Production Systems and Information Engineering
Volume 13 (1), pp. 74-83
doi: https://doi.org/10.32968/psaie.2025.1.7

ANALYSIS OF FUNCTIONALLY GRADED DISKS USING NEURAL NETWORKS AND FINITE ELEMENT SOFTWARE

DÁVID GÖNCZI
University of Miskolc, Hungary
Institute of Applied Mechanics
david.gonczi@uni-miskolc.hu

Abstract. This paper investigates the thermomechanical analysis of disks using deep neural networks and finite element software. The structural components are made from functionally graded materials and are subjected to combined mechanical and thermal loading. The neural network is trained using a dataset obtained through simulations performed by programming a commercially available finite element software Abaqus. Both the programs of the finite element software and the neural network-based solver application are written in Python programming language.

Keywords: neural networks, FEM, FGM, Abaqus scripting

1. Introduction

As technology advances, the demand for new materials with special properties is also growing. In many cases, solving a problem requires a material that is simultaneously hard, heat-resistant, or ductile. To address this issue, metals are combined with other metals or non-metal components to improve their characteristics. One method for producing such materials is to combine them in a solid state, known as composite materials. In addition, functionally graded materials (FGMs) have also emerged, where the sharp interfaces of composites between the constituent materials are eliminated. Instead of a sharp internal boundary, where failure might initiate, a graded (gradual) interface is formed, providing a smooth transition from one material to another, thus improving multiple characteristics of the material (such as the heat resistance).

Numerous studies have explored the mechanics of functionally graded materials (FGMs) from different viewpoints. Several books offer solutions to linear elastic problems in non-homogeneous structures [1, 2]. Additionally, a wide range of research papers have introduced analytical, semi-analytical, and numerical approaches for addressing thermomechanical problems in simple structural components, such as hollow spherical bodies, cylinders, plates, beams, and disks. The analytical and semi analytical methods work only for certain, simplified problems, while generalized numerical techniques, such as finite element method (FEM) can deal with a wide variety of problems. In the latter case the creation of the models is time consuming and often expensive. Papers [3-5] present analytical and semi-analytical techniques to solve simple thermoelastic problems of radially graded disks.

Neural networks (NN) can be efficient tools for the thermomechanical analysis of structural elements and components. Research related to deep NNs is extremely popular nowadays. Numerous articles address their applications as well as their potential for further development. There are several papers, such as [6-9], that focus on the development and optimization capabilities of deep neural networks and radial basis function networks. Contributors [10] compared finite element method and physics-informed neural networks to numerically solve different linear and non-linear partial differential equations. In the study, considering the solution time and accuracy, in most cases PINNs could not beat FEM, although in both cases there are methods to improve the results. Papers, such as [11-13] applied neural networks to different areas of finite element method, the constitutive equations, equation of motions etc.

This paper deals with the utilization of deep neural networks for the thermoelastic design problems of discs made from functionally graded materials. Finite element method, and consequently finite element software represent an effective and widely used approach for solving such problems. However, it also comes with several limitations. These include the time required to create and solve models, as well as the often high cost of the software. In this work, we aim to explore possibilities for reducing model preparation time (e.g., through the development of plugins) and to investigate the potential of replacing finite element simulations with neural networks.

Consider a disk made of functionally graded material. The problem is treated as axisymmetric, and thus we work in cylindrical coordinate system (r, φ, z). To solve the problem, we use the preprocessor and solver modules of the commercial finite element software Abaqus CAE (complete Abaqus environment). Geometrically, the disk is defined by its inner and outer radii (R_1, R_2), while its thickness and half-thickness are denoted by $2h_f(r) = h(r)$.

