Received: 24 December 2021 | Accepted: 29 December 2021
DOI: 10.1002/nme.6918

RESEARCH ARTICLE

WILEY

Deep learned one-iteration nonlinear solver for solid mechanics

Tan N. Nguyen[1] | Jaehong Lee[2] | Liem Dinh-Tien[3] | L. Minh Dang[4]

[1]Department of Architectural Engineering, Sejong University, Gwangjin-gu, Seoul, Republic of Korea
[2]Deep Learning Architecture Research Center, Sejong University, Gwangjin-gu, Seoul, Republic of Korea
[3]Faculty of Fundamental Sciences, Van Lang University, Ho Chi Minh city, Vietnam
[4]Department of Information Technology, FPT University, Ho Chi Minh city, Vietnam

Correspondence
L. Minh Dang, Department of Information Technology, FPT University, Ho Chi Minh City, Vietnam.
Email: minhdl3@fe.edu.vn
Tan N. Nguyen, Department of Architectural Engineering, Sejong University, 209 Neungdong-ro, Gwangjin-gu, Seoul 05006, Republic of Korea.
Email: tnnguyen@sejong.ac.kr

Abstract
Nowadays, there are a lot of iterative algorithms which have been proposed for nonlinear problems of solid mechanics. The existing biggest drawback of iterative algorithms is the requirement of numerous iterations and computation to solve these problems. This can be found clearly when the large or complex problems with thousands or millions of degrees of freedom are solved. To overcome completely this difficulty, the novel one-iteration nonlinear solver (OINS) using time series prediction and the modified Riks method (M-R) is proposed in this paper. OINS is established upon the core idea as follows: (1) Firstly, we predict the load factor increment and the displacement vector increment and the convergent solution of the considering load step via the predictive networks which are trained by using the load factor and the displacement vector increments of the previous convergence steps and group method of data handling (GMDH); (2) Thanks to the predicted convergence solution of the load step is very close to or identical with the real one, the prediction phase used in any existing nonlinear solvers is eliminated completely in OINS. Next, the correction phase of the M-R is adopted and the OINS iteration is started at *the predicted convergence point* to reach the convergent solution. The training process and the applying process of GMDH are continuously conducted and repeated during the nonlinear analysis in order to predict the convergence point at the beginning of each load step. Through numerical investigations, we prove that OINS is powerful, highly accurate and only needs about one iteration per load step. Thus, OINS significantly saves number of iterations and a huge amount of computation compared with the conventional methods. Especially, OINS not only can detect limit, inflection, and other special points but also can predict exactly various types of instabilities of structures.

KEYWORDS
deep learning, nonlinear, one iteration, solver

1 | INTRODUCTION

For an arbitrary nonlinear analysis, an appropriate nonlinear solver should be chosen to solve the nonlinear equation as well as to trace completely the equilibrium path. In this regards, the Riks method[1,2] was proposed to trace completely nonlinear equilibrium paths even when they had a “snap-back” or “snap-through” but the method was not really appropriate to the conjunction with finite element analysis. Thereafter, the modified Riks method (M-R)[3] which could be readily implemented in any commercial finite element software was developed. As a matter of fact, M-R is powerful but still requires numerous iterations for a nonlinear analysis as found in Reference 3. To improve the robustness and the efficiency of the Newton method, MIP Newton method based on the relaxation of the constitutive equations at each integration point was proposed.[4,5] As the outstanding advantages of MIP Newton method, it can fast converge and withstand large increments. Especially, MIP Newton method can be well applied to high slenderness structures. Because of the importance of nonlinear solvers with respect to computational mechanics and their wide application, many solvers have been proposed to reduce number of iterations per load step and computation as the following: optimization-based iterative technique[6] and residual areas-based iterative technique,[7] dynamic relaxation techniques,[8-10] multipoint methods-based path following techniques,[11] a novel method to transform the discretized governing equations,[12] a data-driven nonlinear solver (DDNS),[13] an improved predictor-corrector method,[14] Koiter–Newton method with a superior performance for nonlinear analyses of structures,[15,16] etc. It is observed that employing time series prediction to reduce number of iterations of nonlinear solvers is very rare except the DDNS.[13] It is noted that DDNS only can save 40%–50% number of iterations per load step of an nonlinear analysis compared with the M-R.[13] This paper aims to minimize number of iterations per load step by proposing the novel one-iteration nonlinear solver (OINS) which is established by the M-R and GMDH. GMDH is a deep learning technique for time series prediction. It is noted that OINS can trace naturally and completely arbitrary nonlinear equilibrium paths even when they have a “snap-back” or “snap-through.”

Deep learning can be considered as the most popular field at this time. Its applications are found in both academia and industry such as: machine health monitoring,[17] fault-tolerant control,[18,19] natural language processing,[20] material design,[21,22] bankruptcy prediction,[23] structural engineering,[24] computer vision and pattern recognition,[25] Bayesian analysis for computational mechanics,[26,27] etc. Deep learning is a subfield of machine learning and established from a lot of algorithms.[28] Machine learning is the field which provides computers a general learning ability only from data.[29] In deep learning field, time series prediction is known as a very attractive and promising technique. Time series prediction was successfully investigated for stock market prediction, speech recognition, music recognition, etc. A comprehensive review of time series prediction can be found in Reference 30. In attempts to develop time series prediction, various types of networks have been proposed. Among them, long short-term memory (LSTM) network[31] was proposed and investigated successfully for making predictions, processing and classifying using time series data. Besides, convolutional neural network (CNN) is a type of deep learning network which has the adaptability[32-35] and group method of data handling (GMDH) was proposed as a self-organizing deep learning model.[36] GMDH is considered as a polynomial neural network and widely used in areas: pattern recognition, forecasting optimization, data mining, etc. GMDH networks possess some outstanding advantages such as: highly accurate prediction, self organization-based training procedure, excellent identification for nonlinear systems, etc. Especially, GMDH networks can be built and predict well upon its self-organization principle even with using a very small amount of data.[37] The applications of GMDH networks to solid mechanics can be found in References 13,38.

This paper aims to propose the novel OINS which is established by the M-R and the advantages of GMDH. To the best of authors knowledge, OINS is the most powerful solver which can minimize number of iterations per load step. Through numerical investigations, we prove that OINS is powerful and highly accurate. In the case of the equilibrium path is traced through limit or special points, the proposed solver might need two iterations per load step but number of average iteration per step for whole analysis is about one. Thus, OINS significantly saves number of iterations and a huge amount of computation compared with the existing solvers. The core idea as well as the numerical implementation of OINS and the main differences between OINS, DDNS[13] and the modified Riks method[3] are completely presented in Section 4.2. The paper is organized as follows: Section 2 presents analysis of shells using first-order shear deformation theory. Nonlinear isogeometric analysis of shells is provided in Section 3. The novel OINS based on GMDH is proposed in Section 4. Numerical results are presented and discussed in Section 5. Several notable conclusions are drawn in Section 6 to close the paper.

2 | NONLINEAR ANALYSIS OF ISOTROPIC SHELLS USING FIRST-ORDER SHEAR DEFORMATION THEORY

The aim of this paper is to verify the reliability and efficiency of the proposed solver (OINS) via geometrically nonlinear analysis of shells. The detailed formulation of the analysis can be found in Reference 39. The nonlinearity is established using the von Karman assumption and the Total Lagrangian approach. Firstly, a shell in Figure 1 is investigated. The strain vectors using first-order shear deformation shell theory (FSDT) are expressed as[39,40]

$$\begin{aligned}\boldsymbol{\varepsilon} &= \left\{\varepsilon_{xx} \quad \varepsilon_{yy} \quad \gamma_{xy}\right\}^T = \boldsymbol{\varepsilon}_0 + z\boldsymbol{\kappa}_b \\ \boldsymbol{\gamma} &= \left\{\gamma_{xz} \quad \gamma_{yz}\right\}^T = \boldsymbol{\varepsilon}_s,\end{aligned} \tag{1}$$

with

$$\begin{aligned}&\boldsymbol{\varepsilon}_0 = \boldsymbol{\varepsilon}_L + \boldsymbol{\varepsilon}_N; \quad \boldsymbol{\varepsilon}_L = \begin{Bmatrix} u_{0,x} + \frac{w_0}{R} \\ v_{0,y} \\ u_{0,y} + v_{0,x} \end{Bmatrix}; \quad \boldsymbol{\varepsilon}_N = \frac{1}{2}\begin{Bmatrix} w_{0,x}^2 \\ w_{0,y}^2 \\ 2w_{0,xy} \end{Bmatrix}; \\ &\boldsymbol{\kappa}_b = \begin{Bmatrix} \beta_{x,x} \\ \beta_{y,y} \\ \beta_{x,y} + \beta_{y,x} \end{Bmatrix}; \quad \boldsymbol{\varepsilon}_s = \begin{Bmatrix} -\frac{u_0}{R} + w_{0,x} + \beta_x \\ w_{0,y} + \beta_y \end{Bmatrix},\end{aligned} \tag{2}$$

and $\boldsymbol{\varepsilon}_N$ is the nonlinear strain vector which can be re-expressed as

$$\boldsymbol{\varepsilon}_N = \frac{1}{2}\mathbf{A}\boldsymbol{\theta}; \quad \mathbf{A} = \begin{bmatrix} w_{0,x} & 0 \\ 0 & w_{0,y} \\ w_{0,y} & w_{0,x} \end{bmatrix}; \quad \boldsymbol{\theta} = \begin{Bmatrix} w_{0,x} \\ w_{0,y} \end{Bmatrix}. \tag{3}$$

We assume that Ω is the initial configuration of the shell. According to the total Lagrangian approach, the virtual work equation is represented as

$$\int_{\Omega} \hat{\boldsymbol{\sigma}}^T \delta\hat{\boldsymbol{\varepsilon}} \mathrm{d}\Omega = \int_{\Omega} \delta\overline{\mathbf{u}}^T \mathbf{f}_s \mathrm{d}\Omega, \tag{4}$$

$\mathbf{f}_s = \{f_x \quad f_y \quad f_z\}^T$ denotes the external load vector. In addition, $\overline{\mathbf{u}}^T = \{u_x \quad u_y \quad u_z\}$ and $u_z = w_0$ with w_0 is the radial deflection. When the shell is only subjected to a radial load $f_z = \lambda f_0$, the virtual work equation can be rewritten as follows

$$\int_{\Omega} \hat{\boldsymbol{\sigma}}^T \delta\hat{\boldsymbol{\varepsilon}} \mathrm{d}\Omega = \lambda \int_{\Omega} \delta w_0 f_0 \mathrm{d}\Omega, \tag{5}$$

