MALAYSIAN JOURNAL OF BIOENGINEERING AND TECHNOLOGY

Enhancing Electrical Engineering Education with Python AI: A Case Study at Pibulsongkram Rajabhat University

Sunun Tati[1], Thaweesak Tanaram[2,*], Nuttee Thungsuk[3], Akharakit Chaithanakulwat[3], Teerapod Naebnean[4]

[1]*Department of Electrical and Computer Engineering, Faculty of Engineering, Naresuan University, Phitsanulok 65000, Thailand*
[2]*Faculty of Industrial Technology, Pibulsongkram Rajabhat University, Phitsanulok 65000, Thailand*
[3]*Faculty of Science and Technology, Dhonburi Rajabhat University, Samut Prakan 10540, Thailand*
[4]*Faculty of Agricultural Technology and Industrial Technology, Nakhon Sawan Rajabhat University, Nakhon Sawan 60000, Thailand*

**Corresponding author: t.tanaram.t@psru.ac.th*

ARTICLE INFO

Received:8 September 2025
Accepted:22 September 2025
Online:30 September 2025
eISSN: 3036-017X

ABSTRACT

This paper specifically applies artificial intelligence (AI) techniques to power flow analysis, a core topic in the Power System Analysis course. Power flow problems involve iterative numerical computations that require computer-based solutions. The study was conducted at Pibulsongkram Rajabhat University, which currently offers an undergraduate program in Electrical Engineering. The researchers found that the utilization of AI programs requires high-performance computing systems to handle intensive processing tasks. Therefore, a series of tests was conducted using complex computational problems on laboratory-based computers in order to evaluate both the performance and the processing time of the computing systems for AI program execution. In this study, a 3, 9, 14, and 30-bus power system model was utilized as a test case. To evaluate the feasibility and performance of AI-assisted power flow analysis in an educational setting, the developed Python-based AI program, which uses Artificial Neural Networks (ANN), was executed on four types of computing platforms: a MacBook, a high-performance laptop, a mainstream consumer-grade laptop, and a standard desktop computer available in university laboratories. Experimental results showed that while the MacBook and high-performance laptop delivered the fastest in 2844 sec and 3986 sec of computation times, the other platforms also proved adequate for educational use. In contrast, the consumer-grade laptop and laboratory PC, in 7422 sec and 9244 sec of computation times, exhibited noticeably slower performance. Although computing performance varied, all tested platforms proved to be adequate for educational purposes, especially in supporting the teaching of AI applications and Python programming in power system analysis. This study demonstrates that Python-based AI tools can effectively enhance engineering education by supporting both technical knowledge and computational skills.

Keywords: Artificial intelligence in education, power flow analysis python, electrical engineering education, educational technology.

1. Introduction

1.1 Python-based AI program

Artificial Intelligence (AI) has increasingly played a significant role in the field of education, encompassing the development of instructional media, teaching management methodologies, question generation, and automated answering tools [1-3]. Nevertheless, there has been limited focus on educating students to develop AI systems aimed at solving engineering problems or assisting in scientific analysis. To address this gap, the Department of Electrical Engineering, Pibulsongkram Rajabhat University, has integrated into its curriculum the development of AI-based programming concepts using the Python language to solve problems in electrical engineering.

A free alternative to commercial platforms is the programming language Python, which is available under an open-source license. Python is a scripting language with a straightforward and easy to learn syntax. Scientific libraries like NumPy and SciPy are internally implemented in C, so that mathematical analysis and data manipulation routines are carried out efficiently [4]. Python has gained significant popularity for open-source projects, especially in scientific applications. Since a large variety of libraries are freely available in Python, Python applications can be easily extended with third-party libraries. They can also be parallelized on computational clusters without license or compatibility constraints. Consequently, many recently developed tools for power system analysis are implemented in Python [5-10].