A schematic representation of the problem is shown in Fig. 1. Given the applied loads and boundary conditions illustrated in the figure, the problem can be considered axisymmetric and static. The structure is subjected to combined thermal and mechanical loads. The disk is subjected to internal and external pressures (p_1, p_2) at the inner and outer cylindrical boundary surfaces $(r = R_1, R_2)$, which depends on the positional coordinate. Optionally we can apply body forces coming from the rotation (ω) of the disk. The thermal boundary conditions can be defined either as a first-kind – e.g., prescribed temperature field, given surface temperatures $T(r = R_1) = T_1, T(r = R_2) = T_2$ – or as third-kind boundary conditions given by the environmental temperature and the corresponding heat transfer coefficient h_c. The material parameters are the elasticity parameters of Hooke's law (C_{ij}), the coefficient of linear thermal expansion (α), the thermal conductivity (λ) and the density (ϱ).

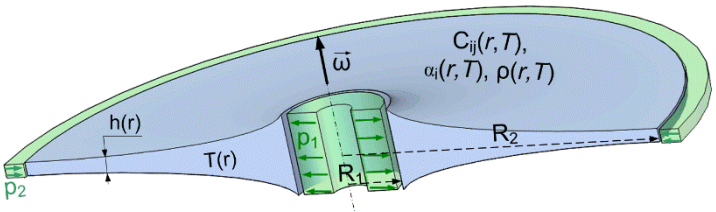

Figure 1. The sketch of the problem.

To create the neural network, Python programming language and Tensorflow are used.

2. Simulation of the problem

Abaqus CAE (Complete Abaqus Environment) is one of the leading general-purpose finite element software. It is widely used in the industry due to its ability to solve a broad range of nonlinear engineering problems. In the early versions of Abaqus, only the processor module was implemented, which was written in Fortran. Over the years, the software architecture of the program have been extended, modules have been developed and added to the core of the software. These modules can be programmed using Python. Abaqus/CAE has a modular architecture built around three main layers. The Graphical User Interface (GUI) layer provides the interactive modeling environment, menus, and viewport for geometry, meshing, and results visualization. It is implemented in C++ and wrapped with Python hooks so that nearly every GUI action corresponds to an equivalent Python command. The Application Programming Interface (API) layer is the Python-based scripting interface. It exposes Abaqus objects (parts, materials, loads, steps, jobs, etc.) and constants. The core solver layer is the underlying finite element solvers (Standard, Explicit, CFD) are compiled executables that run analysis jobs. CAE communicates with them by writing input files, launching solver processes, and then reading the resulting output database (.odb) files for post-processing. The Python interpreter embedded in Abaqus/CAE acts as the "glue" between the GUI and solver, allowing full automation, customization, and integration with external workflows.

The main types of files generated and used during simulation include the core input file (.inp) for defining the problem for the processor, it's the processor's main input. The subroutine files are required for the modification of the core equations inside the processor module. Certain features are only accessible through these files. The .cae files are the pre-processor files, essential for setting up tasks in the pre-processor and they are version dependent. The output of the simulation is in the .odb file, which is used by the postprocessor to evaluate the results. The python script files can modify and control the simulation process. The preprocessor has a built-in python interpreter, which can be utilized in several ways during simulation. Abaqus provides a powerful Python scripting interface that allows full automation and customization of finite element simulations.

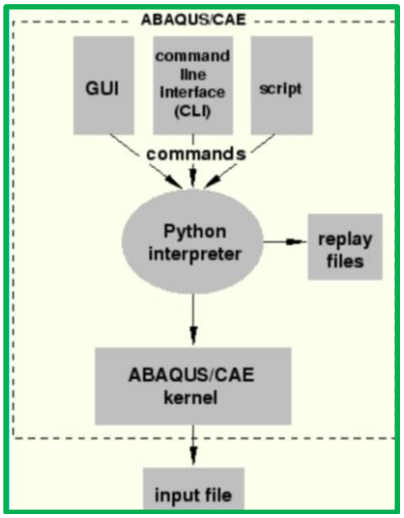

Figure 2. The preprocessor of Abaqus CAE.