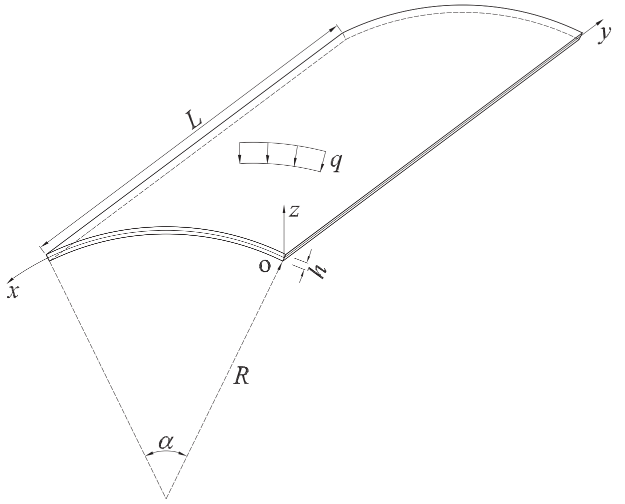


FIGURE 1 Geometry and load description of a panel

λ is the load factor and $\hat{\boldsymbol{\sigma}}$ denotes the stress resultant vector which is computed as follows

$$\hat{\boldsymbol{\sigma}} = \left\{\boldsymbol{\sigma}_p \quad \boldsymbol{\sigma}_b \quad \boldsymbol{\sigma}_s\right\}^T, \tag{6}$$

with the in-plane stress vector

$$\boldsymbol{\sigma}_p = \{N_x \quad N_y \quad N_{xy}\}^T = \left\{\int_{-h/2}^{h/2} (\sigma_x \quad \sigma_y \quad \tau_{xy})\mathrm{d}z\right\}^T, \tag{7}$$

the bending stress vector

$$\boldsymbol{\sigma}_b = \{M_x \quad M_y \quad M_{xy}\}^T = \left\{\int_{-h/2}^{h/2} (\sigma_x \quad \sigma_y \quad \tau_{xy})z\mathrm{d}z\right\}^T, \tag{8}$$

the shear stress vector

$$\boldsymbol{\sigma}_s = \{Q_x \quad Q_y\}^T = \left\{\int_{-h/2}^{h/2} (\tau_{xz} \quad \tau_{yz})\mathrm{d}z\right\}^T. \tag{9}$$

The relation between the generalized strain vector $\hat{\boldsymbol{\varepsilon}}$ and the stress resultant vector $\hat{\boldsymbol{\sigma}}$ can be represented through Hooke's law as follows

$$\hat{\boldsymbol{\sigma}} = \hat{\boldsymbol{D}}\hat{\boldsymbol{\varepsilon}}; \quad \hat{\boldsymbol{D}} = \begin{bmatrix} \mathbf{D}^p & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}^b & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}^s \end{bmatrix}; \quad \hat{\boldsymbol{\varepsilon}} = \begin{Bmatrix} \boldsymbol{\varepsilon}_L \\ \boldsymbol{\kappa}_b \\ \boldsymbol{\varepsilon}_s \end{Bmatrix} + \begin{Bmatrix} \boldsymbol{\varepsilon}_N \\ \mathbf{0} \\ \mathbf{0} \end{Bmatrix}, \tag{10}$$

with

$$\mathbf{D}^p = \frac{Eh}{1-\upsilon^2}\overline{\mathbf{D}}; \quad \mathbf{D}^b = \frac{Eh^3}{12(1-\upsilon^2)}\overline{\mathbf{D}}; \quad \mathbf{D}^s = \kappa\frac{Eh}{2(1+\upsilon)}\mathbf{I}, \tag{11}$$

and

$$\overline{\mathbf{D}} = \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix}; \quad \mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{12}$$

Note that $\kappa = 5/6$ denotes the shear correction factor[41-45] while ν is Poisson's ratio, E is Young's modulus, and h is the thickness of the shell.

3 | NONLINEAR ISOGEOMETRIC ANALYSIS OF SHELLS

This section briefly presents geometrically nonlinear analysis of shells based on isogeometric analysis and FSDT. As mentioned earlier, the detailed formulation of the analysis is found in Reference 39 for nonlinear analysis of a functionally graded carbon nanotube-reinforced composite (FG-CNTRC) shell. As known, an isotropic shell can be considered as a particular case of a FG-CNTRC shell. Thus, the present formulation for isotropic shells is identical with the formulation for FG-CNTRC shells in References 39 except a minor difference in computing the material matrices as shown in Equations (10)–(12). At ith iteration and mth load increment, a system of linear incremental equations is expressed as

$$\mathbf{K}_T(\mathbf{q}_m)\Delta^i\mathbf{q}_m = {}^i\mathbf{F}_{\mathrm{ext},m} - {}^i\mathbf{F}_{\mathrm{int},m}, \tag{13}$$

with

$$\mathbf{K}_T = \int_\Omega \left[\begin{Bmatrix} \mathbf{B}_A^L \\ \mathbf{B}_A^b \\ \mathbf{B}_A^s \end{Bmatrix} + \begin{Bmatrix} \mathbf{B}_A^N \\ \mathbf{0} \\ \mathbf{0} \end{Bmatrix} \right]^T \begin{bmatrix} \mathbf{D}^p & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}^b & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}^s \end{bmatrix} \left[\begin{Bmatrix} \mathbf{B}_A^L \\ \mathbf{B}_A^b \\ \mathbf{B}_A^s \end{Bmatrix} + \begin{Bmatrix} \mathbf{B}_A^N \\ \mathbf{0} \\ \mathbf{0} \end{Bmatrix} \right] d\Omega + \int_\Omega \left(\mathbf{B}_A^g\right)^T \begin{bmatrix} N_x & N_{xy} \\ N_{xy} & N_y \end{bmatrix} \mathbf{B}_A^g d\Omega, \tag{14}$$

the load vector is expressed as

$${}^i\mathbf{F}_{\text{ext},m} = ({}^i\lambda_m + \Delta^i\lambda_m) \int_\Omega f_0 \begin{Bmatrix} 0 & 0 & N_A & 0 & 0 \end{Bmatrix}^T d\Omega = ({}^i\lambda_m + \Delta^i\lambda_m)\mathbf{F}_0. \tag{15}$$

$\mathbf{F}_0$ denotes the referenced load vector while N_A stands for the NURBS (nonuniform rational B-spline) basic function. We compute the internal force as follows[39]

$${}^i\mathbf{F}_{\text{int},m} = {}^i\mathbf{K}_m {}^i\mathbf{q}_m, \tag{16}$$

with

$${}^i\mathbf{K}_m = \int_\Omega \left[\begin{Bmatrix} \mathbf{B}_A^L \\ \mathbf{B}_A^b \\ \mathbf{B}_A^s \end{Bmatrix} + \begin{Bmatrix} \mathbf{B}_A^N \\ \mathbf{0} \\ \mathbf{0} \end{Bmatrix} \right]^T \begin{bmatrix} \mathbf{D}^p & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}^b & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}^s \end{bmatrix} \left[\begin{Bmatrix} \mathbf{B}_A^L \\ \mathbf{B}_A^b \\ \mathbf{B}_A^s \end{Bmatrix} + 0.5 \begin{Bmatrix} \mathbf{B}_A^N \\ \mathbf{0} \\ \mathbf{0} \end{Bmatrix} \right] d\Omega. \tag{17}$$

In this study, we use the novel OINS to solve the nonlinear equation. For any load increments, an iterative process is conducted and the iteration can be stopped when the convergence criterion is met as follows

$$e = \frac{\left\| {}^i\lambda_m \mathbf{F}_0 - {}^i\mathbf{F}_{\text{int},m} \right\|}{\left\| ({}^i\lambda_m + \Delta^i\lambda_m)\mathbf{F}_0 \right\|} < 10^{-3}. \tag{18}$$

We obtain the incremental solutions via solving Equation (13). Thereafter, the load factor λ as well as the displacement vector $\mathbf{q}$ of the iteration are computed and updated as follows[3]

$$\begin{aligned} {}^{i+1}\lambda_m &= {}^i\lambda_m + \Delta^i\lambda_m \\ {}^{i+1}\mathbf{q}_m &= {}^i\mathbf{q}_m + \Delta^i\mathbf{q}_m \\ \Delta^i\mathbf{q}_m &= \Delta^i\mathbf{q}_{R,m} + \Delta^i\lambda_m \mathbf{q}_{F,m}, \end{aligned} \tag{19}$$

$\mathbf{q}_{F,m}$ denotes a displacement vector created by a reference force vector while $\Delta^i\mathbf{q}_{R,m}$ stands for that caused by the residual load vector as follows

$$\begin{aligned} \Delta^i\mathbf{q}_{R,m} &= [\mathbf{K}_T(\mathbf{q}_m)]^{-1}({}^i\lambda_m\mathbf{F}_0 - {}^i\mathbf{F}_{\text{int},m}) \\ \mathbf{q}_{F,m} &= [\mathbf{K}_T(\mathbf{q}_m)]^{-1}\mathbf{F}_0. \end{aligned} \tag{20}$$

4 | THE NOVEL OINS BASED ON GMDH

4.1 | Group method of data handling

4.1.1 | An introduction

GMDH[36,46] is considered as a self-organizing deep learning method which is widely used for time series prediction problems. Difference from other deep learning networks, during the training stage number of neurons of the GMDH network continually changes to improve the performance of the network. For the mathematical description of GMDH, we investigate a nonlinear relation as

$$\phi = f(x_1, x_2, \dots, x_n), \tag{21}$$

with $x_1, x_2, \dots, x_n$ and ϕ respectively denote the input and output of the system while f is known as the nonlinear connection function. Using the Kolmogorov–Gabor form, Equation (21) is rewritten as follows[36,46]

$$\phi = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k + \dots \tag{22}$$

Because of each partial description (or neuron) takes two inputs and it delivers only one output, number of neurons in the first layer can be determined as follows[36,46]

$$m = C_2^n = \frac{n^2 - n}{2}, \tag{23}$$

n denotes number of inputs. In addition, outputs of these neurons are determined as follows

$$\begin{aligned} y_{11} &= f_{11}(x_1, x_2) \\ y_{12} &= f_{12}(x_1, x_3) \\ &\dots.. \\ y_{1m} &= f_{1m}(x_{n-1}, x_n). \end{aligned} \tag{24}$$

After training the first layer, we apply the selection criterion to eliminate the neurons that give the poorest predictions while ones that meet the criterion are kept. It should be noted that the selected neurons are considered as a subset of the original ones as

$$\begin{aligned} &\left[\hat{y}_{11}, \hat{y}_{12}, \dots, \hat{y}_{1\hat{m}}\right] \subset \left[y_{11}, y_{12}, \dots, y_{1m}\right]; \\ &\left[\hat{f}_{11}, \hat{f}_{12}, \dots, \hat{f}_{1\hat{m}}\right] \subset \left[f_{11}, f_{12}, \dots, f_{1m}\right], \end{aligned} \tag{25}$$

$\hat{m}$ denotes number of the selected neurons in the first layer ($\hat{m} < m$). After selecting the best neurons in the first layer, we add the second layer to the network with number of neurons in this new layer $p = C_2^{\hat{m}}$. Outputs of the second layer are also computed similar to Equation (24) as follows

$$\begin{aligned} y_{21} &= f_{21}(\hat{y}_{11}, \hat{y}_{12}) \\ y_{22} &= f_{22}(\hat{y}_{11}, \hat{y}_{13}) \\ &\dots.. \\ y_{2p} &= f_{2p}(\hat{y}_{1(\hat{m}-1)}, \hat{y}_{1\hat{m}}). \end{aligned} \tag{26}$$

Selecting the best neurons in a layer and adding a new layer to the network are continuously performed. When the stop criteria provided in Section 4.1.3 are met, the training procedure finishes.