Python, with its extensive ecosystem of AI and scientific computing libraries such as TensorFlow, PyTorch, Scikit-learn, Pandapower, and PyPSA, offers a cost-effective and versatile alternative [4][10-11]. It works well with simulation environments, making it useful for analyzing, modeling, and optimizing complex electrical systems [4-5], [11-12]. Students are expected to understand AI algorithms and Python programming and be able to design and deploy AI-based solutions for engineering analysis [1–3][5][13].

1.2 Power Flow Solution

Power flow analysis is the first and basic procedure for investigating problems in power system operation and planning. Depending on the generating unit and transmission system, it solves the steady state operation with voltages and branch power flow within the power system. Power Flow provides a steady-state operation of the power system, without considering system processes. So, the mathematical model of the power flow problem could also be a nonlinear algebraic equation system without differential equations. The power flow model is vital for various sorts of studies like short-circuit analysis, stability, and harmonic studies. This power flow model provides the network data and an initial steady-state condition for these studies. The model equations of the power flow problem are non-linear, so iterative methods such as Newton-Raphson, Gauss-Seidel, and fast-decoupled power flow methods are used to solve them [6-7][12].

In this paper, a review of the most widely used Python application tool, the artificial neural network (ANN) for power flow solution, is presented.

2. Methodology

2.1 Power Flow Problem

The power flow problem is fundamental in electrical power systems, providing information for the operation and planning of a grid. Given a set of operating conditions, it involves calculating the voltages at each bus and the power flowing through each transmission line. The Newton-Raphson method is a well-established iterative approach to solving these nonlinear algebraic equations efficiently due to its rapid convergence characteristics. The core Equations (1) and (2) covering the power flow problem are the power balance equations for each bus. For a bus i, the real power Pi and reactive power Qi are defined as follows [6-7][12]:

$$P_i = V_i \sum_{j=1}^{n} V_j \left(G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij} \right) \tag{1}$$

$$Q_i = V_i \sum_{j=1}^{n} V_j \left(G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij} \right) \tag{2}$$

Equations (1) and (2), where Vi represents the voltage magnitude at bus I, Gij and Bij are the conductance and susceptance of the line between buses i and j, and θij is the phase angle difference between buses i and j.

The iterative Newton-Raphson method updates guesses for the voltage magnitude and angle at each bus by solving a system of equations derived from the Taylor series expansion of the power balance equations. The Jacobian matrix J, in Equation (3) which contains the partial derivatives of the power mismatches with respect to the voltage magnitude and angle, plays a crucial role in this update:

$$J = \begin{bmatrix} \dfrac{\partial P}{\partial \theta} & \dfrac{\partial P}{\partial V} \\ \dfrac{\partial Q}{\partial \theta} & \dfrac{\partial Q}{\partial V} \end{bmatrix} \tag{3}$$

At each iteration, the power mismatch between the current state and the power specifications is calculated in Equations (4) and (5):

$$\Delta P = P_{Std.} - C_{Cal..} \left(V, \theta \right) \tag{4}$$

$$\Delta Q = Q_{Std.} - Q_{Cal.} \left(V, \theta \right) \tag{5}$$

Here, PStd. and QStd. in Equation (4) and (5) are the specified real and reactive power, while PCal. and QCal. are the calculated values from the current estimates of voltage magnitude V and phase angle θ

The Jacobian in Equation (6) is used to determine the changes ΔV and Δθ to be applied to the voltage magnitudes and phase angles to reduce the power mismatch:

$$\begin{bmatrix} \dfrac{\Delta \theta}{\Delta V} \end{bmatrix} = -J^{-1} \begin{bmatrix} \dfrac{\Delta P}{\Delta Q} \end{bmatrix} \tag{6}$$

The voltage magnitude V and phase angle θ at each bus are updated accordingly in Equations (7) and (8):

$$\theta_{new} = \theta + \Delta\theta \tag{7}$$

$$V_{new} = V + \Delta V \tag{8}$$

The iterative process continues until the power mismatches ΔP and ΔQ in Equations (7) and (8) are within acceptable tolerances, indicating convergence to a solution. This research aims to address the challenges posed by the integration of renewable energy sources into the power grid, which introduces variability and uncertainty into the power flow problem. By enhancing the Newton-Raphson method, the study seeks to improve the stability and efficiency of power system operations.

