CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

In the past several decades, there has been a rapid growth in the power grid all over the world which eventually led to the installation of a large number of new transmission and distribution lines. Moreover, the deregulation of electric power has increased the need for reliable and uninterrupted electric power supply to the end users who are very sensitive to power outage.

One of the biggest problems in electrical power system is the interruption or the discontinuity of power supply which is caused by occurrence of faults. These faults are inevitable and normally are an abnormal flow of current in a power system's component. They cannot be completely avoided since some of them occur due to natural reasons which are beyond human control. Hence, when the power system has a well-coordinated protection system, it detects any kind of abnormal flow of current in the power system, identifies the fault type, accurately locate the position of the fault in the power system network and isolate it. The isolation of the fault must be very fast to avoid damage of power equipment and power outage. Also, the faults must be cleared very fast so as to restore the power to the isolated areas. The clearing of the faults is done using protective devices which sense the fault, respond immediately and disconnect the faulty section from the good ones (S. K. Gupta Nagrat and Kotari, 2003). To protect the power system transmission lines, faults must be detected and isolated accurately. The control center of a power system contains large member of alarms which receives signals from different protection schemes for different types of fault.

The operators in the control center must work on the large amount of data obtained and know the required fault information to enable them determine fault types and their parameters. Due to the large number of calculations needed to be made so as to obtain this required information for the fault, it takes a longer time.

The large number of data and calculation required for the determination fault types and their parameters are the challenges to the protection of electrical power system transmission line. But conventionally, when a fault occurs in the power systems, the fault current increases and voltage decreases. These faults are classified as transcient faults.

Transient faults are the type of fault which occurs on the power system but last for a short period of time, ranging from micro seconds to one second (Gupta, 2006, Nagrat and Kotari, 2003, Murty 2007).

Transients can be classified depending on their speed of occurrence as follows:

(a) Extreme fast transient: This type of transient is caused by lightning and switching on and off of the supply areas. It carries the surge parameters (electromagnetic waves or over - currents and voltages) along on the transmission line with the speed of light (3.0×10^8 m/s). A power system with a good protection system will bury travelling charges in the surge into the ground through the lightning arresters (Gupta, 2006, Nagrat and Kotari, 2003, Murty, 2007).

(b) Medium fast transient: Most medium fast transients are short circuit faults. It occurs on the exposed overhead transmission lines when the insulation is broken down by the over– voltage caused by the surge, birds and other mechanical causes. Such short circuits if allowed for a long time may result to thermal damage of the electrical equipment.

(c) Slow transient: This is a transient stability which occurs as a result of short circuit in any part of the power system. It results to an instantaneous partial collapse of the system bus voltage, generator power output etc.

During these transients, the system is subjected to the greatest stress from excessive over current or voltage. These faults depending upon their severity can cause a severe damage to the power system (Nagrat and Kotari, 2003, Gupter, 2004, and Rao M. & Hasabe, 2012).

In some extreme cases, there may be a complete shut-down of the power system component or a black-out of a large area of supply.

(d) Persistent faults: These consist of two types namely:

i. Symmetrical faults:

ii. Unsymmetrical faults:

The methods of detection of these faults are distinctively unique, in the sense that, there is no one general method of detection of the faults. Thus, an automatic detection of these faults can greatly enhance the power systems reliability because, the faster we restore power, the more money and valuable time we save.

Method of detection of faults on the transmission lines can be broadly classified into the following categories, even though each method is superior over the other in different characteristics. This means that some are specially used with high efficiency, less error percentage, simple, fast in operation and produces accurate result. These include;

- Impedance measurement - based method
- Travelling wave - based method

- Signal processing - based method (Fourier Transform (FT), Fast Fourier Transform (FFT), Wavelet and S- Transform)

- Artificial Intelligent system - based method

Artificial Intelligent systems have been in use for fault diagnosis in power systems for quite some years now. Four major artificial intelligent based methods are widely used in the power system engineering field for fault diagnosis [Sai R. et al, (2013), Rao M. &Hasabe, (2012)]

They are;

- Expert system method

- Fuzzy logic systems (FL)

- Genetic algorithm method (GA)

- Artificial neural network method (ANN)

In this work, two algorithms for signal processing methods are developed for the detection of fault on any transmission line network. These methods will be compared with each other based on which has less percentage error, accuracy, robustness and efficiency.

1.2 Statement of Problem

When the normal current flowing on the Power System Transmission Line is diverted from the intended path to another path, the diversion to another is caused by the presence of fault on the transmission line. This fault developed on the electric transmission circuit can be as a result of reduction in the insulation strength between the phase conductors and earthed screen surrounding the conductors.

Various methods have been employed by researchers for detecting these faults. Among the methods is the Signal Processing method which include; Fourier Series, Fourier Transform, Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT), Wavelet Transform, S – Transform etc.

Many research works show that the Signal Processing techniques for fault detection in Power System Engineering has not been widely and deeply implemented [ ]. It is not yet cleared enough how the signal processing technique can be used for fault detection on the Power System Transmission Line. Moreso, no definite signal processing mathematical procedure has shown a clear and better result after implemented for the detection of fault [ ].

Researchers and students find it difficult to understand and implement this signal processing technique for fault detection and analysis on the Power System Transmission Lines and other electrical circuits [ ].

In this dissertation, a new simplified DFT and FFT mathematical and Simulation algorithmic approaches are presented to solve the problem of complexity, clearity and incomprehensible of DFT and FFT applications in power system AC transmission line circuit analysis.

1.3 Aim of the Study

The aim of this dissertation is to develop a fault detection model for Power System Transmission Lines using Discrete Fourier Transform (DFT).

1.4 Objectives of the Study

The objectives of this study are as follows;

(a) To extract and sample the three - phase pre-fault and fault voltage and current data obtained from Onitsha 330kV Transmission Line Station for implementation in the research methodology.

(b) To develop a mathematical analytical procedure based on DFT and FFT equations for fault detection on the transmission line faults.

(c) To model a Matlab/Simulink block diagrams using the mathematical analytical procedure based on DFT and FFT equations for fault detection on the transmission line faults.

(d) To simulate the model for three phase pre-fault and fault conditions without and with DFT and FFT simulink models respectively.

(e) To test (simulate) the Matlab/Simulink DFT and FFT models on Onitsha – Enugu 330kV and IEEE 14 – Bus System transmission line.

(f) To compare the result of mathematical and the simulation methods

(g) To compare the result of DFT and FFT models applied to 330KV Onitsha – Enugu and that of IEEE 14 – Bus System transmission line.

1.5 Significance of the Study

The significance of this study is, that, it exposes a detailed knowledge of the application of Fast Fourier Transform signal analysis method of fault diagnosis in power system engineering and show a faster, error free, less ambiguous, simpler, accurate and efficient method that can be employed in electrical power system for quick detection of faults.

It also serves as a guide or solution to power system protection problems in the utility companies like EEDC, Researchers in various research institutions (energy institutions), Government officers, and Policy makers.

1.6 Scope of the Study

This research can be implemented on any power system transmission (short, medium or long lines). Onitsha – Enugu 330kV power system transmission line is used as a case study.

CHAPTER TWO

LITERATURE REVIEW

2.1. Power System Engineering Protection

According to Christopoulus, 1999, Francisco, 2020, Gupter, 2004 and Nagrat et al (2004), Power System Engineering Protection is the application of various switching devices well-coordinated to link each other, to protect the power system components/equipment against any occurrence of fault and to isolate that faulty section to avoid damage of the power system equipment or that faulty section.

2.1.1 Power System Protection Equipment

Christopoulus, 1999, Francisco, 2020, Gupter, 2004 and Nagrat et al (2004), also explained that power system protection equipment are those devices or switching devices that when linked or coordinated together, connected to any of the power system components and protect a particular power system component against fault. They include protective relays, circuit breakers, batteries (dc source), voltage and current transformers.

2.1.2 Power System Protection Scheme

Christopoulus, 1999, Francisco, 2020, Gupter, 2004 and Nagrat et al (2004), continued that, the power system protection scheme is a system containing the protective devices linked together and when connected to a power system component protects it against any fault.

Each power system component has a particular protection scheme designed and suitable for it. Figure 2.1 shows a typical protection scheme for power system transmission line.

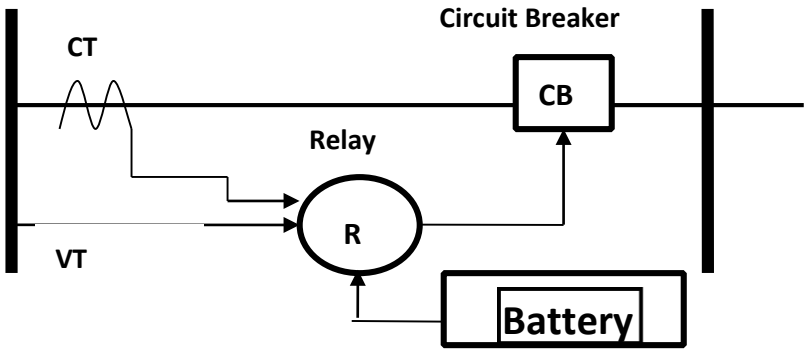

Figure 2.1 A typical protection scheme for power system transmission line.

2.2 Components of Power System

According to Shih, 2012, Francisco, 2019 and Anshika R (2019), Power system components includes: Power Plant (Power generation unit): This unit generates electricity of various voltage capacities.

Transformers: This includes the step – up or down of the voltage level for transmission and distribution.

Transmission lines: This unit carries the power from one place to another.

Distribution or Substations: Here the voltage is step – up or down to meet the load demand.

2.2.1 Generation Unit

This unit comprise of the generators, motors, transformers, bus-bars etc. This equipment must be protected against any fault to ensure continuous generation and transmission of power to the remaining power system components. Generators are subject to several adverse conditions. These conditions can be placed into one of these categories; first, the faults that occur within the generator itself. Second is the fault or abnormal operation that occurs outside the generator, but involves the generator in one way or the other. Generator unit faults are grouped into internal and external generator faults (Fangyu et al, 2019 and Klingerman et al, 2011).

2.2.1.1 Internal Faults

According to Shanaya, 2020, generator internal faults can be grouped into stator winding faults, field winding or Rotor circuit faults and abnormal operating condition.

He continued that Stator winding faults occurs mainly becaused of the insulation breakdown of stator coils. Furthermore, he established that stator winding faults are grouped into phase – to – earth faults, phase - to – phase faults and inter – turn faults. Shanaya continued that stator winding faults are severe and can cause serious damage of the generator part within a shortest possible time. He however concluded that phase – to – phase faults and phase – inter turn faults are not common and are more difficult to be detected.

According to Shanaya, 2020, field winding or rotor circuit fault can be grouped as conductor – to – earth fault or inter – turn faults and are caused by severe mechanical and thermal stress.

2.2.1.2 External Fault

These include system disturbances and hazardous operational conditions external to the generator.

- Generator motoring
- Over-voltage and under-voltage
- Inadvertent energizing which result in non-synchronous connection.
- Unbalanced current (pole disagreement or pole flash over).
- Overload and over temperature operation (loss of cooling)
- Under and over-frequency operation
- Loss of synchronism (out and stop) Sub-synchronous resonance etc.

All these fault conditions can be prevented when the generator is protected with differential protection scheme. Figure 2.2 illustrates a typical differential protection scheme for generator protection (Gupter, 2004, and Klingerman et al, 2011).

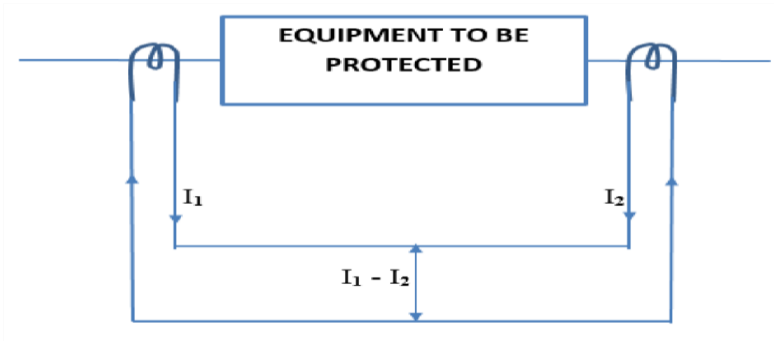

Figure 2.2 Differential protection schemes

2.2.2 Motor Protection

Motor and generator have many similarities in area of physical features and operation. Since most motors can operate in generator mode and vice versa. Motor need protection for internal winding, ground faults, thermal over load, single – phase failure and loss of load etc. Since motors are electromechanical devices. It means that, they operate electrically and mechanically and both operating mode can experience failures. Statistically, (40-50) % of motor failure is due to rotor bearing, (25-35) % is due to stator faults, while approximately 10% is due to rotor electrical faults.

However, fuses are the most widely used protection device for motors. They must be sized to handle starting current and be able to clear sustained winding and ground faults in case of any. Though large motors can be protected using time over-current relay for both phase and ground faults (Gupter, 2004, C. L. Wadhwa, 2010 and Klingerman et al, 2011).

2.2.3 Transmission Line Faults

The faults that occur on the transmission line may be classified into balanced and unbalanced faults. They sometimes are referred as symmetrical and unsymmetrical faults respectively. They are all referred as three phase faults. The three-phase symmetrical fault can also be called simultaneous short circuit across all the three phases of the transmission line. This fault rarely occurs, but is the most severe and dangerous type of fault among other types of faults that occur on the transmission line and power system at large. This is because; the network is balanced (Saadat, 2006).

The three phases can also be called three phase short circuit or shunt fault. Moreover, the power system must be protected against flow of heavy short circuit currents which can cause a permanent damage to major equipment in the system. The protection is by disconnecting the faulty section of the system using of circuit breakers and protective relaying.

Most importantly, one must be able to calculate, approximately at least the size of the protective reactors which must be inserted in the system to limit the short circuit currents to a value that is not beyond the one the circuit breakers can withstand.

The rupturing capacity of the circuit breakers during short circuit fault is dependent on the symmetrical short circuit current magnitude. The magnitude of the short circuit current can simply be calculated in line with the settings of the relay and fault current values of the unsymmetrical (unbalanced) faults (Gupta, 2004 and C. L. Wadhwa, 2010).

2.3. Balanced Fault

It can be seen clearly that, in application of impedance method for fault diagnosis on the transmission line network, we need to apply it first, to balance or symmetrical fault. This symmetrical fault is no other fault than the symmetrical three phase short circuit fault. Secondly, to unbalanced or unsymmetrical fault which include, single line to ground, double line to ground and single line to line faults on the transmission line.

2.3.1. Short Circuit (SC) Three Phase Fault

According to Gupta, (2004), symmetrical fault is first calculated by determining the voltage at any point (bus voltage) on the transmission line, the current during the fault in any of the branch network and the value of reactance necessary to limit the fault current to any desired value. He continued that, these parameter values form an impedance value and data for selection of circuit breakers and design of protection scheme for the short circuit fault is provided.

However, Saadat, (2006), Nagrat and Kotari, (2003), stated that, the three-phase short circuit fault is the simultaneous short circuits across all the three phases which does not occur frequently but is the most severe type of fault encountered. This is because, the network is balanced. They explained that, short circuit current can be determined using thevenin's, bus impedance matrix, shunt admittance and symmetrical component methods.

In their work, Madueme, et al (2015) agreed with Saadat, Nagrat and Kotari, (2003) and stated that, the short circuit fault current can be determined using the same method. But sighted a case study of Enugu-Nkalagu - Abakaliki 132kv transmission line in Nigeria.

According to Tharaja, (2001) and Jensen C. F. (2014), if fault occur on the transmission line the short circuit current from the generating unit will have a value limited by the impedance of the generator and the point of fault. He continued that, this means that, the knowledge of the impedance of different equipment and circuit on the system is very important for the determination of symmetrical fault current. However, for transmission line, the impedance limiting the fault is mostly resistive. The short circuit current calculated will be used for the setting of the relay, selection

of circuit breakers and the design of the protection scheme for the power system component.

In their study, Karl and David, (2010) explained that, short circuit fault is an accident or intentional low resistance or impedance connection established between two points in an electric circuit that by-passes part of circuit. The current in the electric circuit flows through the path of least resistance and if an alternative path is created where two points in a circuit are connected with low resistance or impedance then current will flow between the two points through the alternative path.

However, in SC conditions, the normal level of current flow is suddenly increased by a factor of hundreds or even thousands which is deadly magnitude called SC current. He concluded that SC current is divided into symmetrical and asymmetrical current. When the wave form of the current is symmetrical about the zero axes, it is called symmetrical and when it not about the zero axes is called asymmetrical. Thus, SC current is a mixture of symmetrical and asymmetrical current.

2.3.2. Methods of Calculating Short Circuit (SC) Current

Calculations of Short Circuit (SC) current involve the representation of the entire power system impedances from the point of the SC back to the source (S) of the SC current. The value of the impedance depends on the SC current ratings for the devices or equipment under consideration.

The following methods can be applied in SC computation;

I. Impedance method

II. Composition method

III. Conventional method

2.3.2.1. Impedance Method

According to Karl and David, (2010), this method is reserved for low voltage network. The main objective is to calculate the SC current and use the data obtained for the design of good protective scheme.

Here, the SC is given as I_K and can be obtained by

$$I_K = \frac{U_F}{Z_K} \tag{2.1}$$

Where I_K is the per phase current, Z_K is the resulting impedance per phase from the voltage source to the fault point. U_F is the phase voltage of the voltage source.

U_F and Z_K can be seen as the Thevenin's voltage and impedance respectively. For the transmission line, the resistance and the reactance of the line conductors are generally expressed in terms of ohms-per phase per unit length for voltage above 600V. The resistance becomes so small that it is normally omitted and the reactance of the line conductors is estimated to be equal to the impedance of the line. If the length of the line is less than 1000feet, then, the entire impedance can be omitted with negligible error. Thus, the resistance of the transmission line is given;

$$R_L = \rho(l\ km/Amm^2)\ \Omega \tag{2.2}$$

$\rho = 1.7 \times 10^{-8}$ Ωm (copper conductor) (2.3)

$\rho = 2.6 \times 10\text{-}8$ Ωm (aluminum conductor) (2.4)

reactance of the transmission line is given as

$X_L = l.K$ (2.5)

K = 0.4 (copper conductor) (Karl and David, (2010) (2.6)

There are other components on the transmission line such as Circuit Breakers (CB), buses, and connections whose impedances are generally not considered in the calculation of total impedance, but have significant impedance effect.

The value of short circuit current of these component can be neglected only when the protection engineer wants to use the low rated circuit component.

2.3.2.2. Composition Method

In their work, Faruqul, (2014), explained that, the method is used when the characteristics (parameter value) of power supply are not known. The impedance of the system is calculated based on the estimation of the short circuit current at its origin. Power factor given as

$pf = \cos\theta = \frac{R}{X}$ assumed (2.7)

Where R is the resistance and X is the reactance of the line.

θ is the phase angle between the R and X.

The power factor (pf) is assumed to be identical at the origin of the circuit and the fault location point.

Another assumption is that, the primary impedance of the successive sections on power system transmission line is sufficiently similar in their characteristics to justify the replacement of vector addition of the impedances by algebraic addition.

Therefore, this approximation is used to calculate the short circuit current value with sufficient accuracy for the selection of CBs and protection scheme design. Given that the short circuit power is S_K.

Thus,

$$S_K = \sqrt{3} \,.\, U.\, I_K \tag{2.8}$$

Where, U is the normal voltage of the line before the occurrence of short circuit.

I_K is the short circuit current when the fault occurred.

If the impedance is known, we can then obtain short circuit power S_K as;

$$S_K = \frac{U^2}{Z_0} \tag{2.9}$$

Where Z_0 is the transmission line known impedance.

2.3.2.3 Unbalanced Fault

According to Saadat, (2006), Nagrat and Kotari, (2003), unbalanced fault are those faults that occur frequently on the power system. These faults occur mostly on the power system. Mostly, they occur on the transmission lines. These faults include single line to ground, line to line and double line to ground faults. They sometimes refer to as unsymmetrical faults. In his book, Saadat, (2006), only applied bus – impedance matrix, symmetrical component methods in both mathematical and matlab program approach.

Conventionally, bus impedance matrix and symmetrical component methods are real methods for determining fault current, voltage and impedance of a line during any of the unbalanced fault.

These fault parameters are to be used for the selection of CBs and for the protection scheme design. Since these unbalanced or unsymmetrical faults are mainly the types of fault that frequently occur on the transmission line.

This work majorly dwells on the application of FFT method in fault diagnosis on the transmission line with distance protection scheme. Each of these methods is applied to the Nigerian 41 – bus transmission line network with distance protection scheme. Each transmission line is differentiated using their respective location name, line distance and other parameters such as Generator, transformer, line voltage and current and load ratings.

2.3.2.4 Conventional Method of Fault Study

The conventional methods include; three phase short circuit fault calculation, bus – impedance (calculation and Matlab program) and symmetrical component. These methods can be applied for the diagnosis of fault on the transmission line.

The three-phase short circuit calculation has been dwelt on previously in section 2.3. However, conventional method dwelt more on the bus-impedance and symmetrical component methods using calculations and Matlab program approaches. We have to bear in mind that, the aim of using these methods is to determine the fault parameters which are voltage, current and impedance on the line during fault condition. These fault parameters were used for the selection of CB's and to design the distance protection scheme for the transmission lines.

The work of the protection scheme is to ensure fault is protecting the system and sending tripping signal to the CB to disconnect or isolate the faulty area, ensure the fault is cleared and that power is restored.

2.4 Bus – Impedance and Symmetrical Component Method of Fault Study

According to Nagrath and Kotari, (2003), a set of three balanced voltage and current (phasors) are characterized by equal magnitudes and inter phase difference of 120°. These sets are said to have a sequence abc (positive sequence) values phase b lagging a by 120° and phase c lagging b by 120°.

They also considered the sequence impedance of the transmission lines. Here, a fully transposed transmission line carrying unbalanced currents is used to illustrate symmetrical component application, obtaining the symmetrical component voltage, current, impedance and their faulty parameter values which are the parameters obtained when the symmetrical component method is applied for the diagnosis of faulty transmission line.

Though Saadat, (2006) applied the same method, but in his work, he combined both the symmetrical and bus – impedance matrix method and also incorporated them with matlab program such that, when the program is ran will bring out the results as the faulty parameters (voltage, current and impedance parameter values) for various type of unbalanced faults.

This is shown clearly when Madueme et al, (2015), applied symmetrical component method for the diagnosis of faults on transmission line a case study of Enugu – Nkalagu – Abakaliki transmission line. In their work, each type of unbalanced fault likely to occur on the line where shown and has their own voltage, current and impedance parameter and their symmetrical component values.

Williams, (2012), explained that, unbalanced faults can also be referred as shunt faults and can be computed using symmetrical component method. He modeled the transmission line using the three-phase compensator network on Matlab Simulink. The three phases contain six 350MVA generators, 60Hz and transmitting 735kV power through the transmission line of 600km. the transmission line is spitted into two 300km lines connected to B_1, B_2 and B_3.

He increased the transmission capacitors representing 40% of the line reactance. Both lines were also shunt compensated by a 330MVAR shunt reactance. The shunt and series compensator are both located at the B_2.

2.5 Classification of Power System Transmission Line

Power system transmission line can be defined as high voltage electrical conductor (circuit) carrying current and voltage from the electrical power generation (sending end) to a transmitting station (receiving end). It can as well be a high voltage circuit transferring electrical power from one transmitting station to another.

Table 2.1 shows the classification of transmission lines into local service, distribution, sub - transmission and transmission line (Gupter, 2004, and Klingerman et al, 2011).

Table 2.1 Power system circuit classification (Gupter, 2004)

S/N	CLASSIFICATION	OPERATING VOLTAGE (kV)	FUNCTION
1	Local service consumer feeders	220V to 415V	Circuits owned by the customer consumer feeders
2	Distribution	11kV to 33kV	Circuit from transmission station to local service transformer
3	Sub – transmission Generator Step – up	33kV to 132kV 11kV to 33kV	Circuit from generators to step – up transformer stations Transmission station to distribution stations
4	Transmission	132kV to 330kV	Circuit that carry large bulk power from points of one transmission sub-station to another

Generally, utility service owns the local service transmission line circuits and classified based on voltage, load and uses. The local utility includes; residential, agricultural, commercial and industrial. Their protective devices are fuses, circuit breakers and ground fault interpreters (GFI). The other transmission lines are as well classified and explained by the Table 2.1 above (Gupter, 2004 and Personal et al, 2013). The power transmission lines incur greatest number of faults than other power system components.

According to Saadat, (2006), and Zimmerman K. (2010), faults on power system are divided into three phase balanced and unbalanced faults. They occur on the transmission line very often and about 75% of the faults that occur on powers system occur on the transmission line. This is because, the transmission line are widely

branched, have greater length, operate under variable weather conditions and are subject to the action of atmospheric disturbances of electrical in nature. Most of these faults are caused by the breakdown at normal voltage which causes deterioration or aging of insulation and unpredictable events such as blowing of heavy winds, tree falling across transmission lines, vehicle colliding with electric towers or poles, aircraft colliding with lines, switch surges or lightning strokes etc.

The Tables 2.2 and 2.3 illustrate the various causes of faults and their percentage variation of equipment that may be affected or damaged during such fault occurrence (Gupta, 2004).

Table 2.2 Percentage of various causes of faults on the transmission line (Gupter, 2004 and Nagrat et al, 2003)

S/N	Causes	% of total causes
1	Lightning	12
2	Sleet, wind, mechanical (jumping conductors)	20
3	Apparatus/equipment failure	20
4	Switching to a fault	20
5	Miscellaneous (tree falling on lines, bird-age, vandalization, accidents etc.)	28

Table 2.3 Frequency of fault occurrence in different equipment used in power system (Gupter, 2004 and Nagrat et al, 2003)

S/N	Equipment	% of total causes
1	Overhead line	50
2	Underground Cable	10
3	Transformers	10
4	Switch gears	15
5	Engineering control equipment	3
6	Instrument transformers (CTs and VTs)	2
7	Miscellaneous	10

2.6 **Fault Diagnosis using Wavelet Transform (WT)**

This is a form of wave limited time whose average value is zero. Abdelsalem (2008) also commented that, WT is one of most widely used signal processing methods in determining fault parameters of a transmission line. They continued that, the discrete wavelet transform (DWT) can be applied to a measured voltage or current signal which is input signal of the system.

He continues that WT can be used in two ways namely continuous and discrete wave transform. The most important factor in WT is to determine the level of decomposition and the mother wavelet.

Where $1 \leq k \leq N$.

In Time – frequency – Domain method, short time Fourier transform (STFT) extracts the relevant time-amplitude information from the wave signal travelling on the

transmission line. Short time Fourier Transform is used to overcome the shortcomings of the Discrete Fourier Transform (DFT). He continues after dividing the signal into small segments which was assumed stationary, the signal is multiplied by a window function w (t – τ) within the Fourier integral. If the window length is infinite, the discrete Fourier transform will be applied. Therefore, in other to obtain a stationary signal, the window length must be short enough, in that case, STFT will be used. The narrower the window, the better time resolution and better stationary but the frequency resolution will be poor.

According to Abdelsalam (2008) one problem with STFT is that, it becomes difficult for one to obtain the spectral components that exist at that particular point is time. Thus, STFT is defined in term as of current signal travelling along the transmission line during fault occur.

$$\mathrm{STFT}\ (t, w) = \int_{-\infty}^{\infty} I(t).\, w(t - \tau).\, e^{-jwt} dt \qquad (2.10)$$

Where I (t) is the measured signal (current), w is the frequency of the signal, w (t - τ) is the window function, τ is the translation factor, and t is the time of application of the processing of the signal.

He highlighted that the wavelet transform is a new and powerful method of analysis travelling wave signals on the transmission line. He continued that it, provides multiple resolutions in both time and frequency domain unlike, FT and STFT. The windowing of wavelet transform can be adjusted automatically for low and high frequency application. This means that, it uses short time intervals for high frequency components and longtime intervals for low frequency components. It decomposes signals into scales applying its analyzing functions called mother wavelet. The mother wavelet high frequency version of wavelet transform is used for temporal analysis while main frequency analysis is done using dilated, low

frequency version of the mother wavelet. Wavelet transform, satisfies the requirement of both time and frequency localization of the travelling wave signal of electrical power moving alone the transmission especially during fault occurrence. Gives a function x (t), we can obtain a continuous wavelet transform of the function x (t) using;

$$\mathrm{CWT}\ (a, b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x\ (t)\ \psi^{*\frac{(t-b)}{a}} dt \tag{2.11}$$

Where a and b are the scale and the translation parameters respectively. Ψ (t) is the mother wavelet which is a band-pass filter while ψ^* is its complex conjugate form. The factor is used to ensure each seated wavelet function has the same energy as the wavelet basic function. Abdelsalam (2008) also, explained that the wavelet transform of a given sample travelling wave signal can be determined by applying the DWT gives as

$$\mathrm{DWT}(k, n, m) = \frac{1}{\sqrt{a_0^m}} \sum_n x\ [n]\ \psi^{\left(\frac{k-nb_0 a_0^m}{a_0^m}\right)} \tag{2.12}$$

Where Ψ (t) is the mother wavelet and the sealing and translation parameter 'a' and 'b' are replaced with a^m_0 and $nb_0\ a^m_0$ respectively, while n and m are integer variables.

He added that, mother wavelet Ψ(t) need in fault location can be selected by considering the DaubechiesCoiflets, Symlets and Biorthogonal wavelets discretely represented in Matlab.

The best of them has high correlation with the high frequency travelling wave signals on a typical transmission line network. We can also test the mother wavelet using their smoothness and regularity factor.

Patel and Patel (2012), applied the discrete wavelet transform method for in diagnosing fault on the transmission line. In their work, they applied wavelet transform for the analysis of transient voltage and event signals during fault occurrence carrying high frequency components of the signals.

According to them, these high frequencies carrying fault also carry relevant information regarding the types and location of the faults. They continued that, multi-resolution analysis method of the discrete wavelet transform is used which decomposes the original signal into low frequency signal called approximations and high frequency signal called details. They also mentioned the importance of mother wavelet and the importance of number of multiple decomposition steps.

According to them, the number of decomposition step is influenced by sampling frequencies of the original signal such that, in the first decomposition step, the signal is decomposed into D_1 component of high frequency band and A_1 component of low frequency band. While the D_1 component frequency band is $(f_S)/ (2\text{-}f_S/4)$ H_Z, the A_1 component frequency band is $(f_s/40)$ H_Z, where f_S is the sampling frequency in the second decomposition, D_2 component frequency band becomes $(f_S)/(4\text{-}f_s/8)$ H_Z, while A_Z component of frequency band becomes $f_S/8\text{-}0$ H_Z. The signal of the desired component can be obtained through repetitions determined by comparing the scale of sampling frequency with that of the frequency component of the desired signal.

In their work, Chengzong et al (2010) stated that the performance of wavelet transform highly depends on the selection of the mother wavelet ψt. They continued that, to perform wavelet transform, we consider other mother wavelet such as Daubechies (Db), symlets, coiflects, Biorthogonas etc. They adopted the multi resolution analysis method, to decompose the voltage and current signals during fault (power swing). Among the mother wavelet listed above, they choose Daubechies mother wavelet family as the most suitable wavelets in multi-resolution

analysis. Lots of trials were carried out to find the desired wavelet for fault diagnosis especially in differentiate fault from power swing.

This is to enable them identify the most sensitive component reflecting certain frequency band which can clearly show fault starting and clearing times but will have less influence under power swing. On this note, multi-resolution analysis based on Daubechies-8 (Db8) mother wavelet was chosen for the investigation.

In his study, Reddy (2009) applied the coiflet mother wavelet transform method for the analysis of fault on the transmission line. His choice was in order to obtain differences in magnitudes between the faulted and healthy phases in the case of coiflet is much smaller than the corresponding other types of mother wavelets. In general, the essence of using wavelet transform is to best analyze the travelling wave high current and voltage signal magnitudes during fault on the transmission line.

These wave signals are then transformed and used as inputs of continuous wavelet transform Matlab tool box which processes the signal shows if fault occurred on the line or not.

Roshni et al (2012), applied FFT and WT in detecting and classifying fault on the transmission line. In their work, only the line to ground (L - G) fault were analyzed and its result was comparison of FFT and WT methods. They introduced the L – G fault on the Matlab modeled transmission line under certain time and simulated. The simulation result show that, the FFT algorithm gave a fault current of 2.62mA, while WT gave fault current of 0.061mA. This means that, the WT gave more accurate result than FFT, but both were used in the presence noise.

2.6.1 Multi-Resolution Analysis (MRA) According to Wikipedia, 2017, a Wavelet is a wave – like oscillation with amplitude that starts from zero, increases, and deceases back to zero. Wavelet Transform is a time – frequency representation of the signal.

Although the time and frequency resolution problems are results of a physical phenomenon (Heisenberg uncertainty principle) and exist regardless of the transform used, it is possible to analyze any signal by using MRA. MRA, as the name implies, analyzes the signal at different frequencies with different resolutions. Every spectral component is not resolved equally as was the case in the STFT.

MRA is designed to give good time resolution and poor frequency resolution at high frequencies and good frequency resolution and poor time resolution at low frequencies. This approach makes sense especially when the electrical fault signal at hand which is the transmission line voltage and current signals during faulty conditions has high frequency components for short durations and low frequency components for long durations of time. Fortunately, the signals that are encountered in practical applications are often of this type. For example, the following figures 2.3 and 2.4 shows transmission line fault voltage and current wavelet signals of this type. It has a relatively low frequency component throughout the entire signal and relatively high frequency components for a short duration somewhere around the middle (Patel M. and Patel R. N., 2012).

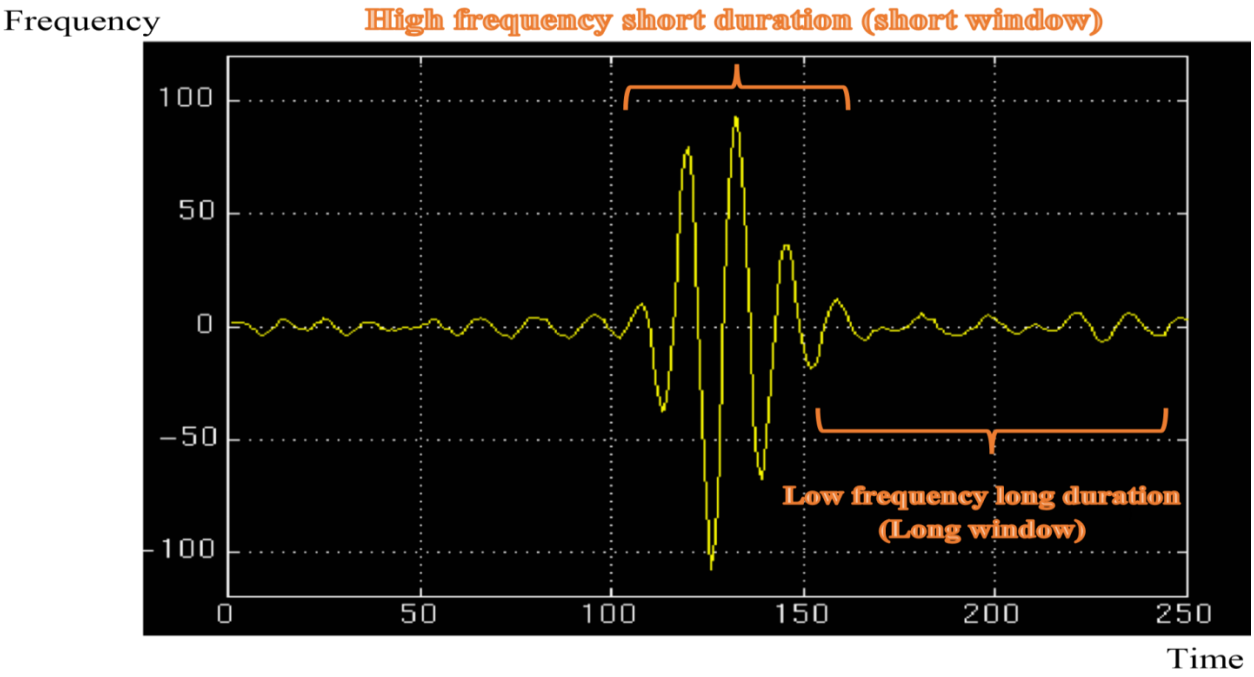

Figure 2.3: A typical Wavelet Signal (Bashier E. 2016) and Mauresh R., 2014)

2.6.2 Continuous Wavelet Transform (CWT)

The continuous wavelet transform was developed as an alternative approach to the short time Fourier transforms to overcome the resolution problem. The wavelet analysis is done in a similar way to the STFT analysis, in the sense that the signal is multiplied with a function {wavelet}, similar to the window function in the STFT, and the transform is computed separately for different segments of the time-domain signal. However, there are two main differences between the STFT and the CWT they are: In Continuous Wavelet Transform (CWT),

1. The transforms of the windowed signals are not taken, and therefore single peak will be seen corresponding to a sinusoid, i.e., negative frequencies are not computed while STFT does the opposite.

2. Also, the width of the window is changed as the transform is computed for every single spectral component, which may be the most significant characteristic of the wavelet transform, but not so the STFT (Rao M., and Hasabe R. P., 2013).

The continuous wavelet transform is defined by this equation:

$$\mathrm{CWT}_{X}^{\psi}(\tau, s) = \psi_{X}^{\psi}(\tau, s) = \frac{1}{\sqrt{s}} \int x(t)\psi^{*}\frac{t-\tau}{s}\, dt \qquad (2.13)$$

As seen in the above equation, x(t) is the signal to be analyzed, (t) is the mother wavelet. The transformed signal is a function of two variables, tau and s, the translation and scale parameters, respectively.

(t) or psi(t) is the transforming function called the mother wavelet. The term mother wavelet gets its name due to two important properties of the wavelet analysis as explained below:

The term wavelet means a small wave signal carrying energy (Figure 2.8). The smallness refers to the condition that this (window) function is of finite length (compactly supported). The wave refers to the condition that this function is oscillatory. The term mother implies that the functions with different region of support that are used in the transformation process are derived from one main function, or the mother wavelet. In other words, the mother wavelet is a prototype for generating the other window functions.

The term translation is used in the same sense as it was used in the STFT above; it is related to the location of the window, as the window is shifted through the signal. This term, obviously, corresponds to time information in the transform domain. However, we do not have a frequency parameter, as we had before for the STFT.

Instead, we have scale parameter which is defined as 1/frequency (sec). The term frequency is reserved for the STFT. Scale is described in more detail in the next section.

2.6.3 The Scale

The parameter scale in the wavelet analysis is similar to the scale used in maps. As in the case of maps, high scales correspond to a non-detailed global view (of the signal), and low scales correspond to a detailed view. Similarly, in terms of frequency, low frequencies (high scales) correspond to a global information of a signal (that usually runs through the entire signal), whereas high frequencies (low scales) correspond to a detailed information of a hidden pattern in the signal (that usually lasts a relatively short time). Cosine signals corresponding to various scales are good examples and are shown in the following figure.

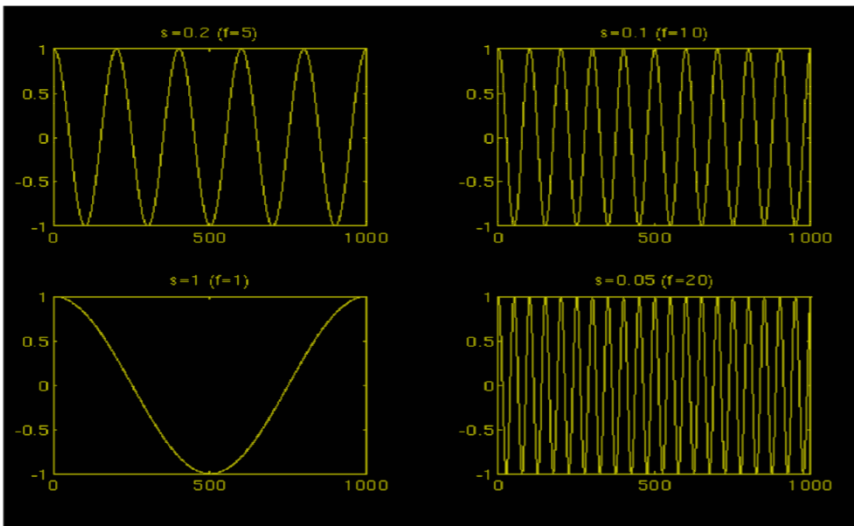

Figure 2.4 Wavelet function with different window functions (Bashier E. 2016) and Mauresh R., 2014).

Describing the window function of the Figure 2.4, the four cosine signals are four forms of one cosine signal showing different scales and frequencies but with same dimension. Here, vertical axis has 1 as the highest magnitude, while the horizontal has 1000 as the highest magnitude. The scale is function of time and frequency of the wave signal. Such that, if scale, s is given as [illegible]. where f is the frequency. Thus, when the frequency (number of resolution of the signal) is 5, the scale becomes 0.2 (Figure. 2.4) as the case may be.

Fortunately in practical applications, low scales (high frequencies) do not last for the entire duration of the signal, unlike those shown in the Figure. 2.9, but they usually appear from time to time as spikes. High scales (low frequencies) usually last for the entire duration of the signal.

Scaling, as a mathematical operation, either dilates or compresses a signal. Larger scales correspond to dilated (or stretched out) signals (large window size) and small scales correspond to compressed signals (small window size). All of the signals given in the figure 2.4 are derived from the same cosine signal, i.e., they are dilated or compressed versions of the same function. In the above figure, $s = 0.05$ is the smallest scale, and $s = 1$ is the largest scale.

In terms of mathematical functions, if $f(t)$ is a given function, $f(st)$ corresponds to a contracted (compressed) version of $f(t)$, $s > 1$ and to an expanded (dilated) version of $f(t)$, $s < 1$.

However, in the definition of the wavelet transform, the scaling term is used in the denominator, and therefore, the opposite of the above statements holds, i.e., scales $s > 1$ dilates the signals whereas scales $s < 1$, compresses the signal. This interpretation of scale will be used throughout this text.

2.6.4 Computation of the CWT

To explain the above equation 2.40 of CWT, x(t) is the signal to be analyzed. The mother wavelet is chosen to serve as a prototype for all windows in the process. All the windows that are used are the dilated (or compressed) and shifted versions of the mother wavelet (t).

Once the mother wavelet is chosen the computation starts with s = 1 and the continuous wavelet transform is computed for all values of s, smaller and larger than 1. However, depending on the signal, a complete transform is usually not necessary. For all practical purposes, the signals are band-limited, and therefore, computation of the transform for a limited interval of scales is usually adequate.

For convenience, the procedure will be started from scale s = 1 and will continue for the increasing values of s, i.e., the analysis will start from high frequencies and proceed towards low frequencies. This first value of s will correspond to the most compressed wavelet. As the value of s is increased, the wavelet will dilate.

The wavelet is placed at the beginning of the signal at the point which corresponds to time = 0. The wavelet function at scale 1 is multiplied by the signal and then integrated over all times. The result of the integration is then multiplied by the constant number 1/sqrt{s}. This multiplication is for energy normalization purposes so that the transformed signal will have the same energy at every scale. The final result is the value of the transformation, (the value of the continuous wavelet transform at time zero and scale s = 1). In other words, it is the value that corresponds to the point tau = 0, s = 1 in the time-scale plane.

The wavelet at scale s = 1 is then shifted towards the right by tau amount to the location t = tau, and the above equation 2.13 is computed to get the transform value at t = tau, s = 1 in the time-frequency plane.

This procedure is repeated until the wavelet reaches the end of the signal. One row of points on the time-scale plane for the scale s = 1 is now completed.

Then, s is increased by a small value. Note that, this is a continuous transform, and therefore, both tau and s must be incremented continuously. However, if this transform needs to be computed by a computer, then both parameters are increased by a sufficiently small step size. This corresponds to sampling the time-scale plane. The above procedure is repeated for every value of s. Every computation for a given value of s fills the corresponding single row of the time-scale plane. When the process is completed for all desired values of s, the CWT of the signal has been calculated. The figures below illustrate the entire process step by step.

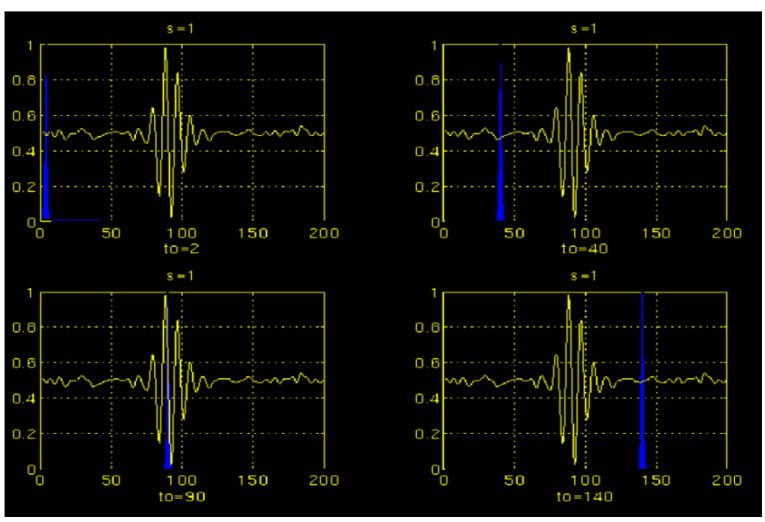

Figure 2.5: Wavelet signals with different values of tau (τ) (Bashier E. 2016) and Mauresh R., 2014)

The signal and the wavelet function are shown for four different values of tau (t). The signal is similar to Figure 2.5. The scale value is 1, corresponding to the lowest scale, or highest frequency. Note how compact it is (the blue window). It should be as narrow as the highest frequency component that exists in the signal. Four distinct locations of the wavelet function are shown in the figure at $t_o = 2$, $t_o = 40$, $t_o = 90$, and $t_o = 140$. At every location, it is multiplied by the signal. Obviously, the product is not zero only where the signal falls in the region of support of the wavelet, and it is zero elsewhere. By shifting the wavelet in time, the signal is localized in time, and by changing the value of s, the signal is localized in scale (frequency).

If the signal has a spectral component that corresponds to the current value of s (which is 1 in this case), the product of the wavelet with the signal at the location where this spectral component exists gives a relatively large value. If the spectral component that corresponds to the current value of s is not present in the signal, the product value will be relatively small, or zero. The signal in Figure 2.5 has spectral components comparable to the window's width at s = 1 around t = 100 ms.The continuous wavelet transform of the signal in Figure 2.5 will yield large values for low scales around time 100 ms, and small values elsewhere. For high scales, on the other hand, the continuous wavelet transform will give large values for almost the entire duration of the signal, since low frequencies exist at all times (Patel M. and Patel R. N., 2012).

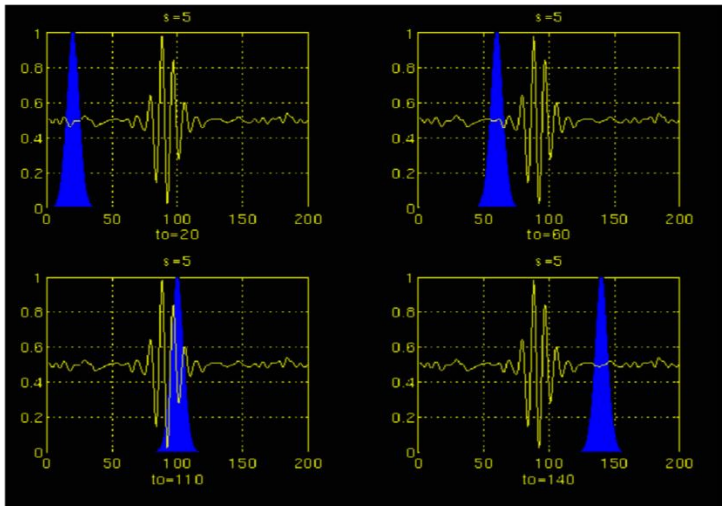

Figure 2.6 Wavelet signals with different values of tau (τ) and increase in window width and scale (decrease in frequency) (Bashier E. 2016) and Mauresh R., 2014)

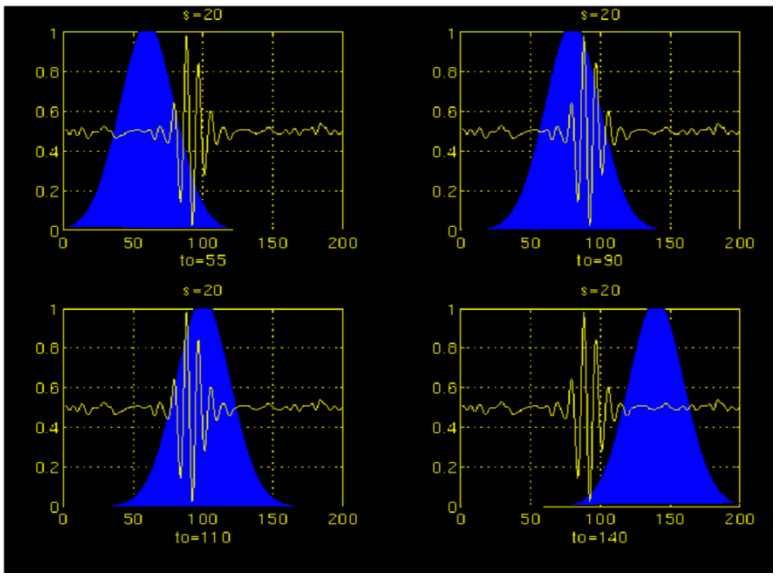

Figure 2.7 Wavelet signals with different values of tau (τ) and wider increase in window width and scale (decrease in frequency) (Bashier E. 2016) and Mauresh R., 2014)

Figures 2.6 and 2.7 illustrate the same process for the scales s = 5 and s = 20, respectively. Note how the window width changes with increasing scale (decreasing frequency). As the window width increases, the transform starts picking up the lower frequency components. But, if the window width is decreased, the transform picks up the higher frequency component of the signal. As a result, for every scale and for every time (interval), one point of the time-scale plane is computed. The computations at one scale construct the rows of the time-scale plane, and the computations at different scales construct the columns of the time-scale plane. For a particular type of fault, fault current whose wave signal can be analyzed using the knowledge of CWT or DWT in terms of high frequency of current magnitude results equal to fault current signal low frequency of current magnitude equal to normal current signal both of the transmission line, all in relationship with s, tau and t_o. Now, let's take a look at an example, and see how the wavelet transform really looks like. Consider the non-stationary signal in Figure 2.7. This is similar to the example given for the STFT, except at different frequencies. As stated on the figure 2.13, the signal is composed of four frequency components at 30 Hz, 20 Hz, 10 Hz and 5 Hz.

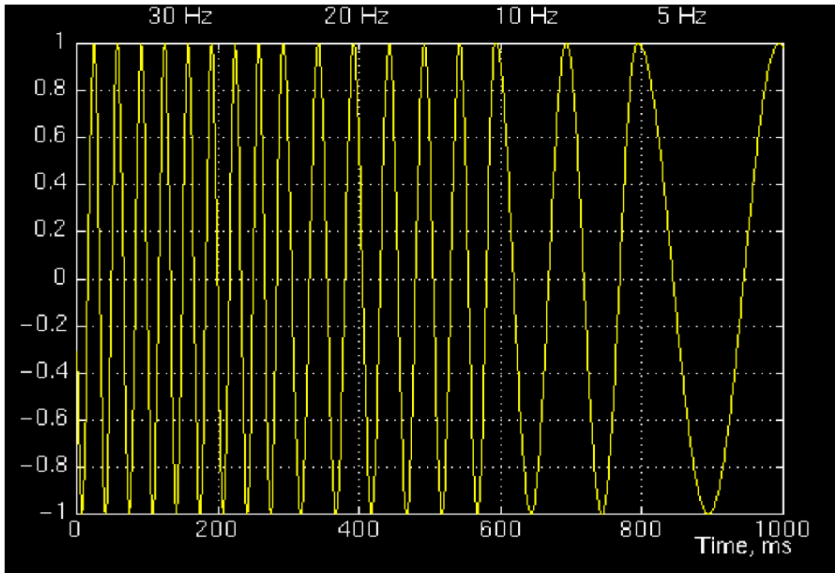

Figure 2.8 Continuous wavelet transform (CWT) of signal (Bashier E. 2016) and Mauresh R., 2014)

Note that the axes are translation and scale, not time and frequency. However, translation is strictly related to time, since it indicates where the mother wavelet is located. The translation of the mother wavelet can be thought of as the time elapsed since t = 0. The scale, however, has a whole different story. Remember that the scale parameter s in equation 2.13 is actually the inverse of frequency. In other words, whatever we said about the properties of the wavelet transform regarding the frequency resolution, inverse of it will appear on the figures showing the WT of the time-domain signal.

Note that in Figure 2.7 that smaller scales correspond to higher frequencies, i.e., frequency decreases as scale increases, therefore, that portion of the graph with scales around zero, actually correspond to highest frequencies in the analysis, and that with high scales correspond to lowest frequencies. Remember that the signal had 30 Hz (highest frequency) components first, and this appears at the lowest scale

at translations of 0 to 30. Then followed by the 20Hz component, second highest frequency and so on. The 5Hz component appears at the end of the translation axis and at higher scales (lower frequencies).

Now, remember that, these resolution properties unlike the STFT which has a constant resolution at all times and frequencies, the WT has a good time and poor frequency resolution at high frequencies, and good frequency and poor time resolution at low frequencies. Figure 2.7shows the same WT in Figure 2.6 from another angle to better illustrate the resolution properties: In Figure 2.7, lower scales (higher frequencies) have better scale resolution (narrower in scale, which means that it is less ambiguous what the exact value of the scale) which correspond to poorer frequency resolution. Similarly, higher scales have scale frequency resolution (wider support in scale, which means it is more ambitious what the exact value of the scale is), which correspond to better frequency resolution of lower frequencies.

The axes in Figure 2.7 and 2.8 are normalized and should be evaluated accordingly. Roughly speaking the 100 points in the translation axis correspond to 1000 ms, and the 150 points on the scale axis correspond to a frequency band of 40 Hz (the numbers on the translation and scale axis do not correspond to seconds and Hz, respectively, they are just the number of samples in the computation).

2.6.5 Time and Frequency Resolutions

In this section, we will take a closer look at the resolution properties of the wavelet transform. Remember that the resolution problem ones of the drawbacks or limitations STFT, thus WT is preferred.

The illustration in Figure 2.9 is commonly used to explain how time and frequency resolutions should be interpreted. Every box in Figure 2.9 corresponds to a value of the wavelet transform in the time-frequency plane. Note that boxes have a certain non-zero area, which implies that the value of a particular point in the time-frequency plane cannot be known. All the points in the time-frequency plane that falls into a box are represented by one value of the WT.

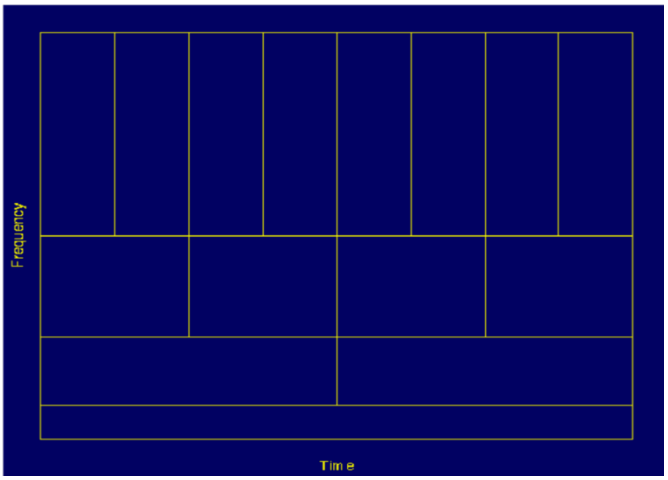

Figure 2.9: Time – Frequency relationship graph (Bashier E. 2016) and Mauresh R., 2014)

Taking a closer look at Figure 2.9, first, we will notice that, although the widths and heights of the boxes change, the area is constant. That is each box represents an equal portion of the time-frequency plane, but giving different proportions to time and frequency. Note that at low frequencies, the height of the boxes are shorter (which corresponds to better frequency resolutions, since there is less ambiguity regarding the value of the exact frequency), but their widths are longer (which correspond to poor time resolution, since there is more ambiguity regarding the value of the exact time). At higher frequencies the width of the boxes decreases, i.e., the time resolution

gets better, and the heights of the boxes increase, (the frequency resolution gets poorer).

Before we conclude this section, it is important to say how the partition looks like in the case of STFT. Recall that in STFT the time and frequency resolutions are determined by the width of the analysis window, which is selected once for the entire analysis, (both time and frequency resolutions are constant). Therefore the time-frequency plane consists of squares in the STFT case.

Regardless of the dimensions of the boxes, the areas of all boxes, both in STFT and WT, are the same. In summary, the area of a box is fixed for each window function (STFT) or mother wavelet (CWT), whereas different windows or mother wavelets can result in different areas. However, all areas are lower bounded by 1/4pi. That is, we cannot reduce the areas of the boxes as much as we want due to the Heisenberg's uncertainty principle. On the other hand, for a given mother wavelet the, dimensions of the boxes can be changed, while the area remain the same. This is exactly what wavelet transform does (Patel M. and Patel R. N., 2012).

Dhanashri D. et al (2016) used DWT to detect fault on the transmission line and ANN to classify the faults. In their work, a fault was introduced between bus 4 and 1 ranged at a distance of 0.25meters at equal four locations and with inception angle of 0, 45, and 90 degrees. They simulated a power system using PSCAD, ATP and MATLAB/SIMULINK and got results. Their results were compared with that of 9 – bus IEEE network in PSCAD, ATP and MATLAB/SIMULINK.

In their work, the generated voltage and current were recorded after simulation in PSCAD and then decomposed in Matlab using DWT in various frequency bands

ranges. They also calculated various energy levels of the DWT voltage and current output results and used them as inputs to the ANN as classifier.

All the fault cases were shown in their work, but with concentration to the fault case with distance 0.5m. Finally, the faults cases were classified using ANN and found out that between the three softwares used, PSCAD gave clearer and better classified results with ANN than other softwares.

Similar to Dhanashri's work, Prakash K. et al (2017) simulated a 230KV, 50Hz power system transmission line with Matlab/Simulink and used WT to decompose the voltage and current signal obtained from it. The WT detects the instant faults based on filtering them through a set of low – pass and high – pass filters. The voltage and current signals obtained after the filtering for A – B and ABCG faults clearly indicate that fault has been detected as compared with the standard fault characteristics and normal condition.

Azah M et al (2018) used continuous wavelet transform (CWT) and S – Transform for fault detection of multiple power quality disturbance, incipient fault and prediction of voltage sag sources that originates from u investpstream or downstream. According to their analysis, S – Transform proved very effective in analysis and detection of the multiple power quality and incipient faults.

Bilal M. and Umar S. used only current signal as input data for their investigation of unsymmetrical and three phase fault on the transmission line using DWT. They used five (5) level decompositions and obtained its approximation and detail coefficient. Their results show that when the system is working under normal operating condition, the normalized values are less than the faulty threshold signal values.

In their work, Preethi M. and Manik H. used a positive sequence component as input data to DWT. The maximum coefficients of the positive sequence current obtained after simulation of faults from all buses were compared with that of normal condition in order to detect the faulty bus on the line. However, the algorithm was able to detect the fault with high accuracy.

2.7 Review of Related Literature

According to Abdelsalam, (2008), Signal processing method is divided into time domain, frequency domain and time-frequency domain methods. Where, statistical analysis and signal derivative belong to time domain method.

2.7.1 Fourier Transform (FT)

Fourier transform (FT) is the transformation technique that transforms signals (electrical signals of current or voltage) from the continuous time domain to its corresponding frequency domain and vice versa. He continued that, Fourier transformation is applied for periodic and non – periodic signals. Fourier transform (FT) belongs to frequency domain approach while short time Fourier transform, wavelet transform, filter bank, mother wavelet selection and wavelet details selection belong to time-frequency domain method. Fourier transform is grouped into continuous time Fourier transform (CTFT), discrete time Fourier transform (DTFT) and discrete Fourier transform (DFT). An algorithm of Fourier transform called Fast Fourier Transform was developed for fast operation of Fourier transformation and covered up some limitations of FT.

In his work Mohammed (2013) explained that Fourier application breaks down signals into constituent sinusoids of different frequencies and infinite duration. He continued that, Fourier transform can be applied mathematically in transforming the output signal from time domain to frequency domain. The short time Fourier Transform maps out small section of signal into two dimensional functions of time and frequency. The STFT and FT has only a little difference between them, which is that, the STFT signal is divided into large number of small segments which can be assumed to be stationary. Thus, a window function "w" is chosen of the segment. The width of the window must be equal to the segment of the signal selected.

Sadiku and Alexander (2009), stated that Fourier transform helps us to extend the frequency spectrum to non-periodic functions. It assumed a non-periodic function to be periodic with an infinite period, represent it as integrally as Fourier series integrally a periodic function. FT transforms a non-periodic in time domain into the frequency domain even if the function is non-periodic. He continues that while Laplace Transform handles circuits with inputs for time $t > 0$, Fourier transform handles circuits with input for time $t < 0$ and $t > 0$.

Abdelsalem (2008) employed the use of Fourier transform in diagnosing fault on the transmission line. In their work, they stated that Fourier Transform (FT) and Short-Time Fourier Transform (STFT) are among most widely used signal processing methods for diagnosis of fault in the power system. They continued that only frequency components of the wave form on the transmission line would be obtained when Fourier transform is applied. He explained that FT is the most popular transformation that can be applied to travelling wave signals on the transmission line to obtain their frequency components appearing in the fault signal. He continued that FT and its inverse give a one – to – one relationship between the time domain x(t)

and frequency domain x(w). Given a signal I(t), the FT is frequency domain (FT)(w) is defined as equations (2.14) and (2.15).

$$\mathrm{FT}\ (\mathrm{w}) = \int_{-\infty}^{\infty} \mathrm{I}(\mathrm{t}).\, \mathrm{e}^{-\mathrm{jwt}}\, \mathrm{dt} \tag{2.14}$$

Where w is the continuous frequency variable while the discrete form of FT (w) is given as

$$\mathrm{DFT}\ (\mathrm{K}) = \frac{1}{\mathrm{N}} \sum_{\mathrm{n}=1}^{\mathrm{N}} \mathrm{I}\ [\mathrm{n}].\, \mathrm{e}^{\frac{-\mathrm{j}\pi\mathrm{kn}}{\mathrm{N}}} \tag{2.15}$$

2.7.2 Discrete Fourier Transform (DFT)

Gerard, (2003), applied the DFT for online computation of the harmonics of the symmetrical components of the line to line voltages and current information. Their work measured the harmonics of the symmetrical component of a functional three-phase induction motor.

