UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

CHARACTERIZATION OF CAVITY PD AND CORONA UNDER IMPULSE VOLTAGE

Tibello Vincent Mphuti

A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Electrical Engineering.

Johannesburg, 2nd March 2025

Declaration

I declare that this dissertation is my own unaided work, except where otherwise acknowledged. It is being submitted for the Master of Science in Engineering degree at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any other university.

Signed this 2nd day of March 2025

Tibello Vincent Mphuti

Abstract

Understanding the phenomenon of Partial Discharge (PD) under impulse voltage is essential to improving the reliability of High Voltage (HV) equipment. PD in HV equipment can cause severe insulation deterioration, resulting in breakdown, production loss, and high repair costs. HV equipment insulation can fail due to PD activities initiated by impulse voltages, such as lightning strikes and switching operations, which are inevitable phenomena. The latter has been exacerbated by load-shedding switching operations in South Africa. PD detection technology is well developed under Alternating Current (AC) frequency voltage conditions. However, there is inadequate knowledge of the detection and interpretation of PD in non-power frequencies. The challenge in PD measurements is to detect PD during the impulse voltage due to the overlaps between the test impulse voltage and the PD frequencies. In the present work, PD characteristics in cavity PD and corona discharges are studied using a developed balanced PD detection circuit to detect PD current pulses effectively. A balanced PD detection circuit effectively separates interferences induced by the impulse voltage from PD current pulses. The experimental results showed a distinctive characteristic of the cavity PD with main and reverse discharges located respectively during the rise-time and fall-time phases of the applied lightning impulse voltage. In contrast, positive and negative corona discharges are observed only in the rise-time part of the applied lightning impulse voltage. MATLAB® Simulink® is also used to simulate cavity PD and corona discharges under lightning impulse voltage, and the characteristics of the PD pulses correlate with the experimental results. The experimental circuit and the simulation model can be developed and used as a standardized PD measurement technique under non-power frequencies.

To: Love (Wife)

Thank you, Love, for believing in me and for the daily Read Me! Notes, my ketones of ambition.

From: Love (Husband)

Acknowledgments

I want to thank my wife (Ofentse) for her support and encouragement, my children (Nyakallo, Omphile, and Oarabile) for their patience, and the support and guidance I received from Professor Cuthbert Nyamupangedengu and Professor Ken Nixon. I wouldn't have been able to do this without them. I also want to thank Kgotso Hlapolosa and Victor Madonsela for allowing me to adopt their simulation model. Lastly, thanks to everyone who contributed to my research work from the School of Electrical and Information Engineering, my colleagues and friends. I love you all.

Contents

Publication

T.V. Mphuti and C. Nyamupangedengu, "Characterization of Cavity PD Under Impulse Voltage in Polyethylene Insulation" *2023 31st Southern African Universities Power Engineering Conference (SAUPEC)*, pp. 1-5, University of Johannesburg, January 2023

List of Figures

List of Tables

List of Abbreviations

AC Alternating Current. ii, 1

asl Above Sea Level. 19

FEMM Finite Element Method Magnetics. 30, 34

HDPE High-Density Polyethylene. 16

HFCT High-Frequency Current transformers. 9

HV High Voltage. ii, 1

kA kiloampere. 7

kV kilovolt. 7

LI Lightning Impulse. 7

OLI Oscillating Lightning Impulse voltage. 7

OSI Oscillating Switching Impulse voltage. 7

PD Partial Discharge. ii, 1

PDIV Partial Discharge Inception Voltage. 48

SI Switching Impulse. 7

TS Technical Specification. 5

UHF Ultra High Frequency. 9

UWB Ultra-Wide Band. 8

Chapter 1

Introduction

Partial Discharge (PD) is a localized electrical discharge that partially bridges weaker insulation material between the conductors and may span adjacent conductors. PD occurs if the dielectric strength of an insulation material is exceeded under regular voltage application and abnormal voltages due to the intensity of the localized ionization, which can lead to insulation breakdown [1]. A PD detection system is essential to analyse and diagnose the dielectric condition of insulation material in High Voltage (HV) power systems. Typical partial discharges are corona (gas discharge) due to a non-uniform field at sharp edges in air, surface discharges due to interfacing of different dielectric materials, cavity discharges due to cavities formed in solid or liquid insulation, and treeing channels due to the high field intensity in insulation material at sharp edges, which continuously deteriorate insulation material [2]. Electrical power plant equipment experiences normal operating voltages and encounters lightning and switching impulses [1]. PD detection under Alternating Current (AC) frequency voltage is well developed; however, more needs to be known about the efficacy of detecting and measuring PD under impulse voltages [1, 3]. A new PD measurement system that can effectively detect and measure PD on incipient defects under impulse voltage will benefit industries. Making informed decisions about the dielectric strength of the insulation material of the HV equipment can help mitigate insulation breakdown, which can cause catastrophic failure [4]. HV equipment insulation can fail due to PD activities

initiated by impulse voltages, such as lightning and switching impulse voltages. The latter has been exacerbated by load-shedding switching operations in South Africa. However, cavities and corona defects in HV equipment are common during manufacturing processes and due to unsafe handling during transportation. As explained in the next section, such insulation defects under the influence of impulse voltage stress become a problem.

1.1 Problem Statement

PD in HV equipment can cause insulation aging, deterioration, and thus breakdown, resulting in loss of production and high breakdown costs. Industries are experiencing HV equipment insulation breakdown due to PD phenomena, which are not detectable under impulse voltages. In normal operations, most HV power equipment is subjected to AC voltages. Transient voltages in the form of lightning and switching impulses, that is, lightning strikes and load shedding switching operations or under regular plant operation, always find their way into the system, thus creating additional stress on the insulation of the equipment [5]. PD behaviour under AC power frequency is adequately known. However, less is known about the behaviour of the PD activity under impulse voltages due to the interference induced by the impulse voltages. The detection of PD behaviour under impulse voltages makes it challenging to separate discharge pulses from impulse voltages due to the noise caused by impulse voltages [6].

1.2 Research Question

The aim and objective of this research is to answer the following questions:

- Where do cavity PD and corona discharges occur relative to impulse voltage?
- How are cavity PD discharges different from corona discharges under impulse?

1.3 The dissertation document structure

The dissertation document is structured as follows:

Chapter 2: Presents the background of the project and literature review in PD detection methods and PD measurement systems.

Chapter 3: Presents proposed experimental setup design and development followed in this investigation.

Chapter 4: Presents the experimental results, analysis, and discussion.

Chapter 5: Presents the experimental design and development of the cavity PD and corona model in MATLAB® Simulink®.

Chapter 6: Presents the conclusions.

Appendix A: This Appendix presents the C Code for cavity PD and corona used for simulation.

Appendix B: This Appendix presents a balanced PD detection circuit sensitivity test.

Appendix C: This Appendix presents experimental cavity PD pulse shapes under impulse voltage shorts.

Appendix D: This Appendix presents experimental corona PD pulse shapes under impulse voltage shorts.

Appendix E: This Appendix comprises a conference paper presented at the 31st Southern African Universities Power Engineering Conference (SAUPEC), 2023.

The flow of this document is as follows: Each chapter begins with a high-level overview of the rest of the chapter and ends with the information that will be addressed in the next chapter.

Chapter 2

Literature Review

This chapter reviews the literature on PD detection and measurements under impulse conditions. It highlights the importance of PD diagnoses under impulse voltage and the challenge of enabling PD detection under impulse conditions. The efficacy and applicability of the PD detection technique are also discussed under standard impulse voltages. Different types of PDs are investigated under impulse voltages, and the results are also shared. Lastly, the proposed PD detection and measurement technique is discussed under impulse conditions.

2.1 Background

PD measurements under impulse voltage can provide helpful information about incipient defects in HV equipment. Therefore, it is essential to diagnose the insulation condition of the HV equipment under impulse voltages [5]. PD measurement techniques under AC voltage have been well established using the conventional technique defined in the standard IEC 60270 [7]. During service, HV equipment is subjected to AC operating voltage and transient voltages, such as lightning and switching impulse voltages, which are superimposed on the AC operating voltage. This phenomenon can result in PD activity, which may continue under AC operating voltage due to insulation degradation or aging caused by the nature of PD

activity initiated by transient voltages [8]. PD detection under impulse voltages concerns the separation of the PD signal from the noise induced by the transient voltages. Under impulse voltage, the frequency content of the PD overlaps that of the supply voltage. The conventional PD detection and measurement technique is incompatible with detecting PD under impulse voltage. The unconventional technique, electromagnetic and acoustic methods, is based on the measurement of electrical signals in the frequency range of kilohertz to GHz and, therefore, is considered a better alternative [5], recommended by IEC Technical Specification (TS) 62478 [9]. Other unconventional techniques include measuring the magnitude of the PD current using a capacitive probe or measuring the PD transient electric field through an inductive probe. The discussion of PD detection and measurement is as follows.

2.2 PD Detection and Measurement

It is essential to detect and measure PD activity, as this will inform the decision to modify, repair, or replace the insulation system before insulation breakdown or plant damage can occur. In addition, it will positively impact the repair cost associated with the severity of the defect or damage. This information can be used to perform a risk analysis for insulation degradation or aging [10]. Non-electric methods can be used to locate the PD source; however, these have limitations based on the type and location of the source of PD. Visual location is possible primarily when the discharge occurs on exposed surfaces or within a transparent dielectric. Acoustic detection methods need to be more accurate in determining the location of PD discharge if additional accessories are used, such as a stethoscope, a listening stick, or a microphone. The drawback is that the additional tools should be placed very close to the discharge site, which is still being determined at times.

The electrical method of PD detection mitigates the difficulties of visual and acoustic detection methods [10]. Amongst other deficiencies, the latter require dark or quiet conditions, which are both challenging to attain, especially under field conditions. In addition, electrical detection methods are more sensitive and provide more information on the nature and intensity of the discharge [10, 11]. Con-

sequently, research in PD activity has diversified, especially regarding discharge behavioural aspects such as nature and form of discharge, detection sensitivity, degradation of insulation exposed to PD, discharge pulse quantities, and pulse repetition rate [12]. The electrical method of PD detection, which uses sensors, can have limitations in providing accurate information due to electromagnetic interference. The sensor's sensitivity is hindered when it is located near any source of electromagnetic fields, which leads to the use of internal sensors. Internal sensors are, in turn, exposed to the influence of electromagnetic fields from standard operating voltages and overvoltages, such as those caused by lightning and switching impulses [4]. As mentioned earlier, in ultra-wideband PD detection, unconventional methods under impulse voltages can be used to measure the frequency from hundreds of kHz to GHz. The circuits comprise the inductive probe to detect the magnetic field of the PD current, a capacitive probe to detect the electric field of the PD transients, an antenna to detect the electromagnetic radiation of the PD [5], and an acoustic sensor to detect the acoustic energy of the PD event.

Electrical methods of PD detection using inductive and capacitive probes connected to sensitive measuring instruments, such as oscilloscopes, can not only detect PDs amplitude, polarity, and distribution but also give an indication of the location of PD using the applied voltage as a reference [10]. PD mechanism, visual phenomena, and waveforms are influenced by factors such as the source of voltage type, voltage magnitude, pressure in gaseous insulation, presence of the dielectric barrier, and gas composition. Other factors such as PD test circuitry, voltage above PD inception, electrode configuration, gap distance, temperature, humidity, and ambient noise levels [1] also impact PD measurements. PD occurring during the front time near the peak of the impulse is referred to as the main discharge with positive polarity. HV equipment is often subjected to harmonics and transients while in operation. This research is based on impulse voltage, which is discussed further in the next section.

2.3 Impulse Voltages

Lightning Impulse (LI) over-voltages are typically experienced by outdoor electrical equipment such as overhead lines and substations. The magnitudes of the LI voltages are usually in the order of thousands of kilovolt (kV). Each lightning stroke can inject currents up to hundreds of kiloampere (kA) into transmission systems. LI can be classified as whole or chopped lightning impulse voltages [13]. The standard chopped LI has a time-to-chopping range of 2 µs to 5 µs on the front and tail, respectively. In the IEC 60060-1 [14], LI test voltages are characterized by a front time of $T_1 = 1.2\,\mu s$ and time to half-value of $T_2 = 50\,\mu s$. Switching Impulse (SI) voltages are like LI but have a longer duration. SI are transient overvoltages that can have high magnitudes and are related to the operating voltage of power equipment. SI results from internal overvoltages caused by switching operations or faults [13]. Unlike LI, SI time parameters are defined concerning a true origin, and the standard switching impulse voltage, as determined by the IEC 60060-1 [14], is 250/2500 µs. SI impulses are characterized by a time-to-peak (T_p). T_p is the time from the true origin to the maximum value of an impulse [13]. Kuffel [15] states that the actual shape of the overvoltages is entirely different from that for testing purposes. However, it is necessary to simplify them to simulate their effects. Table 2.1 below summarizes the tolerances for standard impulse voltages.

Table 2.1: Tolerances as defined by the IEC 60060-1 [14] for standard impulse voltages

Tolerances		
Parameter	**LI**	**SI**
Test Voltages	±3%	±3%
T_1	±30%	-
T_p	-	±20%
T_2	±20%	±60%

IEC 60060-3 [16] recommends four impulse voltages for the impulse voltage withstand test for power equipment on-site testing. They are aperiodic LI, Oscillating Lightning Impulse voltage (OLI), aperiodic SI, and Oscillating Switching Impulse voltage (OSI). The waveform parameters are shown in Table 2.2 below.

Table 2.2: Parameters as defined by the IEC 60060-3 [16] for impulse voltages

Parameters			
Voltage type	T_1 (µs)	T_2 (µs)	f (kHz)
LI	0.8-20	20-400	-
OLI	0.8-20	20-400	15-400
SI	40-100	1000-4000	-
OSI	40-100	1000-4000	1-5

Inductances of circuit elements of an impulse voltage generator can introduce oscillations into the test voltage, thereby producing what is known as oscillating impulse voltages. In some cases, oscillations are required for on-site testing (IEC 60060-3) [16], and "damped alternating voltage" is used for field PD measurements [17]. This technique will not be developed in the present work as it requires adaptation of an oscillating impulse voltage generator, and oscillating over-voltages are not central to this investigation. Various PD detection and measurement studies using different detection techniques under impulse voltages have been conducted and are discussed in the following section.

2.4 PD Measurement Under Impulse Voltages

HV insulation materials are capacitive, and when subjected to impulse voltages, displacement currents of higher frequency content identical to impulse are developed within the dielectric [18]. If the voltage stress exceeds the inception voltage, PD can occur. PD pulses also have an impulse shape, and for shunt-based current measurements, displacement currents overlap significantly [18]. Using sensors, Garcia-Colon [4] investigated PD detection under impulse voltage on power transformers. Garcia-Colon [4] reported challenges experienced due to the sensitivity of the measuring system, such as when using acoustic PD sensors. Arrays of Ultra-Wide Band (UWB) sensors were located inside the transformer. Filters were installed on each UWB PD sensor with a cut-off lower frequency to filter out the impulse test voltage frequencies. Each UWB sensor had an amplifier for transmission of PD signal under both lightning and switching impulse voltage

application. However, sensors did not detect PD signals during the impulse voltage front time. Garcia-Colon [4] reported the sensor's surge protector operated faster than the duration of the front time of the applied impulse voltage. PD activities were discarded during impulse front time. Using UWB sensors had an advantage over acoustic sensors in terms of susceptibility to external electromagnetic fields. However, when internally positioned in a transformer, regular operation, which includes switching, poses disturbances to PD detection. Positioning of sensors also affected response time, which makes PD detection under lightning impulses more difficult [4].

