Deep Autoencoder based Energy Method for the Bending, Vibration, and Buckling Analysis of Kirchhoff Plates

Xiaoying Zhuangc,d,e,1, Hongwei Guoc,2, Naif Alajlanf,3, Timon Rabczukab,∗

aDivision of Computational Mechanics,
Ton Duc Thang University,
Ho Chi Minh City, Vietnam
bFaculty of Civil Engineering,
Ton Duc Thang University,
Ho Chi Minh City, Vietnam
cInstitute of Continuum Mechanics,
Leibniz Universität Hannover,
Hannover, Germany
dDepartment of Geotechnical Engineering,
Tongji University,
Shanghai, China.
eKey Laboratory of Geotechnical and Underground Engineering of Ministry of Education,
Tongji University,
Shanghai, China.
fALISR Laboratory,
College of Computer and Information Sciences,
King Saud University,
P. O. Box 51178, Riyadh 11543, Saudi Arabia

Abstract

In this paper, we present a deep autoencoder based energy method (DAEM) for the bending, vibration and buckling analysis of Kirchhoff plates. The DAEM exploits the higher order continuity of the DAEM and integrates a deep autoencoder and the minimum total potential principle in one framework yielding an unsupervised feature learning method. The DAEM is a specific type of feedforward deep neural network (DNN) and can also serve as function approximator. With robust feature extraction capacity, the DAEM can more efficiently identify patterns behind the whole energy system, such as the field variables, natural frequency and critical buckling load factor studied in this paper. The objective function is to minimize the total potential energy. The DAEM performs unsupervised learning based on random generated points inside the physical domain so that the total potential energy is minimized at all points. For vibration and buckling analysis, the loss function is constructed based on Rayleigh’s principle and the fundamental frequency and the critical buckling load is extracted. A scaled hyperbolic tangent activation function for the underlying mechanical model is presented which meets the continuity requirement and alleviates the gradient vanishing/explosive problems under bending analysis. The DAEM can be easily implemented

∗Corresponding author

Email address: timon.rabczuk@tdtu.edu.vn (Timon Rabczuk)
1email: zhuang@ikm.uni-hannover.de
2email: ghway0723@gmail.com
3email: najlan@ksu.edu.sa
and we employed the Pytorch library and the LBFGS optimizer. A comprehensive study of the DAEM configuration is performed for several numerical examples with various geometries, load conditions, and boundary conditions.

Keywords: Deep learning, Autoencoder, Activation function, Energy method, Kirchhoff plate, Vibration, Buckling.

1. Introduction

Thin plate models are commonly used in engineering and mechanics [1] due to their computational efficiency. Their mechanical analysis are of major importance in engineering practice. Due to the limitations of analytical methods, a variety of numerical methods have been developed including the finite element method [2, 3], boundary element method [4, 5], meshfree method [6, 7], isogeometric analysis (IGA) formulations [8], numerical manifold method [9, 10, 11] and recently deep learning based methods [12, 13, 14].

Deep learning was proposed in 2006 [15, 16]. It is an unsupervised feature learning method with neural network architectures including multiple hidden layers [17]. Equipped with this hierarchical structure, it can extract information from complicated raw input data with multiple levels of abstraction through a layer-by-layer process [18]. Various variants such as multilayer perceptron (MLP), convolutional neural networks (CNN) and recurrent/recursive neural networks (RNN) [19] have been developed and applied to e.g. image processing [20, 21], object detection [22, 23], speech recognition [24, 25], biology [26, 27] and even finance [28, 29].

Artificial neural networks (ANN) can be traced back to the 1940’s [30] but they became especially popular in the past few decades due to the vast development in computer science and computational science such as backpropagation technique and advances in deep neural networks. Due to the simplicity and feasibility of ANNs to deal with nonlinear and multi-dimensional problems, they were applied in inference and identification by data scientists [31]. They were also adopted to solve partial differential equations (PDEs) [32, 33, 34] but shallow ANNs are unable to learn the complex nonlinear patterns effectively. With improved theories incorporating unsupervised pre-training, stacks of auto-encoder variants, and deep belief nets, deep learning has become also an interesting alternative to classical methods such as FEM.

According to the universal approximation theorem [35, 36], any continuous function can be approximated by a feedforward neural network with one single hidden layer. However, the number of neurons of the hidden layer tends to increase exponentially with increasing complexity and non-linearity of a model. Recent studies show that DNNs render better approximations for nonlinear functions [37]. Some researchers employed deep learning for the solution of PDEs. E et al. developed a deep learning-based
A numerical method for high-dimensional parabolic PDEs and back-forward stochastic differential equations [38, 39]. Raissi et al. [40] introduced physics-informed neural networks for supervised learning of nonlinear partial differential equations. Beck et al. [41] employed deep learning to solve nonlinear stochastic differential equations and Kolmogorov equations. Sirignano and Spiliopoulos [42] provided a theoretical proof for deep neural networks as PDE approximators, and concluded that it converged as the number of hidden layers tend to infinity. Anitescu et al. [12], Guo et al. [13], and Nguyen-Thanh et al. [14] applied deep neural networks for finding the solutions for second and forth order boundary value problems.

The learning ability of deep neural networks has been enhanced with different architectures, such as deep belief network (DBN) or deep autoencoder (DAE) [43, 44]. DAE is widely used in dimensionality reduction and feature learning. It has also been proven to be an effective way to learn and describe latent codes that reflect meaningful variations in data with an encoding and decoding layer [45]. DAE seems therefore ideally suited for learning and describing the underlying physical patterns from the governing partial differential equation or associated potential energy. In this paper, we therefore propose a deep autoencoder based energy learning method for Kirchhoff plate analysis. In this context, we exploit the higher order continuity of the DAE approximation.

The paper is organised as follows: First, we describe the Kirchhoff plate model. Then we introduce the basic elements of the deep autoencoder theory, and present a tailored activation function for this mechanical model. Subsequently, the deep autoencoder based energy method is presented. Finally, we demonstrate the efficiency and accuracy of the DAEM for various benchmark problems.
2. Kirchhoff plate model

Consider a Kirchhoff plate as shown in Figure 1. The displacement field can be expressed as:

\[
\begin{align*}
    u(x, y, z) &= -z \frac{\partial w}{\partial x}, \\
    v(x, y, z) &= -z \frac{\partial w}{\partial y}, \\
    w(x, y, z) &= w(x, y).
\end{align*}
\]

(2.1)

where the relation between lateral deflection \(w(x, y)\) of the middle surface \((z = 0)\) and rotations about the \(x,y\)-axis is given by

\[
\begin{align*}
    \theta_x &= \frac{\partial w}{\partial x}, \\
    \theta_y &= \frac{\partial w}{\partial y}.
\end{align*}
\]

(2.2)

The transversal deflection of the mid-plane is regarded as field variable and the corresponding bending and twisting curvatures are generalized strains:

\[
\begin{align*}
    k_x &= -\frac{\partial^2 w}{\partial x^2}, \\
    k_y &= -\frac{\partial^2 w}{\partial y^2}, \\
    k_{xy} &= -2 \frac{\partial^2 w}{\partial x \partial y}.
\end{align*}
\]

(2.3)

The geometric equations can be then obtained as:

\[
\begin{align*}
    k &= \begin{pmatrix} k_x \\ k_y \\ k_{xy} \end{pmatrix} = - \begin{pmatrix} \frac{\partial^2 w}{\partial x^2} \\ \frac{\partial^2 w}{\partial y^2} \\ 2 \frac{\partial^2 w}{\partial x \partial y} \end{pmatrix} = Lw,
\end{align*}
\]

(2.4)

with \(L\) being the differential operator defined as

\[
L = - \left( \begin{array}{ccc} \frac{\partial^2}{\partial x^2} & \frac{\partial^2}{\partial y^2} & 2 \frac{\partial^2}{\partial x \partial y} \end{array} \right)^T.
\]

