Research Article

Annals of Computational Physics and Material Science

Amplifying Sine Unit: An Oscillatory Activation Function for Deep Neural Networks to Recover Nonlinear Oscillations Efficiently

Jamshaid Ul Rahman[1,2], Faiza Makhdoom[2], Dianchen Lu[1]*

[1]*School of Mathematical Sciences, Jiangsu University 301 Xuefu road, Zhenjiang 212013, China.*

[2]*Abdus Salam School of Mathematical Sciences, GC University, Lahore 54600, Pakistan*

***Corresponding Author**
Dianchen Lu, School of Mathematical Sciences, Jiangsu University, China.

Submitted: 2024, Jan 15; **Accepted:** 2024, Feb 13; **Published:** 2024, Mar 05

Citation: Rahman, J. U., Makhdoom, F., Lu, D. (2024). Amplifying Sine Unit: An Oscillatory Activation Function for Deep Neural Networks to Recover Nonlinear Oscillations Efficiently. *Ann Comp Phy Material Sci, 1*(1), 01-12.

Abstract

Many industrial and real-life problems exhibit highly nonlinear periodic behaviors and the conventional methods may fall short of finding their analytical or closed-form solutions. Such problems demand cutting-edge computational tools with increased functionality and reduced cost. Recently, deep neural networks have gained massive research interest due to their ability to handle large data and universality to learn complex functions. In this work, we put forward a methodology based on deep neural networks with responsive layers structure to deal nonlinear oscillations in microelectromechanical systems. We incorporated some oscillatory and non-oscillatory activation functions such as growing cosine unit known as GCU, Sine, Mish and Tanh in our designed network to have a comprehensive analysis on their performance for highly nonlinear problems. Integrating oscillatory activation functions with deep neural networks definitely outperform in predicting the periodic patterns of underlying systems. To support oscillatory actuation for nonlinear systems, we have proposed a novel oscillatory activation function called Amplifying Sine Unit denoted as ASU which is more efficient than GCU for complex vibratory systems such as microelectromechanical systems. Experimental results show that the designed network with our proposed activation function ASU is more reliable and robust to handle the challenges posed by oscillations.

Keywords: Amplifying Sine Unit, Neural Networks, Microelectromechanical Systems, Nonlinear Dynamics, Non-Monotonic, Oscillatory Activation Function.

1. Introduction

Nonlinear problems are prevalent in several fields of science and technology due to the complex and periodic interactions between the components involved in scientific and industrial systems. These systems can be effectively modeled by differential equations (DEs) that contain information about their nonlinearity and periodicity. One of the major challenges in dealing with such strongly nonlinear and periodic DEs is to provide accurate analytical solutions while maintaining computational efficiency. To address this challenge, it is necessary to develop novel analytical and computational tools that can handle the complexities of these nonlinear systems. By doing so, we can gain deeper insights into the behavior of such systems and devise more efficient and effective strategies for controlling and optimizing them. Therefore, modeling and simulation of dynamical systems require critical mathematical thinking and sophisticated computational tools to simulate their solutions [1-3]. Recently, many semi-analytic iterative methods have been put forward by researchers to address the nonlinear oscillatory behaviors of underlying systems [4-6]. The deep neural networks (DNNs) have grown as the most effective architectures to deal with complex patterns and relationships present in data making them well-suited for systems with various interacting components [7]. DNNs are capable of learning complex patterns and relationships from high-dimensional data through multiple layers of nonlinear transformations. This makes them well-suited for systems with numerous interacting components that require the extraction of high-level representations from raw data. The effectiveness of DNNs stems from their ability to leverage large amounts of data to learn complex features in an end-to-end manner, without requiring explicit feature engineering. The hierarchical structure helps DNNs to learn increasingly complex patterns or interactions present in data [8]. Depending upon the nature of the problem being handled by DNNs, the choice of appropriate activation function is crucial [9]. It is commonly understood in the field of Deep Learning that linear behaviors can be easily predicted by stacking multiple layers of neural networks that solely collect linear combinations from the former layers. We can say, that the linear behaviors might easily be predicted by stacking up some layers of the network which are just collecting the linear combinations from previous layers as (1)

$$a_i = \sum_i w_i x_i + b_i \tag{1}$$

Therefore, using linear activation functions for a network is not worth enough as their role can easily be replaced by a single linear layer network shown in Fig. 1.

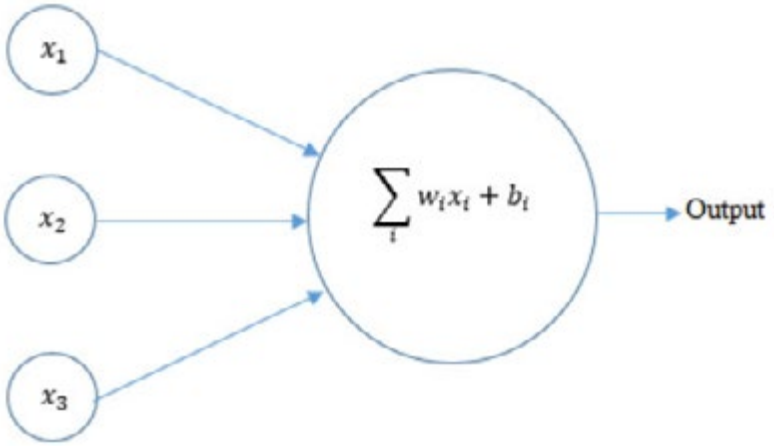


Figure 1: A single-layer neural network that can only perform linear operation. Such a network can only handle linear problems or can help draw linear decision boundaries

Nonlinear activation functions play a significant role in catching the nonlinearity present in data being assigned to DNNs for training. Monotonic nonlinear activation functions such as Sigmoid, Tanh, and especially ReLU are being abundantly used by researchers and industry [10-13]. Despite their significant contribution to many scientific applications, these activation functions may appear problematic during backpropagation when the parameters are updated using negative gradients [14]. For instance, sigmoidal activations undergo vanishing gradient problems and ReLU-based networks may suffer dying ReLU problems which may result in decreased performance and slower convergence [15,16]. These gradient-based issues during backpropagation can be avoided by the utilization of SELU and ELU which are the variants of ReLU [17,18]. In the recent past, all the research was based on the development of non-oscillatory and monotonic activation functions. Swish and Mish introduced the effective and powerful use of non-monotonic functions as activations of DNNs for various scientific and industrial problems [19-22]. GCU, an oscillatory non-monotonic activation function revolutionized the domain of activation functions by breaking the custom of utilizing only non-oscillatory activations [23]. GCU provided the single neuron solution to the XOR problem as a neuron with oscillatory activation can individually perform nonlinear decisions. Thus, oscillatory activation functions are advantageous to perform the complex assigned tasks with a lesser number of neurons and are computationally cheaper with enhanced performance [24].