4.1.2 | Calculate coefficients of partial descriptions

As mentioned earlier, each neuron receives two inputs while it produces one output. We assume that two inputs are x_i and x_j, the output of that neuron is determined via the following partial description[36,46]

$$y = a_0 + a_1 x_i + a_2 x_j + a_3 x_i^2 + a_4 x_j^2 + a_5 x_i x_j, \tag{27}$$

or rewritten in the following matrix form

$$y = \mathbf{xA}, \tag{28}$$

with

$$\mathbf{x} = \left\{ 1 \quad x_i \quad x_j \quad x_i^2 \quad x_j^2 \quad x_i x_j \right\};$$
$$\mathbf{A} = \left\{ a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \right\}^T. \tag{29}$$

As seen in Equation (27), the output can be determined via the inputs and the coefficients of partial description. In order to compute these coefficients, we consider a time series prediction problem and assume that there are s samples achieved from a system in Equation (21) as follows

$$\begin{aligned} \phi(1) &= f(x_1(1), x_2(1), \dots, x_n(1)) \\ \phi(2) &= f(x_1(2), x_2(2), \dots, x_n(2)) \\ &\dots.. \\ \phi(s) &= f(x_1(s), x_2(s), \dots, x_n(s)). \end{aligned} \tag{30}$$

At time t, the neuron takes two inputs $x_i(t), x_j(t)$ and its output is determined as follows

$$y(t) = a_0 + a_1 x_i(t) + a_2 x_j(t) + a_3 x_i^2(t) + a_4 x_j^2(t) + a_5 x_i(t) x_j(t), \tag{31}$$

with $t = 1, 2, \dots, s$. We make $y(t) = \phi(t)$ and Equation (31) is re-expressed as

$$\boldsymbol{\phi} = \mathbf{XA}, \tag{32}$$

and

$$\mathbf{X} = \begin{bmatrix} 1 & x_i(1) & x_j(1) & x_i^2(1) & x_j^2(1) & x_i(1)x_j(1) \\ 1 & x_i(2) & x_j(2) & x_i^2(2) & x_j^2(2) & x_i(2)x_j(2) \\ \dots. & & & & & \\ 1 & x_i(s) & x_j(s) & x_i^2(s) & x_j^2(s) & x_i(s)x_j(s) \end{bmatrix}. \tag{33}$$

$$\boldsymbol{\phi} = \left\{ \phi(1) \quad \phi(2) \quad \dots \quad \phi(s) \right\}^T. \tag{34}$$

From Equation (32), the coefficients of partial description can be achieved via the following equation

$$\mathbf{A} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{\phi}. \tag{35}$$

4.1.3 | Selection and stop criteria

When the neurons outputs are known, the selection criterion upon the mean squared error (MSE) is applied to select the best neurons in each layer. According to this criterion, the neurons that give the poorest results are removed and ones that fit the criterion are kept. To this end, all the data are separated into two subsets including the training dataset and the testing dataset. Then, the selection process is conducted over the testing dataset. MSE of an output is determined as[36,46]

$$\text{MSE}(y_{ln}) = \frac{1}{s} \sum_{t=1}^{s} (y_{ln}(t) - \phi(t))^2, \tag{36}$$

where s denotes number of samples of a system. $y_{ln}(t)$ expresses the neuron's output at time t while $\phi(t)$ denotes the target of network. In addition, the neurons which provide the least error are chosen for the next step.

For stop criterion, the GMDH training process includes: (1) Add layers; (2) Compute coefficients of partial descriptions and then outputs; (3) Remove neurons that give poor predictions. In the training stage, the obtained outputs of the present layer are recognized as the inputs of the subsequent layer. The training stage can be stopped when the considering layer only remains one neuron after applying the selection step or when the new layer which is added and trained does not improve the performance of the network. In this case, the new layer is removed. Then, in the previous layer the neuron that has the best prediction is kept and the rest neurons are removed. Finally, the *trimming* step is conducted to obtain the final network configuration.[46,47] As the illustration, the training process of GMDH using four inputs is described in Figure 2. Note that the removed neurons are drawn in the lighter color.

4.2 | The novel OINS

In each load step, M-R begins its iterations at the conventional starting point (or the previously converged solution point). As known, M-R requires many iterations and computation to reach the convergent solution. To overcome completely this difficulty, the novel OINS using M-R and time series prediction is proposed in this paper. Some existing nonlinear solvers have the predictor phase using an extrapolation of the previously evaluated equilibrium points.[13,14] The key concept of these solvers is: iterations of each load step are begun at *the new starting point* which is determined by machine learning and the previously evaluated equilibrium points. However, these solvers still require many iterations and computational efforts to reach the converged solution of each load step.[13,14] To overcome this drawback in this paper, OINS begins its iterations at *the predicted convergence point* which is very close to or coincides with the real convergence point of each step. For illustrations, iterative processes of M-R and OINS are described in Figure 3. OINS is established upon the core idea as follows: (1) Firstly, we predict the load factor and the displacement vector increments and the convergent solution of the current load step via the predictive networks which are trained by using the load factor and the displacement vector increments of the previous convergence steps and GMDH; (2) Thanks to the predicted convergence solution of the load step is very close to or identical with the real one, the prediction phase used in any existing nonlinear solvers

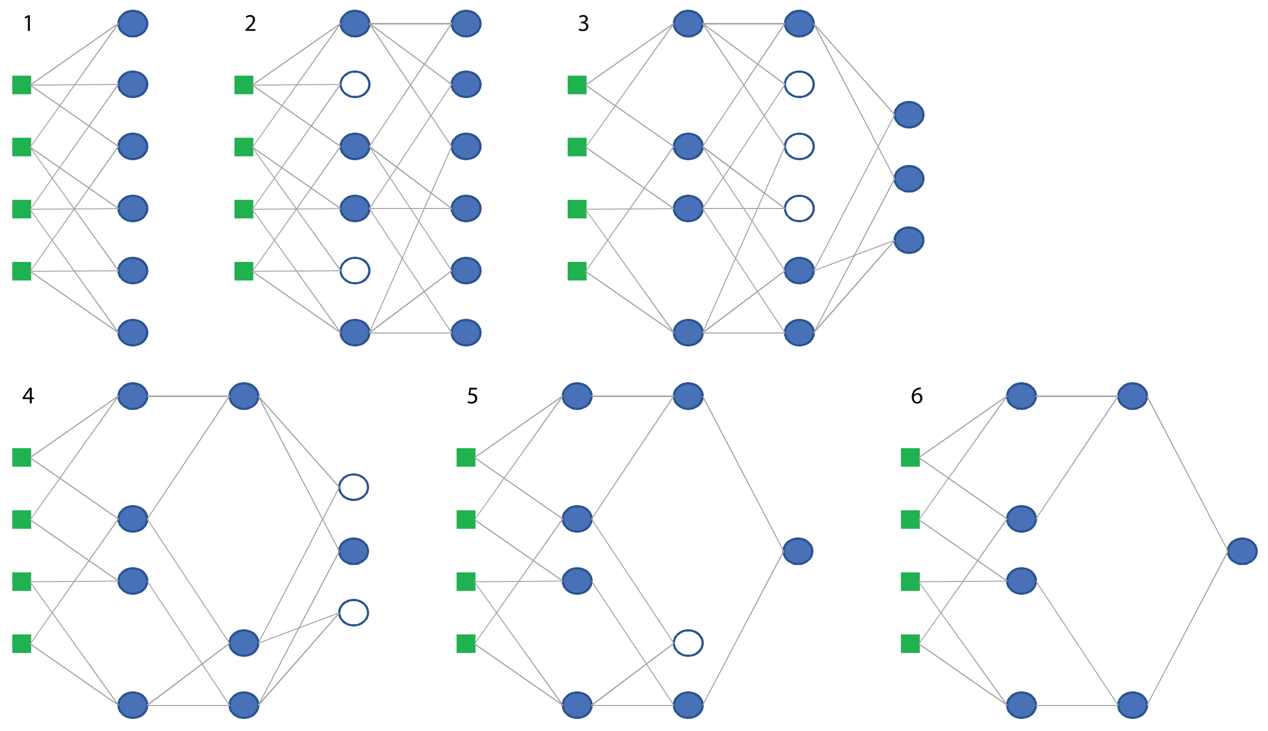
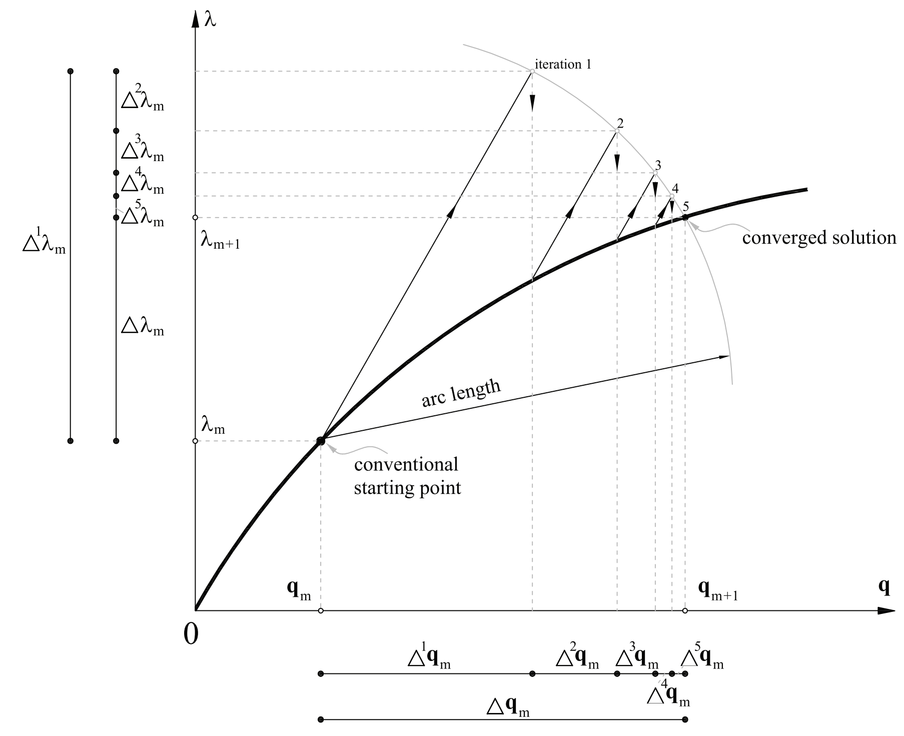
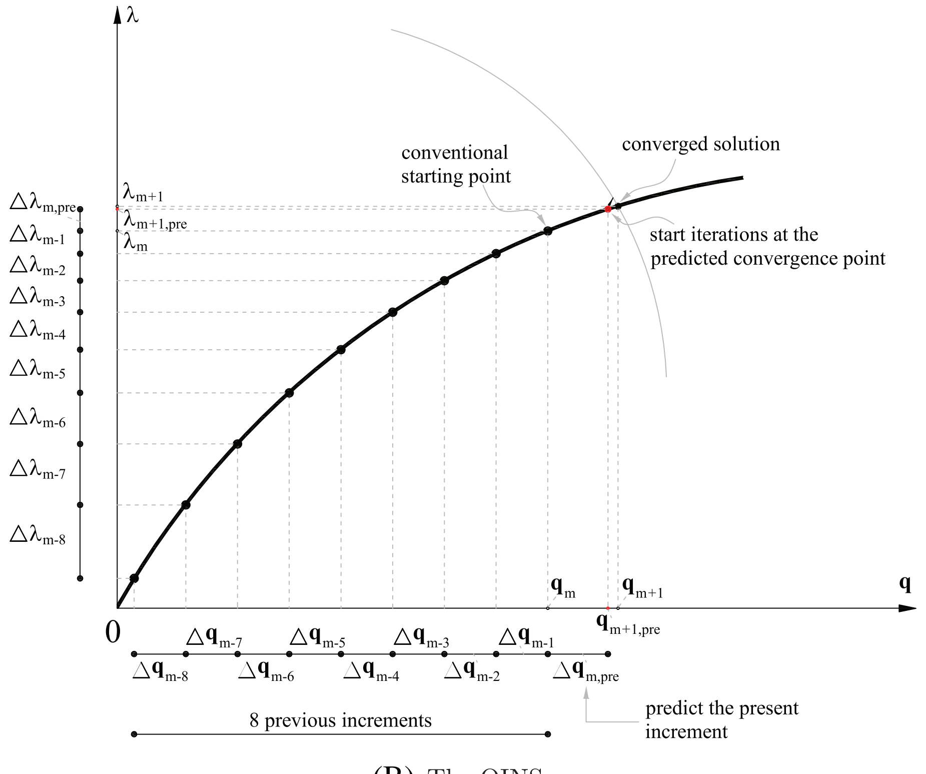