Gerard also used DFT to extract the voltage and current signal of the harmonic symmetrical component so as to see observe and record the extent of increase in the magnitude of the three-phase current and decrease in the voltage magnitude of the symmetrical components due to the fault occurrence on the transmission line.

Sarath, (2017), employed a form of DFT called N – DFT. The N – DFT is the type of DFT in which the DFT of an entire frequency spectrum is divided by the number of sample N = 400 with Matlab frequency of 20kHz.

Maher, 2010, explained that The DFT provides uniformly spaced samples of the Discrete-Time Fourier Transform (DTFT) and requires N^2 complex multiplies and N(N-1) complex additions. He continued that one can take advantage of the

symmetrical and periodicity of complex exponential $e^{\frac{-j2\pi}{N}}$ and two length $\frac{N}{2}$ DFT of a function that takes less computation than one length N DFT of a function. Maher also explained that FFT saves time of computation and reduces the length of computation than DFT.

According to Prasad et al, 2019, DFT was used for the estimation of three-phase phasors in power system relaying. The estimated magnitude of the fundamental phasors of the three-phase currents using DFT were used as actuating signals for the detection of faults using Fault Detection Unit (FDU).

2.7.3 Fast Fourier Transform Method (FFT)

According to Mohammed (2013), time and frequency domain signals are the most signals in practice and that in many applications of signal processing, the frequency content of the signals contains the most relevant and discrete information. Thus various mathematical transform are used to analyses those signal processes. The transmission line contains the high frequency of current and voltage waves including other parameters like reactance, resistance, capacitance, admittance and conductance which are all used for analyses of the waveform.

In this section, we will consider very particularly the high frequency current and voltage signals that travels along the conductors (cable) of the transmission lines. Fast Fourier Transform (FFT) is among the frequency domain signal methods of analyses of these travelling waves fault signals.

Roshni et al (2012), Rao and Hasabe (2013) adopted the method FFT method and compared it with the Wavelet transform method. They used the FFT to diagnose fault on the transmission lines. In their work, FFT was used to locate fault on a

simple transmission line connected to generating stations and load centers. Their method was able to locate the unbalanced faults (L – G, LL – G, L – L, and LLL) using Matlab/Simulink software.

Mamis and Arkan (2011), applied FFT method for detection, location and analysis of fault on the transmission line. In their work, they used the frequency of first harmonic with travelling wave (TW) technique for the location of fault on the transmission line. They used state space technique to obtain their transient voltage and current waveform after fault. Their wave signal of time was converted into frequency domain using the FFT at 50Hz. Also, the harmonics in the spectrum of their voltage and current signals with high amplitudes are related with the fault location.

Fourier Transform (FT) been a transformation technique that transforms a signal from the continuous – time domain (t – domain) to the corresponding continuous – frequency domain (s – domain) and vice versa can be used for both periodic (time dependent) and non – periodic (time – independent) signals.

Today, FT is used in many different areas including all branches of engineering. Although it is probably the most popular transform being used (especially in electrical engineering), it is not the only one. There are many other transforms that are used quite often by engineers and mathematicians. They include; Hilbert transform, short-time Fourier transform (STFT), Wigner distributions, the Radon Transform, and wavelet transform.

FT and WT are reversible transforms that allows going back and forwarding between the raw and processed (transformed) signals. However, only either of them is

available at any given time. That is no frequency information in the time-domain signal, and no time information is available in the frequency domain signal. The natural question that comes to mind is that, is it necessary to have both the time and the frequency information at the same time?

The answer depends on the particular application and the nature of the signal in hand. Remember that, FT gives the frequency information of the signal, which means that it tells us how much of each frequency exists in the signal, but it does not tell us when (time) these frequency components exist.

$$X(f) = \int_{-\infty}^{\infty} x(t).\, e^{-2j\pi ft}\, dt \tag{2.16}$$

$$x(t) = \int_{-\infty}^{\infty} X(f).\, e^{2j\pi ft}\, df \tag{2.17}$$

Considering the above equations, t stands for time, f stands for frequency, and x represents for the signal at hand in time domain and the X represents the signal in frequency domain. This information is used to distinguish the two representations of the signal. Where equation (2.16) is called the Fourier transform of x(t), and equation (2.17) is called the inverse Fourier transform of X(f).

The signal x(t), is multiplied with an exponential term (see equation 2.17), at some certain frequency f, and then integrated over all times.

$$X_{a_n}(f) = \int_{-\infty}^{\infty} x(t)\cos 2\pi ft\, dt \tag{2.18}$$

$$X_{b_n}(f) = \int_{-\infty}^{\infty} x(t)\sin 2\pi ft\, dt \tag{2.19}$$

Equation (2.18 and 2.19) are cosine of frequency f (real part), and sine of frequency f (imaginary part). Both represent the Fourier Transforms of an electrical alternate current signal. Multiplying the original signal with a complex expression which has

sines and cosines of frequency f, we then integrate the product. If the result of this integration (which is nothing but some sort of infinite summation) is a large value, then we say that: the signal x(t) has a dominant spectral component at frequency "f". This means that, a major portion of this signal is composed of frequency f (there is fault). If the integration result is a small value, then this means that the signal does not have a major frequency component of "f" in it (no fault). If this integration result is zero, then the signal does not contain the frequency "f" at all (not an AC component). The large and small portions of frequency occurred as a result of the presents of fault or no fault on the line respectively.

It is of particular interest here to see how this integration works: The signal is multiplied with the sinusoidal term of frequency "f". If the signal has a high amplitude component of frequency "f", then that component and the sinusoidal term will coincide, and the product of them will give a (relatively) large value. This shows that, the signal "x", has a major frequency component of "f" and if the signal was generated from a faulty transmission line, the large frequency level in it shows that the signal contains information about the faulty conditions of the power system transmission line as will be seen on the FFT Simulink scopes in chapter four.

However, if the signal does not have a frequency component of "f", the product will yield zero, which shows that, the signal does not have a frequency component of "f" and no information about the faulty condition of the line. If the frequency "f", is not a major component of the signal "x(t)", then the product will give a (relatively) small value. This shows that, the frequency component "f" in the signal "x", has small amplitude, in other words, it is not a major component of "x" (RobiP. 2013).

2.7.4 Short Time Fourier Transform (STFT) Assuming the signal is stationary, and the region where the signal can be assumed to be stationary is too small, then, we can look at the signal from narrow windows, narrow enough that the portion of the signal seen from these windows are indeed stationary. This aspect of the Fourier transform is called the Short Time Fourier Transform (STFT).

There is only a minor difference between STFT and FT. In STFT, the signal is divided into small enough segments, where these segments (portions or windows) of the signal can be assumed to be stationary. In view of this, we can choose a window function "w". The width of this window must be equal to the segment of the signal where its stationarity is valid.

This window function is first located to the very beginning of the signal. Assuming the width of the window is "T". That is, the window function is located at t = 0. At this time instant t = 0, the window function will overlap with the first T/2 seconds (taking all time units in seconds). The window function and the signal are then multiplied. By so doing, only the first T/2 seconds of the signal is being chosen, with the appropriate weighting of the window. The result of this transformation is the FT of the first T/2 seconds of the signal. If this portion of the signal is stationary, as it is assumed, then there will be no problem and the obtained result will be a true frequency representation of the first T/2 seconds of the signal (Müslüm A. 2010).

The next step is the shifting of the window (for t1 seconds) to a new location, multiplying with the signal, and taking the FT of the product. This procedure is followed; until the end of the signal is reached by shifting the window with "t1" seconds intervals.

The following definition of the STFT summarizes all the above explanations in one line:

$$\mathrm{STFT}_{X}^{(\omega)}(t,f) = \int_{t} [x(t).\omega^{*}(t-t').e^{-2\pi ft}dt \qquad (2.20)$$

Considering the above equation (2.20), x(t) is the signal itself, w(t) is the window function, and * is the complex conjugate. As you can see from the equation (2.20), the STFT of the signal is nothing but the FT of the signal multiplied by a window function. For every t' and f, a new STFT coefficient is computed.

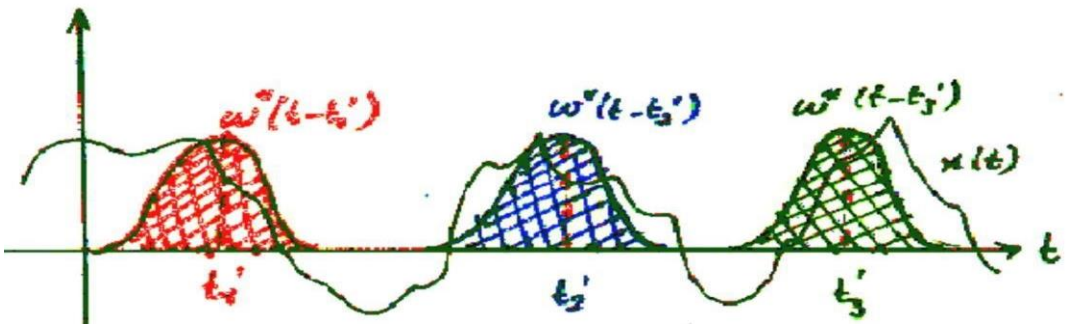

Figure 2.10 The Windowing functions (FFT)Bashier E. (2016) and Mauresh R., (2014)

The figure 2.10 describes windowing functions. The first one starting from the left side shows the window located at $t = t_1'$, the middle one shows $t = t_2'$, and the one extreme right shows the window located at $t = t_3'$. These correspond to the three different FTs at three different times. Therefore, we will obtain a true time-frequency representation (TFR) of the signal.

The problem with STFT is that, when Heisenberg Uncertainty Principle which states that, one cannot know the exact time - frequency representation of a signal (one cannot know the particular spectral components exist at a particular point in time) is

applied to time frequency information of a signal makes STFT not too good for time – frequency analysis. What we need to know is the time intervals in which certain band of frequencies exist, which is a resolution problem. Another drawback is that, the STFT has something to do with the width of the window function that is used. To be technically correct, this width of the window function is known as the support of the window (Rao M. and Hasabe R. P., 2014). Remember that in the FT there is no resolution problem in the frequency domain, i.e., we know exactly what frequencies exist; similarly, we know there is no time resolution problem in the time domain, since we know the value of the signal at every instant of time. Conversely, the time resolution in the FT, and the frequency resolution in the time domain are zero, since we have no information about them. What gives the perfect frequency resolution in the FT is the fact that the window used in the FT is its kernel, the $e^{-j\omega t}$ function, which lasts at all times from minus infinity to plus infinity.

In STFT, the window is of finite length, and covers only a portion of the signal, which causes the frequency resolution to get poorer. What I mean by getting poorer is that, we no longer know the exact frequency components that exist in the signal, but we only know a band of frequencies that exist:

In FT, the kernel function, allows us to obtain perfect frequency resolution, because the kernel itself is a window of infinite length. In STFT is window is of finite length, and we no longer have perfect frequency resolution. Making the length of the window in the STFT infinite, just like as it is in the FT, to get perfect frequency resolution, therefore, we will lose all the time information, and end up with the FT instead of STFT. If we use a window of infinite length, we get the FT, which gives perfect frequency resolution, but no time information. Furthermore, in order to

obtain the stationarity, there is need to have a very short window, in which the signal is stationary. The narrower we make the window, the better the time resolution, and better the assumption of stationarity, but poorer the frequency resolution. Thus, narrow window gives good time resolution, poor frequency resolution. While, wide window gives good frequency resolution, poor time resolution. The graph of figure 2.11 below explains the effects of this issue of poor resolution. The window function is simply a Gaussian one in the form below:

$$w(t) = e^{(-a*(\frac{t^2}{2}))} \tag{2.21}$$

Where ‘a’ determines the length of the window, and t is the time. The graph shows four window functions of varying regions of support, determined by the value of ‘a’. The example below is computed with the value, a = 0.001.

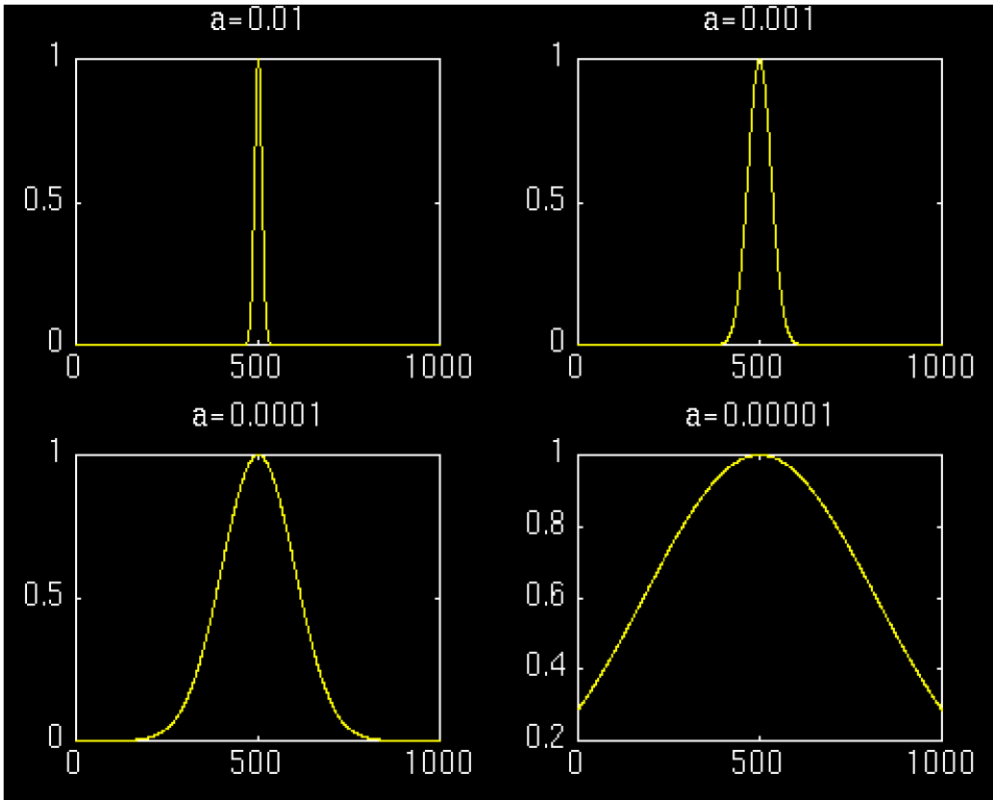

Figure 2.11 Gaussian window function show that wider window is better that narrow window (FFT)Bashier E. (2016) and Mauresh R., (2014)

Note that the four peaks are well separated from each other in time. Also note that, in frequency domain, every peak covers a range of frequencies, instead of a single frequency value. Now let's make the window wider, and look at the third window (the second one was already shown in the first example).

Notice that, the peaks are not well separated from each other in time, unlike the previous case, however, in frequency domain the resolution is much better.

These figures above have illustrated the implicit problem of resolution of the STFT. Anyone who would like to use STFT is faced with this problem of resolution. What kind of a window to use? Narrow windows give good time resolution, but poor frequency resolution. Wide windows give good frequency resolution, but poor time

resolution; furthermore, wide windows may violate the condition of stationarity. The problem, of course, is a result of choosing a window function, once and for all, and uses that window in the entire analysis. The answer, of course, is application dependent: If the frequency components are well separated from each other in the original signal, then we may sacrifice some frequency resolution and go for good time resolution, since the spectral components are already well separated from each other.

However, the Wavelet transform (WT) solves the dilemma of resolution to a certain extent, as we will see in the next part. But, before looking WT, let us consider the application of another aspect of FT called Fast Fourier Transform (FFT)Bashier E. (2016) and Mauresh R., (2014).

Most of the signals in practice are Time-Domain (TD) signals in their raw format. That is, whatever that signal is measuring, is a function of time. In other words, when we plot the signal, one of the axes mostly horizontal is time (independent variable), and the other mostly vertical (dependent variable) is usually the amplitude. When we plot time-domain signals, we obtain a time-amplitude representation of the signal. This representation is not always the best representation of the signal for most signal processing related applications. In many cases, the most distinguished information is hidden in the frequency content of the signal. The frequency spectrum of a signal is basically the frequency components (spectral components) of that signal. The frequency spectrum of a signal shows what frequencies exist in the signal (Robi P., 2013).

A typical example is the biological signals, such as the ECG signal (Electrocardiography, graphical recording of heart's electrical activity). Here, the cardiologists already know the typical shape of a healthy ECG signal. Any

significant deviation from that shape is usually considered to be a symptom of a pathological condition.

This pathological condition, however, may not always be quite obvious in the original time domain signal. Cardiologists usually use the time-domain ECG signals which are recorded on strip-charts to analyze ECG signals. Recently, new computerized ECG recorders/analyzers which utilize the frequency information to decide whether a pathological condition exists have been in use. Though, a pathological condition can sometimes be diagnosed more easily when the content of the signal is analyzed. This, of course, is only one simple example why frequency content might be useful.

In their work, Zhan W. et al, (2018) studied the performance of the Fast Fourier Transform (FFT) as a fault detection method and compared its result with that of wavelet transform by using synthetic waveforms (waveforms created by combining measured waveform of normal noise from inverter in DC PV array with that of arcing events). They used a composite signal with duration of one second and which is synthesized by the combination of inverter noise and fault signal sampled at the rate of 1MHz frequency (1/T).

Zhan et al later selected the 7th decomposition signal which covered a frequency band of 3.9 KHz – 7.8 KHz from the wavelet transform analyzed. The analysis shows that the temporal waveforms for the selected frequency band clearly indicate a timing extinction of the arc.

Dibya D. and Sinha A. K. (2016) developed an algorithm for tasks of detection, classification and location faults on the transmission lines. In their work, they compared the performance of FFT and WT based on the three tasks. They

concentrated on only three phase short circuit fault using current as input parameter and a minimum resolution of Daubehies 8 wavelet transform to obtain the decomposed current signal.

Two test cases such as L – G and LLL faults were used by Dibya and Sinha (2016) to compare the performance of DFT and FFT. In their result analysis, they concluded that DWT technique gave a better result in the case of fault classification and location. But they stated that the DFT technique provided a better performance in predicting fault distance when L – G and LLL faults were simulated. However, they stated that both techniques were able to detect the fault due to increase in current magnitude.

Two flowcharts were developed, one for DFT and other for DWT applications. Each flowchart explains the detection, classification and location processes.

Kumarraja A. et al (2019) did a similar work to that of Dibya, but a little difference is that instead of using normal line current signal, they used absolute current phasor obtained through fundamental phasor technique-based approach. Another difference is that they used two data samples or parameters which are voltage and current signals measured from the impedance relay. These signals were used as inputs to the DFT and DWT algorithms. After the simulations, the absolute current phasors provided clear discrimination between faulted phase and healthy phase. Zero sequence absolute current phasors show the involvement of ground during the condition of fault.

They further validated their work by testing them on a 400KV, 50Hz two terminal bus transmission line by considering different events such as variation of resistance, distance, inception angle and type of fault using Matlab/Simulink. They conclude that the algorithms are correct.

Another work similar to that of Dibya is that of Sarath M. and Reji P. (2017), here they identified and classified three types of unsymmetrical (L-G, LL-G, L-L) faults and one symmetrical (LLL) fault by analyzing the frequency of the equivalent voltage phasor angle and phasor estimation of the buses in power system. Their work was tested and validated using IEEE 9 – bus network in Matlab.

2.8: Summary of review of related literature

I. The following gaps were noted from the related research of other authors reviewed in this research work.

II. None of the authors of the reviewed related literatures mentioned above used real data such as pre-fault and faulty voltage or current from any existing transmission line station. Instead they used simulated data obtained from a modeled transmission line using Matlab/Simulink. In this work, real pre-fault and faulty voltage and current data obtained from Onitsha transmission station were used.

III. DFT and FFT algorithm was not modeled with Matlab/Simulink by the same authors in their work, but this research work did use Matlab/Simulink R2017a.

IV. Percentage error between the mathematical and simulation approaches was done in this research work, but none of the authors used mathematical approach or finding percentage error between simulation and any other method used.

V. Sarath M. and Reji P., 2017 employed a version of DFT called N – DFT as a type of DFT in which the DFT of an entire frequency spectrum is divided by the number of samples N = 400 with Matlab sampling frequency of 20kHz. However, this research used the same 20kHz samplying frequency but 6400

number of samples, N for each of voltage and current parameters. This enables the achievement of clear and standard satisfactory results.

VI. Maher considered a DFT 8 (N = 8) and 16 (N = 16) – point twindle butterfly flow variable computation to demonstrate in general that DFT works only when number of samples N is a multiple of two (2). But in this research, 4 (N = 4) – point representing either of voltage or current three – phases (a, b and c) with an arbitrary 0 added to them making them 4 samples to represent the three – phases of a real power system transmission line.

VII. In all the reviewd literatures that used both DFT and FFT, a comparative accesment of DFT and FFT application in their work was not done, but this research did and concluded that DFT is better than FFT interms of clearity, ambiguity and comformity of result with a standard one.

CHAPTERTHREE

METHODOLOGY

3.1 Methodology

In this chapter, two signal processing techniques, the Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) are presented and were used for the development of fault detection algorithms for power system transmission lines.

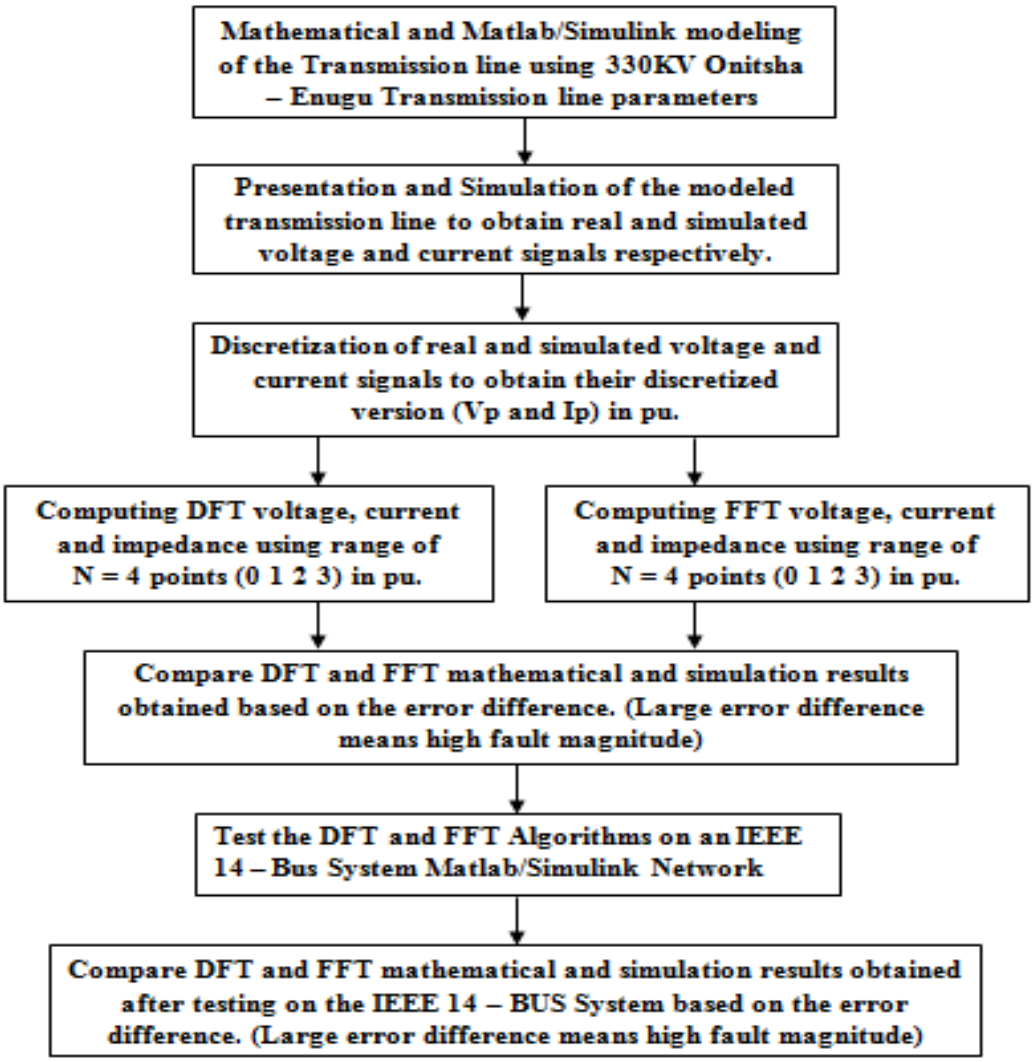

Figure 3.1: Block diagram of the methodological procedure

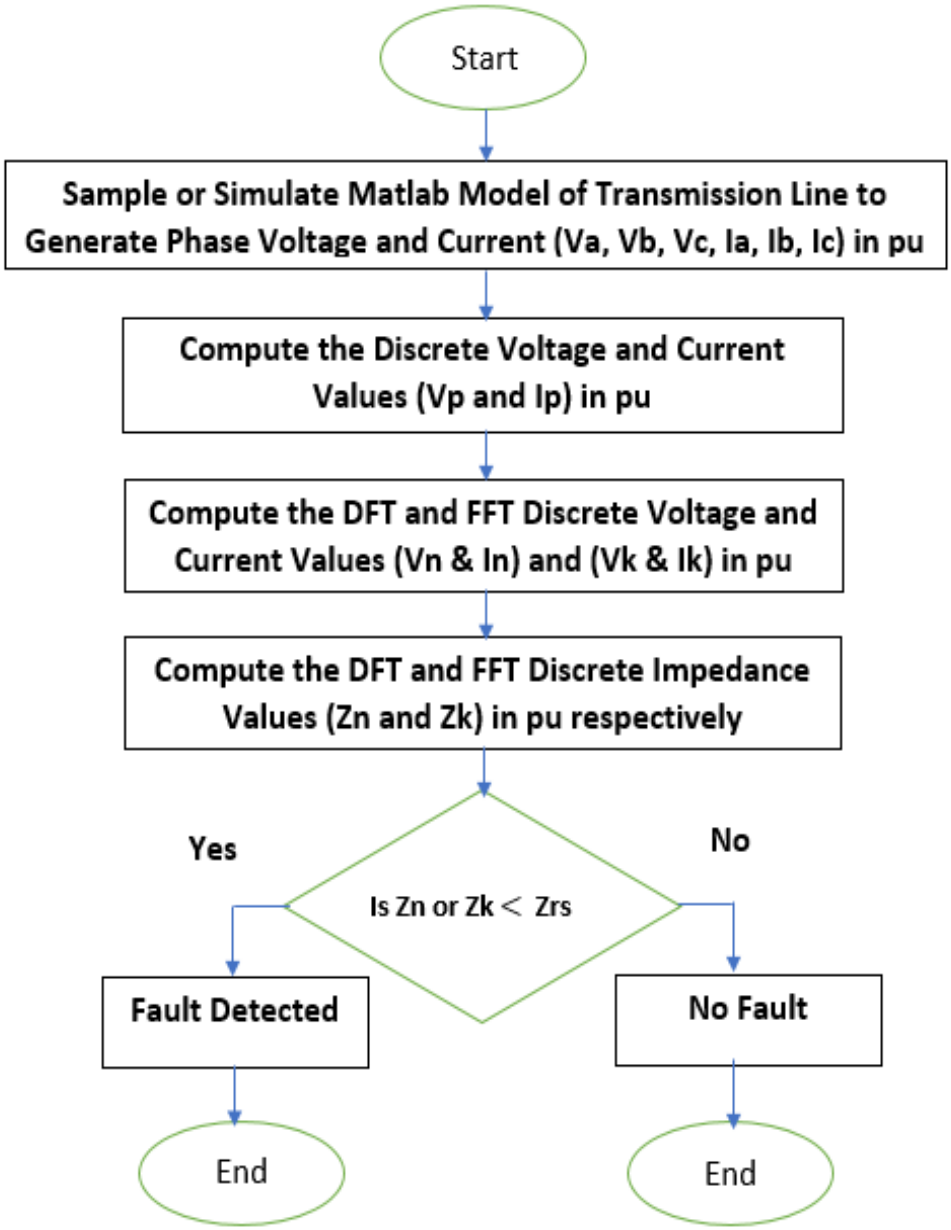

Figure 3.2: Flowchart diagram for the detection algorithm

Figures 3.1 and 3.2 illustrate the methodological process followed to achieve the objectives of this research work. Considering figures 3.1 and 3.2, the methodology is divided into two parts; mathematical (analytical) and simulation (computer) approaches.

Table 3.1 Per Unit Real Pre-fault Voltage and Current values from Onitsha transmission station

S/No	Input Vector (Per Unit Phase Voltage and Current)						Fault Type/Zone
	V_{apf}	V_{bpf}	V_{cpf}	I_{apf}	I_{bpf}	I_{cpf}	
1	1.4652	1.3250	1.3492	0.6586	0.2596	0.0000	A – G Zone 2
2	1.4841	1.3598	1.3311	0.3032	0.4212	0.3318	B – G Zone 1
3	1.4818	1.3576	1.3258	0.2977	0.3432	0.3262	C – G Zone 2
4	1.0542	1.2581	1.3215	0.0000	0.0000	0.0000	A – B Zone 1
5	1.8932	1.0879	1.1250	0.0000	0.0000	0.6628	B – C Zone 2
6	1.0231	1.3215	1.3520	0.0000	0.0000	0.0000	A – C Zone 3
7	0.4700	0.4600	0.4700	0.2200	0.1050	0.1250	AB – G Zone 2
8	0.4200	0.5000	0.4600	0.1480	0.1450	0.1650	BC – G Zone 1
9	0.4800	0.4300	0.4000	0.2000	0.1480	0.1450	AC – G Zone 1
10	0.4000	0.4500	0.4900	0.2200	0.1400	0.1490	ABC Zone1

Table 3.2 Per Unit Real Fault Voltage and Current values from Onitsha transmission station

S/No	Input Vector (Per Unit Phase Voltage and Current)						Fault Type/Zone
	V_{af}	V_{bf}	V_{cf}	I_{af}	I_{bf}	I_{cf}	
1	1.4562	1.3311	1.3273	0.7279	0.4355	0.4982	A – G Zone 2
2	1.4333	0.9091	1.2795	0.6292	5.1247	0.1716	B – G Zone 1
3	1.4636	0.8371	1.2803	0.3761	0.2804	9.5717	C – G Zone 2
4	2.5433	1.3501	1.7540	0.0000	0.8234	0.0000	A – B Zone 1
5	1.9265	1.1129	1.1523	0.0000	0.0000	0.0000	B – C Zone 2
6	2.2540	1.0251	1.6345	0.0000	0.0000	0.7315	A – C Zone 3
7	1.5600	1.1200	1.4720	2.5600	3.1050	0.1250	AB – G Zone 2
8	1.4300	1.5600	1.5610	0.4480	4.0450	3.1450	BC – G Zone 1
9	1.4402	1.3320	1.4020	5.4200	0.1480	4.2650	AC – G Zone 1
10	0.5000	0.5500	0.6900	1.9000	0.2400	0.2590	ABC Zone1

Two types of samples are to be used; they are pre-fault and fault samples which are shown in table 3.1 and 3.2. They were obtained as real time signals values in pu and were computed using equations 3.1 to 3.6 below.

$$v_a = V_p \sin(\theta + \emptyset) \quad (3.1)$$

$$v_b = V_p \sin(\theta + \emptyset) \quad (3.2)$$

$$v_C = V_p \sin(\theta + \emptyset) \quad (3.3)$$

$$i_a = I_p \sin(\theta + \varphi) \quad (3.4)$$

$$i_b = I_p \sin(\theta + \varphi) \quad (3.5)$$

$$i_C = I_p \sin(\theta + \varphi) \quad (3.6)$$

According to Sadiku and Alexandra, Fourier Series (FS) of a function v(t) or i(t) is a presentation that resolves v(t) or i(t) into a frequency component of DC and AC consisting of an infinite series of harmonic sinusoids.

Though, there are no much harmonics on the transmission line, it is assumed negligible.

Thus, Fourier Series of a function v(t) or i(t) is given as;

$$v(t) = V_0 + V_1 + V_2 + \ldots + V_n \quad (3.7)$$

$$i(t) = I_0 + I_1 + I_2 + \ldots + I_n \quad (3.8)$$

This implies,

$$v(t) = V_0 + \sum_{n=1\infty}^{\infty} |V_n| \cos(nw_0 t + \emptyset_n) \quad (3.9)$$

$$i(t) = I_0 + \sum_{n=1\infty}^{\infty} |I_n| \cos(nw_0 t + \psi_n) \text{ (Sadiku and Alexandra)} \quad (3.10)$$

Where V_0 and I_0 are DC components of the Fourier Series, $|V_n|$ and $|I_n|$ are sequence AC components of the Fourier Series and are given as;

$$n = 1, 2, 3, \ldots \tag{3.11}$$

$$\omega_0 = 2\pi f \tag{3.12}$$

$$f = 50\text{Hz} \tag{3.13}$$

The Fourier Series (FS) representation of AC signal components of the power supply on the transmission line are sets of series connected sinusoidal sources with each source having its own amplitude and frequency which are evidence of fault detection that is to say that, when the amplitude and frequency of current I_n during fault occurrence is far greater than its values before the fault, it then means that they are evidence that fault has occurred on the transmission line and has been detected (Sadiku M. N. O and Alexandra C. K., 2009).

However, Fourier Transformation (FT) can be performed on the voltage or current signals to obtain clearer and closer value that corresponds to the fault characteristics on the transmission line. These characteristics include increase in current magnitude and reduction in voltage magnitude during and after fault occurrence.

Therefore, Fourier Transform of the functions or voltage and current signals is given as;

$$V(\omega) = \mathcal{F}[(V(t)] = \int_{-\infty}^{\infty} V(t)e^{-j\omega t}\,dt \tag{3.14}$$

$$I(\omega) = \mathcal{F}[(I(t)] = \int_{-\infty}^{\infty} I(t)e^{-j\omega t}\,dt \tag{3.15}$$

Where for AC signals, $n \geq 1$, while for the DC signals $n = 0$, $j = \sqrt{-1}$ (Sadiku and Alexandra).

The above Fourier series (FS) and Transform (FT) are majorly used when a function or signal is continuous (infinite), moving from n = 0 to ∞ or -∞ to ∞ for Fourier series and Transform respectively.

But, for a non –continuously (finite) signal such as three phase pre-fault and fault voltage and current of a transmission line, Discrete Fourier Transform (DFT) is used or its faster algorithm called Fast Fourier Transform (FFT).

The DFT of the signal v(t) and i(t) is given by;

$$V_n = \sum_{n=1}^{N-1} v(t)\, e^{\frac{-2\pi n}{N}} \tag{3.16}$$

$$I_n = \sum_{n=1}^{N-1} i(t)\, e^{\frac{-2\pi n}{N}} \tag{3.17}$$

$$n = 0, 1, 2, 3, \ldots, N - 1 \tag{3.18}$$

N = Number of samples

T = Time interval between the two consecutive sampling data.

f = Frequency or sampling rate of two consecutive samples.

The FFT of a signal v(t) and i(t) is given by;

$$V_k = \sum_{n=0}^{\frac{N}{2}-1} v(2n)\, W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} v(2n-1)\, W_{\frac{N}{2}}^{nk} \tag{3.19}$$

$$I_k = \sum_{n=0}^{\frac{N}{2}-1} i(2n)\, W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} i(2n-1)\, W_{\frac{N}{2}}^{nk} \tag{3.20}$$

3.2 Mathematical Development of Equations for DFT Fault Detection Model

3.2.1 Computation of DFT Pre – fault Voltage

The three - phase pre-fault and fault values of voltages and currents in Table 3.1 and 3.2 are discretized using equations (3.21) and (3.22) respectively.

Table 3.1 and 3.2 (TCN Onitsha, 2018) show the pre-fault and fault voltage and current values in per unit obtained from Onitsha transmission line station. The values are referred to as calculated values, while the ones obtained during simulation are referred to as simulated values.

$$v_p(n) = \frac{2}{3}[\, v_a + v_b(n)e^{\frac{j2\pi}{N}} + v_c(n)e^{\frac{-j2\pi}{N}} \,] \tag{3.21}$$

$$i_p(n) = \frac{2}{3}[\, i_a + i_b(n)e^{\frac{j2\pi}{N}} + i_c(n)e^{\frac{-j2\pi}{N}} \,] \tag{3.22}$$

Where, equations (3.21) and (3.22) are referred to as discrete equations for the determination of DFT discrete phase quantities $v_p(n)$and $i_p(n)$(Capolino G. A. et al, 2003).

But, since N = 3 samples (V_a, V_b and V_C or I_a, I_b and I_C), n = 1 for AC.

Therefore, substituting the values of N and the pre – fault three phase values, we have;

$$v_p(n) = \frac{2}{3}[\, 0.4 + 0.45e^{\frac{j2\pi}{3}} + 0.49e^{\frac{-j2\pi}{3}} \,] = 0.8932\text{pu} \tag{3.23}$$

$$i_p(n) = \frac{2}{3}[\, 0.22 + 0.14e^{\frac{j2\pi}{3}} + 0.149e^{\frac{-j2\pi}{3}} \,] = 0.3393\text{pu} \tag{3.24}$$

Where $e^{\frac{j2\pi}{3}} = \cos\left(\frac{2\pi}{3}\right) + j\sin\left(\frac{2\pi}{3}\right)$ (3.25)

and

$$e^{-\frac{j2\pi}{3}} = \cos\left(\frac{2\pi}{3}\right) - j\sin\left(\frac{2\pi}{3}\right) \quad (3.26)$$

The discretized phase values are used for the computation of the sinusoidal phasors v(t) and i(t). Since the DFT is designed to be computed using N equals to multiple of 2, such as 2, 4, 8, 16 etc, but our real time three phase pre-fteault and fault values are three (N = 3). Therefore, computation of discrete DFT voltage and current is performed using N = 4 finite points (n = 0, 1, 2, 3). Thus, the sinusoidal phasors are determined for the 4 points using equations (3.27) and (3.28).

$$v_n(t) = |V_p| \sin(nw_0 t + \emptyset_n) \quad (3.27)$$

$$i_n(t) = |I_p| \sin(nw_0 t + \psi_n)$$ (Sadiku M. N. O and Alexandra C. K., 2009) (3.28)

Assuming$\emptyset = 45^0, 50^0, 55^0, 60^0$,$\varphi = 45^0, 50^0, 55^0, 60^0$, $\theta = \omega = 2\pi f$,

$f = 50\text{Hz}$, V_0 and I_0 = 0 for DC, $t = 1\text{sec}$ (Instantaneous) and $n = 0, 1, 2, 3$

$$v_0(t) = |0.8932| \sin(2\pi x50x1 + 45) = 0.7785\text{pu} \quad (3.29)$$

$$v_1(t) = |0.8932| \sin(2\pi x50x1 + 50) = -0.1991\text{pu} \quad (3.30)$$

$$v_2(t) = |0.8932| \sin(2\pi x50x1 + 55) = -0.8914\text{pu} \quad (3.31)$$

$$v_3(t) = |0.8932| \sin(2\pi x50x1 + 60) = -0.3067\text{pu} \quad (3.32)$$

Therefore,

DFT of $v_n(t)$ is given by;

$$V_n[N = 0\ 1\ 2\ 3]^T = \begin{pmatrix} 1 & 1 & 1 & 1 \ldots & 1 \\ 1 & W & W2 & W3\ldots & WN\text{-}1 \\ 1 & W2 & W4 & W6\ldots & WN\text{-}2 \\ 1 & WN\text{-}1 & WN\text{-}2 & WN\text{-}3\ldots & W \end{pmatrix} [v_n(t)] = [v_0(t)\ v_1(t)\ v_2(t)\ v_3(t)]^T \tag{3.33}$$

Equations (3.33) and (3.39) represents the matrix computational process of DFTof the sinusoidal voltage signals of equations (3.29) to (3.32).

$$W = \exp\frac{(-j2\pi)}{N} \text{ and } W = W^{2N} = 1 \tag{3.34}$$

For a 4 – point DFT application, where N = 4

Thus,

$$W = \exp\frac{(-j2\pi)}{N} = \exp\frac{(-j2\pi)}{4} = \exp\frac{(-j\pi)}{2} = a \tag{3.35}$$

$$a2 = -j, \tag{3.36}$$

$$a3 = -ja = -a^* \tag{3.37}$$

$$a4 = -1 \tag{3.38}$$

$$V_n[N = 0\ 1\ 2\ 3]^T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} [v_n(t)] = [v_0(t)\ v_1(t)\ v_2(t)\ v_3(t)]^T \tag{3.39}$$

$$v_n(t) = [v_0(t), v_1(t), v_2(t), v_3(t)]^T = [0.7785, -0.1995, -0.8914, -0.3067]^T \tag{3.40}$$

Evaluating equation (3.34), we have;

$$Vn\ [N = 0, 1, 2, 3]^T = [-0.6191, 1.6733, 0.3933, 1.7460]^T \quad (3.41)$$

Where, equation (3.41) is the output result of mathematical computation of DFT pre-fault voltage.

3.2.2 Mathematical Computation of DFT Pre – fault Current

Using equations (3.22, 3.24, 3.25, 3.26, 3.28, 3.35, 3.36, 3.37, 3.38, and 3.39), mathematical computation of DFT pre – fault current is performed with I_p =.0.3392

$$i_0(t) = |0.3392| \sin(2\pi x50x1 + 45) = 0.2956pu \quad (3.42)$$

$$i_1(t) = |0.3392| \sin(2\pi x50x1 + 50) = -0.0076pu \quad (3.43)$$

$$i_2(t) = |0.3392| \sin(2\pi x50x1 + 55) = -0.3385pu \quad (3.44)$$

$$i_3(t) = |0.3392| \sin(2\pi x50x1 + 60) = -0.1165pu \quad (3.45)$$

Therefore,

DFT of $i_n(t)$ is given by;

This implies;

$$I_n[N = 0\ 1\ 2\ 3]^T = \begin{pmatrix} 1 & 1 & 1 & 1 \ldots & 1 \\ 1 & W & W2 & W3 \ldots & WN\text{-}1 \\ 1 & W2 & W4 & W6 \ldots & WN\text{-}2 \\ 1 & WN\text{-}1 & WN\text{-}2 & WN\text{-}3 \ldots & W \end{pmatrix} [i_n(t)] = [i0(t)\ i1(t)\ i2(t)\ i3(t)]^T \quad (3.46)$$

$$I_n[N = 0\ 1\ 2\ 3]^T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} t)] = [i0(t)\ i1(t)\ i2(t)\ i3(t)]^T \quad (3.47)$$

$$i_n(t) = [i_0(t)\ i_1(t)\ i_2(t)\ i_3(t)]^T = [0.2956\ -0.0076\ -0.3385\ -0.1165]^T \quad (3.48)$$

Evaluating equation (3.47), we have;

$$I_n\,[N = 0, 1, 2, 3]^T = [-0.1670, 0.6434, 0.0812, 0.6461]^T \quad (3.49)$$

Where, equation (3.49) is the output result of mathematical computation of DFT pre-fault current.

3.2.3 Mathematical Computation of DFT Three Phase Fault Voltage

The **same** procedure is followed to mathematically compute the DFT three phase fault voltage, but the three phase fault voltages to be used are given as;

$V_a = 0.153pu$, $V_b = 0.113pu$, $V_c = 0.098pu$

Evaluating equations (3.33) to (3.39), we obtain that;

$V_p = 0.157pu$

Therefore, using equations (3.32) to (3.37), DFT of $v_n(t)$ is computed and the output result is given as;

$$v_n(t) = [v_0(t), v_1(t), v_2(t), v_3(t)]^T = [0.1368, -0.035, -0.1567, -0.054]^T$$

Thus, DFT three phase fault voltages are given as;

V_n [N = 0, 1, 2, 3]^T = [-0.1087, 0.2941, 0.0691, 0.306.]^T (3.50)

Where, equation (3.50) is the output result of mathematical computation of DFT three phase fault voltage.

3.2.4 Mathematical Computation of DFT Three Phase Fault Current

The sameprocedure is followed to mathematically compute the DFT three phase fault voltage, but the three phase fault voltages to be used are given as;

I_a = 1.9pu, I_b = 0.24pu, I_c = 0.259pu

Evaluating equations (3.46 and (3.47), we obtain that;

Ip = 1.5992pu

Therefore, using equations (3.32) to (3.37), DFT of $i_n(t)$ is computed and the output result is given as;

$i_n(t) = [i_0(t)\ i_1(t)\ i_2(t)\ i_3(t)]^T = [1.3938\text{-}0.3564\text{-}1.5960\text{-}0.5491]^T$

Thus, DFT three phase fault currents are given as;

I_n [N = 0, 1, 2, 3]^T = [-1.1077, 2.9899, 0.7053, 3.1239]^T (3.51)

Where, equation (3.51) is the output result of mathematical computation of DFT three phase fault current.

3.3 Mathematical Development of Equations for FFT Fault Detection

3.3.1 Computation of FFT Pre – fault Voltage

Since we obtained the real time three phase pre-fault and fault values of voltage and current and has computed their discrete versions V_P and I_P in the previous section.

Therefore, applying Butterfly flow process shown in figure 3.3, computation of pre-fault and three phase fault signals using FFT is performed (Capolino G. A. et al 2003)

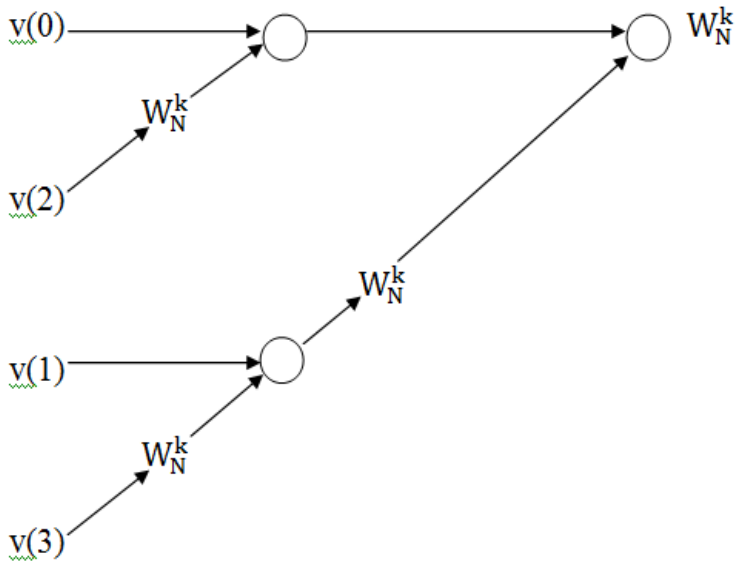

Figure 3.3: Flow Diagram for FFT computation of both Pre-fault and Three Phase Fault conditions at 4 – point (N = 4)

The flow diagram is interpreted using equation (3.52).

$$V_N(k) = \left[v(0) + v(2)W_N^{2k}\right] + W_N^k[v(1) + v(3)W_N^{2k}] \quad (3.52)$$

Where k = 0, 1, 2, 3 points and n = 0, 1, 2, …, N – 1 finite point.

Where, $W_4^0 = 1$, $W_4^1 = -j$, $W_4^2 = -1$ and $W_4^3 = j$

$$V_4(0) = [v(0) + v(2)W_4^0] + W_4^0\,[v(1) + (v(3)\,W_4^0)] \quad (3.53)$$

$$= [0.7785 + (-0.8914)1] + 1[-0.1995 + (-0.3067)1] = -0.6191pu$$

$$V_4(1) = [v(0) + v(2)W_4^2] + W_4^1\,[v(1) + (v(3)\,W_4^2)] \quad (3.54)$$

$$= [0.7785 + (-0.8914)-1] + -j[-0.1995 + (-0.3067)-1] = 1.6183pu$$

$$V_4(2) = [v(0) + v(2)W_4^0] + W_4^2\,[v(1) + (v(3)\,W_4^0)] \quad (3.55)$$

$$= [0.6316 + (-0.7266)1] + -1[-0.0125 + (-0.8924)1] = 0.8099pu$$

$$V_4(3) = [v(0) + v(2)W_4^2] + W_4^3\,[v(1) + (v(3)\,W_4^2)] \quad (3.56)$$

$$= [0.6316 + (-0.7266)-1] + j[-0.0125 + (-0.8924)-1] = 1.6183pu$$

3.3.2 Computation of FFT Pre-fault Current

$$I_4(0) = [i(0) + i(2)W_4^0] + W_4^0\,[i(1) + (i(3)\,W_4^0)] \quad (3.57)$$

$$= [0.2399 + (-0.2465)1] + 1[-0.0047 + (-0.3390)1] = -0.3503pu$$

$$I_4(1) = [i(0) + i(2)W_4^2] + W_4^1\,[i(1) + (i(3)\,W_4^2)] \quad (3.58)$$

$$= [0.2399 + (-0.2465)-1] + -j[-0.0047 + (-0.3390)-1] = 0.1521pu$$

$$I_4(2) = [i(0) + i(2)W_4^0] + W_4^2\,[i(1) + (i(3)\,W_4^0)] \quad (3.59)$$

$$= [0.2399 + (-0.2465)1] + -1[-0.0047 + (-0.3390)1] = 0.3371pu$$

$$I_4(3) = [i(0) + i(2)W_4^2] + W_4^3\,[v(1) + (i(3)\,W_4^2)] \quad (3.60)$$

$$= [0.2399 + (-0.2465)-1] + j[-0.0047 + (-0.3390)-1] = 0.5902pu$$

3.3.3 Computation of FFT Three Phase Fault Voltage

$V_4(0) = [v(0) + v(2)W_4^0] + W_4^0 [v(1) + (v(3) W_4^0)]$ (3.61)

$= [0.9869 + (1.0332)1] + 1[1.0109 + (1.0538)1] = 4.0848pu$

$V_4(1) = [v(0) + v(2)W_4^2] + W_4^1 [v(1) + (v(3) W_4^2)]$ (3.62)

$= [0.9869 + (1.0332)-1] + -j[1.0109 + (1.0538)-1] = -0.0463pu$

$V_4(2) = [v(0) + v(2)W_4^0] + W_4^2 [v(1) + (v(3) W_4^0)]$ (3.63)

$= [0.9869 + (1.0332)1] + -1[1.0109 + (1.0538)1] = -0.0446pu$

$V_4(3) = [v(0) + v(2)W_4^2] + W_4^3 [v(1) + (v(3) W_4^2)]$ (3.64)

$= [0.9869 + (1.0332)-1] + j[1.0109 + (1.0538)-1] = -0.0463pu$

3.3.4 Computation of FFT Three Phase Fault Current

$I_4(0) = [i(0) + i(2)W_4^0] + W_4^0 [i(1) + (i(3) W_4^0)]$ (3.65)

$= [1.3608 + (1.4246)1] + 1[1.3614 + (1.4530)1] = 5.5992pu.$

$I_4(1) = [i(0) + i(2)W_4^2] + W_4^1 [i(1) + (i(3) W_4^2)]$ (3.66)

$= [1.3608 + (1.4246)-1] + -j[1.3614 + (1.4530)-1] = -0.1116pu$

$I_4(2) = [i(0) + i(2)W_4^0] + W_4^2 [i(1) + (i(3) W_4^0)]$ (3.67)

$= [1.3608 + (1.4246)1] + -1[1.3614 + (1.4530)1] = -0.029pu$

$I_4(3) = [i(0) + i(2)W_4^2] + W_4^3 [i(1) + (i(3) W_4^2)]$ (3.68)

$= [1.3608 + (1.4246)-1] + j[1.3614 + (1.4530)-1] = -0.1116pu$

With the DFT and FFT mathematical computations results for the pre – fault and three phase fault signals, impedance Z_{0123} values which are also important factors that proves whether fault had occurred on the transmission line or not is computed.

The results of the DFT and FFT mathematical computations for the pre – fault and three phase fault signals are tabulated and shown in chapter four.

3.4 Simulation Approach for Development of Fault Detection Model using DFT and FFT

The DFT and FFT mathematical computational processes were adopted in this section. Each is modeled using Matlab/Simulink R2017a. The transmission line is also modeled using Matlab/Simulink with the transmission line data obtain from Onitsha transmission station.

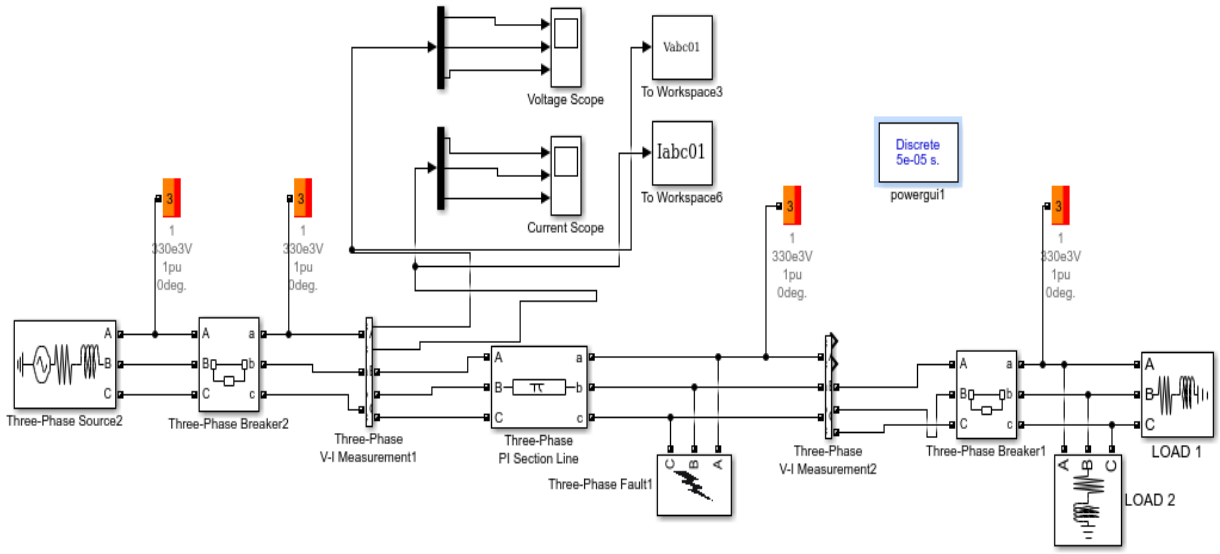

Figure 3.4: Matlab/Simulink Model of 330KV, 96Km Onitsha – Enugu Power System Transmission Line

The three phase voltages V_a, V_b, V_c and currents I_a, I_b, and I_c were generated using figure 3.4 and recorded when the system is on load, at pre – fault (normal) and fault conditions.

Three - phase fault is also simulated on the line using figure 3.4 when the line is on load and three phase fault voltage and current are generated and recorded.

3.4.1 Computation of DFT Pre – fault and three phase fault Voltage

Before computing the DFT parameters, the discretization equations (3.21) and (3.22) were modeled using Matlab/Simulink and shown in figure 3.13 and 3.14.

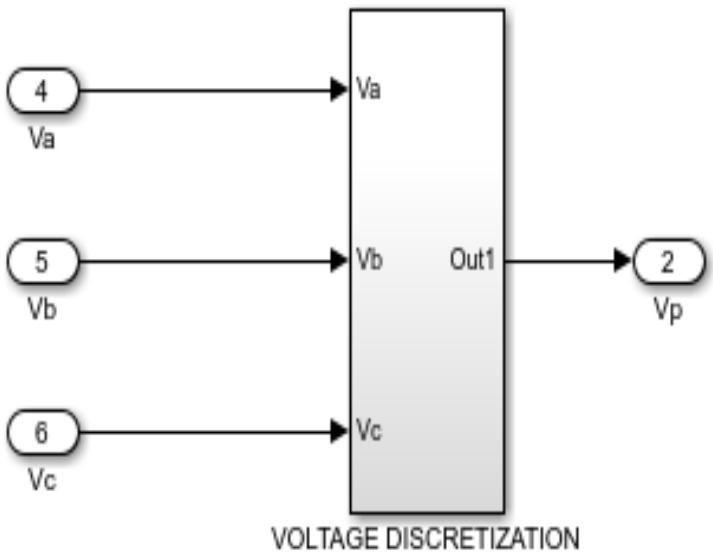

Figure 3.5: Matlab/Simulink Model of Voltage Discretization Equation (3.21)

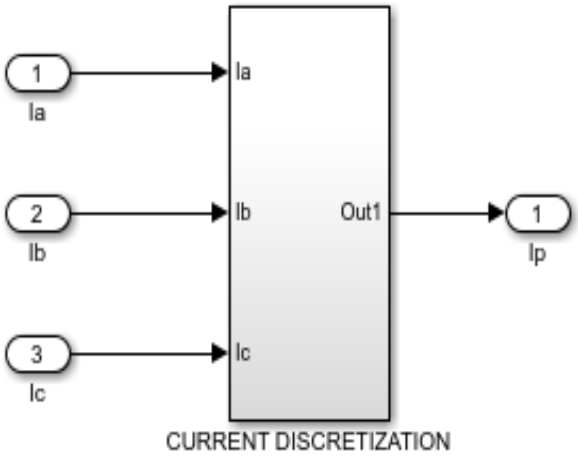

Figure 3.6: Matlab/Simulink Model of Current Discretization Modeled using Equation (3.22)

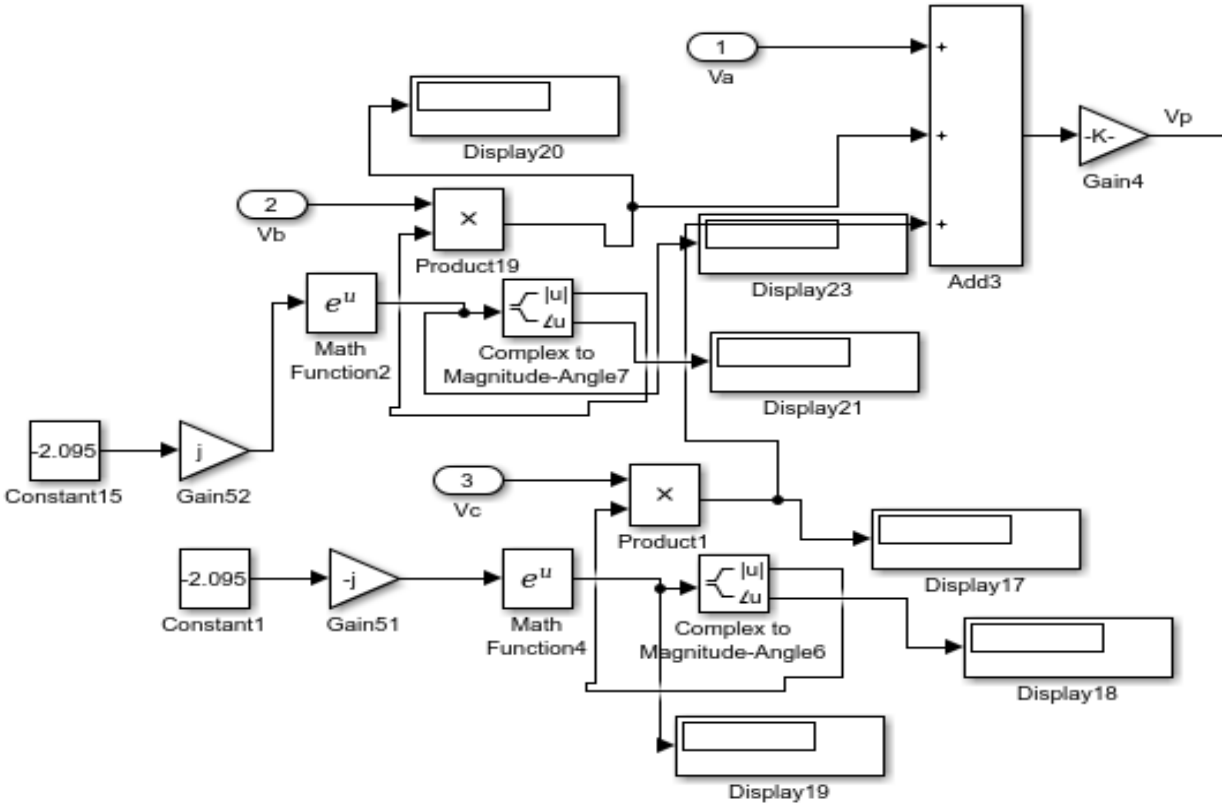

Figure 3.7: Internal Structure of Matlab/Simulink Model of Voltage DiscretizationModeled using Equation (3.21)

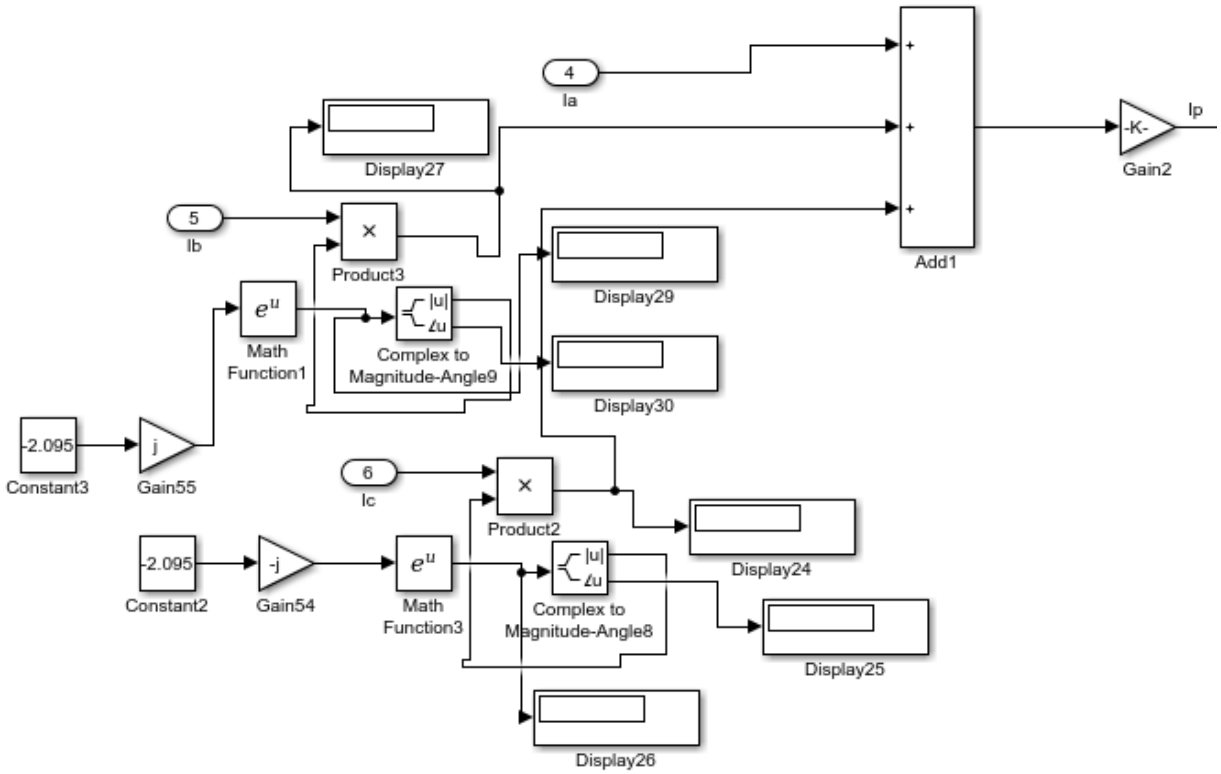

Figure 3.8: Internal Structure of Matlab/Simulink Model of Current Discretization Modeled using Equation (3.22)

The sinusoidal phasors v(t) and i(t) of equations (3.27) and (3.28) respectively were modeled using Matlab/Simulink and are shown in figures 3.15 and 3.17 respectively. Simulation of figure 3.15 and 3.17 generates three phase voltage and current sinusoidal (V_n (t) and I_n (t)) values for the 4 – points.