2.2 Artificial Neural Networks (ANN)

An ANN is a mathematical model that simulates the human brain's ability to learn and remember via the learning process. The concept of ANNs is to create mathematical models to mimic brain activity, as shown in Fig. 1. ANN neural simulations using the data inputs are multiplied by different weight values. The adjustment weight values are analyzed and interpreted by the activation function to make the correct or closest output. There are many positive results to using the ANN structures for different functions, such as feed-forward, feedback, and competitive. In addition, the ANN has learned to adjust the weights. Therefore, the network can learn and show the desired behaviour. The learning of ANN could be separated into two types: supervised learning and unsupervised learning. Multilayer perceptron (MLP) [14] consists of three different layers: an input layer, a hidden layer, and an output layer, in which the hidden layer may contain more than one layer, as shown in Fig. 2 [14]. Each layer consists of one or more nodes on which the data are sent from the data layer to the output layer, with the weight adjustment in each node. Moreover, a backpropagation technique [14] is a method that optimizes the weights of the nodes. interconnection of the difference between the output and the desired target.

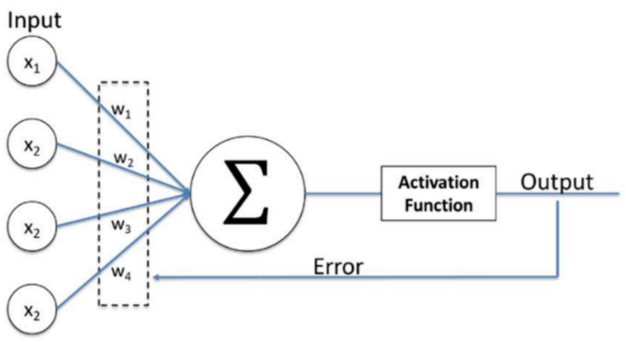

Fig. 1: Example of a simple machine learning model [5].

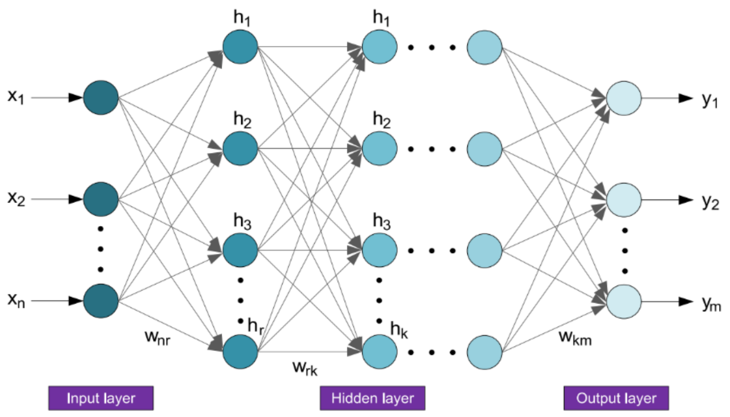

Fig. 2: Example of a simple machine learning model [6].

The training of a multilayer neural network requires the initialization of a weighted value whose value is a real number derived from an initial random value. The learning of a backpropagation technique on an ANN consists of the learning rate, the momentum constant, and the training function. The learning rate (η) is a coefficient that represents network learning, where values should range from 0.05 to 0.5 [14]. Note that the learning rate was high because the weight value of the network was high. On the other hand, the learning rate was low due to the low weight value of the network. The momentum constant (α) is a damping coefficient for suppressing excessive changes in the weighting, which can increase the stability of the network in another way. The momentum value should be between 0 and 1, and it must correspond to the learning rate; i.e., if the learning rate is high, then the momentum value should be low to ensure that the weight change is not too large. The train function has several functions available to train for each forecasting model.