Other than activation functions, the training and efficiency of DNNs can also be influenced by the right choice of loss functions and optimization algorithms depending on the demand of the problem [25,26]. For instance, recognition and classification tasks in the field of computer vision may work well with the modifications of softmax loss including Sphereface and additive parameter approaches [27-32]. On the other hand, the task of approximating the solutions to DEs using neural networks can be modeled well by applying simple squared loss functions [33,34]. We are limiting our discussion to just elaborating on the role of different oscillatory and non-oscillatory activation functions in dealing with highly nonlinear and periodic systems using DNNs.

A new oscillatory activation function ASU given by (1) has also been proposed and its behavior is graphically presented in Fig. 2.

$$ASU(a)=a.\sin(a) \qquad (1)$$

The ASU is non-monotonic and as input gets larger, its oscillations tend to amplify.

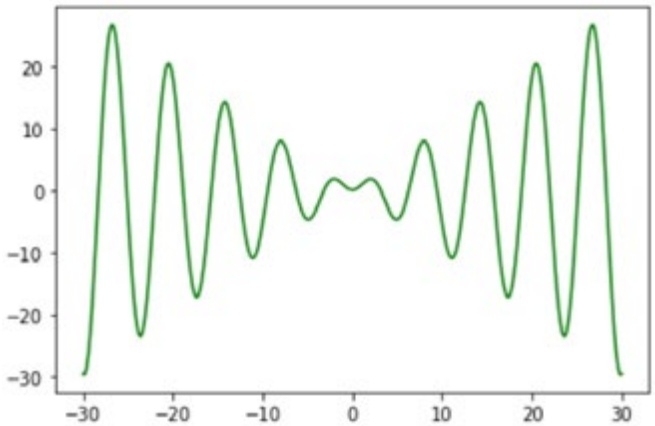


Figure 2: Graphical illustration of proposed activation function ASU that efficiently deals with nonlinear periodic problems.

We have designed DNNs with modifications using different activation functions for highly nonlinear oscillatory systems [35-37]. Outputs of DNNs are being compared with numerical solutions of LSODA routines [38] offered by a FORTRAN77 library which include Adams-Bash forth and BDF methods [39-41]. Our analysis provides a comprehensive assessment of the accuracy and reliability of DNNs in comparison to the well-established numerical methods for solving differential equations. The implementations are executed by PyTorch and numerical simulations are performed by SciPy Odeint library [42-45].

2. Methodology

This section covers the detailed methodology adopted to capture complex nonlinear oscillations. Modifications in DNNs based on oscillatory and non-oscillatory activation functions are summarized in the following sub-sections.

2.1 Formulation of DNN

The strategy is to first mathematically model the problem as a differential equation i.e.

$$Lf - p = 0 \qquad (2)$$

where L is the differential operator, f is the dependent variable wished to be explored and p represents a general expression for the remaining terms involved in the mathematical model. If f_N is the solution by DNN then it should have to satisfy (2) with the imposed initial or boundary conditions. To fulfill the criteria of satisfying the subjected conditions, f_N can be set to undergo the following transformation

$$\tilde{f} = f_0 + (1 - e^{-(t-t_0)})f_N \qquad (3)$$

where f_0 is the imposed initial condition and (3) is one of the choices made to ensure the satisfaction of subjected conditions. The next step for $\tilde{f}$ to be a valid solution is that the residual $L\tilde{f}$-p should identically be zero. The loss function given by (4) is formulated to train the network so that after a thorough optimization process the residuals are as minimized as possible.

$$Loss = (L\tilde{f} - p)^2 \qquad (4)$$

The detailed design and implementation of our DNN models are discussed in section 2.4.

2.2 Activation functions taking part in DNN-based approximations

To recover the dynamics of highly nonlinear systems that undergo periodic motions the choice of oscillatory activation functions is more robust, feasible, and time and cost-saving. It is still possible to use non-oscillatory activation functions for such problems but it would be more prone to high computational cost, training loss, and time consumption. We conducted a thorough analysis of five distinct activation functions to investigate their effectiveness in handling a highly nonlinear oscillatory system. The results of our study provide valuable insights into the suitability of various activation functions for addressing complex and nonlinear problems. This information can be used to improve the performance of Deep Neural Networks in a variety of applications. To validate our stance, we analyzed the behaviors of five different activation functions for a highly nonlinear oscillatory system.

Table 1 provides the detail of activation functions and their nature which have a significant effect on their performance and Fig. 3 provides the graphical illustrations of proposed ASU and other activation functions. It can be observed that Tanh is monotonic, whereas ASU, GCU, and Sine are oscillatory and non-monotonic. On the other hand, Mish is non-oscillatory and non-monotonic.