Each of the voltage and current discretization model has outputs which are the V_p and I_p respectively. These discretized phase values are to be sampled into the sinusoidal phasors v(t) and i(t) of equations (3.27) and (3.28) for the 4 points.

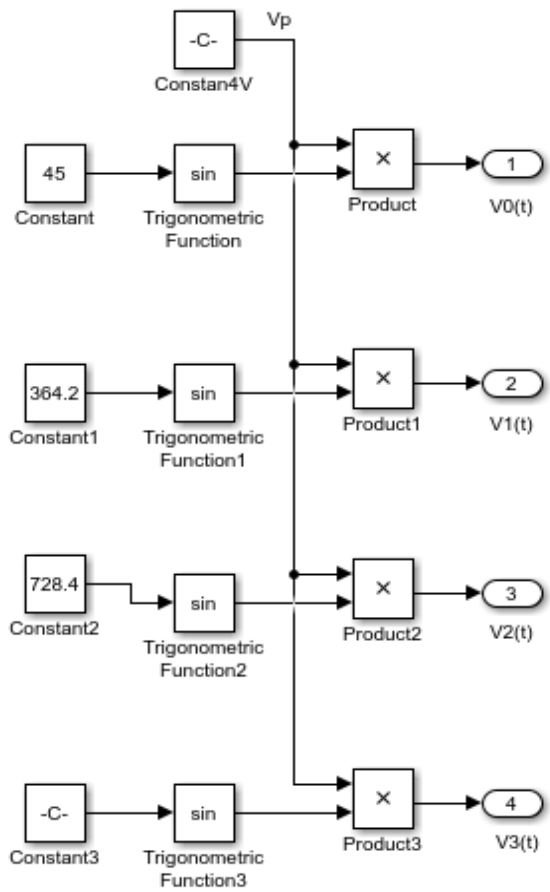

Figure 3.9: Matlab/Simulink Modeling and computation of prefault and three phase fault voltage signals ($V_n(t)$) for n = 0, 1, 2, 3 using Equation (3.29 to 3.32)

Therefore,

With $v_n(t) = [v_0(t), v_1(t), v_2(t), v_3(t)]^T$, the DFT of $v_n(t)$ which is Vn[0 1 2 3] is computed using the matlab/simulink model of figure 3.16.

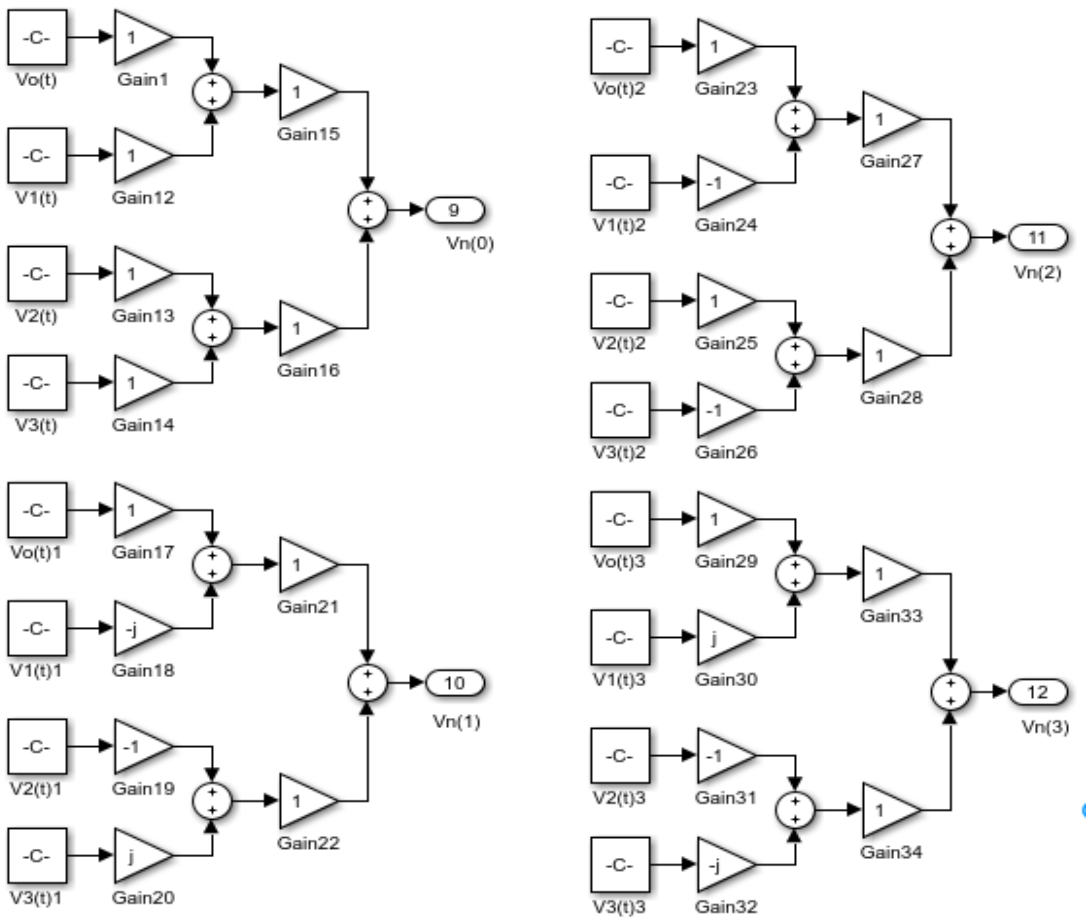

Figure 3.10: Matlab/Simulink Modeling for DFT computation of voltage signals (V_n) for n = 0, 1, 2, 3 using Equation (3.34 to 3.40) above.

Thus, V(n) $[N = 0, 1, 2, 3]^T$ is determined.

Where, equation (3.56) is the output result of computation of DFT pre-fault or three phase fault voltage using simulation method.

3.4.2 Computation of DFT Pre – fault and three phase fault Current

Using equations (3.21 to 3.39), computation of DFT current using simulation method and figure 3.17is performed.

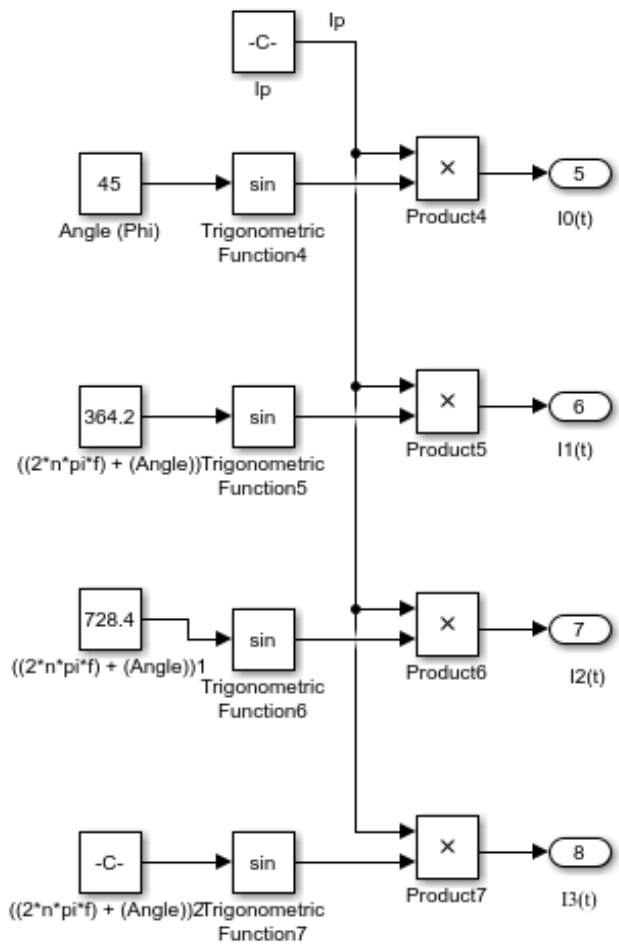

Figure 3.11: Matlab/Simulink Modeling and computation of current signals ($I_n(t)$) for n = 0, 1, 2, 3 using Equations (3.42 to 3.45)

Therefore,

DFT of $i_n(t)$ is computed using figure3.17;

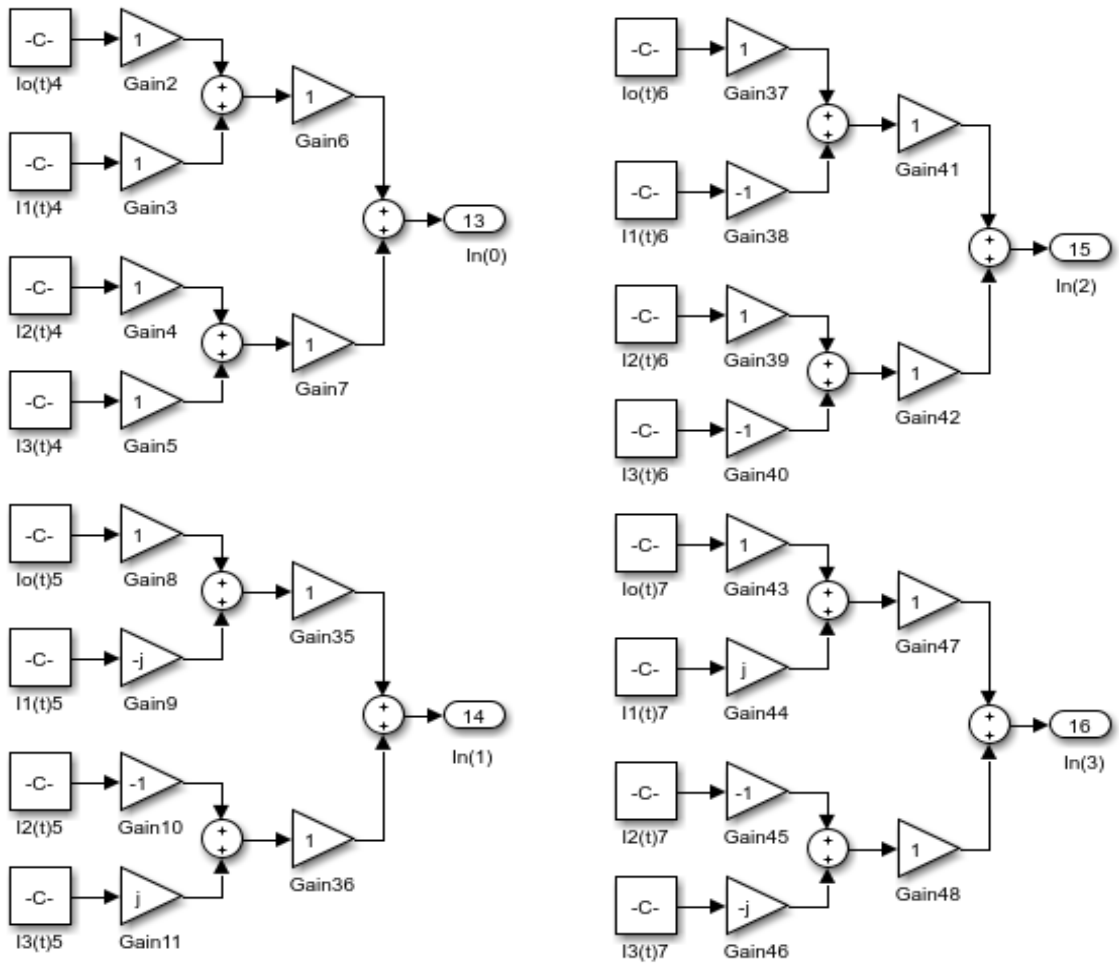

Figure 3.12: Matlab/Simulink Modeling for DFT computation of current signals (I(n)) for n = 0, 1, 2, 3 using Equation (3.34 to 3.40).

Thus,

I(n) $[N = 0, 1, 2, 3]^T$ is determined and recorded.

3.5 Computer Development of Fault Detection Algorithm using FFT

3.5.1 Computation of FFT Pre – fault and three phase fault Voltages

Since we obtained the real time three phase pre-fault and fault values of voltage and current and has computed their discrete versions V_P and I_P in the previous section.

Therefore, applying Butterfly flow process shown in figure 3.3 and equations (3.52 to 3.56), computation of pre-fault and three phase fault signals using FFT is performed using simulation approach (Capolino G. A. et al 2003).

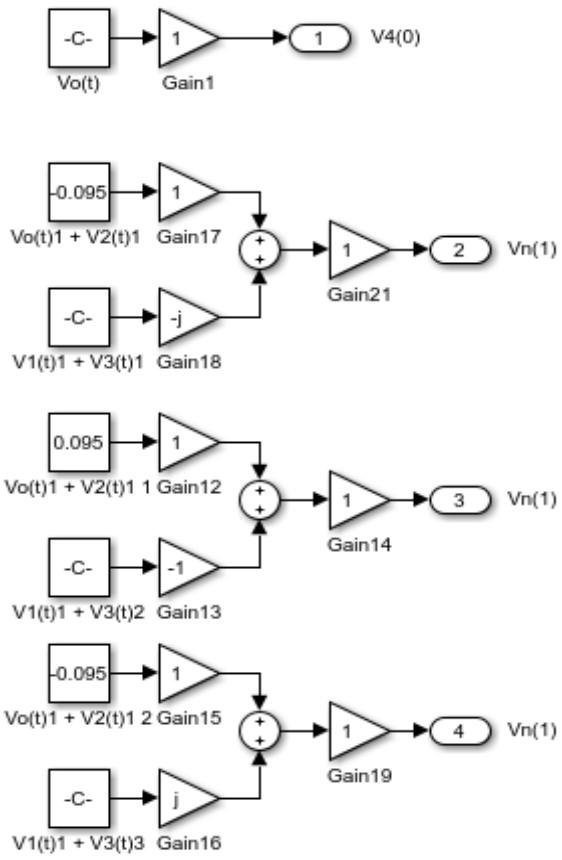

Figure 3.13: Matlab/Simulink Model for FFT computation of both Pre-fault and Three Phase Fault Voltages at 4 – point (N = 4)

Figure 3.19 is used for the computation of FFT Pre-fault and three phase fault voltages. But the values of pre-fault $V_n(t)$ is different from that of three phase fault $V_n(t)$. Then $V_k(n) = V_k(0, 1, 2, 3)$ were obtained as FFT pre-fault and three phase fault computational results.

3.5.2 Computation of FFT Pre-fault and Three Phase Fault Current

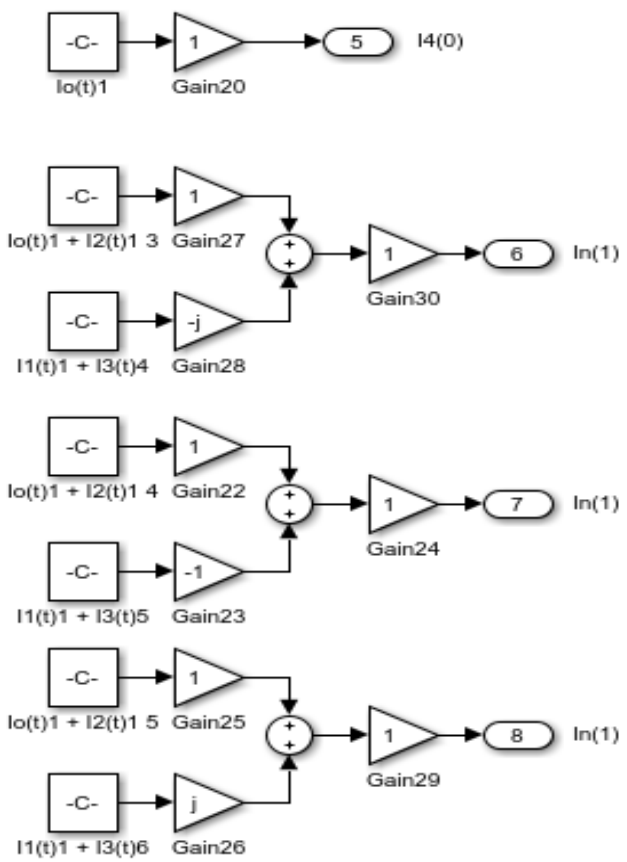

Figure 3.14: Matlab/Simulink Model for FFT computation of both Pre-fault and Three Phase Fault Currents at 4 – point (N = 4)

Figure 3.14 is used for the computation of FFT Pre-fault and three phase fault currents. But the values of pre-fault $I_n(t)$ is also different from that of three phase

fault $I_n(t)$. Then $I_k(n) = I_k(0, 1, 2, 3)$ were obtained as FFT pre-fault and three phase fault computational results.

With the DFT and FFT computational results for the pre – fault and three phase fault signals using simulation approach, impedance Z_{0123} values which are also important factors that proves whether fault had occurred on the transmission line or not were computed.

The results of the DFT and FFT computations for the pre – fault and three phase fault signals using mathematical and simulation approaches are tabulated and shown in chapter four.

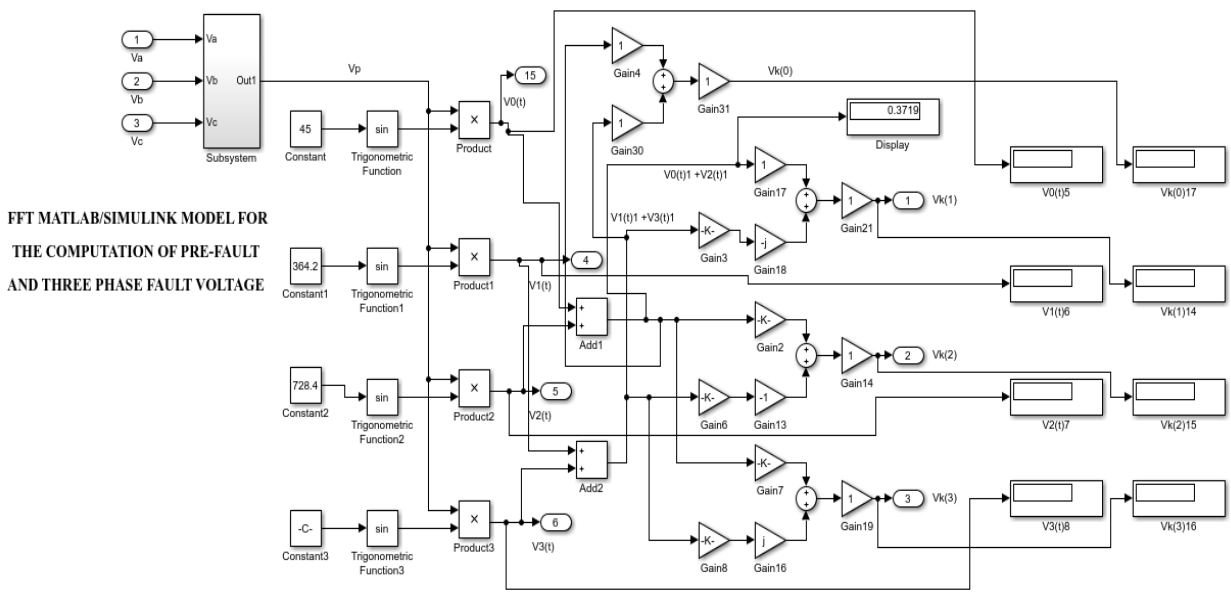

Figure 3.15: FFT Fault DetectionMatlab/Simulink Model for the Computation Voltage

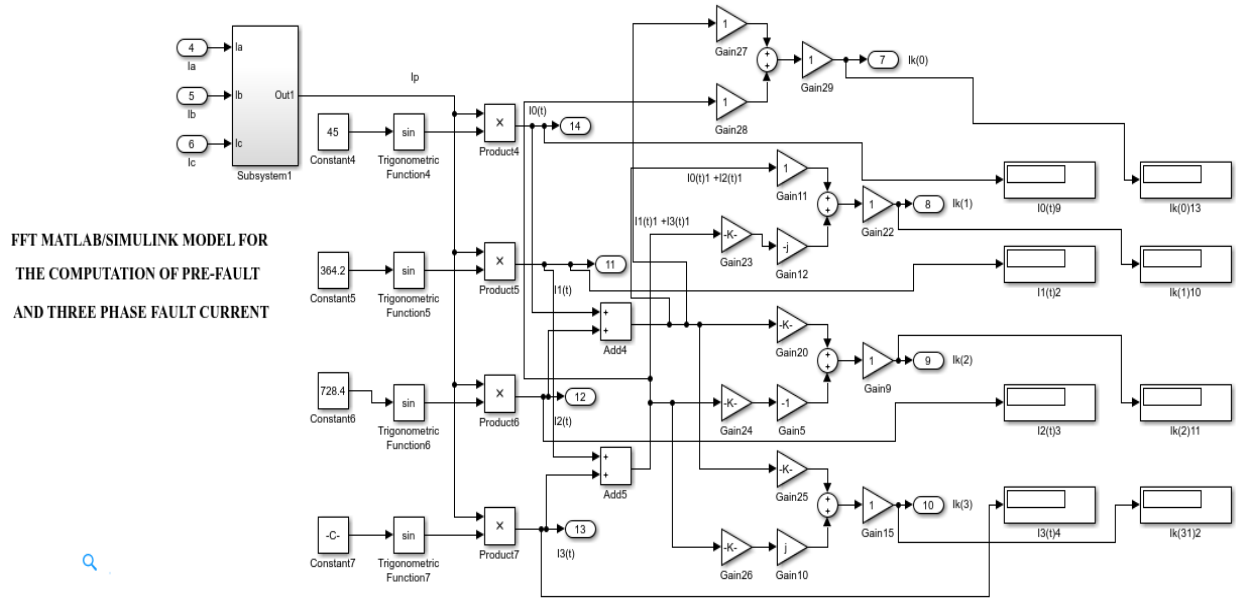

Figure 3.16: FFT Fault Detection Matlab/Simulink Model for the Computation Current

3.6 **Modeling of the transmission line**

Considering our case study area, Onitsha – Enugu 330KV transmission line which is 96Km distance. It corresponds to the line length less than 100km and belongs to short line category.

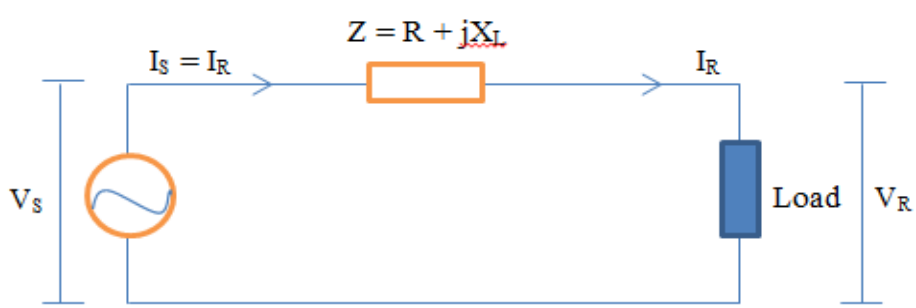

Figure 3.17. Equivalent circuit of short transmission line

The shunt admittance or shunt reactance (jωcl) of the transmission line is very small enough to be negligible resulting to the simple equivalent circuit of figure3.1.

The relationship between the sending and receiving – ends voltage and currents can be written as;

$$\begin{bmatrix} V_S \\ I_S \end{bmatrix} = \begin{bmatrix} 1 & Z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_R \\ I_R \end{bmatrix} \tag{3.69}$$

Thus,

$$|V_S| = [\,(\,|V_R|\cos\emptyset_R + \,|I|R)^2 + \,(\,|V_R|\sin\emptyset_R + \,|I|X_L)^2]^{\frac{1}{2}} \tag{3.70}$$

$$|V_S| = [\,(\,|V_R|^2 + \,|I|^2(R + X_L)^{\,2} + \,2(\,|V_R||I|(R\cos\emptyset_R + \,X_L\sin\emptyset_R)]^{\frac{1}{2}} \tag{3.71}$$

$$|V_S| = |V_R|[1 + \frac{2|I|R\cos\emptyset_R}{|V_R|} + \frac{2|I|X_L\sin\emptyset_R}{|V_R|} + \frac{2|I|^2X_L(R^2 + X_L{}^2)}{|V_R|^2}\quad]^{\frac{1}{2}} \tag{3.72}$$

$$\frac{2|I|^2X_L(R^2 + X_L{}^2)}{|V_R|^2} \cong 0 \tag{3.73}$$

Then,

$$|V_S| = |V_R|[1 + \frac{2|I|R\cos\emptyset_R}{|V_R|} + \frac{2|I|X_L\sin\emptyset_R}{|V_R|}\quad]^{\frac{1}{2}} \tag{3.74}$$

However, by binomial expansion, and retaining first order terms, we obtain that,

$$|V_S| = |V_R|[1 + \frac{2|I|R\cos\emptyset_R}{|V_R|} + \frac{2|I|X_L\sin\emptyset_R}{|V_R|}\quad]^{\frac{1}{2}} \tag{3.75}$$

$$|V_S| = |V_R| + \quad |I|(R\cos\emptyset_R \ + \ X_L\sin\emptyset_R) \tag{3.76}$$

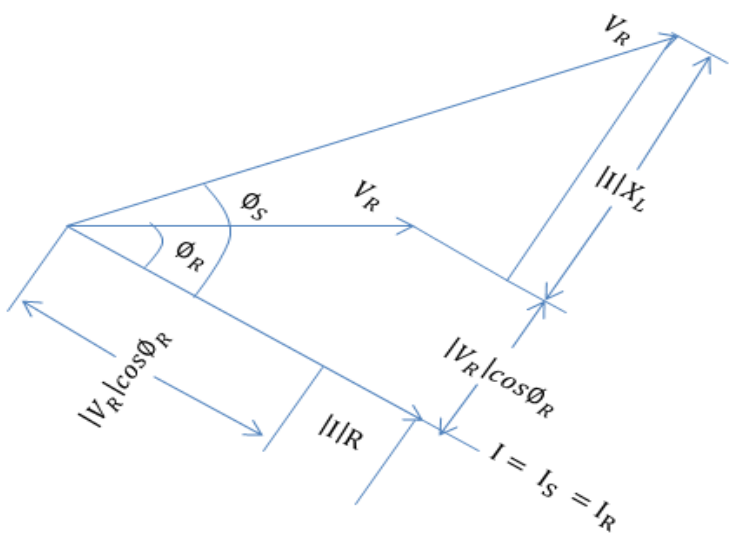

Figure 3.18: Phasor diagram of distance 96Km Onitsha –Enugu transmission line (less than 100Km)

3.6.1 Determination of transmission line parameters

The parameters of the transmission line such as V, I, Z, X and R can be determined under two conditions namely, pre – fault (No fault) and post – fault (under fault) conditions.

3.6.1.1 Line positive and negative sequence resistance

According to WesterveltS. M., (2012), line and sequence parameters of a transmission line can be determined using the following;

$$R_{sdc} = \frac{\rho_{To} l}{A} \tag{3.77}$$

Where the $R_{sdc}$is the conductor series resistance at temperature T_o, $\rho_{To}$is the conductor resistivity at any temperature T_o in degree Celsius. L is the actual length of line; A is the cross-sectional area of the line conductor.

Considering Aluminum conductor material with resistivity of 2.85×10^{-8} ohm.m at 20.00429^{O}C and conductivity of 3.77×10^{7}ohm. m. The temperature corresponding to the resistivity value given above is used as a reference temperature of theR_{sdc}.

$$\frac{R_{t2}}{R_{t1}} = \frac{\beta_{Ro} + t_2}{\beta_{Ro} + t_1} \tag{3.78}$$

Here, there are three factors that determine the transmission line sequence resistance. They are ambient temperature, power system operating frequency and the current density of the line. If the temperature is varied, the conductor resistance will change linearly with respect to change in the temperature. Equation 3.78 can be used to obtain the new change in resistance of the conductor when the ambient temperature is varied

Table 3.3 Classification and representation of transmission lines

S/NO	TYPE OF LINE	LINE LENGTH (Km)	MODEL OF REPRESENTATION	REMARKS
1	Short Line	≤ 100	Series Impedance ($R + jX_L$)	Neglects Shunt X_C
2	Medium Line	100 to 250	Normal π or T network	Lumped parameters
3	Long Line	> 250	Equivalent π or T network	Distributed parameters

Table 3.4: Conductor materials and conductivity value

S/NO	MATERIALS	PARAMETER	VALUE
1	Aluminum hard drawn 61% conductivity	β_{Ro}	228
2	Copper annealed 100% conductivity	β_{Ro}	234.5
3	Copper hard drawn 97.3% conductivity	β_{Ro}	241.5

R_{t2} is the conductor resistance at any temperature t_2 in degree Celsius, R_{t1} is the conductor resistance at any temperature t_1 in degree Celsius. β_{Ro} is the material specific temperature constant inferred at zero resistance. Typical values are shown on table 3.2.

The conductor resistance can now be determined using equation (3.9) & (3.10) as

$$R_{Tdc} = R_{Sdc} \text{ x } \frac{R_{t2}}{R_{t1}} = \frac{\rho_{To} l}{A} \text{x} \frac{R_{t2}}{R_{t1}} = \frac{\beta_{Ro} + t_2}{\beta_{Ro} + t_1} \quad (3.79)$$

Where R_{Tdc} is the dc resistance of the conductor at a specific temperature, t. The dc resistance varies on the transmission line per foot and varies with temperature as seen in the equation 3.3. The value obtained using equation (3.10) represent part of the total sequence conductor resistance in AC application. In DC application resistance are considered homogeneous across the cross section of the conductor. But in AC, a time varying, non – uniform flux within the conductor causes electrons to flow near the surface of the conductor. This actually reduces the cross-sectional area of the conductor and increase its resistance. This change in resistance of the line from DC to AC conditions is referred to as the Skin effect.

3.6.1.2 Skin Effect

The distribution of current throughout the cross section of a conductor is uniform only when DC is passing through it. On the centrally, when AC is flowing through a conductor, the current is not uniformly distributed over the cross section in the sense that, the current density (time varying, non – uniform flux) is higher at the surface of the conductor compared to the centre of the conductor. This effect become more pronounced as the frequency is increased.

Thus, increase in frequency of a transmission line results to increase in current density at the surface of the conductor than at the centre of the conductor and, thus result to increase in current field and its effectto the surrounding. This phenomenon is referred to as Skin Effect. It also results to faults on the transmission line due to little shift (decrease or increase) in frequency of line (Westervelt S. M., 2012).

$$R_{SAC} = K \text{ x } R_{SDC} \tag{3.80}$$

$$X = 0.063598 \sqrt{\left(\frac{\mu F}{R_{DC(m)}}\right)} \tag{3.81}$$

Where R_{SAC} is the series resistance of the transmission line conductor at 50Hz frequencyR_{SDC} is the DC resistance of the conductor at any temperature T_O in degree Celsius. K is a value function and a function of X factor in equation 3.5. f is the conductor operational frequency, μ is the conductor permeability of which μ = 1.0 is the permeability of aluminum since it not a magnetic material and $R_{DC(m)}$ is the conductor DC resistance in ohm per metre. X is the tabular value used to determine K value from appendix 1. AC resistance can then be determined using equation (3.12) and (3.13).

3.6.1.3 Line, Positive and Negative Inductance

In general, the spacing D_{ab}, D_{bc}, &D_{ac} between the conductors of three phase transmission line are not equal (TuranG., 1988, Gupta B. L., 2009, and Nagrat and Kotari, 2004). For any conductor configuration, the average value of inductance and capacitance can be found by representing the transmission line by one with equivalent equilateral spacing (D_{eq}), see equation (3.14).

$$D_{eq} = D_m = (D_{ab}D_{bc}D_{ca})^{\frac{1}{3}} \tag{3.82}$$

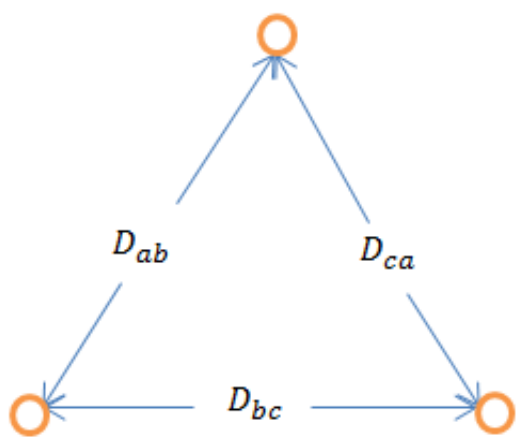

Figure 3.19: Equilateral Spacing of the conductors of the three - phase transmission line

The average inductance per phase is given by;

$$L_n = 2 \text{ x } 10^{-7} \text{Ln } \frac{D_{eq}}{D_S} \text{ (h/m)} \tag{3.83}$$

Where n is a particular phase (a, b or c)

$$X_L = 2\pi fL \tag{3.84}$$

$$L_1 = \frac{\mu}{2} + 2\text{Ln } \frac{1}{r} + 2\text{Ln } (D_{ij}) \tag{3.85}$$

$$X_{L1} = 2\pi f L_1 \quad (3.86)$$

Where L is the self-inductance of the conductor considered near field and adjacent conductor. $\frac{\mu}{2}$ is inductance within the conductor, $2Ln\ \frac{1}{r}$ is the inductance in the near field (1 foot) of the conductor, $2Ln\,(D_{ij})$ is the inductance of the conductor considering linkage to an adjacent conductor greater than 1 foot away and at a distance of j from i. (D_{ij}) is the distance between the conductor being considered and the return path conductor.

Inductive reactance, X_L can be determined using equation (3.86).

The positive sequence inductance and it reactance can be obtained using equations (3.85) and (3.86).

In AC power system, transmission line has real and imaginary or complex quantities such as reactive components which must be considered in addition to Resistance. Since current flow through a complete path with current retiring to the source, the minimum number of paths for a complete circuit is two. Also load currents produce an electric field that surrounds a conductor and radiates outward and finally affects the retiring path conductor circuit. These lines of flux impact an inductive reactance within itself and contribute to the conductor's overall self – inductance.

Equation (3.85) can be used to obtain the positive sequence inductance for two conductor lines i and j which can replaced with phase ab, bc or ca and when the distance between them is in excess of one foot as in three phase high voltage overhead transmission lines.

This chapter consists of the methodological procedure for the development of the algorithm for fault diagnosis on 330/132KV power system transmission line using signal analysis, Discrete Fourier Transform (DFT) method.

Figure 3.20 illustrates the typical single transmission line diagram used in this study. The system parameters were used for the modeling of the transmission line using Matlab/Simulink 2016a were obtained from Onitsha transmission line station and are given as; transmission line distance is 96Km, real power 87.7MW, reactive power 21.40MVar, active power 90.27MVA with 50Hz frequency. The system also consists of three buses and one line. The zones of protection are given as zone1 = 76.8Km, zone 2 = 144Km, zone 3 = 240Km all from bus A.

The performance of the transmission line is dependent on the performance of the power system conditions. The transmission line is grouped into short, medium and long lines. Short line is 100km, medium line is 100 – 250km and long line is 250km and above.

The transmission line is represented by simplified models of circuits and mathematical equations which are expressed and simplified through certain appropriate assumptions.

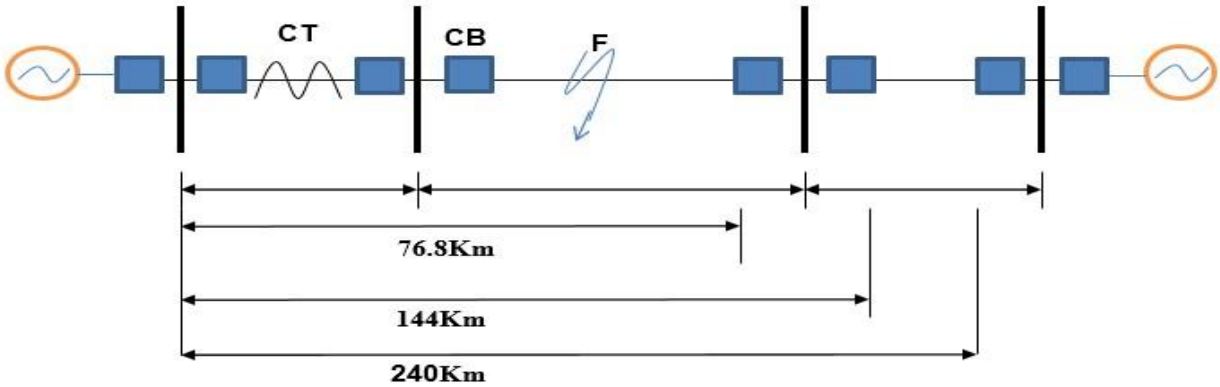

Figure 3.20: Onitsha to Enugu 330KV Single Transmission Line Diagram

The values of the parameters of the transmission line obtained were used for the modeling of the three - phase short transmission line shown in figure 3.26.

CHAPTER FOUR

SIMULATION AND RESULT ANALYSIS

4.1 RESULTS OBTAINED THROUGH MATHEMATICAL APPROACH

The per unit voltage and current values of table 3.3 and 3.4 are magnitude or amplitude values which when sampled into the equations 3.61 and 3.62 respectively is used to obtain the discrete values tabulated in table 4.1 and 4.2.

Table 4.1: Pre-fault discrete voltage, current and impedance values

S/N	$v_p(n)$	$i_p(n)$	$z_p(n)$	FAULTS
1	2.7591	0.6121	4.5076	L – G
2	0.9331	0.2999	3.1114	LL – G
3	2.4220	0.0000	0.0000	L – L
4	0.8932	0.3393	2.6295	LLL

Table 4.2: Three phase fault discrete voltage, current and impedance values

S/N	$v_p(n)$	$i_p(n)$	$z_p(n)$	FAULTS
1	2.7425	1.1076	2.5488	L – G
2	2.0997	3.8590	0.5441	LL – G
3	3.0976	0.5488	5.6443	L – L
4	0.1570	1.5992	0.0982	LLL

Table 4.1 and 4.2 illustrate the discrete values (V_p and I_p) obtained for pre-fault and fault conditions under four categories of fault, one symmetrical and three unsymmetrical faults.

The DFT is applied to these discrete values following the methodological procedure stated in chapter three and using equation 3.51 and 3.52 to determine the time – domain and frequency – domain components which contain the information such as fault information about the transmission line.

Table 4.3: Mathematical Approach DFT Application for Pre-fault Condition at N – POINTS (k = 0,…, N – 1)

S/N	**N - POINTS**	$\mathbf{V_{0,1,2,3}}$ for N = 4	$\mathbf{I_{0,1,2,3}}$ for N = 4	V(n)	I(n)	Z(n)	FAULTS
1	0	0.7785	0.2956	-0.6191	-0.1670	3.7072	NIL
2	1	-0.1995	-0.0076	1.6733	0.6434	2.6007	NIL
3	2	-0.8914	-0.3385	0.3933	0.0812	4.8300	NIL
4	3	-0.3067	-0.1165	1.7460	0.6461	2.7024	NIL

Table 4.4: Mathematical Approach DFT Application for Three Phase Fault Condition at N – POINTS (k = 0,…, N – 1)

S/N	**N - POINTS**	$\mathbf{V_{0,1,2,3}}$ for N = 4	$\mathbf{I_{0,1,2,3}}$ for N = 4	$V_n(N)$	$I_n(N)$	$Z_n(N)$	FAULTS
1	0	0.1368	1.3938	-0.1087	-1.1077	0.0981	LLL
2	1	-0.0350	-0.3564	0.2941	2.9899	0.0984	LLL
3	2	-0.1567	-1.5960	0.0691	0.7053	0.0980	LLL
4	3	-0.0540	-1.5491	0.3060	3.1239	0.0980	LLL

Table 4.5: Mathematical Approach FFT Application for Pre-Fault Condition at N – POINTS (k = 0,…, N – 1)

S/N	**N - POINTS**	$\mathbf{V_{0,1,2,3}}$ for N = 4	$\mathbf{I_{0,1,2,3}}$ for N = 4	$V_k(N)$	$I_k(N)$	$Z_k(N)$	FAULTS
1	0	0.7785	0.2399	-0.6191	-0.3503	1.7673	LLL
2	1	-0.1995	-0.0047	1.6183	0.1521	10.6397	LLL
3	2	-0.8914	-0.2465	0.8099	0.3371	2.4026	LLL
4	3	-0.3067	-0.3390	1.6183	0.5902	2.7420	LLL

Table 4.6: Table 4.4: Mathematical Approach FFT Application for Three Phase Fault Condition at N – POINTS (k = 0,…, N – 1)

S/N	**N - POINTS**	$\mathbf{V_{0,1,2,3}}$ for N = 4	$\mathbf{I_{0,1,2,3}}$ for N = 4	$V_k(N)$	$I_k(N)$	$Z_k(N)$	FAULTS
1	0	0.9869	1.3608	4.0848	5.5992	0.7295	LLL
2	1	1.0109	1.3614	-0.0463	-0.1116	0.4149	LLL
3	2	1.0332	1.4246	-0.0446	-0.0290	1.5380	LLL
4	3	1.0538	1.4530	-0.0463	-0.1116	0.4149	LLL

The DFT and FFT results obtained using mathematical approach are tabulated on table 4.3& 4.4 and table 4.5 & 4.6 for three phase (LLL) pre-fault and fault conditions respectively.

To obtain the time – domain and frequency – domain waveform and spectrum diagram representing $V_n(N)$, $I_n(N)$and $V_k(N)$, $I_k(N)$ respectively, we plot V(n) and I(n) individually against time for time domain and against the frequency for

frequency domain (spectrum waveform) using matlabwindow digital signal processing waveform command (wvtool).

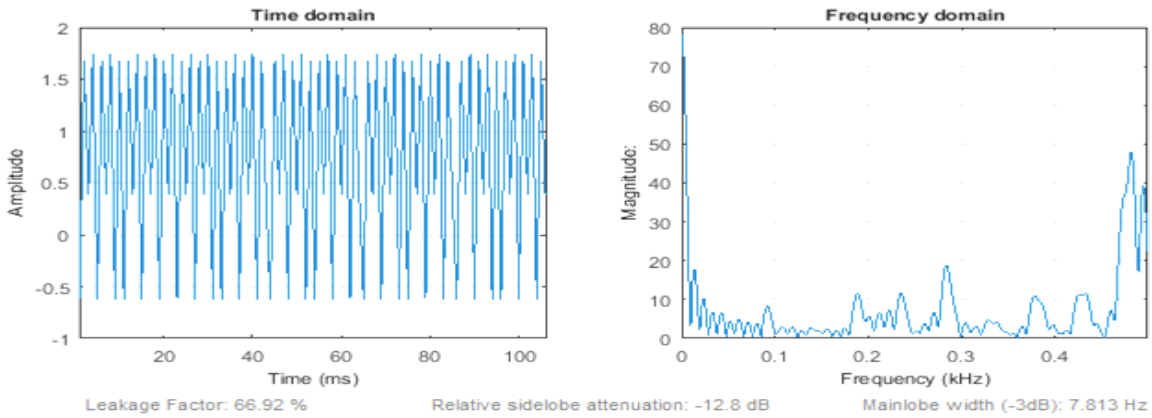

Figure 4.1: Mathematical Approach Pre-Fault DFT Voltage (Vn) Signal Waveform

Figure 4.1 is the time and frequency – domain pre-fault DFT voltage signal waveform obtained by plotting the V_n against time and frequency respectively. The waveform is not sinusoidal even at No fault condition. It also shows that the maximum amplitude of V_n is 1.8pu at time of 2msecs and largest magnitude of 50pu at 0.48kHz.

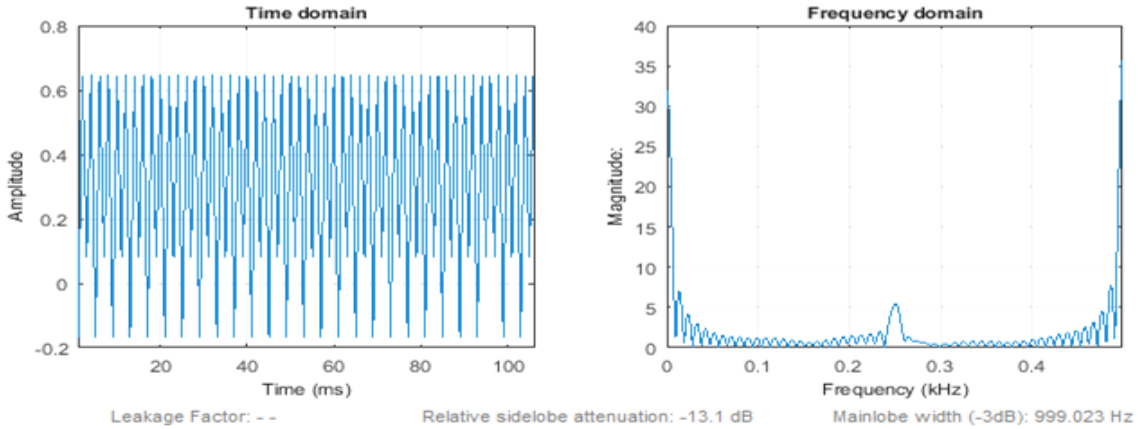

Figure 4.2: Mathematical Approach Pre-Fault DFT Current (In) Signal Waveform

Figure 4.2 is the time and frequency – domain pre-fault DFT current signal waveform obtained by plotting the I_n against time t and frequency respectively. The waveform is not sinusoidal even at No fault condition. It also shows that the maximum amplitude of I_n is 0.65pu at time of 2msecs and largest magnitude of 5pu at 0.25kHz.

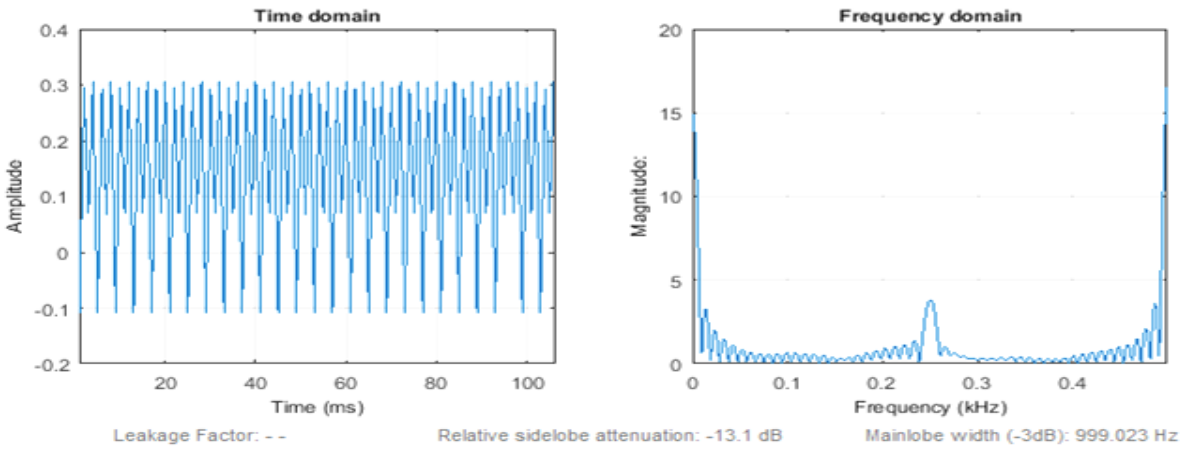

Figure 4.3: Mathematical Approach Three Phase Fault DFT Voltage (Vn) Signal Waveform

Figure 4.3is the time and frequency – domain three phase fault DFT voltage signal waveform obtained by plotting the V_n against time t and frequency respectively. The waveform is under three phase fault condition. It also shows that the maximum amplitude of V_n is 0.3pu at time of 2msecs and largest magnitude of 4.8pu at 0.25kHz.

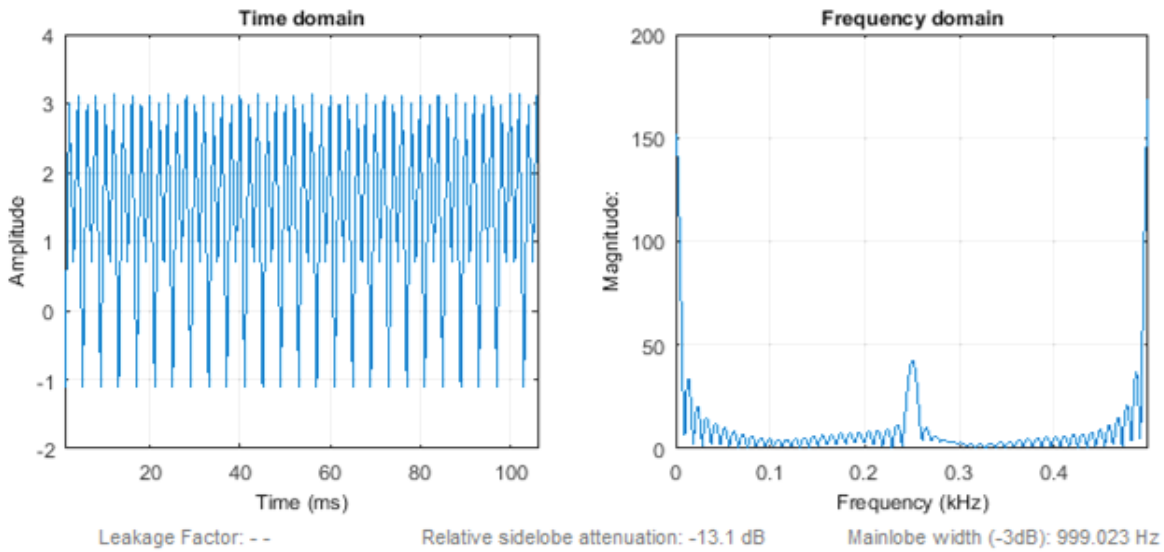

Figure 4.4: Mathematical Approach Three Phase Fault DFT Current (In) Signal Waveform

Figure 4.4 is the time and frequency – domain three phase fault DFT current signal waveform obtained by plotting the I_n against time t and frequency respectively. It also shows that the maximum amplitude of I_n is 3pu at time of 2msecs and largest magnitude of 48pu at 0.25kHz.

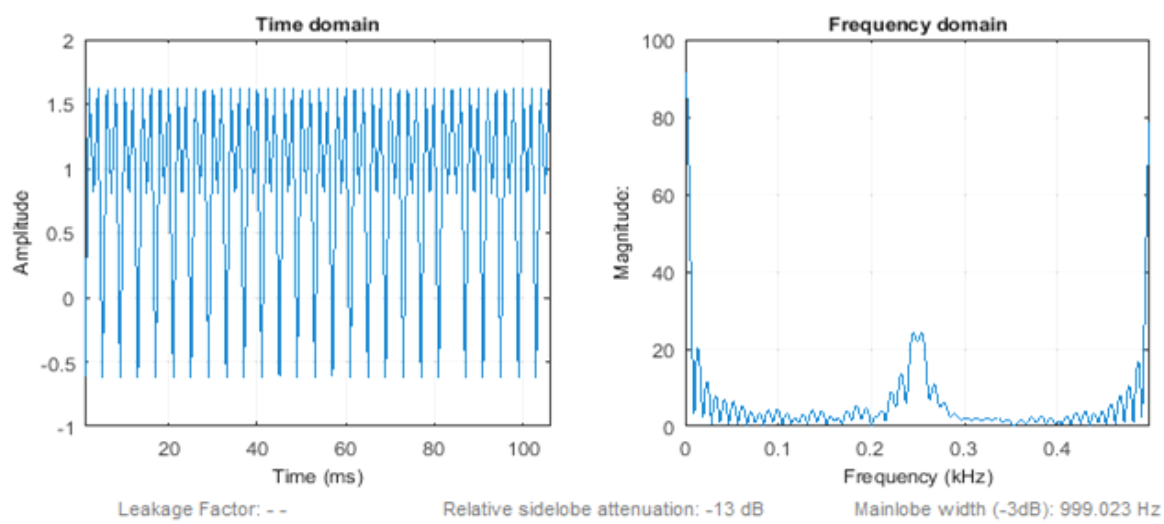

Figure 4.5: Mathematical Approach Pre-Fault FFT Voltage V(k) Signal Waveform

Figure 4.5 is the time and frequency – domain pre-fault FFT voltage signal waveform obtained by plotting the V_k against time t and frequency respectively. The waveform is not sinusoidal even at No fault condition. It also shows that the maximum amplitude of V_k is 1.6pu at time of 2msecs and largest magnitude of 28pu at 0.25kHz.

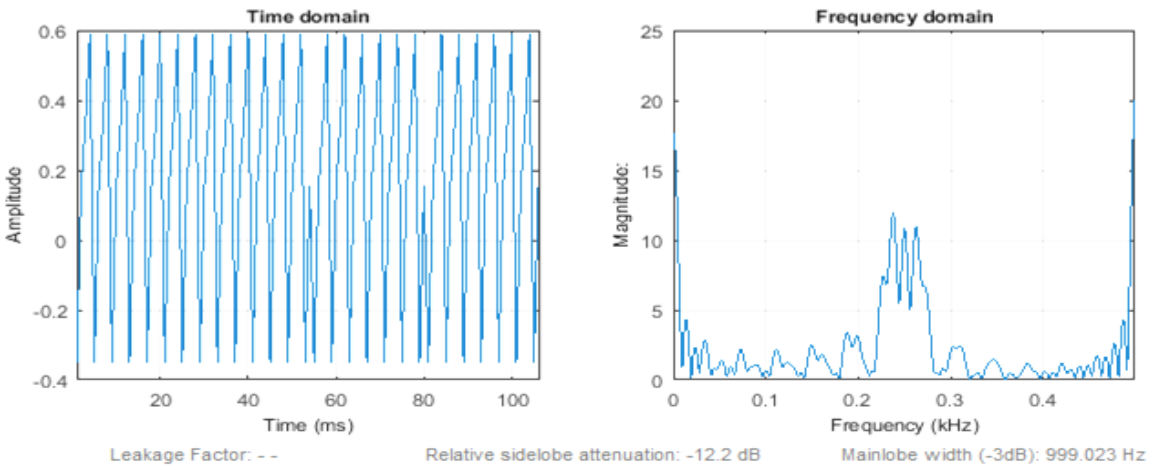

Figure 4.6: Mathematical Approach Pre-Fault FFT Current I(k) Signal Waveform

Figure 4.6 is the time and frequency – domain pre-fault FFT voltage signal waveform obtained by plotting the I_k against time t and frequency respectively. The waveform is not sinusoidal even at No fault condition. It also shows that the maximum amplitude of I_k is 0.6pu at time of 2msecs and magnitude of 13pu at 0.25kHz.

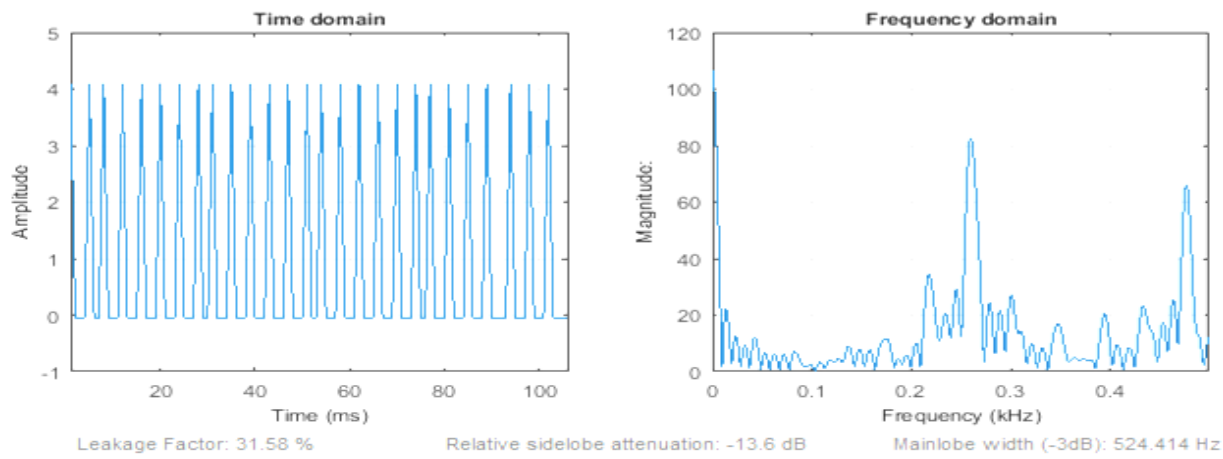

Figure 4.7: Mathematical Approach Three Phase Fault FFT Voltage (V_k) Signal Waveform

Figure 4.7is the time and frequency – domain three phase fault FFT voltage signal waveform obtained by plotting the V_k against time t and frequency respectively. The waveform is under three phase fault condition. It also shows that the maximum amplitude of V_k is 4pu at time of 2msecs and largest magnitude of 80pu at 0.25kHz.

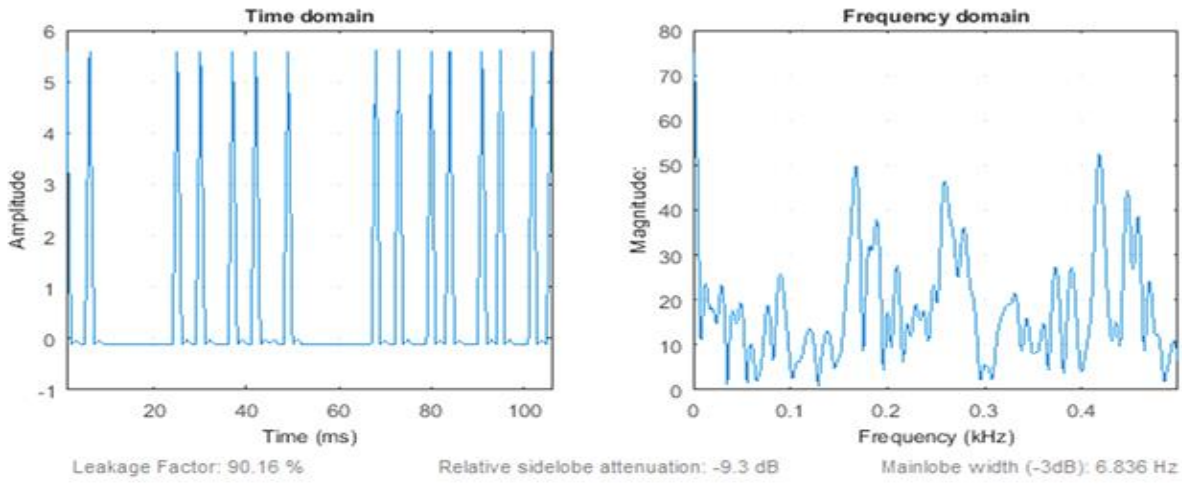

Figure 4.8: Mathematical Approach Three Phase Fault FFT Current I(k) Signal Waveform

Figure 4.8 is the time and frequency – domain three phase fault FFT current signal waveform obtained by plotting the I_K against time t and frequency respectively. It also shows that the maximum amplitude of I_K is 5.5pu at time of 2msecs and largest magnitude of 50pu at 0.25kHz.

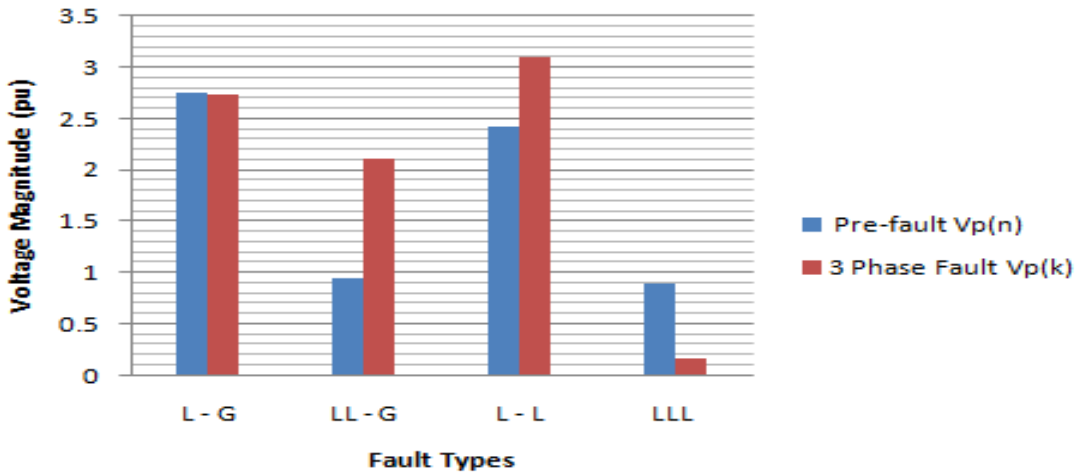

Figure 4.9: Mathematical Approach Pre-fault and Three Phase fault Voltages

Figure 4.9 show that pre-fault voltage is higher than that of three phase fault by 0.8pu. This is because of the occurrence of fault.

