta_{j,k}$ represents the $(j,k)^{\text{th}}$ element of the vector $\boldsymbol{L}_{mn}^{\text{hammock}}(t)$. The function $f(\cdot)$ corresponds to a component of the vector-valued function $\boldsymbol{\mathcal{F}}$. A structural schematic of the neural network architecture of the FZNNHRN, as described by Eq. (16), is provided in Fig. S1.

3 DPFPP-based real-time dynamic path planning

To better approximate real-world scenarios, half of all obstacles in the environment are designed to be dynamic. Among these dynamic obstacles, 25% move periodically in the horizontal direction, another 25% move in the vertical direction, and the remaining 50% follow random trajectories.

To enhance the system's autonomy and robustness, an obstacle avoidance mechanism is integrated into the robot. When an obstacle is detected along the robot's path, a new path from the current position to the destination is replanned. This ensures reliable completion of navigation tasks in complex and dynamic environments.

Fig. S2 shows the final snapshot of the dynamic path planning results for all five algorithms under identical environmental conditions. The complete animated GIF illustrating the full obstacle avoidance process is provided as a separate supplementary video file.

Algorithm S1 Directional potential field path planning

Algorithm S1 presents the implementation details of the directional potential field path planning (DPFPP). The logical execution flow of this algorithm is further illustrated in the flowchart in Fig S3.

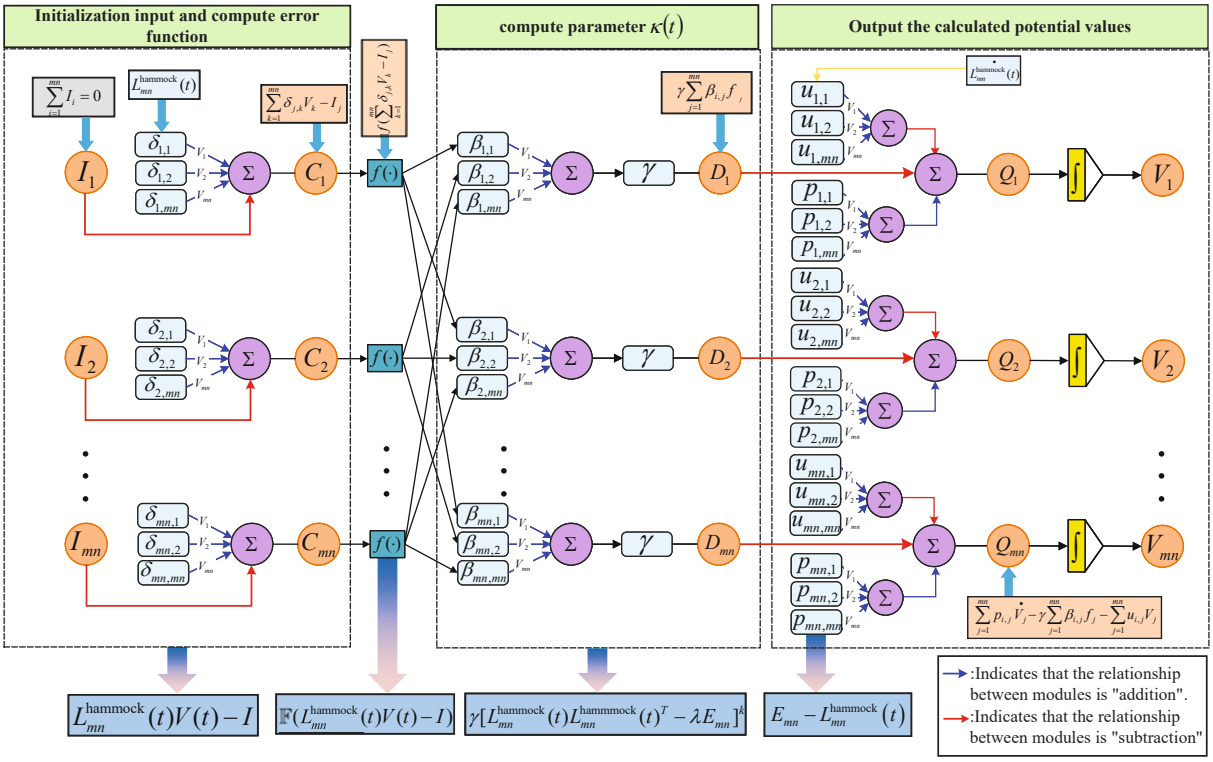

Fig. S1 Circuit schematics of FZNNHRN in the form of Eq. (16)

