

DEEP NEURAL NETWORK CONTROLLER-BASED POWER QUALITY IMPROVEMENT USING SHUNT ACTIVE POWER FILTER



H.M.R.T. Jayanthi 180264K

M.A.M. Nijas 180421P

N.V.S. Sawandi 180575N

EE4203 - Design Project

Department of Electrical Engineering

University of Moratuwa

June 2023


Declaration

We declare that this project includes our own work, and this report does not incorporate without acknowledgment any material previously submitted for a Degree or Diploma in any other University or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgment is made in the text. Also, we hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute our report, in whole or in part in print, electronic, or another medium. We retain the right to use this content in whole or part in future work (such as a book or article).


H.M.R.T. Jayanthi

180264K Signature.....  ...

M.A.M. Nijas

180421P Signature... 

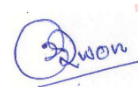
N.V.S. Sawandi

180575N Signature..... 

Date : 05/06/2023

Above candidates have carried out research for B.Sc. Engineering undergraduate design project under my supervision

Signature of the supervisor - Prof. Udayanga Hemapala



.....

Signature of the co-supervisor – Prof. Sisil Kumarawadu

.....

Date:

Acknowledgement

The successful completion and outcome of this Final Year Design Project required a lot of guidance and immense support from many people. We hereby use this opportunity to express our sincere gratitude to each person who guided and supported us throughout the project. We would like to extend our heartfelt gratitude to our project supervisor, Prof. Udayanga Hemapala and co-supervisor Prof. Sisil Kumarawadu for the enormous support and encouragement given throughout the project. We are greatly indebted to them for the continued guidance and useful advice amidst this difficult time. We would like to express our gratitude to the academic staff of the Department of Electrical Engineering for the guidance, suggestions, and support during the project and progress reviews. Finally, we would like to thank all our colleagues for their never-ending support during the project. In addition, we would like to extend our gratitude to our parents, families, and others who have not been mentioned here.

Abstract

Power quality issues, such as harmonics, reactive power, and voltage distortions, have become a significant concern in modern power systems due to the increasing use of nonlinear loads. The presence of harmonics in power systems can lead to various adverse effects, including increased power losses, overheating of equipment, reduced power quality, and interference with communication and control systems. Therefore, mitigating harmonics and maintaining a sinusoidal waveform is crucial for ensuring reliable and efficient operation of power systems.

Earlier techniques for harmonic reduction primarily relied on passive filters, which are simple and cost-effective solutions. However, passive filters have limitations in terms of their fixed operating characteristics and inability to adapt to changing load conditions. Another widely used technique for harmonic reduction is the application of shunt active power filters (SAPFs). SAPFs offer improved performance compared to passive filters, as they can provide dynamic compensation and adapt to varying load conditions.

A DNN controller is proposed for a fast estimation of the compensating current of a shunt active power filter. The DNN controller is designed to learn the nonlinear mapping between the reference current and the actual current of the SAPF, enabling accurate and adaptive control under varying load conditions. The DNN controller is trained using a large dataset generated through simulations covering a wide range of operating scenarios. The DNN controller can be easily updated or retrained with new data, allowing it to adapt to evolving power system conditions and accommodate future load changes. This feature significantly improves the long-term effectiveness and reliability of the power quality improvement system.

The proposed DNN controller based SAPF provides an efficient and intelligent solution for power quality improvement in modern power systems. Its adaptive and accurate control capability, combined with flexibility and scalability, makes it a promising approach for mitigating power quality issues caused by nonlinear loads.

Table of Contents

Declaration	2
Acknowledgement.....	3
Abstract	4
Table of Contents.....	5
List of Tables.....	8
Abbreviations	9
1. Introduction	10
1.1. Power Quality Issues	10
1.2. Total Harmonic Distortion.....	11
1.3. Existing Techniques for THD Reduction.....	12
1.4. Background and Motivation.....	13
1.5. Project Objective and Scope	15
1.6. Methodology	15
1.6.1 Project Flow	16
2. Literature Review	17
2.1 Shunt Active Power Filter.....	17
2.2. Instantaneous Power Theory	18
2.2.1. Clarkes Transformation.....	18
2.2.2. Calculation Of P And Q From Voltage and Current Vectors	19
2.2.3. Use Of the P-Q Theory for Shunt Current Compensation	21
2.2.4. Designing Hysteresis Band Current Controller	22
2.3. Control Of DC-Link Voltage	23
2.4. ANN Controller.....	25
2.5. Research Gap	26
3. Design and Simulation.....	27
3.1 SRF based SAPF	27
3.1.1. Case 1 - Non-linear load without Compensator	28
3.1.2. P, Q Components Calculation.....	28
3.1.3. Compensated Reference Current Calculation	29
3.1.4. Hysteresis Band Current Controller.....	30
3.1.5. Voltage Source Inverter	31

3.1.6. Case 2 - System Mathematical Model	32
3.2 Case 3 - PI Controller based SAPF	33
3.3 Case 4 - DNN Controller based SAPF	34
3.4 Case 5 - ANN Predictive Controller based SAPF	35
3.4.1. Model Architecture.....	36
3.4.2. Model Training	37
3.4.3. Model Evaluation.....	38
3.4.4. Results and Analysis	38
4. Results and Discussion	41
4.1 Case-1 Results	41
4.2 Case-2 Results	42
4.3 Case-3 Results	44
4.4 Case 4 - Results	45
4.5 Case 5 - Results	46
4.6 Results Comparison.....	47
5. Conclusion	48
4. References.....	49

List of Figures

Figure 1 - The shunt active power filter topology	7
Figure 2 - Reference Compensating Current Calculation.....	21
Figure 3 - Arrangement of SAPF.....	22
Figure 4 - MATLAB Model of Non-linear load without Compensator	28
Figure 5 - P, Q Components Calculation.....	29
Figure 6 - Compensated Reference Current Calculation	30
Figure 7 - Hysteresis Band Current Controller.....	31
Figure 8 - Voltage Source Inverter.....	31
Figure 9 - Block diagram of the SRF based SAPF.....	32
Figure 10 - MATLAB Model of SRF based SAPF	33
Figure 11 - MATLAB Model of PI Controller based SAPF	34
Figure 12 - Neural network implementation for PI controller.....	34
Figure 13 - Training & Validation Loss Curve	39
Figure 14 - Source, Load current without compensator	Error! Bookmark not defined.
Figure 15 - THD% without compensator	Error! Bookmark not defined.
Figure 16 - Source, Load, Compensating Current of SRF based SAPF.....	Error! Bookmark not defined.
Figure 17 - Source Voltage, Source Current, DC link voltage of SRF based SAPF	Error! Bookmark not defined.
Figure 18 - Reference Compensating, Compensating Current of SRF based SAPF	41
Figure 19 - THD% of SRF based SAPF.....	41
Figure 20 - Source Voltage, Source Current, DC link voltage of PI controller based SAPF	41
Figure 21 - Source, Load, Compensating Current of PI controller based SAPF	41
Figure 22 - THD% of PI controller based SAPF	41
Figure 23 - Source, Load, Compensating Current of DNN controller based SAPF.....	42
Figure 24 - Source Voltage, Source Current, DC link voltage of Predictive controller based SAPF.....	42
Figure 25 - Reference Compensating, Compensating Current of Predictive controller based SAPF	42
Figure 26 - Source, Load, Compensating Current of Predictive controller based SAPF.....	47

List of Tables

Table 1 - System Parameters 27

Table 2 - Results Comparison..... 47

Abbreviations

SAPF – Shunt Active Power Filter

THD - Total Harmonic Distortion

APF – Active Power Filter

SRF - Synchronous Reference Frame

DNN – Deep Neural Network

ANN - Artificial Neural Network

1. Introduction

In addition to the problem area and history of the research, this chapter will discuss the goals and objectives of the study, its novelty, its challenges, and its scope. This chapter's objective is to provide a broad overview of the research being done.

1.1. Power Quality Issues

Power quality is a critical aspect of modern power systems, and ensuring high-quality power delivery is of paramount importance. Power quality refers to the reliability, stability, and cleanliness of the electrical power supplied to consumers. It encompasses various factors, including voltage stability, frequency control, harmonics, voltage sags and swells, and interruptions. The importance of maintaining high-quality power delivery in modern power systems cannot be overstated. It directly impacts the performance, efficiency, and reliability of electrical systems and equipment.

While other power quality issues such as voltage fluctuations, power interruptions, and voltage imbalances are important, eliminating harmonics holds particular significance due to its wide-ranging impact on equipment performance. Harmonics can have detrimental effects on the performance and lifespan of sensitive electrical equipment. Nonlinear loads, such as variable speed drives, electronic devices, and power converters, generate harmonics that can disrupt the normal operation of equipment. Harmonics cause excessive heating, increased vibration, and additional stress on equipment components, leading to premature failures, reduced efficiency, and increased maintenance costs. By eliminating harmonics, the reliability and performance of equipment can be significantly improved.

1.2. Total Harmonic Distortion

The sources of harmonics in power systems are primarily nonlinear loads, which include electronic devices, power electronic converters, and equipment with rectifiers or inverters. Nonlinear loads draw non-sinusoidal currents from the power supply, leading to distorted waveforms. Harmonics cause distortions in the voltage and current waveforms, resulting in waveform deformations.

Harmonics introduce deviations from the sinusoidal shape of the voltage waveform. These distortions result in waveform irregularities such as flat tops, notches, and uneven peaks. Voltage waveform distortion affects the performance of electrical devices, particularly those sensitive to voltage quality. Nonlinear loads that generate harmonics can experience voltage distortion at their terminals, leading to reduced efficiency, increased heating, and potential malfunctions.

Harmonics also affect the current waveform, introducing deviations from the ideal sinusoidal shape. Nonlinear loads draw non-sinusoidal currents, characterized by distorted waveforms with high-frequency components. These harmonic currents can lead to increased losses, overheating, and reduced efficiency in electrical equipment and distribution systems.

The presence of harmonics elevates the Total Harmonic Distortion (THD) level, which quantifies the harmonic content in a waveform. By dividing the fundamental amplitude by the square root of the sum of the squares of the harmonic amplitudes, THD is calculated. High THD levels indicate significant harmonic distortion, compromising power quality. Excessive THD can result in equipment malfunctions, inaccurate metering, and increased energy losses.

1.3. Existing Techniques for THD Reduction

There are several existing techniques for THD (Total Harmonic Distortion) reduction in power systems. Passive filters are designed using passive components such as resistors, capacitors, and inductors. They are connected in parallel with the load to provide a low-impedance path for the harmonic currents. Passive filters work by absorbing or diverting the harmonic currents, reducing their impact on the power system. However, passive filters are fixed in their harmonic elimination capabilities and may not be effective for a wide range of harmonic frequencies.

Multi-pulse transformers, such as 12-pulse or 24-pulse transformers, are used to reduce harmonics in power systems. These transformers utilize phase shifting and additional winding configurations to cancel out certain harmonics. By introducing phase shifts between the transformers' primary windings, the harmonics can be distributed across multiple phases, reducing their impact on the system.

APFs use active electronic components and advanced control algorithms to detect and inject compensating currents that cancel out the harmonic currents. APFs can be classified into two types: shunt active power filters and series active power filters. Shunt active power filters are connected in parallel with the load and inject currents to cancel out the harmonics, while series active power filters are connected in series with the load and inject voltages to mitigate harmonics. APFs provide more flexibility and accuracy in harmonic compensation compared to passive filters.

1.4. Background and Motivation

SAPFs provide active compensation for harmonics, whereas conventional methods such as passive filters offer passive compensation. Active compensation means that SAPFs generate compensating currents that actively cancel out the harmonic currents, resulting in effective harmonic reduction. In contrast, passive filters only provide a low-impedance path for harmonic currents to flow through, without actively mitigating them.

SAPFs exhibit a fast and dynamic response to changes in the harmonic conditions. They can quickly detect and respond to variations in the harmonic currents, allowing for real-time compensation. This dynamic response ensures that harmonics are continuously monitored and mitigated, even in scenarios with varying loads or changing harmonic profiles. Conventional methods may have slower response times and may not be as effective in adapting to dynamic harmonic conditions.

The SRF-based SAPF operates by measuring the distorted current and generating compensating currents in real-time to cancel out harmonics and correct other power quality issues. The key principle behind SRF control is to transform the distorted current signals from the time domain to the synchronous reference frame, where the positive- and negative-sequence components can be extracted separately. This transformation allows for the accurate extraction and analysis of the harmonic components, enabling precise compensation.

DNN-based SAPF can provide more accurate and fast harmonic compensation compared to conventional Synchronous Reference Frame (SRF) method. The deep learning algorithms employed in DNN models can capture intricate relationships between input measurements and desired output responses.

By training the DNN model with a diverse dataset of harmonic scenarios, it can provide accurate and fast estimation of the compensating current. The adaptive nature of DNN-based SAPF enables it to adapt and learn from changing harmonic conditions and system dynamics. It can continuously analyze and adjust its control parameters based on real-time measurements, allowing for optimal harmonic compensation. This adaptability ensures effective harmonics reduction even in scenarios with varying load characteristics or harmonic profiles, leading to improved overall system performance.

The motivation is to overcome the limitations of conventional method and provide an innovative solution for precise and adaptive harmonics reduction. By integrating deep neural network (DNN) controllers with a shunt active power filter (SAPF), the aim is to enhance the accuracy, adaptability, and efficiency of harmonics mitigation.

1.5. Project Objective and Scope

Our primary objective in this research project is to achieve accurate and fast reduction of harmonics under varying load conditions. For that we focus on developing a DNN controller based SAPF that can effectively learn and capture the complex relationships between input measurements (such as voltage and current waveforms) and desired output responses (such as compensating currents). Sub objective is to evaluate the performance of the proposed DNN-based harmonics reduction technique and compare it with conventional SRF based SAPF.

1.6. Methodology

- Problem identification and understanding of the requirements.
- Carrying out a comprehensive literature review and gathering information on existing works related to the project scope.
- Identifying the strengths and weaknesses of the relevant literature and critically evaluating the essential findings and conclusions.
- Create an optimum design that serves the objectives well.
- Model, simulate and analyze the proposed design.
- Documentation

1.6.1 Project Flow

- First phase

In here we researched the problems and possible solutions for reducing THD. Hand calculations were involved in this phase.

- Second Phase

In that period, we develop the mathematical model for SRF based SAPF using MATLAB Simulink. We get a data set for train the DNN model.

- Third Phase

Finally, we modify existing SAPF model with DNN controller. Get Result and Conclusion about the overall project. Compare it with conventional SRF based SAPF.

This chapter gives a thorough explanation of why this research is done and includes all necessary references. As a result, we briefly covered the research project's background in this section. This chapter explains why authors should address this issue, what implications have resulted from it, and why we should fill this gap in our knowledge. Not only that, but this chapter also focused on the novelty of this study.