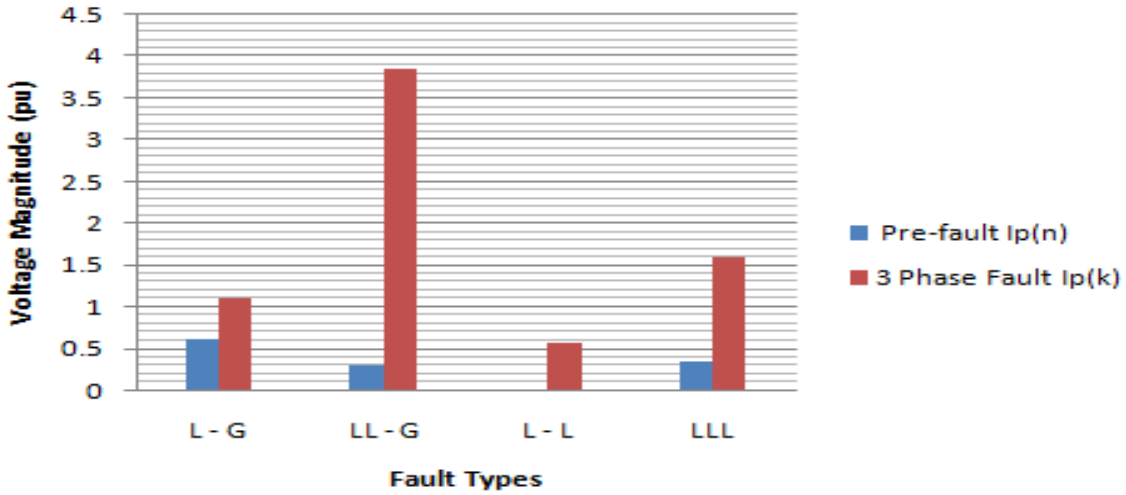

Figure 4.10: Mathematical Approach Pre-fault and Three Phase fault Currents

Here, the four three phase fault types have their currents higher than that of pre-fault. Three phase L – G fault is higher than it's pre-fault by 0.5pu, LL – G by 3.2pu, L – L by 0.5 and LLL by 1.2pu. This corresponds to the characteristics of fault, which is that, if fault occurs on the transmission line, the line current increases while the voltage will be reduced.

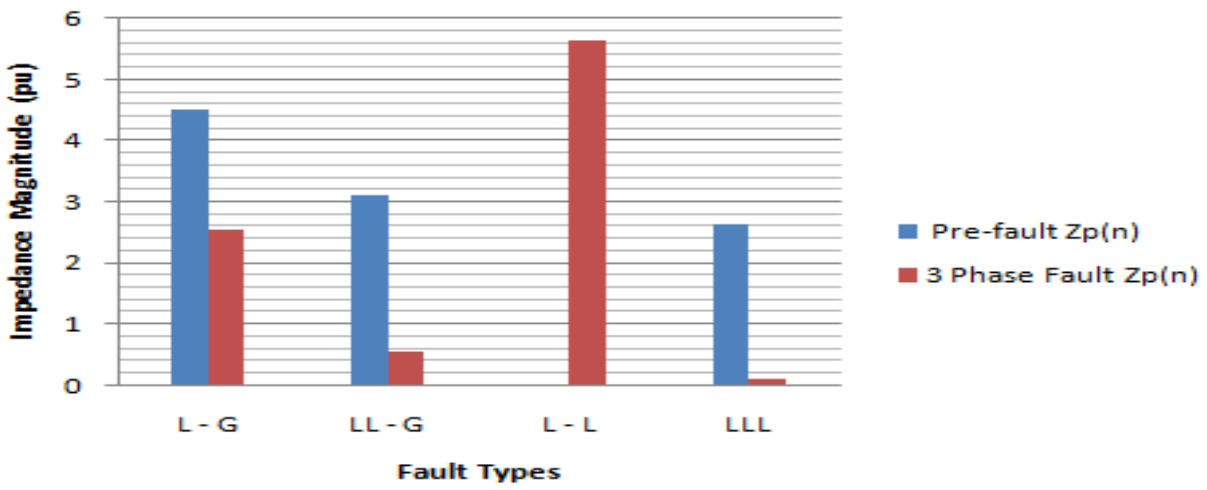

Figure 4.11: Mathematical Approach Pre-fault and Three Phase fault Impedances

Since the impedance of three phase fault is reduced to about 0.01pu from 2.6pu especially at three phase fault (LLL), that means the presence of fault reduces the impedance, voltage and increase the current.

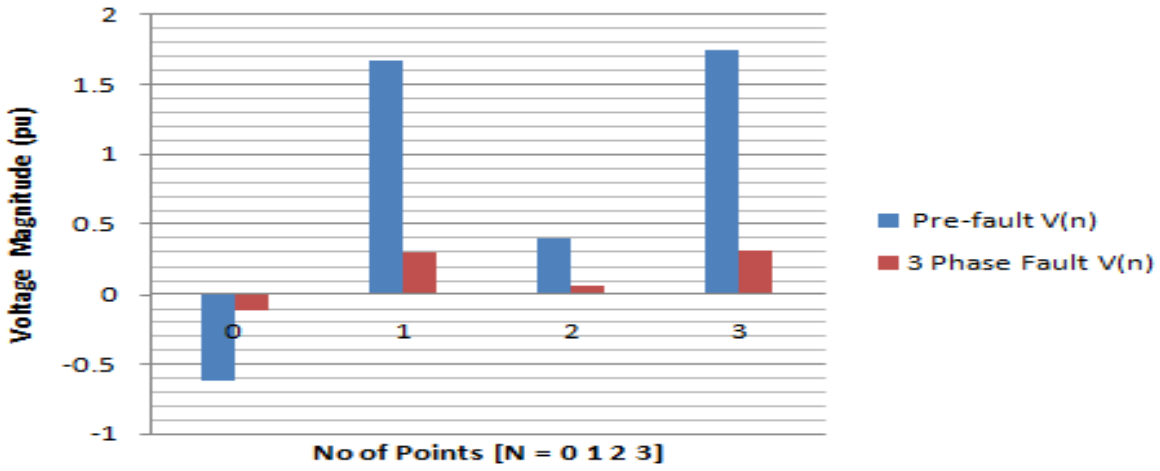

Figure 4.12: Mathematical Approach DFT Pre-fault and Three Phase fault Voltages

Figure 4.12 show that three phase fault voltage is lower than that of pre-fault by -0.4pu, 1.2pu, 0.4pu and 1.4pu at N = 0, 1, 2, and 3 respectively. This conformsto the fault characteristics.

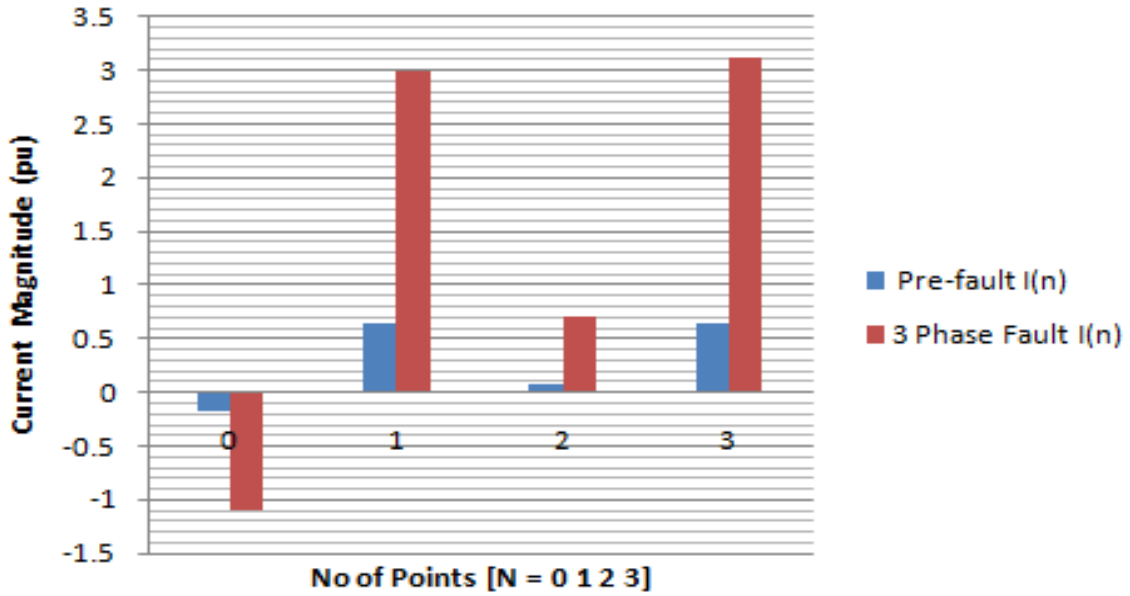

Figure 4.13: Mathematical Approach DFT Pre-fault and Three Phase fault Currents

Here, the three phase fault current is higher than that of pre-fault by by -0.9pu, 2.4pu, 0.7pu and 2.5pu at N = 0, 1, 2, and 3 respectively pu. This corresponds to the characteristics of fault, such that, under fault condition, the transmission line current increases while the voltage will be reduced.

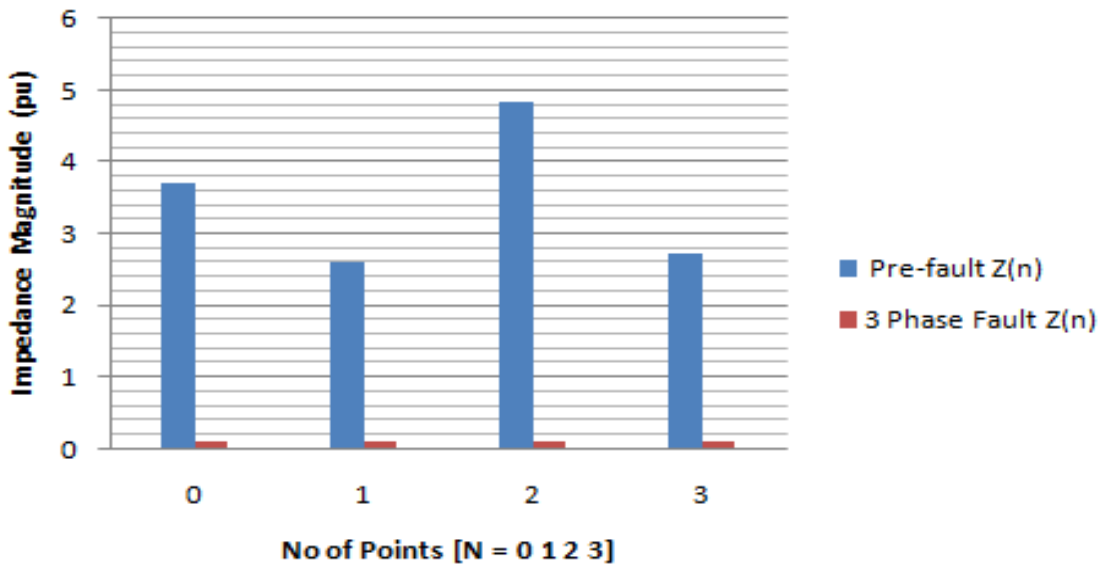

Figure 4.14: Mathematical Approach DFT Pre-fault and Three Phase fault Impedances

The impedance of three phase fault is reduced to about 0.01pu at all the N-points, whichmean the presence of fault reduces the impedance, voltage and increases the current.

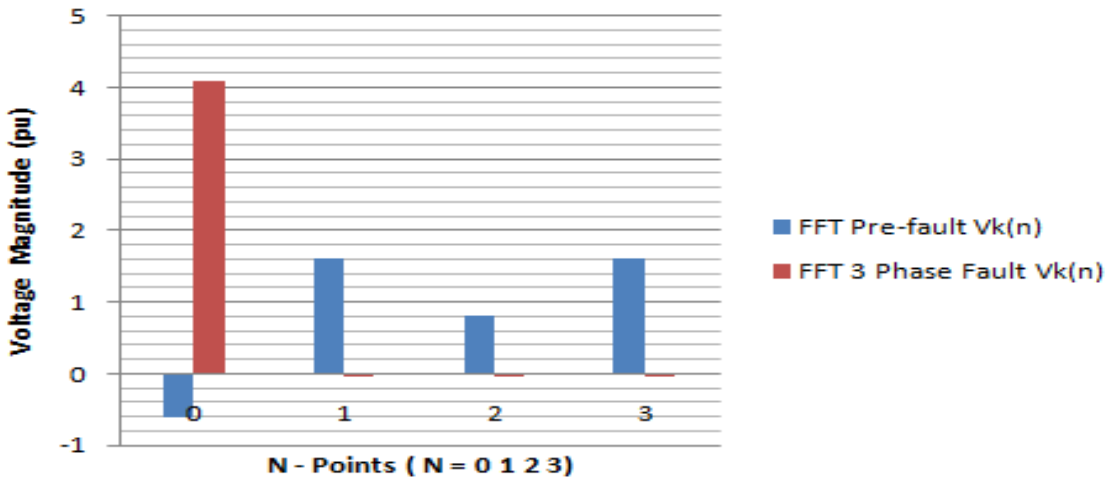

Figure 4.15: Mathematical Approach FFT Pre-fault and Three Phase fault Voltages

Here the pre-fault voltage is higher than the three phase faults by 1.6, 0.8 and 1.6 at N = 1, 2, and 3 respectively, but the three phase fault is higher than pre-fault by 3.7pu at N = 0.

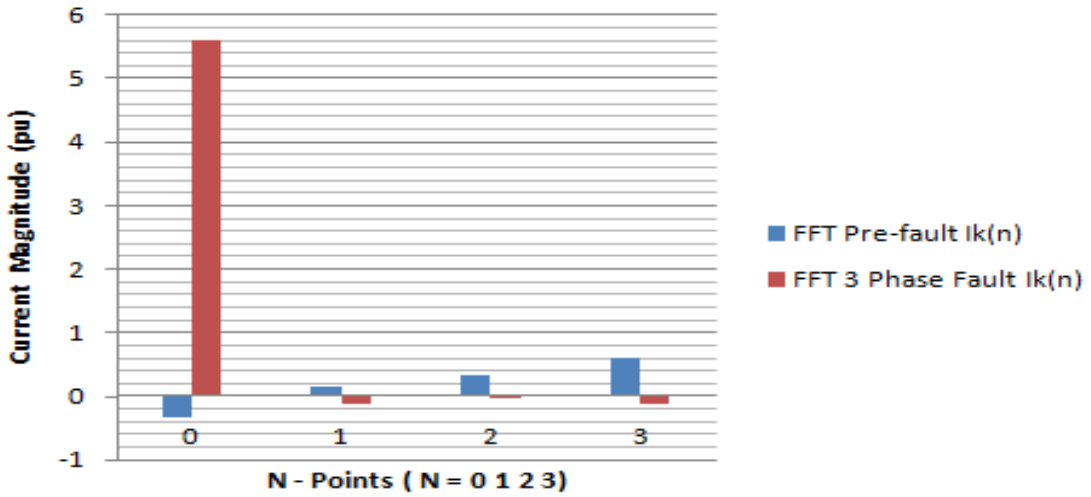

Figure 4.16: Mathematical Approach FFT Pre-fault and Three Phase fault Currents

Figure 4.16 show that three phase fault current is higher than that of pre-fault by 5.4pu only at N = 0 but are less than the pre-fault values at N = 1, 2 and 3. This can be accepted to conform to transmission line fault characteristics.

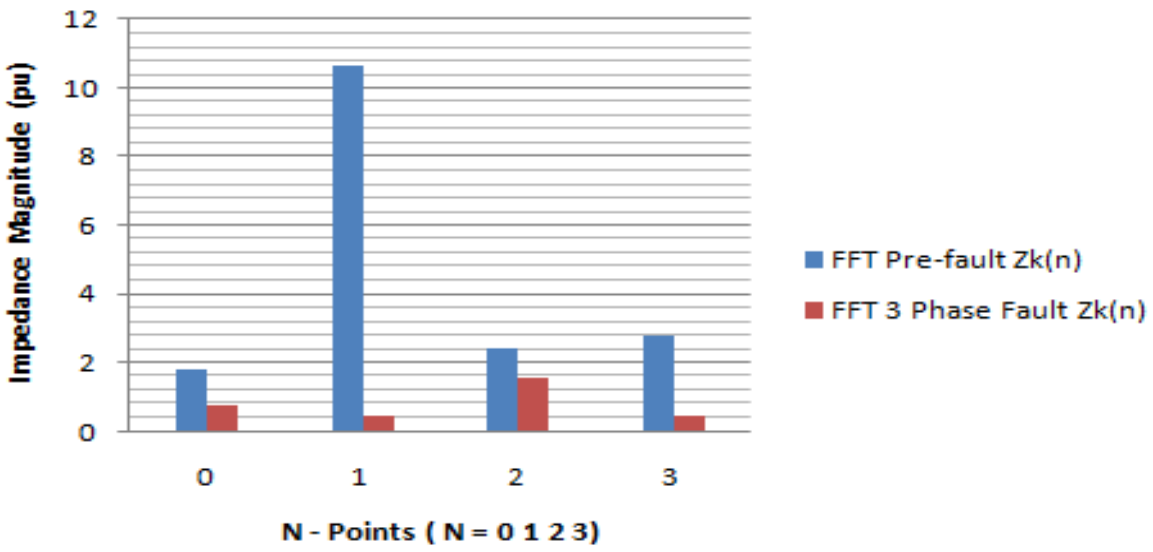

Figure 4.17: Mathematical Approach FFT Pre-fault and Three Phase fault Impedances

4.2 RESULTS OBTAINED THROUGH COMPUTER (SIMULATION) APPROACH

Table 4.7: Computer Approach Pre-fault discrete voltage, current and impedance values

S/N	$v_p(n)$	$i_p(n)$	$z_p(n)$	FAULTS
1	-3.4440	-0.2854	12.0673	L – G
2	-1.0310	-0.4488	2.2972	LL – G
3	-1.0310	-0.4488	2.2972	L – L
4	-1.0310	-0.4488	2.2972	LLL

Table 4.8: Computer Approach Three phase fault discrete voltage, current and impedance values

S/N	$v_p(n)$	$i_p(n)$	$z_p(n)$	FAULTS
1	-1.7430	-4.2862	0.4067	L – G
2	-0.1310	-2.4183	0.0054	LL – G
3	-0.6540	-1.2438	0.5258	L – L
4	-0.0550	-0.9488	0.0579	LLL

Table 4.9: Computer Approach Three phase Post fault discrete voltage, current and impedance values

S/N	$v_p(n)$	$i_p(n)$	$z_p(n)$	FAULTS
1	-1.7430	-4.2862	0.4067	L – G
2	-0.1310	-2.4183	0.0054	LL – G
3	-0.6540	-1.2438	0.5258	L – L
4	-0.0550	-0.9488	0.0579	LLL

Table 4.7, 4.8 illustrate the discrete values (V_p and I_p) obtained for pre-fault and fault conditions under four categories of fault, one symmetrical and three unsymmetrical faults.

The DFT is applied to the discrete values following the methodological procedure stated in chapter three and using equation 3.51 and 3.52 to determine the time – domain and frequency – domain components which contain the information such as fault information about the transmission line.

Table 4.9: Computer Approach DFT Application for Pre-fault Condition at N – POINTS (k = 0,…, N – 1)

S/N	**N - POINTS**	$\mathbf{V_{0,1,2,3}}$ for N = 4	$\mathbf{I_{0,1,2,3}}$ for N = 4	V(n)	I(n)	Z(n)	FAULTS
1	0	-2.4360	-0.2018	-0.6422	0.2790	-2.3017	LLL
2	1	0.0144	0.0063	-2.1690	-0.0860	25.2209	LLL
3	2	1.0300	0.3261	-2.7310	-0.3040	8.9836	LLL
4	3	1.0300	0.4484	-4.2000	-0.7010	5.9914	LLL

Table 4.10: Computer Approach DFT Application for Three Phase Fault Condition at N – POINTS (k = 0,…, N – 1)

S/N	**N - POINTS**	$\mathbf{V_{0,1,2,3}}$ for N = 4	$\mathbf{I_{0,1,2,3}}$ for N = 4	V(n)	I(n)	Z(n)	FAULTS
1	0	-2.4340	-0.2024	0.5777	-0.6412	-1.1099	LLL
2	1	0.0144	0.0063	-0.8650	-2.1680	2.5064	LLL
3	2	1.0300	0.3258	-0.3309	-2.7300	8.2502	LLL
4	3	1.0300	0.4480	-0.9699	-4.1990	4.3293	LLL

Table 4.9: Computer Approach DFT Application for Post-fault Condition at N – POINTS (k = 0,…, N – 1)

S/N	**N - POINTS**	$\mathbf{V_{0,1,2,3}}$ for N = 4	$\mathbf{I_{0,1,2,3}}$ for N = 4	V(n)	I(n)	Z(n)	FAULTS
1	0	-2.4360	-0.2018	-0.5122	0.2350	-2.1796	LLL
2	1	0.0144	0.0063	-2.1270	-0.1460	14.5685	LLL
3	2	1.0300	0.3261	-2.5000	-0.2700	9.2593	LLL
4	3	1.0300	0.4484	-4.1400	-0.6710	6.1699	LLL

Table 4.11: Computer Approach FFT Application for Pre-fault Condition at N – POINTS (k = 0,…, N – 1)

S/N	**N - POINTS**	$\mathbf{DFTV_{0,1,2,3}}$ for N = 4	$\mathbf{DFTI_{0,1,2,3}}$ for N = 4	FFT V(k)	FFT I(k)	FFT Z(k)	FAULTS
1	0	-2.4360	-0.2018	-0.6422	0.5790	-0.3718	LLL
2	1	0.0144	0.0063	-2.1690	-0.8581	2.5277	LLL
3	2	1.0300	0.3261	-0.6711	0.5664	-1.1849	LLL
4	3	1.0300	0.4484	-3.4650	-0.6503	5.3283	LLL

Table 4.12: Computer Approach FFT Application for Three Phase Fault Condition at N – POINTS (k = 0,…, N – 1)

S/N	**N - POINTS**	**$DFTV_{0,1,2,3}$** for N = 4	**$DFTI_{0,1,2,3}$** for N = 4	FFT V(k)	FFT I(k)	FFT Z(k)	FAULTS
1	0	-2.4340	-0.2024	-0.2414	2.5777	0.0936	LLL
2	1	0.0144	0.0063	-1.1680	-1.7465	0.6688	LLL
3	2	1.0300	0.3258	-0.0303	0.5651	-0.0536	LLL
4	3	1.0300	0.4480	-1.440	-2.6504	0.5433	LLL

Table 4.13: Computer Approach FFT Application for Post-fault Condition at N – POINTS (k = 0,…, N – 1)

S/N	**N - POINTS**	**$V_{0,1,2,3}$** for N = 4	**$I_{0,1,2,3}$** for N = 4	V(k)	I(k)	Z(k)	FAULTS
1	0	-2.4360	-0.2018	-0.5332	0.1520	-3.5079	LLL
2	1	0.0144	0.0063	-2.2490	-0.0780	28.8333	LLL
3	2	1.0300	0.3261	-0.6310	-0.3120	8.4327	LLL
4	3	1.0300	0.4484	-4.1720	-0.6580	6.3404	LLL

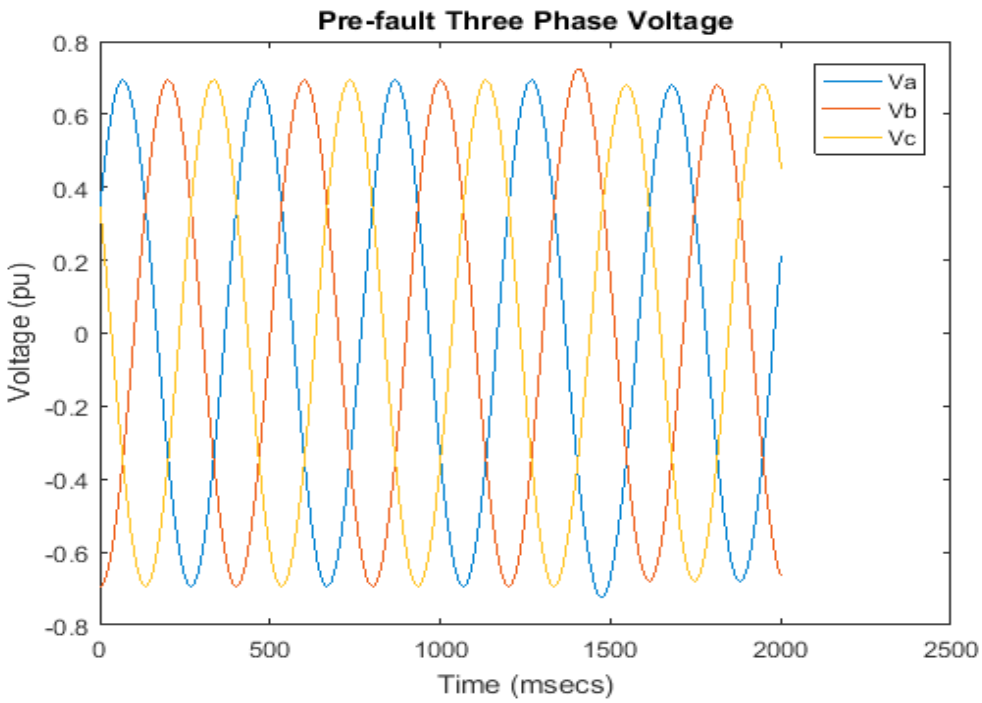

Figure 4.18: Computer Approach Pre-faultThree Phase Voltage Waveform

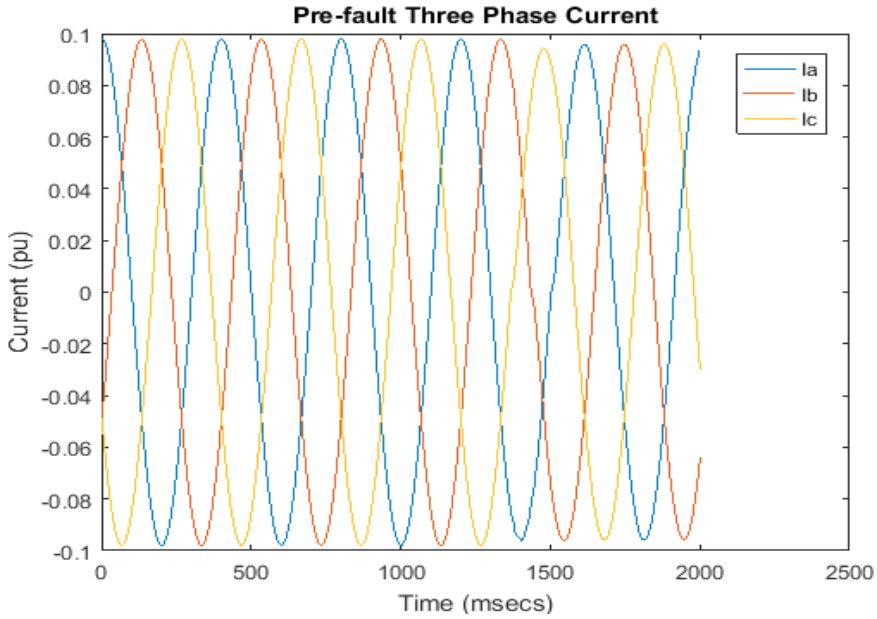

Figure 4.19: Computer Approach Pre-fault Three Phase Current Waveform

Figures 4.18 and 4.19 are three phase voltage and current graph showing the initial (No fault) condition of the transmission line for the three phase fault occurred on it.

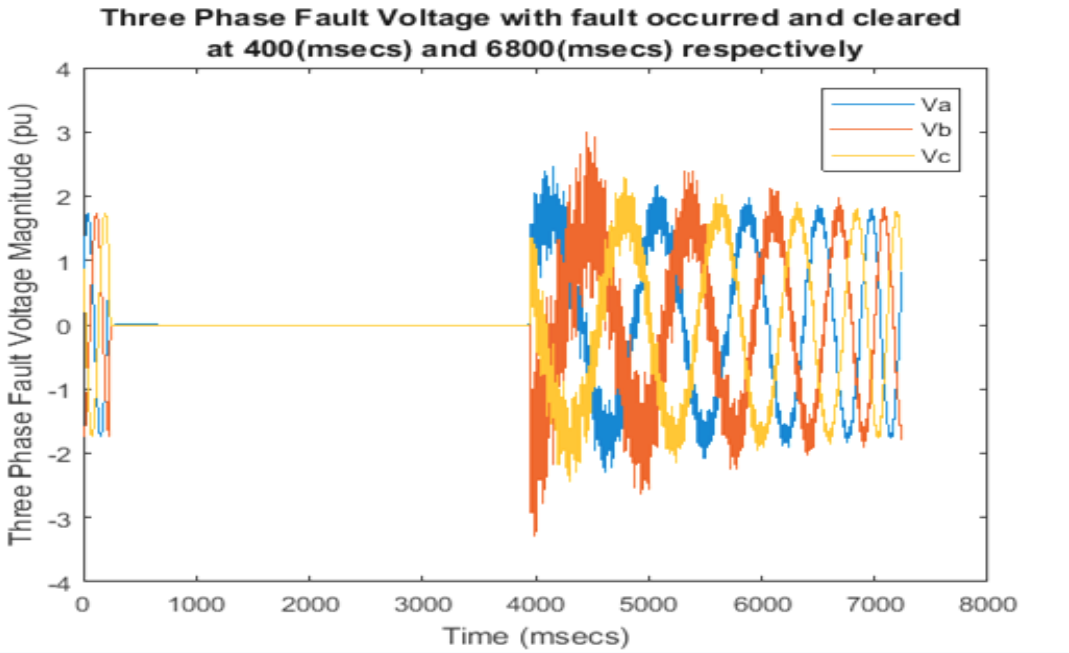

Figure 4.20: Computer Approach Three Phase Fault Voltage Waveform with fault voltage occurred and cleared at 400msecs and 4000msecs respectively.

Here, the post-fault voltage signal is not purely sinusoidal it still contains some harmonics caused by the three phase fault. The effect continued from 4000msecs with voltage magnitude of 2.0pu to 3.0pu and finally 2.0pu, but later returned sinusoidal and never to its original signal value of 1.8pu.

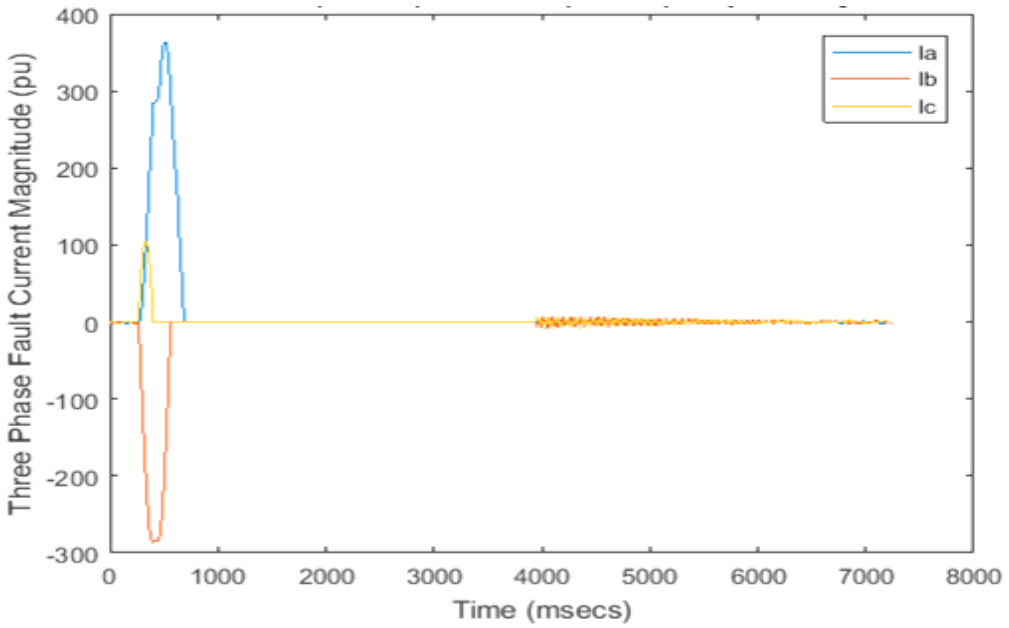

Figure 4.21: Computer Approach Three Phase Fault Current Waveform with 380pu fault current magnitude occurred and cleared at 400msecs and 4000msecs respectively.

Here, the effect of three phase fault is obviously seen that the current magnitude is as large as 380pu, though the high harmonics later decayed when the fault was cleared at 4000msecs.

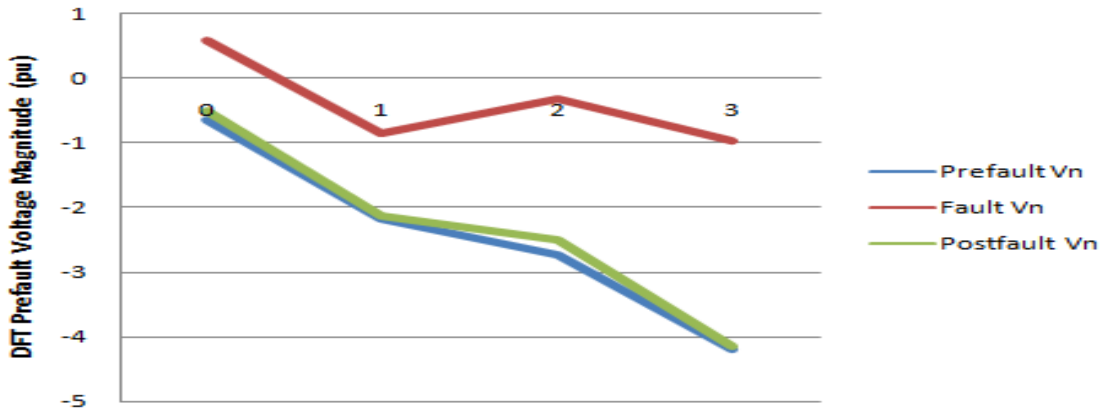

Figure 4.22: Computer Approach DFT Pre-fault, Fault and Post-Fault Voltage Relationship

DFT Pre-fault, three phase fault and post-fault voltage signal were plotted to show the variations between the three states of voltage and similarity between the pre-fault and post-fault voltage.

From the figure 4.22, it is observed that both the pre-fault and post-fault voltage signals are almost equal at -0.5pu as against 0.6pu when fault occurred.

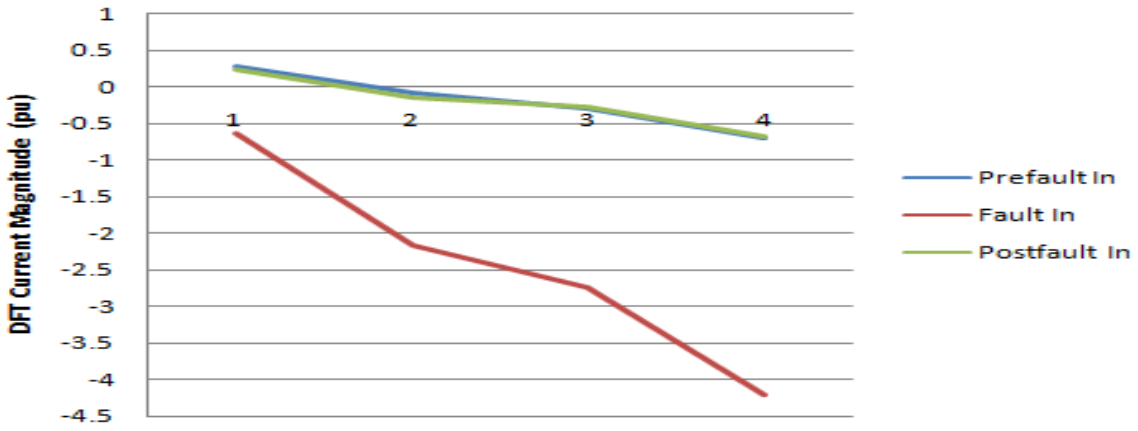

Figure 4.23: Computer Approach DFT Pre-fault, Fault and Post-Fault Current Relationship

The DFT pre-fault, three phase fault and post-fault current signal were also plotted to show the variations between the three states of current and similarity between the pre-fault and post-fault voltage.

It can be seen from the figure 4.23 that both pre-fault and post-fault currents are almost the same with 0.3pu as against -0.6pu fault value.

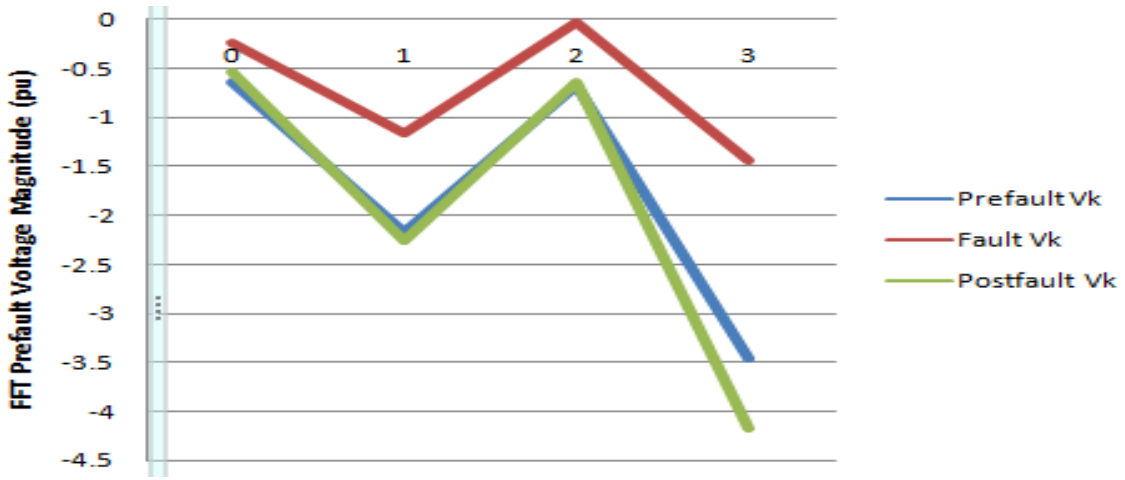

Figure 4.24: Computer Approach FFT Pre-fault, Fault and Post-Fault Voltage Relationship

The FFT pre-fault, three phase fault and post-fault voltage signal were also plotted to show the variations between the three states of voltage and similarity between the pre-fault and post-fault voltage.

Figure 4.24 show that both pre-fault and post-fault voltages are almost the same with but with little difference of 0.01pu against the fault value of -0.3pu.

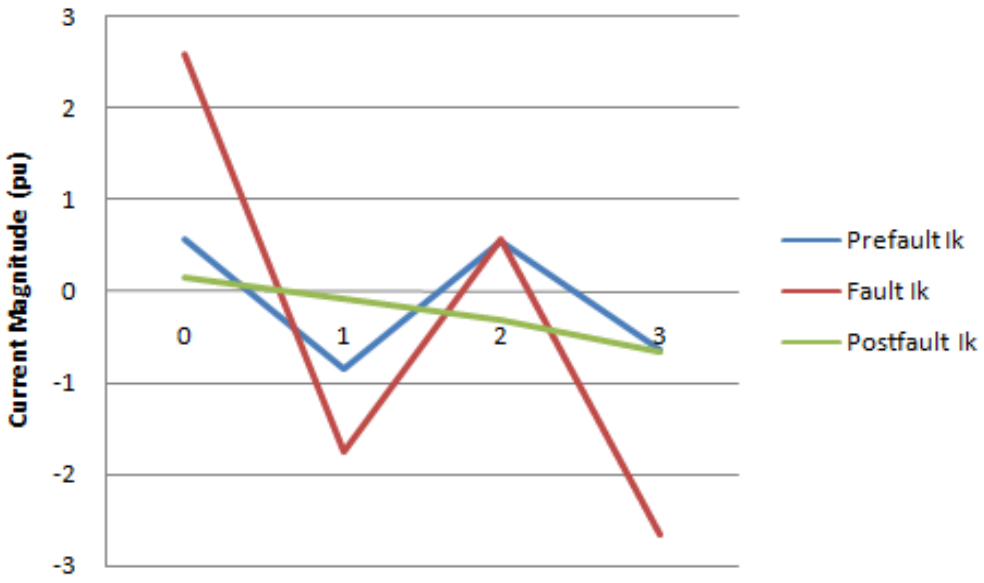

Figure 4.25: Computer Approach FFT Pre-fault, Fault and Post-Fault Current Relationship

It can be seen from the figure 4.25 that both pre-fault and post-fault currents are not the same after fault clearing. The post-fault current signal is 0.1pu but continue to decrease as number of point N increases.

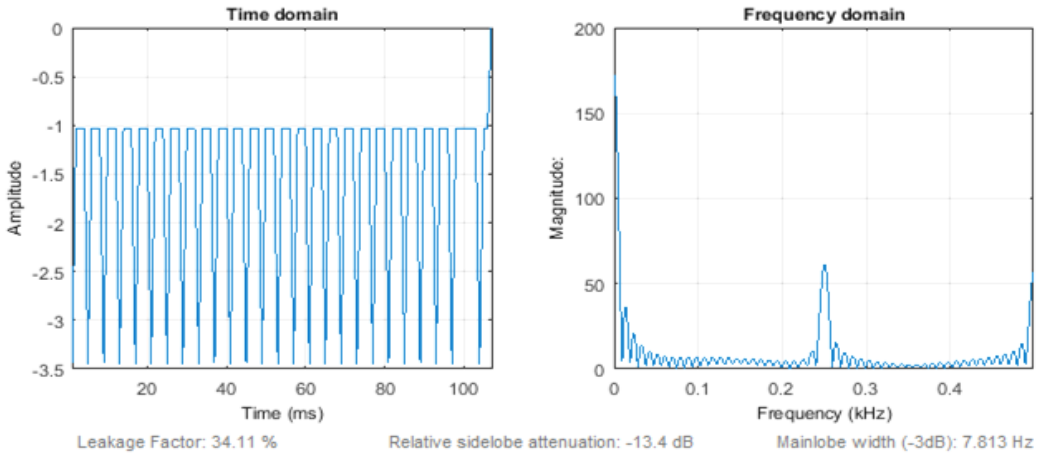

Figure 4.26: Computer Approach Pre-Fault DFT Voltage (Vn) Signal Waveform

Figure 4.26is the time and frequency – domain pre-fault DFT voltage signal waveform obtained by plotting the V_n against time t and frequency respectively. The waveform is not sinusoidal even at No fault condition. It also shows that the maximum amplitude of V_n is -0.6pu maximum and -3.5 minimum at time of 2msecs and largest magnitude of 70pu at 0.25kHz sampling frequency.

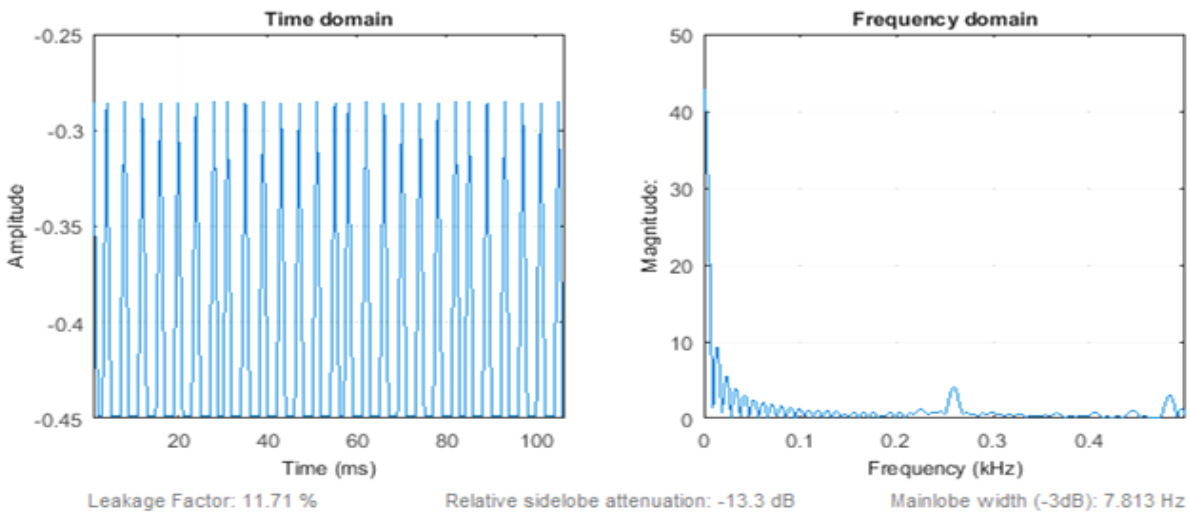

Figure 4.27: Computer Approach Pre-Fault DFT Current (In) Signal Waveform

Figure 4.27 is the time and frequency – domain pre-fault DFT current signal waveform obtained by plotting the I_n against time t and frequency respectively. It also shows that the maximum amplitude of I_n is -0.25pu maximum and -0.45 minimum at time of 2msecs and largest magnitude of 5pu at 0.25kHz.

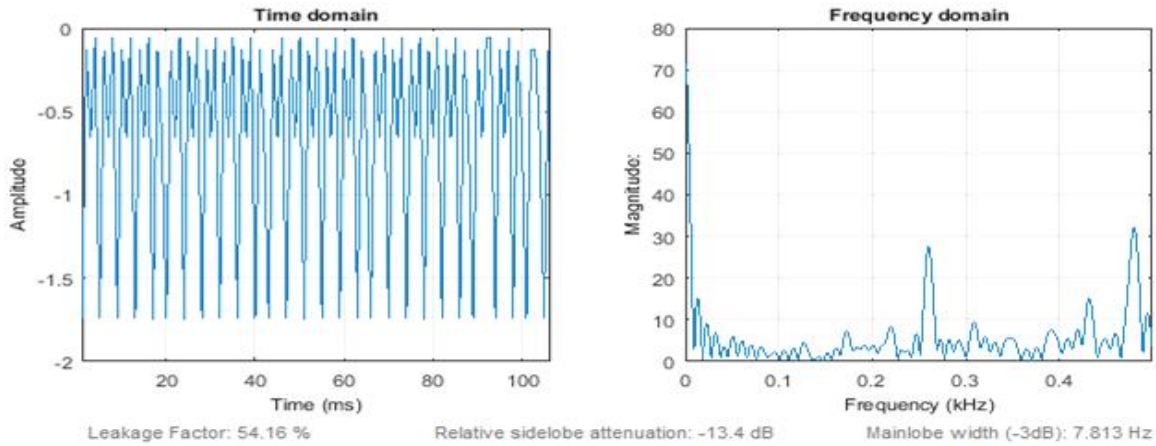

Figure 4.28: Computer Approach Three Phase Fault DFT Voltage (Vn) Signal Spectrum Waveform

Figure 4.28 is the time and frequency – domain three phase fault DFT voltage signal waveform obtained by plotting the V_n against time t and frequency respectively. It also shows that the maximum amplitude of V_n is -1.75pu at time of 2msecs and largest magnitude of 28pu at 0.25kHz.

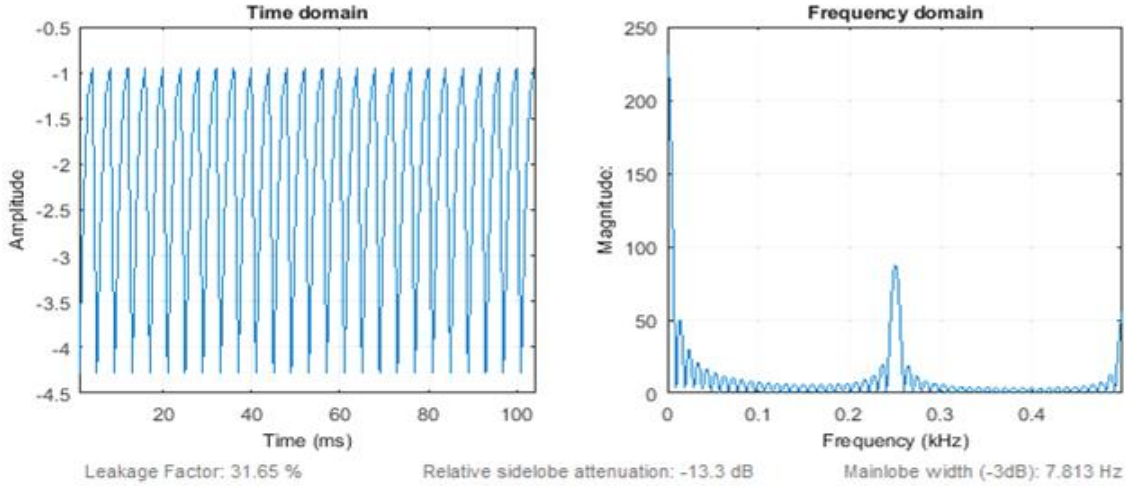

Figure 4.29: Computer Approach Three Phase Fault DFT Current I(n) Signal Spectrum Waveform

Figure 4.29 is the time and frequency – domain three phase fault DFT current signal waveform obtained by plotting the I_n against time t and frequency respectively. It also shows that the maximum amplitude of I_n is -4.25pu at time of 2msecs and largest magnitude of 80pu at 0.25kHz.

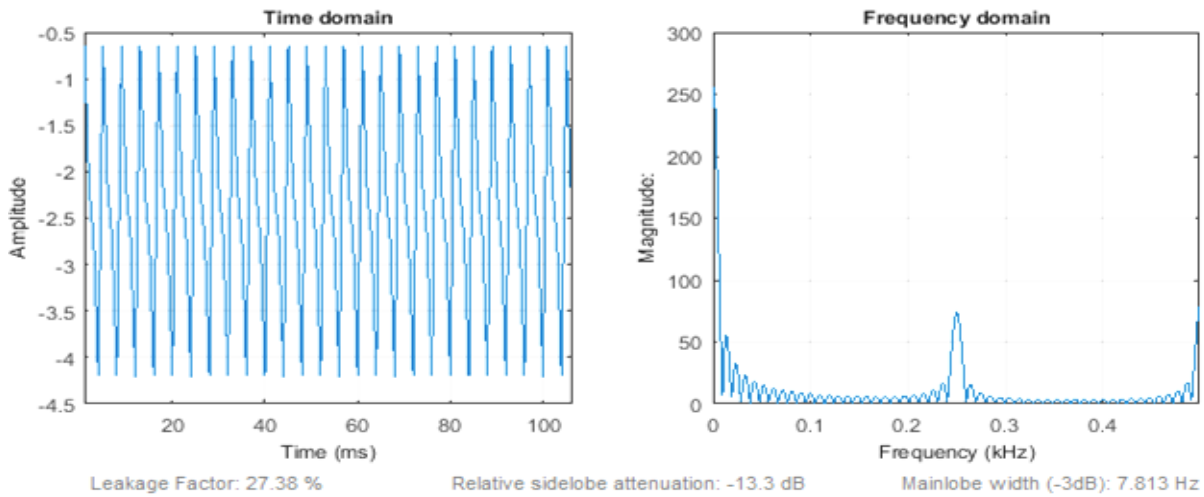

Figure 4.30: Computer Approach Pre-Fault FFT Voltage V(k) Signal Spectrum Waveform

Figure 4.30 is the time and frequency – domain pre-fault FFT voltage signal waveform obtained by plotting the V_k against time t and frequency respectively. It also shows that the maximum amplitude of V_k is -4.25pu at time of 2msecs and largest magnitude of 75pu at 0.25kHz.

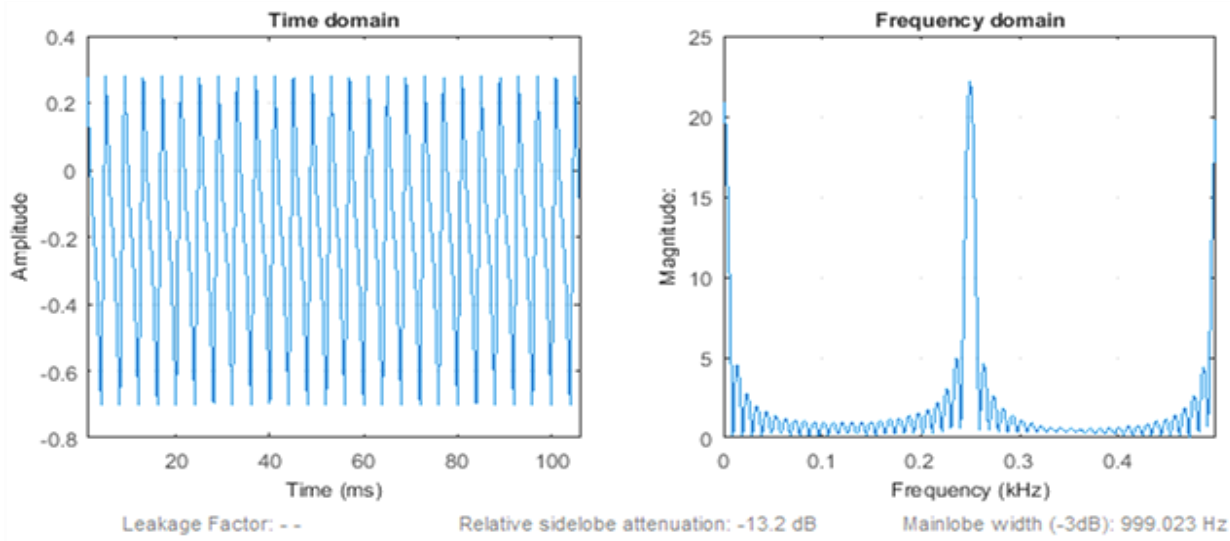

Figure 4.31: Computer Approach Pre-Fault FFT Current I(k) Signal Spectrum Waveform

Figure 4.31 is the time and frequency – domain pre-fault FFT current signal waveform obtained by plotting the I_k against time t and frequency respectively. It also shows that the maximum amplitude of I_k is 0.3pu at time of 2msecs and largest magnitude of 22.25pu at 0.25kHz.

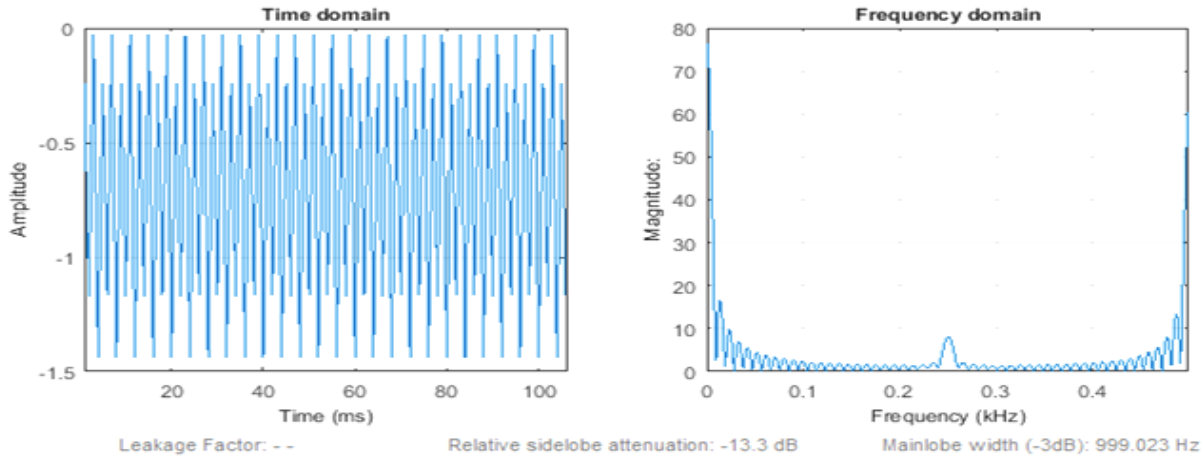

Figure 4.32: Computer Approach Three Phase Fault FFT Voltage V(k) Signal Spectrum Waveform

Figure 4.32 is the time and frequency – domain three phase fault FFT voltage signal waveform obtained by plotting the V_k against time t and frequency respectively. It also shows that the maximum amplitude of V_k is -1.5pu at time of 2msecs and largest magnitude of 10pu at 0.25kHz.

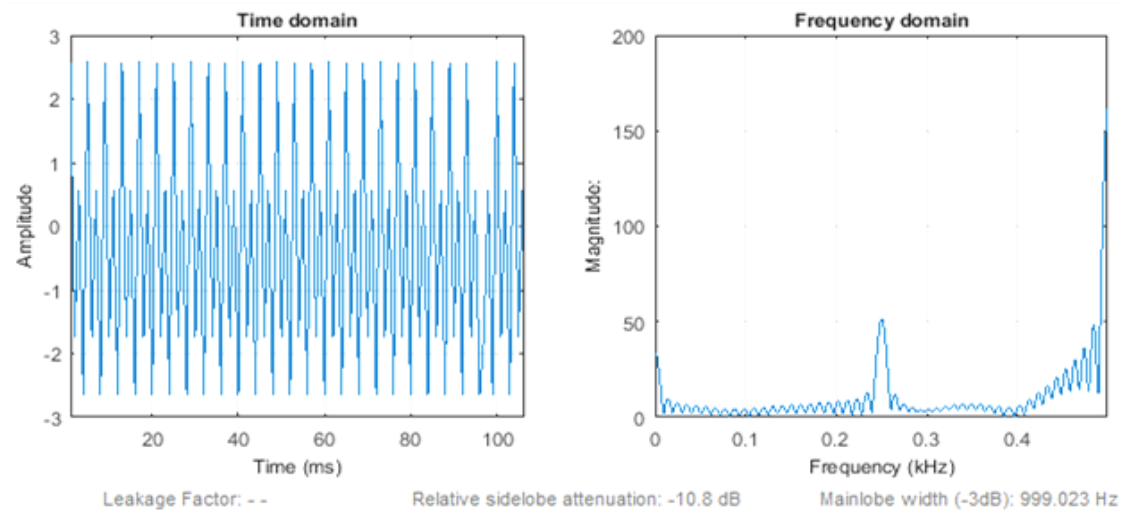

Figure 4.33: Computer Approach Three Phase Fault FFT Current I(k) Signal Spectrum Waveform

Figure 4.33is the time and frequency – domain three phase fault FFT current signal waveform obtained by plotting the I_k against time t and frequency respectively. It also shows that the maximum amplitude of I_k is 2.8pu at time of 2msecs and largest magnitude of 50pu at 0.25kHz.

According to Robi P. (2013) and Capolino et al (2003) in chapter two of this research, the frequency spectrum of any signal is basically the frequency components of that signal. It shows that the frequencies of that signal tell more about the condition of the signal (healthy or faulty conditions). He continued that, whenever a waveform or its magnitude or frequency components changes from its original form of normal healthy condition, it then means that the signal whose

frequency components changes is not healthy and has been affected by unwanted disturbance which is referred to as fault in power system.

However, such signals such as voltage and current fault signals of a transmission line obtained when the DFT or FFT algorithm is applied using mathematical or computer (simulation) approachesand contain fault information that describes the faulty condition of the transmission line. Thus, the mathematical and computer algorithm has been used to detect three phase fault occurred on the transmission line.

Comparing the pre-fault and three phase fault DFT and FFT signal waveforms with respect to tables 4.3, 4.4, 4.5, 4.6, …, 4.12, one can easily observe an increase in the amplitude or magnitude of the three phase fault current signals as againstthe amplitude or magnitude of the three phase fault voltage signal. This is opposite of what we have in the amplitude or magnitude of the pre-fault voltage and current signals.

For instance, using mathematical approach, DFT pre-fault voltage gives 14pu while its current gives 0.6pu. But DFT three phase fault voltage gives 2.0pu while that of current gives 2.8pu. The FFT pre-fault voltage gives 1.6pu while its current is 1.25pu. But FFT three phase fault voltage gives 1.6pu while its current is 2.4pu.

This is in conformity with the characteristics of fault on the power system, were the increase in DFT and FFT fault current magnitude and decrease in the voltage magnitude is because of the presence of fault on the line which caused increase in current, thus over-current fault (three phase fault).

While using computer(simulation) approach, DFT pre-fault voltage gives -0.60pu maximum and -4.2pu minimum while its current gives 0.6pu maximum and -0.9pu minimum. But DFT three phase fault (three phase fault) voltage gives -0.57pu maximum and -4.15pu minimum.

The same is observed in DFT discretized voltage and current magnitudes. Where Vn = [V0 V1 V2 V3]^T gives varied values and have V_3= 7.0pu which is the largest magnitude among other N – point values with I_3 = 1.0pu. But when DFT is applied for three phase fault simulation, it gave a fault current magnitude of I_3 = -400pu which is very large current magnitude.

However, FFT applied for pre-fault condition gave V2 = 5pu as the largest magnitude and current I_2 = 0.75pu. While when simulated with three phase fault gives V(k) = [V0 V1 V2 V3] = 0, I_3 = -320pu and I_2 = 220pu.

This sharp increase is an evidence of fault. Thus, due to the presence of fault on the transmission line, the transmission line under fault condition is characterized by the increase in current magnitude greater than the voltage magnitude making the impedance smaller than the relay preset value. On this condition, the relay will send a tripping signal to the circuit breaker to isolate the faulty section of the line.