Algorithm S1 Directional potential field path planning algorithm

Initialize environment parameters (m, n), obstacle set $\mathcal{O}$, start node S, target node T, potential map $\boldsymbol{U}$, and set maximum iterations $N_{\max}$
Set current node $C \leftarrow S$, initialize visited set $\mathcal{V} \leftarrow \{S\}$, and previous direction $\mathbf{d} \leftarrow (0, 0)$
If $C = T$, terminate and output the path
If path length is 1, select neighbor p minimizing $\|p - T\|_2$ with $p \notin \mathcal{O} \cup \mathcal{V}$
    append p to path, set $\mathbf{d} \leftarrow p - C$, update $C \leftarrow p$, and go to Step 3
Generate up to three candidate nodes: forward $\mathbf{d}$, CCW 45°, CW 45° rotations
For each candidate c, compute score
    $S(c) = U(c) + w\,[1 - \cos\theta(c)]$
    where $\theta(c)$ is the angle between $c - C$ and $T - C$, $U(c)$ is the potential value from $\boldsymbol{U}$
    and w is the penalty weight
Select $c^* = \arg\min_c S(c)$ among valid candidates
If $\|c^* - T\|_2 < \|C - T\|_2$
    move to c^*
Else select neighbor minimizing $\|p - T\|_2$ with $p \notin \mathcal{O} \cup \mathcal{V}$
If no valid move exists
    If path length > 1
        backtrack to previous node; else terminate with empty path
Append new node to path, update $\mathcal{V}$ and $\mathbf{d}$, set $C \leftarrow$ new node, and go to Step 3

Note: $\cos\theta(c) = \dfrac{(c - C) \cdot (T - C)}{\|c - C\|_2 \, \|T - C\|_2}$

Proof of Eq. (34)

It can be noted that the inequality $\lambda_{\min}(\boldsymbol{A})\mathrm{tr}(\boldsymbol{B}) \leq \mathrm{tr}(\boldsymbol{AB}) \leq \lambda_{\max}(\boldsymbol{A})\mathrm{tr}(\boldsymbol{B})$ holds when the matrix $\boldsymbol{A}$ is symmetric and positive semi-definite, while $\boldsymbol{B}$ is symmetric. Here, $\lambda_{\min}(\boldsymbol{A})$ and $\lambda_{\max}(\boldsymbol{B})$ denote the minimum eigenvalue of matrix $\boldsymbol{A}$ and the maximum eigenvalue of matrix $\boldsymbol{B}$, respectively. Since $\boldsymbol{L}_{mn}^{\text{hammock}}(t)$