The version of Python embedded in Abaqus depends on the version, Abaqus 2017: Python 2.7, Abaqus 2021: Python 3.7. The interpreter is integrated with Abaqus' internal APIs for scripting and automation. The Python interpreter inside Abaqus is sandboxed, meaning not all third-party packages are available by default. Abaqus ships with a set of default Python packages, these include abaqus and abaqusConstants, that are core Abaqus scripting commands/API, caeModules are interfaces to Abaqus/CAE functions. The odbAccess package is used to access the output database files. Numpy, sys and os packages are also included in modern Abaqus versions. The Abaqus CAE system extends the default Python packages with more than 500 classes (or objects) and numerous methods that operate between them. These are grouped into three main categories: session, mdb, and odb. Furthermore we can manually install packages into the Python interpreter of Abaqus using the embedded python executable, but it is very tricky and limited. Many users instead bridge Abaqus with external scripts, e.g., by exporting data as .csv or .npy, processing it in external Python, then feeding it back. The API architecture is object-oriented, covers geometry, meshing, loading, analysis, postprocess. Ther are a lot of important use cases of python scripting. In case of parametric modeling, Python scripting allows users to create geometric models and meshes programmatically based on input parameters. This is especially useful when analyzing families of parts with similar features. With batch job submission, scripts can automate the setup and execution of multiple simulations. This is commonly used in design studies, where many variants of a model need to be analyzed under different conditions without manual intervention or in case of training data generation for NNs. Users can employ Python to extract and process results from Abaqus output databases. This is valuable when handling large numbers of simulations or when consistent, customized result extraction is required across studies. Optimization loops enable integration of Abaqus simulations into optimization routines. For example, users can implement topology optimization or parameter tuning by running Abaqus analyses in a loop, modifying input parameters according to performance. Material modeling pipelines and GUI Plugins can be used to automate complex workflows involving material behavior evaluation. Additionally, users can extend the Abaqus/CAE interface itself by developing custom GUI plugins that incorporate specific modeling tools and shortening simulation time.

The main components of the simulation program include:

- specifying the initial configuration settings,
- generating auxiliary points required for geometry construction (e.g., edge reference points, partition centers),
- drawing the planar cross-section to be revolved, followed by structured meshing of the resulting full domain,
- creating sets of points, surfaces, and other set objects required for boundary conditions, result evaluation, and additional modeling components,
- generating material properties and assigning them to sections, then assigning sections and modeling considerations to the appropriate geometry,
- assembling the complete model,
- defining and configuring the main steps of the simulation,
- specifying loads and boundary conditions within the appropriate analysis steps,
- setting up the mesh structure, characteristics, and the applied element type (which is CAX8RT in our case), and generating the mesh,
- preparing the simulation job and allocating computational resources,
- finalizing the job configuration and launching the simulation,
- waiting for the simulation to complete, then reading the result file,
- exporting the required data to a file, documenting the results, and logging any

errors encountered.

The curved edges of the shaped disk (h_f) in the considered plane are constructed from straight line segments connecting parabolic points. This choice is also justified by the fact that, when using linear isoparametric elements, the geometry is interpolated accordingly (with line segments). The set of points defining this edge was constructed in such a way that the lengths of the inner and outer cylindrical edges of the disk can be used as parameters, denoted by $h_f(r = R_1) = h_{f1}, h_f(r = R_2) = h_{f2}$, respectively. Thus, the equation of the parabola can be expressed as follows:

$$\begin{aligned} ar^2 + br + c = 0, \\ b = \big(h_{f2} - h_{f1}\big)(R_2 - R_1) - a(R_2 + R_1), \\ c = h_{f1} - R_1\Big(\big(h_{f2} - h_{f1}\big)(R_2 - R_1) - aR_2\Big), \end{aligned} \tag{1}$$