Name	Function	Nature
Tanh	$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Non-oscillatory and monotonic
Mish	$f(z) = z.\tanh(\log(1 - e^z))$	Non-oscillatory and non-monotonic
Sine	$f(z) = \sin(z)$	Oscillatory and non-monotonic
GCU	$f(z) = z.\cos(z)$	Oscillatory and non-monotonic
ASU	$f(z) = z.\sin(z)$	Oscillatory and non-monotonic

Table 1: A mathematical representation and nature of different activation functions

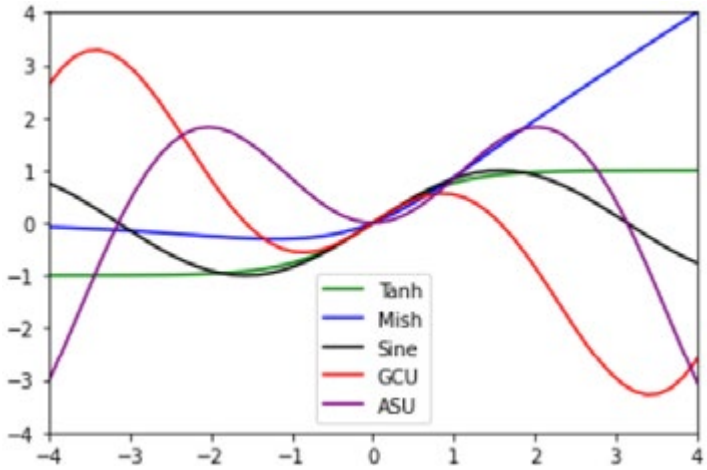


Figure 3: Plots of different oscillatory and non-oscillatory activation functions for nonlinear periodic systems

2.3 Applications in MEMS

Microelectromechanical systems (MEMS) are a cutting-edge technology that involves the fabrication of miniature, sophisticated, and reliable devices on small silicon chips, utilizing microfabrication techniques [46]. In recent years, MEMS have captured widespread attention among researchers and have become an essential component of almost every modern industry. Despite their potential applications, MEMS exhibit highly nonlinear periodic behaviors, which can pose significant challenges for classical approaches. As a result, there is an urgent need for state-of-the-art, efficient, and carefully devised techniques to handle these sensitive microstructures. This part explores the use of proposed DNNs with various activation functions to address the highly nonlinear and periodic behaviors exhibited by MEMS. The proposed ASU activation function is intended to amplify the sinusoidal input signal, thereby increasing its effectiveness in capturing the complex nonlinear dynamics of MEMS. Through our experiments, we demonstrate that ASU can achieve a better performance than GCU in modeling complex vibratory systems with higher accuracy and lower computational costs. Our findings demonstrate that DNNs are capable of providing effective solutions for MEMS and can be utilized for real-world applications.

2.3.1 Electrically actuated clamped-clamped MEMS

Suppose a clamped-clamped MEMS with a microbeam of length l, width b, thickness h and density ρ as shown in Fig. 4. The system is actuated through an electrostatic force given by (5)

$$F_e = \frac{bV^2\varepsilon_V}{2}\left[\frac{1}{(d-W)^2} - \frac{1}{(d+W)^2}\right] \qquad (5)$$

where V denotes the applied voltage, ε_V is dielectric constant and d is the initial gap between substrate and beam. The deflection of beam in electrically actuated MEMS [47] can be modeled as (6)

$$(A_0 + A_1u^2 + A_2u^4)\frac{d^2u}{dt^2} + A_3u + A_4u^3 + A_5u^5 + A_6u^7 = 0 \qquad (6)$$

subjected to initial conditions (7)

$$u(0) = \frac{\pi}{3}, u'(0) = 0 \qquad (7)$$

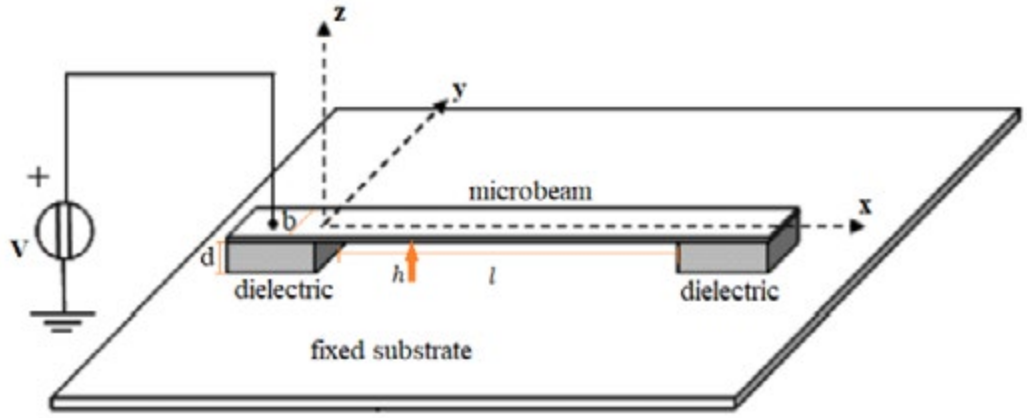


Figure 4: Clamped-clamped microbeam MEMS undergoing electrostatic force-based actuation where d is the initial gap between the beam and fixed substrate

2.4 Implementation

The designed DNNs (utilizing activation functions discussed in Table 1) consist of three hidden layers with 128 neurons each and one output unit which spits out $\tilde{f}$. In order to train the network for our modeled MEMS, the squared loss is being calculated using (4) and networks' weights are adjusted using Adam as an optimizer. Implementations are being executed by PyTorch and Fig. 5 hierarchically defines the steps involved in DNN-based approximations.

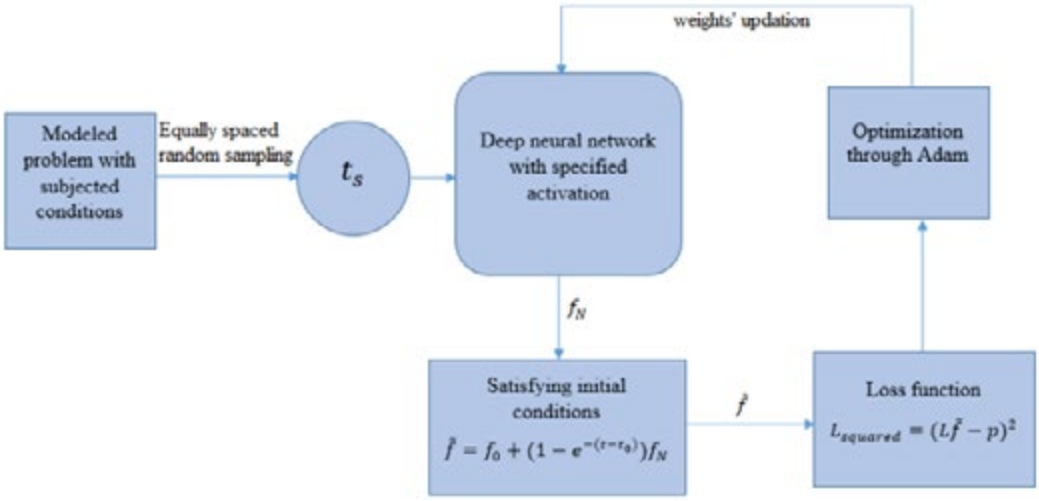


Figure 5: Flowchart representation of the methodology followed to simulate the dynamics of MEMS using DNNs

Outputs of DNNs are then being compared with the numerical simulations performed in SciPy. In order to have a comprehensive view of overall performance, an error analysis for each simulation is also being performed. For a given time domain, the error values simply represent the difference between the solution values approximated by DNNs and other numerical techniques.