Han *et al.* [17] adopted the Ultra High Frequency (UHF) sensor PD detection technique. In this study, the sensor is placed in front of the SF_6 insulated test cell of the needle-plane gap. An inductive probe is also used to determine the detection sensitivity for field studies. Detection sensitivity between the two techniques is compared by investigating PD activity behaviour on a needle-plane defect model in SF_6 when subjected to negative LI. PD signal is based on a UHF sensor and a High Pass (HP) filter for pulse current detection with a current transducer. PD measurements by pulse current detection were not recognized due to noise induced by lightning impulse voltage; However, the UHF method has higher sensitivity to external interference, and PD activity was detected during the impulse. Han *et al.* [17] reported that the UHF method has a better anti-interface capability; however, it is unsuitable for field tests as the pulse current detection method is susceptible to external interferences. The inductive probe is mainly adopted for experimental purposes, and the works are discussed below.

Wu *et al.* [5, 8] adopted the unconventional method for PD measurements at non-power frequencies. They investigated the effects of transient voltages on HV cables with an artificial defect under laboratory conditions. The test object is subjected to AC, impulse voltages of different polarities, and superimposed voltages. Two High-Frequency Current transformers (HFCT) of the same polarity were installed at both ends of the cable to measure the PD signals. Pulse polarity analysis was used to separate PD signals from noise and to mitigate disturbances. Wu *et al.* [5, 8] reported the position of the phase angle at which the impulse occurs, and the impulse polarity has no significant influence on the PD activity behaviour; therefore,

no PD was detected during the impulse moment. An additional band-pass filter was added to cope with disturbances. With the second filter added, disturbances were reduced, leading to limited measurements of PD pulses due to the dead zone being longer than the impulse front time [5]. Wu *et al.* [5, 8] reported PD activities with a longer impulse front time and a switching impulse [8]. From the latter, the inductive probe detection technique can detect PD signals during a longer front time, which is convenient for switching impulse investigation. Due to the induced noise, the PD current's magnetic fields cannot be detected during lightning impulse voltage application. Adding more filters reduced the chances of detecting the main PD activity, which is essential in the present investigation. Impulse voltage interferences, ground noise, and electromagnetic interferences make it challenging to detect PD under impulse voltage, which calls for separating interferences from the PD pulses without filtering them.

Li *et al.* [19] investigated the activity behaviour of PD in a polypropylene film under a needle-plane electrode setup, with a variable needle-plane gap distance under a lightning impulse of variable voltage amplitude. The PD detection system was based on a micro-current transducer with an anti-interference solution. Li *et al.* [19] reported no PD signal was detected due to trigger gap discharge interference being inevitable during every spark. Hayakawa *et al.* [20] adopted a similar PD detection technique. Hayakawa *et al.* [20] investigated the electrical insulation characteristics of the High-Temperature Superconducting cable insulation model. The test object was subjected to AC voltages superimposed on the lightning impulse voltage, resulting in PD signals not detectable during the impulse voltage application. Using an inductive probe for detecting PD under the lightning impulse voltage by Li *et al.* [19] was unsuccessful in investigating the corona. Therefore, corona behaviour under impulse voltage is still a 'holy grail', which is interesting in the present research.

Li *et al.* [21] later reported PD detection while investigating PD activity behaviour on needle-plane SF_6/epoxy interfaces model during the peak of lightning impulse voltage. The test object was subjected to all four types of impulse voltage waveforms. A measurement impedance was used to measure PD signals. A filter was also used to eliminate impulse noise and displacement current caused by the

impulse voltage [21]. PD detection techniques such as inductive probes and UHF sensors have been adopted for PD measurements under impulse voltages, and filters have been added to suppress noise. The addition of filters cannot separate the PD signal and the noise induced during the application of impulse voltage. Instead, it suppresses PD signals during the application of an impulse voltage. Another technique is the balanced bridge method, which has a higher interference rejection ability than the latter PD techniques. According to Liu *et al.* [22], there are some challenges in using the balanced bridge method under the HV application. The detection technique is discussed below as a basis for developing such a technique for use in the present study.

The balanced bridge method for PD detection consists of resistors and identical adjustable capacitors [22]. Noise can be suppressed when the electric bridge is balanced; however, the distribution of parameters can cause the bridge not to be balanced [22]. The balanced bridge method again requires that the identical capacitance samples, with and without PD, have power consumption compatible with the input resistors. Mitra *et al.* [3] and Densley *et al.* [6] reported discharges detected at approximately 5 pC using a bridge PD detection circuit. The output from the capacitance bridge was fed to the primary winding of a 1:1 pulse transformer. They investigated PD activity behaviour on artificial air-filled cavities of different dimensions in solid insulation under impulse voltage of 1/50 µs and 500/3000 µs. Kreuger [23] investigated PD measurement on a three-core belted power cable using the balanced bridge method. They reported that different PD results were obtained due to the sensitivity limitations of the measuring circuit.

The sensitivity of the balanced bridge circuit is directly proportional to the capacitance of the test objects and the interference of the impulse voltage [22]. The suppression of the noise is inversely proportional to the frequency bandwidth. The narrower the bandwidth, the higher the noise suppression ratio; however, the interference signals are only restrained at a fixed frequency when the bridge is balanced [22]. Therefore, in this investigation of cavity PD and corona under impulse voltage from literature, inductive probes, capacitive probes, antenna to detect PD, and acoustic sensors will not be adopted. From the latter, the antenna to detect PD and acoustic sensors are susceptible to electromagnetic interferences, whether

being used externally or internally, which makes them more defenceless to the challenge of realizing the detection of PD during the application of an impulse voltage. Whereas inductive and capacitive probes require filtering, suppressing PD signals with noise.

The balanced bridge method in the latter will also not be adopted in the present work due to the following sensitivity limitations of the measuring circuit:

- Input resistors and identical adjustable capacitors are required to balance the bridge.
- Test specimen power consumption must be compatible with the input resistors.
- The sensitivity of the balanced bridge circuit is directly proportional to the capacitance of the test specimen.
- No full PD frequency spectrum due to interference signals restrained at a fixed frequency when the bridge is balanced.

Generally, the balanced bridge method for PD detection requires two similar specimens and adjustable capacitors and resistors. An RLC filtering unit circuit comprising of identical windings on a toroidal core for PD detection was developed in Chile in 2017 [24]. The principle is not detailed, and no results are presented. However, the concept of a balanced bridge method will be used in this research with considered benefits and changes made to mitigate the sensitivity limitation of the existing measuring circuit to achieve PD detection under impulse voltage. The proposed balanced PD detection circuit comprises a high permeability ferrite toroid that is recommended for high-frequency application. The toroid core will comprise standard primary windings on which the test specimen will be connected. A similar specification PD specimen and a no PD specimen are adopted to allow signal cancellation. The toroid core will have a secondary winding on which PD will be measured using an oscilloscope. No discrete capacitors and resistors will be used for tuning.

The advantages of high permeability epoxy-coated ferrite toroid are:

- Higher interference rejection ability.
- Optimal impedance performance across the frequency range.
- Superior frequency response.
- Epoxy is rated at 200 °C for continuous operation.
- The voltage breakdown rating is 2 kV, which is wire-to-wire.
- It is cost-effective and easy to design.
- The equivalent inductance, resistance, and capacitance of the bridge are fixed as they depend on the winding wire with constant parameters.

PD detection under switching impulse, superimposed voltages will not be investigated due to laboratory limitations, and damped alternating voltages will also not be developed in the present work as it requires adaptation of an oscillating impulse voltage generator, and oscillating over-voltages are not central to this investigation. Wu *et al.* [5, 8] reported the position of the phase angle in which the impulse occurs, and the impulse polarity is insignificant to the behaviour of the PD activity. Therefore, only positive lightning impulses will be investigated in the present work. Few have partially reported PD measurements under lightning impulse voltages and are of interest in the present work. Controllable artificial defects for introducing the cavity PD and the corona will be used in this investigation, and the test specimen specifications are detailed in the next methodology chapter.

2.5 Summary and pointers to the next chapter

Understanding PD phenomena under impulse voltage is essential for improving HV equipment's reliability. The challenge is to enable PD detection during impulse voltage, due to the overlaps between the impulse voltage and the PD frequencies. PD detection and measurement techniques are discussed, as are the pros and cons of the detection systems. The circuit developed for this research work is

discussed in the next chapter. The next chapter covers experimental design, which includes the test specimen, impulse voltage, and the proposed balanced PD detection circuit. Lastly, the procedure for acquiring the experimental measurements is described.

Chapter 3

Experimental Setup Design and Development

This chapter presents the experimental setup and procedure for investigating cavity PD and corona characteristics under impulse voltage application. Experimental test specimens, an impulse voltage generator, and a balanced PD detection circuit optimization procedure are discussed to achieve the efficacy of PD pulse detection under impulse voltage.

3.1 Experimentation System Design and Setup

Detection and measurement of the energy emitted from a PD event can provide information about its location and source. PD detection circuits are based on the principles of processing energies emitted by PD events. Various PD detection techniques are discussed, noting that the ability to detect PD activity under impulse voltage depends on circuit sensitivity and limitations. In various ultra-wideband PD detection methods, that is, inductive probe and electromagnetic radiation detection using an antenna, PD signals under switching impulse voltage were reported under extended rise time of impulse voltage. PD signals have not been reasonably detected under a lightning impulse voltage. Therefore, it is important to address this issue to ensure accurate results and to add to existing knowledge.

The experimental setup is developed to investigate the characteristics of the cavity PD and corona under impulse voltage, with the aim of separating the PDs from the interference caused by the impulse voltage. The presented work adopts a balanced PD detection circuit, and the experimental setup is discussed in this section.

3.2 Experimental Work

The research was performed in an HV laboratory using a miniature impulse voltage generator of a maximum output voltage of 21 kV to introduce the lighting impulse voltage to the test specimen. The test specimen was assembled using the material available in the laboratory, and an off-the-shelf ferrite toroid core was purchased. The wire used to wind the ferrite toroid core to accomplish a balanced filtering circuit was also available in the laboratory. A digital Rigol oscilloscope DS1064B with a sampling frequency of 2 GS/s and a bandwidth of 60 MHz was used to measure PD activity from the ferrite toroid core to investigate the characteristics of cavity PD and corona under impulse voltage application. The experimental design is discussed in the following section.

3.2.1 Experimental design

The test specimen for the cavity PD experiment was a parallel plate electrode consisting of brass electrodes and sandwiched polyethylene sheets with one sheet having a cavity, as shown in Figure 3.1. The dielectric material was High-Density Polyethylene (HDPE) due to its high dielectric strength [25]. This study finds HDPE interesting because PD usually occurs within polyethylene defects in the HV system, which simulate cable insulation [26]. The complex relative permittivity is 2.3 on frequency band 1.0 kHz to 30 MHz for HDPE [27]. Ahmed *et al.* [28] reported that the average strength breakdown of HDPE is 79 kV/mm, and its advantage is that it is very flexible and has a high impact strength.

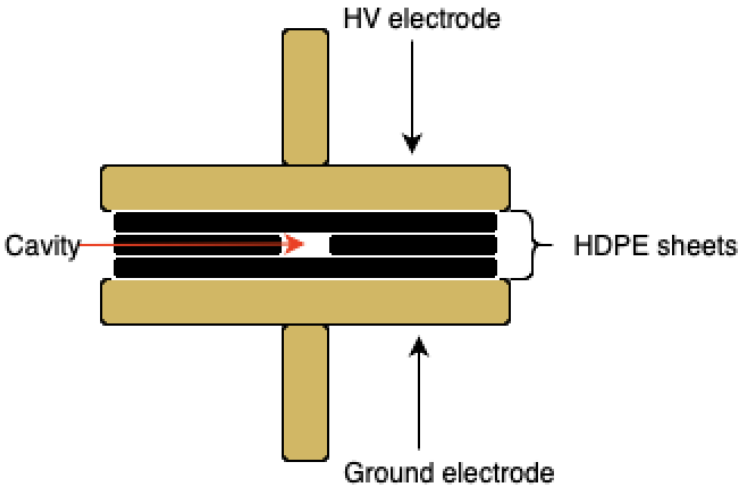


Figure 3.1: Parallel plate electrode test cell setup with an air-filled cavity.

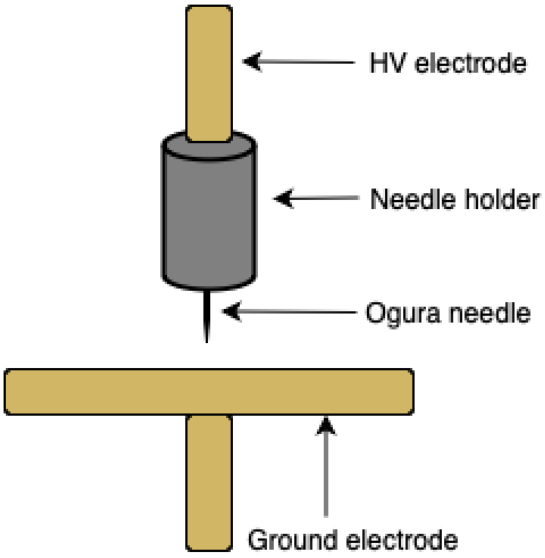


Figure 3.2: Needle-plane electrode setup for simulating corona discharges.

A needle-plane setup was used for the corona, shown in Figure 3.2. The Ogura™ needle of tip radius 5 µm was used to simulate corona due to its chemical properties and a very well defined needle tip profile which can withstand transient voltage

stresses without influencing PD pulses of interest. During the experiment, the cavity PD specimen was immersed in mineral oil to prevent surface discharges. Several identical test specimens were used for repeatable measurements.

3.2.2 Test specimens design

Figure 3.3 shows a parallel plate electrode test specimen with an air cavity embedded in insulation.