where the governing parameter of the shape is denoted by a. The material behavior (and the approximation of material properties) is represented using homogeneous sections. This means that, instead of using the continuous material property distribution function $M(\mathbf{r})$, a discretized representation with constant values is applied. For each section, the material property is calculated, for example, at the center ($\mathbf{r}_{mid}$) of the corresponding subdomain, which means $M_{section} = M(\mathbf{r} = \mathbf{r}_{mid})$. Naturally, the finer the discretization, the more accurately it approximates the original continuous (graded) distribution. When the material composition varies only in the radial direction, then the midpoint is given by $\mathbf{r}_{mid} = 0{,}5(r_{i+1} + r_i)\mathbf{e}_r$. One of the most widely used one-dimensional distribution function can be expressed as

$$M_{FGM}(r, T) = [M_1(T) - M_2(T)]\left(\frac{r - R_1}{R_2 - R_1}\right)^m + M_2(T). \tag{2}$$

Naturally, more complex functions can also be implemented within the discussed method. The temperature dependence of material properties can be incorporated into our model, as this is supported by the material module. A custom function was developed for computing the material properties, which calculates the values to be assigned to each layer based on an expression provided as a string and the corresponding parameter values. Using the given geometric parameters, the program constructs the full geometry of the disk and partitions it into subdomains according to the user-defined resolution. It then generates the appropriate boundary conditions and loads accordingly. The problem is solved as a coupled thermoelastic simulation. Our program can be implemented following an object-oriented approach, but a procedural paradigm may also be applicable. When using object-oriented programming, it becomes evident that even in cases with a minimal number of parameters, the program must handle a significant amount of data members. This should be considered during the design phase of the program, where the use of a design pattern such as the Builder pattern is recommended. If multiple simulations are to be executed with certain fixed parameter sets, it is advisable to implement Director to facilitate the repeated use of those configurations. Figure 3 shows an example of the model of a disk displayed by the postprocessor. This two-dimensional axisymmetric problem has multiple boundary conditions (traction: purple arrows as pressure; thermal: yellow squares) and an additional kinematic boundary condition at the symmetry plane of the disk indicated by the orange triangles at the midplane along the horizontal direction.

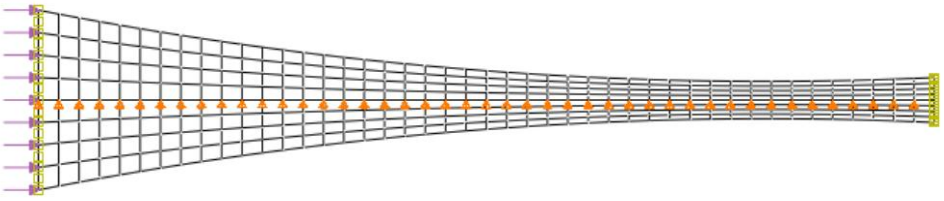

Figure 3. Example for the geometry and boundary conditions of a disk created by the program.

When the simulation is completed, the output database is used to get the variables we need. These field variables can be the deformation (displacement field), the strain or stress tensors or the equivalent stresses. The von Mises equivalent stress distribution will be used for the main design function of the disk.

Based on the program created to solve these thermoelastic disk problems, we can create plugins, which simplifies the model creation process. Abaqus provides simple tools to build a very simple GUI and bind it to the script we created. Figure 4 shows an example of this GUI.

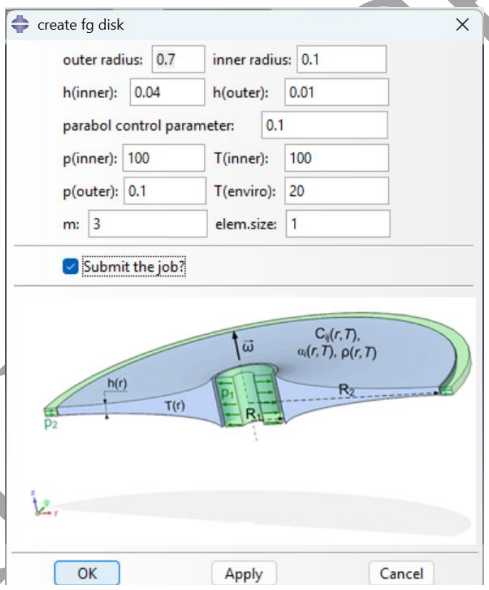

Figure 4. The plugin of the disk simulation in the preprocessor of Abaqus.