2. Literature Review

This chapter gives a comprehensive summary of the domain of THD reduction control techniques using SAPF carried out in various literatures. A better overview of the domain can be gained after reading this chapter.

2.1 Shunt Active Power Filter

The shunt active power filter operates as a current source, injecting harmonic components with a phase shift of 180 degrees that are equal to those produced by the load to balance the load current by injecting an equal but opposite compensation current. To reduce the harmonic content of the source current, the utility generates compensatory current. To make the source current entirely sinusoidal, a voltage source inverter is typically utilized as an active power filter. It generates nonlinearities that are the opposite of those in the load.

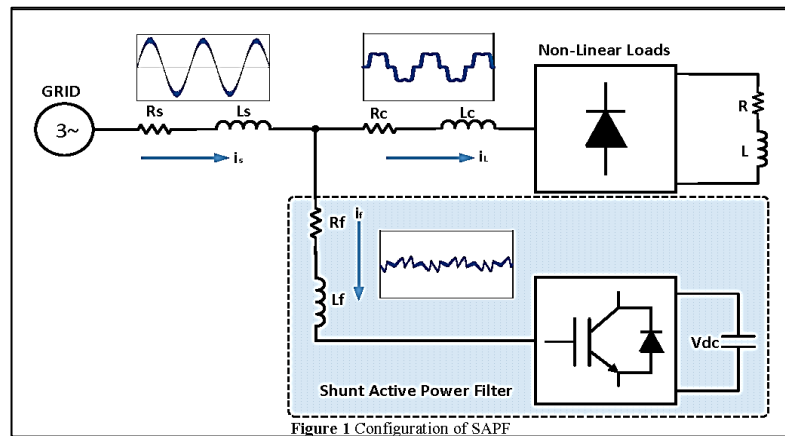


Figure 1 - The shunt active power filter topology

2.2. Instantaneous Power Theory

P-Q theory is another name for instantaneous power theory. P-Q theory is founded on the transition of abc to $\alpha\beta 0$. The instantaneous powers described in the time domain form the foundation of P-Q THEORY. It can be used with or without a neutral wire in 3-phase systems. It is valid in both the steady state and the transitory state. The construction of controllers for power filtering can be done extremely effectively and efficiently using this principle. The P-Q theory specifies instantaneous power on these axes and transforms currents and voltages on 0 axes. The 3-phase voltages and currents are converted to $\alpha\beta 0$ stationary reference frames using the 0 transformation, also referred to as CLARKE'S transformation.

2.2.1. Clarkes Transformation

The Clarke's transformation or $\alpha\beta 0$ transformation converts the three-phase instantaneous voltage (V_a , V_b , and V_c) on the abc axes to the $\alpha\beta 0$ axes (V_0 , V_α , and V_β). According to, the Clarkes transformation matrix,

$$\begin{bmatrix} V_0 \\ V_\alpha \\ V_\beta \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix}$$

The inverse Clarkes transformation matrix is given by,

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \frac{1}{\sqrt{2}} & 1 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{1}{\sqrt{2}} & -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} V_0 \\ V_\alpha \\ V_\beta \end{bmatrix}$$

The same formulas still apply to three-phase currents. Being power invariant is one of the benefits of the $\alpha\beta 0$ transformation. They separate the abc components from the zero sequence components. There are no zero sequence components in a three-phase, three-wire system, therefore i_0 can be taken out. Like this, there are no zero sequence voltages in a balanced four-wire system, therefore V_0 can be removed. The Clarkes and inverse Clarkes matrices can therefore be rewritten as,

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix}$$

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix}$$

While the α, β axes are orthogonal to one another, the abc axes are specifically 120 degrees apart from one another.

2.2.2. Calculation Of P And Q From Voltage and Current Vectors

From α, β axes, Instantaneous voltage vector is defined as,

$$e = V_\alpha + j V_\beta$$

The instantaneous current vector is defined as,

$$I = i_\alpha + j i_\beta$$

In a complex plane, where the real axis is the α axis and the imaginary axis is β the axis, these instantaneous vectors can be represented. These values depend on the instantaneous voltages and currents, therefore being functions of time.

We obtain e and I from the definitions given above.

$$P_{3\phi} = V_\alpha i_\alpha + V_\beta i_\beta$$

$$Q_{3\phi} = V_\beta i_\alpha - V_\alpha i_\beta$$

Above equations can be written in matrix form.

$$\begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} v_{\alpha} & v_{\beta} \\ -v_{\beta} & v_{\alpha} \end{bmatrix} \begin{bmatrix} i_{\alpha} \\ i_{\beta} \end{bmatrix}$$

The currents i_{α} , i_{β} in terms of p , q , which serve as the foundation for calculating the compensating currents, can be obtained by taking the inverse of the aforementioned matrix.

$$\begin{bmatrix} i_{\alpha} \\ i_{\beta} \end{bmatrix} = \frac{1}{v_{\alpha}^2 + v_{\beta}^2} \begin{bmatrix} v_{\alpha} & v_{\beta} \\ v_{\beta} & -v_{\alpha} \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix}$$

The above current components can be defined as Instantaneous active current on the α axis,

$$i_{\alpha p} = \frac{v_{\alpha}}{v_{\alpha}^2 + v_{\beta}^2} p$$

Instantaneous reactive current on the α axis,

$$i_{\alpha q} = \frac{v_{\beta}}{v_{\alpha}^2 + v_{\beta}^2} q$$

Instantaneous active current on the β axis,

$$i_{\beta p} = \frac{v_{\beta}}{v_{\alpha}^2 + v_{\beta}^2} p$$

Instantaneous reactive current on the β axis,

$$i_{\beta q} = \frac{-v_{\alpha}}{v_{\alpha}^2 + v_{\beta}^2} q$$

Similarly the instantaneous powers can be defined as $P_{\alpha p}$, $P_{\alpha q}$, $P_{\beta p}$, $P_{\beta q}$

Where,

$$p_{\alpha p} = v_{\alpha} \cdot i_{\alpha p} = \frac{v_{\alpha}^2}{v_{\alpha}^2 + v_{\beta}^2} p$$

$$p_{\alpha q} = v_{\alpha} \cdot i_{\alpha q} = \frac{v_{\alpha} v_{\beta}}{v_{\alpha}^2 + v_{\beta}^2} q$$

$$p_{\beta p} = v_{\beta} \cdot i_{\beta p} = \frac{v_{\beta}^2}{v_{\alpha}^2 + v_{\beta}^2} p$$

$$p_{\beta q} = v_{\beta} \cdot i_{\beta q} = \frac{-v_{\alpha} v_{\beta}}{v_{\alpha}^2 + v_{\beta}^2} q$$

2.2.3. Use Of the P-Q Theory for Shunt Current Compensation

The compensation of unwanted currents is a crucial application of the p-q theory. If a source is powering a nonlinear load that has a shunt compensator to make up for it. The shunt compensator functions as a three-phase regulated current source that can draw any collection of reference currents that are selected at arbitrary.

One can distinguish the average and oscillating components of the computed real and imaginary power of the load (P and Q). The undesirable components of the loads' real and imagined powers that need to be compensated as chosen. The terms -Pc and -Qc stand for the power that needs to be adjusted. In order to stress that the compensator should produce an exact inverse of the undesired power drawn by the non-linear load, a compensating current is included. Currently, it is accepted practice for source current to include some load current and compensatory current. Then the three-phase compensating current references i_{ca} , i_{cb} , and i_{cc} are instantaneously calculated using the inverse transformation from $\alpha\beta$ to abc.

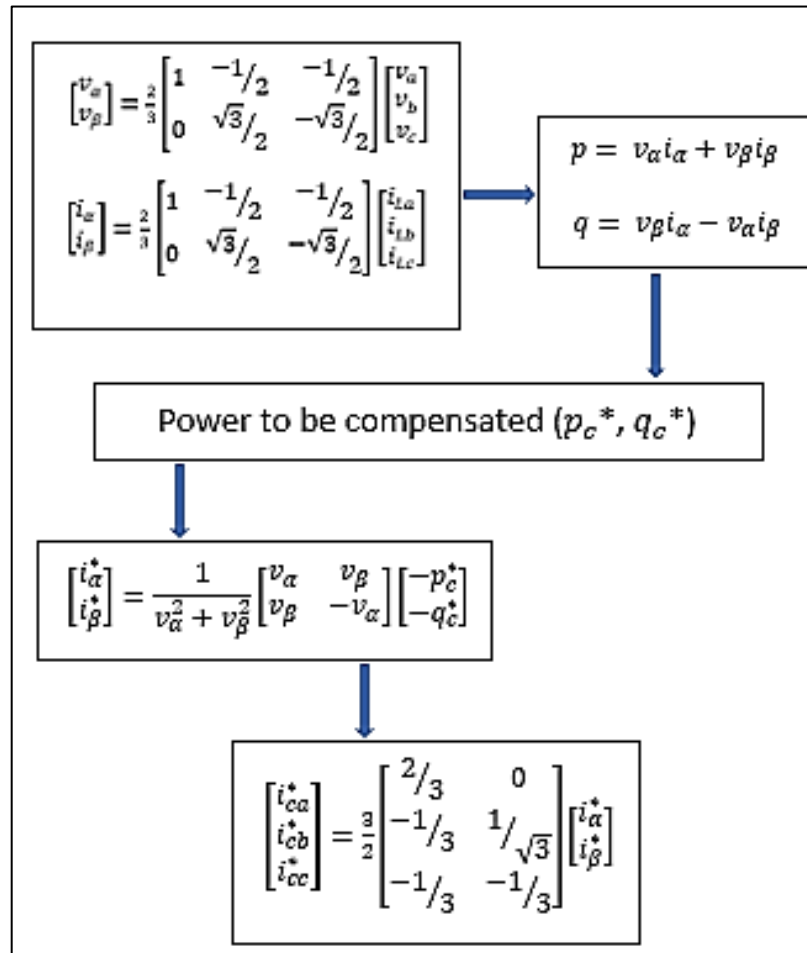


Figure 2 - Reference Compensating Current Calculation

The block diagram for the generation of compensating currents is shown in detail in the picture above. The Clarke's transformation converts the three-phase voltages and currents to $\alpha\beta$ axes. The instantaneous powers are then determined. The compensating powers are then determined. Then i_α , i_β are computed. Following their knowledge, the inverse Clarke transformation is carried out, and the compensatory currents (I_{ca} , I_{cb} , and I_{cc}) are determined.

2.2.4. Designing Hysteresis Band Current Controller

The SAPF switching pattern is decided by the current controller. The switching logic is expressed as follows:

- In the inverter leg, the upper switch is "OFF" and the lower switch is "ON" if $I_{\text{measured}} > I_{\text{reference}}$.
- The upper switch in the inverter leg should be "ON" and the bottom switch should be "OFF" if $I_{\text{measured}} < I_{\text{reference}}$.

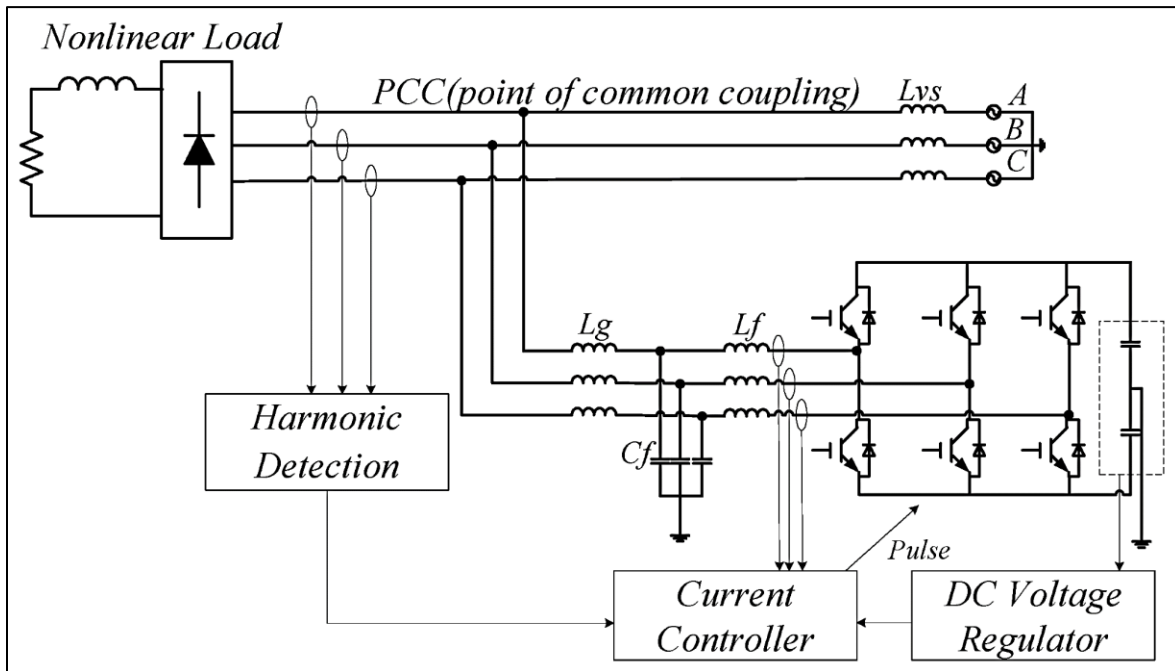


Figure 3 - Arrangement of SAPF

For all 3 legs, the identical switching sequence is used. There is no possibility of a dead short because the signals sent to the two switches on the same leg are complementary. The current generated by the inverter increases when the top switch is set to "ON" because a positive voltage is applied. Negative voltage causes the current produced when the bottom switch is "ON" to drop. In this case, the reference currents are surrounded by a hysteresis band. In this way, the inverter's generated currents are controlled to remain inside a given reference value's hysteresis band.

2.3. Control Of DC-Link Voltage

The dc link voltage shows a dynamic behavior with the load change in the system. It undergoes corresponding positive or negative changes according to the load addition or load demounting. A PI controller regulates the dc link voltage at a desired value. In order to maintain the dc link voltage at desired value, the capacitor draws or injects an additional small amount of current ($i_{s\alpha}$) from the electrical network apart from the compensating current (i_c) component. The dynamic behavior of the dc link voltage is first analyzed in this section.

From the power balance equation,

$$P_{dc} = c_{dc} v_{dc} \frac{dv_{dc}}{dt} = v_{dc} i_{dc} \quad (1)$$

P_{dc} is the power that needed to maintain the dc link voltage.

$$P_{dc} = \sum_{i=a,b,c} v_{si} i_{sai} - \sum_{i=a,b,c} R_f (i_{sai}^2 + i_{ci}^2) - \frac{1}{2} \sum_{i=a,b,c} L_f \frac{d}{dt} (i_{sai}^2 + i_{ci}^2) \quad (2)$$

Where v_s is the source voltage, R_f and L_f are the resistance and inductance of the inductor that is connected in between PCC and APF.