3. Results and discussion

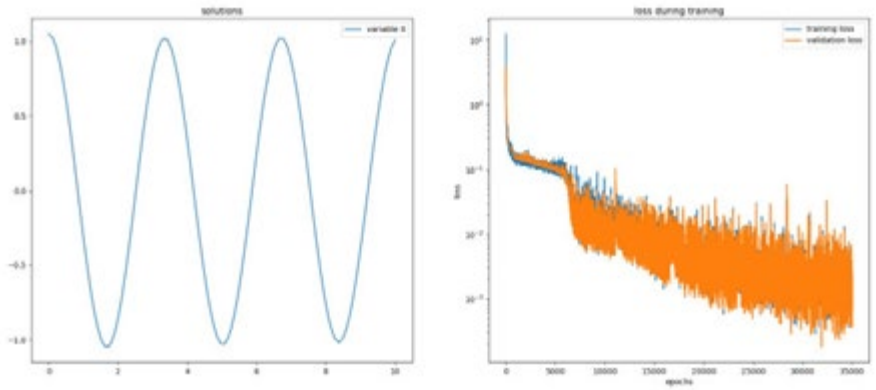
This section provides a detailed discussion on the outcomes of DNNs for electrically actuated MEMS. The performance of networks for each activation function has been discussed case by case.

3.1 DNN-based Simulations with non-oscillatory activation functions

We performed two experiments to recover the dynamics of MEMS modeled mathematically by (3.2) and (3.3). First experiment utilizes Tanh activations throughout the hidden layers and the other one is with Mish as an activation function. The training of DNNs with non-oscillatory activation functions needed large number of epochs, thus was slow and time consuming.

3.1.1 Utilization of Tanh

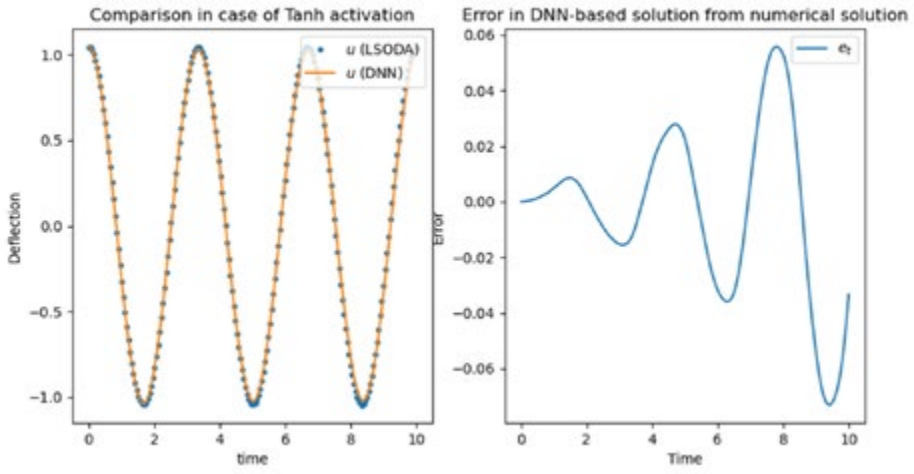
The network with Tanh activations was able to approximate the solution to the underlying MEMS after a training process of 35,000 epochs. Fig. 6(a) provides the solution curve approximated by DNN with Tanh activations. Fig. 6(b) provides the history of both the training and validation loss.



(a) DNN-based solution (b) Training and validation loss

Figure 6: The recovered solution and loss history in case of Tanh as an activation function for the network

The lowest calculated loss is 0.00017960761963920115 and it took 18 minutes and 33 seconds to accomplish the training and get the solution. Fig. 7(a) illustrates the comparison of DNN-based and numerical solutions and Fig. 7(b) demonstrates the incurred error in the specified time span.



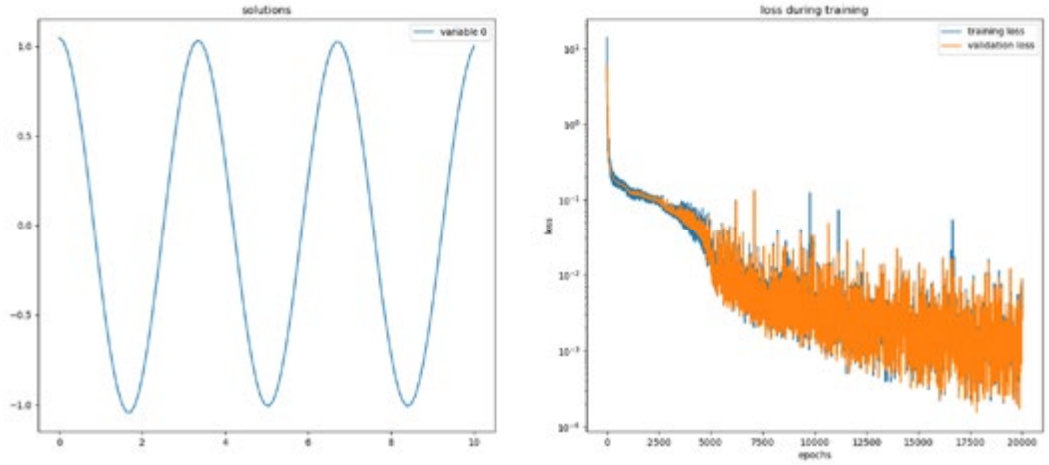
(a) Numerical and DNN approximations *(b) Error in case of Tanh activation*

Figure 7: Comparison and error illustrations of solutions approximated by both the DNNs and numerical solvers

The error simply represents difference between the solutions approximated by our designed networks and numerical simulations.