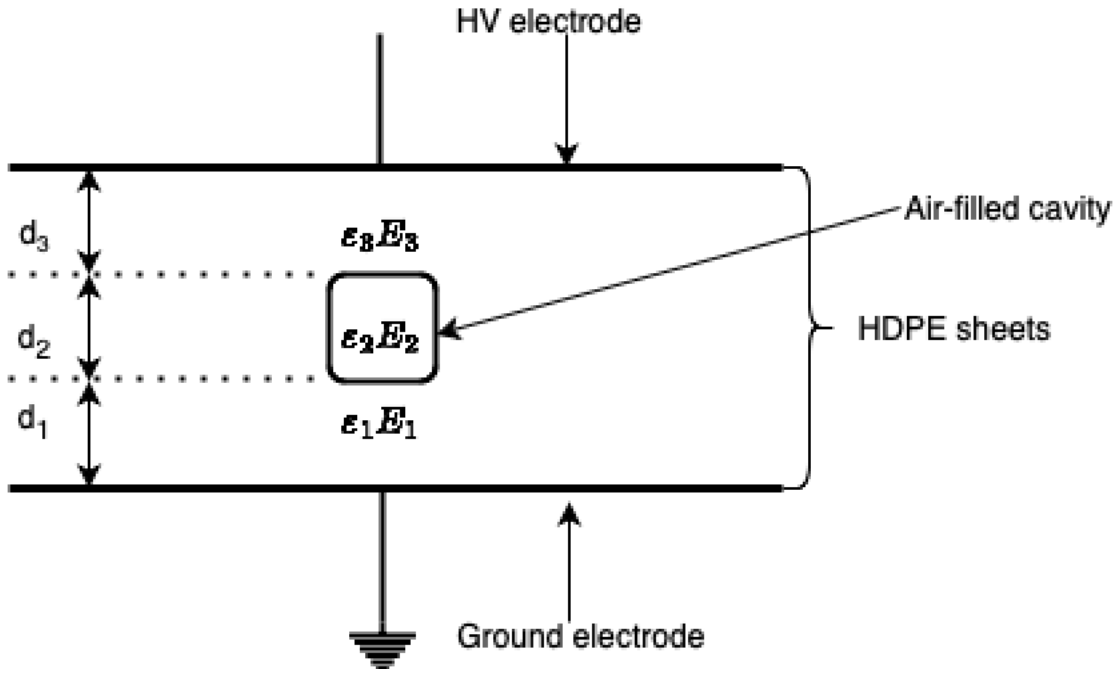


Figure 3.3: Test specimen of air-filled cavity embedded in insulation.

The electric field strength in the sections adjacent to the air cavity and inside the cavity and the corresponding voltages across the sections are expressed as given in equations 3.1 and 3.3. Let V_1, V_2 and V_3 be the voltages across each dielectric thickness d_1, d_2 and d_3, respectively.

$$\varepsilon_1 E_1 = \varepsilon_2 E_2 = \varepsilon_3 E_3 \tag{3.1}$$

$$\text{For parallel plate electrode setup, } E = \frac{V}{d} \tag{3.2}$$

$$\text{Then}, V_1 = d_1E_1, V_2 = d_2E_2, V_3 = d_3E_3$$

$$\text{Therefore}, V = d_1E_1 + d_2E_2 + d_3E_3 \tag{3.3}$$

Where:

E_1 : Magnitude of field intensity in polyethylene
E_2 : Magnitude of field intensity in the air cavity
E_3 : Magnitude of field intensity in polyethylene
d_1 : Polyethylene thickness
d_2 : Air cavity thickness
d_3 : Polyethylene thickness
$\varepsilon_1 = \varepsilon_3$: The relative permittivity of the dielectric
ε_2 : The relative permittivity of air in the cavity
V : Applied voltage to the test specimen

The stress in the air-filled cavity becomes ε_2 times that in the insulation. Suppose the stress exceeds the air cavity breakdown voltage determined by the Paschen curve. PD will be initiated, which results in a progressive insulation degradation process.

High altitude tests were conducted at Wits University and an altitude of approximately 1740 m Above Sea Level (asl) was recorded [29], for a cavity size of 1.5 mm (per sheet size); thus, at 0.8 bar [29] from the Paschen curve, the minimum breakdown voltage is 4.5 kV. The corresponding electric field in the cavity for spark over to occur is as calculated in equation 3.4 assuming breakdown of air at a standard temperature of 20°C.

$$V_2 = d_2E_2 \tag{3.4}$$

$$E_2 = 3\,\text{kV/mm}$$

Known:

d_1=1.5 mm
d_2=1.5 mm
d_3=1.5 mm
ε_1=2.3

ε_2=1
ε_3=2.3

The voltage V, to be applied on the electrodes to establish an electric field of E_2 = 3 kV/mm in the cavity that will result in the cavity sparking over, is calculated as follows:

$$E_2 = \frac{V_{sparkover}}{\frac{\varepsilon_2 d_1}{\varepsilon_1} + \frac{\varepsilon_2 d_2}{\varepsilon_2} + \frac{\varepsilon_2 d_3}{\varepsilon_2}} \tag{3.5}$$

Rearranging to substitute known values, $V_{sparkover}$ = 8.4 kV. The impulse voltage profile must breach the 8.4 kV value on the rising edge of the impulse voltage waveform for PD to be initiated. Taking into consideration the effect of temperature on $V_{sparkover}$ which is directly proportional to temperature [30], the impulse voltage magnitude was gradually increased until the PD pulses were noticed at approximately 7.9 kV.

Figure 3.4 below shows the simulated electric field profile using Finite Element Method Magnetics.

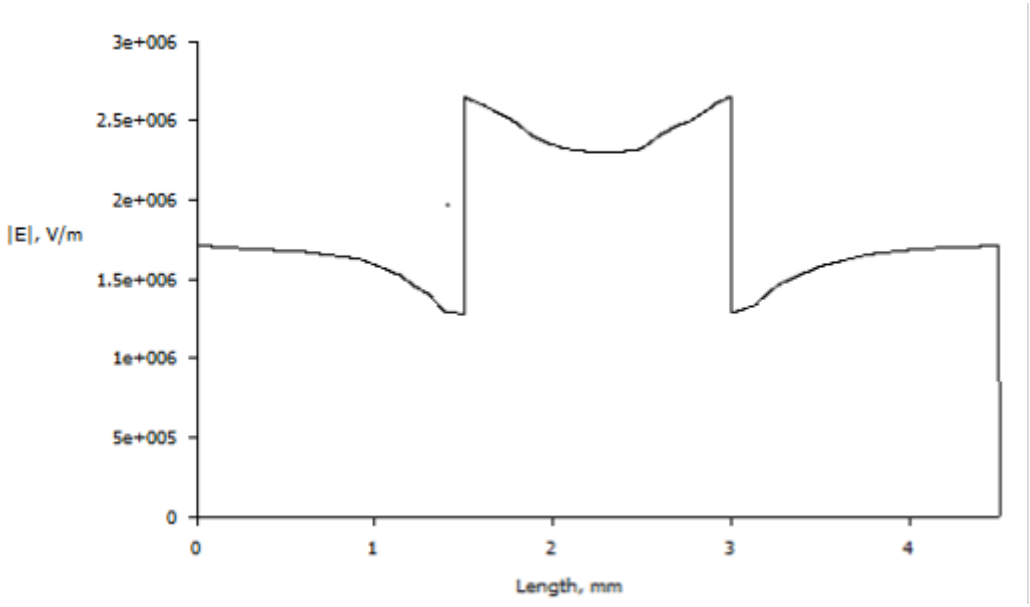


Figure 3.4: Electric field profile of a sandwiched 1.5 mm cavity.

The method used to promote corona discharges is the needle-plane electrode test specimen using an Ogura™ needle, as shown in Figure 3.5. The needle radius was 5 µm, commonly used for corona studies in the laboratory.

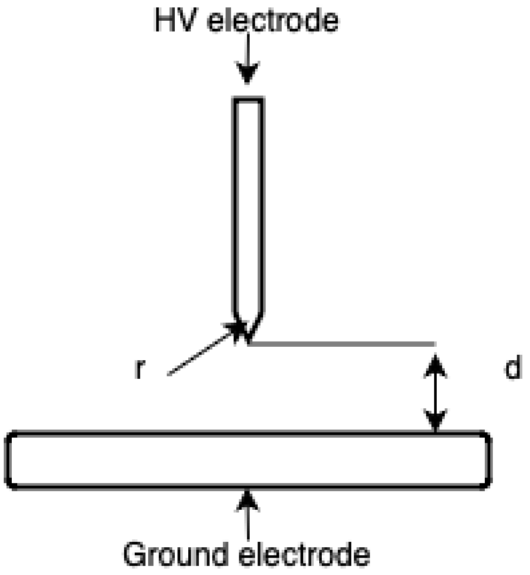


Figure 3.5: Test specimen of the needle-to-plane electrode setup to promote corona discharge.

Mason's calculation formula for the maximum field intensity of a needle–plane electrode setup is shown below.

$$E_{(max)} = \frac{2V}{r \ln(1 + \frac{4d}{r})} \tag{3.6}$$

Where:

V: Applied voltage
d: Distance from needle tip-to-plane electrode
r: Radius of the needle tip
$E_{(max)}$: Is the maximum field intensity between needle tip-to-plane electrode

The electric field intensity of the needle–a plane gap of 10 mm is obtained through simulation.

Figure 3.6 below shows the simulated electric field profile using the Finite Element Method Magnetics software.

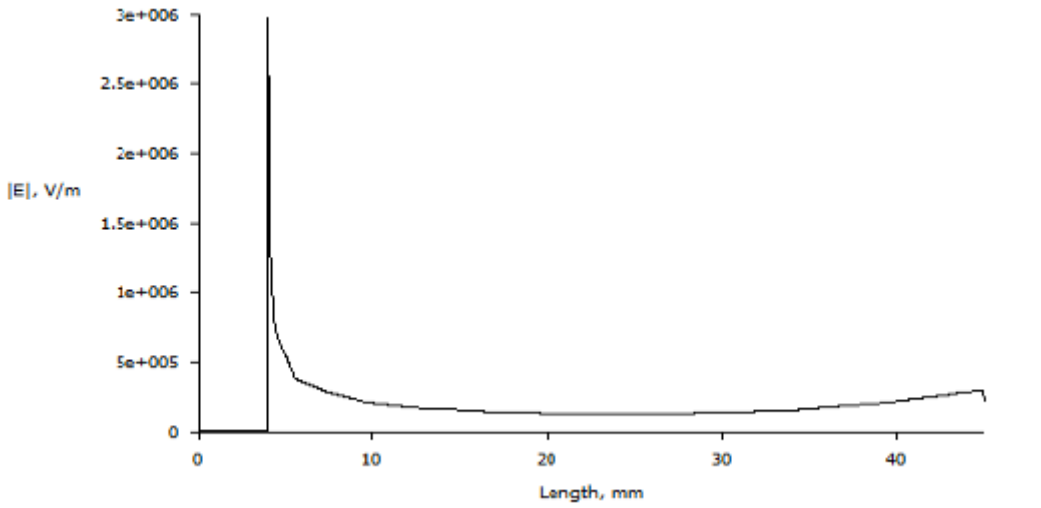
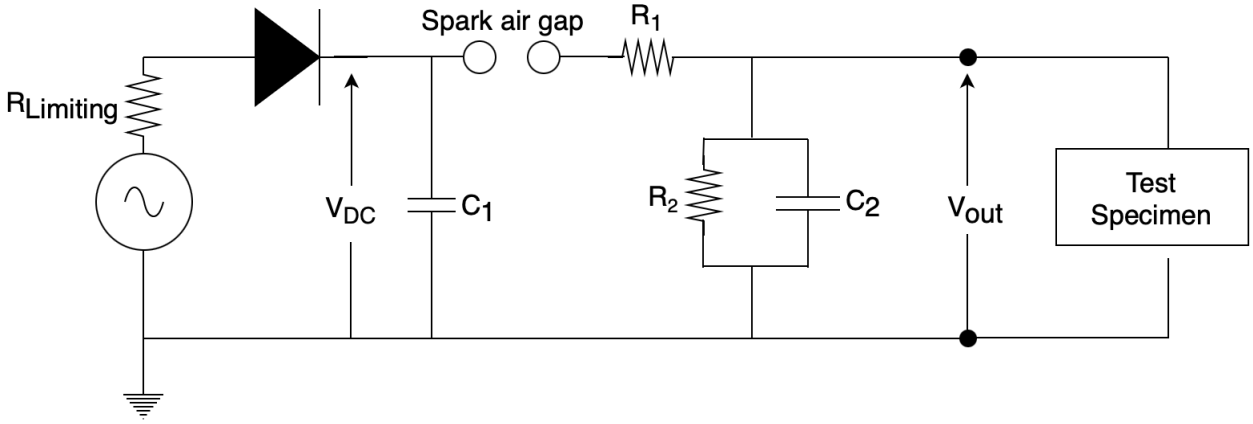


Figure 3.6: Electric field profile of 10 mm **air gap between the needle and grounded electrode.**

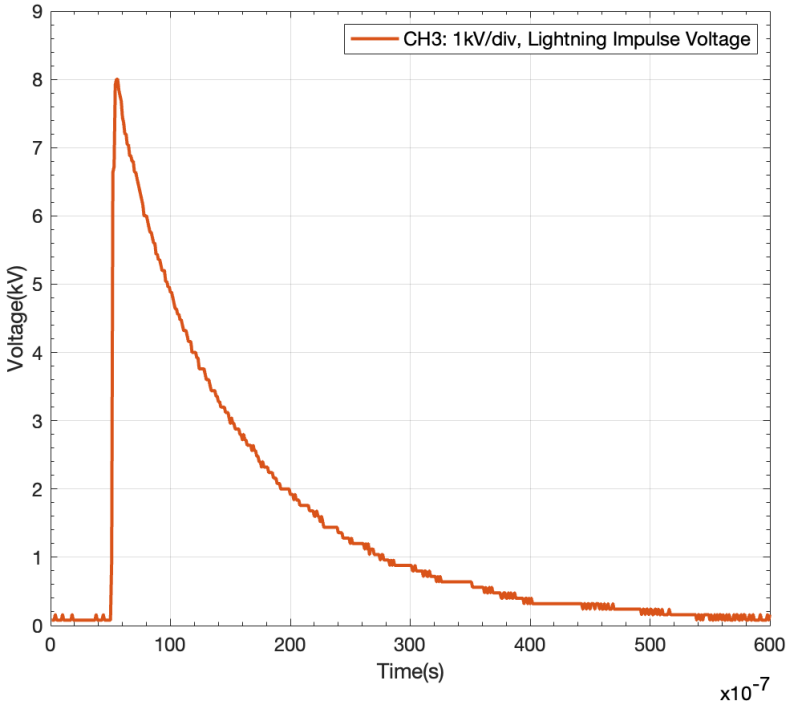
The calculations and simulated results for both test specimens correlate, so both were used in this research to characterize cavity PD and corona under impulse voltage application.

3.2.3 The impulse voltage source: miniature impulse voltage generator

The available miniature impulse voltage generator produced the standard lightning impulse voltage of 8 kV, with a rise time and fall time of 1.2 µs and 52.27 µs, respectively [14] as schematically presented in Figure 3.7. The impulse generator comprises of $R_1 = 58.6\,\Omega$, $R_2 = 1.654\,\text{k}\Omega$, $C_1 = 0.004\,\mu\text{F}$, and $C_2 = 0.01\,\mu\text{F}$.



(a)



(b)

Figure 3.7: (a) Miniature impulse voltage generator circuit diagram, (b) Impulse generator waveform.