For a balance three phase system, above equation becomes,

$$P_{dc} = 3v_s i_{s\alpha} - 3R_f (i_{s\alpha}^2 + i_c^2) - \frac{3}{2} L_f \frac{d}{dt} (i_{s\alpha}^2 + i_c^2) \quad (3)$$

By applying small changes to i_c , $i_{s\alpha}$, v_{dc} and v_s , the following equations can be obtained,

$$i_c = I_c + \Delta i_c \quad (4)$$

$$i_{s\alpha} = I_{s\alpha} + \Delta i_{s\alpha} \quad (5)$$

$$v_{dc} = V_{dc} + \Delta v_{dc} \quad (6)$$

$$v_s = V_s + \Delta v_s \quad (7)$$

In the steady state,

$$3V_s I_{s\alpha} - 3R_f(I_{s\alpha}^2 + I_c^2) = 0 \quad (8)$$

By substituting above equations (4), (5), (6), (7) and (8) in equation (3),

$$C_{dc} V_{dc} \frac{d\Delta v_{dc}}{dt} = 3(\Delta v_s I_{s\alpha} + V_s \Delta i_{s\alpha}) - 6R_f(I_{s\alpha} \Delta i_{s\alpha} + I_c \Delta i_c) - 3L_f \left(I_{s\alpha} \frac{d\Delta i_{s\alpha}}{dt} + I_c \frac{d\Delta i_c}{dt} \right) \quad (9)$$

By performing Laplace transformation and rearranging the equation (9),

$$\Delta V_{dc}(s) = \frac{K G_s(s) G_1(s) G_2(s)}{1 + K G_s(s) G_1(s) G_2(s)} \Delta V_{dc}^*(s) - \frac{G_2(s) G_3(s)}{1 + K G_s(s) G_1(s) G_2(s)} \Delta I_c(s) + \frac{G_2(s) G_4(s)}{1 + K G_s(s) G_2(s) G_4(s)} \Delta V_s(s) \quad (10)$$

In this paper, change in source voltage and dc link voltage is not considered. Therefore, the equation (10) is simplified into below equation.

$$\Delta V_{dc}(s) = - \frac{G_2(s) G_3(s)}{1 + K G_s(s) G_1(s) G_2(s)} \Delta I_c(s) \quad (11)$$

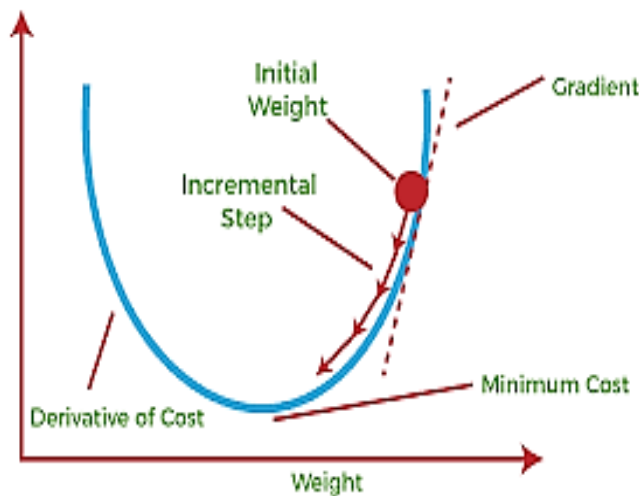
This implies that the compensating current depends on the change in dc link voltage. Therefore, the estimation of compensating current is possible from the change in dc link voltage.

2.4. ANN Controller

The Artificial Neural Network (ANN) controller is a powerful and flexible control algorithm that utilizes the principles of deep learning to make accurate and adaptive control decisions. Unlike traditional control algorithms, which rely on predefined rules and mathematical models, the ANN controller learns directly from data. It consists of multiple layers of interconnected neurons, each performing simple computations and passing the information forward. The network learns by adjusting the connection weights between neurons through a process called training, using a large dataset of input-output pairs.

$$\frac{\partial}{\partial \mathbf{m}} = \frac{2}{N} \sum_{i=1}^N -x_i (y_i - (mx_i + b))$$
$$\frac{\partial}{\partial \mathbf{b}} = \frac{2}{N} \sum_{i=1}^N -(y_i - (mx_i + b))$$

This allows the ANN controller to capture complex relationships and patterns in the data, making it suitable for applications where traditional control approaches struggle to handle nonlinear dynamics or uncertain environments. The trained ANN controller can then be deployed in real-time applications, providing robust and adaptive control performance. By leveraging the power of deep learning, the ANN controller offers a promising approach for achieving high-performance control in various domains, including process control, robotics, and automation systems.



2.5. Research Gap

Although numerous research has been carried out on this domain in order to reduce harmonics, we found a comprehensive approach using the ANN predictive controller based SAPF. We achieve more accurate and fast harmonic compensation compared to conventional Synchronous Reference Frame (SRF) method. This adaptability ensures effective harmonics reduction even in scenarios with varying load characteristics or harmonic profiles, leading to improved overall system performance.

3. Design and Simulation

Design and simulation are an important part in this project. It includes the designing SAPF, PI controller, ANN controller models using MATLAB Simulink and ANN algorithm using Python.

3.1 SRF based SAPF

A MATLAB SIMULINK model is created using instantaneous power theory to simulate SAPF. For the sake of simulation, an IGBT/Diode that is supplying a resistive load is treated as a non-linear load. The simulation is run while keeping the following parameters in mind.

Table 1 - System Parameters

System Parameters	Values
System Frequency	50 Hz
Source Voltage (Phase to Phase) RMS	400 V
Rs, Ls	0.893 ohm, 5.8 mH
DC link capacitance	1 mF
DC link voltage	650 V
Filter inductance	1 mH
Load (R, L)	50 ohm, 20 mH

3.1.1. Case 1 - Non-linear load without Compensator

Consider a three-phase source with a connected non-linear load. The MATLAB model is shown below.

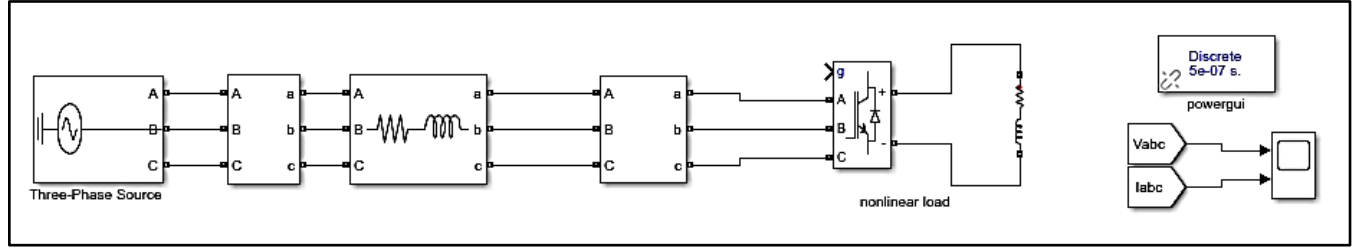


Figure 4 - MATLAB Model of Non-linear load without Compensator

3.1.2. P, Q Components Calculation

First, we measure the load current and source voltage when the non-linear load is connected. By using these values, the Clarke transformation can be done.

The Clarkes transformation matrix is given by,

$$\begin{bmatrix} v_{\alpha} \\ v_{\beta} \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix}$$

$$\begin{bmatrix} i_{\alpha} \\ i_{\beta} \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} i_{La} \\ i_{Lb} \\ i_{Lc} \end{bmatrix}$$

To calculate P and Q components, we use formulas which are shown below.

$$P_{3\phi} = V_{\alpha} i_{\alpha} + V_{\beta} i_{\beta}$$

$$Q_{3\phi} = V_{\beta} i_{\alpha} - V_{\alpha} i_{\beta}$$

The MATLAB model of P, Q components calculation is shown here.

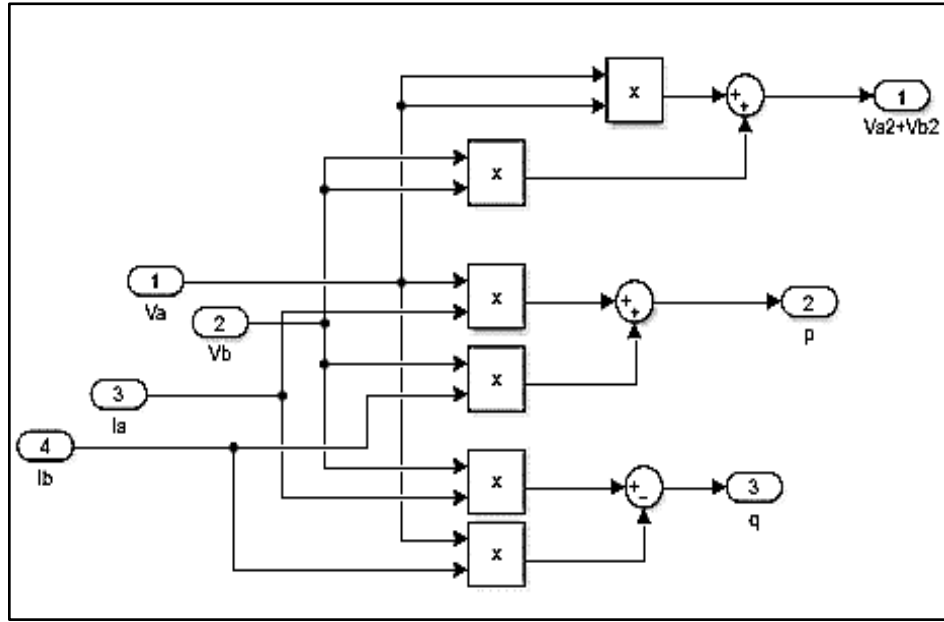


Figure 5 - P, Q Components Calculation

3.1.3. Compensated Reference Current Calculation

Using calculated P, Q components we can calculate i_α , i_β . For this, we use formulas which are shown below.

$$\begin{bmatrix} i_\alpha & i_\beta \end{bmatrix} = \frac{1}{v_\alpha^2 + v_\beta^2} \begin{bmatrix} v_\alpha & v_\beta & v_\beta & -v_\alpha \end{bmatrix} \begin{bmatrix} p & q \end{bmatrix}$$

After knowing them inverse Clarke transformation is performed and the compensating currents are calculated i_{ca} , i_{cb} , i_{cc} .

$$\begin{bmatrix} i_{ca} & i_{cb} & i_{cc} \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & 0 & -\frac{1}{2} & \frac{\sqrt{3}}{2} & \frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} i_\alpha & i_\beta \end{bmatrix}$$

The compensated reference current calculation in MATLAB is shown below.

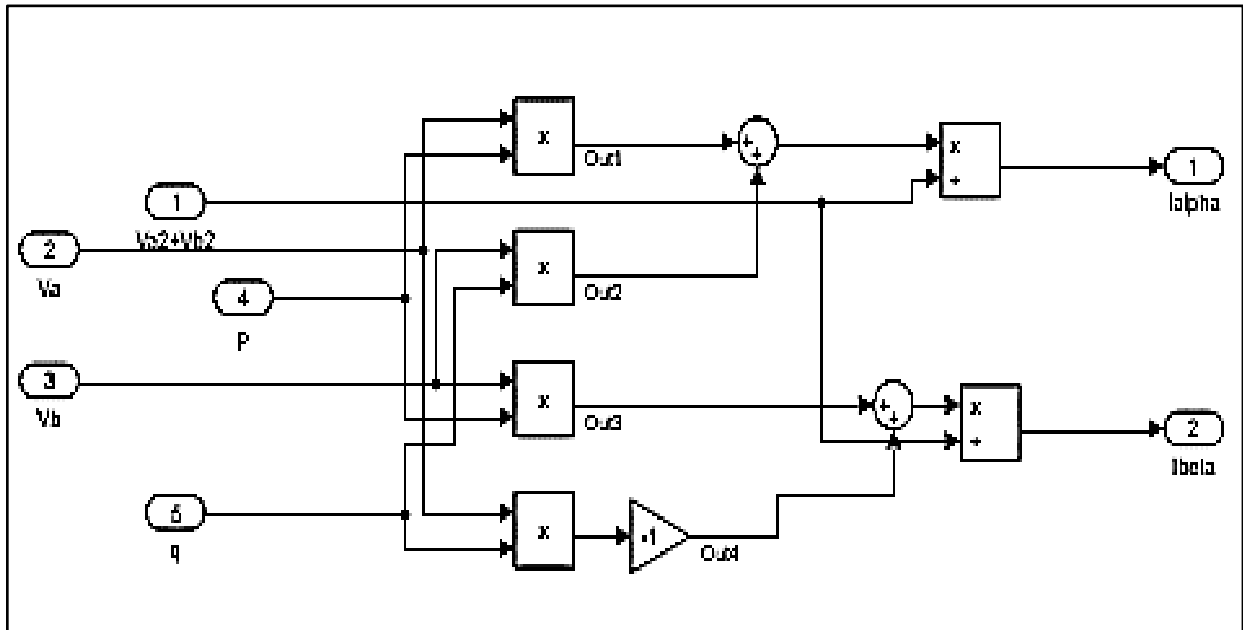


Figure 6 - Compensated Reference Current Calculation

3.1.4. Hysteresis Band Current Controller

This current controller decides the switching pattern in SAPF. The switching logic is formulated as,

- If $I_{\text{measured}} > I_{\text{reference}}$ then upper switch is “OFF” and bottom switch is “ON” in the inverter leg.
- If $I_{\text{measured}} < I_{\text{reference}}$ then upper switch is “ON” and bottom switch is “OFF” in the inverter leg.

The implemented Hysteresis Band Current Controller using MATLAB is shown below.

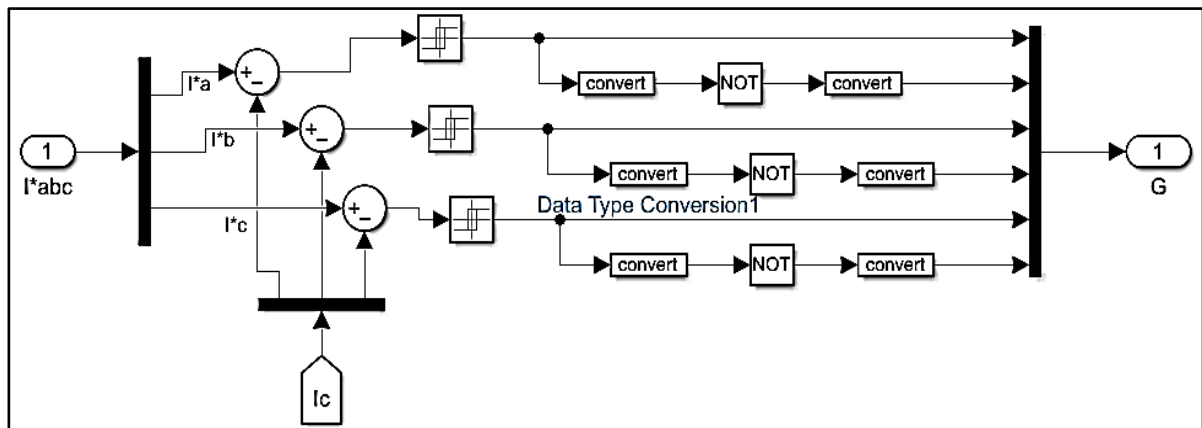


Figure 7 - Hysteresis Band Current Controller

3.1.5. Voltage Source Inverter

Here, three phase current source inverter is used as shunt active power filter.