Accordingly, the bending and twisting moments shown in Figure 1 can be expressed as:

\[
\begin{align*}
    M_x &= -D_0 \left( \frac{\partial^2 w}{\partial x^2} + \nu \frac{\partial^2 w}{\partial y^2} \right), \\
    M_y &= -D_0 \left( \frac{\partial^2 w}{\partial y^2} + \nu \frac{\partial^2 w}{\partial x^2} \right), \\
    M_{xy} &= M_{yx} = -D_0 (1 - \nu) \frac{\partial^2 w}{\partial x \partial y}.
\end{align*}
\]

(2.5)

Here, \(D_0 = \frac{Eh^3}{12(1 - \nu^2)}\) is the bending rigidity, \(E\) and \(\nu\) denote the Young’s modulus and Poisson ratio, and \(h\) is the thickness of the thin plate. It can be rewritten in a Matrix form

\[
M = Dk
\]

(2.6)

with \(D = D_0 \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1 - \nu)/2 \end{bmatrix}\). The shear forces of the mid-surface is obtained in terms of the bending and twisting moments as

\[
\begin{align*}
    Q_x &= \frac{\partial M_x}{\partial x} + \frac{\partial M_{xy}}{\partial y}, \\
    Q_y &= \frac{\partial M_{xy}}{\partial x} + \frac{\partial M_y}{\partial y}.
\end{align*}
\]

(2.7)
The boundary conditions can be categorized into three parts, namely,

$$\partial \Omega = \Gamma_1 + \Gamma_2 + \Gamma_3. \quad (2.8)$$

At the clamped boundary, $$\Gamma_1 : w = \tilde{w}, \frac{\partial w}{\partial n} = \tilde{\theta}_n, w = \tilde{w}, \tilde{\theta}_n$$ are functions of the arc length along this boundary. At the simply supported edge boundary, $$\Gamma_2 : w = \tilde{w}, M_n = \tilde{M}_n$$, where $$\tilde{M}_n$$ is also a function of arc length along this boundary. At the free boundary conditions, $$\Gamma_3 : M_n = \tilde{M}_n, \frac{\partial M_n}{\partial s} + Q_n = \tilde{q}$$, where $$\tilde{q}$$ is the load exerted along this boundary where $$\mathbf{n}, \mathbf{s}$$ corresponds to the normal and tangent directions along the boundaries.

The total potential energy consists of the strain energy $$U$$ and the potential energy of the external forces $$W$$:

$$\Pi = U + W_{ext}. \quad (2.9)$$

where

$$U = \frac{1}{2} D_0 \int_{\Omega} \left[ (k_x + k_y)^2 + 2 \left( 1 - \nu \right) \left( k_{xy}^2 - k_x k_y \right) \right] d\Omega$$

$$= \frac{1}{2} \int_{\Omega} k^T D k d\Omega. \quad (2.10)$$

and

$$W_{ext} = - \left[ \int_{\Omega} p(x, y) w d\Omega + \int_{\Gamma_3} \tilde{q} w d\Gamma - \int_{\Gamma_M} \tilde{M}_n \frac{\partial w}{\partial n} d\Gamma \right], \quad (2.11)$$

with $$\Gamma_M = \Gamma_2 + \Gamma_3$$.

For vibration analysis assuming that the plate is undergoing harmonic vibrations, we can approximate the vibrating mid-surface of the plate by

$$w(x, y, t) = W(x, y) \sin \omega t, \quad (2.12)$$

The maximum kinetic energy can be obtained by choosing $$\cos \omega t = 1$$ as

$$K_{max} = \frac{\omega^2}{2} \int_{\Omega} \rho h W^2 (x, y) d\Omega \quad (2.13)$$

The maximum strain energy $$U_{max}$$ occurs when $$\sin \omega t = 1$$. Accounting for the Rayleigh’s principle, the lowest natural frequency of a vibrating plate can be obtained by setting:

$$K_{max} = U_{max} \quad (2.14)$$

where

$$U = \frac{1}{2} D_0 \int_{\Omega} \left[ (\nabla^2 W)^2 + 2 \left( 1 - \nu \right) \left( \frac{\partial^2 W}{\partial x^2} \right)^2 - \frac{\partial^2 W}{\partial x^2} \frac{\partial^2 W}{\partial y^2} \right] d\Omega$$

The Rayleigh’s quotient can be defined as

$$\omega^2 = \frac{2U_{max}}{\int_{\Omega} \rho h W^2 (x, y) d\Omega}, \quad (2.15)$$

which will be used in the subsequent analysis.
Moreover, we will introduce the energy criterion to the classical eigenvalue buckling of Kirchhoff plates. Bifurcation of an initial configuration of equilibrium occurs \[1\] when the increment in the total potential energy of plate upon buckling equals zero:

\[
\Delta \Pi = 0
\]  
(2.16)

The increment in the total potential energy of the plate upon buckling \( \Delta \Pi \) can be expressed as strain energy of the bending and twisting of a plate \( U \) plus the work done by in-plane forces \( \Pi \),

\[
\Delta \Pi = \frac{D_0}{2} \int_{\Omega} \left[ (\nabla^2 w)^2 + 2(1 - \nu) \left( \frac{\partial^2 w}{\partial x \partial y} \right)^2 - \frac{\partial^2 w}{\partial x^2} \frac{\partial^2 w}{\partial y^2} \right] d\Omega
\]

\[+ \frac{1}{2} \int_{\Omega} \left[ \bar{N}_x \left( \frac{\partial w}{\partial x} \right)^2 + \bar{N}_y \left( \frac{\partial w}{\partial y} \right)^2 + 2 \bar{N}_{xy} \frac{\partial w}{\partial x} \frac{\partial w}{\partial y} \right] d\Omega \]  
(2.17)

Here, \( w \) can denote the perturbed transverse deflection and in-plane forces can be chosen as \( \bar{N}_x = \lambda N_x, \bar{N}_y = \lambda N_y, \bar{N}_{xy} = \lambda N_{xy} \) with \( \lambda \) a reference value of the in-plane force. In practical application, \( N_x, N_y, N_{xy} \) is often set to be unity and thus \( \lambda \) becomes the desired buckling load factor and can be obtained from Equation 2.16

\[
\lambda = \frac{-\frac{D_0}{2} \int_{\Omega} \left[ (\nabla^2 w)^2 + 2(1 - \nu) \left( \frac{\partial^2 w}{\partial x \partial y} \right)^2 - \frac{\partial^2 w}{\partial x^2} \frac{\partial^2 w}{\partial y^2} \right] d\Omega}{\frac{1}{2} \int_{\Omega} \left[ N_x \left( \frac{\partial w}{\partial x} \right)^2 + N_y \left( \frac{\partial w}{\partial y} \right)^2 + 2 N_{xy} \frac{\partial w}{\partial x} \frac{\partial w}{\partial y} \right] d\Omega}
\]  
(2.18)

which will also be used in the subsequent analysis. The minimum of the load parameter is the critical buckling load.

3. Basic theory of a deep autoencoder

In deep learning, engineers have further enhanced the learning ability of deep neural networks with different architectures, such as deep belief network (DBN) or deep autoencoder (DAE), to mention a few. Autoencoders play a fundamental role in unsupervised learning and are widely chosen deep architectures for dimensionality reduction and feature learning, which has been proven to be an effective way to learn and describe latent codes that reflect meaningful variations from raw input data.

3.1. Network architecture

Autoencoders are a specific type of feedforward neural networks including an encoder and a decoder. The basic structure is shown in Figure 2. The network reconstructs the input by mapping them from a high-dimensional space to a low-dimensional space enabling the hidden layer to learn a better representation of the input and the decoder layer then reconstructs the results to another space.
Figure 2: Basic structure of an autoencoder.