3.2.4 Design of the balanced PD detection system

To successfully detect PD signals and suppress interference, it is critical to optimize toroidal windings. The measurement of the signal depends on the primary and secondary winding turns, the permeability of the toroid core, the cross-sectional area, the length, and the frequency response of the toroid core. Experimental work on magnetic cores frequency response characteristics is presented [31]. Magnetic saturation is ignored due to the milliampere level of the PD pulses [32]. The optimization of primary and secondary turns on the toroid core was achieved by injecting a pulse into the primary windings and measuring the optimum output on the secondary windings using a PD calibrator and a digital scope. The balanced PD detection circuit sensitivity optimization test setup is shown in Figure 3.8. The PD calibrator was connected directly to the digital scope, and the magnitude of the PD pulse of 1.1 V_p was measured at 100 pC. The balanced PD detection circuit sensitivity proof test steps are shown in Appendix B.

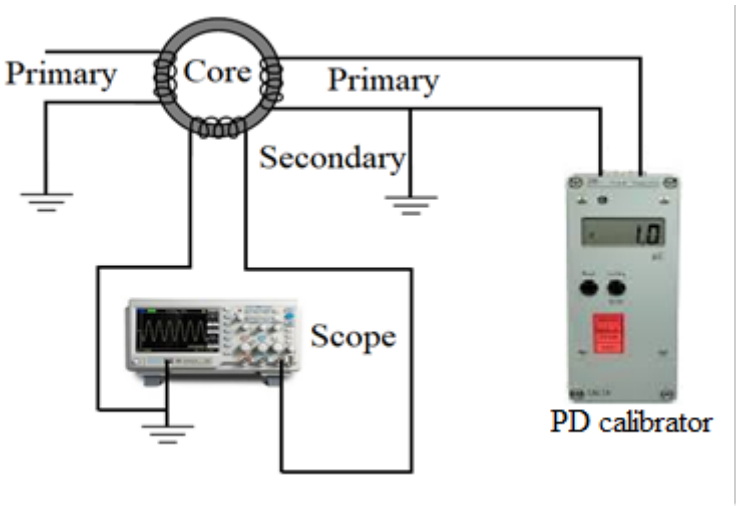


Figure 3.8: PD detection circuit sensitivity optimization.

The toroidal core outside diameter is 36 mm, the inside diameter is 23 mm, and the height is 15 mm coated. The primary turns are varied, and the measured signal is captured. Figure 3.9 shows the optimum measured output signal of 560 mV at six primary turns per leg. As a function of the number of secondary turns, the balanced filtering circuit output gave an optimal output of 1060 mV at four

secondary turns, as shown in Figure 3.10. The chosen coated magnet wire is a copper wire of $1\,\mathrm{mm}^2$.

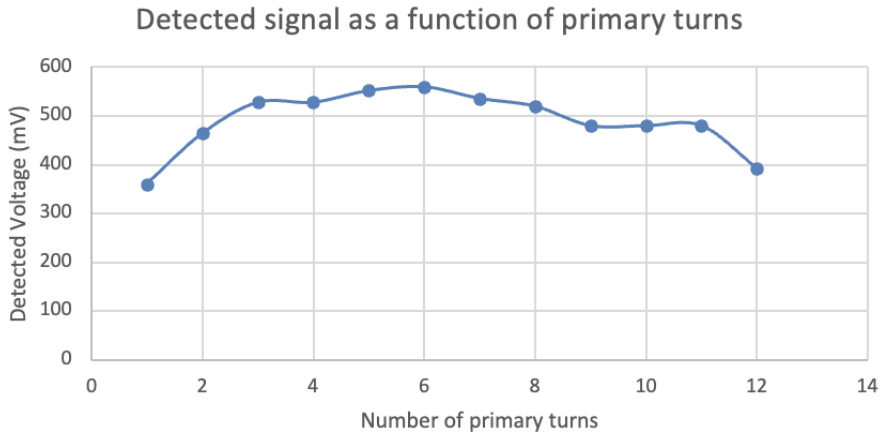


Figure 3.9: PD detection circuit output as a function of the number of primary turns.

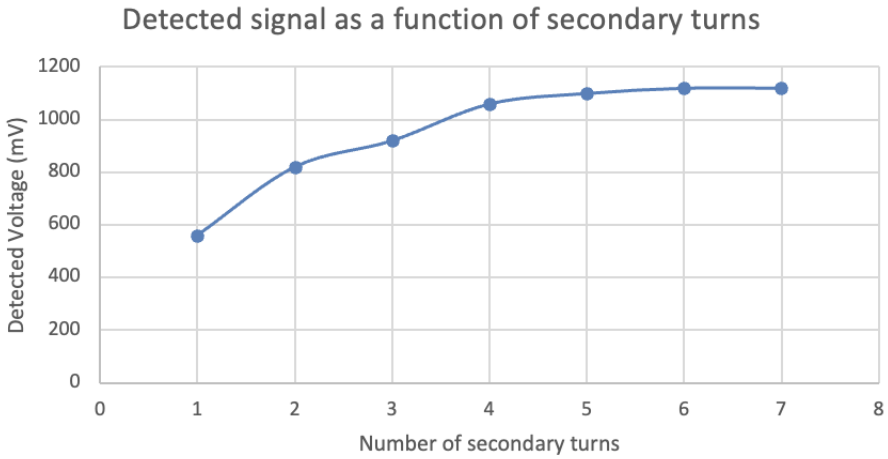


Figure 3.10: PD detection circuit output as a function of the number of secondary turns.

The circuit was constructed to filter impulse voltage noise and the displacement current caused by the impulse voltage. Furthermore, the input and output signal cables are shielded to block electromagnetic interference and ground noise, and the circuit is placed in a shielded box, as shown in Figure 3.12. The effectiveness of shielding was also proven by Karagiannopoulos *et al.* [33]. Figure 3.11 shows the physical construction of the unshielded balanced PD detection circuit.

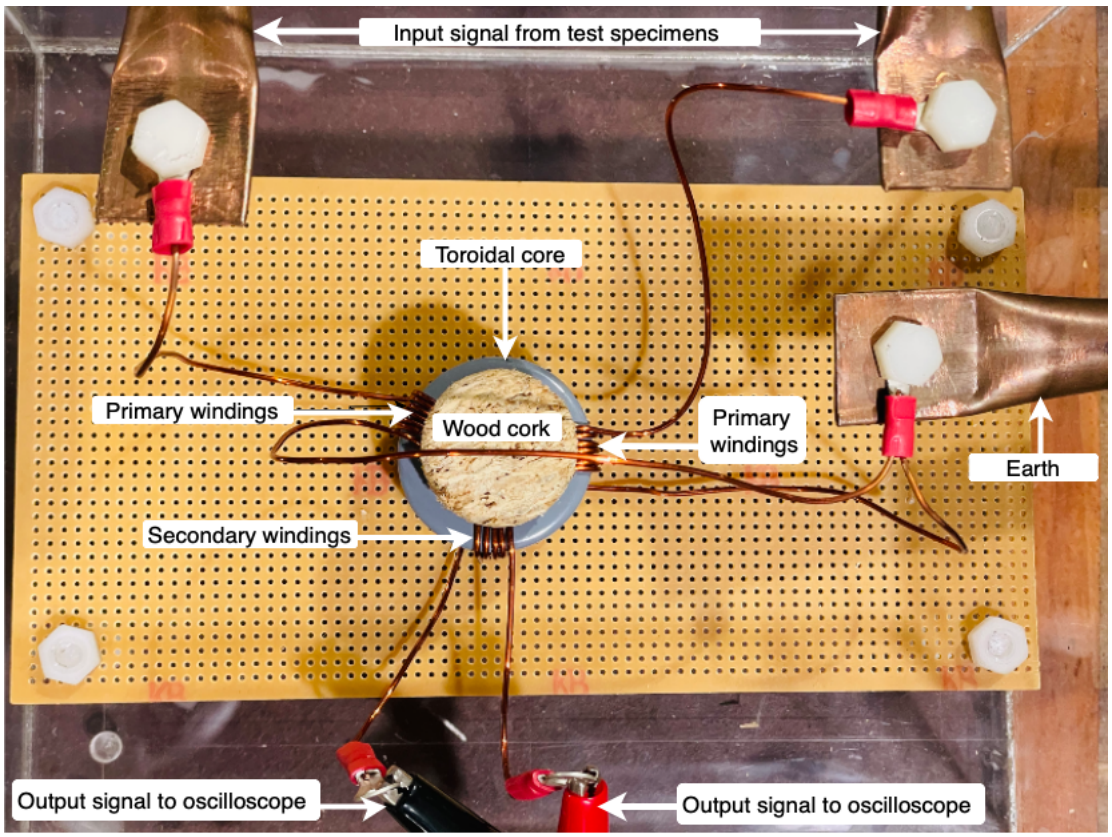


Figure 3.11: Unshielded PD detection circuit

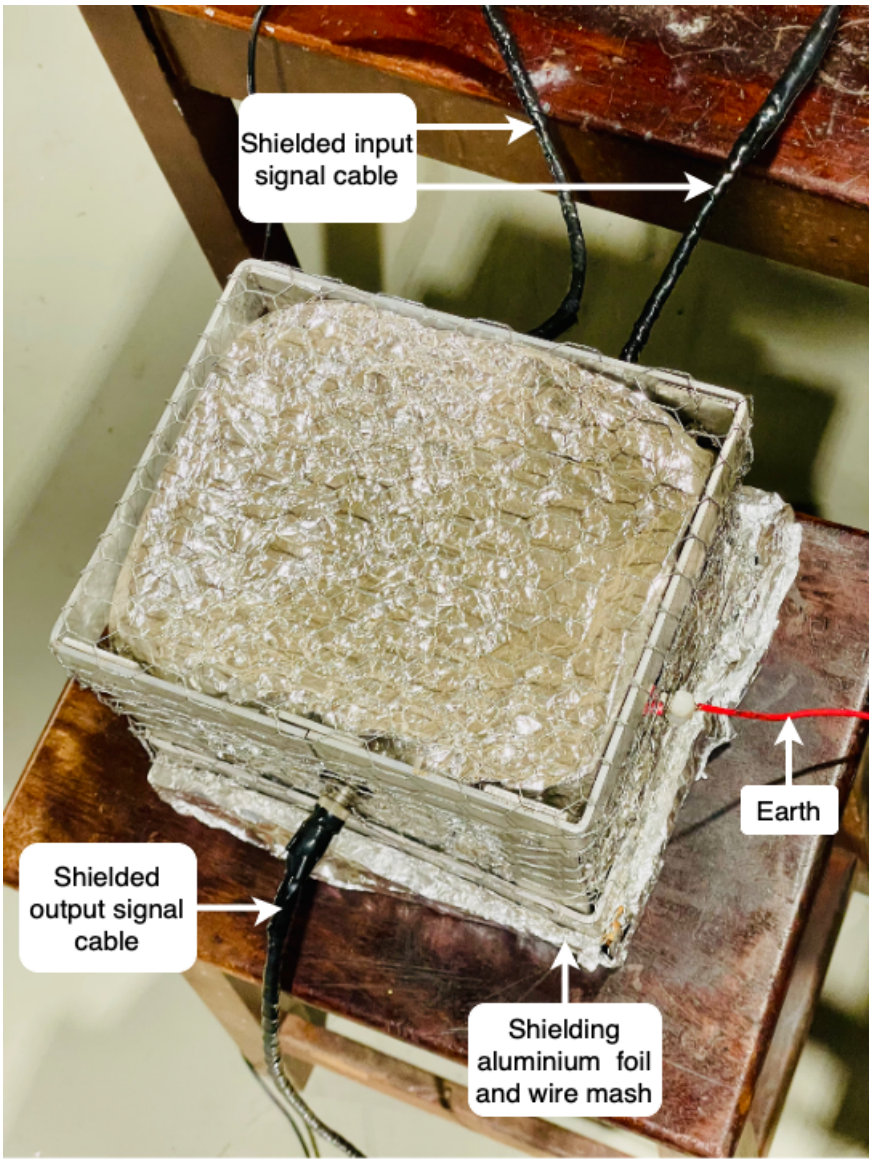


Figure 3.12: Shielded PD detection circuit

The balanced filtering circuit is constructed so that the signals coming from the test specimen cancel each other, leaving only PD signals in the Toroid core. PD signals will be captured on the secondary windings, connected directly to the digital scope, and optimized to achieve a 1:1 ratio (actual value). The Toroidal core data sheet is also attached in Appendix B.

3.3 Cavity PD Measurement Experimental Work

The test specimen showing the cavity PD defect and dimensions in section 3.2.2 was used in the experimental investigation of the characterization of the cavity PD under impulse voltage. The defect simulates cavities embedded in HV insulation systems during manufacturing or assembly or due to stresses while in service.

3.3.1 The experimental test setup

A balanced PD detection circuit comprising standard primary and optimized secondary windings was used with two identical test specimens, one with a cavity defect and the other without a cavity. A Rigol 6 GHz digital scope was used to acquire PD from the secondary windings. It displays both applied impulse voltages measured via the voltage probe and PD magnitudes detected from the balanced PD detection circuit secondary windings. The laboratory experimental test setup is shown in Figure 3.13 for investigating the characteristics of cavity PD under impulse voltage. A picture of the laboratory experimental test setup is also shown in Figure 3.14

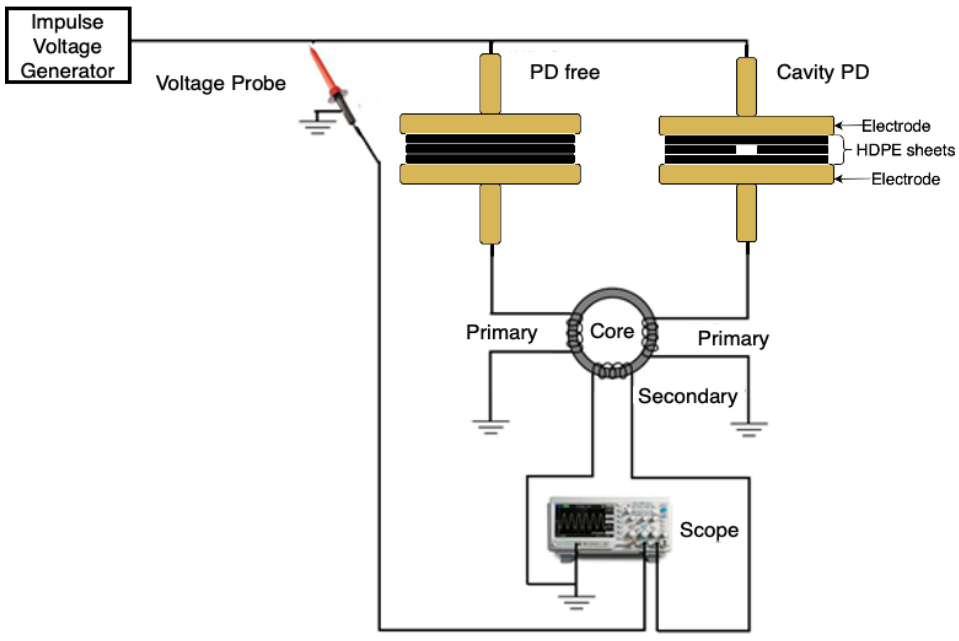


Figure 3.13: Test setup for cavity PD experiment.