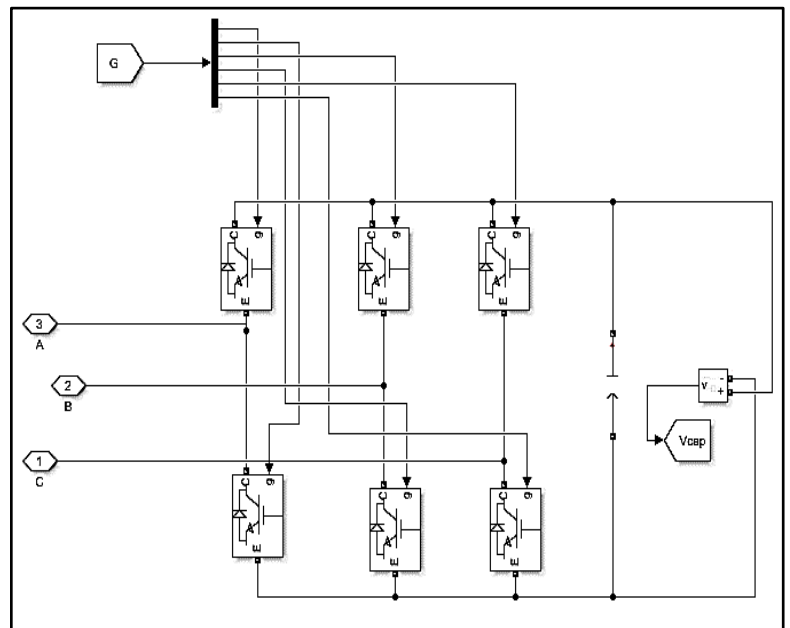


Figure 8 - Voltage Source Inverter

3.1.6. Case 2 - System Mathematical Model

The block diagram of the SRF based shunt active power filter is shown below.

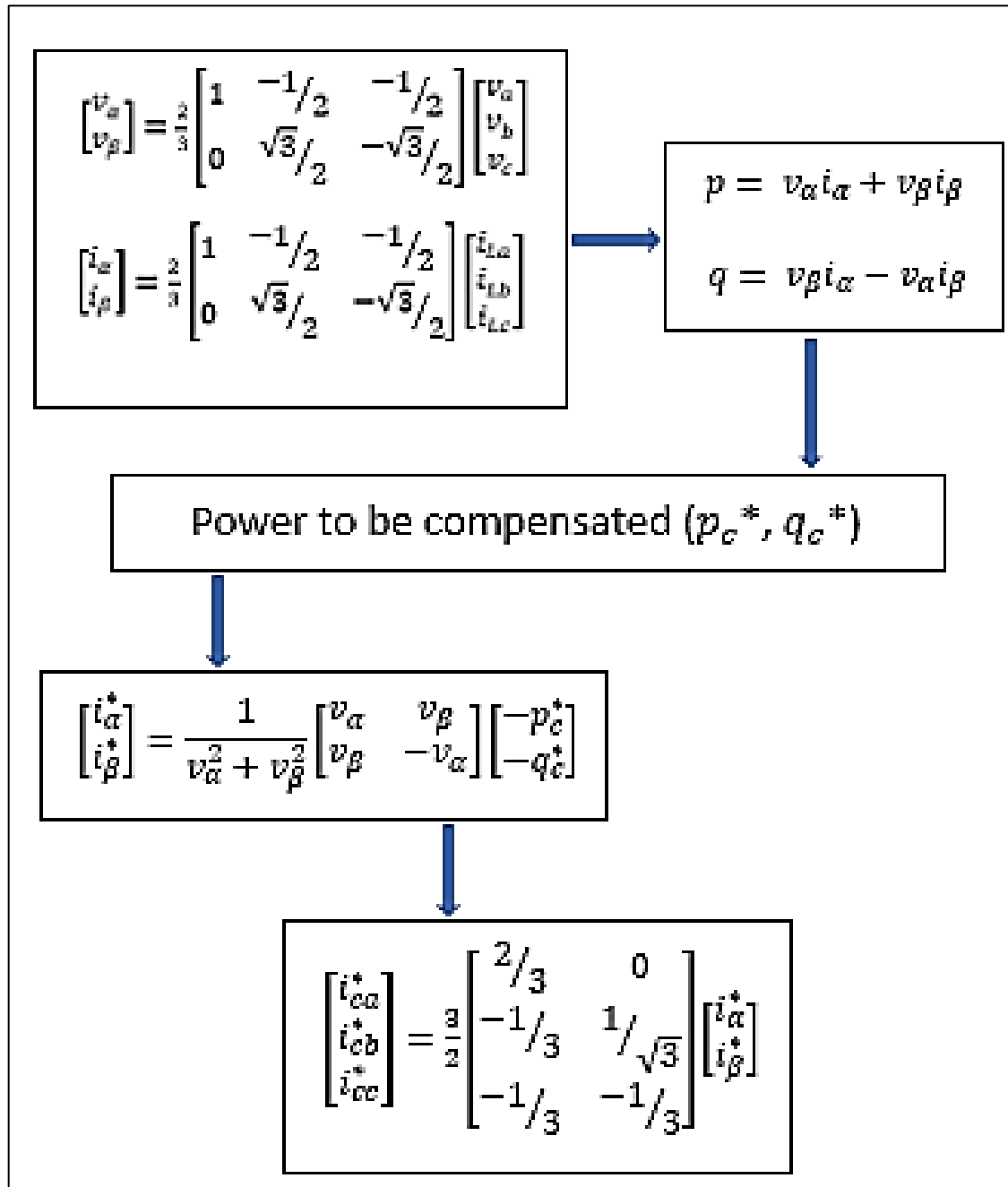


Figure 9 - Block diagram of the SRF based SAPF

Shunt APF has been modeled in MATLAB/Simulink according to the above criterion.

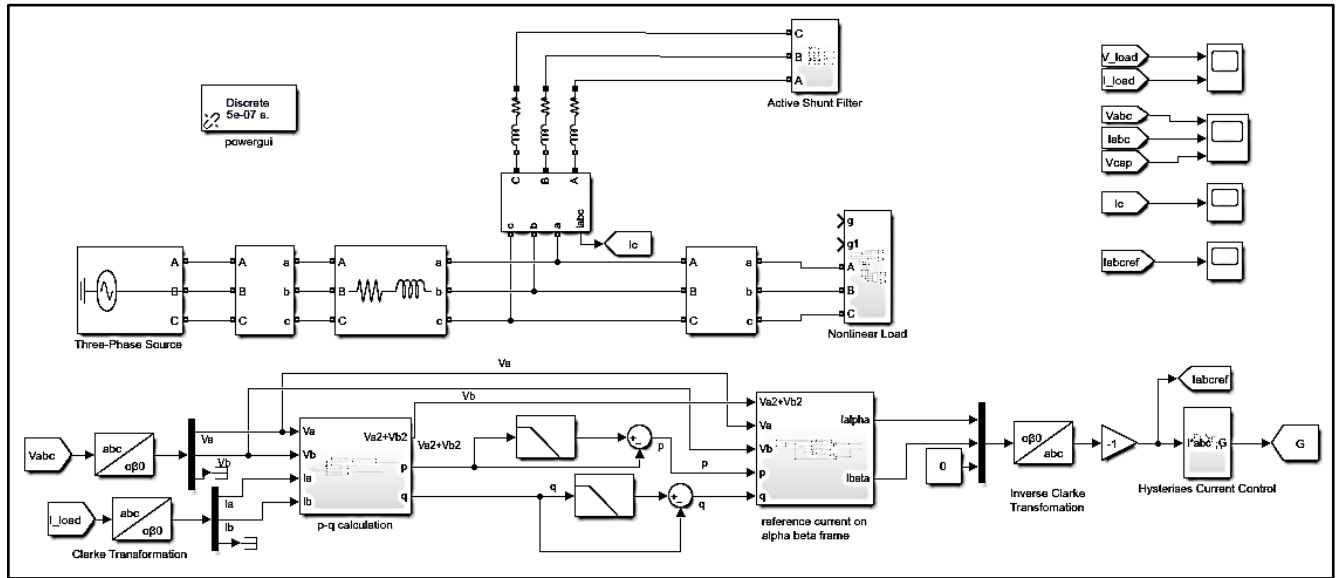


Figure 10 - MATLAB Model of SRF based SAPF

3.2 Case 3 - PI Controller based SAPF

This formula below implies that the compensating current depends on the change in dc link voltage. Therefore, the estimation of compensating current is possible from the change in dc link voltage.

$$P_{dc} = c_{dc} v_{dc} \frac{dv_{dc}}{dt} = v_{dc} i_{dc}$$

In order to regulate the DC link voltage, we use a PI controller. According to trial-and-error method we find PI parameters are,

$$K_p = 10$$

$$K_I = 20$$

The SAPF with PI controller is modeled in MATLAB is shown below.

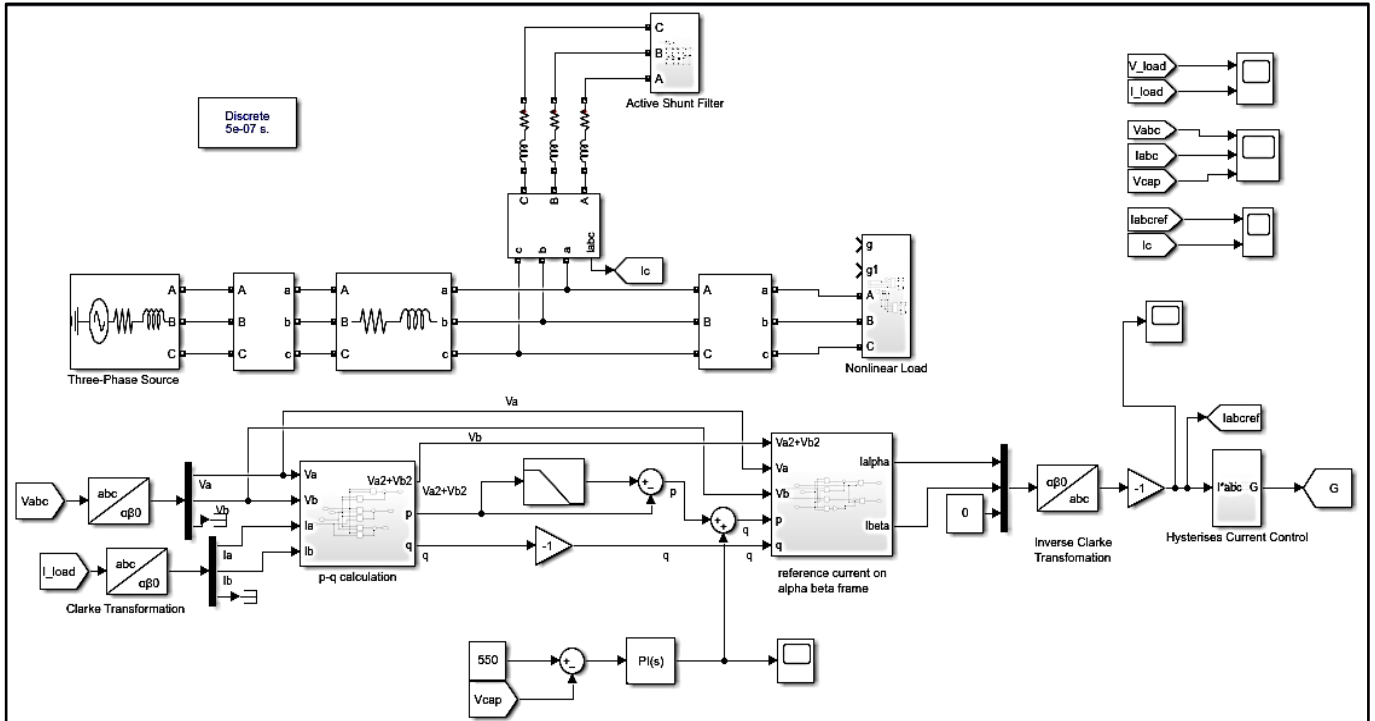


Figure 11 - MATLAB Model of PI Controller based SAPF

3.3 Case 4 - DNN Controller based SAPF

DNN-based PI controller is presented in order to achieve better performance of shunt APF. PI controller has been replaced with a neural network. The neural network has been trained with error input of the PI controller and the target output of the PI controller. The figure which is shown below is the implemented neural network for PI controller action.

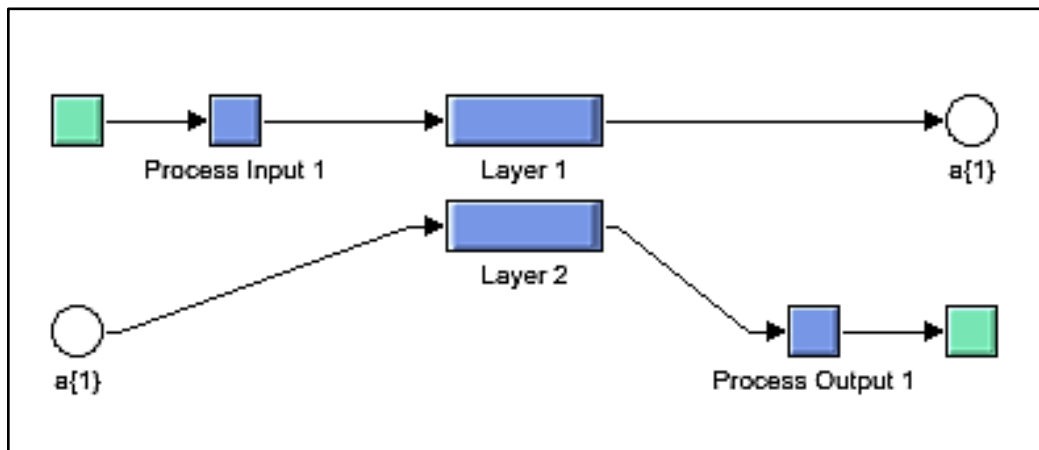


Figure 12 - Neural network implementation for PI controller

3.4 Case 5 - ANN Predictive Controller based SAPF

In this project, the goal is to develop a regression model to predict the output variable PI controller output (PIout) based on the input variable Vdc error. The dataset used for this task consists of error measurements (Vdc) and corresponding PIout values. The objective is to train a neural network model that can accurately predict PIout given Vdc error values. By successfully developing such a model, it would be possible to gain insights into the relationship between these variables and potentially make accurate predictions in real-world scenarios.