3.1.2 Utilization of Mish

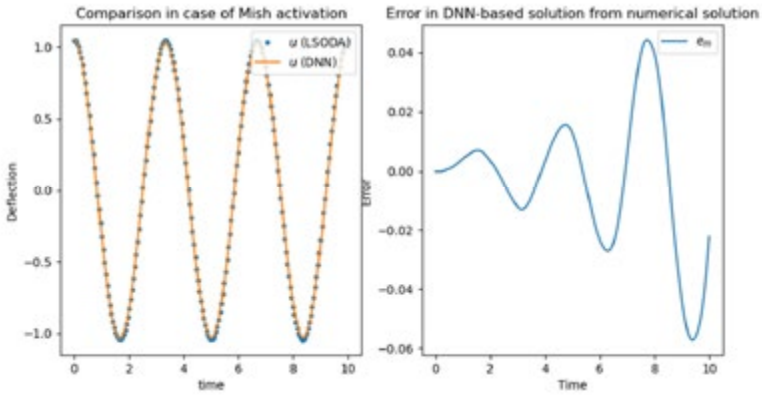
The network using Mish as an activation function recovered the solution after 20,000 training epochs and this process took 17 minutes and 50 seconds. Fig. 8(a) presents the resultant solution of MEMS and Fig. 8(b) provides the loss history. The lowest calculated loss in this case is 0.0001531996326778283.



(a) DNN-based solution (b) Loss in case of Mish

Figure 8: DNN outcomes for the approximation of deflection of electrically actuated microbeam in MEMS

Fig. 9(a) depicts the comparison of outputs and Fig. 9(b) graphically presents the error between DNN and LSODA based approximations.



(a) Comparison of outputs (b) Incurred error

Figure 9: Graphical comparison of solutions from both the techniques and discrepancy between both the approximations

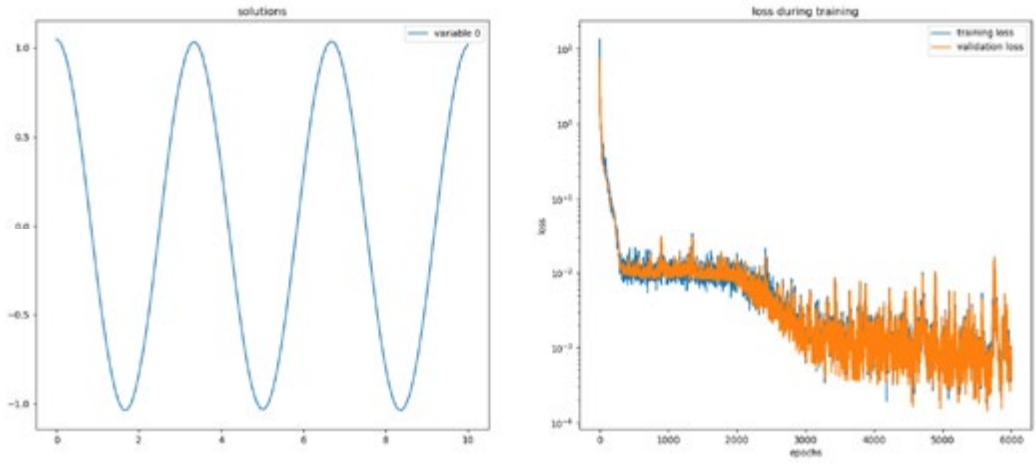
3.2 DNN-based Simulations with oscillatory activation functions

This section discusses the utilization of oscillatory activation functions to approximate the solutions of nonlinear periodic dynamical systems. In comparison to conventional deep neural networks, the utilization of oscillatory activation functions for training deep neural networks has demonstrated significantly improved training performance, including increased training speed and improved training efficiency. These findings suggest that oscillatory activation functions hold great promise for advancing the development of deep learning models.

3.2.1 Deployment of Sine

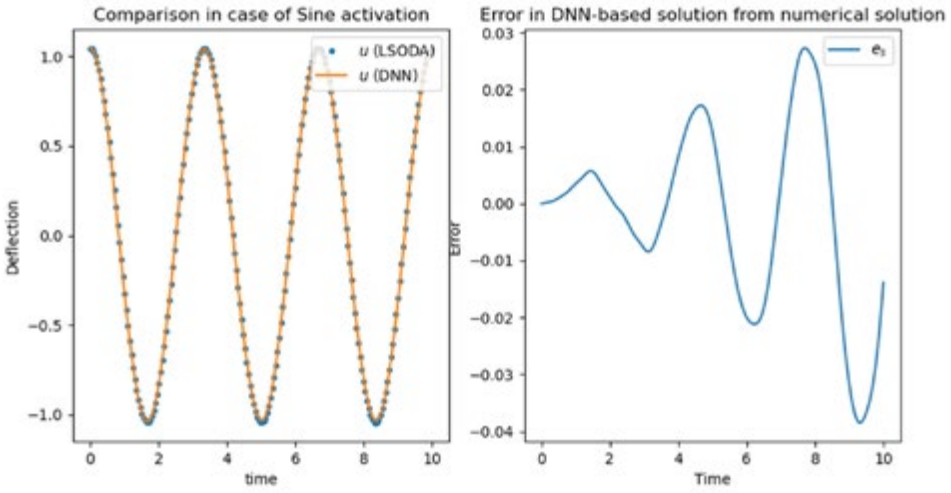
In case of Sine activations, DNN approximated the solution after a training of 6,000 epochs. The training was much faster than the case of non-oscillatory activation functions as it completed in 3 minutes and 16 seconds.

Fig. 10(a) provide the illustrations of recovered solution after training and Fig. 10(b) details the training and validation loss. The lowest calculated loss is 0.00014502210832914825. Fig. 11(a) presents the comparison of solutions from DNN and LSODA solvers while Fig. 11(b) illustrates the corresponding error between their values.