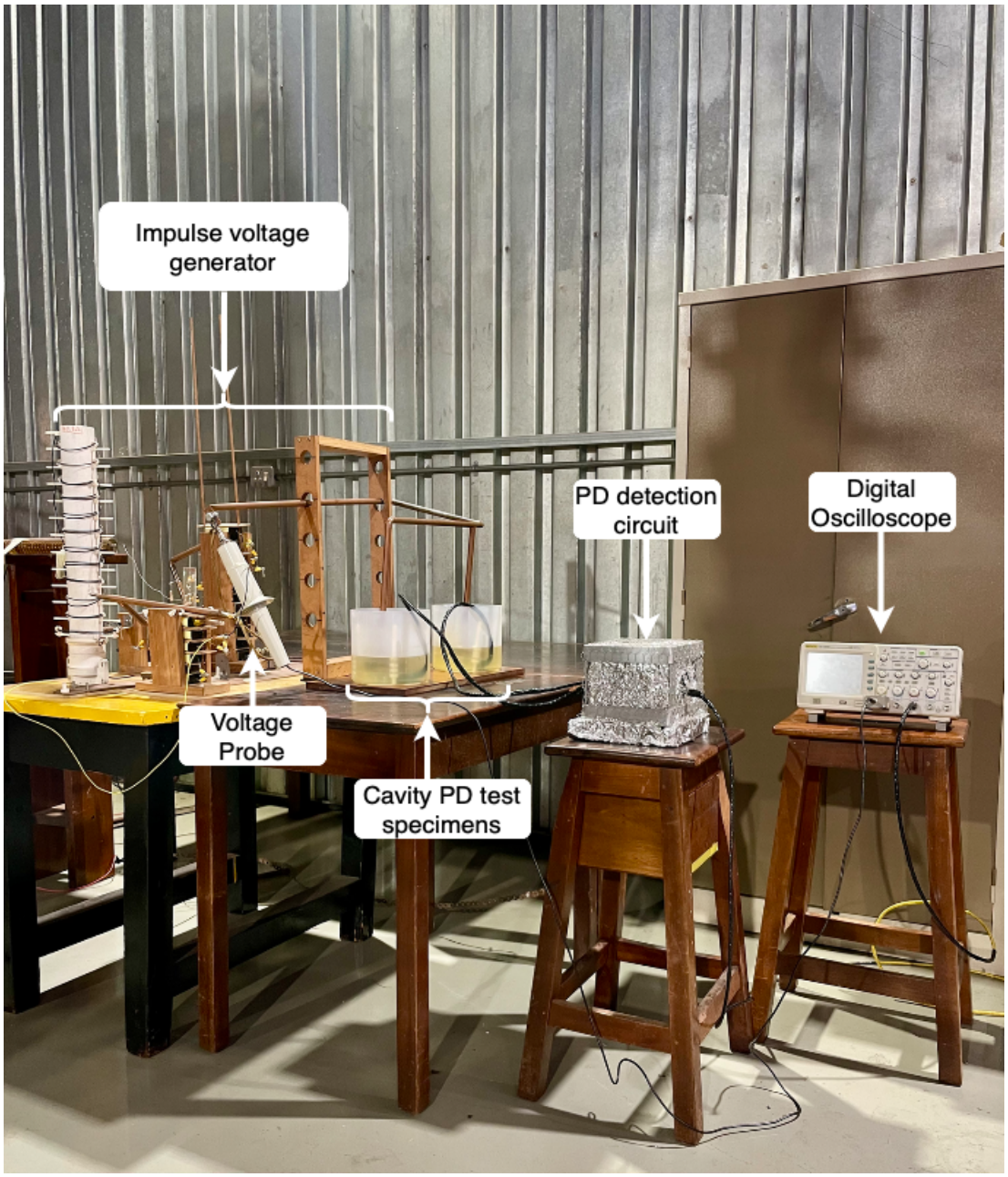


Figure 3.14: A picture of PD detection circuit setup for cavity PD experiment.

3.3.2 The experimental procedure

The test voltage was chosen as substantiated by the calculations and Finite Element Method Magnetics (FEMM) to cause stress in the defect for initiating PDs.

Impulse voltage shots of incremental peak magnitude were applied to the test specimen until PD activity was observed. The same impulse voltage magnitude was kept constant when applying shots to the specimen, and PD signals were recorded in every event. One-minute time intervals between shots were maintained to allow space charge dispersion that would otherwise significantly influence PD activity.

Figure 3.15 shows PD-free specimens connected for noise cancellation benchmarking tests to record output signals for comparisons between PD-free and PD defect specimens. This also enabled any discrepancies or false signals to be noticed. All experiments were conducted with shielded PD detection circuits and input and output signal cables. The cavity PD specimens were immersed in mineral oil during the experiment to prevent surface discharges.

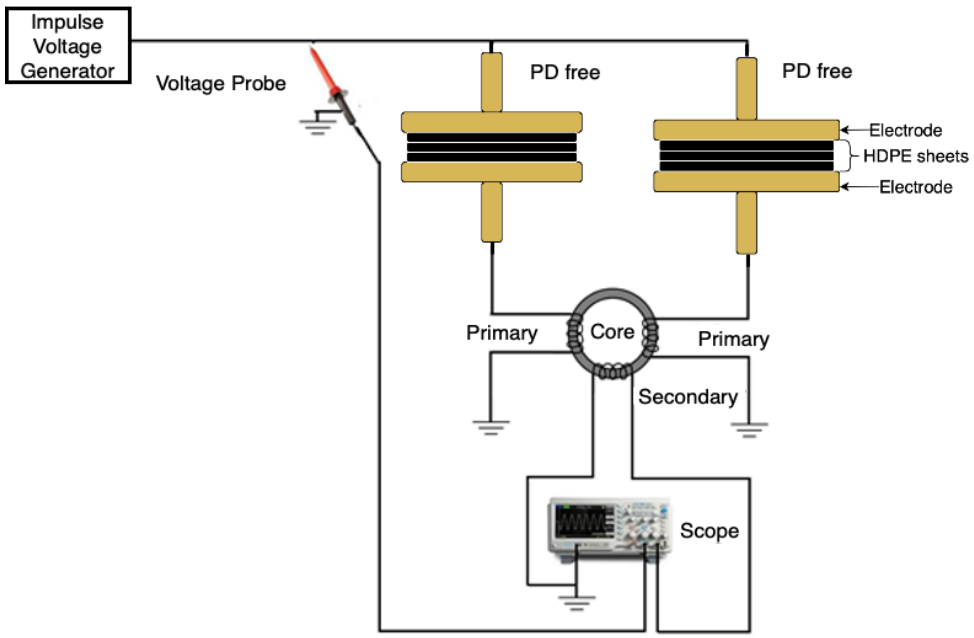


Figure 3.15: Test setup of cavity PD-free specimens for background noise benchmarking and balanced noise filtering proof-concept.

3.4 Corona PD Under Impulse Voltage Experimental Work

Section 3.2.2 discusses the test specimen for corona defect used in the experimental investigation of corona characterization under impulse voltage. The defect simulates imperfections that form sharp edges in the HV insulation or metallic parts in the system during manufacturing or production.

3.4.1 The experimental test setup for corona PD measurements

A balanced PD detection circuit comprising standard primary and optimized secondary windings is used with two specimens with similar specifications, one with a corona and the other without a corona. A Rigol 6 GHz digital scope acquired PD from the secondary windings. It displayed both applied impulse voltages measured using the voltage probe and PD magnitudes detected from the balanced PD detection circuit secondary windings. The laboratory experimental test setup is shown in Figure 3.16 for investigating the characteristics of corona under impulse voltage. A picture of the laboratory experimental test setup is also shown in Figure 3.17. Both positive corona and negative corona were investigated.

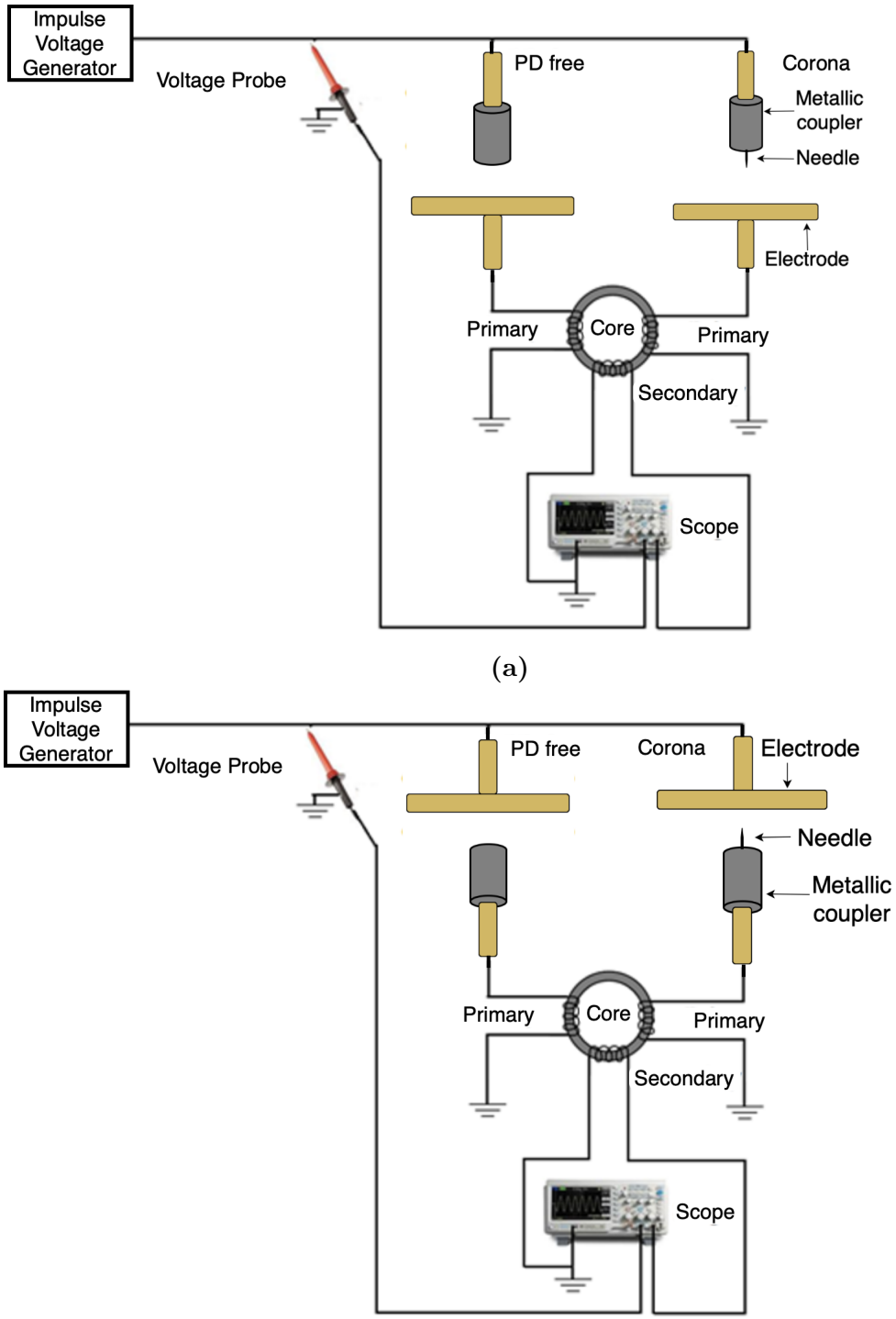


Figure 3.16: (a) Test setup for positive corona (b) Test setup for negative corona experiment.

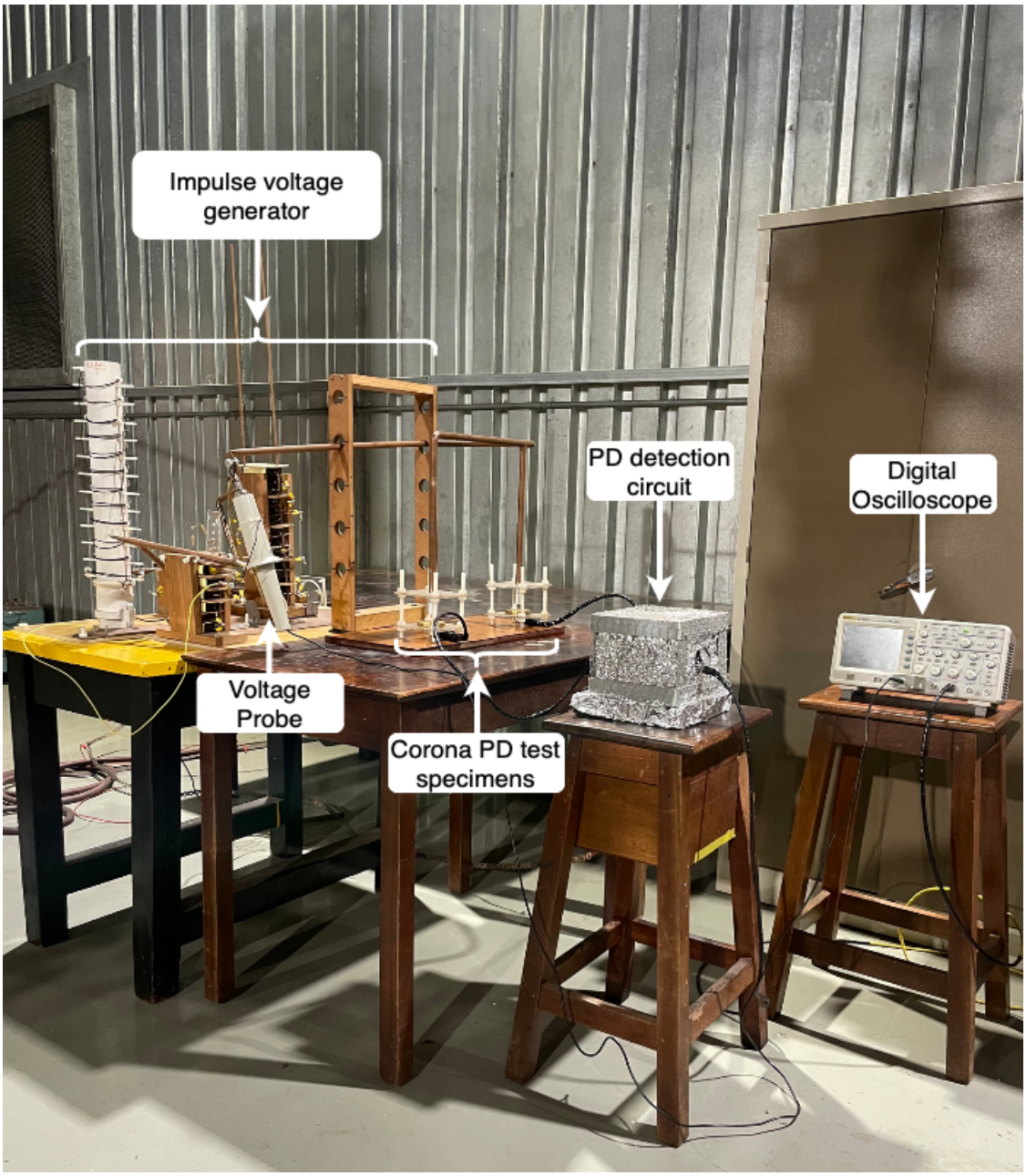


Figure 3.17: A picture of PD detection circuit setup for corona PD experiment.