The flowchart in Fig. 3 illustrates a common training process for a neural network. The process starts by loading the raw data and setting up the network's structure, including its starting weights and biases. After preparing the data, the training loop begins. In each loop, the network makes a forward pass to calculate the outputs for the hidden layer nodes. The resulting outputs are then compared with the expected targets to calculate the corresponding node errors. If a node error exceeds a predefined tolerance, the network updates its weights and biases through an optimization step, after which the forward pass is repeated. Once the node errors fall within the predefined tolerance, the algorithm proceeds to compute the average total error across the entire network. If this overall error also meets the acceptable threshold, the training process is considered complete. Conversely, if the total error exceeds the tolerance, the procedure returns to the data preparation phase to initiate another training epoch. This cycle repeats until the network attains the specified accuracy.

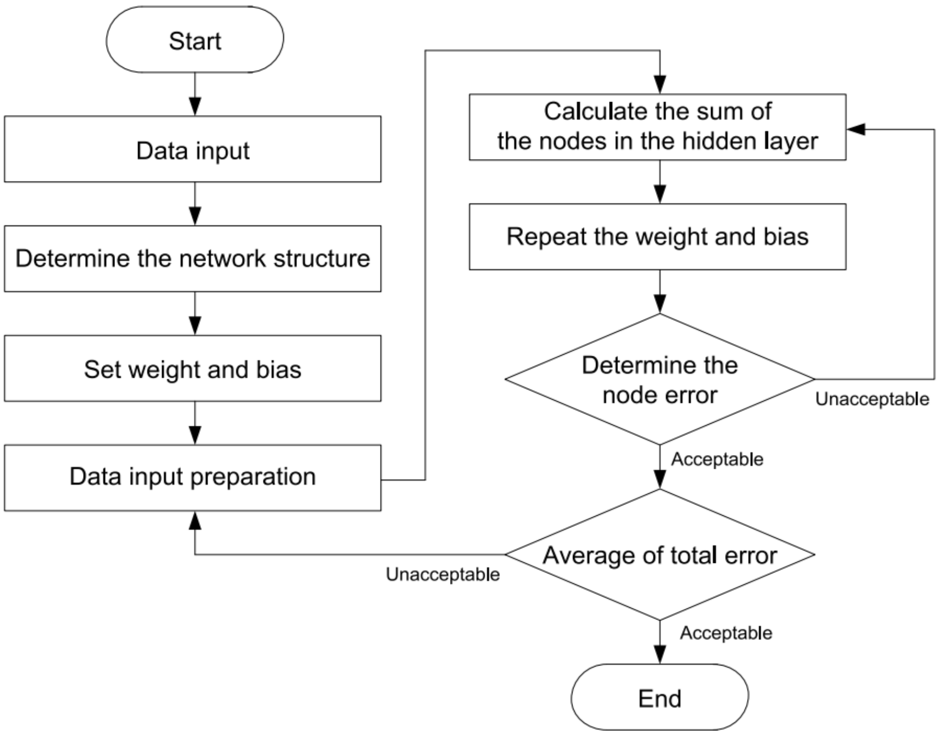

Fig. 3: Flow chart of ANN power flow analysis [6].

2.3 IEEE 30 Bus Test System

IEEE bus systems are used by researchers to implement new ideas and concepts. This technical note describes the details of the IEEE 30-bus system [12][15-16]. A power flow study uses simplified notation, such as a single line diagram and a per-unit system. The system consists of loads, capacitor banks, transmission lines, and generators. Fig. 4 depicts a part of the IEEE 30-bus system for ANN power flow calculation. The proposed solvers are to be tested in the following manner. For various IEEE test cases of different sizes, the computation time and the number of iterations in the ANN power flow analysis method were measured. An error threshold of 10−4 is set as the target accuracy, at which point the iterative algorithm was halted. The number of iterations indicates the convergence efficiency of different implementations of this method, while the computation time reflects the practical applicability of the solvers. Even at the beginning of a sentence.