The data was obtained from MATLAB files and loaded into Python using the scipy.io library. The error measurements (Vdc) and PIout values were combined into a pandas DataFrame, allowing for easier manipulation and analysis. To ensure that the input data falls within a similar range, the MinMaxScaler from the sklearn.preprocessing module was applied to normalize the data. This scaling technique transforms the data to a range of [0, 1], which is beneficial for neural network training.

PYTHON Code

```
import scipy.io as sio
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, InputLayer, Dropout
import matplotlib.pyplot as plt
from tensorflow.keras.metrics import MeanSquaredError
from sklearn.metrics import r2_score
%matplotlib inline

[ ] from google.colab import drive
    drive.mount('/content/drive')

Mounted at /content/drive

[ ] Vdc = sio.loadmat("/content/drive/MyDrive/FYP code/error.mat")
```

3.4.1. Model Architecture

The DNN model architecture used for predicting the PIout values consists of multiple layers, each with a specific number of neurons and activation functions. The model architecture is designed to capture the complex relationships between the input variable (V_{dc}) and the target variable (PIout), enabling accurate predictions.

The input layer of the DNN model is a Flatten layer, which transforms the input data into a one-dimensional array to be processed by the subsequent layers. The input layer does not have any neurons as it only serves as a conduit to pass the input data to the next layer. Following the input layer, there are three dense (fully connected) layers in the DNN model. The first dense layer has 128 neurons and uses the Rectified Linear Unit (ReLU) activation function. ReLU is a commonly used activation function that introduces non-linearity to the model, allowing it to learn complex patterns and relationships in the data.

The second dense layer consists of 64 neurons and also employs the ReLU activation function. This layer further processes the information extracted from the previous layer, capturing more intricate features and patterns in the input data. The third dense layer contains 32 neurons and utilizes the ReLU activation function as well. This layer performs additional transformations and computations to extract high-level representations and capture finer details in the input data. Finally, the output layer of the DNN model consists of a single neuron, which predicts the PIout value. This neuron provides the final regression output based on the learned representations and computations performed in the previous layers.

It's worth noting that the choice of the number of neurons in each layer and the activation function can significantly impact the model's capacity to learn and generalize. Increasing the number of neurons allows the model to capture more complex relationships in the data, while the activation function introduces non-linearity, enabling the model to approximate complex functions and make accurate predictions. By leveraging this architecture, the DNN model can effectively learn the underlying patterns and dynamics between V_{dc} error and PIout, enabling it to predict PIout values accurately for unseen input data.

PYTHON Code

```
[ ] model = tf.keras.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128,activation = "relu"),
    tf.keras.layers.Dense(64,activation = "relu"),
    tf.keras.layers.Dense(32,activation = "relu"),
    tf.keras.layers.Dense(1)
])

[ ] model.compile(
    optimizer = "adam",
    loss = "mse",
    #metrics = [MeanSquaredError()]
)
```

3.4.2. Model Training

The model was trained using the Adam optimizer, which is an adaptive learning rate optimization algorithm. The mean squared error (MSE) loss function was chosen to measure the difference between the predicted and actual PIout values. The model was trained for 10 epochs with a batch size of 32. During training, the model's performance was evaluated using a validation set consisting of a portion of the data that was not used for training. The loss function values were monitored to track the convergence and generalization of the model.

PYTHON Code

```
[ ] history = model.fit(x_train,y_train,batch_size = 32,epochs = 10,validation_data = (x_test,y_test))

Epoch 1/10
60094/60094 [=====] - 91s 1ms/step - loss: 3.3147e-04 - val_loss: 1.6350e-04
Epoch 2/10
60094/60094 [=====] - 89s 1ms/step - loss: 1.4200e-04 - val_loss: 1.3914e-04
Epoch 3/10
60094/60094 [=====] - 78s 1ms/step - loss: 1.3288e-04 - val_loss: 1.2792e-04
Epoch 4/10
60094/60094 [=====] - 80s 1ms/step - loss: 1.2970e-04 - val_loss: 1.1499e-04
Epoch 5/10
60094/60094 [=====] - 82s 1ms/step - loss: 1.2404e-04 - val_loss: 1.1176e-04
Epoch 6/10
60094/60094 [=====] - 90s 1ms/step - loss: 1.2683e-04 - val_loss: 2.2796e-04
Epoch 7/10
60094/60094 [=====] - 80s 1ms/step - loss: 1.5276e-04 - val_loss: 1.2242e-04
Epoch 8/10
60094/60094 [=====] - 91s 2ms/step - loss: 1.2259e-04 - val_loss: 1.1199e-04
Epoch 9/10
60094/60094 [=====] - 81s 1ms/step - loss: 1.2165e-04 - val_loss: 1.1491e-04
Epoch 10/10
60094/60094 [=====] - 90s 1ms/step - loss: 1.1935e-04 - val_loss: 1.1848e-04
```

3.4.3. Model Evaluation

To assess the performance of the trained model, the coefficient of determination (R2 score) was used as an evaluation metric. The R2 score measures the proportion of the variance in the dependent variable (PIout) that can be explained by the independent variable (Vdc). A high R2 score indicates a better fit of the model to the data. The obtained R2 score on the test set provides insights into how well the model can predict PIout based on unseen Vdc values.

PYTHON Code

```
[ ] y_predict = model.predict(x_test)

15024/15024 [=====] - 12s 820us/step

[ ] R2 = r2_score(y_test,y_predict)
R2
0.9793446531959558
```

3.4.4. Results and Analysis

The training and validation loss over epochs were plotted to visualize the learning progress of the model. The decreasing trend in both training and validation loss indicates that the model was effectively learning from the data. The scatter plot comparing the actual and predicted values of PIout illustrates the model's performance visually. A higher concentration of points near the diagonal line indicates a better prediction accuracy. By analyzing these results, it is evident that the developed model successfully captures the underlying relationship between Vdc and PIout.

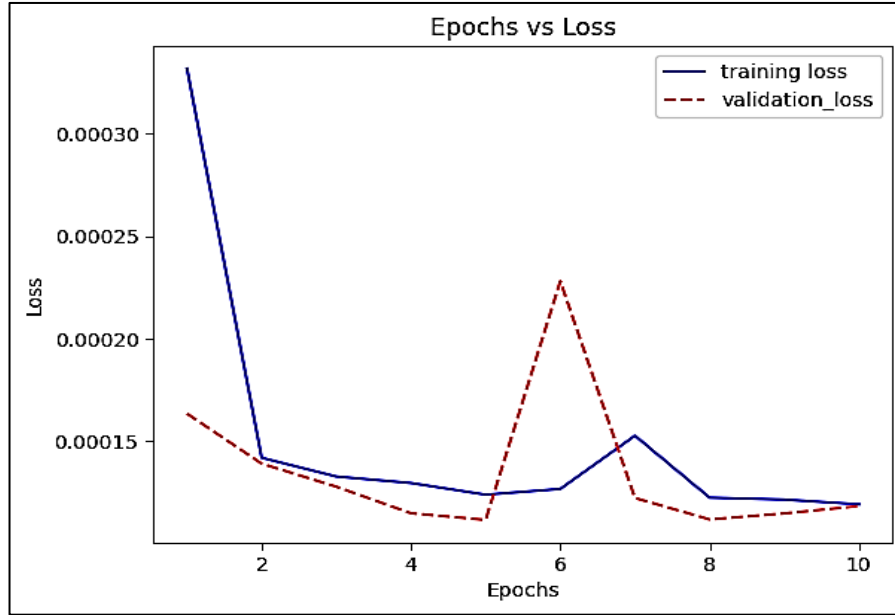


Figure 13 - Training & Validation Loss Curve

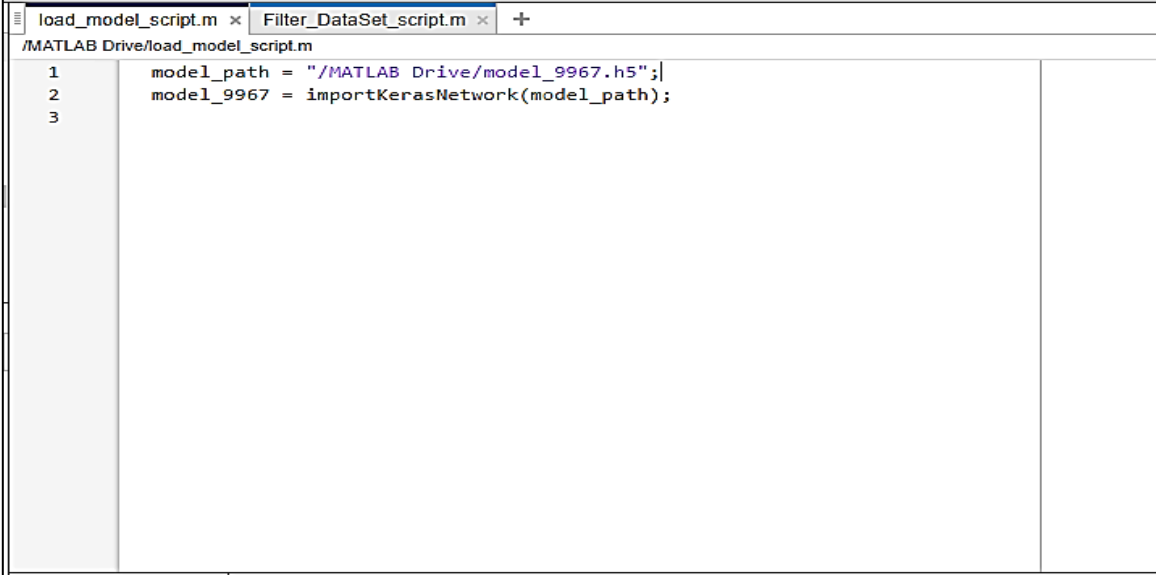
After training the Artificial Neural Network (ANN) controller, the next step was to save the model for future use. This was accomplished by converting the trained model to an .h5 file format. The .h5 file format is a common and widely supported format for storing deep learning models. It contains all the necessary information about the network architecture, model weights, and configuration. By saving the model in this format, it becomes portable and can be easily shared and loaded into different environments.

To load the .h5 file in MATLAB, the KerasNetworkLoader utility was employed. The KerasNetworkLoader is a MATLAB function specifically designed for loading Keras models. It provides a convenient interface to import the saved model into the MATLAB workspace, allowing seamless integration with MATLAB-based workflows. By using the importKerasNetwork, the trained ANN controller was successfully imported into MATLAB, enabling further analysis, evaluation, and deployment of the model in MATLAB-based applications. The ability to convert and load the ANN controller from the .h5 file to MATLAB using the importKerasNetwork provides flexibility and convenience in working with the model in the MATLAB environment. It opens up possibilities for leveraging MATLAB's extensive ecosystem of tools and functions for tasks such as model evaluation, simulation, and integration with other MATLAB-based systems or algorithms.

PYTHON Code

```
[ ] model.save("/content/model_9810.h5")
```

MATLAB Code



The image shows a MATLAB script editor window with two tabs: 'load_model_script.m' and 'Filter_DataSet_script.m'. The active tab is 'load_model_script.m', which contains the following code:

```
1 model_path = "/MATLAB Drive/model_9967.h5";  
2 model_9967 = importKerasNetwork(model_path);  
3
```


4. Results and Discussion

The results are discussed under five cases as below.

4.1 Case-1 Results

When the non-linear load is connected to the source, the source current is distorted as shown below.

The THD of the source current is obtained as 23.95%.

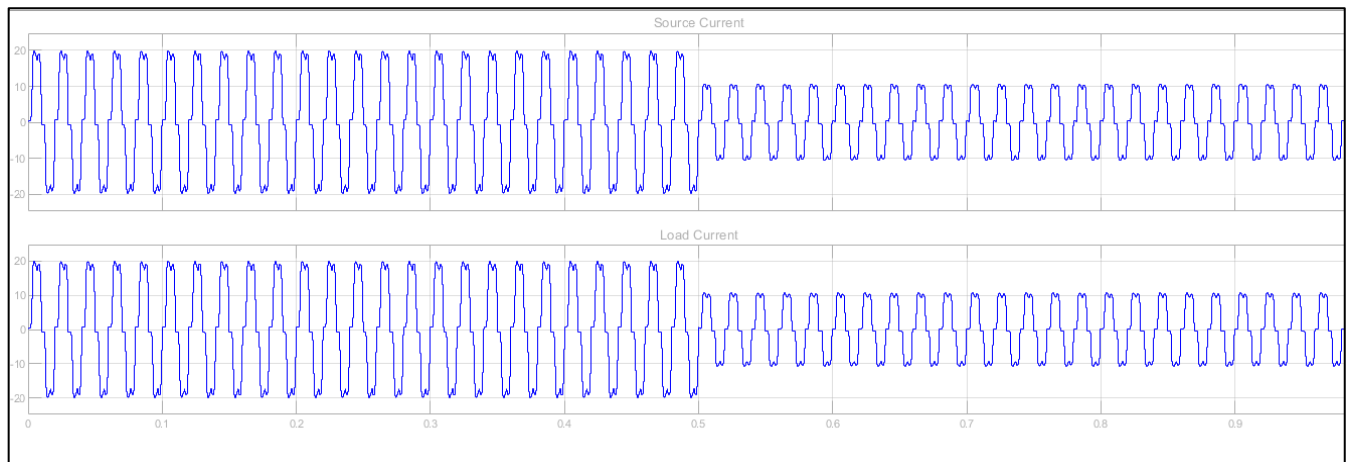


Figure 14 - Source, Load current without compensator

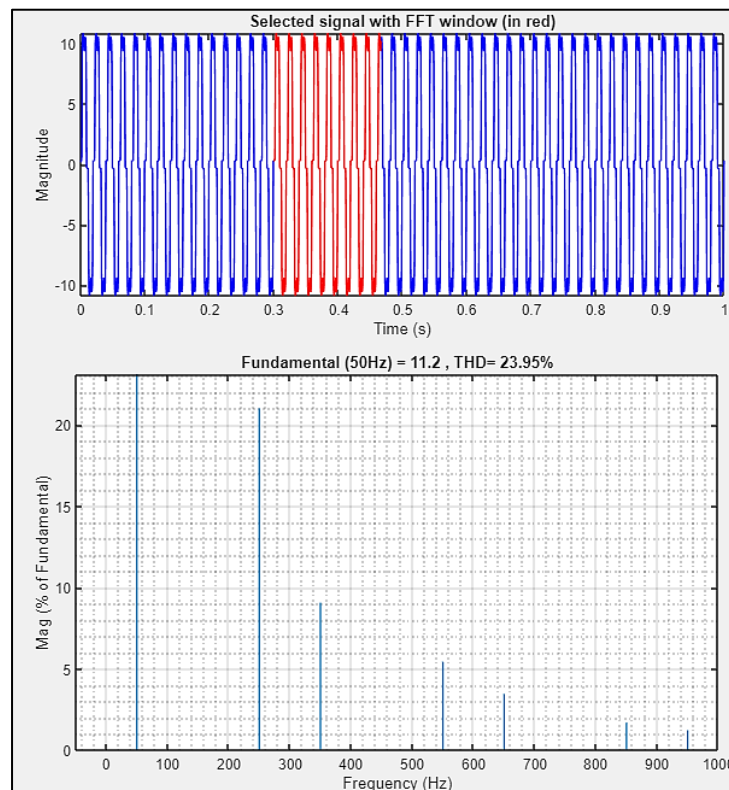


Figure 15 - THD% without compensator

4.2 Case-2 Results

First the simulation is carried out for shunt APF without regulating DC link voltage. Figures below show the simulation results with source, load and compensating current waveforms. A load change happens at 0.5 s. The THD of the source current is obtained as 2.96%. The system takes around 0.1 s to converge when the system is undergoing a load change.