FIGURE 2 A self-organizing deep learning network: group method of data handling



(A) The modified Riks method


λ
conventional starting point
converged solution
start iterations at the predicted convergence point
$\triangle\lambda_{m,pre}$
$\triangle\lambda_{m-1}$
$\triangle\lambda_{m-2}$
$\triangle\lambda_{m-3}$
$\triangle\lambda_{m-4}$
$\triangle\lambda_{m-5}$
$\triangle\lambda_{m-6}$
$\triangle\lambda_{m-7}$
$\triangle\lambda_{m-8}$
λ_{m+1}
$\lambda_{m+1,pre}$
λ_m
0
$\mathbf{q}_m$
$\mathbf{q}_{m+1}$
$\mathbf{q}$
$\mathbf{q}_{m+1,pre}$
$\triangle\mathbf{q}_{m-7}$
$\triangle\mathbf{q}_{m-5}$
$\triangle\mathbf{q}_{m-3}$
$\triangle\mathbf{q}_{m-1}$
$\triangle\mathbf{q}_{m-8}$
$\triangle\mathbf{q}_{m-6}$
$\triangle\mathbf{q}_{m-4}$
$\triangle\mathbf{q}_{m-2}$
$\triangle\mathbf{q}_{m,pre}$
predict the present increment
8 previous increments

(B) The OINS

FIGURE 3 Iterative processes of the modified Riks method (A) and one-iteration nonlinear solver (B)

is eliminated completely in OINS. Next, the correction phase of M-R is adopted and the iteration is started at the predicted convergence point to reach the convergent solution. The training process and the applying process of GMDH are continuously conducted and repeated during analysis to predict the convergence point at the beginning of each load step.

Recently, the DDNS[13] has been developed to reduce number of iterations of nonlinear problems. In Reference 13, it is found that DDNS can save 40%–50% number of iterations of an analysis compared with the modified Riks method. In this paper, numerical procedures of DDNS and the proposed method (OINS) at a typical load step are presented in Figure 4. The main differences between DDNS and OINS are as follows: (1) OINS starts iterations at the predicted convergence point $(\lambda_{m+1,\mathrm{pre}}, \mathbf{q}_{m+1,\mathrm{pre}})$ while DDNS starts iterations at the new starting point $(\lambda_{m,\mathrm{new}}, \mathbf{q}_{m,\mathrm{new}})$; (2) The prediction phase used in any existing nonlinear solvers involving DDNS is eliminated completely in OINS. Accordingly, the flowchart of OINS is more simple and straightforward than that of DDNS. Especially, number of iterations per load step of OINS is much lower than that of DDNS. The solvers including M-R, DDNS, and OINS start iterations at various points as seen in Figure 5.

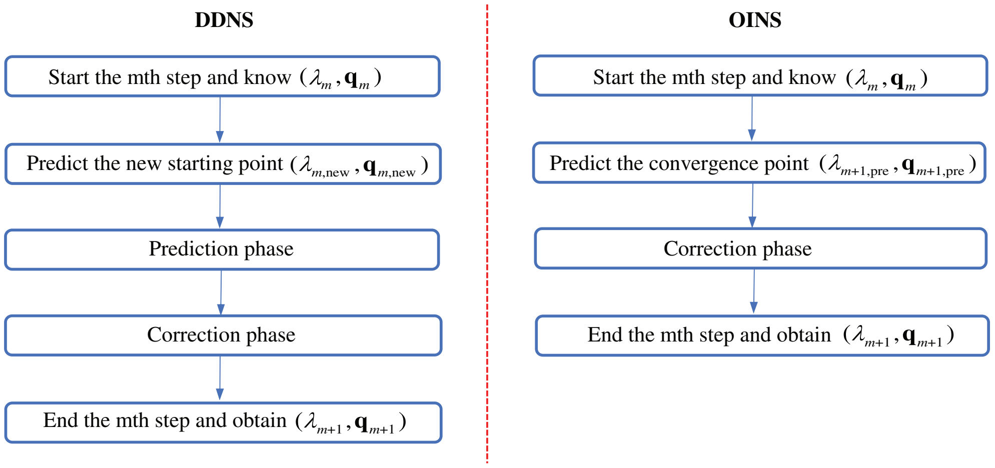

DDNS
Start the mth step and know $(\lambda_m, \mathbf{q}_m)$
Predict the new starting point $(\lambda_{m,\mathrm{new}}, \mathbf{q}_{m,\mathrm{new}})$
Prediction phase
Correction phase
End the mth step and obtain $(\lambda_{m+1}, \mathbf{q}_{m+1})$
OINS
Start the mth step and know $(\lambda_m, \mathbf{q}_m)$
Predict the convergence point $(\lambda_{m+1,\mathrm{pre}}, \mathbf{q}_{m+1,\mathrm{pre}})$
Correction phase
End the mth step and obtain $(\lambda_{m+1}, \mathbf{q}_{m+1})$

FIGURE 4 Numerical procedures of the data-driven nonlinear solver (DDNS)[13] and one-iteration nonlinear solver at a typical step

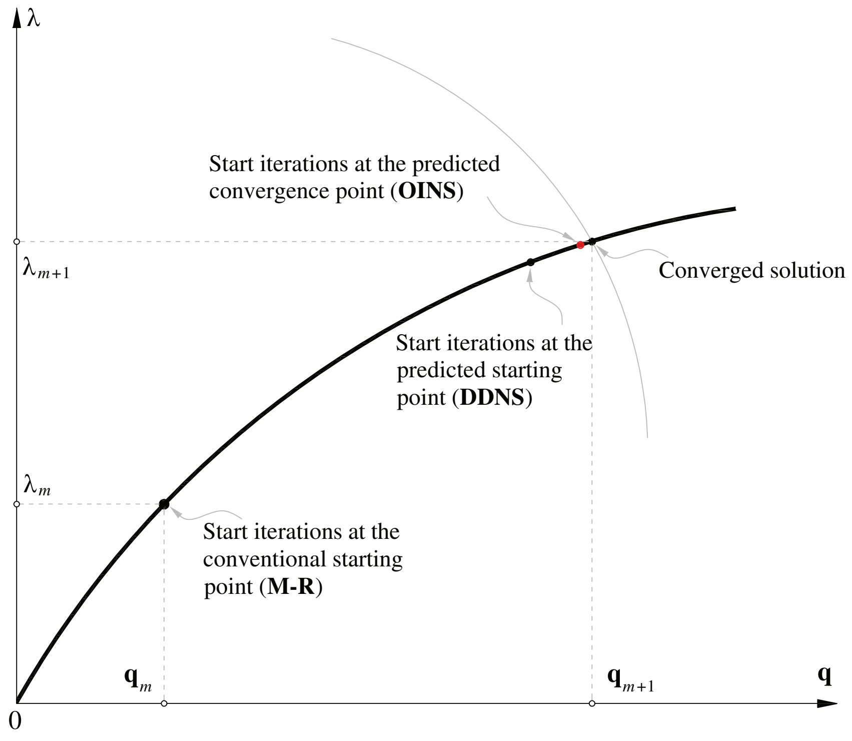

λ
Start iterations at the predicted convergence point (OINS)
λ_{m+1}
Converged solution
Start iterations at the predicted starting point (DDNS)
λ_m
Start iterations at the conventional starting point (M-R)
$\mathbf{q}_m$
$\mathbf{q}_{m+1}$
$\mathbf{q}$
0

FIGURE 5 Modified Riks (M-R), data-driven nonlinear solver, and one-iteration nonlinear solver start iterations at various points