The power flow analysis (also known as load-flow study) is an important tool involving numerical analysis applied to a power system. Unlike traditional circuit analysis, a power flow study usually uses simplified notation such as a one-line diagram and per-unit system, and focuses on various forms of AC power (i.e, reactive, real, and apparent) rather than voltage and current. Power flow analysis is being used for solving the power flow problem by the Newton-Raphson method and the Gauss-Seidel method. This sub-chapter will discuss all two methods generally on formula or mathematical step in order to solve power flow problem [8][17-18] as shown in Fig. 4. Example in PQ bus the value of voltage magnitude and its angle are unknown on the other hand PV bus, the parameter of active and Reactive voltages are known in the study of power flow analysis the frequency is constant [15-16]. In the ANN Python program of power flow study for the distribution system using the Newton-Raphson method for analysis, is performed by modelling different devices of the distribution system, Buses, lines, and loads connected through the system. Python programming to perform load flow analysis by taking different parameters of the distribution system for calculating the various types of analysis, i.e, static or dynamic [11].

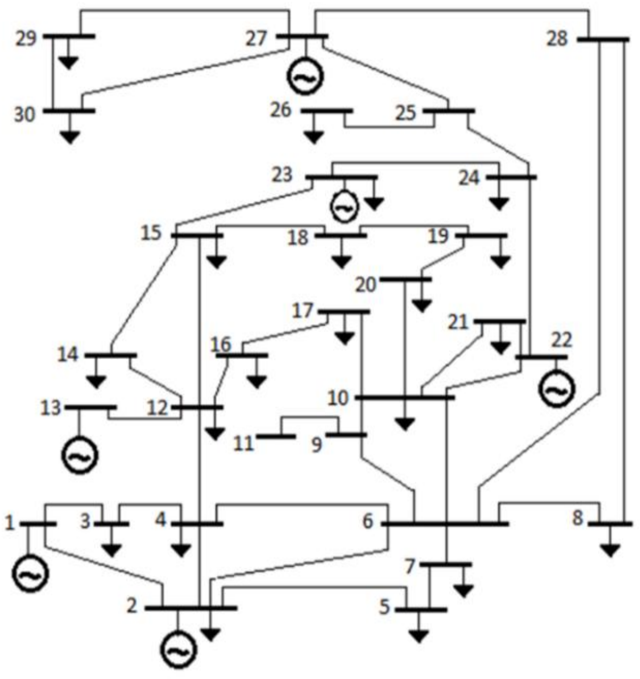

Fig. 4: IEEE Standard 30 Bus Test System [7].

The proposed ANN-based power flow analysis framework aims to predict system states directly from input parameters, thereby reducing the iterative computations required by traditional methods. The training process involves initializing ANN parameters, preparing representative training datasets, performing iterative forward and backward passes to minimize prediction errors, and updating system states until convergence. The following subsections detail each stage of the proposed training procedure.

2.3.1 Initialization

The process begins by initializing the voltage magnitudes V and phase angles θ. The initial weights of the artificial neural network (ANN) are assigned, along with the learning rate and momentum parameters. The error tolerance and the maximum number of iterations are specified, and the iteration counter is set to zero.

2.3.2 Prepare Data

The training dataset is prepared, comprising the input features and their corresponding target outputs.

2.3.3 Main Loop

The core of the algorithm consists of an iterative loop performing the following steps:

a. Forward Pass: A forward propagation is executed using the current ANN weights to generate predicted outputs from the input features.

b. Error Calculation: The mean squared error (MSE) between the predicted outputs and the target outputs is computed.

c. Convergence Check: The algorithm verifies whether the MSE falls below the predefined error tolerance or if the maximum number of iterations has been reached. If either condition is satisfied, the loop terminates.

d. Backpropagation: If convergence has not been achieved, backpropagation is performed to adjust the ANN weights based on the MSE, learning rate, and momentum.

e. System State Update: The updated voltage magnitudes V and phase angles θ are extracted from the adjusted ANN weights.

f. Iteration Increment: The iteration counter is increased, and the loop proceeds to the next loop.

2.3.4 Output Results

After the main loop has converged or reached the maximum iterations, the code outputs the final voltage magnitudes (V), angles (θ), and the performance metrics.