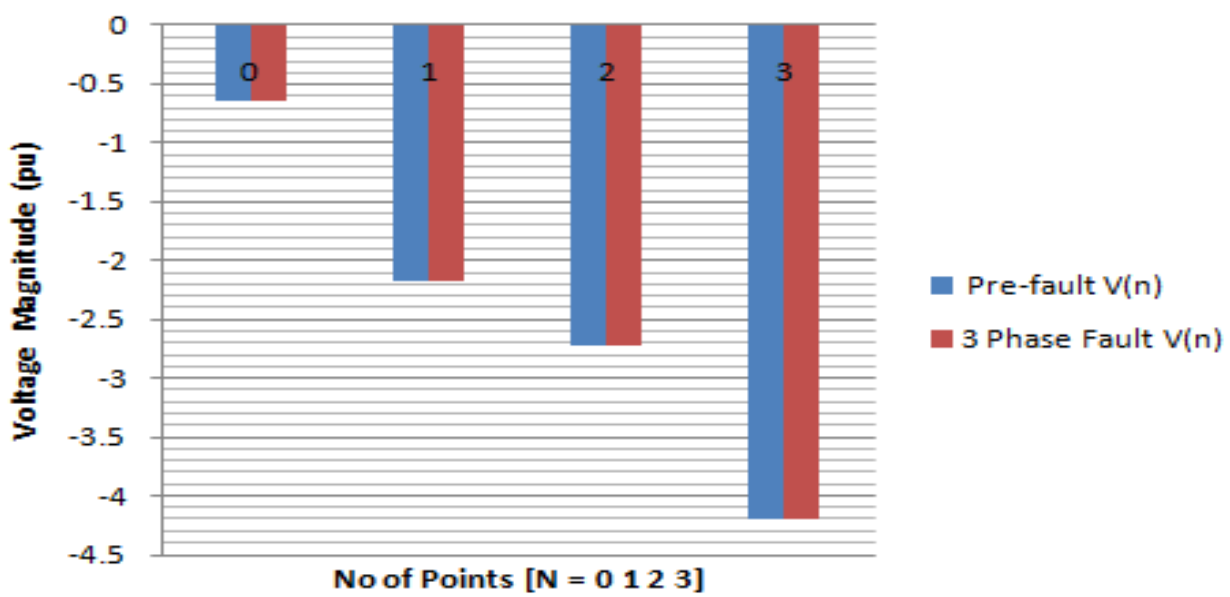

Figure 4.34: Computer Approach DFT Pre-fault and Three Phase fault Voltages

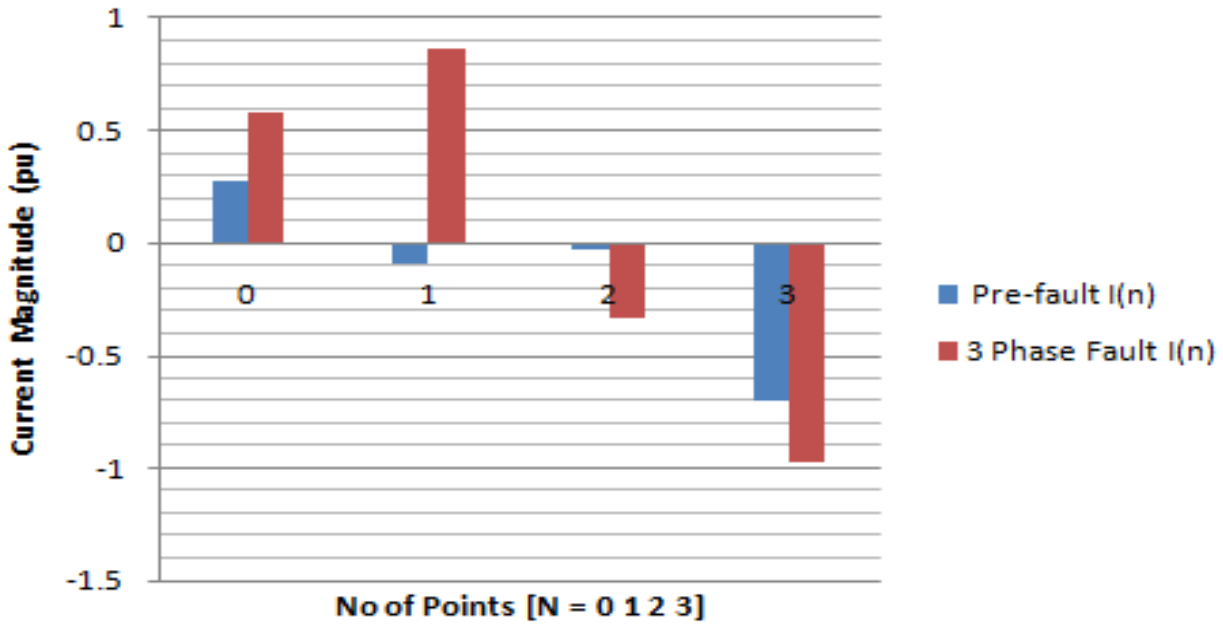

Figure 4.35: Computer Approach DFT Pre-fault and Three Phase fault Currents

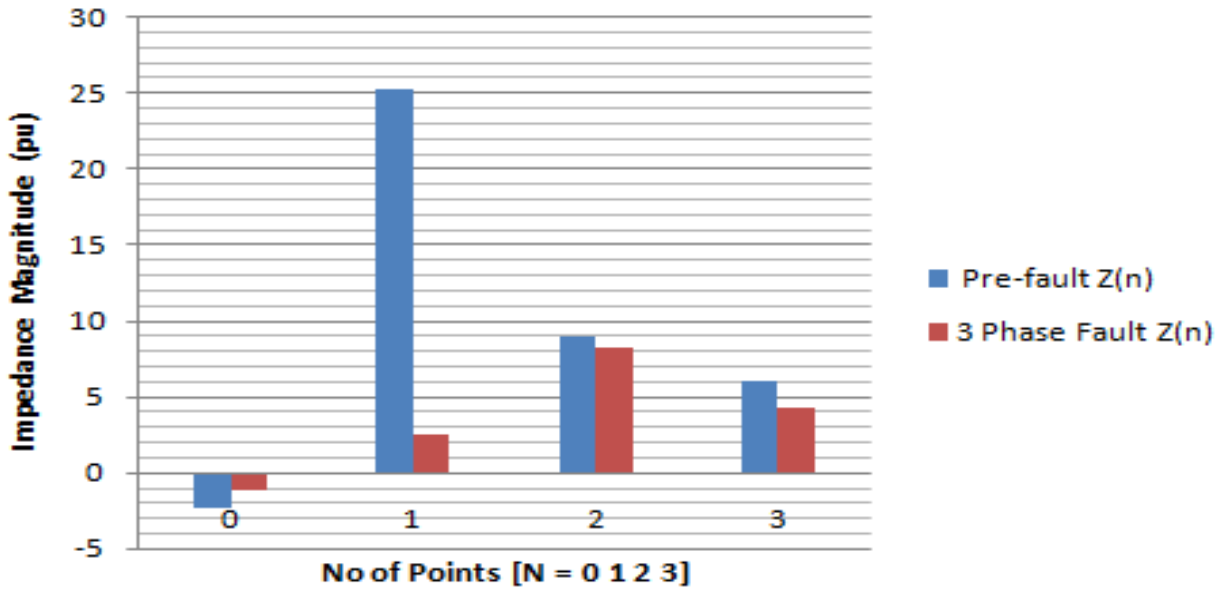

Figure 4.36: Computer Approach DFT Pre-fault and Three Phase fault Impedances

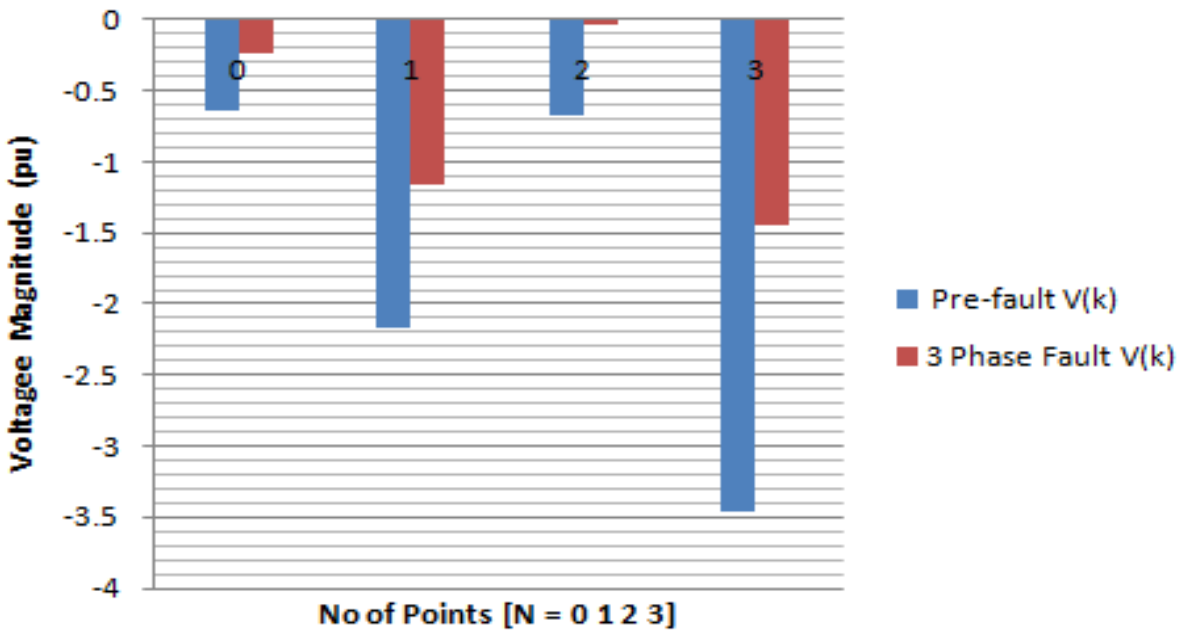

Figure 4.37: Computer Approach FFT Pre-fault and Three Phase fault Voltages

Here, the magnitude of voltage for both pre-fault and three phase fault is equal.

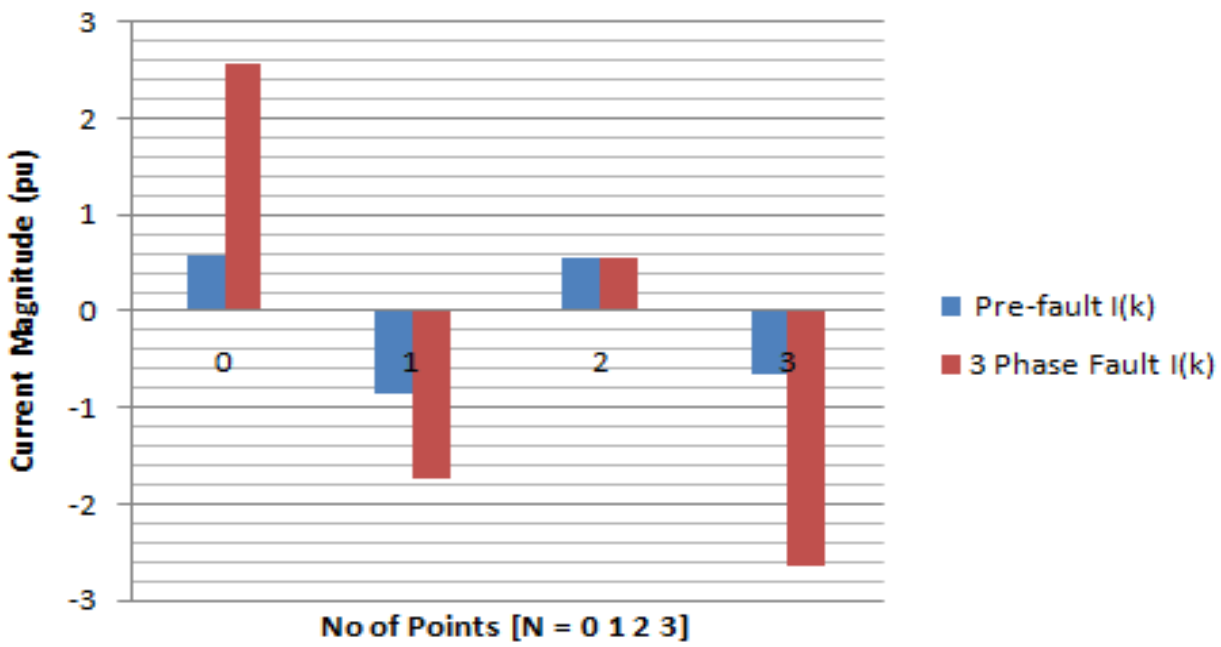

Figure 4.38: Computer Approach FFT Pre-fault and Three Phase fault Currents

The pre-fault and three phase current is almost equal except on N = 1, where the pre-fault current higher than that of three phase fault by 0.6pu.

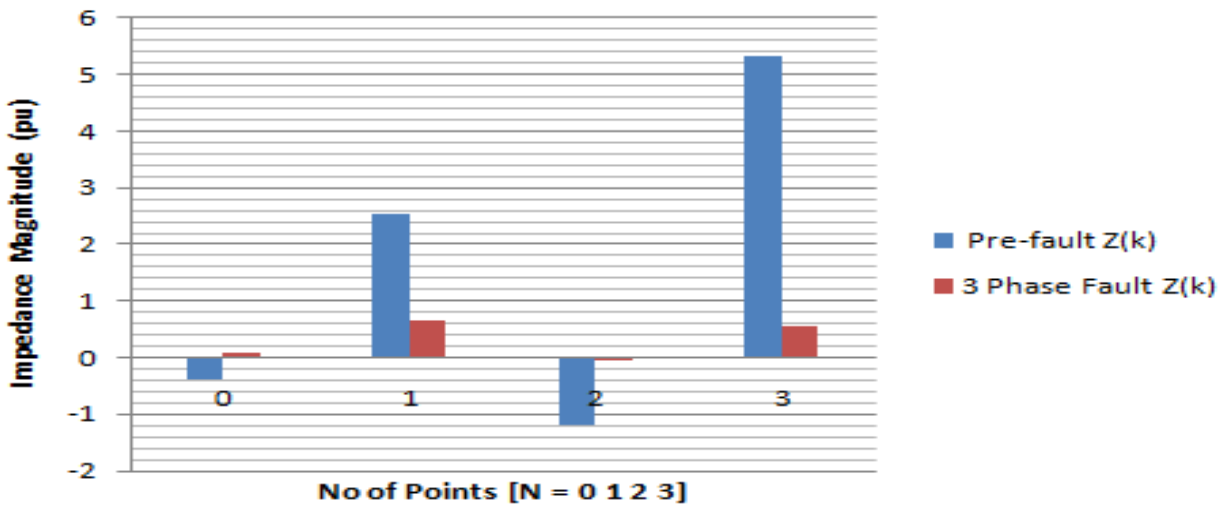

Figure 4.39: Computer Approach FFT Pre-fault and Three Phase fault Impedances

The three phase fault is supposed to be lesser than that of pre-fault value but here it is far higher than that of pre-fault.

4.3 Percentage Error

The percentage error is between the DFT and FFT pre-fault results and that of three phase fault results for computer and mathematical approaches and is determined using the values on the tables 4.1 to 4.12. The percentage error can be determined using equation (4.1).

$$\% \text{ Error} = \frac{\text{Computer value} - \text{Mathematical value}}{\text{Mathematical value}} \, x \, 100 \qquad (4.1)$$

Table 4.14: Percentage error between mathematical and simulation approachesDFT pre-faults voltage, current and impedance values

S/NO	N - POINTS	V(n) (%)	I(n) (%)	Z(n) (%)	FAULT
1	0	-2.50	41.20	-0.06	LLL
2	1	-49.57	45.76	-1.08	LLL
3	2	-207.00	24.92	-8.31	LLL
4	3	-24.54	32.40	-0.76	LLL

Table 4.15: Percentage error between mathematical and simulation approaches DFT three phase fault voltage, current and impedance values

S/NO	N - POINTS	V(n) (%)	I(n) (%)	Z(n) (%)	FAULT
1	0	53.25	53.00	1.00	LLL
2	1	187.39	290.34	0.65	LLL
3	2	6.91	37.44	0.18	LLL
4	3	38.93	221.40	0.18	LLL

Here, the error means difference between the two components. Large difference means large error and large difference is as a result of increase in signal magnitude

due to presents of fault. The magnitude of DFT voltage is larger than that of FFT. Thus, DFT result here is more pronounced.

Table 4.16: Percentage error between mathematical and simulation approaches FFT pre-faults voltage, current and impedance values

S/NO	N - POINTS	V(k) (%)	I(k) (%)	Z(k) (%)	FAULT
1	0	2.31	21.97	0.11	LLL
2	1	52.07	70.60	0.74	LLL
3	2	13.88	16.93	0.82	LLL
4	3	184.67	0.60	307.78	LLL

Table 4.17: Percentage error between mathematical and simulation approaches FFT three phase fault voltage, current and impedance values

S/NO	N - POINTS	V(k) (%)	I(k) (%)	Z(k) (%)	FAULT
1	0	344.34	502.15	0.69	LLL
2	1	212.17	2.51	84.53	LLL
3	2	62.57	53.61	1.17	LLL
4	3	300.00	53.88	5.57	LLL

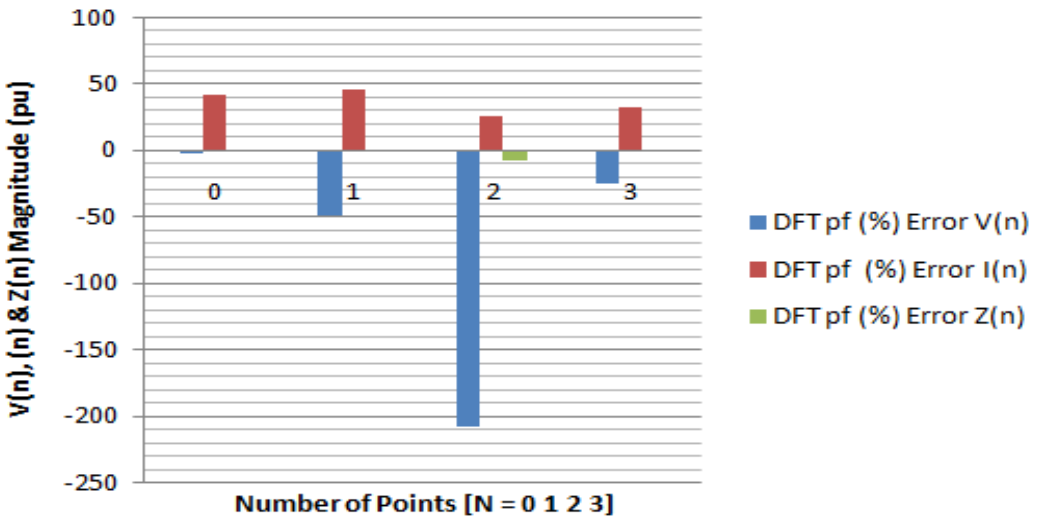

Figure 4.40: Percentage Error Btw DFT Pre-fault Voltages, Currents and Impedance forMathematical and Simulation Approaches

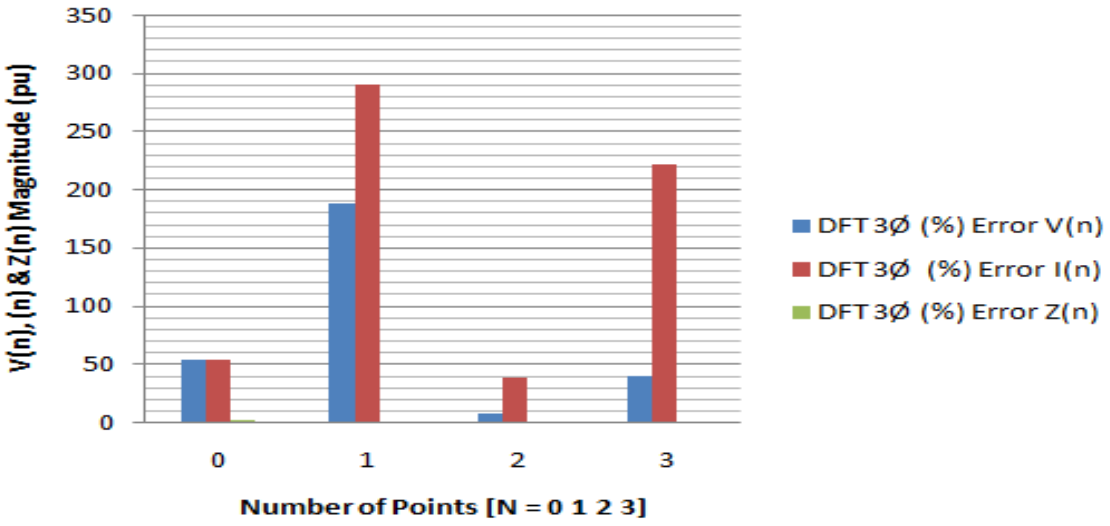

Figure 4.41: Percentage Error Btw DFT Three Phase Fault Voltages, Currents and Impedance forMathematical and Simulation Approaches

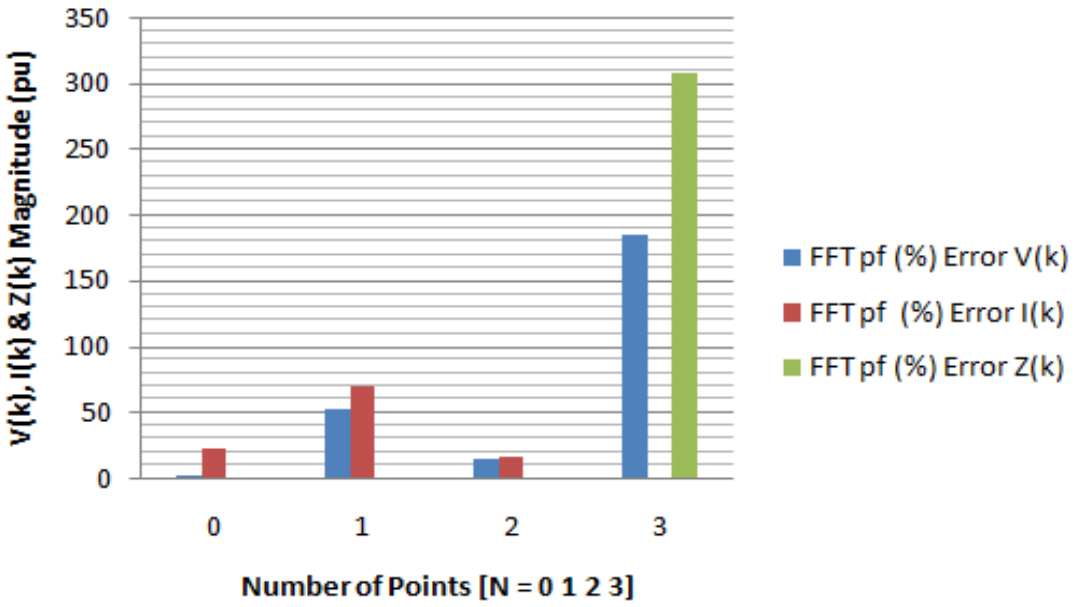

Figure 4.42: Percentage Error Btw FFT Pre-fault Voltages, Currents and Impedance forMathematical and Simulation Approaches

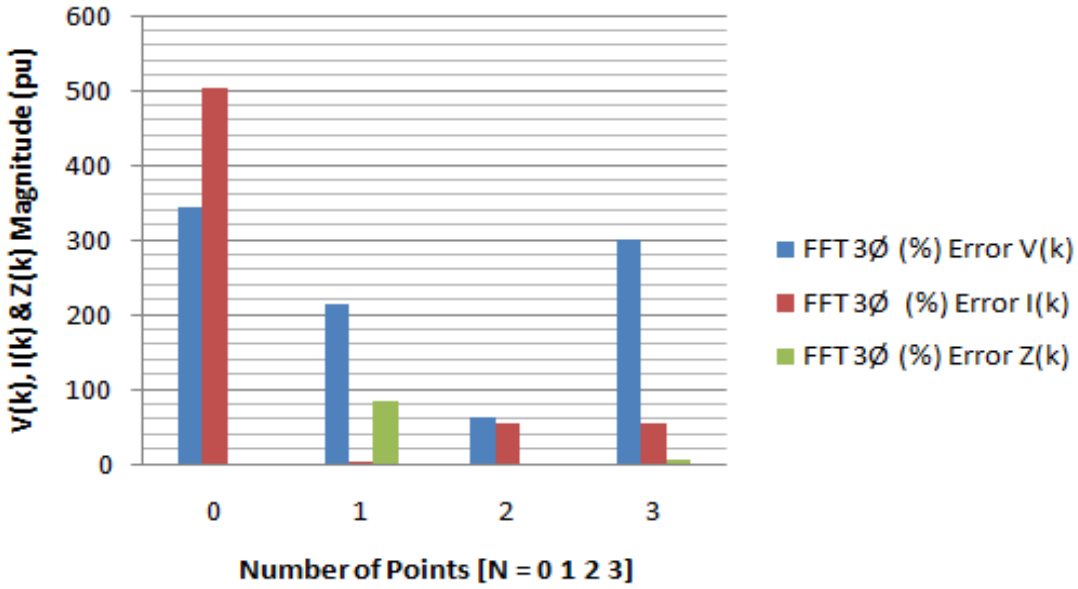

Figure 4.43: Percentage Error Btw FFT Three Phase Fault Voltages, Currents and Impedance forMathematical and Simulation Approaches

Here, the error means difference between the two components. Large difference means large error and large difference is as a result of increase in signal magnitude due to presents of fault. The magnitude of DFT voltage is larger than that of FFT. Thus, DFT result here is more pronounced.

The magnitude of DFT impedance is larger than that of FFT. Thus, it appear that DFT error results proves more closer to the fact that increase in the fault signal magnitude is due to the presents of fault on the transmission line [Robi P., (2013) &Capolino et al, (2003)].

4.4 Test Results of the Implementation of DFT and FFT Detection Model on IEEE 14 – Bus System

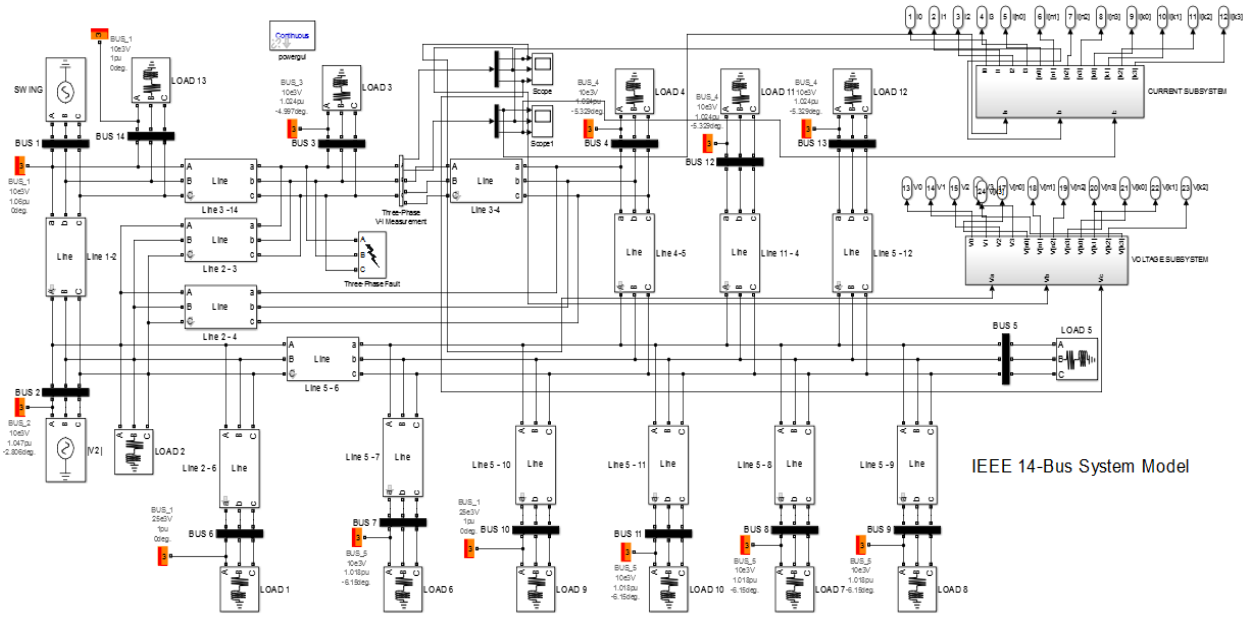

Figure 4.44: IEEE 14 – Bus System Transmission Line Network

The following tables

Table 4.18: DFT Application for Pre-fault Condition at N – POINTS

(k = 0,…, N – 1) on IEEE 14 – Bus System

S/N	**N - POINTS**	$\mathbf{V_{0,1,2,3}}$ for N = 4	$\mathbf{I_{0,1,2,3}}$ for N = 4	V(n)	I(n)	Z(n)	FAULTS
1	0	-2.2040	-0.7934	0.1067	0.8193	0.1302	LLL
2	1	-2.2040	-0.7934	-3.1580	-0.3559	8.8733	LLL
3	2	-2.2040	-0.7934	-4.3590	-0.7883	5.5296	LLL
4	3	-2.2040	-0.7934	-7.5000	-0.1919	39.0829	LLL

Table 4.19: FFT Application for Pre-fault Condition at N – POINTS

(k = 0,…, N – 1) on IEEE 14 – Bus System

S/N	**N - POINTS**	$\mathbf{V_{0,1,2,3}}$ for N = 4	$\mathbf{I_{0,1,2,3}}$ for N = 4	V(k)	I(k)	Z(k)	FAULTS
1	0	-2.2040	-0.7934	0.1067	0.8193	0.1302	LLL
2	1	-2.2040	-0.7934	-3.1580	-0.3559	8.8733	LLL
3	2	-2.2040	-0.7934	-0.0449	0.7971	-0.0563	LLL
4	3	-2.2040	-0.7934	-5.9290	-1.3540	4.3789	LLL

Table 4.20: DFT Application for Three Phase Fault Condition at N – POINTS

(k = 0,…, N – 1) on IEEE 14 – Bus System

S/N	**N - POINTS**	**DFTV$_{0,1,2,3}$** for N = 4	**DFTI$_{0,1,2,3}$** for N = 4	DFT V(n)	DFT I(n)	DFT Z(n)	FAULTS
1	0	-2.2040	-0.7934	1.5330	9.6370	0.1591	LLL
2	1	-2.2040	-0.7934	-2.8250	-4.1860	0.6749	LLL
3	2	-2.2040	-0.7934	-4.4280	-9.2730	0.4775	LLL
4	3	-2.2040	-0.7934	-8.6210	-22.5700	0.3819	LLL

Table 4.21: FFT Application for Three Phase Fault Condition at N – POINTS

(k = 0,…, N – 1) on IEEE 14 – Bus System

S/N	**N - POINTS**	**DFTV$_{0,1,2,3}$** for N = 4	**DFTI$_{0,1,2,3}$** for N = 4	FFT V(k)	FFT I(k)	FFT Z(k)	FAULTS
1	0	-2.2040	-0.7934	1.5330	9.6370	0.1591	LLL
2	1	-2.2040	-0.7934	-2.8250	-4.1860	0.6749	LLL
3	2	-2.2040	-0.7934	1.4500	9.3750	0.1547	LLL
4	3	-2.2040	-0.7934	-6.5240	-15.9200	0.4098	LLL

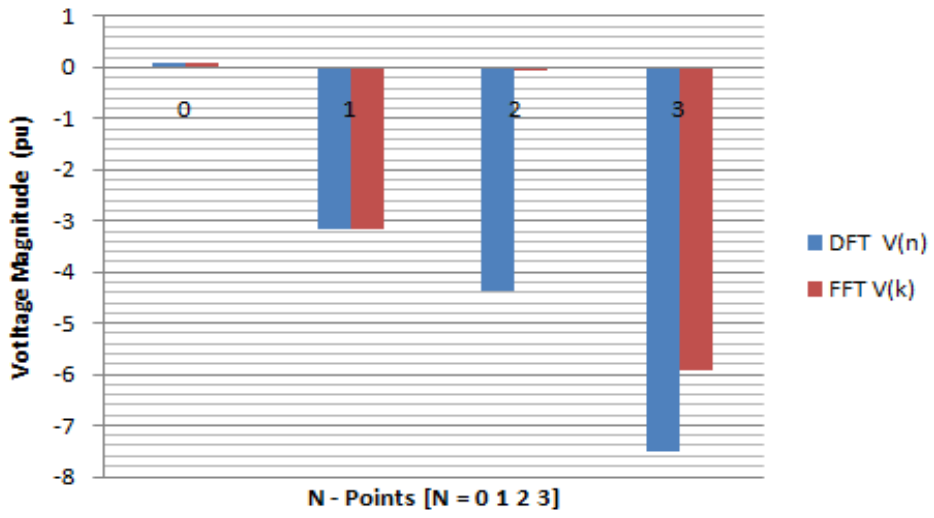

Figure 4.45: IEEE 14 – Bus DFT & FFT Pre-fault Voltages

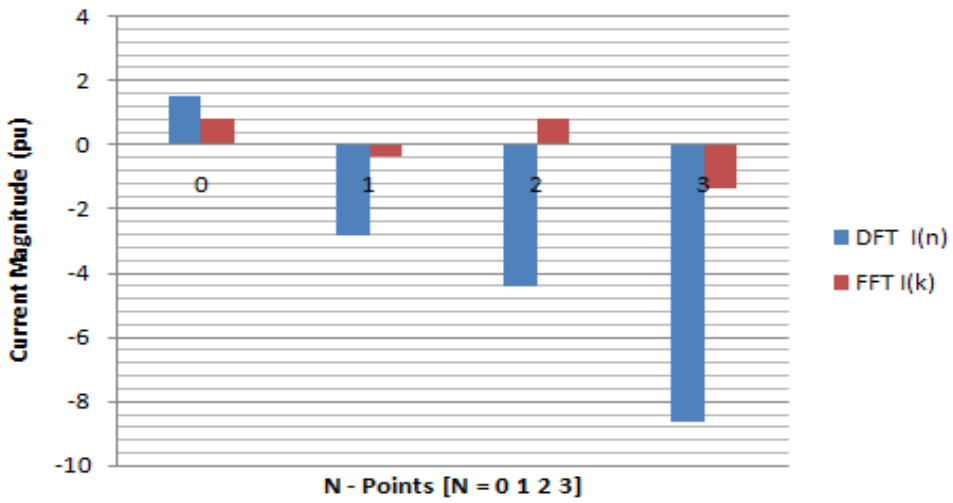

Figure 4.46: IEEE 14 – Bus DFT & FFT Pre-fault Currents

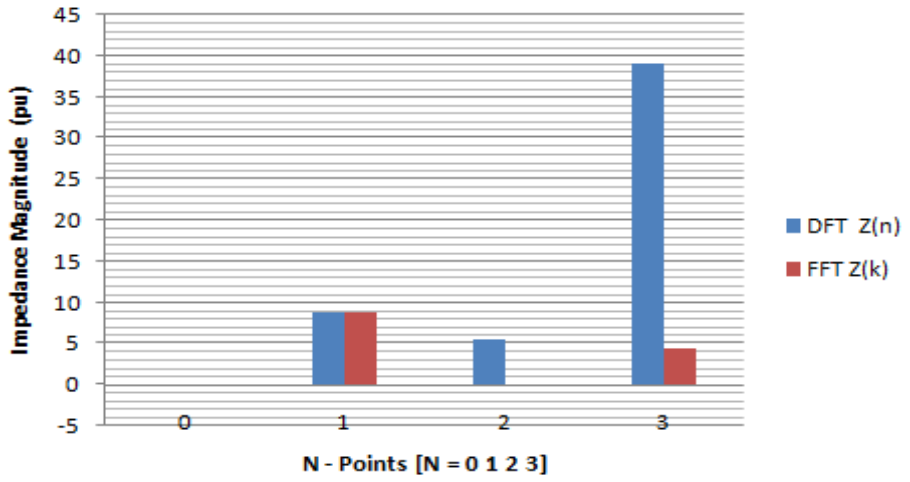

Figure 4.47: IEEE 14 – Bus DFT & FFT Pre-fault Impedances

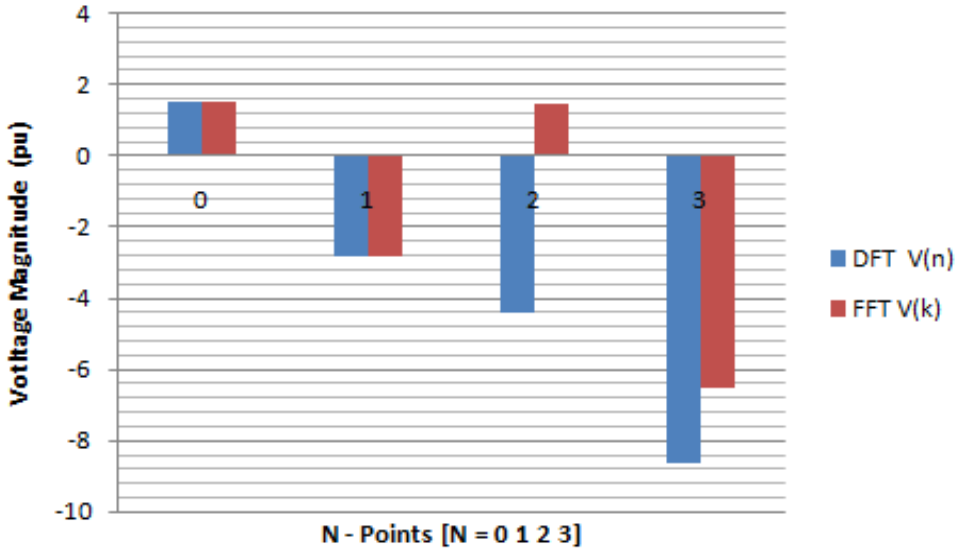

Figure 4.48: IEEE 14 – Bus DFT & FFT three phase fault Voltages

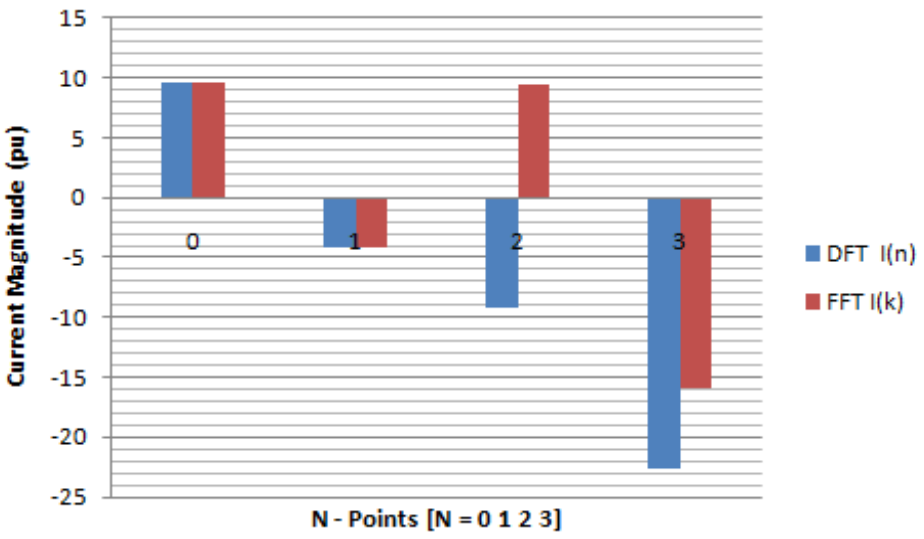

Figure 4.49: IEEE 14 – Bus DFT & FFT three phase fault Currents

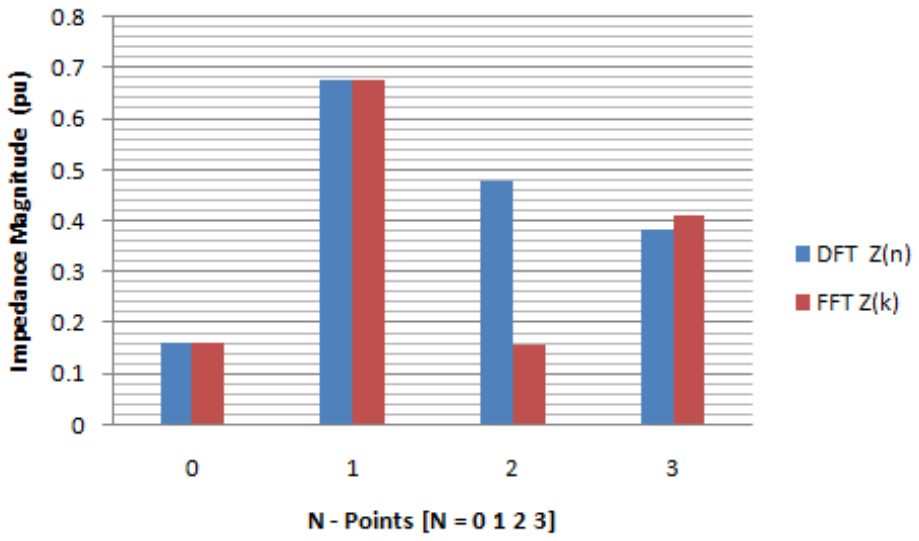

Figure 4.50: IEEE 14 – Bus DFT & FFT three phase fault Impedances

In the IEEE 14 – Bus network, DFT and FFT pre-fault voltages gave almost the same results which are not the case in Nigerian network results. Also, DFT and FFT gave almost same results except at N – Points 2 and 3 that has different results.

Figure 4.45 shows that the DFT pre-fault voltage magnitude is larger than FFT pre-fault magnitude for points 2 and 3 only, but equal with FFT pre-fault for points 0 and 1.

It is also the same in figure 4.46, where the DFT pre-fault current for all the points are larger than that of FFT by 1.0pu, 2.6pu, 3.4pu and 7.8pu for points 0, 1, 2 and 3 respectively.

However in figure 4.47, DFT and FFT pre-fault impedance, FFT pre-fault impedance for point 3 is larger than that of DFT by 6.5pu.

In figure 4.48, it shows that DFT and FFT three phase fault voltage magnitude are equal for points 0 and 1, but for points 2 and 3, the DFT three phase fault voltage is larger than that of FFT by 2.4pu and 2.0pu respectively.

Figure 4.49 show that for points 0, 2 and 3, the DFT and FFT three phase current magnitudes are equal, while for point 1, the DFT is larger than FFT BY 7.0pu.

Also, the impedance of DFT and FFT for points 0 and 1 are equal, while for 2 and 3 the DFT three phase impedance is larger than that of FFT by 28pu.

4.5 Comparative Analysis between the Results of the Application of the Mathematical Detection Model & IEEE 14 – Bus System Network

In this section, comparison between the results of DFT and FFT mathematical developed models implemented on Onitsha 330kV transmission line and IEEE 14 – Bus networks is performed by finding pre-fault and three phase fault voltage and current difference. The results are tabulated in tables 4.22, 4.23, 4.24 and 4.25.

$$\% \text{ Error} = \frac{\text{Nigerian (Mathematical) value} - \text{IEEE 14} - \text{Bus value}}{\text{IEEE 14} - \text{Bus value}} \; x \; 100 \qquad (4.2)$$

Table 4.22: DFT Pre-fault VoltagePercentage Error Difference between Nigerian (Mathematical Approach) and IEEE 14 – Bus Networks

S/N	**N - POINTS**	DFT V(n) Nigerian	DFT V(n) IEEE	Error Difference
1	0	-0.6191	0.1067	480.00
2	1	1.6733	-3.1580	47.00
3	2	0.3385	-4.3590	92.00
4	3	1.7460	-7.5000	77.00

Table 4.23: DFT Three Phase Fault Voltage Percentage Error Difference between Nigerian (Mathematical Approach) and IEEE 14 – Bus Networks

S/N	**N - POINTS**	DFT V(n) Nigerian	DFT V(n) IEEE	Error Difference
1	0	-0.1087	1.5330	92.90
2	1	0.2941	-2.8250	89.60
3	2	0.0691	-4.4280	98.40
4	3	0.3061	-8.6210	96.40

Table 4.24: FFT Pre-fault Voltage Percentage Error Difference between Nigerian (Mathematical Approach) and IEEE 14 – Bus Networks

S/N	**N - POINTS**	FFT V(n) Nigerian	FFT V(n) IEEE	Error Difference
1	0	-0.6191	0.1067	480.00
2	1	1.6183	-3.1580	48.80
3	2	0.8099	-0.0449	1703.80
4	3	1.6183	-5.9290	72.70

Table 4.25: FFT Three Phase Fault Voltage Percentage Error Difference between Nigerian (Mathematical Approach) and IEEE 14 – Bus Networks

S/N	**N - POINTS**	FFT V(n) Nigerian	FFT V(n) IEEE	Error Difference
1	0	4.0848	1.5330	166.46
2	1	-0.0463	-2.8250	98.36
3	2	-0.0446	1.4500	96.90
4	3	-0.0463	-6.5240	99.30

Table 4. 26: DFT Pre-fault Current Percentage Error Difference between Nigerian (Mathematical Approach) and IEEE 14 – Bus Networks

S/N	**N - POINTS**	DFT I(n) Nigerian	DFT I(n) IEEE	Error Difference
1	0	-0.1670	0.8193	79.62
2	1	0.6433	-0.3559	80.77
3	2	0.0812	-0.7883	89.69
4	3	0.6461	-0.1919	236.69

Table 4. 27: DFT Three Phase Fault Current Percentage Error Difference between Nigerian (Mathematical Approach) and IEEE 14 – Bus Networks

S/N	**N - POINTS**	DFT V(n) Nigerian	DFT V(n) IEEE	Error Difference
1	0	-1.1077	9.6370	88.50
2	1	2.9899	-4.1860	28.57
3	2	0.7053	4.2730	83.50
4	3	3.1239	-22.57	86.16

Table 4. 28: FFT Pre-fault Current Percentage Error Difference between Nigerian (Mathematical Approach) and IEEE 14 – Bus Networks

S/N	**N - POINTS**	FFT I(n) Nigerian	FFT I(n) IEEE	Error Difference
1	0	-0.3503	0.8193	57.24
2	1	0.1521	-0.3559	52.64
3	2	0.3371	-0.7971	57.71
4	3	0.5902	-1.3540	56.41

Table 4. 29: FFT Three Phase Fault Current Percentage Error Difference between Nigerian (Mathematical Approach) and IEEE 14 – Bus Networks

S/N	**N - POINTS**	FFT I(n) Nigerian	FFT I(n) IEEE	Error Difference
1	0	5.5992	9.6370	41.89
2	1	-0.1116	-4.1860	97.00
3	2	-0.0290	9.3750	99.70
4	3	-0.1116	-15.92	99.30

According to the power system transmission line fault characteristics, when fault such as three phase fault occurs on the transmission line, the line voltage decreases

very lower than the pre-fault value. At same time, the current on the line drastically increased more than its pre-fault value.

Therefore, considering Tables 4.22 to 4.29, DFT Nigerian (mathematical approach) has its lowest pre-fault voltage magnitude as 0.3395pu while IEEE 14-bus has 0.1067pu. In the same vein, FFT Nigerian (mathematical approach) has its lowest pre-fault voltage magnitude as 0.6191pu while IEEE 14-bus has 0.0449pu.

Also, the DFT Nigerian (mathematical approach) has its largest three phase fault current magnitude as 0.6461pu while IEEE 14-bus has 0.8193pu. In the same vein, FFT Nigerian (mathematical approach) has its largest three phase fault current magnitude as 5.5992pu while IEEE 14-bus has 15.92pu.

However, the difference between the DFT mathematical approach lowest pre-fault voltage and its largest current magnitudes is only 0.3066pu and that of FFT is 4.9801pu which is large enough.

Also, the difference between the DFT IEEE 14 – bus network lowest pre-fault voltage and its largest current magnitudes is 0.7126pu and that of FFT is 15.88pu which is significantly large enough.

With these results above, three phase fault has been successfully detected and cleared using DFT and FFT signal processing techniques since the results conforms to the power system transmission line fault characteristics.

CHAPTER FIVE

CONCLUSION

5.1 Summary of the work

This dissertation has been able to develop and implement a fault detection modelbased on Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) for fault detection on the any transmission line. These fault detection models are developed, implemented using mathematical and computer simulation approaches. They were validated using Matlab/Simulink R2017a and IEEE 14 – bus System networks.

Considering the performance of the two models, the output results obtained and the difficulties encountered, we conclude that, the DFT and FFT produced time – and frequency – domain waveforms and spectrum diagramsrespectively. These diagrams illustrates the faulty conditions of the transmission line and the low and large magnitude of DFT and FFT parameters (voltage and current) respectively against that of no faults are evidence that fault occurred and were detected on the transmission line.

One of the disadvantages of using DFT in diagnosing the fault is it inability to carry detailed fault information about the signals but give a better result than the FFT.

5.2 Contributions to Knowledge

The following discoveries were made in this research as the contributed knowledge in the field of engineering.

I. This research is providing for the first time an analytical and simplified results of an application of a new fault detection system developed using real life parameters to 330kV Nigerian Power Grid System.

II. The dissertation introduces the new simplified signal processing technique which bridges the gap created by the complex and abstract nature of the fundamental signal processing techniques.

5.3 Suggestion for future works

There are still more work to be done in the areas of application of these methods. As a matter of fact, further studies on this work will be very useful especially in the areas ofdiagnosis (detection, classification and location) of fault in power system generation, transmission and distribution and other engineering discipline.

Also, the method can be combined with the artificial intelligence (artificial neural network, genetic algorithm, swam optimization, support vector machines, radial basic functions etc.) to solve some problems in power system protections, power system planning and optimization, stability analysis and other engineering areas.

References

Anshika R., (2019). ‘’ Components of Power Systemectrical Engineering (withDiagrams) Slip Ring Manufactures.

Abdelsalem, Elhaffar A. M., (2008), ‘Power Transmission Line Fault Location Based on Current Traveling Waves’, Espoo 2008, Helsinki University of Technology, Faculty of Electronics, Communications and Automation, ISBN 978-951-22-924-5

Abdelsalem M. A., 2008. Multi – End Travelling Wave Fault Location Based on Current Traveling Wave’. 16 PSCC, Glasgow, Scotland, July 14 – 18, 2008.

Ashrafian A., Rostani M., Gharehpetian G. B., Gholamphasemi M., 2012. ‘Application of Discrete S – Transform for Differential Protection of Power Transformer’. International Journal of Computer and Electrical Engineering, Vol. 4, No 2, April 2012.

Bashier E. and Tayeb M., (2016), ‘Faults Detection in Power Systems Using Artificial Neural.

Bowan Y., Fangyu L., Jin Y., and Wenzhan S., ‘Conditions Monitoring and Fault Diagnosis of Generators in Power Networks’, University of Georgia Atlanta, Georgia USA. 2019. NSF – 1725636, NSF – CNS – 1066391.

Capolino G. A., Henao H. Assaf T. ‘Discrete Fourier Transform for Computation of Symmetrical Components Harmonics’. 2003 IEEE Bologna, Italy.

Chengzong P., Mladenk, Zhang N., 2010. ‘Wavelet Based Method for Transmission Line Fault Detection and Classification During Power Swing’, Texas A & M University College Station, TX77843, USA, WERC 323, MS 3128.

Faruqul A. M. D., Salman S., Hamid A., 2014. 'Short Circuit Current Calculation and Prevention in High Voltage Power Networks'. Blekinge TekniskaHogskola 371799, Karlskrona Sweden.

Francisco M. G. L., 'Power System Protection'. DOI:10,13140/RG.2.2.31364.27523. https://www.researchgate.net/publication/339460150.

Francisco M. G. L., 'Introduction to Power Systems', University of South – Eastern Norway. DOI:10.13140/RG:2.25941.99043

Gupta J. B., (2004), 'A course in Electrical Power System Engineering', S. K. Kataria& Sons New Delhi, India.

Henao H., Assaf T. and Capolino G. A., 2003. 'The discrete Fourier transform for computation of symmetrical components harmonics', Fellow, IEEE.

Jensen C. F., 2014, 'Faults on Transmission line Cables and Current Fault Location Methods', 001: 10, 1007/978 – 3 - 319 – 05398 – 1_2, SpringerInternational Publishing Switzerland, 2014. – Hill Publishing Company Limited New Delhi, India, ISBN 0-07-462299-4.

Klingerman N., Energy D., Finney D., Samineni S., Fischer N., and Haas D., (2011), 'Understanding Generator Stator Ground, Faults and Their Protection Schemes', Schweitzer Engineering Laboratories, Inc.

Karl Z., David C., 'Impedance Based Fault Location experience'. Journal of Reliable Power.Vol. 1.Number 1. July, 2010.

Kumararaja A., Nazeer S., Srinivas V. and Bhanu C. Y., 'Fault Detection, Classification and Location on Transmission Lines Using Fundamental Phasor Base Approach'.

International Journal of Recent Technology and Engineering (IJRTE). ISSN:2277 – 3878. Vol.8, Issue 3, June 2019.

Maher R. C., (2003), 'Introduction to Fast Fourier Transform (FFT) Algorithms'. DSP Laboratory Spring, 2003.

Mamis M. S., Arkan M., (2011), 'FFT Based Fault Location Algorithm for Transmission Lines'

Mohammad A. B., 2013. 'Travelling Waves for Finding the Fault Location on Transmission Line'. Journal of Electrical and Electronics Engineering, 2013, (1): 1 – 19.

Müslüm A. 2010. 'FFT Based Fault Location Algorithm for Transmission Lines'Inonu University, Engineering Faculty, Electrical & Electronics Eng. Dept., Malatya, Turkey, mehmet.mamis@inonu.edu.tr, muslum.arkan@inonu.edu.tr

Murthy P. S. R., 'Power System Analysis', B. S. Publication, 4 – 4 – 309, Giriraj Lane, Sultan Bazar, Hyderabad – 500095AP. www.bspublication.net.

Personal E., García A., Parejo A., Larios D. F., Biscarri F. and León C., (2015), 'A Comparison of Impedance-Based Fault Location Methods for Power Underground Distribution Systems', Department of Electronic Technology, University of Seville, 41011 Seville, Spain. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, (An ISO 3297: 2007 Certified Organization), Vol. 2, Issue 12, December 2013.

Prasad C. D. and Paresh K. N., 'A DFT – ED Based Approach for Detection and Classification of Faults in Electric Power Transmission Network'. Ain Shams Engineering Journal. Vol 10, Issue 1, March 2019, pp. 171 – 178.

Rao M. and Hasabe R. P. (2013), 'Detection and classification of faults on transmission line using wavelet transform and neural network', Walchand College of Engineering, Sangi. ISSN: 2278 – 8948, Volume – 2, Issue – 5, 2013.

Reddy B. R., 2009, 'Detection and Location of Faults on Transmission Line Using Coiflet Mother Wavelet Transform', Journal of Theoretical and Applied Information Technology. 2005 – 2009. www.jatit.org .

Robi P., 2013. 'Tutorial on Signal Analysis Method of Fault Study'

Roshni U. &Niranjan V., Prakash C. &Rao R. S., (2012), 'Location of Faults In Transmission Line Using Fast Fourier Transform and Discrete Wavelet Transform In Power Systems', Undergraduate Academic Research Journal (UARJ), ISSN: 2278 – 1129, Volume-1, Issue-1, 2012.

Sai R., Marutheswar G.V., 'Classification of Power Quality Disturbance Using Wavelet Transform and S – Transform Based Artificial Neural Network.' International Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering ISO 3297: 2007 Certified Organization. Vol. 2, Issue 12, December, 2013.

Saadat H., (2006), 'Power System Analysis', Tata McGraw – Hill PublishingCompany Limited New Delhi, India.

Sadiku M. N. O., Alexander C. K., 'Fundamentals of Electric Circuits', McGraw – Hill Company. Inc., New York City, NY, 10020.

Saurabh J. and Abdul G. S., 'Transmission Line Fault Detection and Classification using Alienation Coefficient Technique for Current Signal'. DOI: 10.1109/12CT.2018.8529447.

Shih M. H., 'Power System Basic Concepts and Application', PDH Centre 5272 Meadow Estates Drive, Fairfax, VA22030 – 6658. www.PDHCentres.com.

Tharaja B. L. and Tharaja A. K., 2001, Electrical Technology', Tata McGraw – HillPublishing Company Limited New Delhi, India.

Turan G.' Modern Power System Analysis'.John Wiley and Sons Inc. publishing. TK 1001. G66. 1987. ISBN 0 - 471 – 85903 – www.dieselserviceandsupply.com

Wadhwa C. L., (2010). ''Electical Power Systems'' Tata McGraw – Hill Publishing Company Limited New Delhi, India.

Williams P. D., 2012.'Analysis of Faults in Overhead Transmission Lines'. Department of Electrical and Electronics Engineering, California State University, Sacramento, 2012.

Zimmerman K. and Costello D., (2010), 'Impedance-Based Fault Location Experience',Schweitzer Engineering Laboratories, Inc. SEL Journal of Reliable Power, Volume 1, Number 1, July 2010.

Matlab/Simulink R2017a Codes for DFT and FFT fault detection on the 330kV 96Km Onitsha - Enugu transmission line

Appendix I

```
% MATLAB CODES FOR FAULT DETECTION ON A 33OKV POWER
SYSTEM TRANSMISSION LINE
% USING DISCRETE FOURIER TRANSFORM (DFT) AND FAST FOURIER
TRANSFORM (FFT)
% Parameters for the fault detection are the three phase voltage and
% current (Va, Vb, Vc, and Ia, Ib, Ic)respectively.
% Vp is called the discrete three phase voltage while Ip is called the
% discrete three phase current.
f = 50;                    % Line frequency
t = 1;
theta = [45 50 55 60];          % Phase difference
N = [0 1 2 3];
wt = 2*pi*f*t;
Va = 0.4;                % Normal Steady State
Vb = 0.45;                % Normal Steady State
Vc = 0.49;                % Normal Steady State
Ia = 0.22;                % Normal Steady State
Ib = 0.14;                % Normal Steady State
Ic = 0.149;                % Normal Steady State
Ns = 4;                    % Number of points

V_average = (Va+Vb+Vc)/3;
I_average = (Ia+Ib+Ic)/3;
Zrs = V_average/I_average;
% To discretize three phase parameters;
Vp = ((2/3)*(Va+(Vb*(exp((1j)*2*pi/Ns)))+(Vc*(exp((-1j)*2*pi/Ns)))));
Ip = ((2/3)*(Ia+(Ib*(exp((1j)*2*pi/Ns)))+(Ic*(exp((-1j)*2*pi/Ns)))));
n = Ns - 1;
vt = ((Vp)*(sin(wt)+theta));
it = ((Ip)*(sin(wt)+theta));
% To compute the DFT of Pre-fault vt and it;
% Vn is the DFT of vt;
% In is the DFT of it;
% Vn is given as sum of the product of vt and (exp((-j)*2*pi)/N);
% In is given as sum of the product of it and (exp((-j)*2*pi)/N);
```

```
% Computation of DFT of vt at No fault and on load conditions;

Vn = vt.*exp(-1j*2*pi*n/Ns);
In = it.*exp(-1j*2*pi*n/Ns);
Vn_mag = abs(Vn);
In_mag = abs(In);

Zn = Vn./In;

Zn_mag = abs(Zn);
fault_result = Zn_mag < Zrs;
figure(1);
bar(Zn_mag);
title('DFT Three Phase Prefault Impedance')
figure(2);
bar(Vn_mag);
title('DFT Three Phase Prefault voltage')
figure(3)
bar(In_mag);
title('DFT Three Phase Prefault current')



Wk = [1 -1j -1 1j];
Vk = [0 0 0 0];
Vk(1) = (vt(1) + vt(3)*Wk(1)) + Wk(1)*(vt(2) + vt(4)*Wk(1));
Vk(2) = (vt(1) + vt(3)*Wk(3)) + Wk(2)*(vt(2) + vt(4)*Wk(3));
Vk(3) = (vt(1) + vt(3)*Wk(1)) + Wk(3)*(vt(2) + vt(4)*Wk(1));
Vk(4) = (vt(1) + vt(3)*Wk(3)) + Wk(4)*(vt(2) + vt(4)*Wk(3));

Ik = [0 0 0 0];
Ik(1) = (it(1) + it(3)*Wk(1)) + Wk(1)*(it(2) + it(4)*Wk(1));
Ik(2) = (it(1) + it(3)*Wk(3)) + Wk(2)*(it(2) + it(4)*Wk(3));
Ik(3) = (it(1) + it(3)*Wk(1)) + Wk(3)*(it(2) + it(4)*Wk(1));
Ik(4) = (it(1) + it(3)*Wk(3)) + Wk(4)*(it(2) + it(4)*Wk(3));

Zk = Vk./Ik;

Zk_mag = abs(Zk);
Vk_mag = abs(Vk);
```

```
Ik_mag = abs(Ik);

fault_result_fft = Zk_mag < Zrs;
figure(4);
bar(Zk_mag);
title('FFT Three Phase Prefault Impedance')
figure(5);
bar(Vk_mag);
title('FFT Three Phase Prefault voltage')
figure(6)
bar(Ik_mag);
title('FFT Three Phase Prefault current')
```

MATLAB CODES BARCHART RESULT FOR THREE PHASE PRE -FAULT COMPUTATION USING DFT

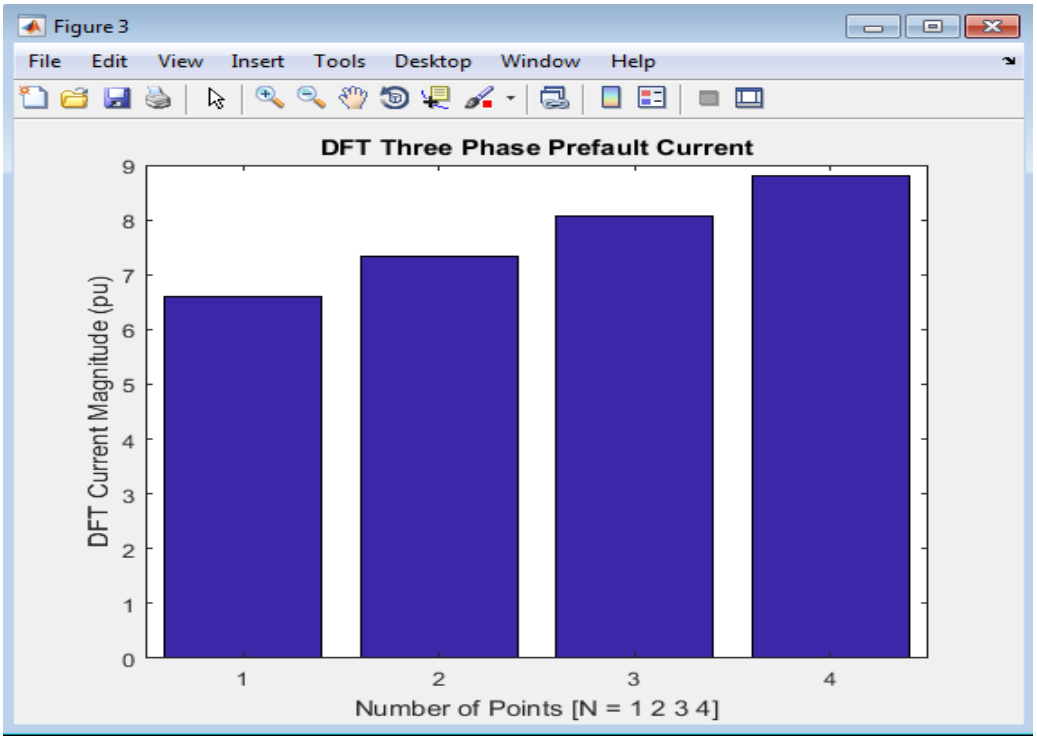

Figure 1: DFT three pre-fault current is 8.9pu maximum at N = 4 or 3 for (N = 0 1 2 3).