As shown in Figure 2, the fully connected feedforward neural network is composed of multiple layers: an input layer, one encoding layer, one "bottleneck" layer, one decoding layer and one output layer. Each layer consists of one or more nodes called neurons, indicated by small coloured circles in Figure 2. For the interconnected structure, every two neurons in neighbouring layers have a connection, which is represented by a connection weight. The weight between neuron $k$ in the hidden layer $l - 1$ and neuron $j$ in hidden layer $l$ is denoted by $w_{jk}^l$. No connection exists among neurons in the same layer as well as in the non-neighbouring layers. Data flows through this neural network via connections between neurons, starting from the input layer, through the encoding layer over the hidden layer to the decoding layer and finally through the output layer. The autoencoder in Figure 2 consists of two parts: an encoder mapping:

$$ f : h = f(x) $$ (3.1)

and a decoding mapping

$$ g : r = g(h) $$ (3.2)

that produces a reconstruction. The autoencoder defines a mapping

$$ AE : g ∘ f : r = g(f(x)). $$ (3.3)

Let $δ$ be the nonlinear activation function on each layer. There are many choices for the activation function and we will propose an improved version of the $tanh$-activation function to analyse the mechanical response of Kirchhoff plates. Combined with weight and bias vector defined on each layer, the nonlinear mapping can be written as

$$ f : h = δ(ω_1x + b_1) $$ (3.4)
Similarly, the decode maps can be written as

\[ g : r = \delta(\omega_2 h + b_2) \] (3.5)

Default activation functions available in Pytorch such as the \textit{tanh} do not necessarily yield the best results for every model. We will show later in numerical experiments that it results sometimes in unstable results for the deep autoencoder based energy method (DAEM). Therefore, we suggest a modified activation function:

\[ \delta(x) = \text{tanh}\left(\frac{\pi}{2} x\right) \] (3.6)

As shown later, this activation function yields better results due to the following reasons: 1.) A larger range of upper values are forced towards +1 and a larger range of lower values are close to −1, with steeper gradients for the mid-range, which can be seen in Figure 3. Thus, the training is spread uniformly through the feedforward neural network. 2.) For Kirchhoff plate problems, Navier successfully solved those problems with trigonometric Fourier series. Considering that \( \sin\left(\frac{\pi}{4} x\right) \), \( \cos\left(\frac{\pi}{4} x\right) \) is periodically changing from −1 to +1, it can be also adopted as activation function suitable for dynamic analysis. However, periodic activation functions might lead to a "rippling" cost function with bad local minima since a low and high input may produce the same output making the neural network very difficult to train. The hyperbolic functions, however, satisfy many identities analogous to the trigonometric identities and also map input between −1 and +1. In this case, the scaled hyperbolic function \( \text{tanh}\left(\frac{\pi}{4} x\right) \) is preferable, as it yields stable and accurate results and will not slow down the training process. This small change
in the activation function has largely improved the Vanishing/Exploding gradient problems in the deep autoencoder neural networks for this model, as will be shown later. The modified new activation has been studied under different layers and different neurons for cases in which Vanishing/Exploding gradient problems are observed for the tanh activation function.

Moreover, the number of neurons on each hidden layer and the number of hidden layers can be arbitrarily chosen and are invariably determined through a trial and error procedure or a pruning technique [46]. In the numerical example section, the detailed configuration of this deep autoencoder has been studied and offers an optimum selection of number of hidden layers while keeping the number of neurons on each layer small. An autoencoder thus defines a function \((g \circ f)(x; \theta)\) depending on the input data \(x\) and parametrised by \(\theta = \{\omega_1, b_1, \omega_2, b_2\}\) consisting of weights and biases in each layer. It provides an efficient way to approximate unknown field variables and identifies those physical codes behind the model.

3.2. Basic algorithm for backpropagation

Like other feedforward neural networks, deep autoencoders can be trained with all techniques in deep learning such as minibatch gradient descent method with gradients computed by backpropagation. Backpropagation is an important and computationally efficient mathematical tool to compute gradients in deep learning [47]. Backpropagation has two main phases, propagation and weight update. The chain rule is recursively applied during the whole process.

For the deep autoencoder based energy method, first, the field variable is approximated by a deep autoencoder \((g \circ f)(x; \theta)\). The components of the linear strain tensor are derivatives of the field variable and can be approximate by a set of deep autoencoder sharing the same hyperparameters. In order to find the hyperparameters of the deep autoencoder including weights and biases, a loss function \(L(f, w)\) is constructed [48]. The backpropagation algorithm for the deep autoencoder can be summarised as:

- **Input**: Input dataset \(x^1, ..., x^n\), prepare activation \(y^1\) for input layer;
- **Feedforward**: For each layer \(x^l, l = 2, 3, ..., L\), compute \(a^l = \sum_k W^l y^{l-1} + b^l\), and \(\sigma(a^l)\);
- **Output error**: Compute the error \(\delta^L = \nabla_{y^L} L \odot \sigma'_L(a^L)\)
- **Backpropagation error**: For each \(l = L - 1, L - 2, ..., 2\), compute \(\delta^l = ((W^{l+1})^T \delta^{l+1}) \odot \sigma'_l(a^l)\);
- **Output**: The gradient of the loss function is given by \(\frac{\partial L}{\partial w^l_{jk}} = y^{l-1}_k \delta^l_j\) and \(\frac{\partial L}{\partial b^l_j} = \delta^l_j\).

Here, \(\odot\) denotes the Hadamard product.
4. Deep autoencoder based energy method

Different from our previous work on deep collocation \[13\], the energy method starts from the principle of minimum energy and finds the solution by minimizing the total potential energy of this system. The energy method is a weak formulation, which has several advantages over the deep collocation method which is based on the strong form. Firstly, the continuity requirements for the approximating function is reduced in the energy based method requiring less gradient computations compared to the deep collocation method. And secondly, natural boundary condition are automatically satisfied in the energy method, which is especially helpful for fourth order problems.

4.1. Energy method for bending analysis

Let us consider Kirchhoff plate bending problems in the context of the DAEM. Recalling Equation \[2.9\] is the total potential energy, the entire problem can be boiled down to minimizing the total potential energy enforcing essential boundary conditions. The transversal deflection $w$ is approximated with the aforementioned deep autoencoder ($g \circ f (x; \theta)$) denoted by $w_h (x; \theta)$. A loss function can thus be constructed to find the approximate solution by minimizing of total potential energy with essential boundary conditions approximated by $w_h (x; \theta)$. Substituting $w_h (x_{\Omega}; \theta)$ into Equation 2.9, Equation 2.10, and Equation 2.12, results in

$$\Pi (x_{\Omega}; \theta) = U (x_{\Omega}; \theta) + W_{\text{ext}} (x_{\Omega}; \theta).$$ (4.1)

where

$$U (x_{\Omega}; \theta) = \frac{1}{2} D_0 \int_{\Omega} [(k_x (x_{\Omega}; \theta) + k_y (x_{\Omega}; \theta))^2 + 2 (1 - \nu) \left( k_{xy} (x_{\Omega}; \theta)^2 - k_x (x_{\Omega}; \theta) k_y (x_{\Omega}; \theta) \right)] d\Omega.$$ (4.2)

and

$$W_{\text{ext}} (x_{\Omega}; \theta) = - \int_{\Omega} p (x_{\Omega}) w^h (x_{\Omega}; \theta) d\Omega + \int_{\Gamma_3} \tilde{q} w^h (x_{\Gamma_3}; \theta) d\Gamma - \int_{\Gamma_M} \tilde{M}_n \frac{\partial w^h (x_{\Gamma_M}; \theta)}{\partial n} d\Gamma,$$ (4.3)

with $\Gamma_M = \Gamma_2 + \Gamma_3$, which yields an energy driven deep neural network $\Pi (x_{\Omega}; \theta)$. Moreover, $k_x (x_{\Omega}; \theta)$, $k_y (x_{\Omega}; \theta)$, and $k_{xy} (x_{\Omega}; \theta)$ can be obtained by substituting $w^h (x_{\Omega}; \theta)$ into Equation 2.3. The boundary conditions from Section 2 can also be learnt by the neural network approximation $w^h (x_{\Gamma}; \theta)$: On $\Gamma_1$, we have

$$w^h (x_{\Gamma_1}; \theta) = \tilde{w}, \quad \frac{\partial w^h (x_{\Gamma_1}; \theta)}{\partial n} = \tilde{\theta}_n.$$ (4.4)