3.4.2 The experimental procedure

The test voltage was chosen as substantiated by the calculations and Finite Element Method Magnetics (FEMM) to cause stress in the defect to initiate PDs.

Impulse voltage shots of incremental peak magnitude were applied to the test specimen until the corona was observed; see Figure 3.18. The same impulse voltage magnitude was kept constant when applying shots to the specimen, and PD signals were recorded in every event.

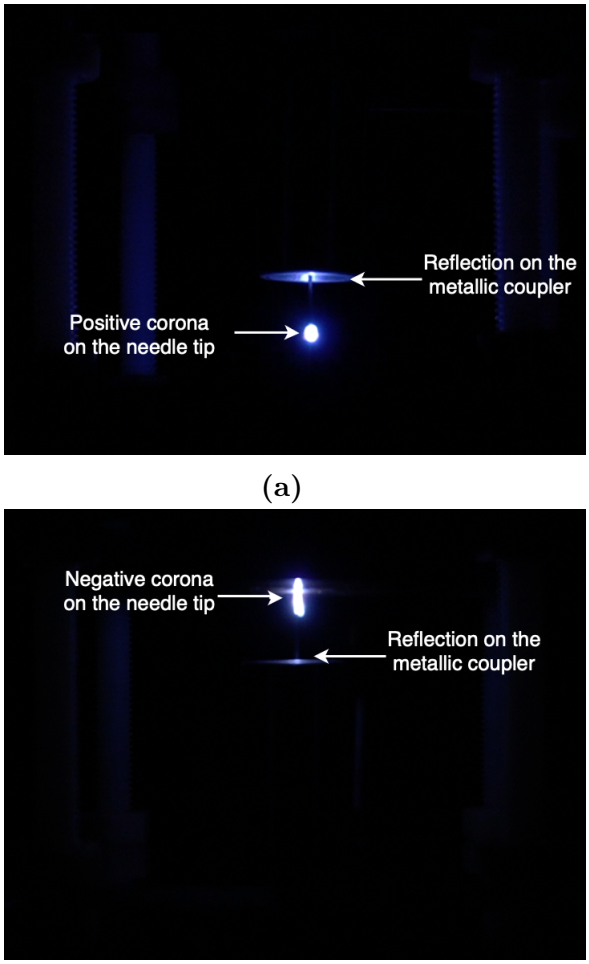


Figure 3.18: Still photograph of (a) positive corona and (b) negative corona discharge taken with a normal camera. Notable thin streamer and lower light reflection intensity on negative corona metallic coupler.

Figure 3.19 below shows corona PD-free specimens connected for noise cancellation benchmarking to record output signals to allow comparisons between PD-free and PD specimens. This will also enable any discrepancies or false signals to be noticed. All experiments are conducted with shielded PD detection circuits and shielded input and output signal cables.

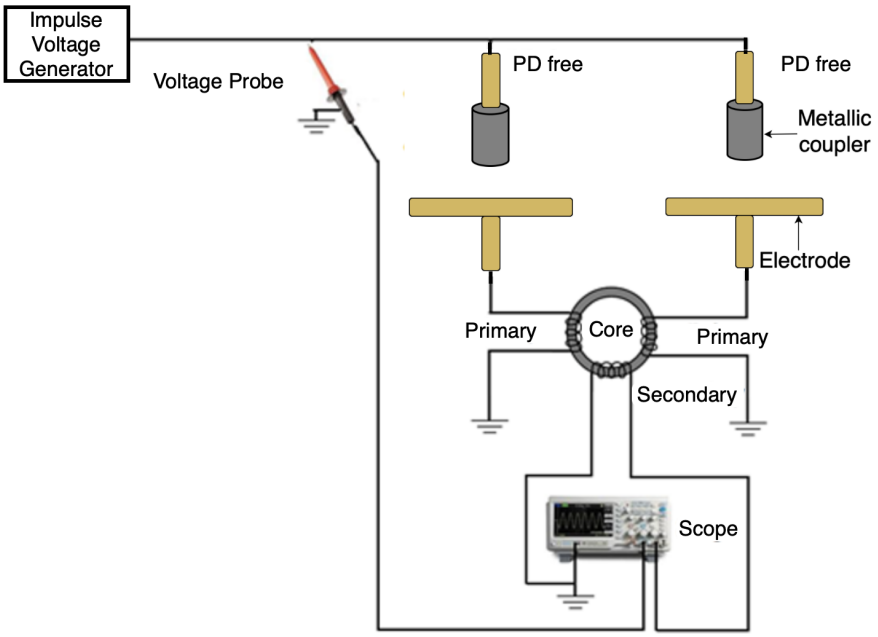


Figure 3.19: Test setup of corona PD-free specimen for noise benchmarking and balanced noise filtering proof-concept.

3.5 Summary and pointers to the next chapter

This chapter outlines the experimental system design and procedure used to achieve the efficacy of PD detection under impulse voltage. The next chapter presents the experimental results of PD pulses obtained for cavity PD and corona. The analysis follows the results, discussing the distinct findings between cavity PD and corona.

Chapter 4

Experimental Results, Analysis and Discussion

This chapter presents the experimental results, analyses, discussion, and comments on the practical challenges encountered in the experimental measurement of cavity PD and corona. The results are interpreted using the cavity PD mechanism theory and the theory of corona discharges.

4.1 Results: Cavity PD

4.1.1 Measurements and observations

The PD mechanism for cavity PD under LI voltage is analyzed. For every LI voltage short, the main PD pulses occur during the rise time of the LI voltage. A reverse PD pulse occurs on the LI voltage tail because a space charge is deposited and trapped on the cavity walls. The established voltage by the residual space charge can cause the generation of reverse discharges of either positive or negative magnitude because of the duality of space charge during the impulse voltage fall time. Space charge accumulation is observed to influence the PD pulse [34] significantly. The space charge generated by PD events under LI voltage shorts can promote or demote main PD and reverse PD under repetitive LI voltage shorts.

From Figure 4.1, it can be seen that the balanced PD detection circuit suppresses the impulse voltage interferences, and for every experimental work, noise cancellation confirmation is done.

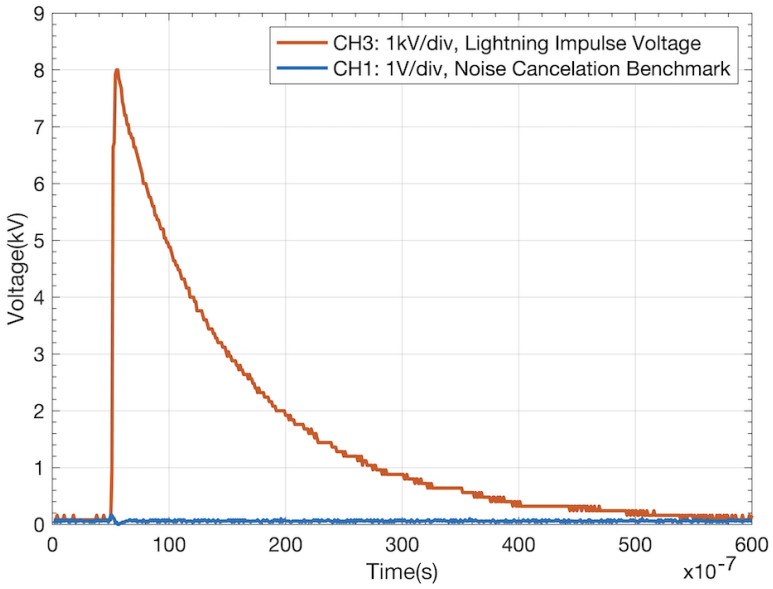


Figure 4.1: Applied impulse voltage waveform and balanced PD detection circuit output with no PD.

Figure 4.2 shows PD signals detected for cavity PD under impulse voltage. The main discharge with positive polarity occurs during the front time near the peak of the impulse voltage, and the reverse discharge with negative polarity occurs during the fall time of the impulse voltage. Other similar pulses that were recorded during repeated measurements are shown in Appendix C.

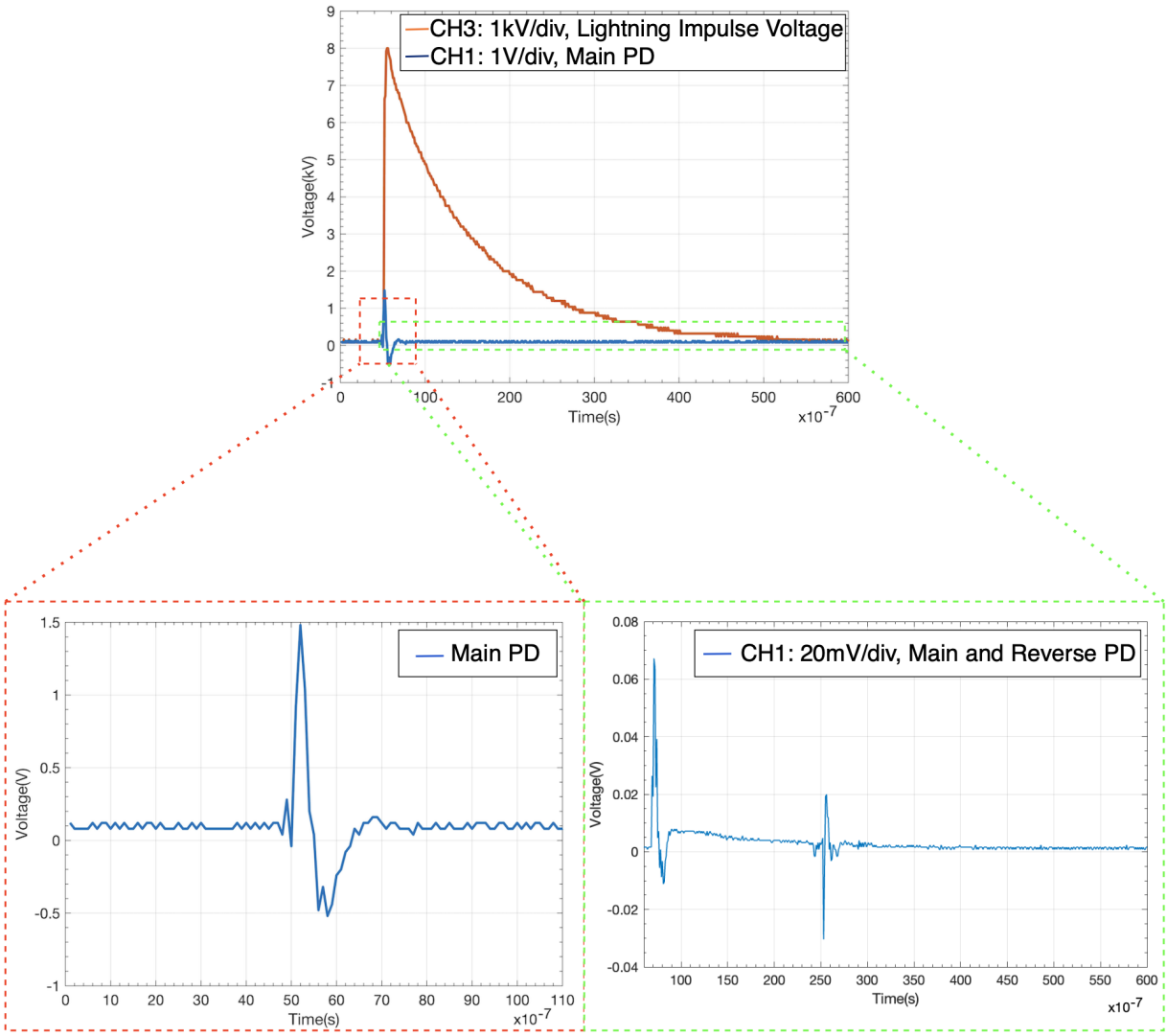


Figure 4.2: Impulse voltage short and detected PD pulses of main PD waveform at the scale of $1\,\mathrm{V/div}$. Notable, both main PD and reverse discharges were noticed at the scale of $20\,\mathrm{mV/div}$. Dashed frames are the zoomed main and reverse PD pulses.

Reverse discharges of positive polarity were detected during the tail time of the lightning impulse voltage. The magnitude of the reverse discharge is small compared to the main PD pulse. Other recorded similar pulses during repeated measurements are also shown in Appendix C.

4.1.2 Interpretation of the measurements

The PD mechanism under lightning impulse voltage is analyzed. During the impulse voltage application, the main PD is detected in the rising part of the impulse voltage. Due to the space charge generated by the main PD event, reverse PD is also detected in the fall phase of the impulse voltage. The residual voltage of the cavity is not zero under impulse voltage stress. Space generated from previous PD events may not necessarily decay to zero. Depending on the polarity and magnitude of the residual voltage, the main PD magnitude is affected, and for the same reason, positive and negative reverse PDs are detected.

As illustrated in Figure 4.3, $E_{impulse}$ is the electric field established in the insulation by the impulse voltage ($V_{impulse}$), and E_{space} is the electric field established by the accumulation of space charge on the cavity walls. E_{defect} is the electric field established in the cavity by the impulse voltage. On the impulse voltage tail, as the voltage magnitude continues to fall, E_{space} becomes greater than E_{defect} such that the resultant electric field ($E_{resultant}$) becomes greater than the PD inception electric field in the cavity, resulting in the initiation of reverse PD.

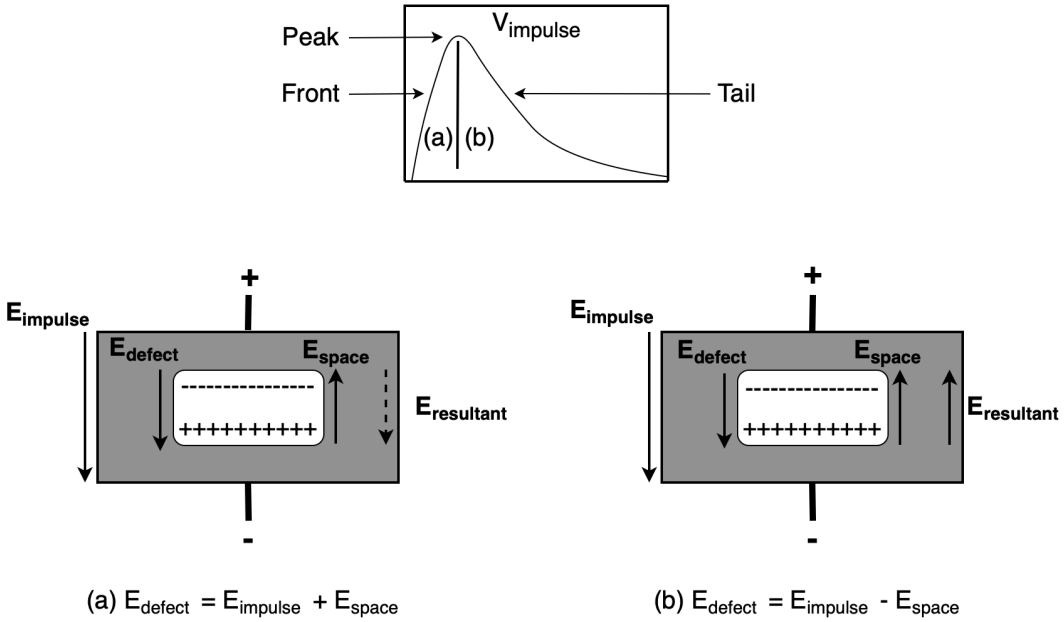


Figure 4.3: Cavity PD internal electric field during application of LI voltage.