3. Results and Discussion

Computational testing to find the value of voltage and reactive power on the Load Bus and PV bus involves the apparent power variable on each busbar, then the admittance value on the line connected to each busbar, and the initial assumed voltage value of 1+0j on the Load Bus as well as PV buses. The parameter analysis was performed using an Artificial Neural Network (ANN) model designed to evaluate the performance of the system under investigation.

The ANN architecture was configured with an input layer corresponding to the measured parameters, one or more hidden layers optimized for feature extraction, and an output layer representing the predicted values. The activation functions, learning rate, and number of neurons per layer were determined through preliminary experiments to ensure model stability and convergence. To assess the model's robustness, the program was executed for 30 independent iterations. In each iteration, the dataset was processed through the ANN, and the resulting prediction error was recorded. The error metric employed was the Mean Squared Error (MSE), which provides a quantitative measure of the deviation between predicted and actual values. The collected error values from all iterations were statistically analysed to determine the mean and standard deviation. Fig. 5 shows these results, which were then compared with the IEEE standard.

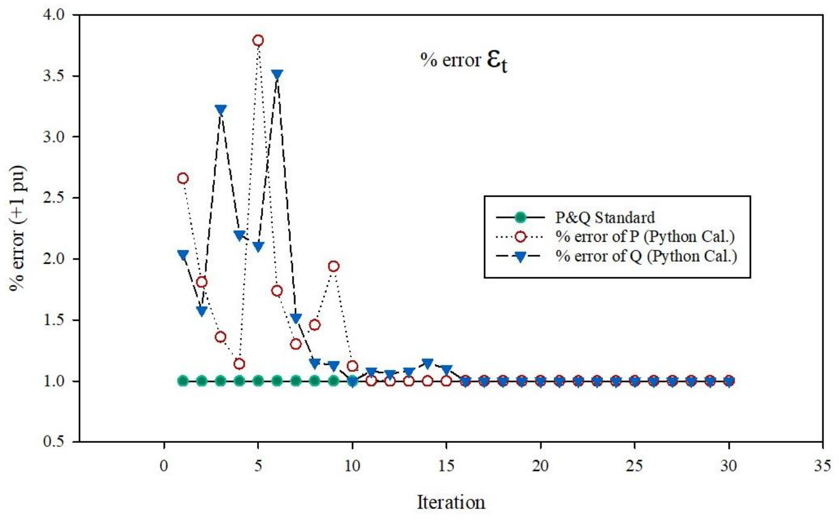

Fig. 5: ANN power flow analysis result.

The experimental evaluation was carried out in the Computer Laboratory of the Department of Electrical Engineering, Pibulsongkram Rajabhat University. The laboratory desktop computers, which possess relatively limited processing capabilities, were employed for testing purposes. In addition, a comparative assessment was conducted using other computing platforms that are expected to be utilized by students during their coursework. The specifications of all systems employed in the experiments are summarized in Table 1.

Table 1: System Specifications for Python AI Program

Platform	Processor (CPU)	RAM	Storage Type	GPU	Operating System
MacBook Pro (2025)	Apple M4 (16-core CPU)	32 GB Unified	1 TB SSD	Integrated (Apple GPU)	macOS Sonoma
ASUS ZenBook (High Performance)	Intel Core i7-12700H @ 2.3GHz	16 GB DDR4	512 GB SSD	NVIDIA RTX 3060	Windows 11 Pro
Consumer-grade Laptop	Intel Core i5-1135G7 @ 2.4GHz	8 GB DDR4	256 GB SSD	Integrated Iris Xe	Windows 10 Home
Lab Desktop PC	AMD Ryzen 3 3200G @ 3.6GHz	8 GB DDR4	1 TB HDD	Integrated Vega 8	Windows 10 Pro

In addition to the computing platforms, the study also incorporated standard IEEE test systems for power system analysis. Some of the most commonly utilized IEEE standard networks include the IEEE 3-bus, IEEE 9-bus, IEEE 14-bus, and IEEE 30-bus systems. In these test systems, the number of buses corresponds directly to the number of nodes in the power system network being analysed.