In the given example (Fig. 4), the user inputs the required data into the appropriate fields and can start the simulation directly from the dialog window without having to leave it. If modifications are needed, the user can generate the simulation files (without selecting the 'Submit the Job' option), make the necessary edits, and then run the simulation in the usual manner, which significantly shortens the solution time.

3. Creating the neural network

For the disk problem, we aim to develop a deep neural network, which requires generating a solution dataset across the space of input parameters $(R_1, R_2, h_1, h_2, a, T_1, p_1, m)$. The material properties of the constituent materials (metal: subscript 1 – ceramics: subscript 2) are

$$E_0 = 330.0GPa, E_1 = 190.0\ \text{GPa}, \alpha_0 = 5.0 \cdot 10^{-6}\ \text{K}^{-1},$$
$$\alpha_1 = 12.2 \cdot 10^{-6}\ \text{K}^{-1}, \nu_0 = 0.25, \nu_1 = 0.3, \lambda_0 = 5\text{W(mK)}^{-1},$$
$$\lambda_1 = 50\ \text{W(mK)}^{-1}.$$

The constant values include the heat transfer coefficient of $22.5\ \text{Wm}^2\text{K}$, $p_2 = 0$ MPa, and the ambient temperatures applied to the lower and upper curved surfaces $(h(r))$, which are $T_{env} = 30°\text{C}$, and $T_2 = 20°\text{C}$, respectively. The components of the functionally graded material are the same two materials defined earlier in the benchmark disk example. We will only accept results where the maximum stress does not exceed 700 MPa.

To implement this process, we can use the *os* package in the Python programming language. Abaqus generates model files for each simulation, which we utilize during the evaluation phase. However, these files are no longer needed once the data has been extracted, so we may delete them while updating the working directory. The results can be stored in a text file (e.g. in CSV format). Our program should be designed to handle issues encountered during the simulation process, it should raise an error when necessary and then proceeds to the next parameter combination to be analyzed. Error messages may be collected and saved in a separate text file. If the stress values exceed the allowable limit, the corresponding parameter sets are logged in a third text file.

In addition, it is recommended to save the progress of the simulation (in case of a crash, or termination due to software or simulation issues). The dataset we generated was based on the following parameter combinations:

$$R_1 = [0.1, 0.15, 0.25], R_2 = R_1 \cdot R_2[0] + R_2[1]:$$
$$R_2 = \left[[1.25, 0.0], [1.6, 0.0], [2.0, 0.0], [0.0, 0.7], [0.0, 1.0]\right],$$
$$a = [-0.25, -0.12, 0, 0.12, 0.25], h_1 = [0.005, 0.025, 0.06, 0.1],$$
$$h_2 = [0.005, 0.025, 0.06, 0.075], p_1 = [0.01, 30.0, 60.0, 100.0, 150.0],$$
$$T_1 = [20.0, 50.0, 100.0, 150.0, 200.0], m = [0.1, 0.5, 1, 5, 10].$$

In the examined case, the simulation was run for a limited combinations of geometric dimensions (for a specific product, the manufacturer generally uses only predefined, usually standardized sizes). To create the NN, we used the packages Panda, Sklearn and Tensorflow of Python. Multiple activation functions, initialization methods were investigated. The best one was the activation function be Selu with LeCun initialization in a fully connected network. To train the network, Nadam optimizer was used. Let us investigate the prediction of the maximum equivalent (von Mises) stress. We begin with a single-layer neural network and gradually increase the number of layers. We investigated the ideal number of neurons between 32 and 512. In the end, we decided to use 512 neurons in every deep layer, and the training process was conducted over 100 to 300 epochs. The evolution of the error, expressed as MAE (Mean Absolute Error), is illustrated in Figure 5. The error is in MPa, in case of 7 layers, its only 0.87 MPa. In case of 3 layers, the error is only 7.2 MPa, which is only a few percent error compared to the average 200 MPa, while the solution time is bit better, than the original FE solver. If we bypass the Abaqus preprocessor and solve the problem directly using the input file and the processor module, the solution time is shorter than that of the neural network. However, the majority of the total simulation time is consumed by model preparation and execution through the preprocessor.