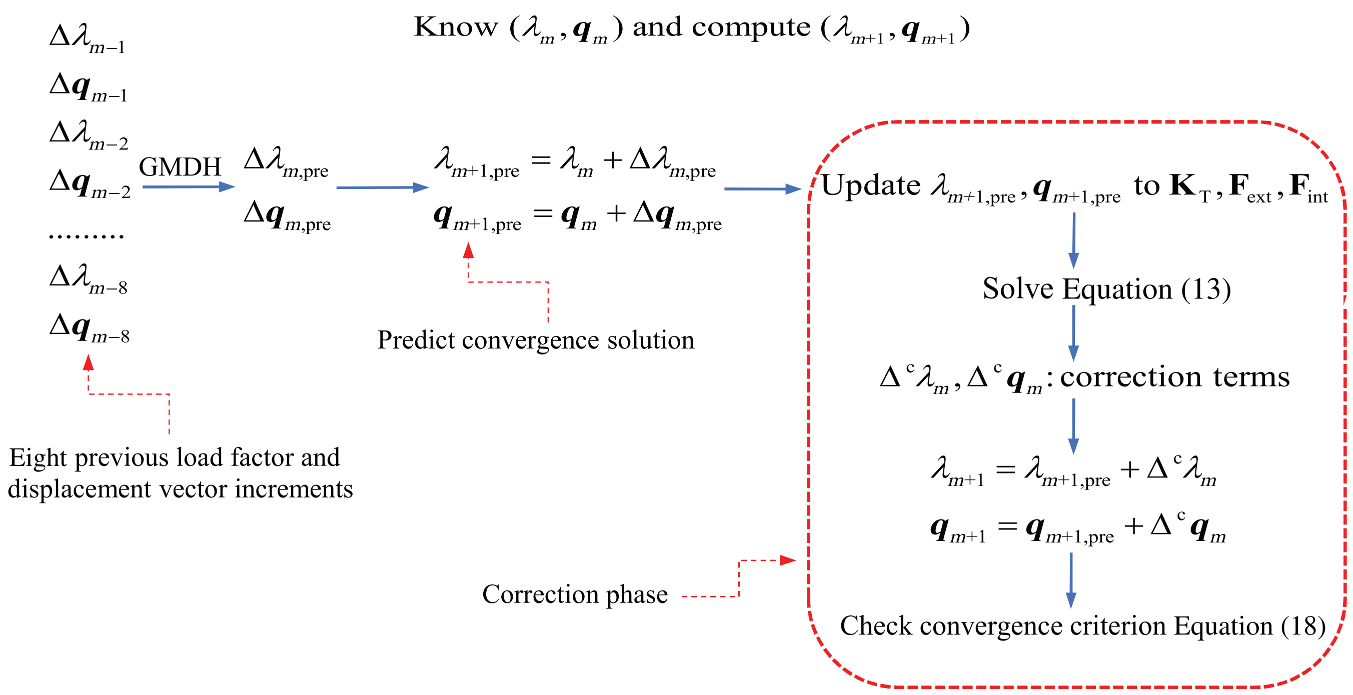

Know $(\lambda_m, \boldsymbol{q}_m)$ and compute $(\lambda_{m+1}, \boldsymbol{q}_{m+1})$
$\Delta\lambda_{m-1}$
$\Delta\boldsymbol{q}_{m-1}$
$\Delta\lambda_{m-2}$
$\Delta\boldsymbol{q}_{m-2}$
....
$\Delta\lambda_{m-8}$
$\Delta\boldsymbol{q}_{m-8}$
GMDH
$\Delta\lambda_{m,\mathrm{pre}}$
$\Delta\boldsymbol{q}_{m,\mathrm{pre}}$
$\lambda_{m+1,\mathrm{pre}} = \lambda_m + \Delta\lambda_{m,\mathrm{pre}}$
$\boldsymbol{q}_{m+1,\mathrm{pre}} = \boldsymbol{q}_m + \Delta\boldsymbol{q}_{m,\mathrm{pre}}$
Predict convergence solution
Eight previous load factor and displacement vector increments
Update $\lambda_{m+1,\mathrm{pre}}, \boldsymbol{q}_{m+1,\mathrm{pre}}$ to $\mathbf{K}_\mathrm{T}, \mathbf{F}_\mathrm{ext}, \mathbf{F}_\mathrm{int}$
Solve Equation (13)
$\Delta^c\lambda_m, \Delta^c\boldsymbol{q}_m$: correction terms
$\lambda_{m+1} = \lambda_{m+1,\mathrm{pre}} + \Delta^c\lambda_m$
$\boldsymbol{q}_{m+1} = \boldsymbol{q}_{m+1,\mathrm{pre}} + \Delta^c\boldsymbol{q}_m$
Check convergence criterion Equation (18)
Correction phase

FIGURE 6 Detailed flowchart of one-iteration nonlinear solver

As described in this figure, OINS starts iterations at the closest point to the convergence point of each load step. This is numerically demonstrated in Section 5 via OINS only needs about one iteration to reach the convergent solution of each load step. In addition, OINS significantly saves number of iterations (about 77%–80%) and a huge amount of computation compared with the modified Riks method. Finally, Figure 6 shows the detailed flowchart of OINS corresponding to the core idea as mentioned earlier.

5 | NUMERICAL RESULTS

In this section, the reliability and efficiency of the proposed method (OINS) is verified via solving nonlinear problems which have different types of equilibrium paths. Abilities of OINS to detect limit points and to predict various types of instabilities are investigated. All the GMDH networks in this paper are trained using the following parameters: five hidden layers, two input data groups, the highest number of neurons in each layer is 25 and eight datasets generated from eight previous increments. 95% of the datasets are used as the training set while 5% of these sets are used as the testing set. With these parameters, we demonstrate numerically that the trained networks give good and stable predictions in the below problems. It should be emphasized that OINS bases on networks and all networks of deep learning need the available data for training. Accordingly, the first eight load steps of an arbitrary analysis in this study are conducted using the M-R solver for generating data. Then, networks are built based on the generated data and OINS is begun from the ninth load step to the end of the analysis. Therefore, the comparisons of computational efficiency between M-R and OINS in the below tables are conducted from the ninth load step. Some below typical problems are solved to prove the high reliability, efficiency and ability of OINS. It is noted that OINS can be used for any problems relating to the nonlinearity. All the below problems are solved using 14×14 cubic elements and 4×4 Gauss points for each element. The material properties are given as follows: Poisson's ratio $\nu = 0.3$ and Young's modulus $E = 3.103\ \ \mathrm{kN/mm^2}$. Hinged and clamped boundaries in numerical models are simply described as

- Hinged

$$u_0 = v_0 = w_0 = \beta_y = 0 \quad \text{at} \ \ x = 0, \alpha R. \tag{37}$$

- Clamped

$$u_0 = v_0 = w_0 = \beta_x = \beta_y = 0. \tag{38}$$

5.1 | Evaluate performance of GMDH networks and accuracy of OINS

We consider a hinged panel subjected to a point load at the center as illustrated in Figure 7. The geometry of panel is described as follows: the length $L = B = 508$ mm, the radius $R = 2540$ mm and the thickness $h = 12.7$ mm. Firstly, we evaluate performance of GMDH for the proposed method via training and applying a following typical network. For data generation, we analyze the panel with eight load steps and obtain eight load factor increments ($\Delta\lambda$) which are considered as eight datasets for training the network. Next, we separate the datasets into two groups: *Train data* including six first data sets $(1 - 6)$ and *Test data* including two next datasets $(7 - 8)$. The training process and the applying process of the GMDH network using two input data groups (or two samples $s = 2$) are summarized in Figure 8. It should be noted that: the more number of input data groups, the higher accuracy of forecasting but the higher computational cost. In this paper, number of input data groups are fixed at two for all networks with the balance between accuracy and computational efficiency. The training process and the applying process are explained briefly as follows: The network is trained using the target data from *Train data* and two input data groups; The output data is obtained via applying the trained network. Then, we compare the obtained output with the target data to evaluate the accuracy of the prediction. For convenience in comparison, both the output data and the target data are divided into two small groups: one group belongs to *Train data*, the rest one belongs to *Test data* as presented in Figure 8. It is interesting that total time for training and applying of the network is extremely low (0.0019 s). This thanks to the very small data is used and the advantages of a self-organizing deep learning network (GMDH). Especially, a very good prediction of the trained network is obtained and presented in Figure 9. High performance of GMDH for the proposed method is verified.

To evaluate the accuracy of the proposed method (OINS), we continue analyzing the hinged panel under a central point load (the above problem). Eight first load steps are performed to obtain eight load factor increments ($\Delta\lambda_0 \sim \Delta\lambda_7$) as presented in Table 1. Then, eight load factor increments are considered as eight datasets for training the GMDH network. Next, we calculate the predicted load factor increment of the ninth load step ($\Delta\lambda_{8,\text{pre}}$) by applying the trained network as shown in Table 1. To evaluate the accuracy of OINS, we typically compute load factor of the ninth load step (λ_9) from the known eighth step using the modified Riks method and the OINS. The obtained results from these two methods are presented in Table 2. It is seen that error of results from these two methods is 0%. Especially, the modified Riks method requires five iterations to reach the convergence solution of the load step while OINS only needs one iteration to converge. It is interesting that the predicted convergence solution is as same as the real convergence solution ($\lambda_{9,\text{pre}} = \lambda_9 = 0.21419$). In addition, using OINS we can save 80% number of iterations compared with M-R. High efficiency and accuracy of OINS are confirmed.