On $\Gamma_2$,

$$w^h (x_{\Gamma_2}; \theta) = \tilde{w}, \quad \tilde{\tilde{M}}_n (x_{\Gamma_2}; \theta) = \tilde{\tilde{M}}_n,$$ (4.5)

where $\tilde{\tilde{M}}_n (x_{\Gamma_2}; \theta)$ can be obtained from Equation 2.5 by combing $w^h (x_{\Gamma_2}; \theta)$. 
On $\Gamma_3$,

$$M_n(x_{\Gamma_3}; \theta) = \tilde{M}_n, \quad \frac{\partial M_{ns}(x_{\Gamma_3}; \theta)}{\partial s} + Q_n(x_{\Gamma_3}; \theta) = \tilde{q}, \quad (4.6)$$

where $M_{ns}(x_{\Gamma_3}; \theta)$ can be obtained from Equation 2.5 and $Q_n(x_{\Gamma_3}; \theta)$ can be obtained from Equation 2.7 by combing $w^h(x_{\Gamma_3}; \theta)$. Note that $n, s$ refer to the normal and tangent directions along the boundaries.

Finally, we construct the loss function for the proposed DAEM, which minimizes the total potential energy subjected to essential boundary conditions:

$$L(\theta) = \Pi + MSE_{\Gamma_w} + MSE_{\Gamma_{\theta_n}}, \quad (4.7)$$

with

$$\Pi = \Pi(x_{\Omega}; \theta),$$

$$MSE_{\Gamma_w} = \frac{1}{N_{\Gamma_w}} \sum_{i=1}^{N_{\Gamma_w}} \left\| w^h(x_{\Gamma_w}; \theta) - \bar{w} \right\|^2,$$

$$MSE_{\Gamma_{\theta_n}} = \frac{1}{N_{\Gamma_{\theta_n}}} \sum_{i=1}^{N_{\Gamma_{\theta_n}}} \left\| \frac{\partial w^h(x_{\Gamma_{\theta_n}}; \theta)}{\partial n} - \bar{\theta}_n \right\|^2,$$

where $\Gamma_w = \Gamma_1 + \Gamma_2, \Gamma_{\theta_n} = \Gamma_1; x_{\Omega} \in R^N, \theta \in R^K$ are the neural network parameters and $L(\theta) = 0, w^h(x; \theta)$ is a solution to transversal deflection.

Note that the proposed DAEM requires a method to evaluate the integrals and also the corresponding quadrature points are deployed as input datasets. We could adopt 'traditional' multivariate numerical quadrature methods such as Gaussian quadrature. A background mesh could therefore be constructed. However, minimizing the total potential energy at those fixed points might result in underfitting. This issue can be avoided by random sampling and therefore, the Monte Carlo integration method [49] is employed in DAEM for the integral calculation. For the two dimensional Monte-Carlo integration method, let us consider the integral $\int_{\Omega} p(x, y) w^h(x; \theta) d\Omega$ in Equation 4.3, which can be evaluated by

$$\int_{\Omega} p(x_{\Omega}) w^h(x_{\Omega}; \theta) d\Omega = \frac{A}{N_{\Omega}} \sum_{i=1}^{N_{\Omega}} p(x_{i,\Omega}) w^h(x_{i,\Omega}; \theta) \quad (4.9)$$

$x_{i,\Omega}$ denoting the dataset generated by the random sampling in the physical domain and $A$ is the area of the middle surface and $N_{\Omega}$ is the number of random distributed points inside the physical domain.

4.2. Energy method for vibration and buckling analysis

The loss function for the vibration and buckling analysis has to be modified. The key objective is to obtain the fundamental natural frequency and critical buckling, respectively. Recalling Equation 2.15 the Rayleigh quotient is defined and derived from Rayleigh’s principle and the lowest natural frequency.
can be retrieved from the minimization of Equation (2.15) accounting for essential boundary conditions. Also the mode shape function is approximated by the deep autoencoder $W^h (\mathbf{x}; \theta)$. Accordingly, the loss function can be defined as:

$$L(\theta) = \frac{2U_{\text{max}} (\mathbf{x}; \theta)}{\int_{\Omega} \rho h W^2 (\mathbf{x}; \theta) d\Omega} + MSE_{\Gamma_w} + MSE_{\Gamma_{\theta_n}},$$

with

$$MSE_{\Gamma_w} = \frac{1}{N_{\Gamma_w}} \sum_{i=1}^{N_{\Gamma_w}} \left\| W^h (\mathbf{x}_{\Gamma_w}; \theta) - \tilde{W} \right\|^2,$$

$$MSE_{\Gamma_{\theta_n}} = \frac{1}{N_{\Gamma_{\theta_n}}} \sum_{i=1}^{N_{\Gamma_{\theta_n}}} \left\| \frac{\partial W^h (\mathbf{x}_{\Gamma_{\theta_n}}; \theta)}{\partial n} - \tilde{\theta}_n \right\|^2,$$

where $\Gamma_w = \Gamma_1 + \Gamma_2$, $\Gamma_{\theta_n} = \Gamma_1$, $x_{\Omega} \in \mathbb{R}^N$, $\theta \in \mathbb{R}^K$ are the neural network parameters. The Monte-Carlo quadrature rule is adopted for calculating the Rayleigh quotient. However, some modification to the loss function is needed to ensure $W^h (\mathbf{x}; \theta)$ is a nontrivial solution. Therefore, we normalize the mode shape function and ensure the inner product of the mode shape function is unity. This leads to the modified loss function

$$L(\theta) = \frac{2U_{\text{max}} (\mathbf{x}; \theta)}{\int_{\Omega} \rho h W^2 (\mathbf{x}; \theta) d\Omega} + k_p \left( \int_{\Omega} W^2 (\mathbf{x}; \theta) d\Omega - 1 \right)^2 + MSE_{\Gamma_w} + MSE_{\Gamma_{\theta_n}},$$

where $k_p$ is a penalty factor. A factor between 1 to 100 already yields good numerical results. The loss function for the buckling analysis can be written as

$$L(\theta) = \lambda (\mathbf{x}; \theta) + k_p \left( \int_{\Omega} (W^h (\mathbf{x}; \theta))^2 d\Omega - 1 \right)^2 + MSE_{\Gamma_w} + MSE_{\Gamma_{\theta_n}},$$

where $\lambda$ is the load factor. From Equation (2.18) we obtain

$$\lambda (\mathbf{x}; \theta) = \frac{- \frac{\partial \mathfrak{E}}{\partial \mathfrak{E}}}{\frac{1}{2} \int_{\Omega} \left( \nabla^2 u^h \right)^2 + 2 (1 - \nu) \left( \frac{\partial^2 u^h}{\partial x^2} \right)^2 + 2 N_{xy} \left( \frac{\partial u^h}{\partial x} \right)^2 \left( \frac{\partial u^h}{\partial y} \right)^2 + 2 N_{yx} \left( \frac{\partial u^h}{\partial y} \right)^2 \left( \frac{\partial u^h}{\partial x} \right)^2 \right) d\Omega},$$

where $u^h$ is the approximation of the transversal deflection by the deep autoencoder, i.e. $u^h (\mathbf{x}; \theta)$. Now, we can find the set of parameters $\theta$ such that the approximated deflection $u^h (\mathbf{x}; \theta)$ minimizes the loss $L(\theta)$, i.e.

$$u^h = \arg \min_{\theta \in \mathbb{R}^K} L(\theta)$$

These hyperparameters are obtained by backpropogation as mentioned before. The L-BFGS [50] optimizer with backpropogation is adopted to tune those hyperparameters of the deep autoencoder with few restrictions. Thus, the solution to thin plate bending, vibration and buckling problems by deep autoencoder based energy method can be reduced to an optimization problem. The general procedure of the
The proposed DAEM can be summarized as follows:

**Algorithm 1**: Procedure for deep encoder based energy method

---

**Input**: Create Random Sampling Points \( \mathbf{x}_{1i} \) inside the physical domain and \( \mathbf{x}_B \) on the boundaries.