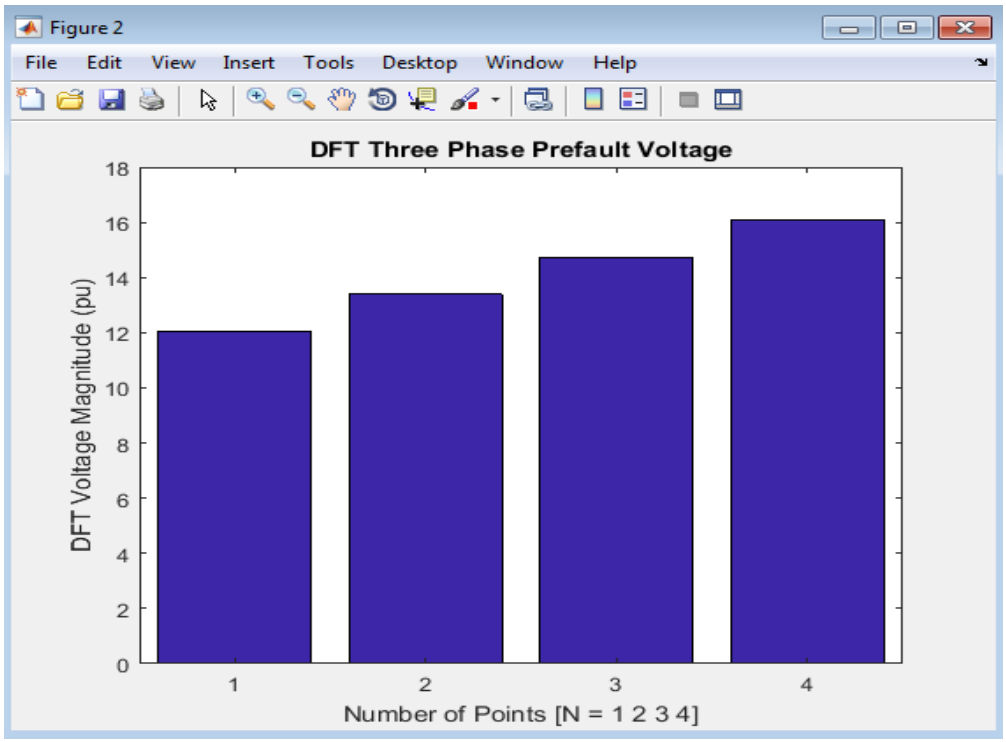

Figure 2: DFT three pre-fault voltage is 16pu maximum at N = 4 or 3 for (N = 0 1 2 3).

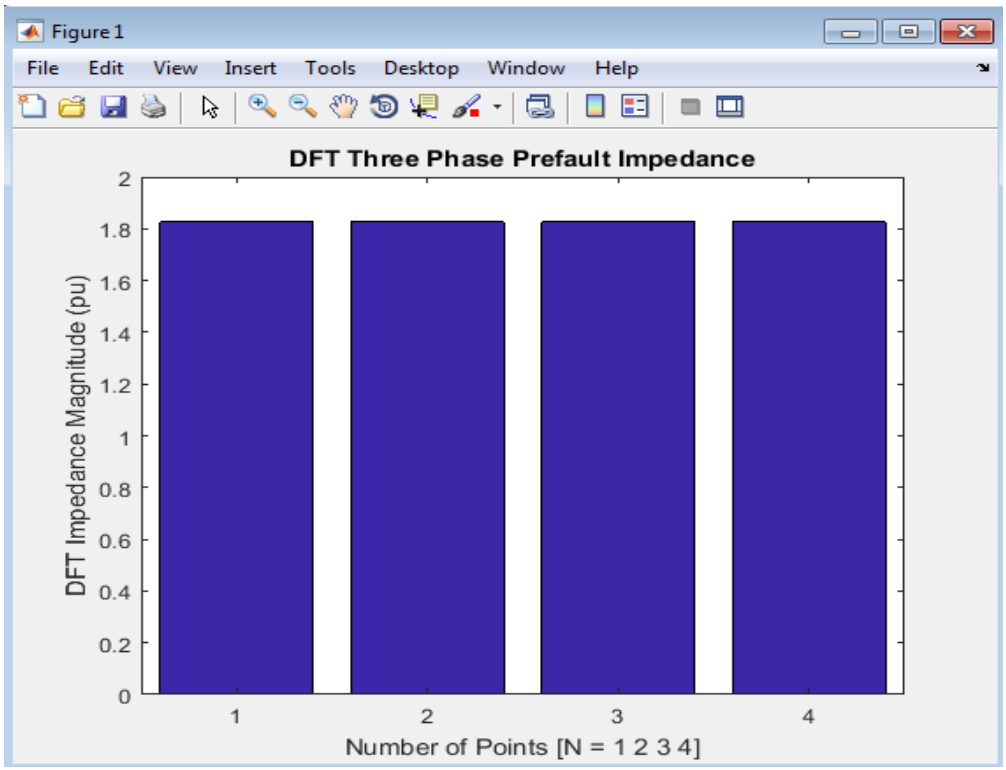

Figure 3: DFT three pre – fault impedance is equal at all points of N = 1.8pu, (N = 0 1 2 3).

Matlab Codes Barchart Result for Three Phase

Pre-Fault Computation Using DFT

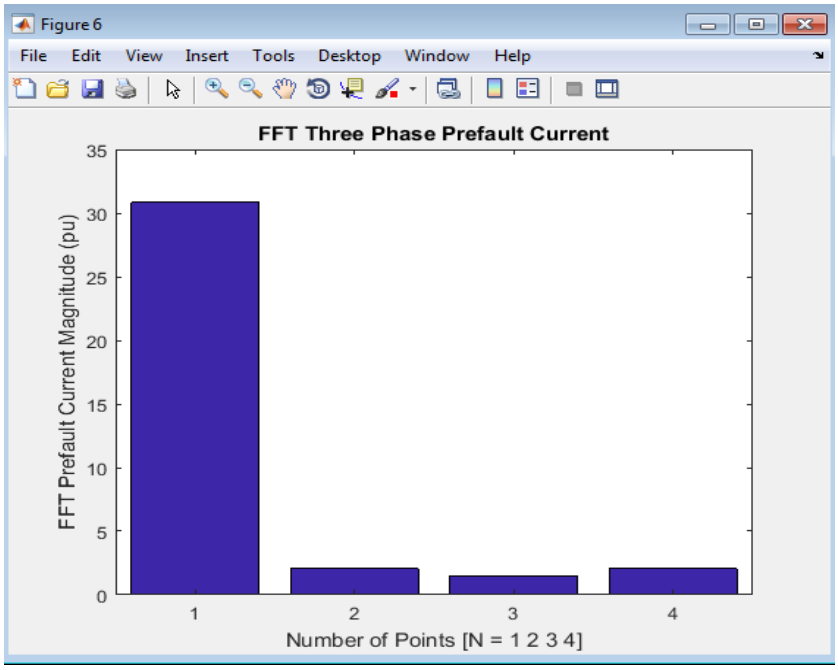

Figure 6: FFT three pre-fault current is 32pu maximum at N = 1 or 0 for (N = 0 1 2 3).

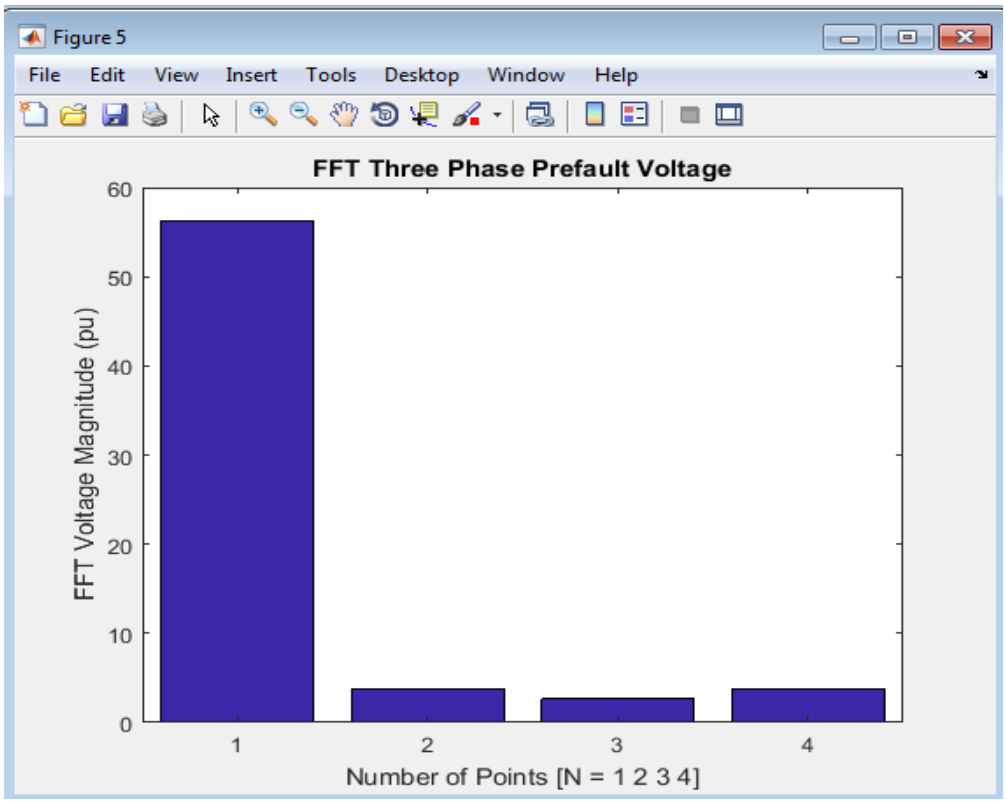

Figure 4: FFT three pre-fault Voltage is 57pu maximum at N = 1 or 0 for (N = 0 1 2 3).

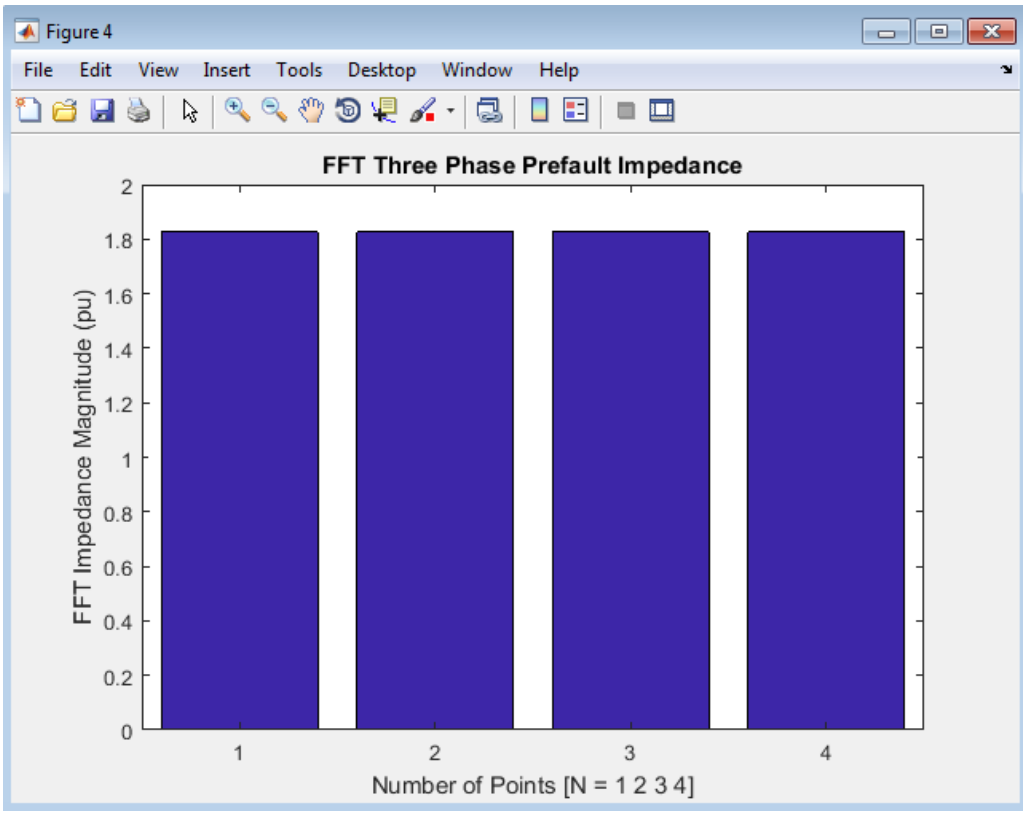

Figure 5: FFT three pre – fault impedance is equal at all points of N, (N = 0 1 2 3).

MATLAB CODES BARCHART RESULT FOR THREE PHASE FAULT COMPUTATION USING DFT

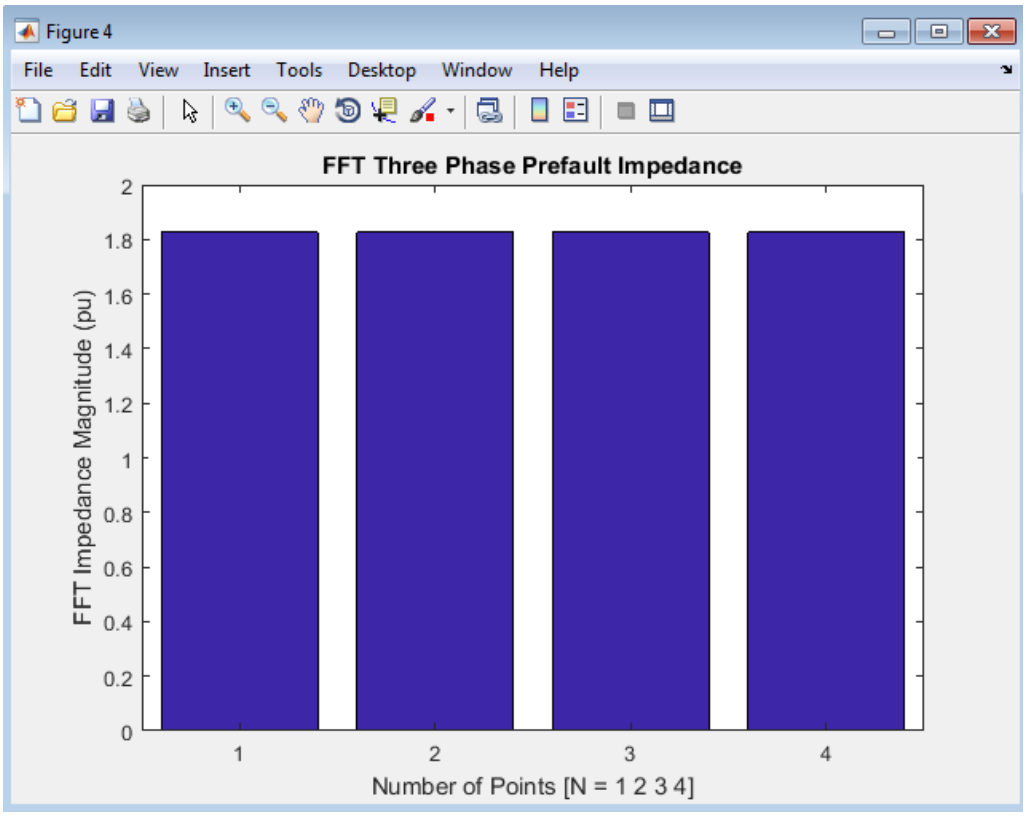

Figure 6: DFT three phase fault current is 76pu maximum at N = 4 or 3 for (N = 0 1 2 3).

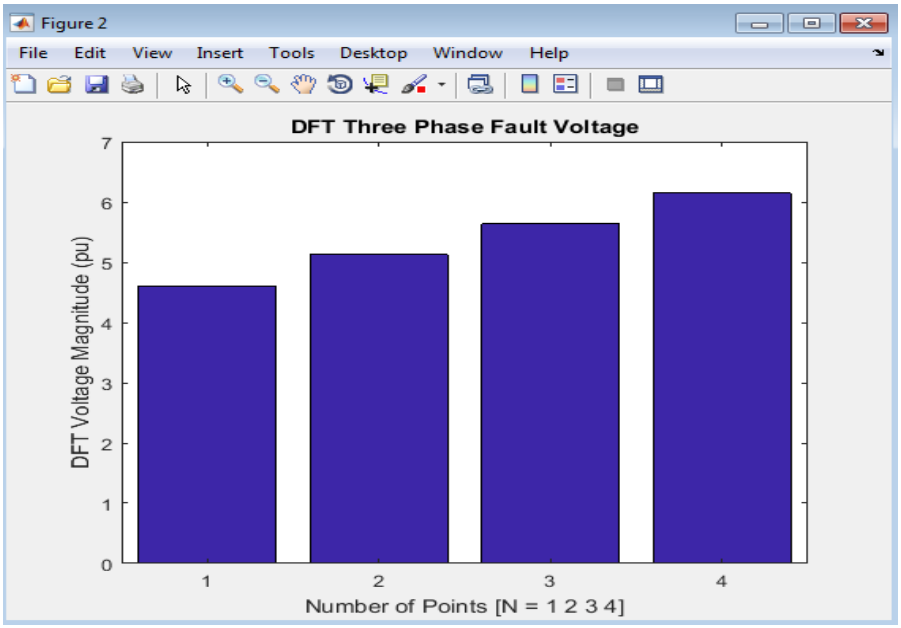

Figure 7: DFT three phase fault voltage is 6pu maximum at N = 4 or 3 for (N = 0 1 2 3).

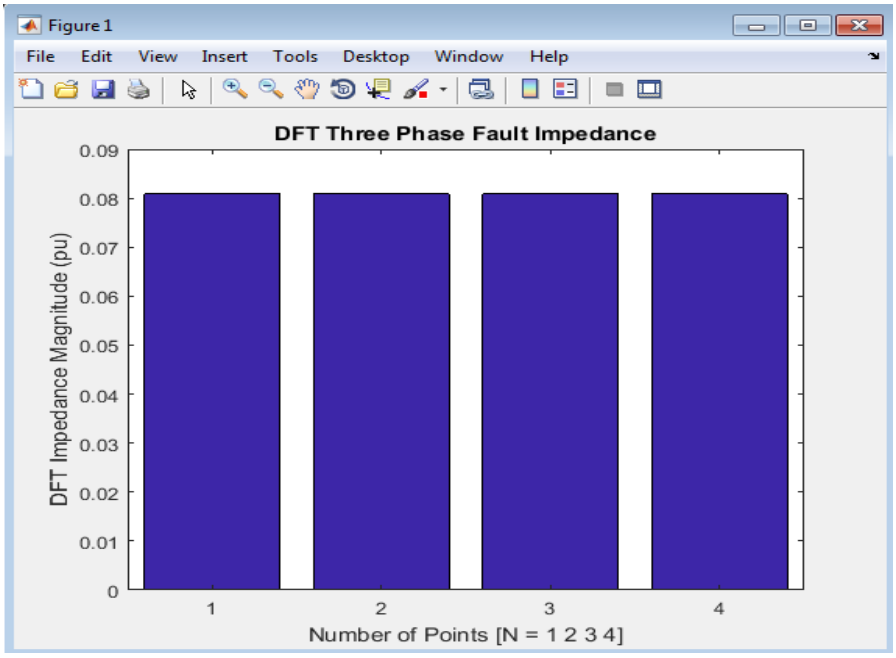

Figure 8: DFT three phase fault impedance is equal at all points of N = 0.08pu, (N = 0 1 2 3).

MATLAB CODES BARCHART RESULT FOR THREE PHASE FAULT COMPUTATION USING FFT

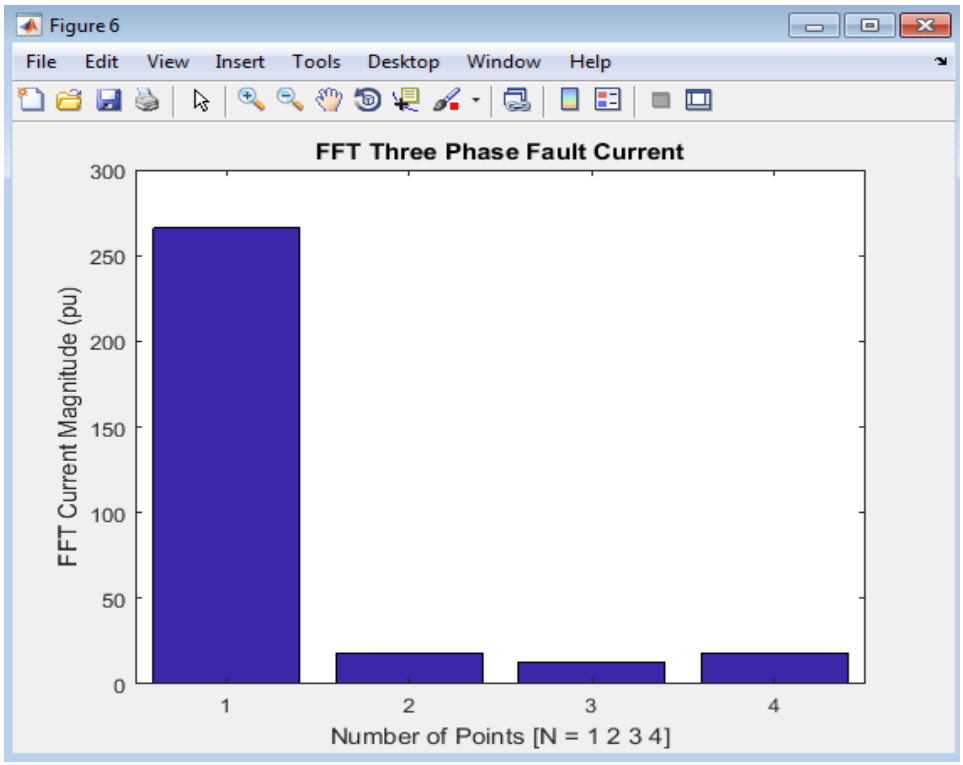

Figure 9: FFT three pre-fault Voltage is 270pu maximum at N = 1 or 0 for (N = 0 1 2 3).

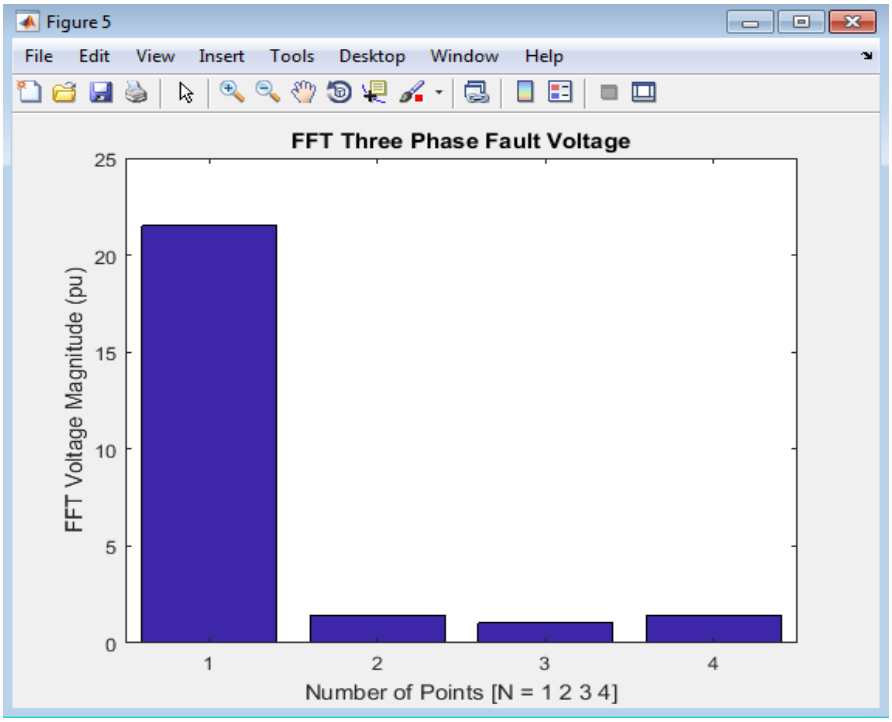

Figure 10: FFT three phase fault Voltage is 23pu maximum at N = 1 or 0 for (N = 0 1 2 3).

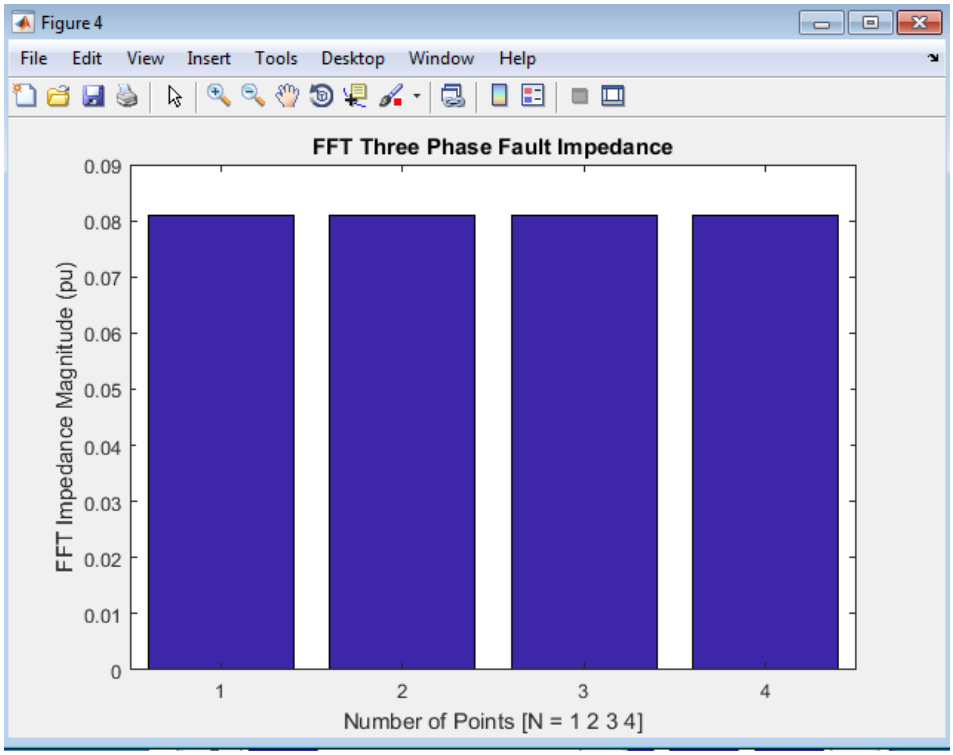

Figure 11: FFT three phase fault impedance is equal at all points of N = 0.08pu, (N = 0 1 2 3).

Appendix II

Matlab/Simulink Codes for the Developed DFT and FFT Model for Fault Detection on the Power System Transmission Line

File: ert_main.c

```
*
* Code generated for Simulink model 'NEWPDFDEVELOPMENTDESERTATIONamusingnow'.
*
* Model version : 1.21
* Simulink Coder version : 8.12 (R2017a) 15-Nov-2019
* C/C++ source code generated on : Fri 15 03:12:04 2019
*
* Target selection: ert.tlc
* Embedded hardware selection: Intel->x86-64 (Windows64)
* Code generation objectives:
* 1. Execution efficiency
* 2. RAM efficiency
* Validation result: Not run
*/

#include <stddef.h>
#include <stdio.h>/* This ert_main.c example uses printf/fflush */
#include "NEWPDFDEVELOPMENTDESERTATIONamusingnow.h"/* Model's header file */
#include "rtwtypes.h"

/*
* Associating rt_OneStep with a real-time clock or interrupt service routine
* is what makes the generated code "real-time". The function rt_OneStep is
* always associated with the base rate of the model. Subrates are managed
* by the base rate from inside the generated code. Enabling/disabling
* interrupts and floating point context switches are target specific. This
* example code indicates where these should take place relative to executing
* the generated code step function. Overrun behavior should be tailored to
* your application needs. This example simply sets an error status in the
* real-time model and returns from rt_OneStep.
```

```
33 */
34 void rt_OneStep(void);
35 void rt_OneStep(void)
36 {
37 static boolean_T OverrunFlag = false;
38
39 /* Disable interrupts here */
40
41 /* Check for overrun */
42 if (OverrunFlag) {
43 rtmSetErrorStatus(rtM, "Overrun");
44 return;
45 }
46
47 OverrunFlag = true;
48
49 /* Save FPU context here (if necessary) */
50 /* Re-enable timer or interrupt here */
51 /* Set model inputs here */
52
53 /* Step the model for base rate */
54 NEWPDFDEVELOPMENTDESERTATIONamusingnow_step();
55
56 /* Get model outputs here */
57
58 /* Indicate task complete */
59 OverrunFlag = false;
60
61 /* Disable interrupts here */
62 /* Restore FPU context here (if necessary) */
63 /* Enable interrupts here */
64 }
65
66 /*
67 * The example "main" function illustrates what is required by your
68 * application code to initialize, execute, and terminate the generated code.
69 * Attaching rt_OneStep to a real-time clock is target specific. This example
70 * illustrates how you do this relative to initializing the model.
```

```
71  */
72  int_T main(int_T argc, constchar *argv[])
73  {
74  /* Unused arguments */
75  (void)(argc);
76  (void)(argv);
77
78  /* Initialize model */
79  NEWPDFDEVELOPMENTDESERTATIONamusingnow_initialize();
80
81  /* Attach rt_OneStep to a timer or interrupt service routine with
82  * period 5.0E-5 seconds (the model's base sample time) here. The
83  * call syntax for rt_OneStep is
84  *
85  * rt_OneStep();
86  */
87  printf("Warning: The simulation will run forever. "
88  "Generated ERT main won't simulate model step behavior. "
89  "To change this behavior select the 'MAT-file logging' option.\n");
90  fflush((NULL));
91  while (rtmGetErrorStatus(rtM) == (NULL)) {
92  /* Perform other application tasks here */
93  }
94
95  /* Disable rt_OneStep() here */
96  return 0;
97  }
98
99  /*
100 * File trailer for generated code.
101 *
102 * [EOF]
103 */
104
```