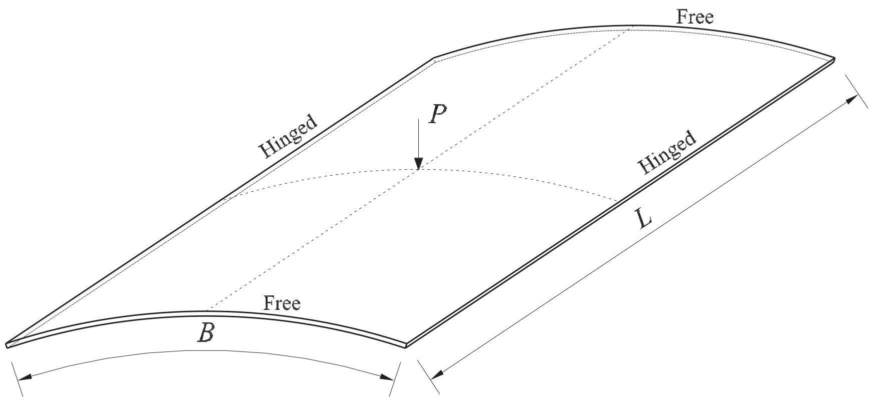


FIGURE 7 A cylindrical panel under a point load

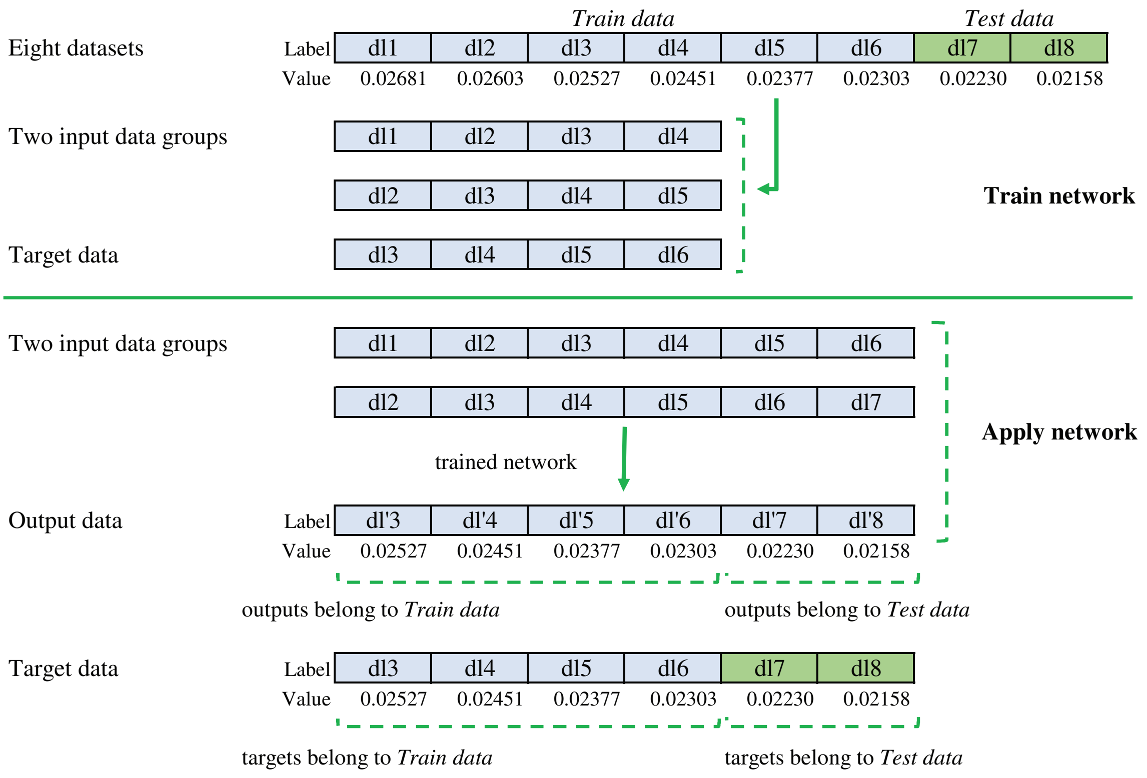
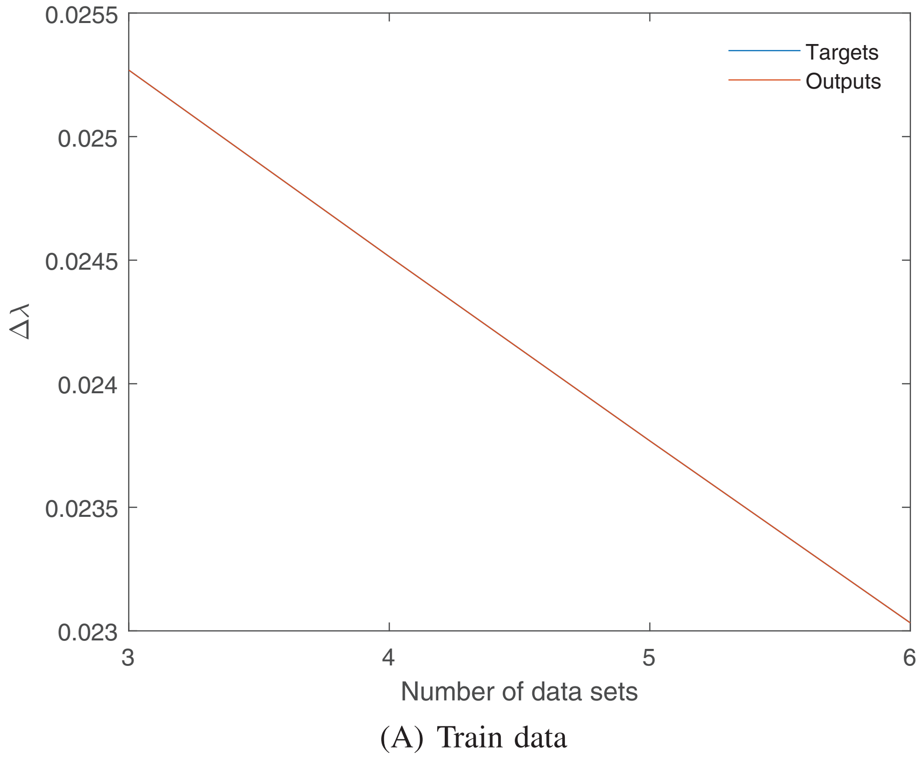


FIGURE 8 Train and apply a group method of data handling network using two input data groups, dl: delta lambda ($\Delta\lambda$)

5.2 | Predict various types of instabilities using OINS

In this section, abilities of OINS to detect limit points and to predict various types of instabilities are investigated. The hinged panel under a central point load in Figure 7 is continued analyzing. As mentioned earlier, the geometry of the panel is described as: $L = B = 508$ mm, $R = 2540$ mm and $h = 12.7$ mm. As seen in Figure 10, the obtained result agrees well with the result of the modified Riks method combined with the present formulation, with that of Crisfield[3] and Sabir,[48] respectively, using the modified Riks and Newton–Raphson methods in the framework of finite element analysis. It is observed that OINS can detect limit points and predict exactly "snap-through type of instability." In addition, Table 3 shows number of average iteration of M-R and OINS for analyzing panel with different arc lengths. It is interesting that number of average iteration of OINS is only one iteration per load step while that of M-R are 4.4–4.6 iterations per load step. It can be concluded that OINS significantly saves number of iterations (about 77%–78%) and saves a huge amount of computation compared with M-R. Again, high efficiency and accuracy of OINS are confirmed. Figure 11 shows the response of the panel when the thickness $h = 25.4$ mm. The obtained result coincides with that of M-R using the same formulation. It is observed that OINS can detect the inflection point and predict exactly "softening-hardening type of instability." From Figures 10 and 11, it is confirmed that the higher thickness, the higher stiffness and bending strength of the panel. Table 4 presents number of average iteration of M-R and OINS for analyzing panel with $h = 25.4$ mm and various arc lengths. Again, it is seen that number of average iteration of OINS is only one iteration per load step while that of M-R are 4.7–4.8 iterations per load step. It can be concluded that OINS significantly saves number of iterations (about 79%) and saves a huge amount of computation compared with the modified Riks method. Figures 12 and 13, respectively, show the responses of the panel when $L = 1.6B$, $h = 6.35$ mm with various arc lengths. It should be noted that in the last case $h = 6.35$ mm, the boundary is clamped-hinged. In both figures, the present results coincide with that of M-R. High accuracy of OINS is confirmed. It is observed that OINS can predict exactly "snap-back type of instability" in these two cases. In both cases, number of average iteration of OINS is about one iteration per load step as presented in Tables 5 and 6. It is concluded that the proposed method (OINS) has a high reliability and accuracy for geometrically nonlinear problems. OINS significantly saves number of iterations and a huge amount of computation compared with the modified



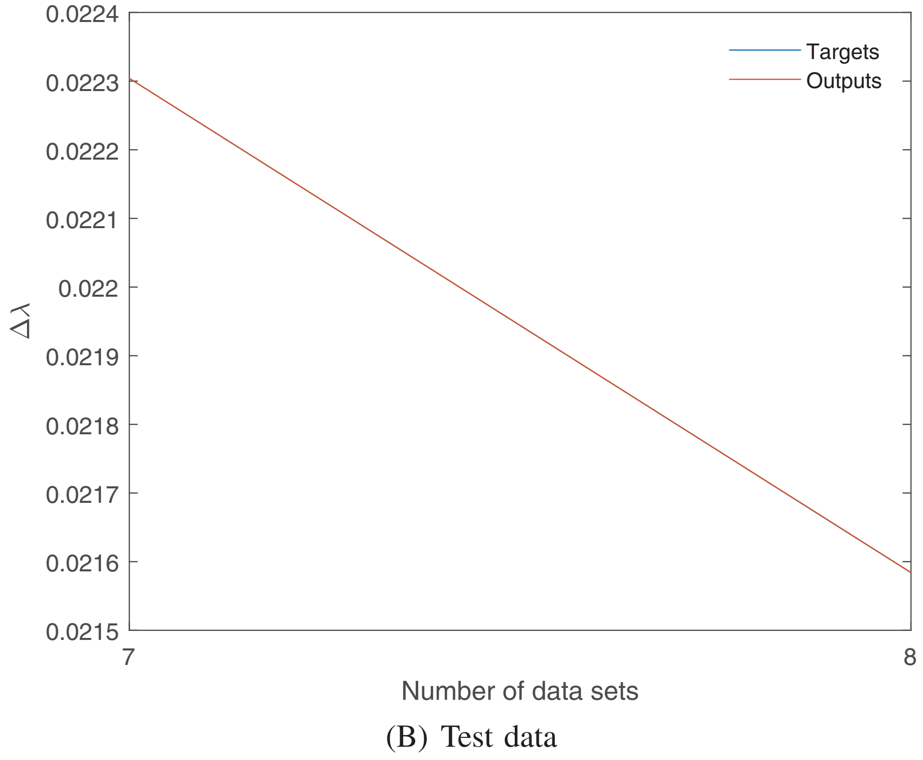


FIGURE 9 High performance of the trained network with total time for training and predicting: 0.0019 s. (A) Train data; (B) test data

TABLE 1 Predict load factor (LF) increment of the ninth step using LF increments of eight previous steps and group method of data handling

Step	1	2	3	4	5	6	7	8		Step	9
$\Delta\lambda_0 \rightarrow \Delta\lambda_7$	0.02681	0.02603	0.02527	0.02451	0.02377	0.02303	0.02230	0.02158	$\xrightarrow{\text{GMDH}}$	$\Delta\lambda_{8,\text{pre}}$	0.02087

TABLE 2 Compute load factor (LF) of the ninth step from the known LF of the eighth step using modified Riks and one-iteration nonlinear solver

			LF increment of the ninth step ($\Delta\lambda_8 = \Delta^1\lambda_8 + \Delta^2\lambda_8 + \Delta^3\lambda_8 + \Delta^4\lambda_8 + \Delta^5\lambda_8$)						
			Prediction phase	**Correction phase**					
Methods	**LF of the eighth step (λ_8)**	**Predicted convergence LF of the ninth step ($\lambda_{9,pre} = \lambda_8 + \Delta\lambda_{8,pre}$)**	$\Delta^1\lambda_8$	$\Delta^2\lambda_8$	$\Delta^3\lambda_8$	$\Delta^4\lambda_8$	$\Delta^5\lambda_8$	**LF of the 9th step (λ_9)**	**Error of LFs obtained from M-R and OINS (%)**
M-R	0.19331	—	0.02123 (e = 0.0989)	−0.00002 (e = 0.3022)	−0.00034 (e = 0.0049)	−1.2E-09 (e = 0.0048)	3.5E-15 (e = 1.6970E − 08)	0.21419 ($\lambda_9 = \lambda_8 + \Delta\lambda_8$)	0
OINS	0.19331	0.21419	—	−0.000000097 (e = 5.2327E − 06)				0.21419 ($\lambda_9 = \lambda_{9,pre} + \Delta\lambda_8$)	

Note: e is computed from Equation (18).