4.1.3 Practical challenges in PD signal measurements

The main challenge with PD detection under impulse test voltage was to filter out the small PD from the impulse voltage, which was achieved through a balanced PD detection circuit. Furthermore, interferences would still occur through radiated electromagnetic emission from the impulse generator. Mitigation in that regard was through the shielding of the PD detection circuit. For each experimental test, noise benchmarking was conducted to prove the circuit sensitivity and ensure that the detected pulses were the cavity PDs. Unlike corona, cavity PDs can not be seen if there is a breakdown. All the recorded PD pulses were measured on a digital oscilloscope, and the input and output cable of the balanced PD detection circuit and the circuit itself were shielded. It is worth noting that the observed reverse PD magnitudes are of small scale 20 mV/div compared to the main PDs scale 1 V/div. Separate oscillographs for showing reverse PDs are captured at 20 mV/div (zoomed) to show a clear oscillograph telling a story.

4.2 Results: Corona

4.2.1 Measurements and observations

Corona in air detected on the needle tip had distinct characteristics similar to those reported in the literature [35, 36, 37, 38, 39, 40]. Positive and negative corona were observed during the rise time of the LI voltage. The positive corona PD magnitudes are higher than the negative corona PD magnitudes. Patiwa *et al.* [41] investigated the polarity effects of corona discharge and stated that positive corona allows more streamer discharge. In contrast, the negative corona had a less glow-like discharge, similar to the observed image in Figure 3.18 (b). Bousiou *et al.* [42], and Hang *et al.* [39] studied corona characteristics dependence on the applied impulse voltage where streamer corona is established. The PDs detected for negative corona were approximately two times lower than positive corona [42].

Unlike air cavities embedded in insulation, where space charge deposited by PD events is retained in air, space charge diffuses away. Consequently, no reverse discharges were detected on the impulse test voltage tail, as shown in Figure 4.5.

The absence of reverse discharges on the tail was also apparent for negative corona under impulse test voltage, as shown in Figure 4.6. Corona was detected at the impulse voltage front time near the peak, which was also reported by Wang *et al.* [37]. Other similar pulses recorded during repeated measurements are shown in Appendix D.

From Figure 4.4, it can be seen that the balanced PD detection circuit suppresses the impulse voltage interferences, and for every experimental work, noise cancellation confirmation is done.

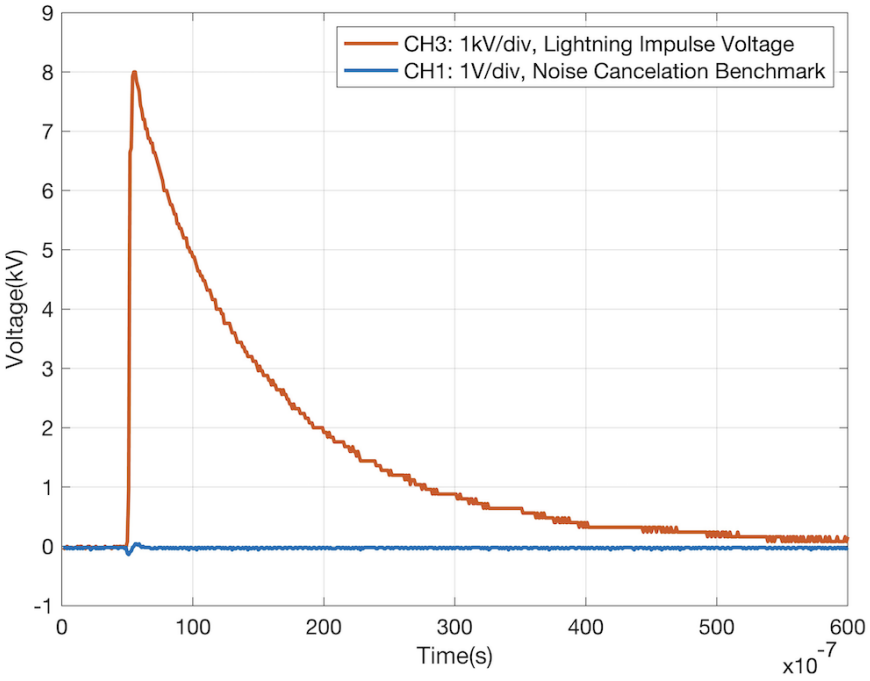


Figure 4.4: Applied impulse voltage waveform and balanced PD detection circuit output with no PD.

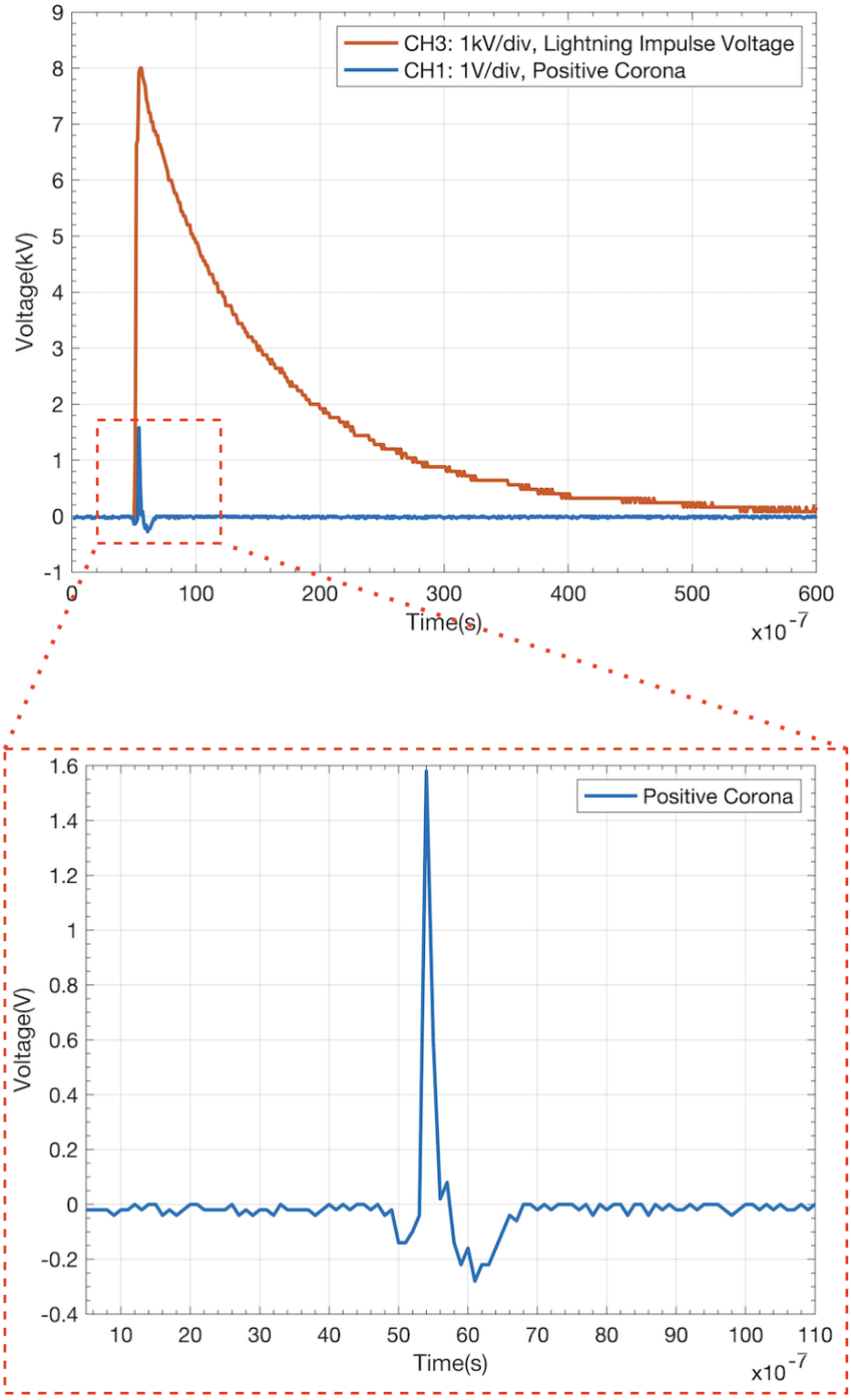


Figure 4.5: Impulse voltage short and positive corona in the air on the HV electrode. Dashed frame: is the zoomed positive corona pulse.

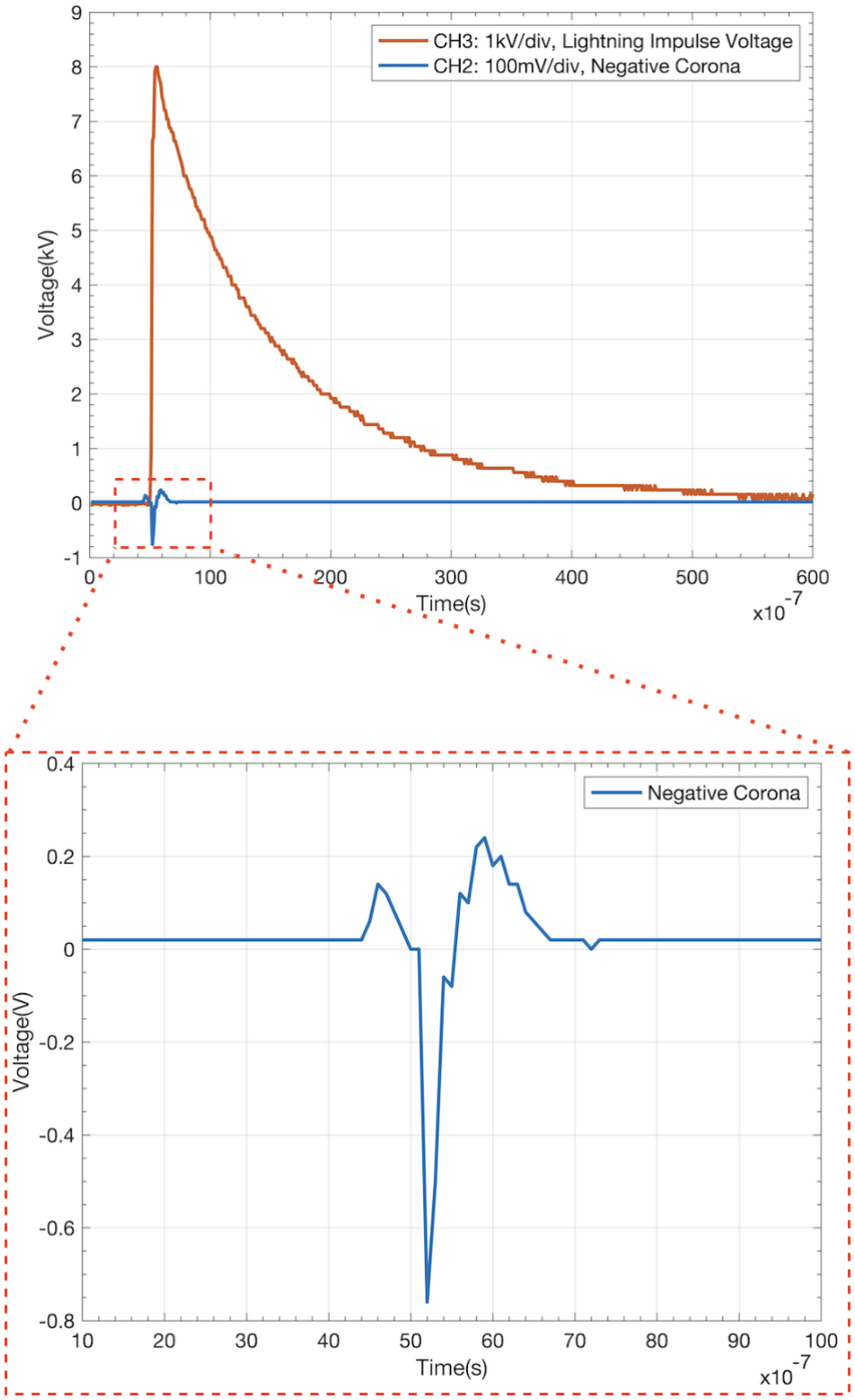


Figure 4.6: Impulse voltage short and negative corona in air on the earth electrode. Dashed frame: is the zoomed negative corona pulse.

4.3 Summary and pointers to the next chapter

The work has been presented, and with the implemented balanced PD detection circuit, cavity PD and corona characterization were achieved, and the key findings were discussed. The following chapter is intended to confirm the experimental results through MATLAB® Simulink® simulation, and the obtained experimental results are compared with the simulation results.

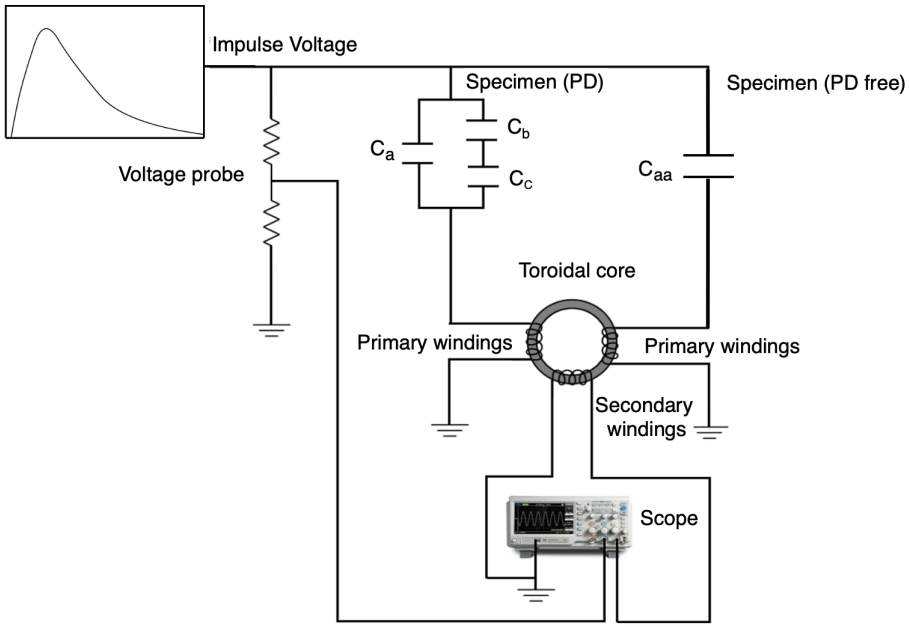
Chapter 5

Validation of Cavity PD and Corona model using MATLAB Simulink

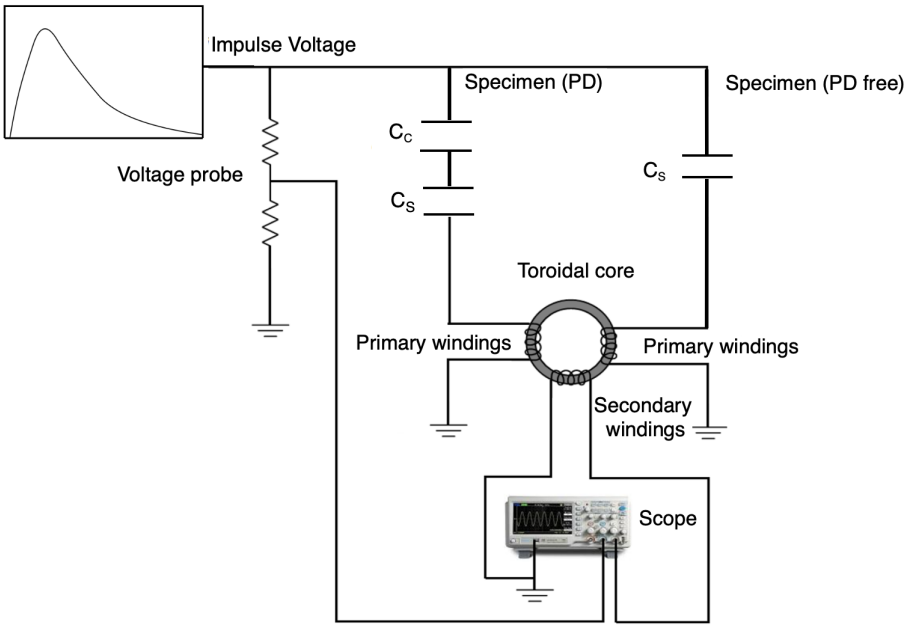
Cavity PD and corona defects have been modeled and simulated with MATLAB® Simulink® to study their characteristics under the standard lightning impulse voltage. This chapter presents the simulation procedure and results that correlate with the experimental results, which validate the experimental results.