**Output**: The predicted field variables

**Data**: Testing data set: \( \mathbf{x}^* \)

1. Fix the number of neurons on input layer \( D_{in} \), hidden layers \( H \), encoding layers \( iH \), \( i \in \{1, \ldots, N_e\} \), and output layer \( D_{out} \); Choosing activation function \( \delta \) and proper optimizer;
   Fixing the number of hidden layers \( N_{hl} \) and number of encoding layers \( N_{ecd} \); Training iteration \( N_{iter} \).

2. **Model Training**:

3. **for** \( i \) from 1 to \( N_{iter} \) **do**

   - (I) calculate activation function \( \mathbf{h}_i \) on hidden layers in Equation 3.4
   - (II) calculate the reconstructed output \( \mathbf{r}_i \) from \( \mathbf{h}_i \) in Equation 3.5
   - (III) choose and compute the loss function from bending loss function Equation 4.7, the vibration loss function Equation 4.12 or the buckling loss function Equation 4.13
   - (IV) back-propogate error gradient and update weights and bias

4. **Inference**: Make predictions and inference based on the trained deep autoencoder.

5. **Numerical Experiments**

   In this section, we demonstrate the performance of DAEM for several numerical examples for plate bending, vibration and buckling analysis. The simulations are done on a 64-bit macOS Mojave server with Intel(R) Core(TM) i7-8850H CPU, 32GB memory. We found that a deep neural structure with less width is preferred over a shallow structures. Hence, we mainly show results for increasing number of hidden layers rather than for increasing number of neurons. The accuracy of the numerical results by using the relative error of maximum deflection and deflection over the whole plate. The relative error is defined as:

   \[
   e = \frac{\|w_{\text{predict}} - w_{\text{analytical}}\|}{\|w_{\text{analytical}}\|} \quad (5.1)
   \]

   Here, \( \| \cdot \| \) refers to the \( l^2 \) norm.

5.1. **Bending analysis**

   We study three benchmark problems including a plate with hole and a plate on the elastic foundation, which can be compared to an analytical solution.
5.2. Square plate under a sinusoidally distributed load

Let us consider a simply-supported square plate under a sinusoidal distribution transverse loading. The sinusoidal distributed load is expressed by

\[ p = \frac{p_0}{D} \sin \left( \frac{\pi x}{a} \right) \sin \left( \frac{\pi y}{b} \right). \] (5.2)

where \( a, b \) indicate the length of the plate. The analytical solution for this problem is given by [51]:

\[ w = \frac{p_0}{\pi D \left( \frac{1}{a^2} + \frac{1}{b^2} \right)} \sin \left( \frac{\pi x}{a} \right) \sin \left( \frac{\pi y}{b} \right). \] (5.3)

We first study the accuracy and efficiency of the proposed activation function. In general, the test is performed with a deep feedforward neural network with 10 neurons per hidden layer. The relative error of the maximum deflection at the central plate and the deflection over the whole plate vs the increasing of hidden layers are shown in Figure 4. The modified hyperbolic tangent activation function \( \tanh \left( \frac{\pi}{2} x \right) \) is less dependent on the number of hidden layers than the original hyperbolic tangent activation function.

![Figure 4](image-url)

Figure 4: Relative error of (a) maximum central deflection and (b) all deflection predicted by Tanh and proposed activation function with DNN
Figure 5: The computational cost of two activation functions with increasing hidden layers.

The computational cost of those two schemes is shown in Figure 5 and is comparable. The relative error of the deflection is shown in Table 1. While the gradient explodes for some problems – indicated by NaN (not a real number) – for the original $Tanh$ activation function, the modified $tanh$ activation function always yields stable results. The encoding configuration refers here to the hidden layers and neuron numbers specified for the encoder. For the decoder a symmetric configuration is adopted.
Table 1: The Relative Error of Deflection for DAEM with different activation functions

| Encoder Configuration       | Relative Error of Deflection |
|-----------------------------|-------------------------------|
|                             | $Tanh(x)$                     | $Tanh\left(\frac{\pi}{2}x\right)$ |
| Encoding layer, [30]        | 0.0063740                     | 0.0060618                         |
| Encoding layer, [40]        | NaN                           | 0.0058869                         |
| Encoding layer, [50]        | 0.0130702                     | 0.0090355                         |
| Encoding layer, [60]        | 0.0077438                     | 0.0104713                         |
| Encoding layers, [30,10]    | 0.0083107                     | 0.0070922                         |
| Encoding layers, [40,10]    | 0.0082238                     | 0.0066074                         |
| Encoding layers, [50,10]    | 0.0061002                     | 0.0082771                         |
| Encoding layers, [60,10]    | 0.0075465                     | 0.0058044                         |
| Encoding layers, [30,20]    | 0.0075115                     | 0.0078926                         |
| Encoding layers, [40,20]    | 0.0087727                     | 0.0086784                         |
| Encoding layers, [50,20]    | NaN                           | 0.0055310                         |
| Encoding layers, [60,20]    | NaN                           | 0.0084229                         |
| Encoding layers, [30,20,10] | 0.0083553                     | 0.0069302                         |
| Encoding layers, [50,30,10] | 0.0121318                     | 0.0077830                         |
| Encoding layers, [60,30,10] | 0.0102968                     | 0.0053692                         |
| Encoding layers, [40,30,20] | 0.0075632                     | 0.0075470                         |
| Encoding layers, [50,30,20] | 0.0067328                     | 0.0065867                         |
| Encoding layers, [60,30,20] | NaN                           | 0.0060054                         |

Next, we study the recommended deep autoencoder configuration by comparing various encoders with varying layers and neurons per layer. As shown in Figure 6, an increasing number of encoding layers results – as expected – in increased computational cost.
The relative errors obtained by different encoder schemes are shown in Figure 7. The results converge to the analytical solution with increasing number of layers. However, more neurons do not necessarily improve the accuracy, especially for encoder [60]. Moreover, the results are already quite accurate with only one encoding layer. In summary, these numerical experiments suggest the change of the DAEM layer configuration rather than increasing the width of the deep neural network.

Next, we test the influence of quadrature points on the accuracy of the solution by calculating the relative error of the maximum deflection and deflection. A series of randomly distributed quadrature points ranging from [100 16900] are used to calculate the integrals. The numerical results are shown in
Associated contour plots are illustrated in Figure 9.

Figure 8: The relative error of deflection for DAEM with increasing quadrature points.

Figure 9: (a) Deflection and (b) Absolute deflection error contour predicted by DAEM.

5.2.1. Annular Plate under uniformly distributed pressure

Next, we study an annular plate, which is simply-supported on the outer circle and free on the inner circle. The analytical solution of this problem is [51]:

\[
\begin{align*}
    w &= \frac{qa^4}{64D} \left\{- \left[ 1 - \left( \frac{r}{a} \right)^4 \right] + \frac{2\alpha_1}{1 + \nu} \left[ 1 - \left( \frac{r}{a} \right)^2 \right] - \frac{4\alpha_2 \beta^2}{1 - \nu} \log \left( \frac{r}{a} \right) \right\},
\end{align*}
\]

where \(\alpha_1 = (3 + \nu)(1 - \beta^2) - 4(1 + \nu)\beta^2\kappa\), \(\alpha_2 = (3 + \nu) + 4(1 + \nu)\kappa\), \(\beta = \frac{b}{a}\), \(\kappa = \frac{\beta^2}{1 - \beta^2} \log \beta\), \(a\), \(b\) being the outer and inner radius of the annular plate, respectively. We also show results for a reference
point \((\frac{a+b}{2}, 0)\) and study different activation functions. We again observe the exploding gradient problem for DAEM with the Tanh activation function, which can be alleviated by the \(Tanh(\frac{x}{\pi})\), see Figure 10. Contour plots of the deflection and absolute deflection are depicted in Figure 11 showing that the predicted deflection agrees well with the analytical solution.