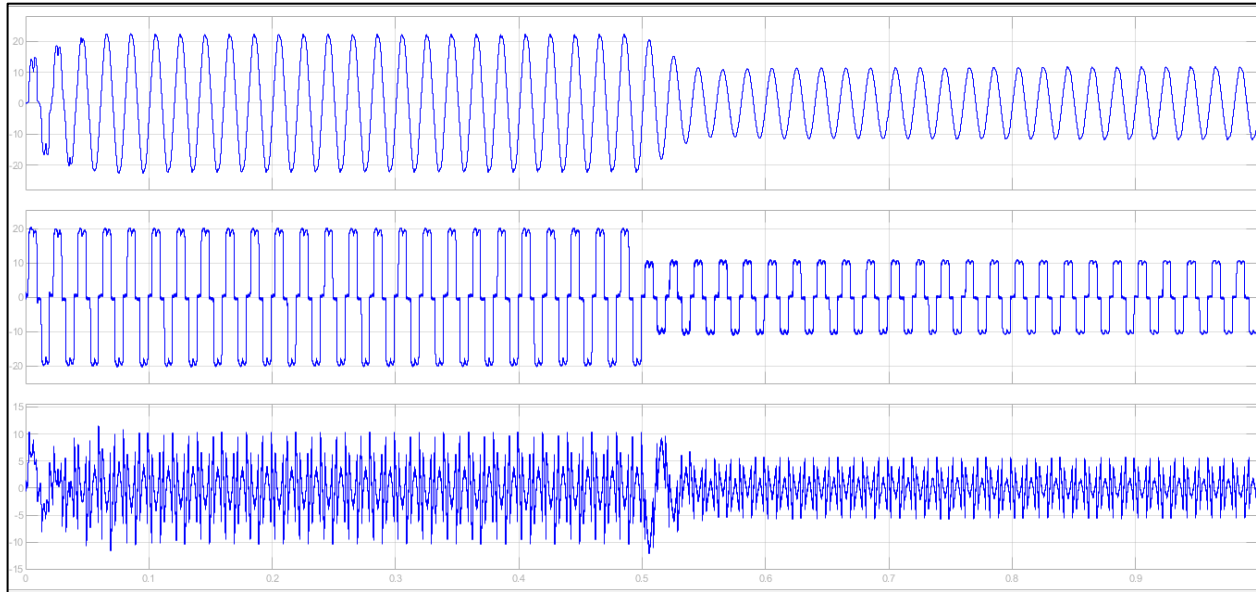


Figure 16 - Source, Load, Compensating Current of SRF based SAPF

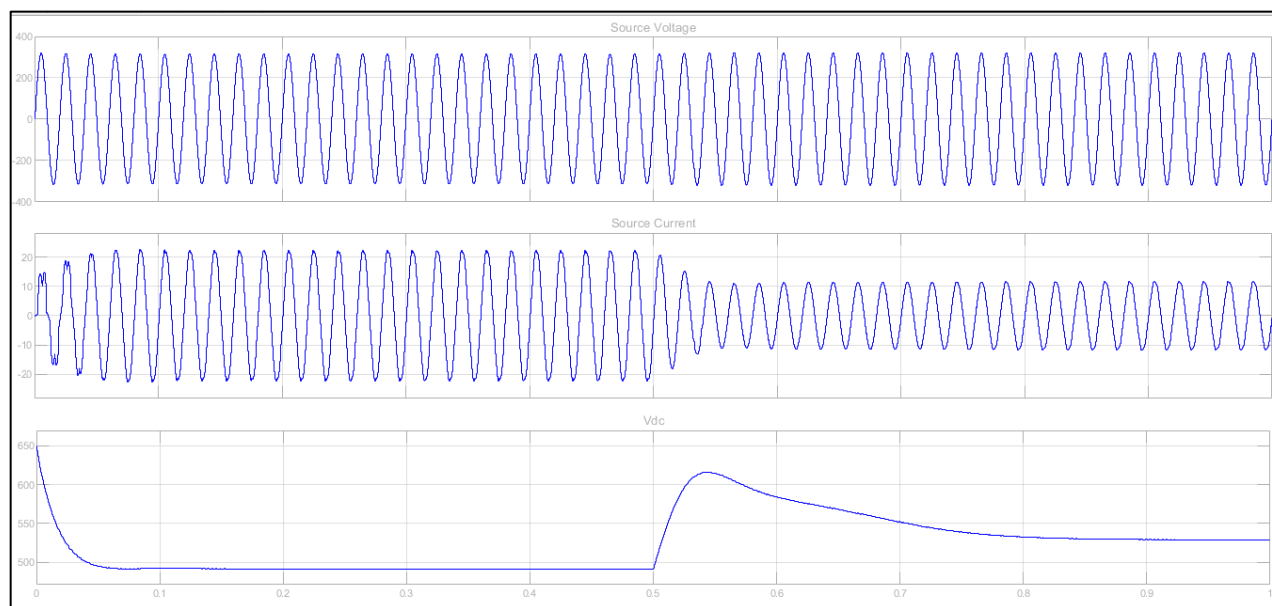


Figure 17 - Source Voltage, Source Current, DC link voltage of SRF based SAPF

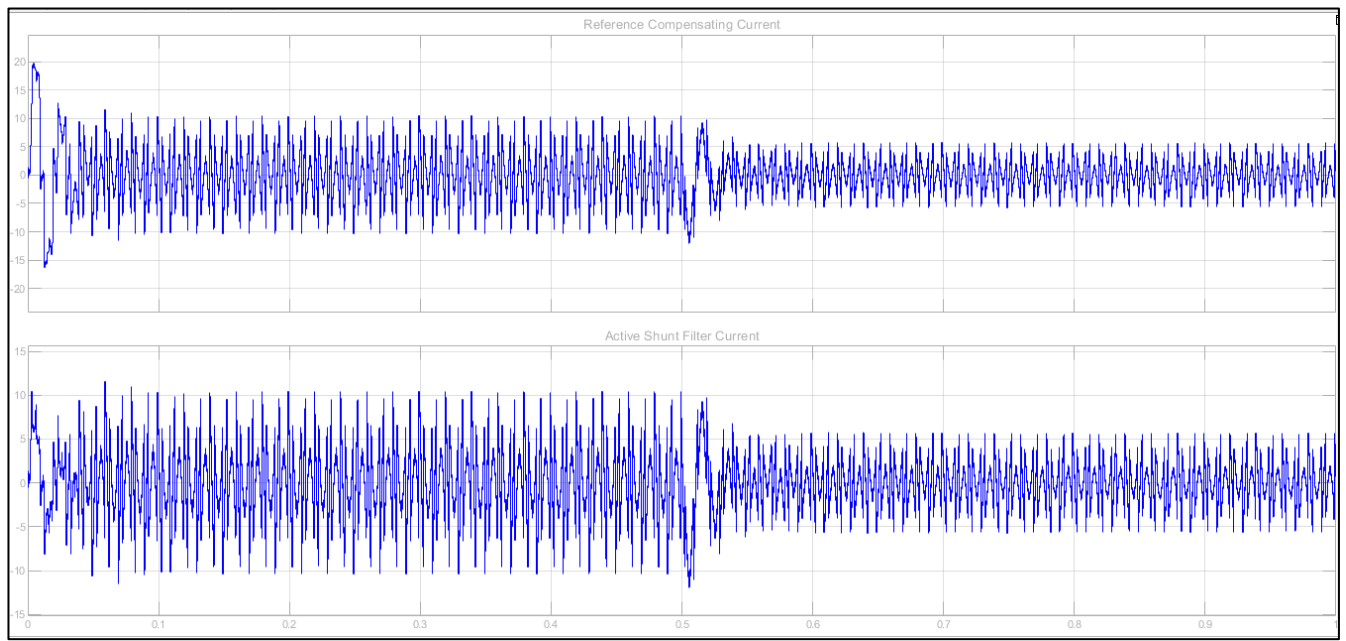


Figure 18 - Reference Compensating, Compensating Current of SRF based SAPF

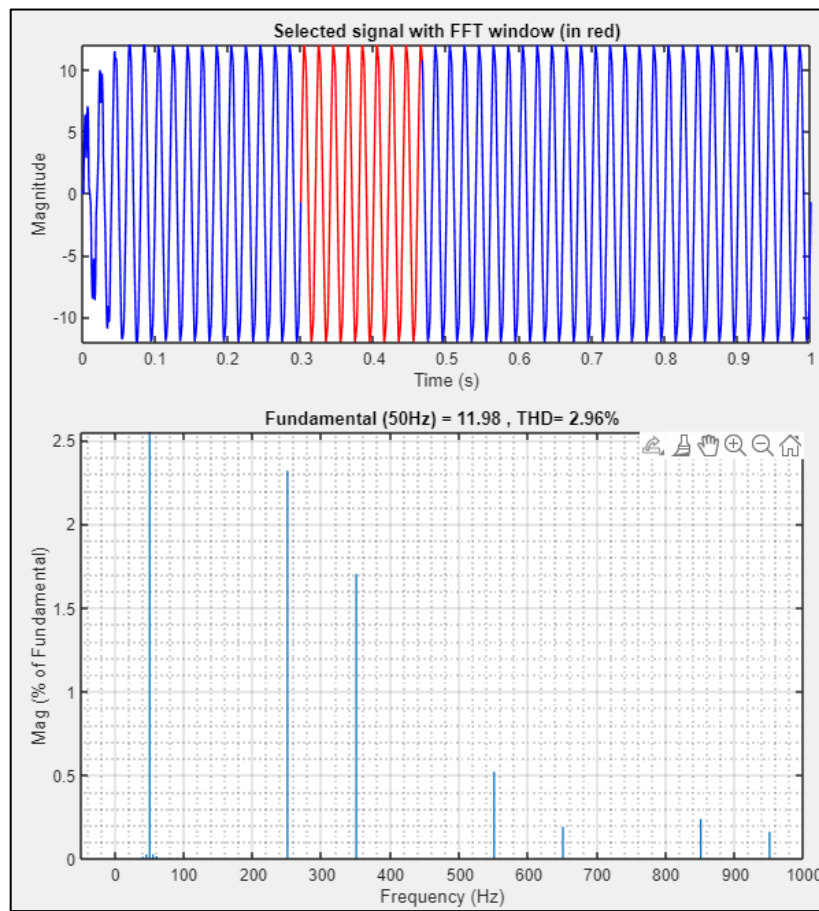


Figure 155 - THD% of SRF based SAPF

4.3 Case-3 Results

Then a modification has been added to the system to regulate the DC link voltage at a desired value and better performance of APF. The PI controller does this voltage regulation. Figures below show the simulation results with source, load and compensating current waveforms. The THD of the source current is 1.40%. Here also the system takes around 0.1 s to converge.

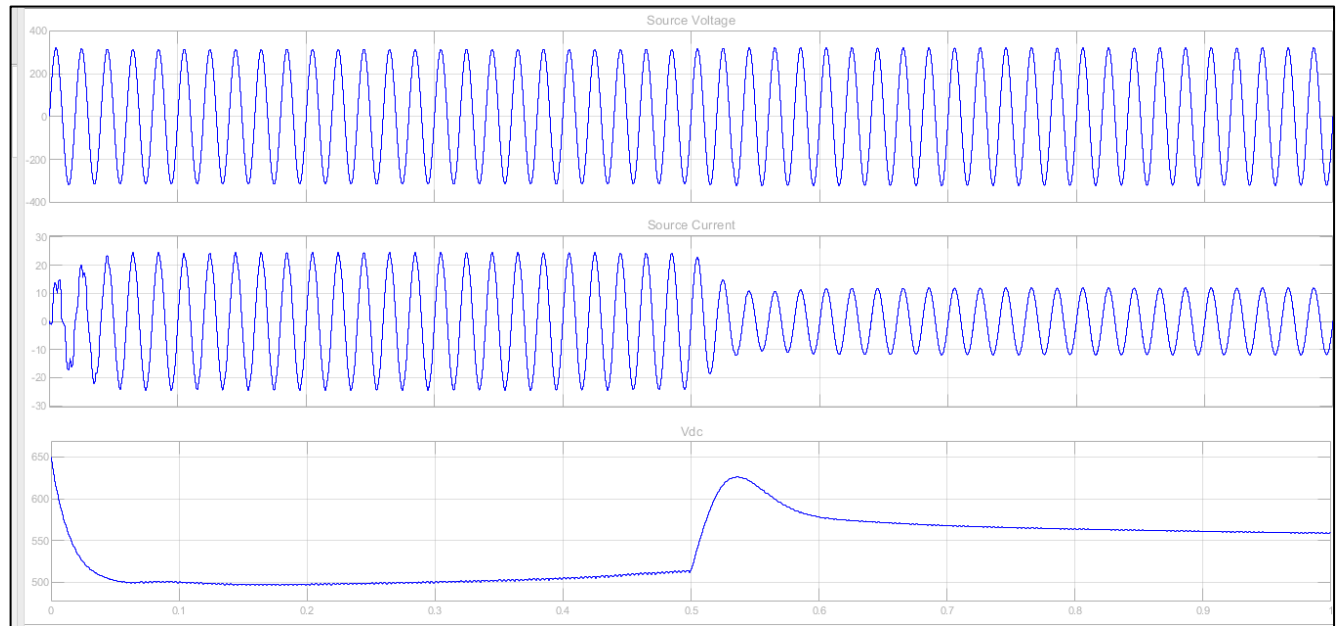


Figure 20 - Source Voltage, Source Current, DC link voltage of PI controller based SAPF