(a) Solution approximated by DNN (b) training and validation loss

Figure 10: DNN-based simulation results and loss history in case of Sine as an activation function



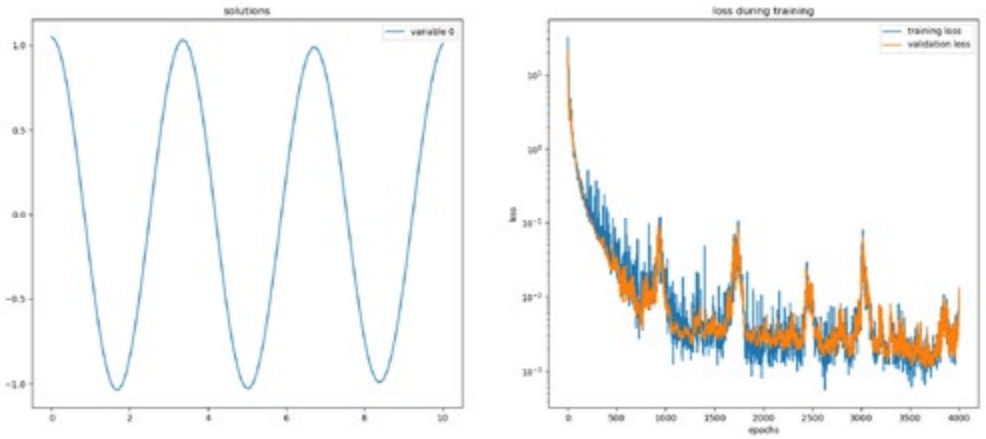
(a) Comparison of outputs (b) Graphical illustration of error

Figure 11: Comparison of approximations for the deflection of beam and error calculations for the case of Sine based DNN

One can observe that the error fairly small when the network used an oscillatory activation i.e. Sine to approximate the solution of a highly nonlinear periodic problem. Sine functions have been observed to enhance the predictive power of DNN models compared to non-oscillatory functions. Therefore, utilizing sine functions as a feature set can lead to better predictions of oscillations in MEMS using DNN models. In other words, it can be said that Sine, unlike non-oscillatory functions, has enhance the ability of DNN to predict the oscillations in MEMS.

3.2.2 Deployment of GCU

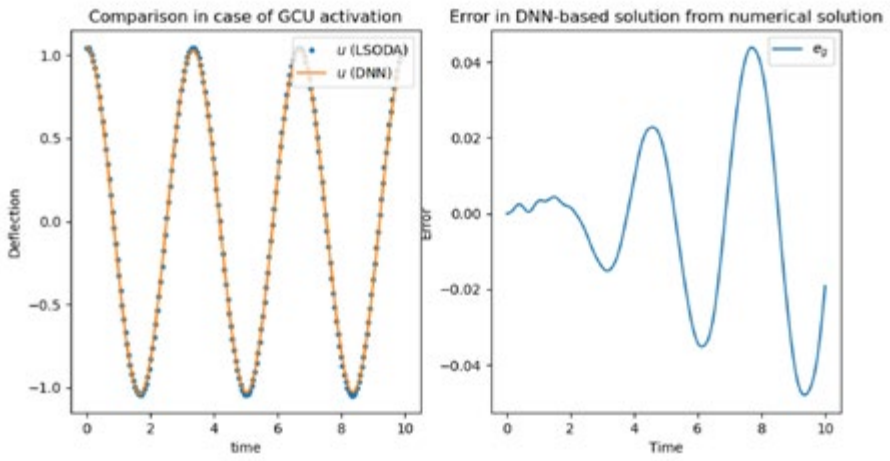
When the hidden layers of DNN were implemented by GCU as an activation function, the efficiency of network enhanced remarkably. The network was able to predict the solution after a training of just 4,000 epochs and the process took 2 minutes 53 seconds. Fig. 12(a) represents the solution predicted by the network and loss history is mentioned by Fig. 12(b).



(a) DNN-based approximation (b) Loss history in case of GCU

Figure 12: Output of network and details of training and validation loss in case of GCU activation function

Fig. 13(a) provides the detailed comparison of DNN-based and numerical solutions. The error analysis presented by Fig. 13(b) is also ensuring the capability of GCU to perceive the high oscillatory patterns.



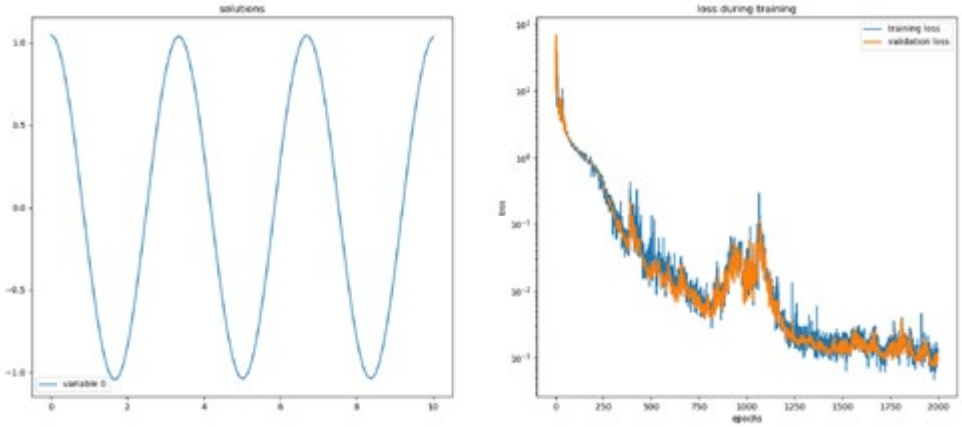
(a) Comparison of solutions (b) Error incurred in case of GCU

Figure 13: Output from DNN being compared by numerical simulations and the error which is representing the difference between the solutions approximated by both the techniques

Now we will discuss the details of last experiment which was performed by utilizing proposed activation function ASU.