Figure 10: Relative error of (a) deflection at reference point and (b) all deflection predicted by Tanh and proposed activation function with DAEM.

Figure 11: (a) Deflection and (b) Absolute deflection contour predicted by DAEM.

5.2.2. Square plate on Winkler foundation

Finally, we study a simply-supported plate on Winkler foundation assuming the foundation’s reaction \(p(x, y)\) is expressed by \(p(x, y) = kw\), \(k\) being the foundation modulus. For a plate on a continuous
Winkler foundation, the potential energy needs to be added to the total potential energy, Equation 4.1:

\[ W_s(\mathbf{x}; \theta) = \int_\Omega k u^h(\mathbf{x}; \theta)^2 \, d\Omega \]  

(5.5)

The analytical deflection is given by [51]:

\[ w = \frac{16p}{ab} \sum_{m=1,3,5,\ldots}^{\infty} \sum_{n=1,3,5,\ldots}^{\infty} \frac{\sin \frac{m\pi x}{a} \sin \frac{n\pi y}{b}}{mn} \pi^4 D \left( \frac{m^2}{a^2} + \frac{n^2}{b^2} \right)^2 + k \]  

(5.6)

Different configurations of the deep autoencoder are tested. The relative errors in the deflection and maximum deflection are shown in Figure 12 and 13 respectively. Increasing the layers leads to more accurate results with increasing encoding layers. For some cases, increasing the width of the neural network does not improve the results, so that a deep neural network is preferable. The computational cost is depicted in Figure 14. As expected, more encoding layers increases the computational cost. However, note that this includes also the training cost. Once the network has been trained, the solution will be obtained much faster.

Figure 12: The relative error of deflection for DAEM with different encoding configurations.
5.3. Vibration analysis

We now apply DAEM to extract the fundamental frequency in a transversal vibration analysis. The results are compared with reference solutions from [52, 53, 54, 55]. Let us consider a square plate with a square cutout as shown in Figure 15. \( \xi \) is the ratio of the inner to outer square size. Various boundary conditions are studied. The non-dimensional fundamental frequency parameter \( \Omega = \frac{\omega L^2}{\sqrt{\rho h/D}} \) with different cutout ratio \( \xi \) is studied. A deep autoencoder with encoding layers \([40, 20]\) is adopted as this architecture provided accurate results for the bending analysis while being computationally efficient.
The nondimensional fundamental frequency is depicted in Table 2. The predicted results agree well with reference results of the HBM method [52], Modified Ritz method [53], FEM [53], and Discrete Ritz method [55].

![Kirchhoff thin plate in the Cartesian coordinate system.](image)

**Figure 15: Kirchhoff thin plate in the Cartesian coordinate system.**

| Cutout ratio $\xi$ | DAEM   | HBM method | Modified Ritz method | FEM    | Discrete Ritz method |
|---------------------|--------|------------|----------------------|--------|----------------------|
| 0                   | 19.7382| 19.7390    | 19.7400              | 19.7520| 19.7390              |
| 0.1                 | 19.3508| 19.4440    | 19.1830              | 19.3570| 19.4130              |
| 0.2                 | 19.0284| 19.1280    | 18.7620              | 19.1200| 19.0380              |
| 0.3                 | 19.3834| 19.4450    | 19.1830              | 19.3570| 19.3910              |
| 0.4                 | 20.8201| 20.7530    | 20.7850              | 20.7320| 20.7240              |
| 0.5                 | 23.4641| 23.4530    | 23.6640              | 23.2350| 23.4410              |
| 0.6                 | 28.2706| 28.3750    | 28.8440              | 28.2410| 28.5260              |
| 0.7                 | 38.1596| 37.5720    | 38.1580              | 35.5790| 37.8920              |
| 0.8                 | 58.0804| 57.4120    | 58.0620              | 57.4520| 57.8380              |
| 0.9                 | 120.9580| 120.0200   | 121.2300             | 120.3900| 120.9900            |

The fundamental mode shapes for different cutout ratios are shown in Table 3 and agree well with results in [52].
Table 3: Fundamental mode shapes predicted by DAEM with different cutout ratio

| Cutout ratio $\xi$ | Fundamental mode shape | Cutout ratio $\xi$ | Fundamental mode shape |
|--------------------|-------------------------|--------------------|-------------------------|
| 0                  | ![Image](image1.png)    | 0.5                | ![Image](image2.png)    |
| 0.1                | ![Image](image3.png)    | 0.6                | ![Image](image4.png)    |
| 0.2                | ![Image](image5.png)    | 0.7                | ![Image](image6.png)    |
| 0.3                | ![Image](image7.png)    | 0.8                | ![Image](image8.png)    |
| 0.4                | ![Image](image9.png)    | 0.9                | ![Image](image10.png)   |

Next, we perform a vibration analysis of the square plate for two other boundary conditions, i.e. 1: all outer edges are clamped and 2. two opposite outer edges are clamped, while the other two outer edges are simply-supported. The cutout ratio of this square plate is selected to be $\xi = 0.4$. The results are...
summarized in Table 4 and agree well with results from [52].

Table 4: Fundamental frequency parameter and mode shapes predicted by DAEM considering different boundary conditions ($\xi = 0.4$)

| Boundary Conditions | $\bar{\Omega} = \omega L^2 \sqrt{\rho h/D}$ | Fundamental mode shape |
|--------------------|------------------------------------------|------------------------|
| CCCC               | 49,3091                                  | ![mode shape]           |
| CSCS               | 35,4996                                  | ![mode shape]           |

5.4. Buckling analysis

Finally, we study a skew plate with different skew angles $\theta$ and aspect ratio $\xi = a/b$ subjected to uniaxial inplane compressive loading, see Figure 16. We consider simply-supported and clamped boundary conditions. The effects of the skew angle, aspect ratio on the critical buckling load factor are computed and compared with the reference solution presented in [56].
Figure 16: Kirchhoff thin plate in the Cartesian coordinate system.

Table 5: Comparison of critical buckling load parameter predicted by DAEM with other reference methods

| Aspect ratio $\xi$ | Skew angle $\theta$ | Non-dimensional critical buckling load parameter $K_{cr}$ |
|-------------------|---------------------|----------------------------------------------------------|
|                   | DAEM                | Rayleigh-Ritz method | FEM | CQUAD4 | CQUAD8 |
| 0.5               | $0^\circ$           | 6.2575               | 6.2500 | 6.2510 | 6.2010 | 6.2180 |
| 0.5               | $15^\circ$          | 7.0172               | 7.0000 | 6.9800 | 6.8550 | 6.9080 |
| 0.5               | $30^\circ$          | 9.9614               | 10.0200 | 9.9400 | 9.8950 | 10.0000 |
| 0.5               | $45^\circ$          | 19.4074              | 19.3000 | 9.4200 | 18.9510 | 19.2520 |
| 1                 | $0^\circ$           | 4.0007               | 4.0000 | 4.0000 | 3.9190 | 4.0000 |
| 1                 | $15^\circ$          | 4.5073               | 4.4800 | 4.4000 | 4.3060 | 4.3550 |
| 1                 | $30^\circ$          | 5.8504               | 6.4100 | 5.9300 | 5.7610 | 5.8750 |
| 1                 | $45^\circ$          | 10.5208              | 12.3000 | 10.3600 | 9.5260 | 9.9540 |
| 1.5               | $0^\circ$           | 4.3710               | 0.0000 | 0.0000 | 4.2560 | 4.2700 |
| 1.5               | $15^\circ$          | 4.6558               | 4.7700 | 4.6800 | 4.6400 | 4.6480 |
| 1.5               | $30^\circ$          | 5.9504               | 6.3700 | 5.8900 | 5.9550 | 5.8650 |
| 1.5               | $45^\circ$          | 9.1843               | 10.9000 | 8.9500 | 9.0760 | 9.1390 |
| 2                 | $0^\circ$           | 3.9354               | 0.0000 | 0.0000 | 3.8850 | 3.9030 |
| 2                 | $15^\circ$          | 4.3499               | 4.3300 | 4.3400 | 4.2710 | 4.3130 |
| 2                 | $30^\circ$          | 5.5677               | 6.0300 | 5.5900 | 5.5960 | 5.6050 |
| 2                 | $45^\circ$          | 8.9418               | 10.3000 | 8.8000 | 8.8550 | 8.8710 |