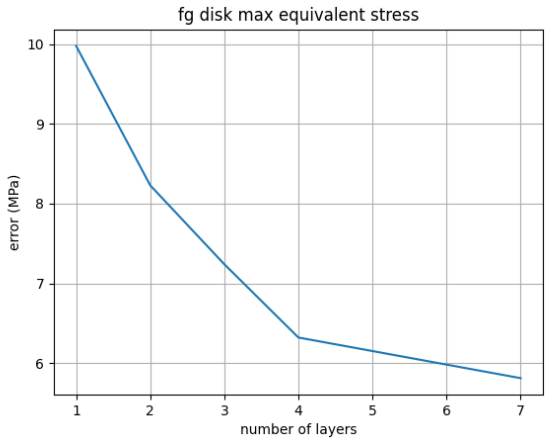

Figure 5. The graph of mean absolute error - number of layers.

In our case, the goal was to predict 44 values corresponding to the equivalent (von Mises) stresses calculated at specific nodes using finite element method. During training, a callback method was used to save the best-performing models. The more parameters we aim to calibrate in our case, the more epochs are typically required for effective training. The training process of the 7-layer neural network is illustrated in Fig. 6. In this case, the model's accuracy showed no significant improvement after 230 epochs, with only a 0.1 MPa reduction in error over the final 70 epochs, furthermore *val_* denotes validation.

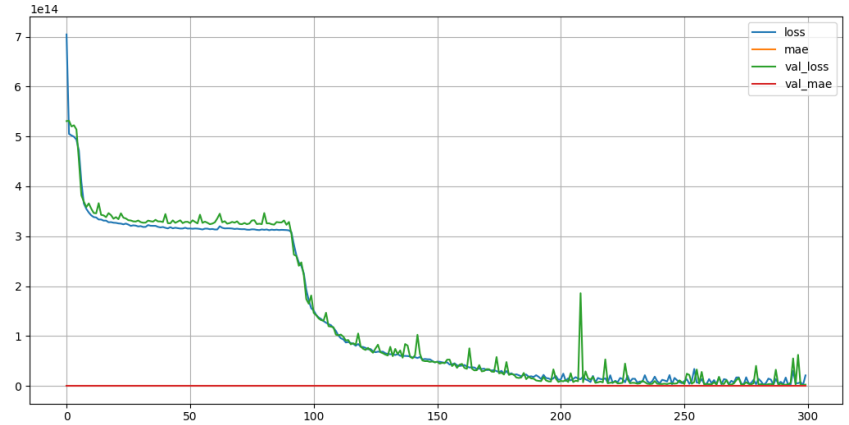

Figure 6. The training process of a 7-layer DNN.

4. Conclusion

In this paper, we investigated the simulation of functionally graded disks using finite element software and deep neural networks. We presented a simulation method, that considered parabolic disks made from functionally graded material. The finite element software provides sufficient flexibility to handle a wide range of thermoelastic problems, beyond the specific cases studied in this paper. We explored how programming capabilities in Abaqus/CAE can be leveraged to reduce solution time and developed a custom GUI plug-in tailored to the investigated problem. The main steps of the solution process and the scripting possibilities within the software were demonstrated. Using Abaqus/CAE, we generated a dataset for a user-defined metal–ceramic functionally graded disk under specified

geometrical, traction, and thermodynamic boundary conditions. This dataset was used to train a neural network, whose predictive accuracy was then evaluated. Even with a relatively small number of layers, the deep neural network (DNN) was able to predict the equivalent stress distribution with only a few percent error. Moreover, the neural network's solution time (not including the training time) was shorter than that of the original finite element simulation when we include the creation of the model and the utilization of the preprocessor, although the accuracy of finite element software is always one of its key advantages.