WORKING MATLAB CODES

File: [NEWPDFDEVELOPMENTDESERTATIONamusingnow.c]

```
1  /*
2  * File: NEWPDFDEVELOPMENTDESERTATIONamusingnow.c
3  *
4  * Code generated for Simulink model
   'NEWPDFDEVELOPMENTDESERTATIONamusingnow'.
5  *
6  * Model version : 1.21
7  * Simulink Coder version : 8.12 (R2017a) 16-Feb-2017
8  * C/C++ source code generated on : Sat Nov 16 03:12:04 2019
9  *
10 * Target selection: ert.tlc
11 * Embedded hardware selection: Intel->x86-64 (Windows64)
12 * Code generation objectives:
13 * 1. Execution efficiency
14 * 2. RAM efficiency
15 * Validation result: Not run
16 */
17
18 #include "NEWPDFDEVELOPMENTDESERTATIONamusingnow.h"
19 #include <math.h>
20 #include <stdlib.h>
21 #define NumBitsPerChar 8U
22
23 /* Private macros used by the generated code to access rtModel */
24 #ifndef rtmIsMajorTimeStep
25 # define rtmIsMajorTimeStep(rtm) (((rtm)->Timing.simTimeStep) ==
   MAJOR_TIME_STEP)
26 #endif
27
28 #ifndef rtmIsMinorTimeStep
29 # define rtmIsMinorTimeStep(rtm) (((rtm)->Timing.simTimeStep) ==
   MINOR_TIME_STEP)
30 #endif
31
32 #ifndef rtmGetTPtr
33 # define rtmGetTPtr(rtm) ((rtm)->Timing.t)
34 #endif
35
36 #ifndef rtmSetTPtr
```

```
37 # define rtmSetTPtr(rtm, val) ((rtm)->Timing.t = (val))
38 #endif
39
40 #ifndef CodeFormat
41 #define CodeFormat S-Function
42 #else
43 #undef CodeFormat
44 #define CodeFormat S-Function
45 #endif
46
47 #ifndef S_FUNCTION_NAME
48 #define S_FUNCTION_NAME simulink_only_sfcn
49 #else
50 #undef S_FUNCTION_NAME
51 #define S_FUNCTION_NAME simulink_only_sfcn
52 #endif
53
54 #ifndef S_FUNCTION_LEVEL
55 #define S_FUNCTION_LEVEL 2
56 #else
57 #undef S_FUNCTION_LEVEL
58 #define S_FUNCTION_LEVEL 2
59 #endif
60
61 #ifndef RTW_GENERATED_S_FUNCTION
62 #define RTW_GENERATED_S_FUNCTION
63 #endif
64
65 #ifndef rtmGetDataMapInfo
66 # define rtmGetDataMapInfo(rtm) NULL
67 #endif
68
69 #ifndef rtmSetDataMapInfo
70 # define rtmSetDataMapInfo(rtm, val)
71 #endif
72
73 #if !defined(RTW_SFUNCTION_DEFINES)
74 #define RTW_SFUNCTION_DEFINES
```

```
75 #ifndef _RTW_COMMON_DEFINES_
76 #define _RTW_COMMON_DEFINES_
77 #endif
78 #endif
79
80 /* Block signals and states (auto storage) */
81 DW rtDW;
82
83 /* External outputs (root outports fed by signals with auto storage) */
84 ExtY rtY;
85
86 /* Real-time model */
87 RT_MODEL rtM_;
88 RT_MODEL *const rtM = &rtM_;
89 extern real_T rt_hypotd_snf(real_T u0, real_T u1);
90 extern real_T rtGetInf(void);
91 extern real32_T rtGetInfF(void);
92 extern real_T rtGetMinusInf(void);
93 extern real32_T rtGetMinusInfF(void);
94
95 /*===========*
96 * Constants *
97 *===========*/
98 #define RT_PI 3.14159265358979323846
99 #define RT_PIF 3.1415927F
100 #define RT_LN_10 2.30258509299404568402
101 #define RT_LN_10F 2.3025851F
102 #define RT_LOG10E 0.43429448190325182765
103 #define RT_LOG10EF 0.43429449F
104 #define RT_E 2.7182818284590452354
105 #define RT_EF 2.7182817F
106
107 /*
108 * UNUSED_PARAMETER(x)
109 * Used to specify that a function parameter (argument) is required but not
110 * accessed by the function body.
111 */
112 #ifndef UNUSED_PARAMETER
```

```
113 # if defined(__LCC__)
114 # define UNUSED_PARAMETER(x) /* do nothing */
115 # else
116
117 /*
118 * This is the semi-ANSI standard way of indicating that an
119 * unused function parameter is required.
120 */
121 # define UNUSED_PARAMETER(x) (void) (x)
122 # endif
123 #endif
124
125 #ifndef INTERP
126 # define INTERP(x,x1,x2,y1,y2) ( (y1)+(((y2) - (y1))/((x2) - (x1)))*((x)-(x1)) )
127 #endif
128
129 #ifndef ZEROTECHNIQUE
130 #define ZEROTECHNIQUE
131
132 typedefenum{
133 NORMAL_INTERP,
134 AVERAGE_VALUE,
135 MIDDLE_VALUE
136 } ZeroTechnique;
137
138 #endif
139
140 extern int_T rt_GetLookupIndex(const real_T *x, int_T xlen, real_T u) ;
141 extern real_T rt_Lookup(const real_T *x, int_T xlen, real_T u, const real_T *y);
142 extern real_T rtInf;
143 extern real_T rtMinusInf;
144 extern real_T rtNaN;
145 extern real32_T rtInfF;
146 extern real32_T rtMinusInfF;
147 extern real32_T rtNaNF;
148 externvoid rt_InitInfAndNaN(size_t realSize);
149 extern boolean_T rtIsInf(real_T value);
150 extern boolean_T rtIsInfF(real32_T value);
```

```
151 extern boolean_T rtIsNaN(real_T value);
152 extern boolean_T rtIsNaNF(real32_T value);
153 typedefstruct{
154 struct{
155 uint32_T wordH;
156 uint32_T wordL;
157 } words;
158 } BigEndianIEEEDouble;
159
160 typedefstruct{
161 struct{
162 uint32_T wordL;
163 uint32_T wordH;
164 } words;
165 } LittleEndianIEEEDouble;
166
167 typedefstruct{
168 union{
169 real32_T wordLreal;
170 uint32_T wordLuint;
171 } wordL;
172 } IEEESingle;
173
174 real_T rtInf;
175 real_T rtMinusInf;
176 real_T rtNaN;
177 real32_T rtInfF;
178 real32_T rtMinusInfF;
179 real32_T rtNaNF;
180 extern real_T rtGetNaN(void);
181 extern real32_T rtGetNaNF(void);
182
183 /*
184 * Initialize rtInf needed by the generated code.
185 * Inf is initialized as non-signaling. Assumes IEEE.
186 */
187 real_T rtGetInf(void)
188 {
```

```
189 size_t bitsPerReal = sizeof(real_T) * (NumBitsPerChar);
190 real_T inf = 0.0;
191 if (bitsPerReal == 32U) {
192 inf = rtGetInfF();
193 }else{
194 union{
195 LittleEndianIEEEDouble bitVal;
196 real_T fltVal;
197 } tmpVal;
198
199 tmpVal.bitVal.words.wordH = 0x7FF00000U;
200 tmpVal.bitVal.words.wordL = 0x00000000U;
201 inf = tmpVal.fltVal;
202 }
203
204 return inf;
205 }
206
207 /*
208 * Initialize rtInfF needed by the generated code.
209 * Inf is initialized as non-signaling. Assumes IEEE.
210 */
211 real32_T rtGetInfF(void)
212 {
213 IEEESingle infF;
214 infF.wordL.wordLuint = 0x7F800000U;
215 return infF.wordL.wordLreal;
216 }
217
218 /*
219 * Initialize rtMinusInf needed by the generated code.
220 * Inf is initialized as non-signaling. Assumes IEEE.
221 */
222 real_T rtGetMinusInf(void)
223 {
224 size_t bitsPerReal = sizeof(real_T) * (NumBitsPerChar);
225 real_T minf = 0.0;
226 if (bitsPerReal == 32U) {
```

```
227 minf = rtGetMinusInfF();
228 }else{
229 union{
230 LittleEndianIEEEDouble bitVal;
231 real_T fltVal;
232 } tmpVal;
233
234 tmpVal.bitVal.words.wordH = 0xFFF00000U;
235 tmpVal.bitVal.words.wordL = 0x00000000U;
236 minf = tmpVal.fltVal;
237 }
238
239 return minf;
240 }
241
242 /*
243 * Initialize rtMinusInfF needed by the generated code.
244 * Inf is initialized as non-signaling. Assumes IEEE.
245 */
246 real32_T rtGetMinusInfF(void)
247 {
248 IEEESingle minfF;
249 minfF.wordL.wordLuint = 0xFF800000U;
250 return minfF.wordL.wordLreal;
251 }
252
253 /*
254 * Routine to get the index of the input from a table using binary or
255 * interpolation search.
256 *
257 * Inputs:
258 * *x : Pointer to table, x[0] ....x[xlen-1]
259 * xlen : Number of values in xtable
260 * u : input value to look up
261 *
262 * Output:
263 * idx : the index into the table such that:
264 * if u is negative
```

```
265 * x[idx] <= u < x[idx+1]
266 * else
267 * x[idx] < u <= x[idx+1]
268 *
269 * Interpolation Search: If the table contains a large number of nearly
270 * uniformly spaced entries, i.e., x[n] vs n is linear then the index
271 * corresponding to the input can be found in one shot using the linear
272 * interpolation formula. Therefore if you have a look-up table block with
273 * many data points, using interpolation search might speed up the code.
274 * Compile the generated code with the following flag:
275 *
276 * make_rtw OPTS=-DDOINTERPSEARCH
277 *
278 * to enable interpolation search.
279 */
280 int_T rt_GetLookupIndex(const real_T *x, int_T xlen, real_T u)
281 {
282 int_T idx = 0;
283 int_T bottom = 0;
284 int_T top = xlen-1;
285 int_T retValue = 0;
286 boolean_T returnStatus = 0U;
287
288 #ifdef DOINTERPSEARCH
289
290 real_T offset = 0;
291
292 #endif
293
294 /*
295 * Deal with the extreme cases first:
296 * if u <= x[bottom] then return idx = bottom
297 * if u >= x[top] then return idx = top-1
298 */
299 if (u <= x[bottom]) {
300 retValue = bottom;
301 returnStatus = 1U;
302 }elseif (u >= x[top]) {
```

```
303 retValue = top-1;
304 returnStatus = 1U;
305 }else{
306 /* else required to ensure safe programming, even *
307 * if it's expected that it will never be reached */
308 }
309
310 if (returnStatus == 0U) {
311 if (u < 0) {
312 /* For negative input find index such that: x[idx] <= u < x[idx+1] */
313 for (;;) {
314
315 #ifdef DOINTERPSEARCH
316
317 offset = (u-x[bottom])/(x[top]-x[bottom]);
318 idx = bottom + (int_T)((top-bottom)*(offset-DBL_EPSILON));
319
320 #else
321
322 idx = (bottom + top)/2;
323
324 #endif
325
326 if (u < x[idx]) {
327 top = idx - 1;
328 }elseif (u >= x[idx+1]) {
329 bottom = idx + 1;
330 }else{
331 /* we have x[idx] <= u < x[idx+1], return idx */
332 retValue = idx;
333 break;
334 }
335 }
336 }else{
337 /* For non-negative input find index such that: x[idx] < u <= x[idx+1] */
338 for (;;) {
339
340 #ifdef DOINTERPSEARCH
```

```
341
342 offset = (u-x[bottom])/(x[top]-x[bottom]);
343 idx = bottom + (int_T)((top-bottom)*(offset-DBL_EPSILON));
344
345 #else
346
347 idx = (bottom + top)/2;
348
349 #endif
350
351 if (u <= x[idx]) {
352 top = idx - 1;
353 }elseif (u > x[idx+1]) {
354 bottom = idx + 1;
355 }else{
356 /* we have x[idx] < u <= x[idx+1], return idx */
357 retValue = idx;
358 break;
359 }
360 }
361 }
362 }
363
364 return retValue;
365 }
366
367 /* 1D lookup routine for data type of real_T. */
368 real_T rt_Lookup(const real_T *x, int_T xlen, real_T u, const real_T *y)
369 {
370 int_T idx = rt_GetLookupIndex(x, xlen, u);
371 real_T num = y[idx+1] - y[idx];
372 real_T den = x[idx+1] - x[idx];
373
374 /* Due to the way the binary search is implemented
375 in rt_look.c (rt_GetLookupIndex), den cannot be
376 0. Equivalently, m cannot be inf or nan. */
377 real_T m = num/den;
378 return (y[idx] + (m * (u - x[idx])));
```

```
379 }
380
381 /*
382 * Initialize the rtInf, rtMinusInf, and rtNaN needed by the
383 * generated code. NaN is initialized as non-signaling. Assumes IEEE.
384 */
385 void rt_InitInfAndNaN(size_t realSize)
386 {
387 (void) (realSize);
388 rtNaN = rtGetNaN();
389 rtNaNF = rtGetNaNF();
390 rtInf = rtGetInf();
391 rtInfF = rtGetInfF();
392 rtMinusInf = rtGetMinusInf();
393 rtMinusInfF = rtGetMinusInfF();
394 }
395
396 /* Test if value is infinite */
397 boolean_T rtIsInf(real_T value)
398 {
399 return (boolean_T)((value==rtInf || value==rtMinusInf) ? 1U : 0U);
400 }
401
402 /* Test if single-precision value is infinite */
403 boolean_T rtIsInfF(real32_T value)
404 {
405 return (boolean_T)(((value)==rtInfF || (value)==rtMinusInfF) ? 1U : 0U);
406 }
407
408 /* Test if value is not a number */
409 boolean_T rtIsNaN(real_T value)
410 {
411 return (boolean_T)((value!=value) ? 1U : 0U);
412 }
413
414 /* Test if single-precision value is not a number */
415 boolean_T rtIsNaNF(real32_T value)
416 {
```

```
417 return (boolean_T)(((value!=value) ? 1U : 0U));
418 }
419
420 /*
421 * Initialize rtNaN needed by the generated code.
422 * NaN is initialized as non-signaling. Assumes IEEE.
423 */
424 real_T rtGetNaN(void)
425 {
426 size_t bitsPerReal = sizeof(real_T) * (NumBitsPerChar);
427 real_T nan = 0.0;
428 if (bitsPerReal == 32U) {
429 nan = rtGetNaNF();
430 }else{
431 union{
432 LittleEndianIEEEDouble bitVal;
433 real_T fltVal;
434 } tmpVal;
435
436 tmpVal.bitVal.words.wordH = 0xFFF80000U;
437 tmpVal.bitVal.words.wordL = 0x00000000U;
438 nan = tmpVal.fltVal;
439 }
440
441 return nan;
442 }
443
444 /*
445 * Initialize rtNaNF needed by the generated code.
446 * NaN is initialized as non-signaling. Assumes IEEE.
447 */
448 real32_T rtGetNaNF(void)
449 {
450 IEEESingle nanF = {{ 0 }};
451
452 nanF.wordL.wordLuint = 0xFFC00000U;
453 return nanF.wordL.wordLreal;
454 }
```

```
455
456 real_T rt_hypotd_snf(real_T u0, real_T u1)
457 {
458 real_T y;
459 real_T a;
460 a = fabs(u0);
461 y = fabs(u1);
462 if (a < y) {
463 a /= y;
464 y *= sqrt(a * a + 1.0);
465 }elseif (a > y) {
466 y /= a;
467 y = sqrt(y * y + 1.0) * a;
468 }else{
469 if (!rtIsNaN(y)) {
470 y = a * 1.4142135623730951;
471 }
472 }
473
474 return y;
475 }
476
477 /* Model step function */
478 void NEWPDFDEVELOPMENTDESERTATIONamusingnow_step(void)
479 {
480 {
481 real_T rtb_Gain23;
482 real_T rtb_Gain26_g;
483 real_T rtb_Product5;
484 real_T rtb_Gain25;
485 real_T rtb_Product6;
486 real_T rtb_Product4_a;
487 real_T rtb_Product;
488 real_T rtb_LookUpTable;
489 real_T rtb_Kv1_idx_0;
490 real_T rtb_Kv1_idx_1;
491 real_T rtb_Kv1_idx_2;
492 real_T rtb_Kv_idx_0;
```

```
493 real_T rtb_Kv_idx_1;
494 real_T rtb_Kv_idx_2;
495 real_T rtb_Sum5_re;
496 real_T rtb_Sum5_im;
497 real_T rtb_Sum11_re;
498 real_T rtb_Sum11_im;
499 real_T rtb_Sum19_re;
500 real_T rtb_Sum19_im;
501
502 /* Sin: '<S75>/Sine Wave A' */
503 if (rtDW.systemEnable != 0) {
504 rtDW.lastSin = sin(314.15926535897933 * ((rtM->Timing.clockTick1) * 5.0E-5));
505 rtDW.lastCos = cos(314.15926535897933 * ((rtM->Timing.clockTick1) * 5.0E-5));
506 rtDW.systemEnable = 0;
507 }
508
509 rtDW.SineWaveA = ((rtDW.lastSin * 0.71812629776318893 + rtDW.lastCos *
510 0.69591279659231431) * 0.99987663248166059 +
511 (rtDW.lastCos * 0.71812629776318893 - rtDW.lastSin *
512 0.69591279659231431) * 0.015707317311820675) *
513 393388.05269097845;
514
515 /* End of Sin: '<S75>/Sine Wave A' */
516
517 /* Sin: '<S75>/Sine Wave B' */
518 if (rtDW.systemEnable_p != 0) {
519 rtDW.lastSin_b = sin(314.15926535897933 * ((rtM->Timing.clockTick1) *
520 5.0E-5));
521 rtDW.lastCos_h = cos(314.15926535897933 * ((rtM->Timing.clockTick1) *
522 5.0E-5));
523 rtDW.systemEnable_p = 0;
524 }
525
526 rtDW.SineWaveB = ((rtDW.lastSin_b * 0.24361501178602257 + rtDW.lastCos_h *
527 -0.96987201528474676) * 0.99987663248166059 +
528 (rtDW.lastCos_h * 0.24361501178602257 - rtDW.lastSin_b *
529 -0.96987201528474676) * 0.015707317311820675) *
530 393388.05269097845;
```

```
531
532 /* End of Sin: '<S75>/Sine Wave B' */
533
534 /* Sin: '<S75>/Sine Wave C' */
535 if (rtDW.systemEnable_b != 0) {
536 rtDW.lastSin_e = sin(314.15926535897933 * ((rtM->Timing.clockTick1) *
537 5.0E-5));
538 rtDW.lastCos_p = cos(314.15926535897933 * ((rtM->Timing.clockTick1) *
539 5.0E-5));
540 rtDW.systemEnable_b = 0;
541 }
542
543 rtDW.SineWaveC = ((rtDW.lastSin_e * -0.96174130954921133 + rtDW.lastCos_p *
544 0.27395921869243262) * 0.99987663248166059 +
545 (rtDW.lastCos_p * -0.96174130954921133 - rtDW.lastSin_e *
546 0.27395921869243262) * 0.015707317311820675) *
547 393388.05269097845;
548
549 /* End of Sin: '<S75>/Sine Wave C' */
550
551 /* S-Function (sfun_spssw_discc): '<S76>/State-Space' incorporates:
552 * Constant: '<S78>/SwitchCurrents'
553 */
554
555 /* S-Function block: <S76>/State-Space */
556 {
557 real_T accum;
558
559 /* Circuit has switches */
560 int_T *switch_status = (int_T*) rtDW.StateSpace_PWORK.SWITCH_STATUS;
561 int_T *switch_status_init = (int_T*)
562 rtDW.StateSpace_PWORK.SWITCH_STATUS_INIT;
563 int_T *SwitchChange = (int_T*) rtDW.StateSpace_PWORK.SW_CHG;
564 int_T *gState = (int_T*) rtDW.StateSpace_PWORK.G_STATE;
565 real_T *yswitch = (real_T*)rtDW.StateSpace_PWORK.Y_SWITCH;
566 int_T *switchTypes = (int_T*) rtDW.StateSpace_PWORK.SWITCH_TYPES;
567 int_T *idxOutSw = (int_T*) rtDW.StateSpace_PWORK.IDX_OUT_SW;
568 real_T *DxCol = (real_T*)rtDW.StateSpace_PWORK.DX_COL;
```

```
569 real_T *tmp2 = (real_T*)rtDW.StateSpace_PWORK.TMP2;
570 real_T *BDcol = (real_T*)rtDW.StateSpace_PWORK.BD_COL;
571 real_T *tmp1 = (real_T*)rtDW.StateSpace_PWORK.TMP1;
572 real_T *uswlast = (real_T*)rtDW.StateSpace_PWORK.USWLAST;
573 int_T newState;
574 int_T swChanged = 0;
575 int loopsToDo = 20;
576 real_T temp;
577
578 /* keep an initial copy of switch_status*/
579 memcpy(switch_status_init, switch_status, 9 * sizeof(int_T));
580 memcpy(uswlast, &rtDW.StateSpace_o1[0], 9*sizeof(real_T));
581 do{
582 if (loopsToDo == 1) {/* Need to reset some variables: */
583 swChanged = 0;
584
585 /* return to the original switch status*/
586 {
587 int_T i1;
588 for (i1=0; i1 < 9; i1++) {
589 swChanged = ((SwitchChange[i1] = switch_status_init[i1] -
590 switch_status[i1]) != 0) ? 1 : swChanged;
591 switch_status[i1] = switch_status_init[i1];
592 }
593 }
594 }else{
595 /*
596 * Compute outputs:
597 * ---------------
598 */
599 real_T *Cs = (real_T*)rtDW.StateSpace_PWORK.CS;
600 real_T *Ds = (real_T*)rtDW.StateSpace_PWORK.DS;
601
602 {
603 int_T i1;
604 real_T *y0 = &rtDW.StateSpace_o1[0];
605 for (i1=0; i1 < 21; i1++) {
606 accum = 0.0;
```

```
607
608 {
609 int_T i2;
610 real_T *xd = &rtDW.StateSpace_DSTATE[0];
611 for (i2=0; i2 < 20; i2++) {
612 accum += *(Cs++) * xd[i2];
613 }
614 }
615
616 {
617 int_T i2;
618 const real_T *u0 = rtConstP.SwitchCurrents_Value;
619 for (i2=0; i2 < 9; i2++) {
620 accum += *(Ds++) * u0[i2];
621 }
622
623 accum += *(Ds++) * rtDW.SineWaveA;
624 accum += *(Ds++) * rtDW.SineWaveB;
625 accum += *(Ds++) * rtDW.SineWaveC;
626 }
627
628 y0[i1] = accum;
629 }
630 }
631
632 swChanged = 0;
633
634 {
635 int_T i1;
636 real_T *y0 = &rtDW.StateSpace_o1[0];
637 for (i1=0; i1 < 9; i1++) {
638 newState = (gState[i1] > 0) ? 1 : ((y0[i1]*uswlast[i1] < 0.0) ? 0 :
639 switch_status[i1]);
640 swChanged = ((SwitchChange[i1] = newState - switch_status[i1]) !=
641 0) ? 1 : swChanged;
642 switch_status[i1] = newState;/* Keep new state */
643 }
644 }
```

```
645 }
646
647 /*
648 * Compute new As, Bs, Cs and Ds matrixes:
649 * -------------------------------------
650 */
651 if (swChanged) {
652 real_T *As = (real_T*)rtDW.StateSpace_PWORK.AS;
653 real_T *Cs = (real_T*)rtDW.StateSpace_PWORK.CS;
654 real_T *Bs = (real_T*)rtDW.StateSpace_PWORK.BS;
655 real_T *Ds = (real_T*)rtDW.StateSpace_PWORK.DS;
656 real_T a1;
657
658 {
659 int_T i1;
660 for (i1=0; i1 < 9; i1++) {
661 if (SwitchChange[i1] != 0) {
662 a1 = 1000.0*SwitchChange[i1];
663 temp = 1/(1-Ds[i1*13]*a1);
664
665 {
666 int_T i2;
667 for (i2=0; i2 < 21; i2++) {
668 DxCol[i2]= Ds[i2 * 12 + i1]*temp*a1;
669 }
670 }
671
672 DxCol[i1] = temp;
673
674 {
675 int_T i2;
676 for (i2=0; i2 < 20; i2++) {
677 BDcol[i2]= Bs[i2 * 12 + i1]*a1;
678 }
679 }
680
681 /* Copy row nSw of Cs into tmp1 and zero it out in Cs */
682 memcpy(tmp1, &Cs[i1 * 20], 20 * sizeof(real_T));
```

```
683 memset(&Cs[i1 * 20], '\0', 20 * sizeof(real_T));
684
685 /* Copy row nSw of Ds into tmp2 and zero it out in Ds */
686 memcpy(tmp2, &Ds[i1 * 12], 12 * sizeof(real_T));
687 memset(&Ds[i1 * 12], '\0', 12 * sizeof(real_T));
688
689 /* Cs = Cs + DxCol * tmp1, Ds = Ds + DxCol * tmp2 ******************/
690 {
691 int_T i2;
692 for (i2=0; i2 < 21; i2++) {
693 a1 = DxCol[i2];
694
695 {
696 int_T i3;
697 for (i3=0; i3 < 20; i3++) {
698 Cs[i2 * 20 + i3] += a1 * tmp1[i3];
699 }
700 }
701
702 {
703 int_T i3;
704 for (i3=0; i3 < 12; i3++) {
705 Ds[i2 * 12 + i3] += a1 * tmp2[i3];
706 }
707 }
708 }
709 }
710
711 /* As = As + BdCol*Cs(nSw,:), Bs = Bs + BdCol*Ds(nSw,:) *************/
712 {
713 int_T i2;
714 for (i2=0; i2 < 20; i2++) {
715 a1 = BDcol[i2];
716
717 {
718 int_T i3;
719 for (i3=0; i3 < 20; i3++) {
720 As[i2 * 20 + i3] += a1 * Cs[i1 * 20 + i3];
```

```
721 }
722 }
723
724 {
725 int_T i3;
726 for (i3=0; i3 < 12; i3++) {
727 Bs[i2 * 12 + i3] += a1 * Ds[i1 * 12 + i3];
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }/* if (swChanged) */
736 }while (swChanged > 0 && --loopsToDo > 0);
737
738 if (loopsToDo == 0) {
739 real_T *Cs = (real_T*)rtDW.StateSpace_PWORK.CS;
740 real_T *Ds = (real_T*)rtDW.StateSpace_PWORK.DS;
741
742 {
743 int_T i1;
744 real_T *y0 = &rtDW.StateSpace_o1[0];
745 for (i1=0; i1 < 21; i1++) {
746 accum = 0.0;
747
748 {
749 int_T i2;
750 real_T *xd = &rtDW.StateSpace_DSTATE[0];
751 for (i2=0; i2 < 20; i2++) {
752 accum += *(Cs++) * xd[i2];
753 }
754 }
755
756 {
757 int_T i2;
758 const real_T *u0 = rtConstP.SwitchCurrents_Value;
```

```
759 for (i2=0; i2 < 9; i2++) {
760 accum += *(Ds++) * u0[i2];
761 }
762
763 accum += *(Ds++) * rtDW.SineWaveA;
764 accum += *(Ds++) * rtDW.SineWaveB;
765 accum += *(Ds++) * rtDW.SineWaveC;
766 }
767
768 y0[i1] = accum;
769 }
770 }
771 }
772
773 /* Output new switches states */
774 {
775 int_T i1;
776 real_T *y1 = &rtDW.StateSpace_o2[0];
777 for (i1=0; i1 < 9; i1++) {
778 y1[i1] = (real_T)switch_status[i1];
779 }
780 }
781 }
782
783 /* Gain: '<S3>/Kv1' */
784 rtb_Kv1_idx_0 = 3.7113480951260267E-6 * rtDW.StateSpace_o1[9];
785
786 /* Gain: '<S3>/Kv' */
787 rtb_Kv_idx_0 = 0.00447729929721086 * rtDW.StateSpace_o1[15];
788
789 /* Gain: '<S3>/Kv1' */
790 rtb_Kv1_idx_1 = 3.7113480951260267E-6 * rtDW.StateSpace_o1[10];
791
792 /* Gain: '<S3>/Kv' */
793 rtb_Kv_idx_1 = 0.00447729929721086 * rtDW.StateSpace_o1[16];
794
795 /* Gain: '<S3>/Kv1' */
796 rtb_Kv1_idx_2 = 3.7113480951260267E-6 * rtDW.StateSpace_o1[11];
```

```
797
798 /* Gain: '<S3>/Kv' */
799 rtb_Kv_idx_2 = 0.00447729929721086 * rtDW.StateSpace_o1[17];
800
801 /* Gain: '<S1>/Gain53' incorporates:
802 * Gain: '<S1>/Gain4'
803 * Product: '<S1>/Product1'
804 * Product: '<S1>/Product4'
805 * Sum: '<S1>/Add2'
806 */
807 rtb_Gain23 = ((rtb_Kv1_idx_0 + rtb_Kv1_idx_1) + rtb_Kv1_idx_2) * 0.667 *
808 1.0E+15;
809
810 /* Product: '<S1>/Product14' */
811 rtb_Gain26_g = rtb_Gain23 * 0.85090352453411844;
812
813 /* Product: '<S1>/Product15' */
814 rtb_Product5 = rtb_Gain23 * -0.22286052796839617;
815
816 /* Product: '<S1>/Product16' */
817 rtb_Gain25 = rtb_Gain23 * -0.43451132285096439;
818
819 /* Product: '<S1>/Product17' */
820 rtb_Gain23 *= -0.62430649351725087;
821
822 /* Sum: '<S1>/Sum2' incorporates:
823 * Sum: '<S1>/Sum'
824 * Sum: '<S1>/Sum1'
825 */
826 rtb_Product6 = (rtb_Gain26_g + rtb_Product5) + (rtb_Gain25 + rtb_Gain23);
827
828 /* Outport: '<Root>/Out19' */
829 rtY.Out19 = rtb_Product6;
830
831 /* Sum: '<S1>/Sum5' incorporates:
832 * Gain: '<S1>/Gain18'
833 * Gain: '<S1>/Gain19'
834 * Gain: '<S1>/Gain20'
```

```
835  * Sum: '<S1>/Sum3'
836  * Sum: '<S1>/Sum4'
837  */
838  rtb_Sum5_re = (-0.0 * rtb_Product5 + rtb_Gain26_g) + (0.0 * rtb_Gain23 +
839  -rtb_Gain25);
840  rtb_Sum5_im = -rtb_Product5 + rtb_Gain23;
841
842  /* Outport: '<Root>/Out20' */
843  rtY.Out20.re = rtb_Sum5_re;
844  rtY.Out20.im = rtb_Sum5_im;
845
846  /* Sum: '<S1>/Sum8' incorporates:
847  * Gain: '<S1>/Gain24'
848  * Gain: '<S1>/Gain26'
849  * Sum: '<S1>/Sum6'
850  * Sum: '<S1>/Sum7'
851  */
852  rtb_Product4_a = (rtb_Gain26_g + -rtb_Product5) + (rtb_Gain25 + -rtb_Gain23);
853
854  /* Outport: '<Root>/Out21' */
855  rtY.Out21 = rtb_Product4_a;
856
857  /* Gain: '<S1>/Gain31' */
858  rtb_Gain25 = -rtb_Gain25;
859
860  /* Sum: '<S1>/Sum11' incorporates:
861  * Gain: '<S1>/Gain30'
862  * Gain: '<S1>/Gain32'
863  * Sum: '<S1>/Sum10'
864  * Sum: '<S1>/Sum9'
865  */
866  rtb_Sum11_re = (0.0 * rtb_Product5 + rtb_Gain26_g) + (-0.0 * rtb_Gain23 +
867  rtb_Gain25);
868  rtb_Sum11_im = rtb_Product5 + -rtb_Gain23;
869
870  /* Outport: '<Root>/Out22' */
871  rtY.Out22.re = rtb_Sum11_re;
872  rtY.Out22.im = rtb_Sum11_im;
```

```
873
874 /* Gain: '<S1>/Gain2' incorporates:
875  * Product: '<S1>/Product2'
876  * Product: '<S1>/Product3'
877  * Sum: '<S1>/Add1'
878  */
879 rtb_Gain25 = ((rtb_Kv_idx_0 + rtb_Kv_idx_1) + rtb_Kv_idx_2) * 0.667;
880
881 /* Product: '<S1>/Product12' */
882 rtb_Product5 = rtb_Gain25 * -0.43451132285096439;
883
884 /* Outport: '<Root>/Out23' */
885 rtY.Out23 = rtb_Product5;
886
887 /* Product: '<S1>/Product13' */
888 rtb_Gain26_g = rtb_Gain25 * -0.62430649351725087;
889
890 /* Outport: '<Root>/Out24' */
891 rtY.Out24 = rtb_Gain26_g;
892
893 /* Product: '<S1>/Product8' */
894 rtb_Gain23 = rtb_Gain25 * 0.85090352453411844;
895
896 /* Product: '<S1>/Product9' */
897 rtb_Gain25 *= -0.22286052796839617;
898
899 /* Sum: '<S1>/Sum16' incorporates:
900  * Sum: '<S1>/Sum12'
901  * Sum: '<S1>/Sum13'
902  */
903 rtb_Product = (rtb_Gain23 + rtb_Gain25) + (rtb_Product5 + rtb_Gain26_g);
904
905 /* Outport: '<Root>/Out25' */
906 rtY.Out25 = rtb_Product;
907
908 /* Sum: '<S1>/Sum19' incorporates:
909  * Gain: '<S1>/Gain10'
910  * Gain: '<S1>/Gain11'
```

```
911  * Gain: '<S1>/Gain50'
912  * Sum: '<S1>/Sum17'
913  * Sum: '<S1>/Sum18'
914  */
915  rtb_Sum19_re = (-0.0 * rtb_Gain25 + rtb_Gain23) + (0.0 * rtb_Gain26_g +
916  -rtb_Product5);
917  rtb_Sum19_im = -rtb_Gain25 + rtb_Gain26_g;
918
919  /* Outport: '<Root>/Out26' */
920  rtY.Out26.re = rtb_Sum19_re;
921  rtY.Out26.im = rtb_Sum19_im;
922
923  /* Sum: '<S1>/Sum22' incorporates:
924  * Gain: '<S1>/Gain38'
925  * Gain: '<S1>/Gain40'
926  * Sum: '<S1>/Sum20'
927  * Sum: '<S1>/Sum21'
928  */
929  rtb_LookUpTable = (rtb_Gain23 + -rtb_Gain25) + (rtb_Product5 + -rtb_Gain26_g);
930
931  /* Outport: '<Root>/Out27' */
932  rtY.Out27 = rtb_LookUpTable;
933
934  /* Gain: '<S1>/Gain45' */
935  rtb_Product5 = -rtb_Product5;
936
937  /* Sum: '<S1>/Sum15' incorporates:
938  * Gain: '<S1>/Gain44'
939  * Gain: '<S1>/Gain46'
940  * Sum: '<S1>/Sum14'
941  * Sum: '<S1>/Sum23'
942  */
943  rtb_Gain23 = (0.0 * rtb_Gain25 + rtb_Gain23) + (-0.0 * rtb_Gain26_g +
944  rtb_Product5);
945  rtb_Gain26_g = rtb_Gain25 + -rtb_Gain26_g;
946
947  /* Outport: '<Root>/Out28' */
948  rtY.Out28.re = rtb_Gain23;
```

```
949  rtY.Out28.im = rtb_Gain26_g;
950
951  /* Outport: '<Root>/Out29' incorporates:
952  * Product: '<S1>/Divide0'
953  */
954  rtY.Out29 = rtb_Product6 / rtb_Product;
955
956  /* Product: '<S1>/Divide1' incorporates:
957  * ComplexToMagnitudeAngle: '<S1>/Complex to Magnitude-Angle2'
958  */
959  rtb_Product6 = rt_hypotd_snf(rtb_Sum5_re, rtb_Sum5_im);
960  if (rtb_Sum19_im == 0.0) {
961  rtb_Sum19_re = rtb_Product6 / rtb_Sum19_re;
962  rtb_Sum19_im = 0.0;
963  }elseif (rtb_Sum19_re == 0.0) {
964  if (rtb_Product6 == 0.0) {
965  rtb_Sum19_re = 0.0 / rtb_Sum19_im;
966  rtb_Sum19_im = 0.0;
967  }else{
968  rtb_Sum19_re = 0.0;
969  rtb_Sum19_im = -(rtb_Product6 / rtb_Sum19_im);
970  }
971  }else{
972  rtb_Sum5_re = fabs(rtb_Sum19_re);
973  rtb_Sum5_im = fabs(rtb_Sum19_im);
974  if (rtb_Sum5_re > rtb_Sum5_im) {
975  rtb_Sum5_re = rtb_Sum19_im / rtb_Sum19_re;
976  rtb_Sum19_im = rtb_Sum5_re * rtb_Sum19_im + rtb_Sum19_re;
977  rtb_Sum19_re = rtb_Product6 / rtb_Sum19_im;
978  rtb_Sum19_im = -(rtb_Sum5_re * rtb_Product6 / rtb_Sum19_im);
979  }elseif (rtb_Sum5_im == rtb_Sum5_re) {
980  rtb_Sum19_re = (rtb_Sum19_re > 0.0 ? 0.5 : -0.5) * rtb_Product6 /
981  rtb_Sum5_re;
982  rtb_Sum19_im = (rtb_Sum19_im > 0.0 ? 0.5 : -0.5) * -rtb_Product6 /
983  rtb_Sum5_re;
984  }else{
985  rtb_Sum5_re = rtb_Sum19_re / rtb_Sum19_im;
986  rtb_Sum19_im += rtb_Sum5_re * rtb_Sum19_re;
```

```
987  rtb_Sum19_re = rtb_Sum5_re * rtb_Product6 / rtb_Sum19_im;
988  rtb_Sum19_im = -(rtb_Product6 / rtb_Sum19_im);
989  }
990  }
991
992  /* Outport: '<Root>/Out30' incorporates:
993  * Product: '<S1>/Divide1'
994  */
995  rtY.Out30.re = rtb_Sum19_re;
996  rtY.Out30.im = rtb_Sum19_im;
997
998  /* Outport: '<Root>/Out31' incorporates:
999  * Product: '<S1>/Divide2'
1000 */
1001 rtY.Out31 = rtb_Product4_a / rtb_LookUpTable;
1002
1003 /* Product: '<S1>/Divide3' incorporates:
1004 * ComplexToMagnitudeAngle: '<S1>/Complex to Magnitude-Angle1'
1005 */
1006 rtb_Product6 = rt_hypotd_snf(rtb_Sum11_re, rtb_Sum11_im);
1007 if (rtb_Gain26_g == 0.0) {
1008 rtb_Sum19_re = rtb_Product6 / rtb_Gain23;
1009 rtb_Sum19_im = 0.0;
1010 }elseif (rtb_Gain23 == 0.0) {
1011 if (rtb_Product6 == 0.0) {
1012 rtb_Sum19_re = 0.0 / rtb_Gain26_g;
1013 rtb_Sum19_im = 0.0;
1014 }else{
1015 rtb_Sum19_re = 0.0;
1016 rtb_Sum19_im = -(rtb_Product6 / rtb_Gain26_g);
1017 }
1018 }else{
1019 rtb_Sum5_re = fabs(rtb_Gain23);
1020 rtb_Sum5_im = fabs(rtb_Gain26_g);
1021 if (rtb_Sum5_re > rtb_Sum5_im) {
1022 rtb_Sum5_re = rtb_Gain26_g / rtb_Gain23;
1023 rtb_Sum19_im = rtb_Sum5_re * rtb_Gain26_g + rtb_Gain23;
1024 rtb_Sum19_re = rtb_Product6 / rtb_Sum19_im;
```

```
1025 rtb_Sum19_im = -(rtb_Sum5_re * rtb_Product6 / rtb_Sum19_im);
1026 }elseif (rtb_Sum5_im == rtb_Sum5_re) {
1027 rtb_Sum19_re = (rtb_Gain23 > 0.0 ? 0.5 : -0.5) * rtb_Product6 /
1028 rtb_Sum5_re;
1029 rtb_Sum19_im = (rtb_Gain26_g > 0.0 ? 0.5 : -0.5) * -rtb_Product6 /
1030 rtb_Sum5_re;
1031 }else{
1032 rtb_Sum5_re = rtb_Gain23 / rtb_Gain26_g;
1033 rtb_Sum19_im = rtb_Sum5_re * rtb_Gain23 + rtb_Gain26_g;
1034 rtb_Sum19_re = rtb_Sum5_re * rtb_Product6 / rtb_Sum19_im;
1035 rtb_Sum19_im = -(rtb_Product6 / rtb_Sum19_im);
1036 }
1037 }
1038
1039 /* Outport: '<Root>/Out32' incorporates:
1040 * ComplexToMagnitudeAngle: '<S1>/Complex to Magnitude-Angle5'
1041 * Product: '<S1>/Divide3'
1042 */
1043 rtY.Out32 = rt_hypotd_snf(rtb_Sum19_re, rtb_Sum19_im);
1044
1045 /* Outport: '<Root>/Out33' */
1046 rtY.Out33 = rtb_Gain25;
1047
1048 /* Gain: '<S2>/Gain34' incorporates:
1049 * Gain: '<S10>/Gain4'
1050 * Product: '<S10>/Product19'
1051 * Product: '<S10>/Product4'
1052 * Sum: '<S10>/Add3'
1053 */
1054 rtb_LookUpTable = ((rtb_Kv1_idx_0 + rtb_Kv1_idx_1) + rtb_Kv1_idx_2) * 0.667 *
1055 1.0E+15;
1056
1057 /* Product: '<S2>/Product' */
1058 rtb_Product = rtb_LookUpTable * 0.85090352453411844;
1059
1060 /* Product: '<S2>/Product2' */
1061 rtb_Product4_a = rtb_LookUpTable * -0.43451132285096439;
1062
```

```
/* Sum: '<S2>/Add1' */
rtb_Product6 = rtb_Product + rtb_Product4_a;

/* Product: '<S2>/Product1' */
rtb_Gain25 = rtb_LookUpTable * -0.22286052796839617;

/* Product: '<S2>/Product3' */
rtb_LookUpTable *= -0.62430649351725087;

/* Sum: '<S2>/Add2' */
rtb_Product5 = rtb_Gain25 + rtb_LookUpTable;

/* Outport: '<Root>/Out13' incorporates:
 * Gain: '<S2>/Gain18'
 * Gain: '<S2>/Gain3'
 * Sum: '<S2>/Sum3'
 */
rtY.Out13.re = 2.3659 * rtb_Product5 * -0.0 + rtb_Product6;
rtY.Out13.im = -(2.3659 * rtb_Product5);

/* Outport: '<Root>/Out1' incorporates:
 * Gain: '<S2>/Gain13'
 * Gain: '<S2>/Gain2'
 * Gain: '<S2>/Gain6'
 * Sum: '<S2>/Sum1'
 */
rtY.Out1 = 3.6521 * rtb_Product6 + -(2.3659 * rtb_Product5);

/* Gain: '<S2>/Gain7' */
rtb_Product6 *= 3.6521;

/* Gain: '<S2>/Gain8' */
rtb_Product5 *= 2.3659;

/* Outport: '<Root>/Out2' incorporates:
 * Gain: '<S2>/Gain16'
 * Sum: '<S2>/Sum2'
 */
```

```
1101 rtY.Out2.re = 0.0 * rtb_Product5 + rtb_Product6;
1102 rtY.Out2.im = rtb_Product5;
1103
1104 /* Outport: '<Root>/Out8' */
1105 rtY.Out8 = rtb_Gain25;
1106
1107 /* Outport: '<Root>/Out10' */
1108 rtY.Out10 = rtb_Product4_a;
1109
1110 /* Outport: '<Root>/Out11' */
1111 rtY.Out11 = rtb_LookUpTable;
1112
1113 /* Gain: '<S11>/Gain25' incorporates:
1114 * Product: '<S11>/Product10'
1115 * Product: '<S11>/Product11'
1116 * Sum: '<S11>/Add7'
1117 */
1118 rtb_LookUpTable = ((rtb_Kv_idx_0 + rtb_Kv_idx_1) + rtb_Kv_idx_2) * 0.667;
1119
1120 /* Product: '<S2>/Product4' */
1121 rtb_Product4_a = rtb_LookUpTable * 0.85090352453411844;
1122
1123 /* Product: '<S2>/Product6' */
1124 rtb_Product6 = rtb_LookUpTable * -0.43451132285096439;
1125
1126 /* Sum: '<S2>/Add4' */
1127 rtb_Gain25 = rtb_Product4_a + rtb_Product6;
1128
1129 /* Product: '<S2>/Product5' */
1130 rtb_Product5 = rtb_LookUpTable * -0.22286052796839617;
1131
1132 /* Product: '<S2>/Product7' */
1133 rtb_LookUpTable *= -0.62430649351725087;
1134
1135 /* Sum: '<S2>/Add5' */
1136 rtb_Gain26_g = rtb_Product5 + rtb_LookUpTable;
1137
1138 /* Outport: '<Root>/Out3' incorporates:
```

```
1139  * Sum: '<S2>/Sum7'
1140  */
1141  rtY.Out3 = rtb_Gain25 + rtb_Gain26_g;
1142
1143  /* Outport: '<Root>/Out4' incorporates:
1144  * Gain: '<S2>/Gain12'
1145  * Gain: '<S2>/Gain23'
1146  * Sum: '<S2>/Sum6'
1147  */
1148  rtY.Out4.re = 2.3659 * rtb_Gain26_g * -0.0 + rtb_Gain25;
1149  rtY.Out4.im = -(2.3659 * rtb_Gain26_g);
1150
1151  /* Outport: '<Root>/Out5' incorporates:
1152  * Gain: '<S2>/Gain20'
1153  * Gain: '<S2>/Gain24'
1154  * Gain: '<S2>/Gain5'
1155  * Sum: '<S2>/Sum4'
1156  */
1157  rtY.Out5 = 3.6521 * rtb_Gain25 + -(2.3659 * rtb_Gain26_g);
1158
1159  /* Gain: '<S2>/Gain25' */
1160  rtb_Gain25 *= 3.6521;
1161
1162  /* Gain: '<S2>/Gain26' */
1163  rtb_Gain26_g *= 2.3659;
1164
1165  /* Outport: '<Root>/Out6' incorporates:
1166  * Gain: '<S2>/Gain10'
1167  * Sum: '<S2>/Sum5'
1168  */
1169  rtY.Out6.re = 0.0 * rtb_Gain26_g + rtb_Gain25;
1170  rtY.Out6.im = rtb_Gain26_g;
1171
1172  /* Outport: '<Root>/Out7' */
1173  rtY.Out7 = rtb_Product5;
1174
1175  /* Outport: '<Root>/Out12' */
1176  rtY.Out12 = rtb_Product6;
```

```
1177
1178 /* Outport: '<Root>/Out14' */
1179 rtY.Out14 = rtb_LookUpTable;
1180
1181 /* Outport: '<Root>/Out15' */
1182 rtY.Out15 = rtb_Product4_a;
1183
1184 /* Outport: '<Root>/Out9' */
1185 rtY.Out9 = rtb_Product;
1186
1187 /* Lookup: '<S52>/Look-Up Table' incorporates:
1188 * DigitalClock: '<S52>/Digital Clock'
1189 */
1190 rtb_Gain23 = rt_Lookup(rtConstP.LookUpTable_XData, 6,
1191 ((rtM->Timing.clockTick1) * 5.0E-5), rtConstP.pooled16);
1192
1193 /* Lookup: '<S47>/Look-Up Table' incorporates:
1194 * Clock: '<S47>/Clock'
1195 */
1196 rtDW.LookUpTable_k = rt_Lookup(rtConstP.pooled17, 5, rtM->Timing.t[0],
1197 rtConstP.pooled18);
1198
1199 /* Switch: '<S46>/Switch3' */
1200 rtDW.Switch3 = rtb_Gain23;
1201
1202 /* Lookup: '<S49>/Look-Up Table' incorporates:
1203 * Clock: '<S49>/Clock'
1204 */
1205 rtDW.LookUpTable_c = rt_Lookup(rtConstP.pooled17, 5, rtM->Timing.t[0],
1206 rtConstP.pooled18);
1207
1208 /* Switch: '<S48>/Switch3' */
1209 rtDW.Switch3_e = rtb_Gain23;
1210
1211 /* Lookup: '<S51>/Look-Up Table' incorporates:
1212 * Clock: '<S51>/Clock'
1213 */
1214 rtDW.LookUpTable_f = rt_Lookup(rtConstP.pooled17, 5, rtM->Timing.t[0],
```

```
1215 rtConstP.pooled18);
1216
1217 /* Switch: '<S50>/Switch3' */
1218 rtDW.Switch3_f = rtb_Gain23;
1219
1220 /* Lookup: '<S63>/Look-Up Table' incorporates:
1221 * DigitalClock: '<S63>/Digital Clock'
1222 */
1223 rtb_Gain23 = rt_Lookup(rtConstP.LookUpTable_XData_e, 6,
1224 ((rtM->Timing.clockTick1) * 5.0E-5), rtConstP.pooled16);
1225
1226 /* Lookup: '<S58>/Look-Up Table' incorporates:
1227 * Clock: '<S58>/Clock'
1228 */
1229 rtDW.LookUpTable_b = rt_Lookup(rtConstP.pooled17, 5, rtM->Timing.t[0],
1230 rtConstP.pooled18);
1231
1232 /* Switch: '<S57>/Switch3' */
1233 rtDW.Switch3_j = rtb_Gain23;
1234
1235 /* Lookup: '<S60>/Look-Up Table' incorporates:
1236 * Clock: '<S60>/Clock'
1237 */
1238 rtDW.LookUpTable_co = rt_Lookup(rtConstP.pooled17, 5, rtM->Timing.t[0],
1239 rtConstP.pooled18);
1240
1241 /* Switch: '<S59>/Switch3' */
1242 rtDW.Switch3_o = rtb_Gain23;
1243
1244 /* Lookup: '<S62>/Look-Up Table' incorporates:
1245 * Clock: '<S62>/Clock'
1246 */
1247 rtDW.LookUpTable_i = rt_Lookup(rtConstP.pooled17, 5, rtM->Timing.t[0],
1248 rtConstP.pooled18);
1249
1250 /* Switch: '<S61>/Switch3' */
1251 rtDW.Switch3_n = rtb_Gain23;
1252
```

```
1253 /* Lookup: '<S74>/Look-Up Table' incorporates:
1254 * DigitalClock: '<S74>/Digital Clock'
1255 */
1256 rt_Lookup(rtConstP.LookUpTable_XData_i, 6, ((rtM->Timing.clockTick1) *
1257 5.0E-5), rtConstP.LookUpTable_YData);
1258
1259 /* Lookup: '<S69>/Look-Up Table' incorporates:
1260 * Clock: '<S69>/Clock'
1261 */
1262 rtDW.LookUpTable_ko = rt_Lookup(rtConstP.pooled17, 5, rtM->Timing.t[0],
1263 rtConstP.pooled20);
1264
1265 /* Switch: '<S68>/Switch3' incorporates:
1266 * DataTypeConversion: '<S7>/Data Type Conversion4'
1267 * Logic: '<S7>/Logical Operator1'
1268 */
1269 rtDW.Switch3_i = 0.0;
1270
1271 /* Lookup: '<S71>/Look-Up Table' incorporates:
1272 * Clock: '<S71>/Clock'
1273 */
1274 rtDW.LookUpTable_o = rt_Lookup(rtConstP.pooled17, 5, rtM->Timing.t[0],
1275 rtConstP.pooled20);
1276
1277 /* Switch: '<S70>/Switch3' incorporates:
1278 * DataTypeConversion: '<S7>/Data Type Conversion5'
1279 * Logic: '<S7>/Logical Operator2'
1280 */
1281 rtDW.Switch3_d = 0.0;
1282
1283 /* Lookup: '<S73>/Look-Up Table' incorporates:
1284 * Clock: '<S73>/Clock'
1285 */
1286 rtDW.LookUpTable_a = rt_Lookup(rtConstP.pooled17, 5, rtM->Timing.t[0],
1287 rtConstP.pooled20);
1288
1289 /* Switch: '<S72>/Switch3' incorporates:
1290 * DataTypeConversion: '<S7>/Data Type Conversion6'
```

```
1291 * Logic: '<S7>/Logical Operator3'
1292 */
1293 rtDW.Switch3_g = 0.0;
1294 }
1295
1296 {
1297 real_T HoldSine;
1298 real_T *lastU;
1299
1300 /* Update for Sin: '<S75>/Sine Wave A' */
1301 HoldSine = rtDW.lastSin;
1302 rtDW.lastSin = rtDW.lastSin * 0.99987663248166059 + rtDW.lastCos *
1303 0.015707317311820675;
1304 rtDW.lastCos = rtDW.lastCos * 0.99987663248166059 - HoldSine *
1305 0.015707317311820675;
1306
1307 /* Update for Sin: '<S75>/Sine Wave B' */
1308 HoldSine = rtDW.lastSin_b;
1309 rtDW.lastSin_b = rtDW.lastSin_b * 0.99987663248166059 + rtDW.lastCos_h *
1310 0.015707317311820675;
1311 rtDW.lastCos_h = rtDW.lastCos_h * 0.99987663248166059 - HoldSine *
1312 0.015707317311820675;
1313
1314 /* Update for Sin: '<S75>/Sine Wave C' */
1315 HoldSine = rtDW.lastSin_e;
1316 rtDW.lastSin_e = rtDW.lastSin_e * 0.99987663248166059 + rtDW.lastCos_p *
1317 0.015707317311820675;
1318 rtDW.lastCos_p = rtDW.lastCos_p * 0.99987663248166059 - HoldSine *
1319 0.015707317311820675;
1320
1321 /* Update for S-Function (sfun_spssw_discc): '<S76>/State-Space' incorporates:
1322 * Constant: '<S78>/SwitchCurrents'
1323 */
1324
1325 /* S-Function block: <S76>/State-Space */
1326 {
1327 const real_T *As = (real_T*)rtDW.StateSpace_PWORK.AS;
1328 const real_T *Bs = (real_T*)rtDW.StateSpace_PWORK.BS;
```

```
1329 real_T *xtmp = (real_T*)rtDW.StateSpace_PWORK.XTMP;
1330 real_T accum;
1331
1332 /* Calculate new states... */
1333 {
1334 int_T i1;
1335 real_T *xd = &rtDW.StateSpace_DSTATE[0];
1336 for (i1=0; i1 < 20; i1++) {
1337 accum = 0.0;
1338
1339 {
1340 int_T i2;
1341 real_T *xd = &rtDW.StateSpace_DSTATE[0];
1342 for (i2=0; i2 < 20; i2++) {
1343 accum += *(As++) * xd[i2];
1344 }
1345 }
1346
1347 {
1348 int_T i2;
1349 const real_T *u0 = rtConstP.SwitchCurrents_Value;
1350 for (i2=0; i2 < 9; i2++) {
1351 accum += *(Bs++) * u0[i2];
1352 }
1353
1354 accum += *(Bs++) * rtDW.SineWaveA;
1355 accum += *(Bs++) * rtDW.SineWaveB;
1356 accum += *(Bs++) * rtDW.SineWaveC;
1357 }
1358
1359 xtmp[i1] = accum;
1360 }
1361 }
1362
1363 {
1364 int_T i1;
1365 real_T *xd = &rtDW.StateSpace_DSTATE[0];
1366 for (i1=0; i1 < 20; i1++) {
```

```
1367 xd[i1] = xtmp[i1];
1368 }
1369 }
1370
1371 {
1372 int_T *gState = (int_T*)rtDW.StateSpace_PWORK.G_STATE;
1373
1374 /* Store switch gates values for next step */
1375 *(gState++) = (int_T) rtDW.Switch3;
1376 *(gState++) = (int_T) rtDW.Switch3_e;
1377 *(gState++) = (int_T) rtDW.Switch3_f;
1378 *(gState++) = (int_T) rtDW.Switch3_j;
1379 *(gState++) = (int_T) rtDW.Switch3_o;
1380 *(gState++) = (int_T) rtDW.Switch3_n;
1381 *(gState++) = (int_T) rtDW.Switch3_i;
1382 *(gState++) = (int_T) rtDW.Switch3_d;
1383 *(gState++) = (int_T) rtDW.Switch3_g;
1384 }
1385 }
1386
1387 /* Update for Derivative: '<S47>/Derivative' */
1388 if (rtDW.TimeStampA == (rtInf)) {
1389 rtDW.TimeStampA = rtM->Timing.t[0];
1390 lastU = &rtDW.LastUAtTimeA;
1391 }elseif (rtDW.TimeStampB == (rtInf)) {
1392 rtDW.TimeStampB = rtM->Timing.t[0];
1393 lastU = &rtDW.LastUAtTimeB;
1394 }elseif (rtDW.TimeStampA < rtDW.TimeStampB) {
1395 rtDW.TimeStampA = rtM->Timing.t[0];
1396 lastU = &rtDW.LastUAtTimeA;
1397 }else{
1398 rtDW.TimeStampB = rtM->Timing.t[0];
1399 lastU = &rtDW.LastUAtTimeB;
1400 }
1401
1402 *lastU = rtDW.LookUpTable_k;
1403
1404 /* End of Update for Derivative: '<S47>/Derivative' */
```

```
1405
1406 /* Update for Derivative: '<S49>/Derivative' */
1407 if (rtDW.TimeStampA_l == (rtInf)) {
1408 rtDW.TimeStampA_l = rtM->Timing.t[0];
1409 lastU = &rtDW.LastUAtTimeA_o;
1410 }elseif (rtDW.TimeStampB_k == (rtInf)) {
1411 rtDW.TimeStampB_k = rtM->Timing.t[0];
1412 lastU = &rtDW.LastUAtTimeB_i;
1413 }elseif (rtDW.TimeStampA_l < rtDW.TimeStampB_k) {
1414 rtDW.TimeStampA_l = rtM->Timing.t[0];
1415 lastU = &rtDW.LastUAtTimeA_o;
1416 }else{
1417 rtDW.TimeStampB_k = rtM->Timing.t[0];
1418 lastU = &rtDW.LastUAtTimeB_i;
1419 }
1420
1421 *lastU = rtDW.LookUpTable_c;
1422
1423 /* End of Update for Derivative: '<S49>/Derivative' */
1424
1425 /* Update for Derivative: '<S51>/Derivative' */
1426 if (rtDW.TimeStampA_f == (rtInf)) {
1427 rtDW.TimeStampA_f = rtM->Timing.t[0];
1428 lastU = &rtDW.LastUAtTimeA_l;
1429 }elseif (rtDW.TimeStampB_e == (rtInf)) {
1430 rtDW.TimeStampB_e = rtM->Timing.t[0];
1431 lastU = &rtDW.LastUAtTimeB_l;
1432 }elseif (rtDW.TimeStampA_f < rtDW.TimeStampB_e) {
1433 rtDW.TimeStampA_f = rtM->Timing.t[0];
1434 lastU = &rtDW.LastUAtTimeA_l;
1435 }else{
1436 rtDW.TimeStampB_e = rtM->Timing.t[0];
1437 lastU = &rtDW.LastUAtTimeB_l;
1438 }
1439
1440 *lastU = rtDW.LookUpTable_f;
1441
1442 /* End of Update for Derivative: '<S51>/Derivative' */
```

```
1443
1444 /* Update for Derivative: '<S58>/Derivative' */
1445 if (rtDW.TimeStampA_o == (rtInf)) {
1446 rtDW.TimeStampA_o = rtM->Timing.t[0];
1447 lastU = &rtDW.LastUAtTimeA_f;
1448 }elseif (rtDW.TimeStampB_i == (rtInf)) {
1449 rtDW.TimeStampB_i = rtM->Timing.t[0];
1450 lastU = &rtDW.LastUAtTimeB_g;
1451 }elseif (rtDW.TimeStampA_o < rtDW.TimeStampB_i) {
1452 rtDW.TimeStampA_o = rtM->Timing.t[0];
1453 lastU = &rtDW.LastUAtTimeA_f;
1454 }else{
1455 rtDW.TimeStampB_i = rtM->Timing.t[0];
1456 lastU = &rtDW.LastUAtTimeB_g;
1457 }
1458
1459 *lastU = rtDW.LookUpTable_b;
1460
1461 /* End of Update for Derivative: '<S58>/Derivative' */
1462
1463 /* Update for Derivative: '<S60>/Derivative' */
1464 if (rtDW.TimeStampA_fe == (rtInf)) {
1465 rtDW.TimeStampA_fe = rtM->Timing.t[0];
1466 lastU = &rtDW.LastUAtTimeA_c;
1467 }elseif (rtDW.TimeStampB_n == (rtInf)) {
1468 rtDW.TimeStampB_n = rtM->Timing.t[0];
1469 lastU = &rtDW.LastUAtTimeB_o;
1470 }elseif (rtDW.TimeStampA_fe < rtDW.TimeStampB_n) {
1471 rtDW.TimeStampA_fe = rtM->Timing.t[0];
1472 lastU = &rtDW.LastUAtTimeA_c;
1473 }else{
1474 rtDW.TimeStampB_n = rtM->Timing.t[0];
1475 lastU = &rtDW.LastUAtTimeB_o;
1476 }
1477
1478 *lastU = rtDW.LookUpTable_co;
1479
1480 /* End of Update for Derivative: '<S60>/Derivative' */
```

```
1481
1482 /* Update for Derivative: '<S62>/Derivative' */
1483 if (rtDW.TimeStampA_b == (rtInf)) {
1484 rtDW.TimeStampA_b = rtM->Timing.t[0];
1485 lastU = &rtDW.LastUAtTimeA_h;
1486 }elseif (rtDW.TimeStampB_p == (rtInf)) {
1487 rtDW.TimeStampB_p = rtM->Timing.t[0];
1488 lastU = &rtDW.LastUAtTimeB_n;
1489 }elseif (rtDW.TimeStampA_b < rtDW.TimeStampB_p) {
1490 rtDW.TimeStampA_b = rtM->Timing.t[0];
1491 lastU = &rtDW.LastUAtTimeA_h;
1492 }else{
1493 rtDW.TimeStampB_p = rtM->Timing.t[0];
1494 lastU = &rtDW.LastUAtTimeB_n;
1495 }
1496
1497 *lastU = rtDW.LookUpTable_i;
1498
1499 /* End of Update for Derivative: '<S62>/Derivative' */
1500
1501 /* Update for Derivative: '<S69>/Derivative' */
1502 if (rtDW.TimeStampA_e == (rtInf)) {
1503 rtDW.TimeStampA_e = rtM->Timing.t[0];
1504 lastU = &rtDW.LastUAtTimeA_b;
1505 }elseif (rtDW.TimeStampB_a == (rtInf)) {
1506 rtDW.TimeStampB_a = rtM->Timing.t[0];
1507 lastU = &rtDW.LastUAtTimeB_a;
1508 }elseif (rtDW.TimeStampA_e < rtDW.TimeStampB_a) {
1509 rtDW.TimeStampA_e = rtM->Timing.t[0];
1510 lastU = &rtDW.LastUAtTimeA_b;
1511 }else{
1512 rtDW.TimeStampB_a = rtM->Timing.t[0];
1513 lastU = &rtDW.LastUAtTimeB_a;
1514 }
1515
1516 *lastU = rtDW.LookUpTable_ko;
1517
1518 /* End of Update for Derivative: '<S69>/Derivative' */
```

```
1519
1520 /* Update for Derivative: '<S71>/Derivative' */
1521 if (rtDW.TimeStampA_j == (rtInf)) {
1522 rtDW.TimeStampA_j = rtM->Timing.t[0];
1523 lastU = &rtDW.LastUAtTimeA_g;
1524 }elseif (rtDW.TimeStampB_pn == (rtInf)) {
1525 rtDW.TimeStampB_pn = rtM->Timing.t[0];
1526 lastU = &rtDW.LastUAtTimeB_m;
1527 }elseif (rtDW.TimeStampA_j < rtDW.TimeStampB_pn) {
1528 rtDW.TimeStampA_j = rtM->Timing.t[0];
1529 lastU = &rtDW.LastUAtTimeA_g;
1530 }else{
1531 rtDW.TimeStampB_pn = rtM->Timing.t[0];
1532 lastU = &rtDW.LastUAtTimeB_m;
1533 }
1534
1535 *lastU = rtDW.LookUpTable_o;
1536
1537 /* End of Update for Derivative: '<S71>/Derivative' */
1538
1539 /* Update for Derivative: '<S73>/Derivative' */
1540 if (rtDW.TimeStampA_g == (rtInf)) {
1541 rtDW.TimeStampA_g = rtM->Timing.t[0];
1542 lastU = &rtDW.LastUAtTimeA_i;
1543 }elseif (rtDW.TimeStampB_j == (rtInf)) {
1544 rtDW.TimeStampB_j = rtM->Timing.t[0];
1545 lastU = &rtDW.LastUAtTimeB_lc;
1546 }elseif (rtDW.TimeStampA_g < rtDW.TimeStampB_j) {
1547 rtDW.TimeStampA_g = rtM->Timing.t[0];
1548 lastU = &rtDW.LastUAtTimeA_i;
1549 }else{
1550 rtDW.TimeStampB_j = rtM->Timing.t[0];
1551 lastU = &rtDW.LastUAtTimeB_lc;
1552 }
1553
1554 *lastU = rtDW.LookUpTable_a;
1555
1556 /* End of Update for Derivative: '<S73>/Derivative' */
```

```
1557 }
1558
1559 /* Update absolute time for base rate */
1560 /* The "clockTick0" counts the number of times the code of this task has
1561 * been executed. The absolute time is the multiplication of "clockTick0"
1562 * and "Timing.stepSize0". Size of "clockTick0" ensures timer will not
1563 * overflow during the application lifespan selected.
1564 */
1565 rtM->Timing.t[0] =
1566 (++rtM->Timing.clockTick0) * rtM->Timing.stepSize0;
1567
1568 {
1569 /* Update absolute timer for sample time: [5.0E-5s, 0.0s] */
1570 /* The "clockTick1" counts the number of times the code of this task has
1571 * been executed. The resolution of this integer timer is 5.0E-5, which is the step size
1572 * of the task. Size of "clockTick1" ensures timer will not overflow during the
1573 * application lifespan selected.
1574 */
1575 rtM->Timing.clockTick1++;
1576 }
1577 }
1578
1579 /* Model initialize function */
1580 void NEWPDFDEVELOPMENTDESERTATIONamusingnow_initialize(void)
1581 {
1582 /* Registration code */
1583
1584 /* initialize non-finites */
1585 rt_InitInfAndNaN(sizeof(real_T));
1586
1587 {
1588 /* Setup solver object */
1589 rtsiSetSimTimeStepPtr(&rtM->solverInfo, &rtM->Timing.simTimeStep);
1590 rtsiSetTPtr(&rtM->solverInfo, &rtmGetTPtr(rtM));
1591 rtsiSetStepSizePtr(&rtM->solverInfo, &rtM->Timing.stepSize0);
1592 rtsiSetErrorStatusPtr(&rtM->solverInfo, ((const char_T **)
1593 (&rtmGetErrorStatus(rtM))));
1594 rtsiSetRTModelPtr(&rtM->solverInfo, rtM);
```

```
1595 }
1596
1597 rtsiSetSimTimeStep(&rtM->solverInfo, MAJOR_TIME_STEP);
1598 rtsiSetSolverName(&rtM->solverInfo,"FixedStepDiscrete");
1599 rtmSetTPtr(rtM, &rtM->Timing.tArray[0]);
1600 rtM->Timing.stepSize0 = 5.0E-5;
1601
1602 /* Start for S-Function (sfun_spssw_discc): '<S76>/State-Space' incorporates:
1603 * Constant: '<S78>/SwitchCurrents'
1604 */
1605
1606 /* S-Function block: <S76>/State-Space */
1607 {
1608 rtDW.StateSpace_PWORK.AS = (real_T*)calloc(20 * 20, sizeof(real_T));
1609 rtDW.StateSpace_PWORK.BS = (real_T*)calloc(20 * 12, sizeof(real_T));
1610 rtDW.StateSpace_PWORK.CS = (real_T*)calloc(21 * 20, sizeof(real_T));
1611 rtDW.StateSpace_PWORK.DS = (real_T*)calloc(21 * 12, sizeof(real_T));
1612 rtDW.StateSpace_PWORK.DX_COL = (real_T*)calloc(21, sizeof(real_T));
1613 rtDW.StateSpace_PWORK.TMP2 = (real_T*)calloc(12, sizeof(real_T));
1614 rtDW.StateSpace_PWORK.BD_COL = (real_T*)calloc(20, sizeof(real_T));
1615 rtDW.StateSpace_PWORK.TMP1 = (real_T*)calloc(20, sizeof(real_T));
1616 rtDW.StateSpace_PWORK.XTMP = (real_T*)calloc(20, sizeof(real_T));
1617 rtDW.StateSpace_PWORK.SWITCH_STATUS = (int_T*)calloc(9, sizeof(int_T));
1618 rtDW.StateSpace_PWORK.SW_CHG = (int_T*)calloc(9, sizeof(int_T));
1619 rtDW.StateSpace_PWORK.G_STATE = (int_T*)calloc(9, sizeof(int_T));
1620 rtDW.StateSpace_PWORK.Y_SWITCH = (real_T*)calloc(9, sizeof(real_T));
1621 rtDW.StateSpace_PWORK.SWITCH_TYPES = (int_T*)calloc(9, sizeof(int_T));
1622 rtDW.StateSpace_PWORK.IDX_OUT_SW = (int_T*)calloc(9, sizeof(int_T));
1623 rtDW.StateSpace_PWORK.SWITCH_STATUS_INIT = (int_T*)calloc(9, sizeof(int_T));
1624 rtDW.StateSpace_PWORK.USWLAST = (real_T*)calloc(9, sizeof(real_T));
1625 }
1626
1627 /* InitializeConditions for S-Function (sfun_spssw_discc): '<S76>/State-Space'
     incorporates:
1628 * Constant: '<S78>/SwitchCurrents'
1629 */
1630 {
1631 int32_T i, j;
```

```
1632 real_T *As = (real_T*)rtDW.StateSpace_PWORK.AS;
1633 real_T *Bs = (real_T*)rtDW.StateSpace_PWORK.BS;
1634 real_T *Cs = (real_T*)rtDW.StateSpace_PWORK.CS;
1635 real_T *Ds = (real_T*)rtDW.StateSpace_PWORK.DS;
1636 real_T *X0 = (real_T*)&rtDW.StateSpace_DSTATE[0];
1637 for (i = 0; i < 20; i++) {
1638 X0[i] = (rtConstP.StateSpace_X0_param[i]);
1639 }
1640
1641 /* Copy and transpose A and B */
1642 for (i=0; i<20; i++) {
1643 for (j=0; j<20; j++)
1644 As[i*20 + j] = (rtConstP.StateSpace_AS_param[i + j*20]);
1645 for (j=0; j<12; j++)
1646 Bs[i*12 + j] = (rtConstP.StateSpace_BS_param[i + j*20]);
1647 }
1648
1649 /* Copy and transpose C */
1650 for (i=0; i<21; i++) {
1651 for (j=0; j<20; j++)
1652 Cs[i*20 + j] = (rtConstP.StateSpace_CS_param[i + j*21]);
1653 }
1654
1655 /* Copy and transpose D */
1656 for (i=0; i<21; i++) {
1657 for (j=0; j<12; j++)
1658 Ds[i*12 + j] = (rtConstP.StateSpace_DS_param[i + j*21]);
1659 }
1660
1661 {
1662 /* Switches work vectors */
1663 int_T *switch_status = (int_T*) rtDW.StateSpace_PWORK.SWITCH_STATUS;
1664 int_T *gState = (int_T*)rtDW.StateSpace_PWORK.G_STATE;
1665 real_T *yswitch = (real_T*)rtDW.StateSpace_PWORK.Y_SWITCH;
1666 int_T *switchTypes = (int_T*)rtDW.StateSpace_PWORK.SWITCH_TYPES;
1667 int_T *idxOutSw = (int_T*)rtDW.StateSpace_PWORK.IDX_OUT_SW;
1668 int_T *switch_status_init = (int_T*)
1669 rtDW.StateSpace_PWORK.SWITCH_STATUS_INIT;
```

```
1670
1671 /* Initialize work vectors */
1672 switch_status[0] = 0;
1673 switch_status_init[0] = 0;
1674 gState[0] = (int_T) 1.0;
1675 yswitch[0] = 1/0.001;
1676 switchTypes[0] = (int_T)2.0;
1677 idxOutSw[0] = ((int_T)0.0) - 1;
1678 switch_status[1] = 0;
1679 switch_status_init[1] = 0;
1680 gState[1] = (int_T) 1.0;
1681 yswitch[1] = 1/0.001;
1682 switchTypes[1] = (int_T)2.0;
1683 idxOutSw[1] = ((int_T)0.0) - 1;
1684 switch_status[2] = 0;
1685 switch_status_init[2] = 0;
1686 gState[2] = (int_T) 1.0;
1687 yswitch[2] = 1/0.001;
1688 switchTypes[2] = (int_T)2.0;
1689 idxOutSw[2] = ((int_T)0.0) - 1;
1690 switch_status[3] = 0;
1691 switch_status_init[3] = 0;
1692 gState[3] = (int_T) 1.0;
1693 yswitch[3] = 1/0.001;
1694 switchTypes[3] = (int_T)2.0;
1695 idxOutSw[3] = ((int_T)0.0) - 1;
1696 switch_status[4] = 0;
1697 switch_status_init[4] = 0;
1698 gState[4] = (int_T) 1.0;
1699 yswitch[4] = 1/0.001;
1700 switchTypes[4] = (int_T)2.0;
1701 idxOutSw[4] = ((int_T)0.0) - 1;
1702 switch_status[5] = 0;
1703 switch_status_init[5] = 0;
1704 gState[5] = (int_T) 1.0;
1705 yswitch[5] = 1/0.001;
1706 switchTypes[5] = (int_T)2.0;
1707 idxOutSw[5] = ((int_T)0.0) - 1;
```

```
1708  switch_status[6] = 0;
1709  switch_status_init[6] = 0;
1710  gState[6] = (int_T) 0.0;
1711  yswitch[6] = 1/0.001;
1712  switchTypes[6] = (int_T)2.0;
1713  idxOutSw[6] = ((int_T)0.0) - 1;
1714  switch_status[7] = 0;
1715  switch_status_init[7] = 0;
1716  gState[7] = (int_T) 0.0;
1717  yswitch[7] = 1/0.001;
1718  switchTypes[7] = (int_T)2.0;
1719  idxOutSw[7] = ((int_T)0.0) - 1;
1720  switch_status[8] = 0;
1721  switch_status_init[8] = 0;
1722  gState[8] = (int_T) 0.0;
1723  yswitch[8] = 1/0.001;
1724  switchTypes[8] = (int_T)2.0;
1725  idxOutSw[8] = ((int_T)0.0) - 1;
1726  }
1727  }
1728
1729  /* InitializeConditions for Derivative: '<S47>/Derivative' */
1730  rtDW.TimeStampA = (rtInf);
1731  rtDW.TimeStampB = (rtInf);
1732
1733  /* InitializeConditions for Derivative: '<S49>/Derivative' */
1734  rtDW.TimeStampA_l = (rtInf);
1735  rtDW.TimeStampB_k = (rtInf);
1736
1737  /* InitializeConditions for Derivative: '<S51>/Derivative' */
1738  rtDW.TimeStampA_f = (rtInf);
1739  rtDW.TimeStampB_e = (rtInf);
1740
1741  /* InitializeConditions for Derivative: '<S58>/Derivative' */
1742  rtDW.TimeStampA_o = (rtInf);
1743  rtDW.TimeStampB_i = (rtInf);
1744
1745  /* InitializeConditions for Derivative: '<S60>/Derivative' */
```

```
1746 rtDW.TimeStampA_fe = (rtInf);
1747 rtDW.TimeStampB_n = (rtInf);
1748
1749 /* InitializeConditions for Derivative: '<S62>/Derivative' */
1750 rtDW.TimeStampA_b = (rtInf);
1751 rtDW.TimeStampB_p = (rtInf);
1752
1753 /* InitializeConditions for Derivative: '<S69>/Derivative' */
1754 rtDW.TimeStampA_e = (rtInf);
1755 rtDW.TimeStampB_a = (rtInf);
1756
1757 /* InitializeConditions for Derivative: '<S71>/Derivative' */
1758 rtDW.TimeStampA_j = (rtInf);
1759 rtDW.TimeStampB_pn = (rtInf);
1760
1761 /* InitializeConditions for Derivative: '<S73>/Derivative' */
1762 rtDW.TimeStampA_g = (rtInf);
1763 rtDW.TimeStampB_j = (rtInf);
1764
1765 /* Enable for Sin: '<S75>/Sine Wave A' */
1766 rtDW.systemEnable = 1;
1767
1768 /* Enable for Sin: '<S75>/Sine Wave B' */
1769 rtDW.systemEnable_p = 1;
1770
1771 /* Enable for Sin: '<S75>/Sine Wave C' */
1772 rtDW.systemEnable_b = 1;
1773 }
1774
1775 /*
1776 * File trailer for generated code.
1777 *
1778 * [EOF]
1779 */
1780
```

MODEL CODES

File: NEWPDFDEVELOPMENTDESERTATIONamusingnow.h

```
/*
* File: NEWPDFDEVELOPMENTDESERTATIONamusingnow.h
*
* Code generated for Simulink model 'NEWPDFDEVELOPMENTDESERTATIONamusingnow'.
*
* Model version : 1.21
* Simulink Coder version : 8.12 (R2017a) 16-Feb-2017
* C/C++ source code generated on : Sat Jan 16 03:12:04 2016
*
* Target selection: ert.tlc
* Embedded hardware selection: Intel->x86-64 (Windows64)
* Code generation objectives:
* 1. Execution efficiency
* 2. RAM efficiency
* Validation result: Not run
*/

#ifndef RTW_HEADER_NEWPDFDEVELOPMENTDESERTATIONamusingnow_h_
#define RTW_HEADER_NEWPDFDEVELOPMENTDESERTATIONamusingnow_h_
#include <stddef.h>
#include <math.h>
#ifndef NEWPDFDEVELOPMENTDESERTATIONamusingnow_COMMON_INCLUDES_
# define NEWPDFDEVELOPMENTDESERTATIONamusingnow_COMMON_INCLUDES_
#include "rtwtypes.h"
#include "rtw_continuous.h"
#include "rtw_solver.h"
#endif/* NEWPDFDEVELOPMENTDESERTATIONamusingnow_COMMON_INCLUDES_ */

/* Macros for accessing real-time model data structure */
#ifndef rtmGetErrorStatus
# define rtmGetErrorStatus(rtm) ((rtm)->errorStatus)
```

```
32 #endif
33
34 #ifndef rtmSetErrorStatus
35 # define rtmSetErrorStatus(rtm, val) ((rtm)->errorStatus = (val))
36 #endif
37
38 #ifndef rtmGetT
39 # define rtmGetT(rtm) (rtmGetTPtr((rtm))[0])
40 #endif
41
42 /* Forward declaration for rtModel */
43 typedefstruct tag_RTM RT_MODEL;
44
45 /* Block signals and states (auto storage) for system '<Root>' */
46 typedefstruct{
47 real_T StateSpace_o1[21]; /* '<S76>/State-Space' */
48 real_T StateSpace_o2[9]; /* '<S76>/State-Space' */
49 real_T StateSpace_DSTATE[20]; /* '<S76>/State-Space' */
50 real_T SineWaveA; /* '<S75>/Sine Wave A' */
51 real_T SineWaveB; /* '<S75>/Sine Wave B' */
52 real_T SineWaveC; /* '<S75>/Sine Wave C' */
53 real_T LookUpTable_k; /* '<S47>/Look-Up Table' */
54 real_T Switch3; /* '<S46>/Switch3' */
55 real_T LookUpTable_c; /* '<S49>/Look-Up Table' */
56 real_T Switch3_e; /* '<S48>/Switch3' */
57 real_T LookUpTable_f; /* '<S51>/Look-Up Table' */
58 real_T Switch3_f; /* '<S50>/Switch3' */
59 real_T LookUpTable_b; /* '<S58>/Look-Up Table' */
60 real_T Switch3_j; /* '<S57>/Switch3' */
61 real_T LookUpTable_co; /* '<S60>/Look-Up Table' */
62 real_T Switch3_o; /* '<S59>/Switch3' */
63 real_T LookUpTable_i; /* '<S62>/Look-Up Table' */
64 real_T Switch3_n; /* '<S61>/Switch3' */
65 real_T LookUpTable_ko; /* '<S69>/Look-Up Table' */
66 real_T Switch3_i; /* '<S68>/Switch3' */
67 real_T LookUpTable_o; /* '<S71>/Look-Up Table' */
68 real_T Switch3_d; /* '<S70>/Switch3' */
69 real_T LookUpTable_a; /* '<S73>/Look-Up Table' */
```

```
70  real_T Switch3_g; /* '<S72>/Switch3' */
71  real_T lastSin; /* '<S75>/Sine Wave A' */
72  real_T lastCos; /* '<S75>/Sine Wave A' */
73  real_T lastSin_b; /* '<S75>/Sine Wave B' */
74  real_T lastCos_h; /* '<S75>/Sine Wave B' */
75  real_T lastSin_e; /* '<S75>/Sine Wave C' */
76  real_T lastCos_p; /* '<S75>/Sine Wave C' */
77  real_T TimeStampA; /* '<S47>/Derivative' */
78  real_T LastUAtTimeA; /* '<S47>/Derivative' */
79  real_T TimeStampB; /* '<S47>/Derivative' */
80  real_T LastUAtTimeB; /* '<S47>/Derivative' */
81  real_T TimeStampA_l; /* '<S49>/Derivative' */
82  real_T LastUAtTimeA_o; /* '<S49>/Derivative' */
83  real_T TimeStampB_k; /* '<S49>/Derivative' */
84  real_T LastUAtTimeB_i; /* '<S49>/Derivative' */
85  real_T TimeStampA_f; /* '<S51>/Derivative' */
86  real_T LastUAtTimeA_l; /* '<S51>/Derivative' */
87  real_T TimeStampB_e; /* '<S51>/Derivative' */
88  real_T LastUAtTimeB_l; /* '<S51>/Derivative' */
89  real_T TimeStampA_o; /* '<S58>/Derivative' */
90  real_T LastUAtTimeA_f; /* '<S58>/Derivative' */
91  real_T TimeStampB_i; /* '<S58>/Derivative' */
92  real_T LastUAtTimeB_g; /* '<S58>/Derivative' */
93  real_T TimeStampA_fe; /* '<S60>/Derivative' */
94  real_T LastUAtTimeA_c; /* '<S60>/Derivative' */
95  real_T TimeStampB_n; /* '<S60>/Derivative' */
96  real_T LastUAtTimeB_o; /* '<S60>/Derivative' */
97  real_T TimeStampA_b; /* '<S62>/Derivative' */
98  real_T LastUAtTimeA_h; /* '<S62>/Derivative' */
99  real_T TimeStampB_p; /* '<S62>/Derivative' */
10  real_T LastUAtTimeB_n; /* '<S62>/Derivative' */
 0
10  real_T TimeStampA_e; /* '<S69>/Derivative' */
 1
10  real_T LastUAtTimeA_b; /* '<S69>/Derivative' */
 2
10  real_T TimeStampB_a; /* '<S69>/Derivative' */
 3
```

```
1044 real_T LastUAtTimeB_a; /* '<S69>/Derivative' */
1045 real_T TimeStampA_j; /* '<S71>/Derivative' */
1046 real_T LastUAtTimeA_g; /* '<S71>/Derivative' */
1047 real_T TimeStampB_pn; /* '<S71>/Derivative' */
1048 real_T LastUAtTimeB_m; /* '<S71>/Derivative' */
1049 real_T TimeStampA_g; /* '<S73>/Derivative' */
1050 real_T LastUAtTimeA_i; /* '<S73>/Derivative' */
1051 real_T TimeStampB_j; /* '<S73>/Derivative' */
1052 real_T LastUAtTimeB_lc; /* '<S73>/Derivative' */
1053 struct{
1054 void *AS;
1055 void *BS;
1056 void *CS;
1057 void *DS;
1058 void *DX_COL;
1059 void *BD_COL;
1060 void *TMP1;
1061 void *TMP2;
1062 void *XTMP;
1063 void *SWITCH_STATUS;
1064 void *SWITCH_STATUS_INIT;
```

```
1255 void *SW_CHG;
1256 void *G_STATE;
1257 void *USWLAST;
1258 void *XKM12;
1259 void *XKP12;
1260 void *XLAST;
1261 void *ULAST;
1262 void *IDX_SW_CHG;
1263 void *Y_SWITCH;
1264 void *SWITCH_TYPES;
1265 void *IDX_OUT_SW;
1266 void *SWITCH_TOPO_SAVED_IDX;
1267 void *SWITCH_MAP;
1268 } StateSpace_PWORK; /* '<S76>/State-Space' */
1269
1270 int_T StateSpace_IWORK[11]; /* '<S76>/State-Space' */
1271 int32_T systemEnable; /* '<S75>/Sine Wave A' */
1272 int32_T systemEnable_p; /* '<S75>/Sine Wave B' */
1273 int32_T systemEnable_b; /* '<S75>/Sine Wave C' */
1274 } DW;
1275
```

```
/* Invariant block signals (auto storage) */

typedefstruct{

const creal_T Gain52; /* '<S1>/Gain52' */

const creal_T MathFunction2; /* '<S1>/Math Function2' */

const creal_T Gain51; /* '<S1>/Gain51' */

const creal_T MathFunction4; /* '<S1>/Math Function4' */

const creal_T Gain55; /* '<S1>/Gain55' */

const creal_T MathFunction1; /* '<S1>/Math Function1' */

const creal_T Gain54; /* '<S1>/Gain54' */

const creal_T MathFunction3; /* '<S1>/Math Function3' */

const creal_T Gain52_c; /* '<S10>/Gain52' */

const creal_T MathFunction2_f; /* '<S10>/Math Function2' */

const creal_T Gain51_n; /* '<S10>/Gain51' */

const creal_T MathFunction4_f; /* '<S10>/Math Function4' */

const creal_T Gain55_k; /* '<S11>/Gain55' */

const creal_T MathFunction1_n; /* '<S11>/Math Function1' */

const creal_T Gain54_m; /* '<S11>/Gain54' */

const creal_T MathFunction3_g; /* '<S11>/Math Function3' */

} ConstB;



/* Constant parameters (auto storage) */