Abbreviations: —, no calculation; M-R, modified Riks method; OINS, one-iteration nonlinear solver.

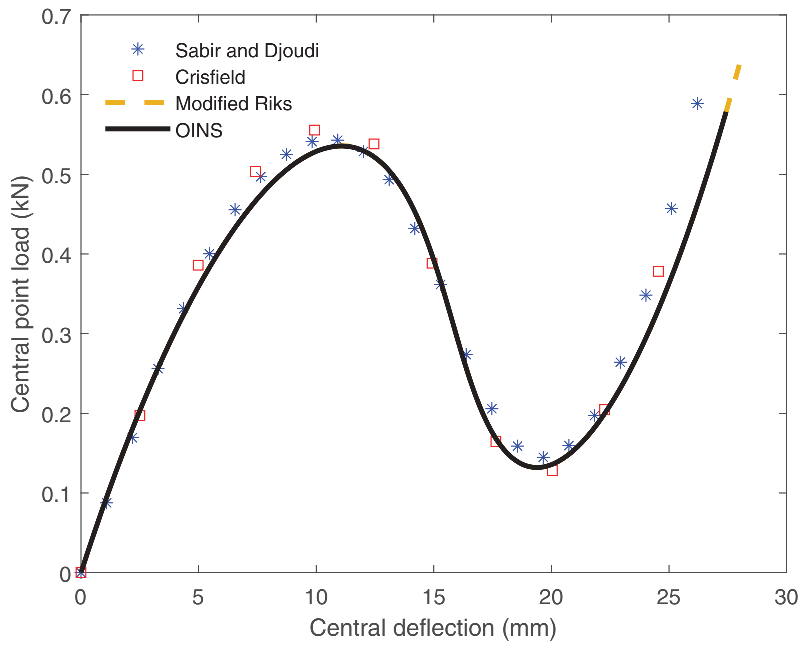


FIGURE 10 Central deflection of a hinged panel with $h = 12.7$ mm, $B = 508$ mm and $L = B$

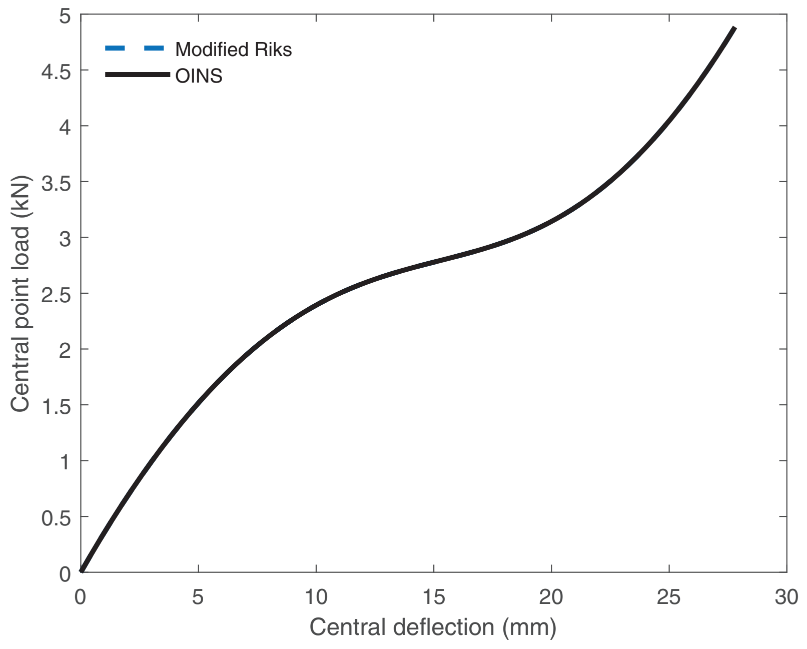


FIGURE 11 Central deflection of a hinged panel with $h = 25.4$ mm, $B = 508$ mm and $L = B$

TABLE 3 Number of average iteration of the modified Riks method and one-iteration nonlinear solver (OINS) for analyzing a hinged panel with $h = 12.7$ mm, $B = 508$ mm, $L = B$

Methods	Arc lengths	Total increments	Total iterations	Number of average iteration
Modified Riks	0.015	142	658	4.6
OINS			143	1.0
Modified Riks	0.012	177	816	4.6
OINS			177	1.0
Modified Riks	0.01	217	965	4.4
OINS			217	1.0

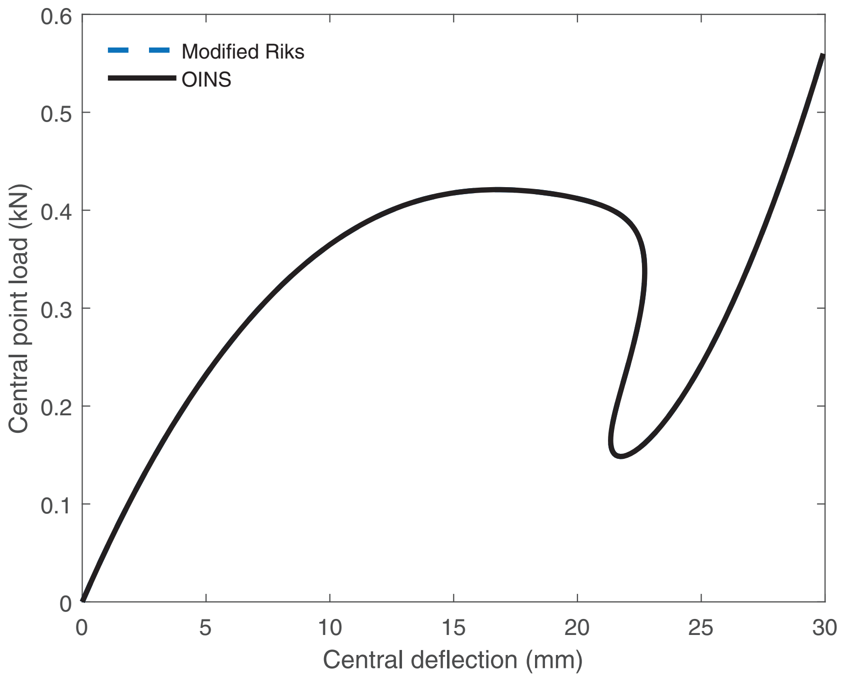


FIGURE 12 Central deflection of a hinged panel with $h = 12.7$ mm, $B = 508$ mm and $L = 1.6B$

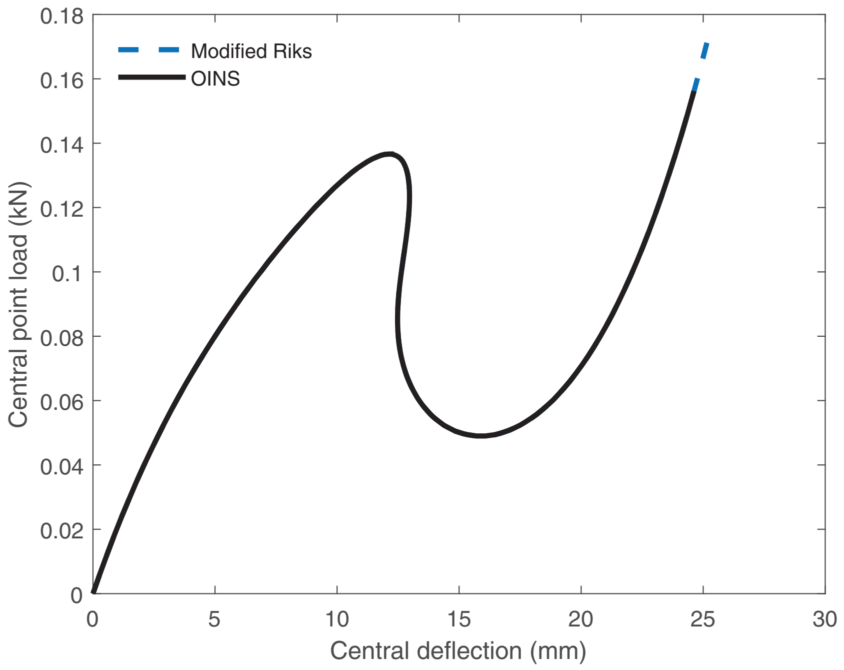


FIGURE 13 Central deflection of a clamped-hinged panel with $h = 6.35$ mm, $B = 508$ mm and $L = B$

TABLE 4 Number of average iteration of the modified Riks method and one-iteration nonlinear solver (OINS) for analyzing a hinged panel with $h = 25.4\,\text{mm}$, $B = 508\,\text{mm}$, $L = B$

Methods	Arc lengths	Total increments	Total iterations	Number of average iteration
Modified Riks	0.025	72	336	4.7
OINS			72	1.0
Modified Riks	0.03	74	355	4.8
OINS			74	1.0
Modified Riks	0.035	62	299	4.8
OINS			62	1.0

TABLE 5 Number of average iteration of the modified Riks method and one-iteration nonlinear solver (OINS) for analyzing a hinged panel with $h = 12.7\,\text{mm}$, $B = 508\,\text{mm}$, $L = 1.6B$

Methods	Arc lengths	Total increments	Total iterations	Number of average iteration
Modified Riks	0.015	160	735	4.6
OINS			183	1.1
Modified Riks	0.012	202	910	4.5
OINS			209	1.0
Modified Riks	0.01	244	1069	4.4
OINS			244	1.0

TABLE 6 Number of average iteration of the modified Riks method and one-iteration nonlinear solver (OINS) for analyzing a clamped-hinged panel with $h = 6.35\,\text{mm}$, $B = 508\,\text{mm}$, $L = B$

Methods	Arc lengths	Total increments	Total iterations	Number of average iteration
Modified Riks	0.015	157	724	4.6
OINS			188	1.2
Modified Riks	0.012	217	987	4.5
OINS			231	1.1
Modified Riks	0.01	244	1090	4.5
OINS			249	1.0

Riks method. Especially, OINS not only detects successfully limit and inflection points but also predicts exactly various types of instabilities of structures.