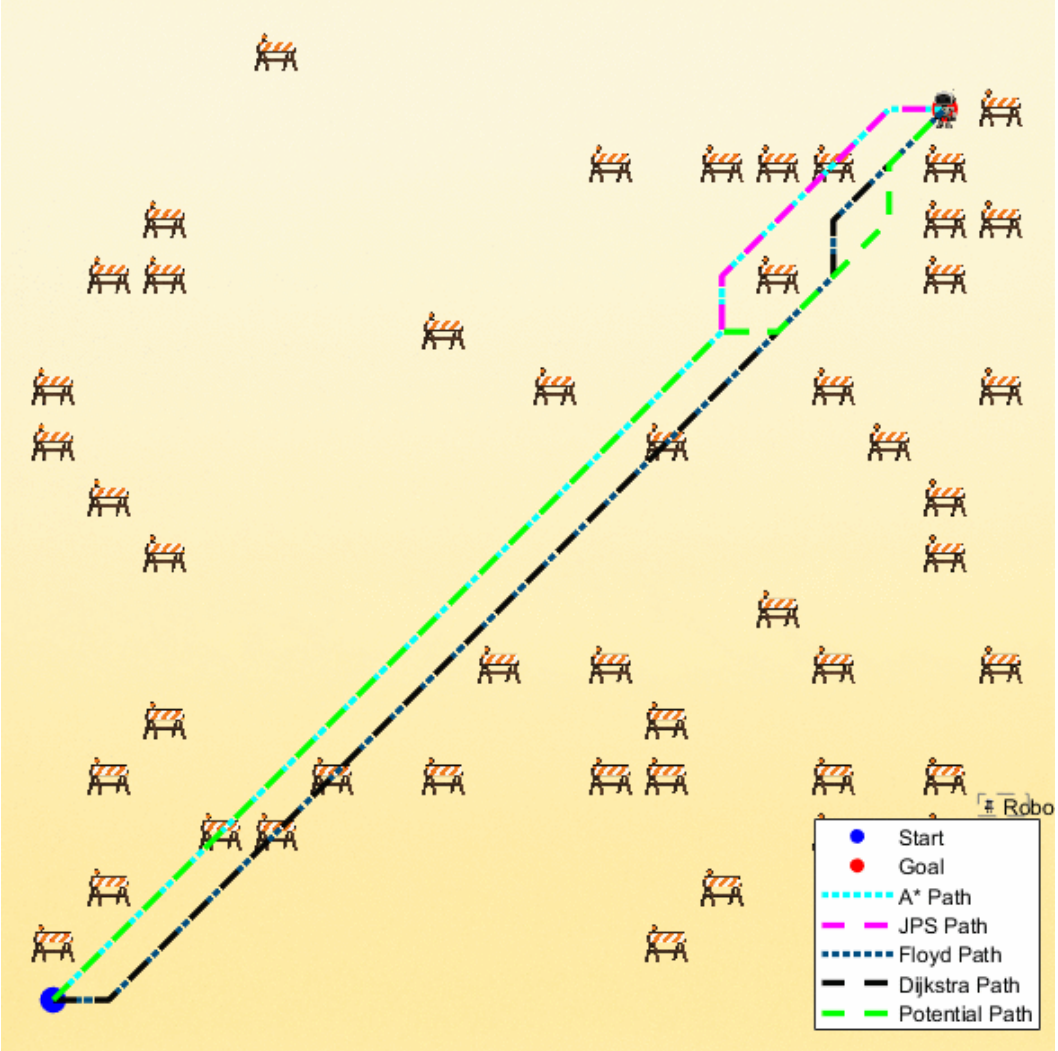

Fig. S2 Dynamic path planning demonstrations for five algorithms

and $(\boldsymbol{L}_{mn}^{\text{hammock}}(t))^{\mathrm{T}}$ share the same set of eigenvalues, it can thus be concluded that

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{L}(\boldsymbol{E}(t),t)}{\mathrm{d}t} &\leq -\gamma\big(\boldsymbol{\alpha}^2(t)+\lambda\big)^k\ \mathrm{tr}\big(\boldsymbol{E}(t)^{\mathrm{T}}\boldsymbol{\mathcal{F}}(\boldsymbol{E}(t))\big)\\ &= -\gamma\big(\boldsymbol{\alpha}^2(t)+\lambda\big)^k\sum_{i=1}^{m}\sum_{j=1}^{n}e_{ij}f(e_{ij}).\end{aligned}\tag{S12}$$

where $\boldsymbol{\alpha}(t)=\lambda_{\min}(\boldsymbol{L}_{mn}^{\text{hammock}}(t))$.

Define $f(\cdot)$ as the primitive function derived from the monotonically increasing and odd activation function $\boldsymbol{\mathcal{F}}(\cdot)$. Accordingly, the function $f(\cdot)$ fulfills the following criteria:

$$f(u)=\begin{cases}>0, & \text{if } u>0,\\ =0, & \text{if } u=0,\\ <0, & \text{if } u<0.\end{cases}\tag{S13}$$

As $e(\cdot)$ serves as the basic function corresponding to the error function $\boldsymbol{E}(\cdot)$, one can deduce that

$$e_{ij}f(e_{ij})=\begin{cases}>0, & \text{if } e_{ij}\neq 0,\\ =0, & \text{if } e_{ij}=0.\end{cases}\tag{S14}$$