The nonlinear encoding layer adopted in this study is [60, 20], which is better suitable for varying geometries. The simply-supported skew plate is studied first and the associated results are shown in Table 5. The numerical results are compared with results of the Rayleigh-Ritz method, FEM, CQUAD4
and CQUAD8 \cite{56}. As the skew angle increases, the critical buckling load parameter $K_{cr} = \frac{\lambda_{cr}^2 h}{\pi^2 D}$ increases. For the clamped skew plate, we exemplary show results for the skew plate with aspect ratio $\chi = 1$. The numerical results can be found in Table 6. The predicted results agree well with the analytical solution.

Table 6: Critical buckling load parameter of clamped skew plate ($\chi = 1$)

| Skew angle $\theta$ | Non-dimensional critical buckling load parameter $K_{cr}$ |
|---------------------|----------------------------------------------------------|
| $0^\circ$           | DAEM 10,0909, Rayleigh-Ritz method 10,0000, FEM 10,0800, CQUAD4 9,8540, CQUAD8 10,0000 |
| $15^\circ$          | DAEM 10,7821, Rayleigh-Ritz method 10,9000, FEM 10,8400, CQUAD4 10,6900, CQUAD8 10,7750 |
| $30^\circ$          | DAEM 13,7013, Rayleigh-Ritz method 13,5800, FEM 13,6000, CQUAD4 13,5030, CQUAD8 13,5370 |
| $45^\circ$          | DAEM 20,9890, Rayleigh-Ritz method 20,4000, FEM 20,7600, CQUAD4 20,0920, CQUAD8 20,1050 |

Table 7: Buckling mode shapes of simply-supported skew plate predicted by DAEM with different aspect ratio and $\theta = 30^\circ$

| Aspect ratio $\chi$ | Buckling mode shape | Aspect ratio $\chi$ | Buckling mode shape |
|---------------------|---------------------|---------------------|---------------------|
| 0,5                 | ![Mode Shape 0.5]   | 1                   | ![Mode Shape 1]     |
| 1,5                 | ![Mode Shape 1.5]   | 2                   | ![Mode Shape 2]     |

Let us focus exemplary on the mode shape of the simply-supported plate with varying aspect ratios and a fixed skew angle of $\theta = 30$. The predicted mode shapes for each case are illustrated in Table 7. The buckling mode shapes for the clamped skew plate and a fixed aspect ratio of $\chi = 1$ and varying skew angle are listed in Table 8. The predicted results agree well with the ones in \cite{56}. Note that once the hyperparameters of this neural network are obtained, the network can be used to predict similar problems quickly and since the training is an unsupervised learning, no target value on the solution is needed.
Table 8: Buckling mode shapes of clamped skew plate predicted by DAEM with different skew angle and $\xi = 1$

| Skew angle $\theta$ | Buckling mode shape | Skew angle $\theta$ | Buckling mode shape |
|---------------------|---------------------|---------------------|---------------------|
| 0°                  | ![0° mode shape](image) | 15°                 | ![15° mode shape](image) |
| 30°                 | ![30° mode shape](image) | 45°                 | ![45° mode shape](image) |

6. Conclusions

In this paper, a deep autoencoder based energy method for bending, vibration and buckling analysis of Kirchhoff plate is proposed. For the proposed method, the deep autoencoder which is suitable for unsupervised feature extraction is combined with the minimum total potential energy principle to solve mechanical analysis of Kirchhoff plate, and it has successfully discovered the underlying physical patterns. Moreover, a tailored activation is proposed for the deep autoencoder based energy method, which has been proven to be more stable and alleviated the gradient explosion problem without compromising computational efficiency. To calculate the total potential energy, the fitted Monte Carlo integration is adopted, and makes the whole method to be truly "meshfree" and very easily accessible. And once the deep autoencoder based energy method is trained, it can predict the physical features readily.

Further, the deep autoencoder based energy method has been applied to extracting fundamental frequency, critical buckling load and corresponding mode shapes based on Rayleigh’s principle. Different numerical examples covering various type of Kirchhoff plate models, including bending, vibration, buckling of plate with different geometries, cutout, boundary and load conditions and even on Winkler foundation have been investigated to validate the proposed method. The accuracy and efficiency of the proposed activation is studied and compared in numerical examples. Further, the favourable deep autoencoder configuration are studied to offer practical guidance for application. Though this is still a
preliminary research, numerical experiments have demonstrated favourable features for this deep autoencoder method for prediction the physical patterns behind the Kirchhoff plate model. And the proposed autoencoder based energy method is more simple and efficient than the deep collocation method, so it can be further applied to more engineering fields.

However, those are just preliminary studies of this method. There are still several issues remain to be addressed, like a more suitable global optimization algorithm, and application to more complicated engineering problems, et al. Those are our future research projects, including the geometric and material nonlinear analysis in solid mechanics and computational fluid mechanics with deep autoencoder based energy method.

Reference

[1] E. Ventsel, T. Krauthammer, Thin plates and shells: theory: analysis, and applications, CRC press, 2001.

[2] K.-J. Bathe, Finite element procedures, Klaus-Jurgen Bathe, 2006.

[3] T. J. Hughes, The finite element method: linear static and dynamic finite element analysis, Courier Corporation, 2012.

[4] J. T. Katsikadelis, The boundary element method for engineers and scientists: theory and applications, Academic Press, 2016.

[5] C. A. Brebbia, S. Walker, Boundary element techniques in engineering, Elsevier, 2016.

[6] V. P. Nguyen, T. Rabczuk, S. Bordas, M. Duflot, Meshless methods: A review and computer implementation aspects, Mathematics and Computers in Simulation 79 (3) (2008) 763 – 813. doi: https://doi.org/10.1016/j.matcom.2008.01.003

[7] T. Q. Bui, M. N. Nguyen, A moving kriging interpolation-based meshfree method for free vibration analysis of kirchhoff plates, Computers & structures 89 (3-4) (2011) 380–394.

[8] V. P. Nguyen, C. Anitescu, S. P. Bordas, T. Rabczuk, Isogeometric analysis: an overview and computer implementation aspects, Mathematics and Computers in Simulation 117 (2015) 89–116.

[9] H. Zheng, Z. Liu, X. Ge, Numerical manifold space of hermitian form and application to kirchhoff’s thin plate problems, International Journal for Numerical Methods in Engineering 95 (9) (2013) 721–739.
[10] H. Guo, H. Zheng, The linear analysis of thin shell problems using the numerical manifold method, Thin-Walled Structures 124 (2018) 366–383.

[11] H. Guo, H. Zheng, X. Zhuang, Numerical manifold method for vibration analysis of kirchhoff’s plates of arbitrary geometry, Applied Mathematical Modelling 66 (2019) 695–727.

[12] C. Anitescu, E. Atroshchenko, N. Alajlan, T. Rabczuk, Artificial neural network methods for the solution of second order boundary value problems, Computers, Materials & Continua 59 (1) (2019) 345–359.