3.2.3 Deployment of ASU

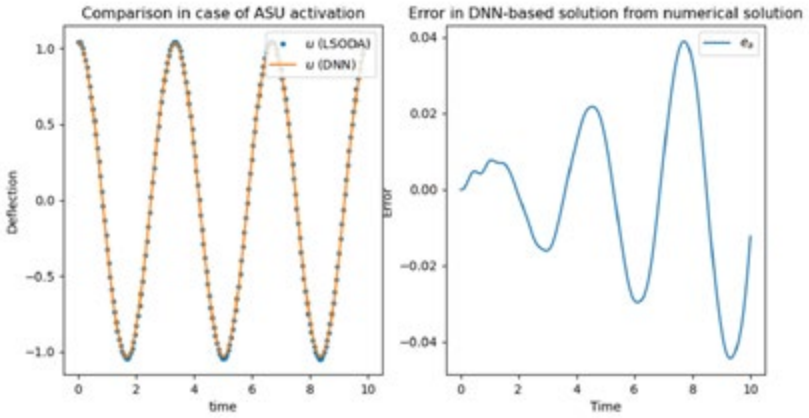
The DNN architecture with ASU activation function was super-fast and vigilant to discover the nonlinear oscillatory patterns of underlying MEMS. The network approximated the solution efficaciously after a training of just 2,000 epochs and the process completed in a short time of just 1 minute and 14 seconds. Fig. 14(a) provides us with the graphical representation of solution whereas the training and validation loss are presented by Fig. 14(b).



(a) Solution approximated by DNN (b) Loss history in case of ASU

Figure 14: Illustrations of approximated solution by DNN utilizing ASU as activation along with training and validation loss history

Fig. 15(a) provides the detailed comparison of DNN-based solution with numerical approximation by LSODA and Fig. 15(b) depicts the error evaluation of the network's output. It can be observed that besides the competence of DNN with ASU to deal nonlinear oscillations rigorously, the error also fairly lies in an acceptable range.



(a) Comparison of solutions (b) Error in case of ASU

Figure 15: Comparison of solutions from both computational techniques and the difference (error) of DNN-based solution from Numerical approximation

Therefore, to study the dynamics of nonlinear periodic systems using DNNs, the choice of oscillatory activation functions is more radical and effective with faster training cum less computational cost. One can have a glance on Table 2 to get an idea about the performance and effectivity of different activation functions being discussed.

Activation function	Training epochs	Training time
Tanh	35,000	18 minutes 33 seconds
Mish	20,000	17 minutes 50 seconds
Sine	6,000	3 minutes 16 seconds
GCU	4,000	2 minutes 53 seconds
ASU	2,000	1 minute 14 seconds

Table 2: Summary of training DNNs to deal the modeled MEMS with different activation functions.

Fig. 16 endorses the use of oscillatory activation functions for the cases of handling nonlinear periodic behaviors of systems. It also confirms the computational feasibility and effectiveness of proposed ASU to deal the complex, periodic and nonlinear dynamical systems.

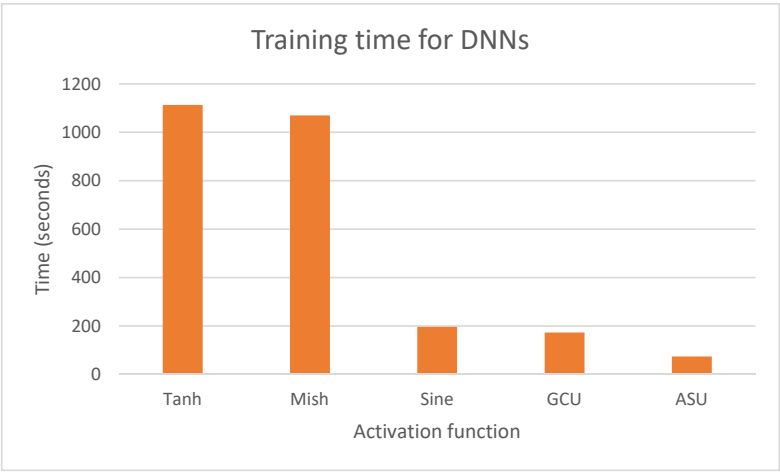


Figure 16: Time taken by each DNN architecture with different activation functions to simulate the solution of MEMS undergoing electric actuations

4. Conclusion

This work provides a comprehensive analysis on the behaviors of different oscillatory and non-oscillatory activation functions regarding highly nonlinear and periodic systems. Experimental results clearly approved the computational feasibility, effectivity and robustness of oscillatory activation functions for handling pure periodic behaviors. The performance of integrated GCU with DNNs for MEMS is better then Sine, Tanh and Mish. The proposed ASU outperformed GCU in predicting the nonlinear periodic behaviors of electrically actuated microbeam under consideration. As a future work, the role of proposed activation function can also be examined on different tasks with other deep network structures, loss functions and optimization techniques.

Acknowledgment:

This work is supported by the National Natural Science Foundation of China (Grant Nos. 12102148 and 11872189), and the Natural Science Research of Jiangsu Higher Education Institutions of China (Grant No. 21KJB110010).