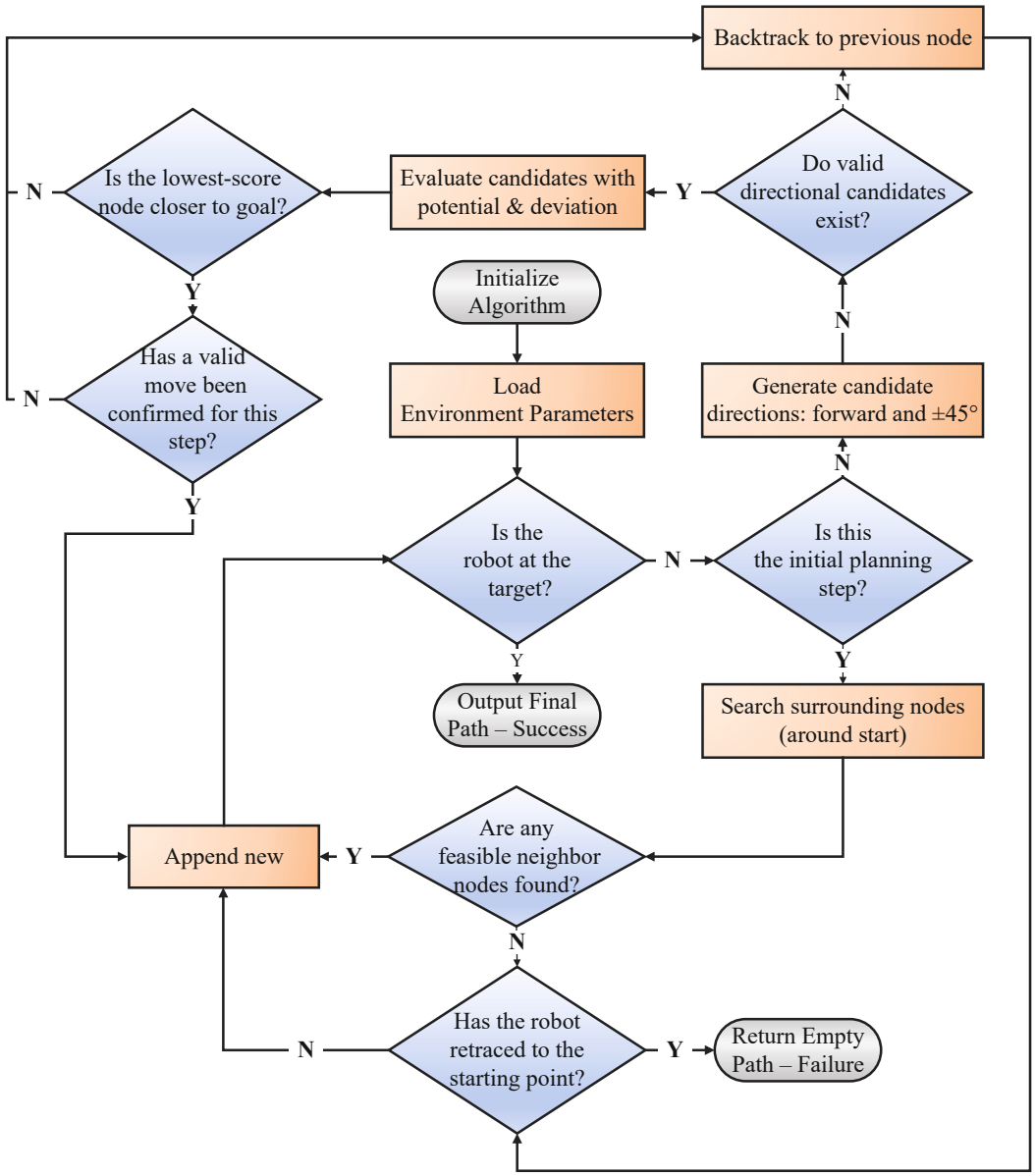

Fig. S3 Algorithm flowchart for path planning (DPFPP)

In conclusion

$$L(\boldsymbol{E}(t),t)=\begin{cases}<0, & \text{if } \boldsymbol{E}(t)\neq 0,\\ =0, & \text{if } \boldsymbol{E}(t)=0.\end{cases} \tag{S15}$$