6 | CONCLUSIONS

In this study, the novel OINS based on time series prediction and the modified Riks method (M-R) has been proposed for nonlinear problems of solid mechanics. Different from the existing nonlinear solvers, OINS is established based on the following key concept:

- Firstly, we predict the load factor, displacement vector increments and the convergent solution of the present load step through the predictive networks which are trained by using the load factor increment and the displacement vector increment of the previous convergence steps and GMDH.
- Thanks to the predicted convergence solution of the load step is very close to or identical with the real one, the prediction phase used in any existing nonlinear solvers is eliminated completely in OINS. Next, the correction phase of M-R is adopted and the iteration is started at *the predicted convergence point* to reach the convergent solution.

It is interesting that total time for training and applying of a GMDH network is extremely low. This thanks to the very small data is used and the advantages of a self-organizing deep learning network (GMDH). The reliability and efficiency of the proposed method (OINS) is verified via solving some geometrically nonlinear problems which have different types of equilibrium paths. It should be noted that OINS can be used for any problems relating to the nonlinearity. From the obtained results, it is concluded that OINS only needs about one iteration per load step. The maximum arc-length to obtain good predictions of GMDH networks and the high efficiency of OINS is 0.015. For nonlinear problems which have the monotonous equilibrium or the equilibrium with an inflection point on it, we can use a larger arc-length for analysis (> 0.015) to reduce number of steps but still ensure the high efficiency of OINS. OINS is powerful and has a high accuracy for nonlinear problems. In addition, OINS significantly saves number of iterations (about 77%–80%) and a huge amount of computation compared with M-R. Especially, OINS not only can detect successfully limit, inflection, and other special points but also can predict exactly various types of instabilities of structures.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Tan N. Nguyen https://orcid.org/0000-0001-7734-2058
Jaehong Lee https://orcid.org/0000-0002-5056-829X
L. Minh Dang https://orcid.org/0000-0001-7691-3453

REFERENCES

1. Riks E. The application of Newton's method to the problem of elastic stability. *J Appl Mech*. 1972;39:1060.
2. Riks E. An incremental approach to the solution of snapping and buckling problems. *Int J Solids Struct*. 1979;15(7):529-551.
3. Crisfield M. A fast incremental/iterative solution procedure that handles "snap-through". *Comput Struct*. 1981;13(1):55-62.
4. Magisano D, Leonetti L, Garcea G. How to improve efficiency and robustness of the Newton method in geometrically non-linear structural problem discretized via displacement-based finite elements. *Comput Methods Appl Mech Eng*. 2017;313:986-1005.
5. Magisano D, Leonetti L, Garcea G. Advantages of the mixed format in geometrically nonlinear analysis of beams and shells using solid finite elements. *Int J Numer Methods Eng*. 2017;109(9):1237-1262.
6. Rezaiee-Pajand M, Naserian R. Geometrical nonlinear analysis based on optimization technique. *Appl Math Model*. 2018;53:32-48.
7. Rezaiee-Pajand M, Naserian R. Using residual areas for geometrically nonlinear structural analysis. *Ocean Eng*. 2015;105:327-335.
8. Rezaiee-Pajand M, Estiri H. Geometrically nonlinear analysis of shells by various dynamic relaxation methods. *World J Eng*. 2017;14(5):381-405.
9. Rezaiee-Pajand M, Estiri H. Finding equilibrium paths by minimizing external work in dynamic relaxation method. *Appl Math Model*. 2016;40(23):10300-10322.
10. Nguyen TN, Nguyen-Xuan H, Lee J. Geometrically nonlinear analysis of functionally graded material plates using an improved moving Kriging meshfree method based on a refined plate theory. *Composite Structures*. 2018;193:268-280.
11. Maghami A, Shahabian F, Hosseini SM. Path following techniques for geometrically nonlinear structures based on Multi-point methods. *Comput Struct*. 2018;208:130-142.
12. Mei Y, Hurtado DE, Pant S, Aggarwal A. On improving the numerical convergence of highly nonlinear elasticity problems. *Comput Methods Appl Mech Eng*. 2018;337:110-127.
13. Nguyen TN, Nguyen-Xuan H, Lee J. A novel data-driven nonlinear solver for solid mechanics using time series forecasting. *Finite Elem Anal Des*. 2020;171:103377.
14. Kim JH, Kim YH. A predictor–corrector method for structural nonlinear analysis. *Comput Methods Appl Mech Eng*. 2001;191(8):959-974.
15. Liang K, Ruess M, Abdalla M. The Koiter–Newton approach using von Kármán kinematics for buckling analyses of imperfection sensitive structures. *Comput Methods Appl Mech Eng*. 2014;279:440-468.
16. Magisano D, Liang K, Garcea G, Leonetti L, Ruess M. An efficient mixed variational reduced-order model formulation for nonlinear analyses of elastic shells. *Int J Numer Methods Eng*. 2018;113(4):634-655.
17. Zhao R, Yan R, Chen Z, Mao K, Wang P, Gao RX. Deep learning and its applications to machine health monitoring. *Mech Syst Signal Process*. 2019;115:213-237.
18. Nguyen NP, Hong SK. Fault-tolerant control of quadcopter UAVs using robust adaptive sliding mode approach. *Energies*. 2019;12(1):95.
19. Nguyen NP, Hong SK. Fault diagnosis and fault-tolerant control scheme for quadcopter UAVs with a total loss of actuator. *Energies*. 2019;12(6):1139.
20. Xiong W, Lu Z, Li B, Hang B, Wu Z. Automating smart recommendation from natural language API descriptions via representation learning. *Futur Gener Comput Syst*. 2018;87:382-391.
21. Mosavi A, Rabczuk T. *Learning and Intelligent Optimization for Material Design Innovation*. Springer International Publishing; 2017.
22. Mosavi A, Rabczuk T, Varkonyi-Koczy AR. *Reviewing the Novel Machine Learning Tools for Materials Design*. Springer International Publishing; 2018.

23. Minh D, Wang HX, Li YF, Nguyen TN. Explainable artificial intelligence: a comprehensive review. *Artif Intell Rev*. 2022. doi:10.1007/s10462-021-10088-y
24. Lee S, Ha J, Zokhirova M, Moon H, Lee J. Background information of deep learning for structural engineering. *Arch Comput Methods Eng*. 2018;25(1):121-129.
25. Brunetti A, Buongiorno D, Trotta GF, Bevilacqua V. Computer vision and deep learning techniques for pedestrian detection and tracking: a survey. *Neurocomputing*. 2018;300:17-33.
26. Rappel H, Beex LAA, Bordas SPA. Bayesian inference to identify parameters in viscoelasticity. *Mech Time-Depend Mater*. 2018;22(2):221-258.
27. Rappel H, Beex LAA, Noels L, Bordas SPA. Identifying elastoplastic parameters with Bayes' theorem considering output error, input error and model uncertainty. *Probab Eng Mech*. 2019;55:28-41.
28. Goodfellow I, Bengio Y, Courville A. *Deep Learning*. MIT Press; 2016.
29. Samuel AL. Some studies in machine learning using the game of checkers. *IBM J Res Dev*. 1959;3(3):210-229.
30. Längkvist M, Karlsson L, Loutfi A. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recogn Lett*. 2014;42:11-24.
31. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput*. 1997;9(8):1735-1780.
32. Sezer OB, Ozbayoglu AM. Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Appl Soft Comput*. 2018;70:525-538.
33. Minh DL, Sadeghi-Niaraki A, Huy HD, Min K, Moon H. Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network. *IEEE Access*. 2018;6:55392-55404.
34. Dang LM, Hassan SI, Im S, Mehmood I, Moon H. Utilizing text recognition for the defects extraction in sewers CCTV inspection videos. *Comput Ind*. 2018;99:96-109.
35. Dang LM, Hassan SI, Im S, Lee J, Lee S, Moon H. Deep learning based computer generated face identification using convolutional neural network. *Appl Sci*. 2018;8(12):2610.
36. Ivakhnenko AG. Polynomial theory of complex systems. *IEEE Trans Syst Man Cybern*. 1971;SMC-1(4):364-378.
37. Ivakhnenko A. The group method of data handling in long-range forecasting. *Technol Forecast Soc Chang*. 1978;12(2):213-227.
38. Nguyen TN, Lee S, Nguyen-Xuan H, Lee J. A novel analysis-prediction approach for geometrically nonlinear problems using group method of data handling. *Comput Methods Appl Mech Eng*. 2019;354:506-526.
39. Nguyen TN, Thai CH, Luu AT, Nguyen-Xuan H, Lee J. NURBS-based postbuckling analysis of functionally graded carbon nanotube-reinforced composite shells. *Comput Methods Appl Mech Eng*. 2019;347:983-1003.
40. Nguyen TN, Thai CH, Nguyen-Xuan H, Lee J. NURBS-based analyses of functionally graded carbon nanotube-reinforced composite shells. *Compos Struct*. 2018;203:349-360.
41. Reddy JN. *Mechanics of Laminated Composite Plates and Shells: Theory and Analysis*. CRC Press; 2003.
42. Nguyen TN, Lee S, Nguyen PC, Nguyen-Xuan H, Lee J. Geometrically nonlinear postbuckling behavior of imperfect FG-CNTRC shells under axial compression using isogeometric analysis. *Eur J Mech A/Solids*. 2020;84:104066.
43. Nguyen TN, Hien TD, Nguyen-Thoi T, Lee J. A unified adaptive approach for membrane structures: form finding and large deflection isogeometric analysis. *Comput Methods Appl Mech Eng*. 2020;369:113239.
44. Nguyen NV, Nguyen-Xuan H, Nguyen TN, Kang J, Lee J. A comprehensive analysis of auxetic honeycomb sandwich plates with graphene nanoplatelets reinforcement. *Compos Struct*. 2021;259:113213.
45. Nguyen NV, Nguyen HX, Lee S, Nguyen-Xuan H. Geometrically nonlinear polygonal finite element analysis of functionally graded porous plates. *Adv Eng Softw*. 2018;126:110-126.
46. Dorn M, Braga AL, Llanos CH, Coelho LS. A GMDH polynomial neural network-based method to predict approximate three-dimensional structures of polypeptides. *Expert Syst Appl*. 2012;39(15):12268-12279.
47. Pham DT, Liu X. Modelling and prediction using GMDH networks of adalines with nonlinear preprocessors. *Int J Syst Sci*. 1994;25(11):1743-1759.
48. Sabir AB, Djoudi MS. Shallow shell finite element for the large deflection geometrically nonlinear analysis of shells and plates. *Thin-Walled Struct*. 1995;21(3):253-267.

How to cite this article: Nguyen TN, Lee J, Dinh-Tien L, Minh Dang L. Deep learned one-iteration nonlinear solver for solid mechanics. *Int J Numer Methods Eng*. 2022;123(8):1841-1860. doi: 10.1002/nme.6918