[13] H. Guo, X. Zhuang, T. Rabczuk, A deep collocation method for the bending analysis of kirchhoff plate, CMC-COMPUTERS MATERIALS & CONTINUA 59 (2) (2019) 433–456.

[14] V. M. Nguyen-Thanh, X. Zhuang, T. Rabczuk, A deep energy method for finite deformation hyper-elasticity, European Journal of Mechanics-A/Solids (2019) 103874.

[15] G. E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural computation 18 (7) (2006) 1527–1554.

[16] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in neural information processing systems, 2007, pp. 153–160.

[17] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436.

[18] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[19] J. Patterson, A. Gibson, Deep learning: A practitioner’s approach, ” O’Reilly Media, Inc.”, 2017.

[20] L. Yang, A. MacEachren, P. Mitra, T. Onorati, Visually-enabled active deep learning for (geo) text and image classification: a review, ISPRS International Journal of Geo-Information 7 (2) (2018) 65.

[21] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, et al., Identifying medical diagnoses and treatable diseases by image-based deep learning, Cell 172 (5) (2018) 1122–1131.

[22] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, et al., Deepid-net: Deformable deep convolutional neural networks for object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 2403–2412.

[23] Z.-Q. Zhao, P. Zheng, S. Xu, X. Wu, Object detection with deep learning: A review, IEEE transactions on neural networks and learning systems (2019).
[24] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al., Deep speech 2: End-to-end speech recognition in english and mandarin, in: International conference on machine learning, 2016, pp. 173–182.

[25] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, K. Shaalan, Speech recognition using deep neural networks: a systematic review, IEEE Access (2019).

[26] T. Yue, H. Wang, Deep learning for genomics: A concise overview, arXiv preprint arXiv:1802.00810 (2018).

[27] T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M. M. Hoffman, et al., Opportunities and obstacles for deep learning in biology and medicine, Journal of The Royal Society Interface 15 (141) (2018) 20170387.

[28] J. Heaton, N. Polson, J. H. Witte, Deep learning for finance: deep portfolios, Applied Stochastic Models in Business and Industry 33 (1) (2017) 3–12.

[29] T. Fischer, C. Krauss, Deep learning with long short-term memory networks for financial market predictions, European Journal of Operational Research 270 (2) (2018) 654–669.

[30] W. S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, The bulletin of mathematical biophysics 5 (4) (1943) 115–133.

[31] F. M. Dias, A. Antunes, A. M. Mota, Artificial neural networks: a review of commercial hardware, Engineering Applications of Artificial Intelligence 17 (8) (2004) 945–952.

[32] I. E. Lagaris, A. Likas, D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE transactions on neural networks 9 (5) (1998) 987–1000.

[33] I. E. Lagaris, A. C. Likas, D. G. Papageorgiou, Neural-network methods for boundary value problems with irregular boundaries, IEEE Transactions on Neural Networks 11 (5) (2000) 1041–1049.

[34] K. S. McFall, J. R. Mahan, Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions, IEEE Transactions on Neural Networks 20 (8) (2009) 1221–1233.

[35] K.-I. Funahashi, On the approximate realization of continuous mappings by neural networks, Neural Networks 2 (3) (1989) 183 – 192. doi:https://doi.org/10.1016/0893-6080(89)90003-8

URL http://www.sciencedirect.com/science/article/pii/0893608089900038

[36] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359 – 366. doi:https://doi.org/10.1016/0893-6080(89)90020-8

URL http://www.sciencedirect.com/science/article/pii/0893608089900208
[37] H. N. Mhaskar, T. Poggio, Deep vs. shallow networks: An approximation theory perspective, Analysis and Applications 14 (06) (2016) 829–848.

[38] E. Weinan, J. Han, A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Communications in Mathematics and Statistics 5 (4) (2017) 349–380.

[39] J. Han, A. Jentzen, E. Weinan, Solving high-dimensional partial differential equations using deep learning, Proceedings of the National Academy of Sciences 115 (34) (2018) 8505–8510.

[40] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707. doi:https://doi.org/10.1016/j.jcp.2018.10.045.

URL http://www.sciencedirect.com/science/article/pii/S0021999118307125

[41] C. Beck, W. E, A. Jentzen, Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations, Journal of Nonlinear Science (Jan 2019). doi:10.1007/s00332-018-9525-3.

URL http://dx.doi.org/10.1007/s00332-018-9525-3

[42] J. Sirignano, K. Spiliopoulos, Dgm: A deep learning algorithm for solving partial differential equations, Journal of Computational Physics 375 (2018) 1339–1364.

[43] H. Shao, H. Jiang, H. Zhao, F. Wang, A novel deep autoencoder feature learning method for rotating machinery fault diagnosis, Mechanical Systems and Signal Processing 95 (2017) 187–204.

[44] J. Yu, X. Zheng, S. Wang, A deep autoencoder feature learning method for process pattern recognition, Journal of Process Control 79 (2019) 1–15.

[45] J. Snoek, R. P. Adams, H. Larochelle, Nonparametric guidance of autoencoder representations using label information, Journal of Machine Learning Research 13 (Sep) (2012) 2567–2588.

[46] S. Anwar, K. Hwang, W. Sung, Structured pruning of deep convolutional neural networks, ACM Journal on Emerging Technologies in Computing Systems (JETC) 13 (3) (2017) 32.

[47] M. A. Nielsen, Neural networks and deep learning, Vol. 25, Determination press San Francisco, CA, USA:, 2015.

[48] K. Janocha, W. M. Czarnecki, On loss functions for deep neural networks in classification, Schedae Informaticae 1/2016 (2017). doi:10.4467/20838476si.16.004.6185.

URL http://dx.doi.org/10.4467/20838476SI.16.004.6185
[49] R. E. Caflisch, Monte carlo and quasi-monte carlo methods, Acta Numerica 7 (1998) 1–49. doi: 10.1017/S0962492900002804

[50] D. C. Liu, J. Nocedal, On the limited memory bfgs method for large scale optimization, Mathematical programming 45 (1-3) (1989) 503–528.

[51] S. P. Timoshenko, S. Woinowsky-Krieger, Theory of plates and shells, McGraw-hill, 1959.

[52] Y. Zhang, C. Wang, D. Pedroso, H. Zhang, Extension of hencky bar-net model for vibration analysis of rectangular plates with rectangular cutouts, Journal of Sound and Vibration 432 (2018) 65 – 87. doi:https://doi.org/10.1016/j.jsv.2018.06.029
URL http://www.sciencedirect.com/science/article/pii/S0022460X18303900

[53] K. Lam, K. Hung, S. Chow, Vibration analysis of plates with cutouts by the modified rayleigh-ritz method, Applied Acoustics 28 (1) (1989) 49 – 60. doi:https://doi.org/10.1016/0003-682X(89)90030-3
URL http://www.sciencedirect.com/science/article/pii/0003682X89900303

[54] K. Liew, S. Kitipornchai, A. Leung, C. Lim, Analysis of the free vibration of rectangular plates with central cut-outs using the discrete ritz method, International Journal of Mechanical Sciences 45 (5) (2003) 941 – 959. doi:https://doi.org/10.1016/S0020-7403(03)00109-7
URL http://www.sciencedirect.com/science/article/pii/S0020740303001097

[55] I. Shufrin, M. Eisenberger, Semi-analytical modeling of cutouts in rectangular plates with variable thickness – free vibration analysis, Applied Mathematical Modelling 40 (15) (2016) 6983 – 7000. doi:https://doi.org/10.1016/j.apm.2016.02.020
URL http://www.sciencedirect.com/science/article/pii/S0307904X16300944

[56] C. Srinivasa, Y. Suresh, W. P. Kumar, Buckling studies on laminated composite skew plates, International Journal of Computer Applications 37 (1) (2012) 35–47.