Fig. 7 illustrates the program execution test results, where the processing times were compared across IEEE test systems with different numbers of buses. The graph in Fig. 7 illustrates the program execution time (in seconds) for Artificial Neural Network (ANN) power flow analysis across various computing platforms as the complexity of the power system network increases. The x-axis represents the number of buses in the IEEE standard test systems: 3, 9, 14, and 30 buses. The y-axis represents the time in seconds it took for the analysis to be completed. The graph compares the performance of four different computing platforms: 1) MacBook Pro (M4), 2) High Performance, 3) Consumer-grade Laptop, and 4) Lab Desktop PC, based on the time taken to perform ANN power flow analysis. The MacBook Pro (M4), represented by the blue line, consistently demonstrates the lowest processing time, making it the most efficient platform for this task. The High-Performance system, shown with an orange line, is the second fastest, with processing times that are higher than the MacBook Pro but lower than the other two platforms. As the power system becomes more complex, the processing time on the consumer-grade laptop gray line increases much more sharply, performing worse than both the MacBook Pro (M4) and the high-performance system. The Lab Desktop PC, represented by the yellow line, is the least efficient of the four, consistently showing the highest processing times across all bus systems. The result clearly demonstrates a positive correlation between the number of buses in the power system and the time required for ANN power flow analysis. As the network complexity increases (more buses), the processing time for all platforms also increases.

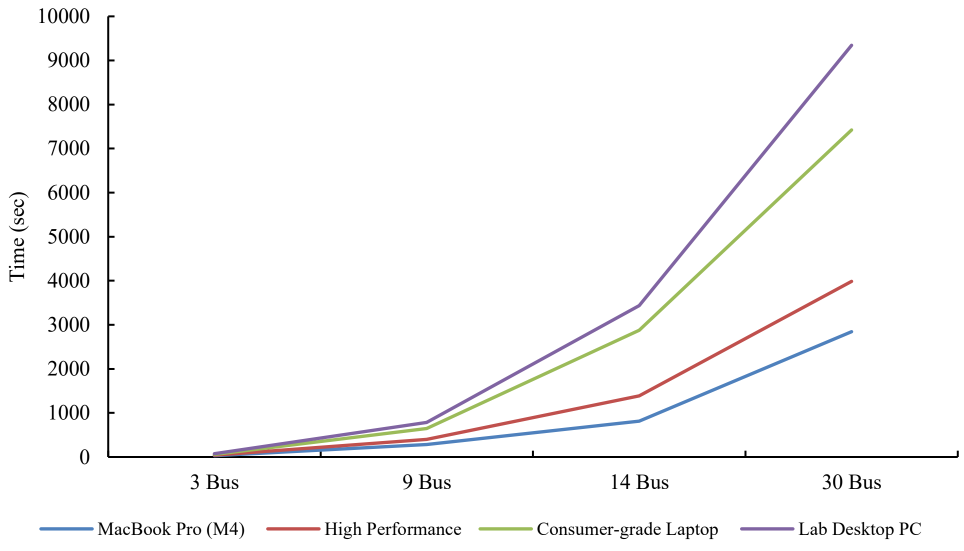

Fig. 7: Comparison of the time of ANN power flow analysis.

4. Conclusion

The experimental results indicate that the processing time is influenced by both the computational capability of the hardware and the size of the test system. The results of the performance evaluation, in which the program processing times are compared for IEEE test systems with varying numbers of buses. The data clearly show that as the number of buses increases, the computational time also increases, particularly for systems with larger network sizes. The performance gap between high-end and consumer-grade hardware becomes more significant with increasing system size. Power flow problems involve iterative numerical computations that require computer-based solutions. In this study, a 3, 9, 14, and 30-bus power system model was utilized as a test case. To evaluate the feasibility and performance of AI-assisted power flow analysis in an educational setting, the developed Python-based AI (ANN) program was executed on three types of computing platforms: a high-performance laptop, a mainstream consumer-grade laptop, and a standard desktop computer available in university laboratories. 30 Bus test system experimental results showed that while the high-performance laptop and MacBook delivered the fastest in 2844 sec and 3986 sec of computation times, the other platforms also proved adequate for educational use. In contrast, the consumer-grade laptop and laboratory PC exhibited noticeably slower times in 7422 sec and 9244 sec. However, consumer-grade laptops and Laboratory PC can be used to learn this complex programming even when it takes a higher processing time than a MacBook and a high-performance laptop.