Proof of convergence analysis for LAF

If the activation function $\boldsymbol{\mathcal{F}}(\cdot)$ adopts an LAF, Eq. (33) can be rewritten as:

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{L}(\boldsymbol{E}(t),t)}{\mathrm{d}t}&=-\gamma\ \mathrm{tr}\Big(\big(\boldsymbol{L}_{mn}^{\mathrm{hammock}}(t)\boldsymbol{L}_{mn}^{\mathrm{hammock}}(t)^{\mathrm{T}}+\lambda\boldsymbol{E}_{mn}\big)^{k}\boldsymbol{E}(t)^{\mathrm{T}}\boldsymbol{E}(t)\Big)\\&\leq-\gamma\left(\boldsymbol{\alpha}^{2}(t)+\lambda\right)^{k}\mathrm{tr}\big(\boldsymbol{E}(t)^{\mathrm{T}}\boldsymbol{E}(t)\big)\\&=-2\gamma\left(\boldsymbol{\alpha}^{2}(t)+\lambda\right)^{k}\boldsymbol{L}(\boldsymbol{E}(t),t).\end{aligned} \tag{S16}$$

Upon analyzing the aforementioned differential inequality $\dot{\boldsymbol{L}}(t)\leq-2\gamma(\boldsymbol{\alpha}^{2}(t)+\lambda)^{k}\boldsymbol{L}(t)$, it can be deduced

that $\boldsymbol{L}(t) \leq \boldsymbol{L}(0)\exp(-2\gamma(\boldsymbol{\alpha}^2(t)+\lambda)^k t)$.

For the FZNNHRN model with a linear activation function, the convergence rate is given by

$$\mathrm{ECR} = \gamma\left(\boldsymbol{\alpha}^2(t)+\lambda\right)^k. \tag{S17}$$

Proof of Theorem 2

Let $f(\cdot)$ denote the elemental function associated with the activation function $\boldsymbol{\mathcal{F}}(\cdot)$, defined as $f(x) = x\cdot\boldsymbol{\varphi}(x)$, where $\boldsymbol{\varphi}(x) > 0$ and $\boldsymbol{\varphi}(x)$ corresponds to the elemental form of $\boldsymbol{\Psi}(x)$. Based on Eq. (S10), the following inequality holds:

$$\frac{d\boldsymbol{L}(\boldsymbol{E}(t),t)}{dt} \leq -\gamma\left(\boldsymbol{\alpha}^2(t)+\lambda\right)^k \sum_{i=1}^{m}\sum_{j=1}^{n} e_{ij}^2\boldsymbol{\varphi}(e_{ij}). \tag{S18}$$

It follows that

$$\boldsymbol{L}(\boldsymbol{E}(t),t) = \begin{cases} <0, & \text{if } \boldsymbol{E}(t)\neq 0,\\ =0, & \text{if } \boldsymbol{E}(t)=0. \end{cases} \tag{S19}$$

By Lyapunov's direct method, the above inequality ensures that $\boldsymbol{E}(t)$ globally converges to zero for any arbitrary initial condition $\boldsymbol{V}(0)$.

References

Jentzen A, Kuckuck B, von Wurstemberger P, 2023. Mathematical introduction to deep learning: methods, implementations, and theory. *arXiv preprint arXiv:231020360*, . https://doi.org/10.48550/arXiv.2310.20360

Jia L, Xiao L, Dai J, et al., 2020. Design and application of an adaptive fuzzy control strategy to zeroing neural network for solving time-variant qp problem. *IEEE Trans Fuzzy Syst*, 29(6):1544-1555. https://doi.org/10.1109/TFUZZ.2020.2981001

Stanimirović PS, Katsikis VN, Zhang Z, et al., 2020. Varying-parameter zhang neural network for approximating some expressions involving outer inverses. *Optim Methods Softw*, 35(6):1304-1330. https://doi.org/10.1080/10556788.2019.1594806

Wu W, Zheng B, 2020. Improved recurrent neural networks for solving moore-penrose inverse of real-time full-rank matrix. *Neurocomputing*, 418:221-231. https://doi.org/10.1016/j.neucom.2020.08.026