```

```
typedefstruct{

/* Expression: zeros(9,1)

* Referenced by: '<S78>/SwitchCurrents'

*/

real_T SwitchCurrents_Value[9];



/* Expression: S.A

* Referenced by: '<S76>/State-Space'

*/

real_T StateSpace_AS_param[400];



/* Expression: S.B

* Referenced by: '<S76>/State-Space'

*/

real_T StateSpace_BS_param[240];



/* Expression: S.C

* Referenced by: '<S76>/State-Space'

*/

real_T StateSpace_CS_param[420];
```

```
/* Expression: S.D
* Referenced by: '<S76>/State-Space'
*/
real_T StateSpace_DS_param[252];


/* Expression: S.x0
* Referenced by: '<S76>/State-Space'
*/
real_T StateSpace_X0_param[20];


/* Expression: sps.tv
* Referenced by: '<S52>/Look-Up Table'
*/
real_T LookUpTable_XData[6];


/* Pooled Parameter (Expression: sps.opv)
* Referenced by:
* '<S52>/Look-Up Table'
* '<S63>/Look-Up Table'
*/
real_T pooled16[6];
```

```
/* Pooled Parameter (Expression: tv)
 * Referenced by:
 * '<S47>/Look-Up Table'
 * '<S49>/Look-Up Table'
 * '<S51>/Look-Up Table'
 * '<S58>/Look-Up Table'
 * '<S60>/Look-Up Table'
 * '<S62>/Look-Up Table'
 * '<S69>/Look-Up Table'
 * '<S71>/Look-Up Table'
 * '<S73>/Look-Up Table'
 */
real_T pooled17[5];


/* Pooled Parameter (Expression: opv)
 * Referenced by:
 * '<S47>/Look-Up Table'
 * '<S49>/Look-Up Table'
 * '<S51>/Look-Up Table'
 * '<S58>/Look-Up Table'
```

```
230 * '<S60>/Look-Up Table'
231 * '<S62>/Look-Up Table'
232 */
233 real_T pooled18[5];
234
235 /* Expression: sps.tv
236 * Referenced by: '<S63>/Look-Up Table'
237 */
238 real_T LookUpTable_XData_e[6];
239
240 /* Expression: sps.tv
241 * Referenced by: '<S74>/Look-Up Table'
242 */
243 real_T LookUpTable_XData_i[6];
244
245 /* Expression: sps.opv
246 * Referenced by: '<S74>/Look-Up Table'
247 */
248 real_T LookUpTable_YData[6];
249
250 /* Pooled Parameter (Expression: opv)
```

```
251 * Referenced by:
252 * '<S69>/Look-Up Table'
253 * '<S71>/Look-Up Table'
254 * '<S73>/Look-Up Table'
255 */
256 real_T pooled20[5];
257 } ConstP;
258
259 /* External outputs (root outports fed by signals with auto storage) */
260 typedefstruct{
261 real_T Out19; /* '<Root>/Out19' */
262 creal_T Out20; /* '<Root>/Out20' */
263 real_T Out21; /* '<Root>/Out21' */
264 creal_T Out22; /* '<Root>/Out22' */
265 real_T Out23; /* '<Root>/Out23' */
266 real_T Out24; /* '<Root>/Out24' */
267 real_T Out25; /* '<Root>/Out25' */
268 creal_T Out26; /* '<Root>/Out26' */
269 real_T Out27; /* '<Root>/Out27' */
270 creal_T Out28; /* '<Root>/Out28' */
271 real_T Out29; /* '<Root>/Out29' */
```

```
creal_T Out30; /* '<Root>/Out30' */

real_T Out31; /* '<Root>/Out31' */

real_T Out32; /* '<Root>/Out32' */

real_T Out33; /* '<Root>/Out33' */

creal_T Out13; /* '<Root>/Out13' */

real_T Out1; /* '<Root>/Out1' */

creal_T Out2; /* '<Root>/Out2' */

real_T Out8; /* '<Root>/Out8' */

real_T Out10; /* '<Root>/Out10' */

real_T Out11; /* '<Root>/Out11' */

real_T Out3; /* '<Root>/Out3' */

creal_T Out4; /* '<Root>/Out4' */

real_T Out5; /* '<Root>/Out5' */

creal_T Out6; /* '<Root>/Out6' */

real_T Out7; /* '<Root>/Out7' */

real_T Out12; /* '<Root>/Out12' */

real_T Out14; /* '<Root>/Out14' */

real_T Out15; /* '<Root>/Out15' */

real_T Out9; /* '<Root>/Out9' */

} ExtY;
```

```
/* Real-time Model Data Structure */

struct tag_RTM {

const char_T * volatile errorStatus;

RTWSolverInfo solverInfo;



/*

* Timing:

* The following substructure contains information regarding

* the timing information for the model.

*/

struct{

uint32_T clockTick0;

time_T stepSize0;

uint32_T clockTick1;

SimTimeStep simTimeStep;

time_T *t;

time_T tArray[2];

} Timing;

};



/* Block signals and states (auto storage) */
```

```
extern DW rtDW;


/* External outputs (root outports fed by signals with auto storage) */
extern ExtY rtY;
externconst ConstB rtConstB; /* constant block i/o */


/* Constant parameters (auto storage) */
externconst ConstP rtConstP;


/* Model entry point functions */
externvoid NEWPDFDEVELOPMENTDESERTATIONamusingnow_initialize(void);
externvoid NEWPDFDEVELOPMENTDESERTATIONamusingnow_step(void);


/* Real-time Model object */
extern RT_MODEL *const rtM;


/*-
* These blocks were eliminated from the model due to optimizations:
*
* Block '<Root>/Current Scope' : Unused code path elimination
* Block '<S1>/Complex to Magnitude-Angle' : Unused code path elimination
```

```
335 * Block '<S1>/Complex to Magnitude-Angle3' : Unused code path elimination
336 * Block '<S1>/Complex to Magnitude-Angle4' : Unused code path elimination
337 * Block '<S1>/Display00' : Unused code path elimination
338 * Block '<S1>/Display1' : Unused code path elimination
339 * Block '<S1>/Display10' : Unused code path elimination
340 * Block '<S1>/Display11' : Unused code path elimination
341 * Block '<S1>/Display111' : Unused code path elimination
342 * Block '<S1>/Display12' : Unused code path elimination
343 * Block '<S1>/Display13' : Unused code path elimination
344 * Block '<S1>/Display14' : Unused code path elimination
345 * Block '<S1>/Display15' : Unused code path elimination
346 * Block '<S1>/Display17' : Unused code path elimination
347 * Block '<S1>/Display18' : Unused code path elimination
348 * Block '<S1>/Display19' : Unused code path elimination
349 * Block '<S1>/Display20' : Unused code path elimination
350 * Block '<S1>/Display21' : Unused code path elimination
351 * Block '<S1>/Display22' : Unused code path elimination
352 * Block '<S1>/Display23' : Unused code path elimination
353 * Block '<S1>/Display24' : Unused code path elimination
354 * Block '<S1>/Display25' : Unused code path elimination
355 * Block '<S1>/Display26' : Unused code path elimination
```

```
356 * Block '<S1>/Display27' : Unused code path elimination
357 * Block '<S1>/Display28' : Unused code path elimination
358 * Block '<S1>/Display29' : Unused code path elimination
359 * Block '<S1>/Display30' : Unused code path elimination
360 * Block '<S1>/Display31' : Unused code path elimination
361 * Block '<S1>/Display33' : Unused code path elimination
362 * Block '<S1>/Display333' : Unused code path elimination
363 * Block '<S1>/Display4' : Unused code path elimination
364 * Block '<S1>/Display5' : Unused code path elimination
365 * Block '<S1>/Display7' : Unused code path elimination
366 * Block '<S1>/Display8' : Unused code path elimination
367 * Block '<S1>/Display9' : Unused code path elimination
368 * Block '<S1>/Displayi1' : Unused code path elimination
369 * Block '<S1>/Displayi11' : Unused code path elimination
370 * Block '<S1>/Displayi2' : Unused code path elimination
371 * Block '<S1>/Displayi3' : Unused code path elimination
372 * Block '<S1>/Displayi33' : Unused code path elimination
373 * Block '<S1>/Divide4' : Unused code path elimination
374 * Block '<S1>/Divide5' : Unused code path elimination
375 * Block '<S1>/Divide6' : Unused code path elimination
376 * Block '<S1>/Divide7' : Unused code path elimination
```

```
37 * Block '<S1>/Scope' : Unused code path elimination
7
37 * Block '<S1>/Scope1' : Unused code path elimination
8
37 * Block '<S1>/Z0(t)' : Unused code path elimination
9
38 * Block '<S1>/Z1(t)' : Unused code path elimination
0
38 * Block '<S1>/Z2(t)' : Unused code path elimination
1
38 * Block '<S1>/Z3(t)' : Unused code path elimination
2
38 * Block '<S1>/Zn(0)1' : Unused code path elimination
3
38 * Block '<S1>/Zn(1)1' : Unused code path elimination
4
38 * Block '<S1>/Zn(2)1' : Unused code path elimination
5
38 * Block '<S1>/Zn(3)1' : Unused code path elimination
6
38 * Block '<Root>/Display' : Unused code path elimination
7
38 * Block '<Root>/Display1' : Unused code path elimination
8
38 * Block '<Root>/Display2' : Unused code path elimination
9
39 * Block '<Root>/Display3' : Unused code path elimination
0
39 * Block '<Root>/Display4' : Unused code path elimination
1
39 * Block '<Root>/Display5' : Unused code path elimination
2
39 * Block '<S2>/Complex to Magnitude-Angle1' : Unused code path elimination
3
39 * Block '<S2>/Complex to Magnitude-Angle2' : Unused code path elimination
4
39 * Block '<S2>/Complex to Magnitude-Angle3' : Unused code path elimination
5
39 * Block '<S2>/Complex to Magnitude-Angle4' : Unused code path elimination
6
39 * Block '<S2>/Complex to Magnitude-Angle5' : Unused code path elimination
7
```

```
398 * Block '<S2>/Complex to Magnitude-Angle6' : Unused code path elimination
399 * Block '<S2>/Complex to Magnitude-Angle7' : Unused code path elimination
400 * Block '<S2>/Display' : Unused code path elimination
401 * Block '<S2>/Divide10' : Unused code path elimination
402 * Block '<S2>/Divide11' : Unused code path elimination
403 * Block '<S2>/Divide12' : Unused code path elimination
404 * Block '<S2>/Divide5' : Unused code path elimination
405 * Block '<S2>/Divide6' : Unused code path elimination
406 * Block '<S2>/Divide7' : Unused code path elimination
407 * Block '<S2>/Divide8' : Unused code path elimination
408 * Block '<S2>/Divide9' : Unused code path elimination
409 * Block '<S2>/Gain30' : Unused code path elimination
410 * Block '<S2>/Gain4' : Unused code path elimination
411 * Block '<S2>/I0(t)9' : Unused code path elimination
412 * Block '<S2>/I1(t)2' : Unused code path elimination
413 * Block '<S2>/I2(t)3' : Unused code path elimination
414 * Block '<S2>/I3(t)4' : Unused code path elimination
415 * Block '<S2>/Ik(0)13' : Unused code path elimination
416 * Block '<S2>/Ik(1)10' : Unused code path elimination
417 * Block '<S2>/Ik(2)11' : Unused code path elimination
418 * Block '<S2>/Ik(31)2' : Unused code path elimination
```

```
419 * Block '<S2>/Scope' : Unused code path elimination
420 * Block '<S2>/Scope1' : Unused code path elimination
421 * Block '<S10>/Display18' : Unused code path elimination
422 * Block '<S10>/Display19' : Unused code path elimination
423 * Block '<S10>/Display21' : Unused code path elimination
424 * Block '<S10>/Display23' : Unused code path elimination
425 * Block '<S10>/Display33' : Unused code path elimination
426 * Block '<S10>/Va1' : Unused code path elimination
427 * Block '<S10>/Vb1' : Unused code path elimination
428 * Block '<S10>/Vc1' : Unused code path elimination
429 * Block '<S11>/Display24' : Unused code path elimination
430 * Block '<S11>/Display26' : Unused code path elimination
431 * Block '<S11>/Display27' : Unused code path elimination
432 * Block '<S11>/Display29' : Unused code path elimination
433 * Block '<S11>/Ip22' : Unused code path elimination
434 * Block '<S2>/Sum8' : Unused code path elimination
435 * Block '<S2>/V0(t)5' : Unused code path elimination
436 * Block '<S2>/V1(t)6' : Unused code path elimination
437 * Block '<S2>/V2(t)7' : Unused code path elimination
438 * Block '<S2>/V3(t)8' : Unused code path elimination
439 * Block '<S2>/Vk(0)17' : Unused code path elimination
```

```
440 * Block '<S2>/Vk(1)14' : Unused code path elimination
441 * Block '<S2>/Vk(2)15' : Unused code path elimination
442 * Block '<S2>/Vk(3)16' : Unused code path elimination
443 * Block '<S2>/Z0(t)' : Unused code path elimination
444 * Block '<S2>/Z1(t)' : Unused code path elimination
445 * Block '<S2>/Z2(t)' : Unused code path elimination
446 * Block '<S2>/Z3(t)' : Unused code path elimination
447 * Block '<S2>/Zk(0)' : Unused code path elimination
448 * Block '<S2>/Zk(1)' : Unused code path elimination
449 * Block '<S2>/Zk(2)' : Unused code path elimination
450 * Block '<S2>/Zk(3)' : Unused code path elimination
451 * Block '<S4>/Kv' : Unused code path elimination
452 * Block '<S4>/Kv1' : Unused code path elimination
453 * Block '<S30>/do not delete this gain' : Unused code path elimination
454 * Block '<S31>/do not delete this gain' : Unused code path elimination
455 * Block '<S32>/do not delete this gain' : Unused code path elimination
456 * Block '<S33>/do not delete this gain' : Unused code path elimination
457 * Block '<S34>/do not delete this gain' : Unused code path elimination
458 * Block '<S35>/do not delete this gain' : Unused code path elimination
459 * Block '<Root>/To Workspace3' : Unused code path elimination
460 * Block '<Root>/To Workspace6' : Unused code path elimination
```

```
461 * Block '<Root>/Voltage Scope' : Unused code path elimination
462 * Block '<S1>/Gain1' : Eliminated nontunable gain of 1
463 * Block '<S1>/Gain12' : Eliminated nontunable gain of 1
464 * Block '<S1>/Gain13' : Eliminated nontunable gain of 1
465 * Block '<S1>/Gain14' : Eliminated nontunable gain of 1
466 * Block '<S1>/Gain15' : Eliminated nontunable gain of 1
467 * Block '<S1>/Gain16' : Eliminated nontunable gain of 1
468 * Block '<S1>/Gain17' : Eliminated nontunable gain of 1
469 * Block '<S1>/Gain21' : Eliminated nontunable gain of 1
470 * Block '<S1>/Gain22' : Eliminated nontunable gain of 1
471 * Block '<S1>/Gain23' : Eliminated nontunable gain of 1
472 * Block '<S1>/Gain25' : Eliminated nontunable gain of 1
473 * Block '<S1>/Gain27' : Eliminated nontunable gain of 1
474 * Block '<S1>/Gain28' : Eliminated nontunable gain of 1
475 * Block '<S1>/Gain29' : Eliminated nontunable gain of 1
476 * Block '<S1>/Gain3' : Eliminated nontunable gain of 1
477 * Block '<S1>/Gain33' : Eliminated nontunable gain of 1
478 * Block '<S1>/Gain34' : Eliminated nontunable gain of 1
479 * Block '<S1>/Gain35' : Eliminated nontunable gain of 1
480 * Block '<S1>/Gain36' : Eliminated nontunable gain of 1
481 * Block '<S1>/Gain37' : Eliminated nontunable gain of 1
```

```
482 * Block '<S1>/Gain39' : Eliminated nontunable gain of 1
483 * Block '<S1>/Gain41' : Eliminated nontunable gain of 1
484 * Block '<S1>/Gain42' : Eliminated nontunable gain of 1
485 * Block '<S1>/Gain43' : Eliminated nontunable gain of 1
486 * Block '<S1>/Gain47' : Eliminated nontunable gain of 1
487 * Block '<S1>/Gain48' : Eliminated nontunable gain of 1
488 * Block '<S1>/Gain49' : Eliminated nontunable gain of 1
489 * Block '<S1>/Gain5' : Eliminated nontunable gain of 1
490 * Block '<S1>/Gain6' : Eliminated nontunable gain of 1
491 * Block '<S1>/Gain7' : Eliminated nontunable gain of 1
492 * Block '<S1>/Gain8' : Eliminated nontunable gain of 1
493 * Block '<S1>/Gain9' : Eliminated nontunable gain of 1
494 * Block '<S2>/Gain11' : Eliminated nontunable gain of 1
495 * Block '<S2>/Gain14' : Eliminated nontunable gain of 1
496 * Block '<S2>/Gain15' : Eliminated nontunable gain of 1
497 * Block '<S2>/Gain17' : Eliminated nontunable gain of 1
498 * Block '<S2>/Gain19' : Eliminated nontunable gain of 1
499 * Block '<S2>/Gain21' : Eliminated nontunable gain of 1
500 * Block '<S2>/Gain22' : Eliminated nontunable gain of 1
501 * Block '<S2>/Gain27' : Eliminated nontunable gain of 1
502 * Block '<S2>/Gain28' : Eliminated nontunable gain of 1
```

```
503 * Block '<S2>/Gain29' : Eliminated nontunable gain of 1
504 * Block '<S2>/Gain9' : Eliminated nontunable gain of 1
505 * Block '<S15>/do not delete this gain' : Eliminated nontunable gain of 1
506 * Block '<S16>/do not delete this gain' : Eliminated nontunable gain of 1
507 * Block '<S17>/do not delete this gain' : Eliminated nontunable gain of 1
508 * Block '<S18>/do not delete this gain' : Eliminated nontunable gain of 1
509 * Block '<S19>/do not delete this gain' : Eliminated nontunable gain of 1
510 * Block '<S20>/do not delete this gain' : Eliminated nontunable gain of 1
511 * Block '<S46>/Data Type Conversion' : Eliminate redundant data type conversion
512 * Block '<S48>/Data Type Conversion' : Eliminate redundant data type conversion
513 * Block '<S50>/Data Type Conversion' : Eliminate redundant data type conversion
514 * Block '<S5>/Switch' : Eliminated due to constant selection input
515 * Block '<S5>/Switch1' : Eliminated due to constant selection input
516 * Block '<S5>/Switch2' : Eliminated due to constant selection input
517 * Block '<S5>/Switch3' : Eliminated due to constant selection input
518 * Block '<S57>/Data Type Conversion' : Eliminate redundant data type conversion
519 * Block '<S59>/Data Type Conversion' : Eliminate redundant data type conversion
520 * Block '<S61>/Data Type Conversion' : Eliminate redundant data type conversion
521 * Block '<S6>/Switch' : Eliminated due to constant selection input
522 * Block '<S6>/Switch1' : Eliminated due to constant selection input
523 * Block '<S6>/Switch2' : Eliminated due to constant selection input
```

```
524 * Block '<S6>/Switch3' : Eliminated due to constant selection input
525 * Block '<S68>/Data Type Conversion' : Eliminate redundant data type conversion
526 * Block '<S70>/Data Type Conversion' : Eliminate redundant data type conversion
527 * Block '<S72>/Data Type Conversion' : Eliminate redundant data type conversion
528 * Block '<S7>/Switch3' : Eliminated due to constant selection input
529 * Block '<S46>/C4' : Unused code path elimination
530 * Block '<S48>/C4' : Unused code path elimination
531 * Block '<S50>/C4' : Unused code path elimination
532 * Block '<S5>/C4' : Unused code path elimination
533 * Block '<S5>/Constant1' : Unused code path elimination
534 * Block '<S5>/Constant2' : Unused code path elimination
535 * Block '<S5>/Constant3' : Unused code path elimination
536 * Block '<S5>/Constant5' : Unused code path elimination
537 * Block '<S5>/com' : Unused code path elimination
538 * Block '<S57>/C4' : Unused code path elimination
539 * Block '<S59>/C4' : Unused code path elimination
540 * Block '<S61>/C4' : Unused code path elimination
541 * Block '<S6>/C4' : Unused code path elimination
542 * Block '<S6>/Constant1' : Unused code path elimination
543 * Block '<S6>/Constant2' : Unused code path elimination
544 * Block '<S6>/Constant3' : Unused code path elimination
```

```
* Block '<S6>/Constant5' : Unused code path elimination

* Block '<S6>/com' : Unused code path elimination

* Block '<S7>/C4' : Unused code path elimination

* Block '<S68>/C4' : Unused code path elimination

* Block '<S70>/C4' : Unused code path elimination

* Block '<S72>/C4' : Unused code path elimination

* Block '<S7>/com' : Unused code path elimination

*/



/*-

* The generated code includes comments that allow you to trace directly

* back to the appropriate location in the model. The basic format

* is <system>/block_name, where system is the system number (uniquely

* assigned by Simulink) and block_name is the name of the block.

*

* Use the MATLAB hilite_system command to trace the generated code back

* to the model. For example,

*

* hilite_system('<S3>') - opens system 3

* hilite_system('<S3>/Kp') - opens and selects block Kp which resides in S3

*

```

```
* Here is the system hierarchy for this model

*

* '<Root>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow'

* '<S1>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/DFT ALGORITHM FOR VOLTAGE AND CURRENT COMPUTATION'
* '<S2>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/FFT MODEL FOR VOLTAGE AND CURRENT COMPUTATION'
* '<S3>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I Measurement1'
* '<S4>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I Measurement2'
* '<S5>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker1'
* '<S6>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker2'
* '<S7>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Fault1'

* '<S8>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Source2'

* '<S9>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/powergui1'

* '<S10>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/FFT MODEL FOR VOLTAGE AND CURRENT COMPUTATION/Subsystem'
* '<S11>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/FFT MODEL FOR VOLTAGE AND CURRENT COMPUTATION/Subsystem1'
* '<S12>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I Measurement1/Mode I'
* '<S13>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I Measurement1/Mode V'
* '<S14>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I Measurement1/Model'
* '<S15>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I Measurement1/Model/I A:'
* '<S16>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I Measurement1/Model/I B:'
* '<S17>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I Measurement1/Model/I C:'
* '<S18>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I Measurement1/Model/U AB:'
```

```
58 * '<S19>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
7 Measurement1/Model/U BC:'
58 * '<S20>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
8 Measurement1/Model/U CA:'
58 * '<S21>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
9 Measurement1/Model/I A:/Model'
59 * '<S22>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
0 Measurement1/Model/I B:/Model'
59 * '<S23>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
1 Measurement1/Model/I C:/Model'
59 * '<S24>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
2 Measurement1/Model/U AB:/Model'
59 * '<S25>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
3 Measurement1/Model/U BC:/Model'
59 * '<S26>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
4 Measurement1/Model/U CA:/Model'
59 * '<S27>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
5 Measurement2/Mode I'
59 * '<S28>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
6 Measurement2/Mode V'
59 * '<S29>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
7 Measurement2/Model'
59 * '<S30>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
8 Measurement2/Model/I A:'
59 * '<S31>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
9 Measurement2/Model/I B:'
60 * '<S32>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
0 Measurement2/Model/I C:'
60 * '<S33>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
1 Measurement2/Model/U AB:'
60 * '<S34>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
2 Measurement2/Model/U BC:'
60 * '<S35>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
3 Measurement2/Model/U CA:'
60 * '<S36>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
4 Measurement2/Model/I A:/Model'
60 * '<S37>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
5 Measurement2/Model/I B:/Model'
60 * '<S38>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
6 Measurement2/Model/I C:/Model'
60 * '<S39>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I
7 Measurement2/Model/U AB:/Model'
```

```
608 * '<S40>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I Measurement2/Model/U BC:/Model'
609 * '<S41>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase V-I Measurement2/Model/U CA:/Model'
610 * '<S42>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker1/Breaker A'
611 * '<S43>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker1/Breaker B'
612 * '<S44>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker1/Breaker C'
613 * '<S45>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker1/Stair Generator'
614 * '<S46>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker1/Breaker A/Model'
615 * '<S47>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker1/Breaker A/Model/Timer'
616 * '<S48>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker1/Breaker B/Model'
617 * '<S49>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker1/Breaker B/Model/Timer'
618 * '<S50>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker1/Breaker C/Model'
619 * '<S51>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker1/Breaker C/Model/Timer'
620 * '<S52>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker1/Stair Generator/Model'
621 * '<S53>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker2/Breaker A'
622 * '<S54>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker2/Breaker B'
623 * '<S55>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker2/Breaker C'
624 * '<S56>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker2/Stair Generator'
625 * '<S57>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker2/Breaker A/Model'
626 * '<S58>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker2/Breaker A/Model/Timer'
627 * '<S59>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker2/Breaker B/Model'
628 * '<S60>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker2/Breaker B/Model/Timer'
```

```
6269 * '<S61>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker2/Breaker C/Model'
6270 * '<S62>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker2/Breaker C/Model/Timer'
6271 * '<S63>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Breaker2/Stair Generator/Model'
6272 * '<S64>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Fault1/Fault A'
6273 * '<S65>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Fault1/Fault B'
6274 * '<S66>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Fault1/Fault C'
6275 * '<S67>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Fault1/Stair Generator'
6276 * '<S68>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Fault1/Fault A/Model'
6277 * '<S69>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Fault1/Fault A/Model/Timer'
6278 * '<S70>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Fault1/Fault B/Model'
6279 * '<S71>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Fault1/Fault B/Model/Timer'
6280 * '<S72>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Fault1/Fault C/Model'
6281 * '<S73>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Fault1/Fault C/Model/Timer'
6282 * '<S74>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Fault1/Stair Generator/Model'
6283 * '<S75>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/Three-Phase Source2/Model'
6284 * '<S76>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/powergui1/EquivalentModel1'
6285 * '<S77>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/powergui1/EquivalentModel1/Gates'
6286 * '<S78>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/powergui1/EquivalentModel1/Sources'
6287 * '<S79>' : 'NEWPDFDEVELOPMENTDESERTATIONamusingnow/powergui1/EquivalentModel1/Status'
```

```
* '<S80>' :
'NEWPDFDEVELOPMENTDESERTATIONamusingnow/powergui1/EquivalentModel1/Yout'
*/

#endif/* RTW_HEADER_NEWPDFDEVELOPMENTDESERTATIONamusingnow_h_ */


/*

* File trailer for generated code.

*

* [EOF]

*/
```

MATLAB CODES FOR DATA UTILIZATION IN MODELING

File: NEWPDFDEVELOPMENTDESERTATIONamusingnow_data.c

```
/*
* File: NEWPDFDEVELOPMENTDESERTATIONamusingnow_data.c
*
* Code generated for Simulink model 'NEWPDFDEVELOPMENTDESERTATIONamusingnow'.
*
* Model version : 1.21
* Simulink Coder version : 8.12 (R2017a) 16-Feb-2017
* C/C++ source code generated on : Sat Jan 16 03:12:04 2016
*
* Target selection: ert.tlc
* Embedded hardware selection: Intel->x86-64 (Windows64)
* Code generation objectives:
* 1. Execution efficiency
```

```
14 * 2. RAM efficiency
15 * Validation result: Not run
16 */
17
18 #include "NEWPDFDEVELOPMENTDESERTATIONamusingnow.h"
19
20 /* Invariant block signals (auto storage) */
21 constConstBrtConstB = {
22 { -0.0, -2.095 }, /* '<S1>/Gain52' */
23
24 { -0.50052376518695874, -0.86572279655965523 },/* '<S1>/Math Function2' */
25
26 { 0.0, 2.095 }, /* '<S1>/Gain51' */
27
28 { -0.50052376518695874, 0.86572279655965523 },/* '<S1>/Math Function4' */
29
30 { -0.0, -2.095 }, /* '<S1>/Gain55' */
31
32 { -0.50052376518695874, -0.86572279655965523 },/* '<S1>/Math Function1' */
33
34 { 0.0, 2.095 }, /* '<S1>/Gain54' */
35
36 { -0.50052376518695874, 0.86572279655965523 },/* '<S1>/Math Function3' */
37
38 { -0.0, -2.095 }, /* '<S10>/Gain52' */
39
40 { -0.50052376518695874, -0.86572279655965523 },/* '<S10>/Math Function2' */
41
42 { 0.0, 2.095 }, /* '<S10>/Gain51' */
43
44 { -0.50052376518695874, 0.86572279655965523 },/* '<S10>/Math Function4' */
45
46 { -0.0, -2.095 }, /* '<S11>/Gain55' */
47
48 { -0.50052376518695874, -0.86572279655965523 },/* '<S11>/Math Function1' */
49
50 { 0.0, 2.095 }, /* '<S11>/Gain54' */
51
```

```
{ -0.50052376518695874, 0.86572279655965523 }/* '<S11>/Math Function3' */
};

/* Constant parameters (auto storage) */
const ConstP rtConstP = {
/* Expression: zeros(9,1)
* Referenced by: '<S78>/SwitchCurrents'
*/
{ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 },

/* Expression: S.A
* Referenced by: '<S76>/State-Space'
*/
{ 0.64237758561285707, 1.15115388188201, -0.00086591040427357394,
-0.000606922629712472, -0.00086591040427291, -0.00060692262971269353,
0.015228656985299217, -0.0034633085720741732, -0.0034633085720738089,
-0.59005826106470238, 0.13420672642093948, 0.13420672642091794,
-0.06020005222876889, -63.066015699288137, -0.10067151859663398,
-0.10067151859664111, -11.841317512184588, 4.2013537395016461E-6,
-4.7816108000289695E-7, -4.7816108000284263E-7, -0.00014454144732907963,
0.999898689893455, 7.6206558216213789E-8, 5.3413707106016556E-8,
7.6206558216155357E-8, 5.3413707106036051E-8, -1.3402351206052749E-6,
3.0479692242512682E-7, 3.0479692242509479E-7, 5.1929517188915267E-5,
-1.1811190464427737E-5, -1.1811190464425841E-5, 5.2980525030640216E-6,
0.0055502786121207556, 8.85984266361259E-6, 8.85984266361322E-6,
0.0010421240441220853, -3.6975038844213448E-10, 4.2081732705980936E-11,
4.208173270597616E-11, -0.00086591040427315121, -0.00060692262971252233,
0.64237758561285674, 1.15115388188201, -0.00086591040427291008,
-0.00060692262971269375, -0.0034633085720736337, 0.015228656985298452,
-0.0034633085720738115, 0.13420672642092726, -0.590058261064679,
0.13420672642091802, -0.060200052228767113, -0.10067151859662808,
-63.066015699288151, -0.10067151859664122, -11.84131751218459,
-4.7816108000299192E-7, 4.2013537395014834E-6, -4.7816108000284274E-7,
7.6206558216176573E-8, 5.3413707106020983E-8, -0.00014454144732907 96,
0.999898689893455, 7.6206558216155357E-8, 5.3413707106036065E-8,
3.0479692242507939E-7, -1.3402351206052076E-6, 3.0479692242509506E-7,
-1.181119046442666E-5, 5.1929517188913194E-5, -1.1811190464425848E-5,
5.2980525030638658E-6, 8.8598426636120733E-6, 0.0055502786121207574,
```

8.8598426636132287E-6, 0.0010421240441220855, 4.2081732705989285E-11,
-3.6975038844212021E-10, 4.2081732705976167E-11, -0.00086591040427315131,
-0.000606922629712522, -0.00086591040427315392, -0.00060692262971252374,
0.64237758561285729, 1.1511538818820097, -0.0034633085720734472,
-0.0034633085720738002, 0.015228656985298829, 0.13420672642092726,
0.13420672642092518, -0.59005826106469372, -0.0602000522287702,
-0.10067151859662811, -0.10067151859663126, -63.066015699288144,
-11.841317512184586, -4.7816108000289239E-7, -4.7816108000292066E-7,
4.2013537395016749E-6, 7.6206558216176586E-8, 5.341370710602095E-8,
7.6206558216176824E-8, 5.34137071060211E-8, -0.00014454144732907966,
0.999898689893455, 3.0479692242506303E-7, 3.0479692242509405E-7,
-1.3402351206052406E-6, -1.181119046442666E-5, -1.1811190464426478E-5,
5.1929517188914495E-5, 5.2980525030641377E-6, 8.8598426636120733E-6,
8.8598426636123529E-6, 0.0055502786121207565, 0.0010421240441220851,
4.2081732705980535E-11, 4.2081732705983017E-11, -3.6975038844213712E-10,
0.00652825545614506, 0.0045756996905523591, 0.0010523760793856613,
0.00073761771933353358, 0.0010523760793856622, 0.00073761771933326839,
0.96124430957131468, 0.00731741065346795, 0.0073174106534674389,
-75.990125356912728, -0.28054646597769645, -0.280546465977672,
-14.32756637555582, 75.762059137670249, 0.24282397379502219,
0.24282397379501267, 14.270760265703899, 0.00058354743668168941,
9.438379834467419E-5, 9.4383798344674081E-5, 0.0010523760793857925,
0.00073761771933373, 0.0065282554561451748, 0.0045756996905523374,
0.0010523760793856619, 0.00073761771933326839, 0.0073174106534672038,
0.96124430957131557, 0.0073174106534674389, -0.28054646597768534,
-75.990125356912742, -0.280546465977672, -14.327566375555826,
0.2428239737949921, 75.762059137670292, 0.24282397379501267,
14.270760265703899, 9.4383798344674149E-5, 0.00058354743668168985,
9.4383798344674068E-5, 0.0010523760793857925, 0.00073761771933372961,
0.0010523760793853983, 0.00073761771933353293, 0.0065282554561454367,
0.0045756996905520729, 0.0073174106534669818, 0.0073174106534674,
0.961244309571315, -0.28054646597768534, -0.28054646597768523,
-75.990125356912728, -14.32756637555582, 0.24282397379499204,
0.24282397379501097, 75.762059137670263, 14.270760265703895,
9.4383798344674068E-5, 9.4383798344674122E-5, 0.00058354743668168974,
1.339819192972152E-6, 9.390885984540307E-7, -3.0473727723088761E-7,
-2.135924788011842E-7, -3.0473727723092488E-7, -2.1359247880120813E-7,
0.000446931018207002, -0.00013988671513345 33, -0.00013988671513345344,

0.982124040588294, 0.005439892341307922, 0.0054398923413079255,
-0.0013094260293516785, 0.017270491384083913, -0.0054100021528103934,
-0.0054100021528103942, 0.0012072934152739059, -1.2798149163193929E-5,
-2.6687163748402834E-8, -2.6687163748403046E-8, -3.0473727723530171E-7,
-2.1359247880414982E-7, 1.33981919297049E-6, 9.390885984566834E-7,
-3.0473727723092456E-7, -2.1359247880120638E-7, -0.00013988671513345,
0.00044693101820699745, -0.00013988671513345336, 0.0054398923413079272,
0.98212404058829406, 0.0054398923413079246, -0.0013094260293516312,
-0.0054100021528100386, 0.017270491384083726, -0.0054100021528103925,
0.0012072934152739091, -2.668716374840348E-8, -1.2798149163193927E-5,
-2.6687163748402858E-8, -3.0473727723530018E-7, -2.1359247880414739E-7,
-3.0473727723411121E-7, -2.1359247880036377E-7, 1.3398191929705924E-6,
9.390885984566312E-7, -0.00013988671513345005, -0.00013988671513345211,
0.00044693101820699794, 0.0054398923413079264, 0.005439892341308001,
0.98212404058829406, -0.0013094260293516405, -0.0054100021528100386,
-0.0054100021528102919, 0.017270491384083646, 0.0012072934152739211,
-2.6687163748403797E-8, -2.6687163748401471E-8, -1.2798149163193931E-5,
7.3034463850458277E-7, 5.1190364085539718E-7, 7.3034463852539194E-7,
5.1190364084878768E-7, 7.3034463846988968E-7, 5.1190364088567088E-7,
0.00016715758794008735, 0.000167157587940099, 0.00016715758794010415,
-0.00699617472909043, -0.0069961747290906645, -0.0069961747290905605,
0.99607172191194482, 0.0064504870784626456, 0.0064504870784629683,
0.0064504870784642242, 0.0036218802458221179, -1.285152349069073E-5,
-1.2851523490690742E-5, -1.285152349069074E-5, 0.00014320121220847131,
0.00010037072642006279, 2.2859036421542044E-7, 1.6022057742607972E-7,
2.2859036420508384E-7, 1.602205774295422E-7, -0.000445726544662831,
0.00013961585258272057, 0.0001396158525827149, 0.017270491384083833,
-0.0054100021528106137, -0.0054100021528102789, 0.0012072934152738835,
0.97717398419582957, 0.0054024552471467436, 0.0054024552471468555,
-0.0022499078144632319, -1.1938595582314855E-7, 2.7153925923767381E-8,
2.7153925923766336E-8, 2.2859036420881483E-7, 1.6022057742684925E-7,
0.00014320121220847 47, 0.00010037072642006278, 2.285903642050855E-7,
1.6022057742954551E-7, 0.00013961585258271287, -0.00044572654466281986,
0.00013961585258271493, -0.0054100021528107421, 0.017270491384083785,
-0.00541000215281028, 0.0012072934152738564, 0.0054024552471468867,
0.97717398419582957, 0.0054024552471468555, -0.0022499078144632041,
2.7153925923769059E-8, -1.1938595582314625E-7, 2.7153925923766336E-8,
2.2859036420881647E-7, 1.6022057742684427E-7, 2.2859036420890194E-7,

```
1.6022057742688726E-7, 0.00014320121220846778, 0.00010037072642006623,
0.00013961585258271, 0.00013961585258271479, -0.00044572654466282571,
-0.0054100021528107421, -0.0054100021528103908, 0.017270491384084014,
0.0012072934152739039, 0.0054024552471468867, 0.0054024552471467011,
0.97717398419582946, -0.0022499078144632588, 2.7153925923767517E-8,
2.7153925923767752E-8, -1.1938595582314943E-7, 0.00014365839293689221,
0.00010069116757491873, 0.00014365839293689224, 0.00010069116757491876,
0.00014365839293689218, 0.00010069116757491879, -0.00016649483949739602,
-0.00016649483949739926, -0.00016649483949739236, 0.0064504870784647551,
0.006450487078463055, 0.0064504870784630481, 0.0036218802458219124,
-0.012021105309878339, -0.012021105309877097, -0.01202110530987709,
0.99325027655660969, -6.5078103975613553E-8, -6.50781039756125E-8,
-6.5078103975611356E-8, 5.6794096476354126E-5, 3.9807377547959862E-5,
-6.4638038576709176E-6, -4.5305251165945817E-6, -6.4638038580750387E-6,
-4.5305251162186295E-6, 0.018394617949760279, -0.0041773389045506332,
-0.0041773389045505109, 76.192077157753516, 0.15887847637278169,
0.15887847637278057, 14.319820782791478, 0.71074839350904562,
-0.16165728284130271, -0.16165728284129435, 0.07251333170652507 5,
0.984266093324909, -9.354765026765006E-5, -9.3547650267649952E-5,
-6.4638038580813085E-6, -4.5305251167603071E-6, 5.6794096476311395E-5,
3.9807377547895522E-5, -6.4638038579662187E-6, -4.5305251161095977E-6,
-0.0041773389045504832, 0.018394617949760536, -0.0041773389045507226,
0.15887847637278243, 76.192077157753516, 0.15887847637278882,
14.319820782791481, -0.16165728284128383, 0.71074839350904662,
-0.1616572828413026, 0.07251333170652613, -9.3547650267649952E-5,
0.98426609332490922, -9.354765026765E-5, -6.4638038580812424E-6,
-4.5305251167602088E-6, -6.4638038576864319E-6, -4.530525116562729E-6,
5.6794096475912124E-5, 3.9807377548379347E-5, -0.0041773389045504875,
-0.0041773389045504788, 0.018394617949760377, 0.15887847637278241,
0.15887847637278485, 76.192077157753516, 14.319820782791481,
-0.16165728284128378, -0.16165728284129888, 0.71074839350904462,
0.07251333170652671 2, -9.3547650267649952E-5, -9.3547650267649979E-5,
0.98426609332490911 },

/* Expression: S.B
* Referenced by: '<S76>/State-Space'
*/
{ 2890.8289465815924, 2026.2021309018958, -1.5241311643242756,
```

-1.068274142120331, -1.524131164323107, -1.0682741421207209,
26.8047024121055, -6.0959384485025367, -6.0959384485018964,
-1038.5903437783054, 236.22380928855475, 236.22380928851683,
-105.96105006128043, -111005.57224241512, -177.19685327225181,
-177.19685327226441, -20842.480882441705, 0.00739500776884269,
-0.00084163465411961878, -0.00084163465411952315, -1.5241311643235316,
-1.0682741421204196, 2890.8289465815924, 2026.2021309018958,
-1.5241311643231072, -1.0682741421207214, -6.0959384485015873,
26.804702412104152, -6.0959384485019008, 236.2238092885332,
-1038.5903437782638, 236.22380928851697, -105.96105006127731,
-177.19685327224144, -111005.57224241515, -177.19685327226455,
-20842.480882441709, -0.00084163465411978575, 0.0073950077688424041,
-0.00084163465411952337, -1.5241311643235316, -1.0682741421204192,
-1.5241311643235365, -1.068274142120422, 2890.8289465815928,
2026.2021309018955, -6.09593844850126, -6.09593844850188, 26.804702412104813,
236.2238092885332, 236.22380928852959, -1038.59034377829,
-105.96105006128275, -177.19685327224147, -177.19685327224704,
-111005.57224241513, -20842.4808824417, -0.00084163465411961076,
-0.00084163465411966031, 0.0073950077688427424, 0.0073950077688465883,
0.0051832124197701678, -0.00084163465414790583, -0.00058990758745320769,
-0.0008416346541480664, -0.00058990758742471431, 2.3951141242316552,
-0.543920153129486, -0.543920153129463, 9920.7670772779675,
20.687142502018137, 20.687142502017629, 1864.5456571593379,
92.544651958410242, -21.048963478100497, -21.0489634781007,
9.4417674473988917, 258.36599375626133, -0.012180589945205001,
-0.012180589945204975, -0.00084163465417099348, -0.00058990758747890252,
0.0073950077688129875, 0.005183212419739085, -0.00084163465411952023,
-0.00058990758739612 64, -0.54392015312945885, 2.3951141242317466,
-0.543920153129519, 20.687142502017839, 9920.76707727797, 20.687142502019796,
1864.5456571593384, -21.048963478098557, 92.544651958411791,
-21.04896347810287, 9.4417674473991369, -0.012180589945204988,
258.36599375626139, -0.012180589945204994, -0.00084163465417099164,
-0.00058990758747890 2, -0.00084163465411959536, -0.00058990758745319468,
0.0073950077688130778, 0.005183212419796167, -0.54392015312945963,
-0.54392015312945818, 2.39511412423169, 20.687142502017835,
20.687142502018158, 9920.7670772779711, 1864.5456571593384,
-21.048963478098553, -21.048963478100518, 92.544651958409588,
9.4417674473991653, -0.012180589945204984, -0.012180589945204993,

258.36599375626133, -0.923544924998389, -0.64731906646360371,
0.35126922807103017, 0.24620704704356056, 0.35126922807633859,
0.24620704704055907, -320.11740429072205, 134.76728060022586,
134.76728060022529, -1.1500031313869129E+6, 382267.10316409433,
382267.10316409427, -72145.574334982783, -12371.507359719295,
5209.776520322609, 5209.77652032245, -365.33389923409345, 7.8963681509973,
-2.0037117837786464, -2.0037117837786473, 0.35126922807785993,
0.2462070470440586, -0.92354492499573493, -0.64731906646355364,
0.35126922807633709, 0.2462070470405554, 134.76728060022535,
-320.11740429071966, 134.76728060022523, 382267.10316409421,
-1.1500031313869131E+6, 382267.10316409427, -72145.574334982855,
5209.7765203223889, -12371.50735971934, 5209.7765203224481,
-365.33389923409828, -2.0037117837786478, 7.8963681509973,
-2.0037117837786473, 0.35126922807785749, 0.24620704704405472,
0.35126922807596606, 0.24620704704228846, -0.92354492499058216,
-0.64731906646651249, 134.76728060022543, 134.7672806002239,
-320.11740429072125, 382267.10316409421, 382267.10316409415,
-1.1500031313869131E+6, -72145.574334982841, 5209.776520322388,
5209.7765203224508, -12371.507359719384, -365.33389923411733,
-2.0037117837786473, -2.0037117837786482, 7.8963681509973016,
7.395007768846588E-6, 5.1832124197701681E-6, -8.4163465414790587E-7,
-5.8990758745320779E-7, -8.4163465414806649E-7, -5.8990758742471434E-7,
0.0023951141242316552, -0.000543920153129486, -0.000543920153129463,
9.9207670772779668, 0.020687142502018138, 0.020687142502017631,
1.864545657159338, 0.092544651958410246, -0.021048963478100497,
-0.0210489634781007, 0.0094417674473988925, 0.25836599375626135,
-1.2180589945205E-5, -1.2180589945204977E-5, -8.4163465417099355E-7,
-5.8990758747890253E-7, 7.3950077688129879E-6, 5.1832124197390854E-6,
-8.4163465411952021E-7, -5.8990758739612645E-7, -0.00054392015312945891,
0.0023951141242317467, -0.000543920153129519, 0.020687142502017839,
9.9207670772779686, 0.0206871425020198, 1.8645456571593384,
-0.021048963478098558, 0.092544651958411786, -0.021048963478102874,
0.0094417674473991371, -1.2180589945204989E-5, 0.2583659937562614,
-1.2180589945204994E-5, -8.4163465417099165E-7, -5.89907587478902E-7,
-8.4163465411959549E-7, -5.8990758745319476E-7, 7.3950077688130777E-6,
5.183212419796167E-6, -0.00054392015312945956, -0.00054392015312945826,
0.00239511412423169, 0.020687142502017836, 0.020687142502018158,
9.92076707727797, 1.8645456571593384, -0.021048963478098554,

```
-0.021048963478100518, 0.0925446519584096, 0.0094417674473991648,
-1.2180589945204986E-5, -1.2180589945204992E-5, 0.25836599375626135 },

/* Expression: S.C
* Referenced by: '<S76>/State-Space'
*/
{ 0.041059439640321431, -2.1647760106839349E-5, -2.1647760106822751E-5,
1.0503384348754117E-7, -1.1954027000072427E-8, -1.1954027000071069E-8,
-1.3117426801654587E-5, 4.989197885486464E-6, 4.9891978854859253E-6,
-1.8106624687141048E-5, 5.3845816694320094E-19, 1.8106624687140512E-5,
-0.0015741336045172879, 1.7902346272080649E-19, 0.0015741336045172877,
1.0503384348754116E-10, -1.1954027000072426E-11, -1.1954027000071069E-11,
4.105943964032143E-5, -2.1647760106838651E-8, -2.164776010682349E-8,
-3.6135361832269911E-6, 1.905163955405345E-9, 1.9051639554038842E-9,
-9.2437597110533633E-12, 1.0520433176495236E-12, 1.052043317649404E-12,
1.1544311562416463E-9, -4.3908653509192887E-10, -4.3908653509188152E-10,
1.5935176913335749E-9, -4.7391493398878104E-23, -1.593517691333528E-9,
1.385354692364286E-7, -1.5754812818929988E-23, -1.3853546923642857E-7,
-9.243759711053363E-15, 1.0520433176495235E-15, 1.052043317649404E-15,
-3.6135361832269911E-9, 1.9051639554052834E-12, 1.9051639554039489E-12,
-2.1647760106828778E-5, 0.041059439640321424, -2.1647760106822754E-5,
-1.1954027000074798E-8, 1.0503384348753709E-7, -1.1954027000071072E-8,
4.9891978854860981E-6, -1.3117426801654059E-5, 4.9891978854858677E-6,
1.8106624687140157E-5, -1.8106624687139926E-5, -2.3037127760972E-19,
0.0015741336045172884, -0.0015741336045172879, -3.2820968165481191E-19,
-1.1954027000074798E-11, 1.0503384348753709E-10, -1.1954027000071071E-11,
-2.1647760106829231E-8, 4.1059439640321423E-5, -2.1647760106823493E-8,
1.9051639554044144E-9, -3.6135361832269907E-6, 1.9051639554038842E-9,
1.0520433176497322E-12, -9.2437597110530047E-12, 1.0520433176494042E-12,
-4.3908653509189661E-10, 1.1544311562415997E-9, -4.390865350918764E-10,
-1.5935176913334964E-9, 1.5935176913334761E-9, 2.0286439086740495E-23,
-1.3853546923642862E-7, 1.385354692364286E-7, 2.888382350137164E-23,
1.0520433176497321E-15, -9.2437597110530064E-15, 1.0520433176494042E-15,
1.9051639554044542E-12, -3.6135361832269902E-9, 1.9051639554039489E-12,
-2.1647760106828781E-5, -2.1647760106828849E-5, 0.041059439640321431,
-1.1954027000072313E-8, -1.1954027000073018E-8, 1.0503384348754189E-7,
4.9891978854861261E-6, 4.9891978854860744E-6, -1.31174268016544E-5,
5.1347814888913492E-20, 1.8106624687140475E-5, -1.8106624687140526E-5,
```

7.9103390504542406E-20, 0.0015741336045172879, -0.0015741336045172881,
-1.1954027000072313E-11, -1.1954027000073018E-11, 1.0503384348754189E-10,
-2.1647760106828612E-8, -2.1647760106828149E-8, 4.1059439640321436E-5,
1.9051639554044144E-9, 1.9051639554044206E-9, -3.613536183226992E-6,
1.0520433176495135E-12, 1.0520433176495755E-12, -9.24375971105343E-12,
-4.390865350918991E-10, -4.390865350918946E-10, 1.1544311562416297E-9,
-4.5316262678105073E-24, -1.5935176913335245E-9, 1.593517691333529E-9,
-6.9880218148479781E-24, -1.385354692364286E-7, 1.385354692364286E-7,
1.0520433176495134E-15, 1.0520433176495755E-15, -9.2437597110534277E-15,
1.9051639554043997E-12, 1.9051639554043593E-12, -3.6135361832269915E-9,
0.00016320638640362277, 2.6309401984641536E-5, 2.6309401984641556E-5,
1.4588685917042238E-5, 2.3595949586168547E-6, 2.3595949586168521E-6,
-0.0015108553094646167, 0.00038188416280875934, 0.00038188416280875989,
-0.0018927394722733761, -6.1201044232461755E-19, 0.0018927394722733763,
0.0018879808790968806, 2.3869795029440866E-19, -0.0018879808790968811,
1.4588685917042238E-8, 2.3595949586168546E-9, 2.3595949586168517E-9,
1.6320638640366627E-7, 2.630940198481711E-8, 2.6309401984466061E-8,
2.6309401984644815E-5, 0.00016320638640362608, 2.6309401984641556E-5,
2.3595949586168542E-6, 1.4588685917042247E-5, 2.3595949586168521E-6,
0.00038188416280875978, -0.0015108553094646171, 0.00038188416280875994,
0.0018927394722733769, -0.0018927394722733769, 3.3445468616832841E-19,
-0.0018879808790968821, 0.0018879808790968817, 5.1417203827952562E-19,
2.3595949586168542E-9, 1.4588685917042247E-8, 2.3595949586168517E-9,
2.630940198460151E-8, 1.6320638640382331E-7, 2.6309401984466061E-8,
2.6309401984644815E-5, 2.6309401984634956E-5, 0.00016320638640363264,
2.3595949586168517E-6, 2.359594958616853E-6, 1.4588685917042242E-5,
0.00038188416280875972, 0.00038188416280875972, -0.0015108553094646169,
-2.7755575615628915E-21, 0.0018927394722733767, -0.0018927394722733767,
-4.73232564246473E-19, -0.0018879808790968813, 0.0018879808790968817,
2.3595949586168513E-9, 2.359594958616853E-9, 1.4588685917042242E-8,
2.6309401984666589E-8, 2.6309401984810532E-8, 1.6320638640345713E-7,
3.3495479824349392E-8, -7.6184319307721754E-9, -7.6184319307731085E-9,
-3.1995372907984829E-7, -6.6717909371007415E-10, -6.671790937100847E-10,
3.7088643194080727E-5, -1.2328460512093937E-5, -1.2328460512093937E-5,
4.9417103706174654E-5, -1.0842021724855045E-22, -4.9417103706174654E-5,
5.6701233842235954E-7, 0.0, -5.6701233842235954E-7, -3.1995372907984831E-10,
-6.6717909371007417E-13, -6.6717909371008467E-13, 3.3495479824360878E-11,
-7.6184319307588837E-12, -7.618431930787953E-12, -7.6184319309931948E-9,

3.3495479824264556E-8, -7.6184319307730588E-9, -6.67179093710107E-10,
-3.1995372907984819E-7, -6.6717909371007529E-10, -1.2328460512093937E-5,
3.7088643194080727E-5, -1.2328460512093937E-5, -4.9417103706174661E-5,
4.9417103706174661E-5, -1.7347234759768072E-22, -5.6701233842233233E-7,
5.6701233842235011E-7, -1.7759231585312563E-20, -6.6717909371010709E-13,
-3.199537290798482E-10, -6.6717909371007528E-13, -7.6184319310077762E-12,
3.3495479824284364E-11, -7.6184319307879045E-12, -7.6184319309931187E-9,
-7.618431930931363E-9, 3.3495479824268639E-8, -6.6717909371012223E-10,
-6.6717909371000611E-10, -3.1995372907984829E-7, -1.2328460512093939E-5,
-1.2328460512093936E-5, 3.7088643194080727E-5, -3.7513395167998452E-21,
-4.9417103706174654E-5, 4.9417103706174661E-5, 1.2750217548429533E-20,
-5.6701233842234143E-7, 5.6701233842232873E-7, -6.6717909371012223E-13,
-6.6717909371000611E-13, -3.1995372907984831E-10, -7.6184319309881581E-12,
-7.6184319309180691E-12, 3.34954798242538E-11, 1.8258615962514704E-8,
1.8258615963556052E-8, 1.8258615960779815E-8, -3.2128808726726815E-7,
-3.2128808726726884E-7, -3.2128808726726868E-7, 1.2431722169892841E-5,
1.2431722169892831E-5, 1.2431722169892836E-5, 1.1644331332494319E-20,
-5.030698080332741E-21, -6.6136332521615777E-21, -1.5959455978986625E-20,
-6.29704621779581E-20, 7.8929918156944721E-20, -3.2128808726726818E-10,
-3.212880872672688E-10, -3.212880872672687E-10, 1.8258615962516559E-11,
1.825861596360245E-11, 1.8258615960735277E-11, 3.5800303052116991E-6,
5.7147591055513906E-9, 5.7147591050321594E-9, -2.9846488955787578E-9,
6.7884814809416824E-10, 6.7884814809411561E-10, 3.9899232420716495E-7,
-1.6802001421520194E-7, -1.6802001421518506E-7, 5.6701233842236684E-7,
-1.6870185803874451E-20, -5.6701233842235E-7, 4.9294288223717075E-5,
-5.5727991665754929E-21, -4.9294288223717069E-5, -2.9846488955787576E-12,
6.7884814809416822E-13, 6.7884814809411561E-13, 3.5800303052116868E-9,
5.7147591055381532E-12, 5.7147591050469468E-12, 5.7147591052204555E-9,
3.5800303052118727E-6, 5.7147591050322446E-9, 6.7884814809425271E-10,
-2.9846488955786411E-9, 6.7884814809411561E-10, -1.6802001421520644E-7,
3.9899232420716442E-7, -1.6802001421518326E-7, -5.6701233842237076E-7,
5.6701233842234757E-7, 2.3180242447740087E-20, -4.9294288223717069E-5,
4.9294288223717069E-5, -1.4094628242311559E-21, 6.7884814809425273E-13,
-2.9846488955786413E-12, 6.7884814809411561E-13, 5.714759105235002E-12,
3.5800303052118518E-9, 5.7147591050470317E-12, 5.71475910522054E-9,
5.7147591052225731E-9, 3.5800303052115263E-6, 6.7884814809417475E-10,
6.7884814809418685E-10, -2.9846488955788012E-9, -1.6802001421520729E-7,
-1.6802001421518974E-7, 3.9899232420717511E-7, -1.7564075194265172E-20,

-5.6701233842236483E-7, 5.6701233842238241E-7, 9.1940344226770776E-21,
-4.9294288223717069E-5, 4.9294288223717062E-5, 6.7884814809417478E-13,
6.788481480941868E-13, -2.9846488955788012E-12, 5.7147591052155915E-12,
5.7147591052093352E-12, 3.5800303052115408E-9, 3.5914598234223051E-6,
3.591459823422306E-6, 3.591459823422303E-6, -1.6269525993904089E-9,
-1.6269525993903547E-9, -1.6269525993903005E-9, 6.2952295776846545E-8,
6.2952295776761352E-8, 6.2952295776761339E-8, 8.5196606713910944E-20, 0.0,
-8.5196606713910944E-20, -6.2319940874466792E-20, -1.7347234759768072E-22,
6.2493413222064478E-20, -1.626952599390409E-12, -1.6269525993903547E-12,
-1.6269525993903006E-12, 3.5914598234223034E-9, 3.5914598234222596E-9,
3.5914598234223476E-9, 1.4198524119083073E-6, -1.6159509644760031E-7,
-1.6159509644763115E-7, 0.049606652333122725, -2.3386912566912519E-6,
-2.3386912566912468E-6, 0.0015161143456456653, -0.00038471562138885285,
-0.000384715621388853, 0.0019008299670345181, 9.783840404509193E-20,
-0.0019008299670345186, 2.1810141908758341E-5, 3.8857805861880479E-20,
-2.1810141908758381E-5, 4.9606652333122726E-5, -2.3386912566912516E-9,
-2.3386912566912467E-9, 1.4198524119087621E-9, -1.6159509644675495E-10,
-1.615950964485229E-10, -1.6159509645203318E-7, 1.4198524119018559E-6,
-1.6159509644215024E-7, -2.3386912566912494E-6, 0.049606652333122739,
-2.3386912566912502E-6, -0.00038471562138885312, 0.0015161143456456656,
-0.00038471562138885275, -0.001900829967034519, 0.0019008299670345186,
3.7608804959177181E-19, -2.1810141908758263E-5, 2.1810141908759093E-5,
-8.2850393212652311E-19, -2.3386912566912492E-9, 4.960665233312274E-5,
-2.3386912566912504E-9, -1.6159509645258076E-10, 1.4198524119029519E-9,
-1.6159509644304197E-10, -1.6159509645203281E-7, -1.6159509644216469E-7,
1.4198524119018733E-6, -2.3386912566912485E-6, -2.33869125669125E-6,
0.049606652333122732, -0.00038471562138885312, -0.00038471562138885312,
0.001516114345645666, -6.1756155744774332E-20, -0.001900829967034519,
0.0019008299670345192, 3.7747582837255325E-19, -2.1810141908758219E-5,
2.1810141908757843E-5, -2.3386912566912487E-9, -2.33869125669125E-9,
4.9606652333122733E-5, -1.6159509645182859E-10, -1.6159509644131932E-10,
1.4198524119009814E-9 **}**,

/ Expression: S.D*
** Referenced by: '<S76>/State-Space'*
**/*
{ -927.72927633546021, -0.038103279108106895, -0.038103279108077676,
0.00018487519422106727, -2.1040866352990472E-5, -2.104086635298808E-5,

-0.023088623124832924, 0.0087817307018385783, 0.00878173070183763,
-0.0318703538266715, 9.477664997125637 9E-16, 0.031870353826670549,
-2.7707093847285718, 3.1510793418278535E-16, 2.7707093847285713,
1.8487519422106727E-7, -2.1040866352990469E-8, -2.104086635298808E-8,
0.072270723664539807, -3.8103279108105664E-5, -3.8103279108078979E-5,
-0.038103279108088292, -927.72927633546021, -0.038103279108077683,
-2.1040866352994646E-5, 0.00018487519422106009, -2.1040866352988083E-5,
0.008781730701837933, -0.023088623124831995, 0.0087817307018375271,
0.031870353826669924, -0.031870353826669522, -4.0548772925846791E-16,
2.7707093847285726, -2.7707093847285722, -5.776978793351064E-16,
-2.1040866352994644E-8, 1.8487519422106012E-7, -2.1040866352988087E-8,
-3.8103279108089082E-5, 0.072270723664539807, -3.8103279108078986E-5,
-0.038103279108088292, -0.038103279108088417, -927.72927633546021,
-2.1040866352990268E-5, -2.1040866352991508E-5, 0.00018487519422106857,
0.0087817307018379833, 0.0087817307018378914, -0.023088623124832598,
9.0379795075682614E-17, 0.031870353826670486, -0.031870353826670576,
1.392337383598354E-16, 2.7707093847285722, -2.7707093847285726,
-2.104086635299027E-8, -2.1040866352991508E-8, 1.8487519422106857E-7,
-3.8103279108088E-5, -3.8103279108087178E-5, 0.072270723664539821,
0.00018487519422116474, -2.1040866353697646E-5, -2.1040866353701661E-5,
-993.54085015609348, -0.00030451474863012505, -0.00030451474863012445,
0.19740920377493251, -0.050092794594466195, -0.050092794594466215,
0.24750199836939871, 1.2739277546329412E-17, -0.24750199836939873,
0.0028398403859127685, 5.0595712240740919E-18, -0.0028398403859127737,
0.006459149843906534, -3.0451474863012505E-7, -3.0451474863012442E-7,
1.8487519422122394E-7, -2.1040866353587577E-8, -2.1040866353817772E-8,
-2.1040866354274838E-5, 0.00018487519422032247, -2.1040866352988005E-5,
-0.00030451474863012472, -993.54085015609348, -0.00030451474863012488,
-0.050092794594466229, 0.19740920377493254, -0.050092794594466181,
-0.24750199836939879, 0.24750199836939873, 4.8969421490145681E-17,
-0.0028398403859127585, 0.0028398403859128665, -1.0787728645615119E-16,
-3.0451474863012468E-7, 0.0064591498439065349, -3.0451474863012489E-7,
-2.1040866354346135E-8, 1.8487519422046741E-7, -2.1040866353104117E-8,
-2.104086635427479E-5, -2.1040866352989889E-5, 0.00018487519422032695,
-0.00030451474863012467, -0.00030451474863012483, -993.54085015609348,
-0.050092794594466229, -0.050092794594466229, 0.1974092037749326,
-8.04110426683204E-18, -0.24750199836939879, 0.24750199836939885,
4.9150120462434039E-17, -0.0028398403859127529, 0.0028398403859127034,

470 -3.0451474863012463E-7, -3.0451474863012484E-7, 0.006459149843906534,
471 -2.10408663542482E-8, -2.1040866352879815E-8, 1.8487519422021084E-7,
472 -0.023088623124959726, 0.0087817307017757553, 0.0087817307019084651,
473 0.19740920377493248, -0.050092794594466153, -0.050092794594466188,
474 -250026.79180734264, -249988.48505147887, -249988.48505147887,
475 -38.306755863775187, 1.6376595054256523E-15, 38.306755863775187,
476 -0.43953209700104762, 3.9665609803418143E-15, 0.43953209700104373,
477 0.00019740920377493251, -5.0092794594466167E-5, -5.0092794594466187E-5,
478 -2.3088623124968338E-5, 8.7817307017727613E-6, 8.7817307019129571E-6,
479 0.0087817307019464972, -0.023088623124893373, 0.0087817307019084427,
480 -0.050092794594466208, 0.19740920377493248, -0.0500927945944662,
481 -249988.48505147887, -250026.79180734264, -249988.48505147887,
482 38.306755863775187, -38.306755863775187, 1.6950739999471809E-15,
483 0.43953209700104312, -0.43953209700104473, 1.4799774741388363E-15,
484 -5.0092794594466194E-5, 0.00019740920377493251, -5.0092794594466187E-5,
485 8.7817307019580921E-6, -2.3088623124901419E-5, 8.78173070191292E-6,
486 0.0087817307019464382, 0.0087817307018991531, -0.023088623124764555,
487 -0.050092794594466195, -0.050092794594466208, 0.19740920377493254,
488 -249988.48505147887, -249988.48505147887, -250026.79180734264,
489 3.0340807708200483E-16, 38.306755863775187, -38.306755863775187,
490 -1.5587756581050435E-15, 0.4395320970010459, -0.43953209700104429,
491 -5.0092794594466187E-5, -5.0092794594466207E-5, 0.00019740920377493256,
492 8.7817307019428828E-6, 8.7817307018961571E-6, -2.3088623124760063E-5,
493 1.848751942211647E-7, -2.1040866353697645E-8, -2.1040866353701662E-8,
494 0.006459149843906534, -3.0451474863012505E-7, -3.0451474863012447E-7,
495 0.00019740920377493254, -5.0092794594466194E-5, -5.0092794594466221E-5,
496 0.00024750199836939867, 1.2739277546329413E-20, -0.00024750199836939878,
497 2.8398403859127688E-6, 5.0595712240740929E-21, -2.8398403859127739E-6,
498 6.4591498439065338E-6, -3.0451474863012504E-10, -3.0451474863012442E-10,
499 1.8487519422122394E-10, -2.1040866353587576E-11, -2.1040866353817771E-11,
500 -2.1040866354274839E-8, 1.8487519422032469E-7, -2.1040866352988007E-8,
501 -3.0451474863012473E-7, 0.006459149843906534 9, -3.0451474863012489E-7,
502 -5.0092794594466228E-5, 0.00019740920377493256, -5.0092794594466187E-5,
503 -0.00024750199836939883, 0.00024750199836939878, 4.8969421490145678E-20,
504 -2.8398403859127586E-6, 2.8398403859128666E-6, -1.078772864561512E-19,
505 -3.0451474863012473E-10, 6.4591498439065355E-6, -3.0451474863012488E-10,
506 -2.1040866354346136E-11, 1.8487519422046743E-10, -2.1040866353104115E-11,
507 -2.1040866354274789E-8, -2.1040866352989887E-8, 1.8487519422032696E-7,

```
-3.0451474863012468E-7, -3.0451474863012484E-7, 0.0064591498439065349,
-5.0092794594466228E-5, -5.0092794594466228E-5, 0.00019740920377493262,
-8.04110426683204E-21, -0.00024750199836939883, 0.00024750199836939889,
4.9150120462434042E-20, -2.8398403859127527E-6, 2.8398403859127036E-6,
-3.0451474863012468E-10, -3.0451474863012483E-10, 6.4591498439065338E-6,
-2.10408663542482E-11, -2.1040866352879813E-11, 1.8487519422021085E-10 },

/* Expression: S.x0
* Referenced by: '<S76>/State-Space'
*/
{ -972441.02882035193, -4.2963876941896492E+8, -3.6830838859826922E+6,
2.8997551271650445E+8, 4.6555249148030486E+6, 1.3966325670246047E+8,
420541.09186925023, -4.6232474219646957E+6, 4.2027063300954513E+6,
-1.2548459929951062E+9, -5.3211341930461388E+9, 6.5759801860412455E+9, 0.0,
-1.3885261159953861E+9, -5.2589905315826788E+9, 6.6475166475780659E+9,
5.8207660913467407E-7, 1.8111694714517388E+6, -5.5447231171458485E+6,
3.7335536456941143E+6 },

/* Expression: sps.tv
* Referenced by: '<S52>/Look-Up Table'
*/
{ 0.0, 0.066616666666666685, 0.066616666666666685, 0.16661666666666669,
0.16661666666666669, 1.1666166666666666 },

/* Pooled Parameter (Expression: sps.opv)
* Referenced by:
* '<S52>/Look-Up Table'
* '<S63>/Look-Up Table'
*/
{ 1.0, 1.0, 0.0, 0.0, 1.0, 1.0 },

/* Pooled Parameter (Expression: tv)
* Referenced by:
* '<S47>/Look-Up Table'
* '<S49>/Look-Up Table'
* '<S51>/Look-Up Table'
* '<S58>/Look-Up Table'
* '<S60>/Look-Up Table'
```

```
546  * '<S62>/Look-Up Table'
547  * '<S69>/Look-Up Table'
548  * '<S71>/Look-Up Table'
549  * '<S73>/Look-Up Table'
550  */
551  { -1.0, 0.0, 1.0E+6, 1.0E+6, 1.000001E+6 },
552
553  /* Pooled Parameter (Expression: opv)
554  * Referenced by:
555  * '<S47>/Look-Up Table'
556  * '<S49>/Look-Up Table'
557  * '<S51>/Look-Up Table'
558  * '<S58>/Look-Up Table'
559  * '<S60>/Look-Up Table'
560  * '<S62>/Look-Up Table'
561  */
562  { 0.0, 0.0, 0.0, 1.0, 1.0 },
563
564  /* Expression: sps.tv
565  * Referenced by: '<S63>/Look-Up Table'
566  */
567  { 0.0, 0.019950000000000002, 0.019950000000000002, 0.05995, 0.05995, 1.05995 },
568
569  /* Expression: sps.tv
570  * Referenced by: '<S74>/Look-Up Table'
571  */
572  { 0.0, 0.019950000000000002, 0.019950000000000002, 0.099950000000000025,
573  0.099950000000000025, 1.09995 },
574
575  /* Expression: sps.opv
576  * Referenced by: '<S74>/Look-Up Table'
577  */
578  { 0.0, 0.0, 1.0, 1.0, 0.0, 0.0 },
579
580  /* Pooled Parameter (Expression: opv)
581  * Referenced by:
582  * '<S69>/Look-Up Table'
583  * '<S71>/Look-Up Table'
```

```
584 * '<S73>/Look-Up Table'
585 */
586 { 1.0, 1.0, 1.0, 0.0, 0.0 }
587 };
588
589 /*
590 * File trailer for generated code.
591 *
592 * [EOF]
593 */
594
```