References

[1] Hetnarski R. B., Eslami, M. R.: Thermal Stresses – Advanced Theory and Applications. Springer, New York, USA, 2010, https://doi.org/10.1007/978-3-030-10436-8

[2] Shen H-S.: Functionally Graded Materials: Nonlinear Analysis of Plates and Shells. CRC Press, London, UK, 2009.

[3] Gönczi D. (2024). Thermoelastic analysis of functionally graded anisotropic rotating disks and radially graded spherical pressure vessels. Journal of Computational and Applied Mechanics, 19 (2), 85-104., https://doi.org/10.32973/jcam.2024.004

[4] Mert Kutsal, S., & Coşkun, S. B. (2024). Analysis of functionally graded rotating disks via analytical approximation methods. *Mechanics Based Design of Structures and Machines*, *52*(9), 6348-6367., https://doi.org/10.1080/15397734.2023.2277736

[5] Alavi, N., Nejad, M. Z., Hadi, A., & Nikeghbalyan, A. (2024). Exact thermoelastoplastic analysis of FGM rotating hollow disks in a linear elastic-fully plastic condition. *Steel and Composite Structures*, *51*(4), 377-389., https://doi.org/10.12989/scs.2024.51.4.377

[6] Kovács L. (2025). Classification Improvement with Integration of Radial Basis Function and Multilayer Perceptron Network Architectures. MATHEMATICS 13 : 9 Paper: 1471, 25 p., https://doi.org/10.3390/math13091471

[7] Mohammadi, M., Kouzani, A. Z., Bodaghi, M., & Zolfagharian, A. (2025). Inverse design of adaptive flexible structures using physical-enhanced neural network. Virtual and Physical Prototyping, 20(1). https://doi.org/10.1080/17452759.2025.2530732

[8] Kovács, L. (2025). Experiments with Neural Network for Optimization. In: Moldovan, L., Gligor, A. (eds) The 18th International Conference Interdisciplinarity in Engineering. Inter-Eng 2024. Lecture Notes in Networks and Systems, vol 1249. Springer, Cham. https://doi.org/10.1007/978-3-031-81685-7_6

[9] Khodadadi, N., Talatahari, S., & Gandomi, A. H. (2024). ANNA: Advanced neural network algorithm for optimisation of structures. Proceedings of the Institution of Civil Engineers-Structures and Buildings, 177(6), 529-551., https://doi.org/10.1680/jstbu.22.00083

[10] Grossmann, T. G., Komorowska, U. J., Latz, J., & Schönlieb, C. B. (2024). Can physics-informed neural networks beat the finite element method?. IMA Journal of Applied Mathematics, 89(1), 143-174. https://doi.org/10.1093/imamat/hxae011

[11] Javadi, A., Tan, T., & Zhang, M. (2023). Neural network for constitutive modelling in finite element analysis. Computer Assisted Methods In Engineering And Science, 10(4), 523-529.

[12] Meethal, R.E., Kodakkal, A., Khalil, M. et al. (2023). Finite element method-enhanced neural network for forward and inverse problems. Adv. Model. and Simul. in Eng. Sci. 10, 6. https://doi.org/10.1186/s40323-023-00243-1

[13] Badia, S., Li, W., & Martín, A. F. (2025). Compatible finite element interpolated neural networks. Computer Methods in Applied Mechanics and Engineering, 439, 117889. https://doi.org/10.1016/j.cma.2025.117889

[14] Abaqus 6.13 online documentation. Dassault Systems. 2015.
[15] Abaqus 2016. Scripting user's guide. Dassault Systems. 2016.