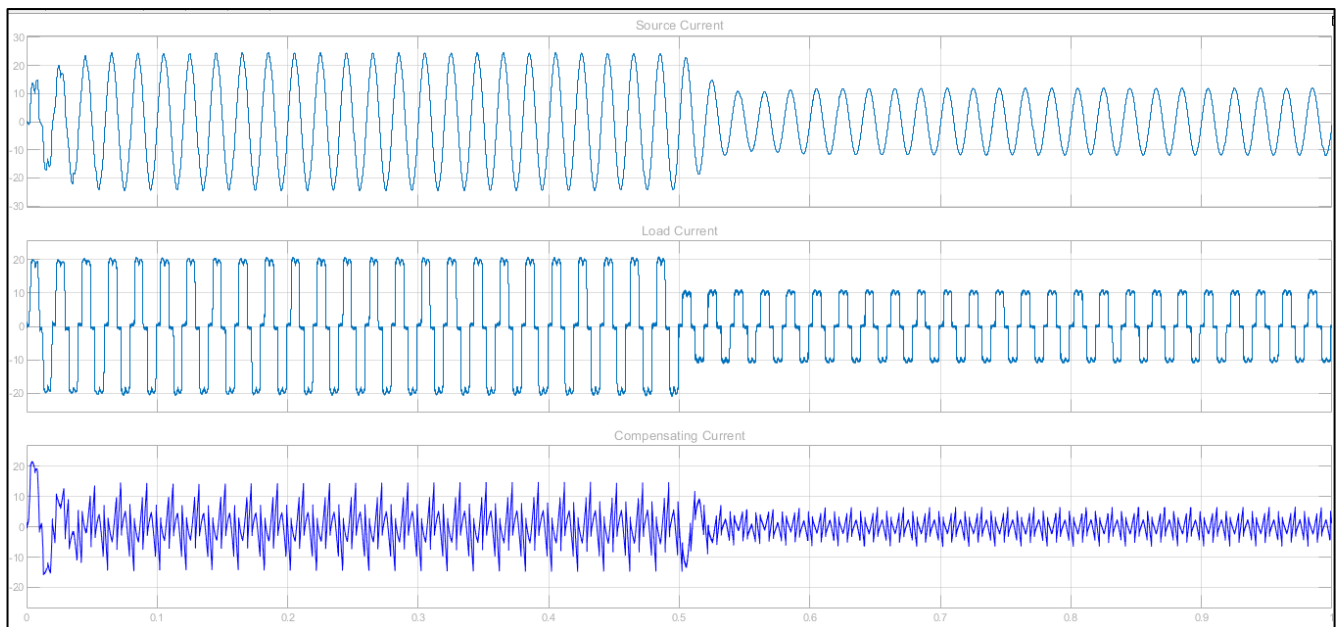


Figure 21 - Source, Load, Compensating Current of PI controller based SAPF

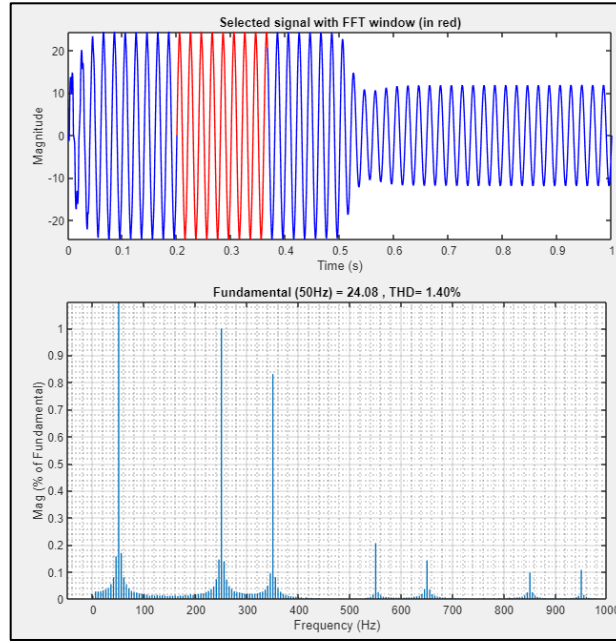


Figure 22 - THD% of PI controller based SAPF

4.4 Case 4 - Results

Next, the neural network-based control on the PI controller is tried. The PI controller is replaced with a deep neural network. Figure below shows corresponding simulation results. The THD of the source current is 0.20%. The system takes around 0.1 s to converge when a load change occurs.

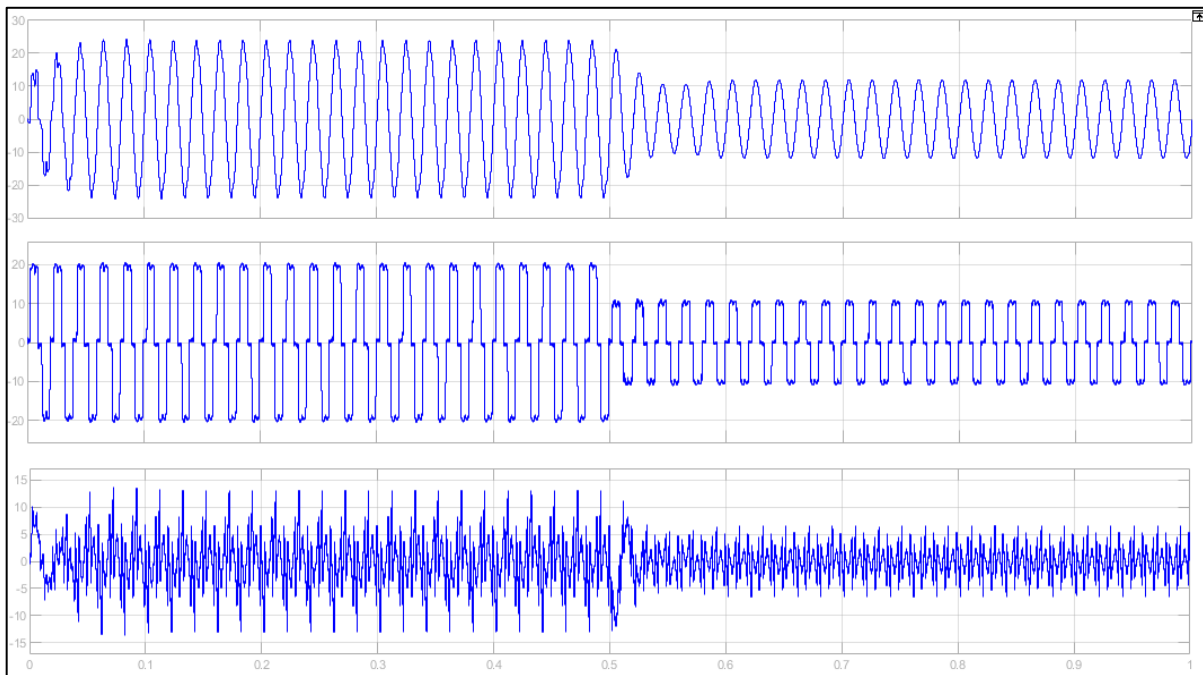


Figure 23 - Source, Load, Compensating Current of DNN controller based SAPF

4.5 Case 5 - Results

The waveform analysis of an APF based on a predictive and adaptive neural network controller has been performed. The figures below consist of simulation results of source current, load current, and compensation current of adaptive-predictive APF. The source current waveform demonstrates an almost sinusoidal behavior. By implementing the predictive and adaptive controller, the Total Harmonic Distortion (THD) is significantly reduced, achieving a reduction of up to 2.79%. The figures provide a visual representation of the effectiveness of the predictive and adaptive controller in achieving the desired waveform and effectively mitigating harmonic distortions showing fast convergence. It takes around 0.05 s to converge after a load change happens. The results presented in this paper highlight the potential of the proposed controller for enhancing power quality in electrical systems.

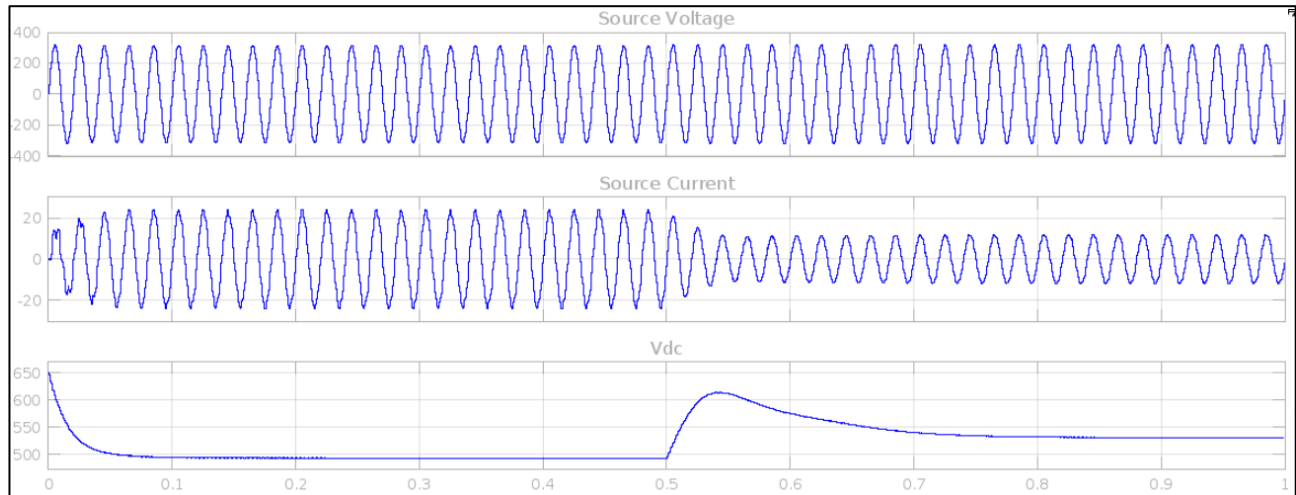


Figure 24 - Source Voltage, Source Current, DC link voltage of Predictive controller based SAPF

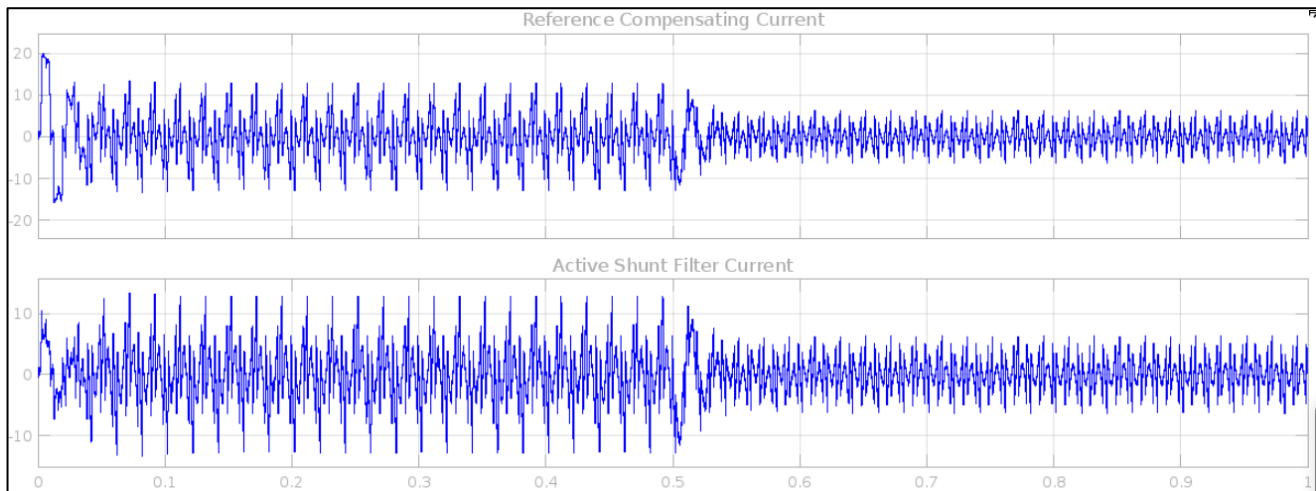


Figure 25 - Reference Compensating, Compensating Current of Predictive controller based SAPF

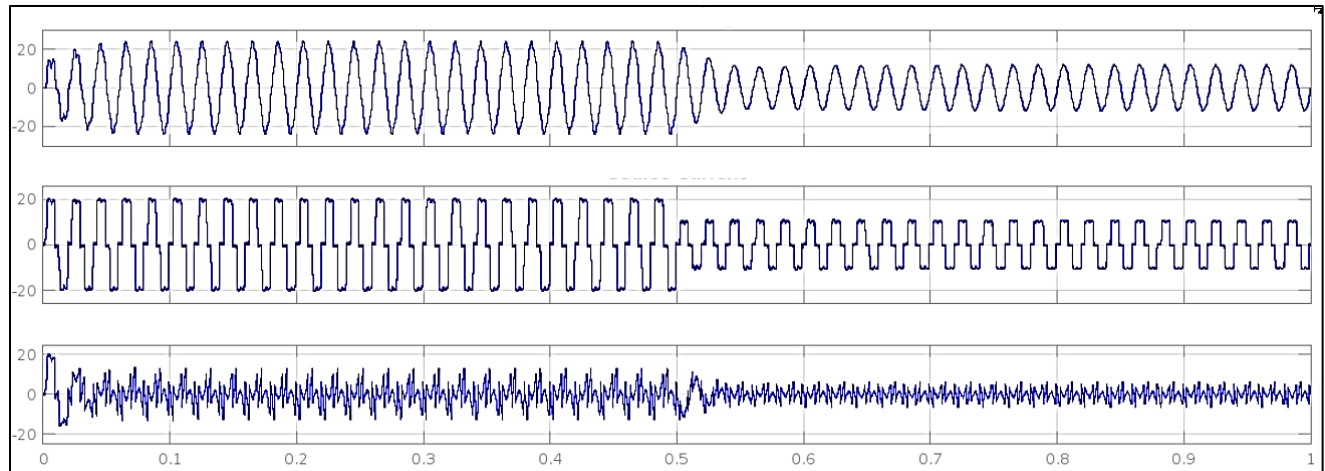


Figure 26 - Source, Load, Compensating Current of Predictive controller based SAPF

4.6 Results Comparison

The table below shows the summarized results of all above cases.

Table 2 - Results Comparison

	THD	Converging time
Without SAPF	23.95%	
SAPF without DC link voltage regulation	2.47%	0.1 s
SAPF with PI controlled DC link voltage regulation	1.40%	0.1 s
SAPF with DNN-based DC link voltage regulation	0.20%	0.1 s
Predictive controller based SAPF	2.79%	0.05 s

5. Conclusion

In conclusion, this project proposed an innovative approach for improving the performance of shunt active power filters (APFs) in power quality enhancement. By leveraging deep neural networks, a significant reduction in Total Harmonic Distortion (THD) was achieved, and the convergence of the harmonic reduction process was accelerated. The utilization of a DNN-based predictive neural network for fast estimation of compensating current and the incorporation of a PI controller for DC link voltage regulation contributed to the superior performance of the proposed model. Simulation results confirmed the effectiveness of the approach in a three-phase balanced system. The findings of this study showcase the potential of DNN-based algorithms in enhancing power quality and protecting sensitive equipment in modern electrical networks.