References

1. Ul Rahman, J., Makhdoom, F., Ali, A., & Danish, S. (2023). Mathematical modeling and simulation of biophysics systems using neural network. *International Journal of Modern Physics B,* 2450066.
2. Arif, F., Majeed, Z., Rahman, J. U., Iqbal, N., & Kafle, J. (2022). Mathematical Modeling and Numerical Simulation for the Outbreak of COVID-19 Involving Loss of Immunity and Quarantined Class. *Computational and Mathematical Methods in Medicine, 2022*.
3. Anjum, N., Ul Rahman, J., He, J. H., Alam, M. N., & Suleman, M. (2022). An efficient analytical approach for the periodicity of nano/microelectromechanical systems' oscillators. *Mathematical Problems in Engineering, 2022*, 1-12.
4. Suleman, M., Lu, D., Yue, C., Ul Rahman, J., & Anjum, N. (2019). He–Laplace method for general nonlinear periodic solitary solution of vibration equations. *Journal of Low Frequency Noise, Vibration and Active Control, 38*(3-4), 1297-1304.
5. Lu, D., Suleman, M., Ramzan, M., & Ul Rahman, J. (2021). Numerical solutions of coupled nonlinear fractional KdV equations using He's fractional calculus. *International Journal of Modern Physics B, 35*(02), 2150023.
6. Rehman, S., Hussain, A., Rahman, J. U., Anjum, N., & Munir, T. (2022). Modified Laplace based variational iteration method for the mechanical vibrations and its applications. *acta mechanica et automatica, 16*(2), 98-102.
7. Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., & Müller, K. R. (2021). Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE, 109*(3), 247-278.
8. Panchal, G., Ganatra, A., Kosta, Y. P., & Panchal, D. (2011). Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers. *International Journal of Computer Theory and Engineering, 3*(2), 332-337.
9. Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. *Towards Data Sci, 6*(12), 310-316.
10. Nwankpa, Chigozie, et al. "Activation functions: Comparison of trends in practice and research for deep learning." *arXiv preprint arXiv:1811.03378* (2018).
11. Narayan, S. (1997). The generalized sigmoid activation function: Competitive supervised learning. *Information sciences, 99*(1-2), 69-82.
12. Lau, M. M., & Lim, K. H. (2018, December). Review of adaptive activation function in deep neural network. In *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)* (pp. 686-690). IEEE.
13. Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375.*
14. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature, 323*(6088), 533-536.
15. Tan, H. H., & Lim, K. H. (2019, June). Vanishing gradient mitigation with deep learning neural network optimization. In *2019 7th international conference on smart computing & communications (ICSCC)* (pp. 1-4). IEEE.
16. Lu, L., Shin, Y., Su, Y., & Karniadakis, G. E. (2019). Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*.
17. Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. *Advances in neural information processing systems, 30*.
18. Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.
19. Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.
20. Misra, D. (2019). Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*.
21. Tripathi, G. C., Rawat, M., & Rawat, K. (2019, October). Swish activation based deep neural network predistorter for RF-PA. In *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)* (pp. 1239-1242). IEEE.
22. Choi, K., Wi, S. M., Jung, H. G., & Suhr, J. K. (2023). Simplification of Deep Neural Network-Based Object Detector for Real-Time Edge Computing. *Sensors, 23*(7), 3777.
23. Noel, M. M., Trivedi, A., & Dutta, P. (2021). Growing cosine unit: A novel oscillatory activation function that can speedup training and reduce parameters in convolutional neural networks. *arXiv preprint arXiv:2108.12943.*
24. Noel, M. M., Bharadwaj, S., Muthiah-Nakarajan, V., Dutta, P., & Amali, G. B. (2021). Biologically inspired oscillating activation functions can bridge the performance gap between biological and artificial neurons. *arXiv preprint arXiv:2111.04020.*
25. Zaheer, R., & Shaziya, H. (2019, January). A study of the optimization algorithms in deep learning. In *2019 third international conference on inventive systems and control (ICISC)* (pp. 536-539). IEEE.
26. Jais, I. K. M., Ismail, A. R., & Nisa, S. Q. (2019). Adam optimization algorithm for wide and deep neural network. *Knowledge Engineering and Data Science, 2*(1), 41-46.
27. Voulodimos, A., Doulamis, N., Doulamis, A., &

Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience, 2018.*

28. Ul Rahman, J., Ali, A., Ur Rehman, M., & Kazmi, R. (2020). A unit softmax with Laplacian smoothing stochastic gradient descent for deep convolutional neural networks. In *Intelligent Technologies and Applications: Second International Conference, INTAP 2019, Bahawalpur, Pakistan, November 6–8, 2019, Revised Selected Papers 2* (pp. 162-174). Springer Singapore.
29. Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., & Song, L. (2017). Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 212-220).
30. Ul Rahman, J., Chen, Q., & Yang, Z. (2020). Additive parameter for deep face recognition. *Communications in Mathematics and Statistics, 8,* 203-217.
31. Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., ... & Liu, W. (2018). Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5265-5274).
32. Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4690-4699).
33. Chen, M., Xue, H., & Cai, D. (2019). Domain adaptation for semantic segmentation with maximum squares loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 2090-2099).
34. Barron, J. T. (2019). A general and adaptive robust loss function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4331-4339).
35. Chen, F., Sondak, D., Protopapas, P., Mattheakis, M., Liu, S., Agarwal, D., & Di Giovanni, M. (2020). Neurodiffeq: A python package for solving differential equations with neural networks. *Journal of Open Source Software, 5*(46), 1931.
36. Zhu, J., Liu, X., Shi, Q., He, T., Sun, Z., Guo, X., ... & Lee, C. (2019). Development trends and perspectives of future sensors and MEMS/NEMS. *Micromachines, 11*(1), 7.
37. Le, H. T., Haque, R. I., Ouyang, Z., Lee, S. W., Fried, S. I., Zhao, D., ... & Han, A. (2021). MEMS inductor fabrication and emerging applications in power electronics and neurotechnologies. *Microsystems & nanoengineering, 7*(1), 59.
38. Postawa, K., Szczygieł, J., & Kułażyński, M. (2020). A comprehensive comparison of ODE solvers for biochemical problems. *Renewable Energy, 156*, 624-633.
39. Hindmarsh, A. C., & Petzold, L. R. (2005). ODEPACK, Initial Value Problems of Ordinary Differential Equation System.
40. Tutueva, A., Karimov, T., & Butusov, D. (2020). Semi-implicit and semi-explicit Adams-Bashforth-Moulton methods. *Mathematics, 8*(5), 780.
41. Nasarudin, A. A., Ibrahim, Z. B., & Rosali, H. (2020). On the integration of stiff ODEs using block backward differentiation formulas of order six. *Symmetry, 12*(6), 952.
42. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems, 32*.
43. Mishra, P. (2022). Introduction to neural networks using PyTorch. In *PyTorch Recipes: A Problem-Solution Approach to Build, Train and Deploy Neural Network Models* (pp. 117-133). Berkeley, CA: Apress.
44. Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & Van Mulbregt, P. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods, 17*(3), 261-272.
45. Ahnert, K., & Mulansky, M. (2011, September). Odeint–solving ordinary differential equations in C++. In *AIP Conference Proceedings* (Vol. 1389, No. 1, pp. 1586-1589). American Institute of Physics.
46. Varanis, M., Silva, A., Mereles, A., & Pederiva, R. (2018). MEMS accelerometers for mechanical vibrations analysis: A comprehensive review with applications. *Journal of the Brazilian Society of Mechanical Sciences and Engineering, 40*, 1-18.
47. Anjum, N., He, J. H., He, C. H., & Ashiq, A. (2022). A Brief Review on the Asymptotic Methods for the Periodic Behaviour of Microelectromechanical Systems. *Journal of Applied and Computational Mechanics, 8*(3), 1120-1140.