Overall, this study shows that Python-based AI tools, using common and affordable computers, can help improve electrical engineering education. By mixing theory about power flow with AI computer methods, students learn important technical ideas and practical programming skills. Python is easy to use and has many helpful libraries, making it a low-cost and effective choice for bringing modern AI techniques into engineering courses.

References

[1] Fahimirad M, Ketmani SS. A review on application of artificial intelligence in teaching and learning in educational contexts. Int J Learn Dev, 2018;8(4):106-118.

[2] Faraasyatul'Alam G, Wiyono BB, Burhanuddin B, Muslihati M, Mujaidah A. Artificial Intelligence in Education World: Opportunities, Challenges, and Future Research Recommendations. Fahima, 2024;3(2):223-234.

[3] Chen L, Chen P. Lin Z. Artificial intelligence in education: A review. IEEE Access, 2020;8:75264-75278.

[4] Thurner L, Scheidler A, Schäfer F, Menke JH, Dollichon J, Meier F, Meinecke S, Braun M. Pandapower—an open-source Python tool for convenient modeling, analysis, and optimization of electric power systems. IEEE Trans Power Syst, 2018;33(6):6510-6521.

[5] Chamim ANN, Putra KT, Al Farisi MF. Power Flow Analysis of Electrical Network Systems Using Gauss-Seidel Method and Python. Emerg Inf Sci Technol, 2023;4(1):28-36.

[6] Dharamjit D. Load flow analysis on IEEE 30 bus system. Int J Sci Res Publ, 2012;2(11):1.

[7] Milano F. An open source power system analysis toolbox. IEEE Trans Power Syst, 2005;20(3):1199-1206.

[8] Ali M, Baluev D, Ali MH, Gryazina E. A novel open source power systems computational toolbox. Proc North Am Power Symp, 2021:1-6.

[9] Nawaz S, Bhardwaj E, Sharma DR, Parmar D. Comparison of Different Simulation Tools for Load Flow Analysis in Power Systems. Int J Adv Eng Manag Sci, 2023;9(5).

[10] Brown T, Hörsch J, Schlachtberger D. PyPSA: Python for Power System Analysis, J Open Res Softw. 2018;6(4):1-15.

[11] Sharma R, Dhillon J. Pypsa: Open source python tool for load flow study. J Phys Conf Ser, 2021;1854(1):012036.

[12] Grishin E, Gerasimov G, Gryazina E. Overview of Python power flow solvers. Proc Int Youth Conf Radio Electron Electr Power Eng, 2024:1-6

[13] Tanaram T, Tati S, Thungsuk N, Chaithanakulwat A. Application of Python-Base Open Source Tool for Power Flow Analysis. Proc Int Conf Sci Technol Educ, 2024

[14] Thungsuk N, Phumemek P, Chaithanakulwat A, Savangboon T, Tanaram T, Tati S, Mungkung N, Arunrungrusmi S, Tanitteerapan T, Tunlasakun K, Yuji T. Application of an Artificial Neural Networks Model

for Prediction of Instability Phenomena in Low Current Vacuum Arc by Cathode Spot Model. IEEE Trans Plasma Sci, 2024;52(4).
[15] Illinois Center for a Smarter Electric Grid, Online, Available FTP: http://publish.illinois.edu/smartergrid/
[16] IEEE_PES-IAS, Online, Available FTP: http://sas.ieee.ca/pesias/seminar_slides/IEEE_PES-IAS_Chapter_24_01_13.pdf
[17] Zhou M, Zhou S, Internet, Open-source and Power System Simulation, Proc IEEE PES Gen Meet, 2007.
[18] Zimmerman RD, Murillo-Sanchez CE, Thomas RJ. MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education, IEEE Trans Power Syst, 2011;26(1):12-19.