4. References

- [1] Bhattacharjee, Kakoli. "Design and simulation of synchronous reference frame based shunt active power filter using SIMULINK." (2013): 2-06.
- [2] Patil, Sagar S., and R. A. Metri. "Power quality improvement using shunt active power filter." *2017 International Conference on Data Management, Analytics and Innovation (ICDMAI)*. IEEE, 2017.
- [3] Bojoi, Radu Iustin, et al. "Current control strategy for power conditioners using sinusoidal signal integrators in synchronous reference frame." *IEEE Transactions on Power Electronics* 20.6 (2005): 1402-1412.
- [4] Afonso, João L., MJ Sepúlveda Freitas, and Júlio S. Martins. "pq Theory power components calculations." *2003 IEEE International Symposium on Industrial Electronics (Cat. No. 03TH8692)*. Vol. 1. IEEE, 2003.
- [5] Watanabe, Edson Hirokazu, et al. "Instantaneous p–q power theory for control of compensators in micro-grids." *2010 International School on Nonsinusoidal Currents and Compensation*. IEEE, 2010.
- [6] Kumaresan, S., and H. Habeebullah Sait. "Design and control of shunt active power filter for power quality improvement of utility powered brushless DC motor drives." *Automatika* 61.3 (2020): 507-521.
- [7] Bhattacharya, Avik, and Chandan Chakraborty. "A shunt active power filter with enhanced performance using ANN-based predictive and adaptive controllers." *IEEE transactions on industrial electronics* 58.2 (2010): 421-428.
- [8] Choudhary, Jayanti, et al. "Artificial intelligence based control of a shunt active power filter." *Procedia Computer Science* 92 (2016): 273-281.
- [9] Alghamdi, Thamer AH, et al. "An artificial neural network based harmonic distortions estimator for grid-connected power converter-based applications." *Ain Shams Engineering Journal* 14.4 (2023): 101916.
- [10] P. K. Ray, "Power quality improvement using VLLMS based adaptive shunt active filter," in *CPSS Transactions on Power Electronics and Applications*, vol. 3, no. 2, pp. 154-162, June 2018, doi: 10.24295/CPSSTPEA.2018.00015.

- [11] M. Iqbal et al., "Neural Networks Based Shunt Hybrid Active Power Filter for Harmonic Elimination," in *IEEE Access*, vol. 9, pp. 69913-69925, 2021, doi: 10.1109/ACCESS.2021.3077065.
- [12] D. R. Reddy, V. Aravindh, M. Kondalu, C. R. Reddy, O. C. Sekhar and A. Pandian, "Power Quality Improvement of Integrated PV System with PID Controller Based Shunt Active Filter," 2022 IEEE North Karnataka Subsection Flagship International Conference (NKCon), Vijaypur, India, 2022, pp. 1-5, doi: 10.1109/NKCon56289.2022.10126567.
- [13] M. Iqbal et al., "Neural Networks Based Shunt Hybrid Active Power Filter for Harmonic Elimination," in *IEEE Access*, vol. 9, pp. 69913-69925, 2021, doi: 10.1109/ACCESS.2021.3077065.
- [14] M. Y. Lada, S. S. Mohamad, J. A. M. Gani, M. R. M. Nawawi and G. C. Kim, "Reduction of harmonic using single phase shunt active power filter based on instantaneous power theory for cascaded multilevel inverter," 2016 IEEE International Conference on Power and Energy (PECon), Melaka, Malaysia, 2016, pp. 702-706, doi: 10.1109/PECON.2016.7951650.
- [15] A. Panchbhai, S. Parmar and N. Prajapati, "Shunt active filter for harmonic and reactive power compensation using p-q theory," 2017 International Conference on Power and Embedded Drive Control (ICPEDC), Chennai, India, 2017, pp. 260-264, doi: 10.1109/ICPEDC.2017.8081097.
- [16] S. Srivastava, Y. Shah, B. Shah, P. Salvi and R. M. Patel, "Implementation and simulation of single phase active shunt power filter," 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), Delhi, India, 2016, pp. 1-4, doi: 10.1109/ICPEICES.2016.7853582.
- [17] N. M. Chamat, V. S. Bhandare, S. P. Diwan and S. Jamadade, "Instantaneous reactive power theory for real time control of three-phase shunt Active Power Filter (SAPF)," 2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014], Nagercoil, India, 2014, pp. 792-796, doi: 10.1109/ICCPCT.2014.7054981.

Appendix I – DNN Predictive Controller Python Code

```
import scipy.io as sio
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler,StandardScaler
import tensorflow as tf
from tensorflow.keras import keras
from tensorflow.keras.layers import Dense,InputLayer,Dropout
import matplotlib.pyplot as plt
from tensorflow.keras.metrics import MeanSquaredError
from sklearn.metrics import r2_score
%matplotlib inline
```

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
Vdc = sio.loadmat("/content/drive/MyDrive/FYP code/error.mat")
```

```
Vdc["error_new"]
```

```
array([[ -100.          ],
       [ -99.9995496],
       [ -99.99987533],
       ...,
       [   2.38196788],
       [   2.38109307],
       [   2.37924516]])
```

```
PI_out = sio.loadmat("/content/drive/MyDrive/FYP code/PIout_with_error.mat")
```

```
PI_out["PIout_new"]
```

```
array([[ -1000.          ],
       [-1000.00054957],
       [-1000.00075333],
       ...,
       [  152.85049884],
       [  152.84177447],
       [  152.82331922]])
```

```
df = pd.DataFrame({"Vdc":Vdc["error_new"].flatten(),"PIout":PI_out["PIout_new"].flatten()},index = None)
```


```
min_scaler = MinMaxScaler()
std_scaler = StandardScaler()
```

```
#df = pd.DataFrame(std_scaler.fit_transform(df),columns = df.columns,index = None)
df = pd.DataFrame(min_scaler.fit_transform(df),columns = df.columns,index = None)
```

```
df.head()
```

	Vdc	PIout
0	0.000000e+00	6.444320e-07
1	3.813592e-07	1.743068e-07
2	1.055499e-06	0.000000e+00
3	2.291006e-06	3.928826e-07
4	4.019214e-06	1.283579e-06

```
df.describe()
```

	Vdc	PIout 
count	2.403736e+06	2.403736e+06
mean	8.921061e-01	9.637251e-01
std	6.941226e-02	7.537121e-02
min	0.000000e+00	0.000000e+00

```
x_train,x_test,y_train,y_test = train_test_split(df[["Vdc"]].values.reshape(-1,1),df[["PIout"]].values.reshape(-1,1),test_size = 0.2)
```

x_train

```
array([[0.88162341],
       [0.92538379],
       [0.9061477 ],
       ...,
       [0.89705609],
       [0.88444858],
       [0.87705332]])
```

y_train

```
array([[0.98141005],
       [0.96170068],
       [0.97285627],
       ...,
       [0.97588359],
       [0.97806355],
       [0.98349263]])
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128,activation = "relu"),
    tf.keras.layers.Dense(64,activation = "relu"),
    tf.keras.layers.Dense(32,activation = "relu"),
    tf.keras.layers.Dense(1)
])
```

```
model.compile(
    optimizer = "adam",
    loss = "mse",
    #metrics = [MeanSquaredError()]
)
```

```
history = model.fit(x_train,y_train,batch_size = 32,epochs = 10,validation_data = (x_test,y_test))
```

```
Epoch 1/10
60094/60094 [=====] - 91s 1ms/step - loss: 3.3147e-04 - val_loss: 1.6350e-04
Epoch 2/10
60094/60094 [=====] - 89s 1ms/step - loss: 1.4200e-04 - val_loss: 1.3914e-04
Epoch 3/10
60094/60094 [=====] - 78s 1ms/step - loss: 1.3288e-04 - val_loss: 1.2792e-04
Epoch 4/10
60094/60094 [=====] - 80s 1ms/step - loss: 1.2970e-04 - val_loss: 1.1499e-04
Epoch 5/10
60094/60094 [=====] - 82s 1ms/step - loss: 1.2404e-04 - val_loss: 1.1176e-04
Epoch 6/10
60094/60094 [=====] - 90s 1ms/step - loss: 1.2683e-04 - val_loss: 2.2796e-04
Epoch 7/10
60094/60094 [=====] - 80s 1ms/step - loss: 1.5276e-04 - val_loss: 1.2242e-04
Epoch 8/10
60094/60094 [=====] - 91s 2ms/step - loss: 1.2259e-04 - val_loss: 1.1199e-04
Epoch 9/10
60094/60094 [=====] - 81s 1ms/step - loss: 1.2165e-04 - val_loss: 1.1491e-04
Epoch 10/10
60094/60094 [=====] - 90s 1ms/step - loss: 1.1935e-04 - val_loss: 1.1848e-04
```

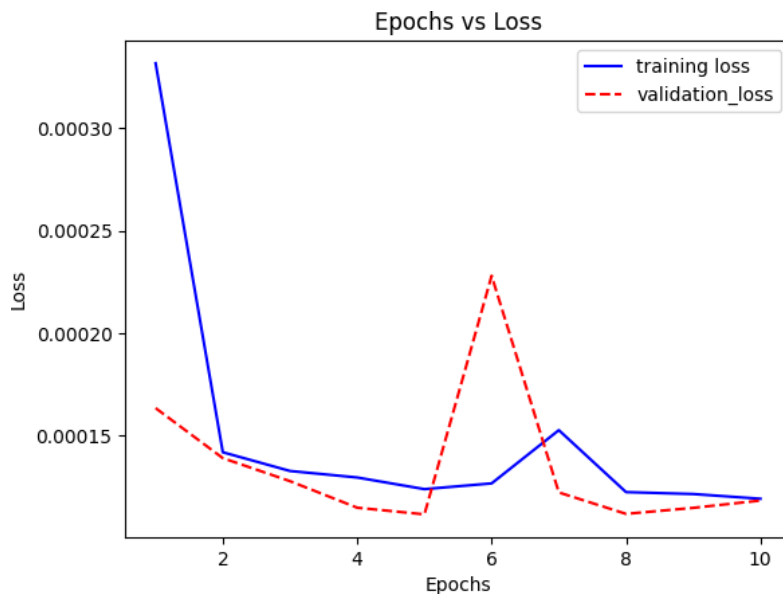
```
y_predict = model.predict(x_test)
```

```
15024/15024 [=====] - 12s 820us/step
```

```
R2 = r2_score(y_test,y_predict)
R2
```

0.9793446531959558

```
train_loss = history.history["loss"]
validation_loss = history.history["val_loss"]
epochs = range(1,len(train_loss)+1)
plt.plot(epochs,train_loss,"b-",label = "training loss")
plt.plot(epochs,validation_loss,"r--",label = "validation_loss")
plt.title("Epochs vs Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



```
model.save("/content/model_9810.h5")
```

x_train,y_train

```
(array([[0.86325636],
        [0.87677389],
        [0.99001129],
        ...,
        [0.8808648 ],
        [0.89700176],
        [0.98247467]]),
 array([[0.98036456],
        [0.98622828],
        [0.99181767],
        ...,
        [0.97934623],
        [0.97654445],
        [0.98450636]]))
```

```
plt.scatter(x_train,y_train)
```

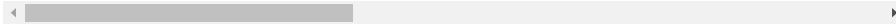
```
from sklearn.ensemble import GradientBoostingClassifier,GradientBoostingRegressor
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression,LinearRegression
from sklearn.svm import SVC,SVR
from sklearn.ensemble import RandomForestClassifier,RandomForestRegressor
gbc = GradientBoostingClassifier()
gbr = GradientBoostingRegressor()
gnb = GaussianNB()
log_reg = LogisticRegression()
li_reg = LinearRegression()
svr = SVR()
rfr = RandomForestRegressor()
```

```
gbr.fit(x_train,y_train)
li_reg.fit(x_train,y_train)
```

```
svr.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_gb.py:437: DataConversionWarn
y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversio
y = column_or_1d(y, warn=True)
```

```
▼ SVR
SVR()
```



```
rfr.fit(x_train,y_train)
```

```
<ipython-input-25-3f392adc2e88>:1: DataConversionWarning: A column-vector y was passed whe
rfr.fit(x_train,y_train)
```

```
▼ RandomForestRegressor
RandomForestRegressor()
```



```
gbr_sc = gbr.score(x_test,y_test)
li_regsc = li_reg.score(x_test,y_test)
svr_sc = svr.score(x_test,y_test)
rfr_sc = rfr.score(x_test,y_test)
```

```
scores = {
    "gbr":gbr_sc,
    "li_reg":li_regsc,
    "svr":svr_sc,
    "rfr":rfr_sc
}
```

```
scores
```

```
{'gbr': 0.982431000309922,
 'li_reg': 0.7557724154970659,
 'svr': -0.37827095178726755,
 'rfr': 0.9814162008918557}
```

```
import joblib
joblib.dump(gbr, 'gbr_model.joblib')
```

```
['gbr_model.joblib']
```

```
import pickle
with open('model.pkl', 'wb') as file:
    pickle.dump(gbr, file)
```

```
import pickle
model_file = "PI_model.pkl"
with open(model_file,"wb") as f:
    pickle.dump(model,f)
```

```
file = "min_max_scaler.pkl"
with open(file,"wb") as f:
    pickle.dump(min_scaler,f)
```

```
with open("/content/gbr.pkl","rb") as f:
    model = pickle.load(f)
```

```
x = zeros(10000001, 1);
t = zeros(10000001, 1);
for i=1 : 10000001
    if PIout_with_loadchange(i,1) == Error_wit_load_change(i,1)
        x(i,1) = PIout_with_loadchange(i,2);    % append i th row to PI_OutNew
        %PI_OutNew(i,2) = PI_Out(i,2);
        t(i,1) = Error_wit_load_change(i,2);    % append i th row to VdcNew
        %errorNew(i,2) = error(i,2);
    else
        %skipdata
        for k=1 : 5
            if PIout_with_loadchange(i+k,1) == Error_wit_load_change(i,1)
                x(i,1) = PIout_with_loadchange(i+k,2);    % append (i+k) th row to PI_OutNew
                %PI_OutNew(i,2) = PI_Out(i+k,2);
                t(i,1) = Error_wit_load_change(i,2);    % append (i+k) th row to VdcNew
                %errorNew(i,2) = error(i,2);
                break
            %else
            % k=k+1 % run next iteration
        end
    end
end
end
end
```

```
model_path = "/MATLAB Drive/model_9967.h5";  
%matlab_model = importKerasNetwork(model_path);  
model_9967 = importKerasNetwork(model_path);
```