5.1 Introduction

This section details the simulation of the cavity PD and corona under lightning impulse voltage using Matlab® Simulink®. The ABC capacitor model approach is adopted to model cavity-embedded and corona-defected insulation. The balance PD detection circuit model is explored, and it comprises PD and PD-free insulation models subjected to lightning impulse voltage. The simulation results confirm the measurement results. Figure 5.1 shows an equivalent circuit for balanced PD detection for (a) cavity PD and (b) corona [43].



(a)



(b)

Figure 5.1: Balanced PD detection equivalent circuit for (a) cavity PD and (b) corona.

The capacitors C_a, C_b, and C_c represent the cavity PD. The ABC model charac-

terizes the cavity-embedded insulation. C_a represents the capacitance of the rest of the healthy insulation, C_b represents the capacitance of the healthy insulation in series with the cavity, and C_c represents the capacitance of the cavity shorted by the discharge arc, and C_{aa} represent the insulation without PD of the cavity. C_c represents the capacitance of the region before the needle tip, which is short-circuited by the discharge arc in series with C_s. For PD-free corona insulation, C_s represents the capacitance of the area between the HV electrode and the ground electrode. The balanced PD detection circuit in a practical experiment consists of windings on the toroidal core used to measure PD. For simulation, the 50 Ω shunt resistor is used to measure the PD [44] through the comparator (filtering circuit). The voltage drop measured across the shunt resistor is connected to the comparator used as a signal cancellation circuit.

The single-stage impulse voltage generator is modeled according to standard IEC 60060-1 [14]. The output of the generator with parameter selection of $R_1 = 20\,\Omega$, $R_2 = 100\,\Omega$, $C_2 = 0.0125\,\mu F$, and $C_1 = 0.5\,\mu F$ is shown in Figure 5.2 below [45].

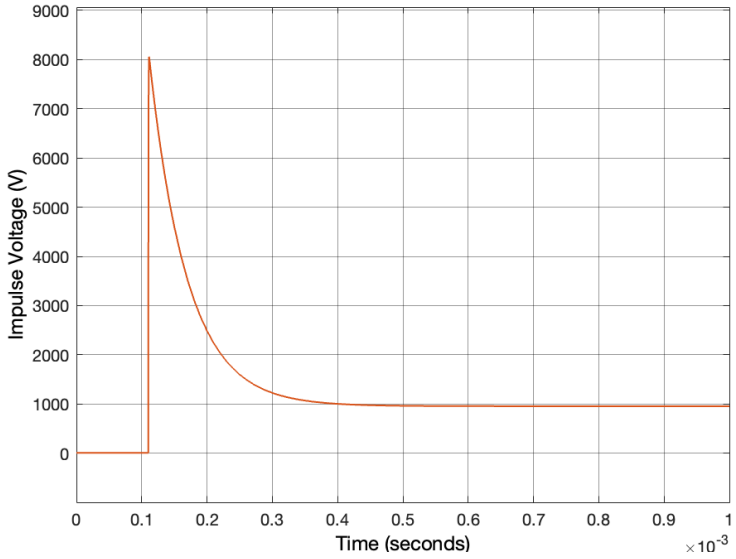


Figure 5.2: Simulated lightning impulse voltage waveform.

The magnitude of the simulated lightning impulse voltage waveform is 8.06 kV and has a front time of 1.942 µs and a tail time of 51.456 µs, which falls within the specification [14].

5.2 Cavity PD

5.2.1 Simulation values

Partial Discharge Inception Voltage (PDIV) is the voltage required to initiate PDs, and PDIV is calculated in section 3.2.1.1 to be 8.412 kV. The capacitor values are calculated to be $C_a = 15.48\,\text{pF}$, $C_b = 0.048\,\text{pF}$, $C_c = 0.042\,\text{pF}$ using the equation 5.1, 5.2 and 5.3 respectively [46].

$$C_a = \frac{\varepsilon_0 \varepsilon_r (a - 2r) b}{c} \tag{5.1}$$

$$C_b = \frac{\varepsilon_0 \varepsilon_r r^2 \pi}{c - h} \tag{5.2}$$

$$C_c = \frac{\varepsilon_0 r^2 \pi}{h} \tag{5.3}$$

Table 5.1 below shows the calculation parameter values.

Table 5.1: Parameter Values for cavity PD ABC model

Symbols and values		
Parameters	**Definition**	**Value**
a	Insulation length	60 mm
b	Insulation width	60 mm
c	Insulation thickness	4.5 mm
r	Radius of cavity	1.5 mm
h	Height of cavity	1.5 mm
ε_0	Free space permittivity	8.854×10^{-12}F/m
ε_r	Insulation relative permittivity	2.3

5.2.2 Seed electron and switch algorithm(PD mechanism)

Figure 5.3 below shows the implemented PD occurrence algorithm [47]. The algorithm proposed is for cavity PD and corona simulation under impulse voltage. For PD to occur, there are two prerequisites: the voltage across the cavity must be greater than or equal to a determined Paschen curve spark over a voltage of

4.5 kV, and there must be a free starting electron within the insulation to initiate an avalanche of electrons [48].

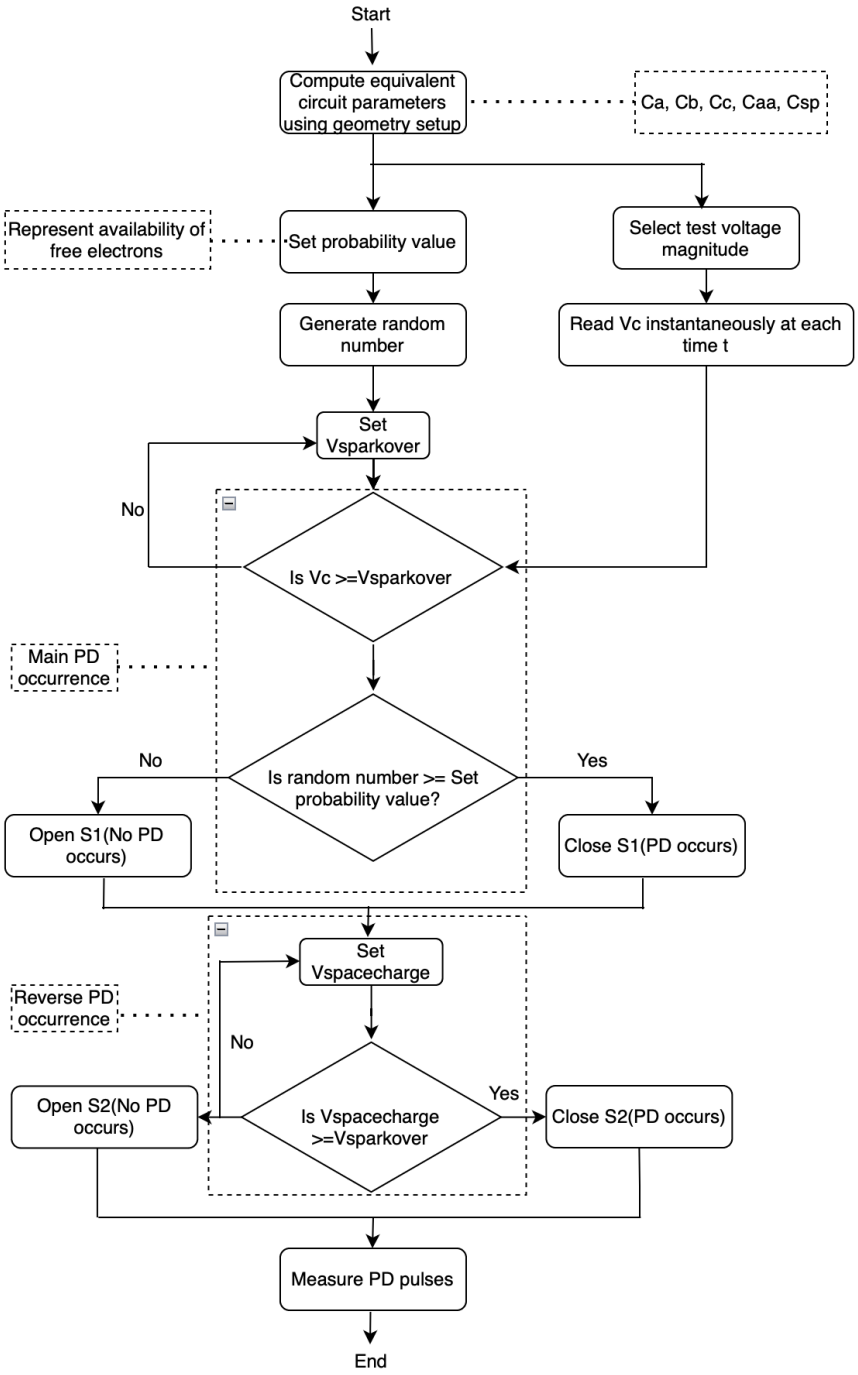


Figure 5.3: Cavity PD occurrence algorithm.

The implemented Matlab® Simulink® circuit for cavity PD under impulse voltage simulation is adopted and shown in Figure 5.4 [47]. The voltage in the cavity (between C_c) is constantly monitored to meet the voltage requirement of the set inception voltage. A random generator block generates a number compared to a set probability value. The switch is closed to simulate a PD when both prerequisites are met. The circuit also simulates space charges using a capacitor (C_{sp}) in series with a switch and the controlled voltage source that transmits the voltage. This allows simulation of space charges in the cavity [47].

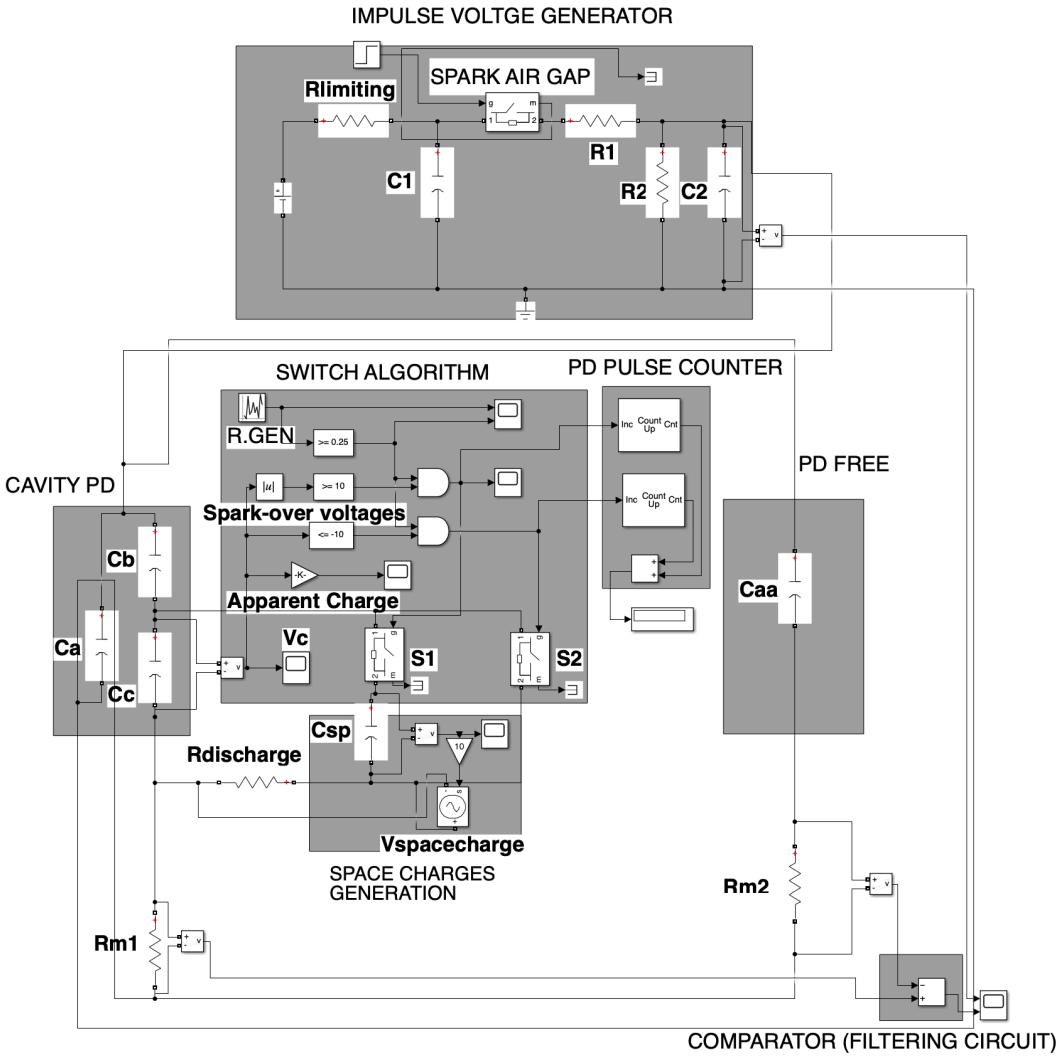


Figure 5.4: Model of cavity PD characterization under impulse voltage. Adopted from [47]

Table 5.2 below shows the parameter values used to simulate Figure 5.4.

Table 5.2: Parameter Values for cavity PD Simulink model

Symbols and values		
Parameters	**Definition**	**Value**
C_{sp}	Space charge capacitor	0.042 pF
R_{m1}	Measuring resistor	50 Ω
R_{m2}	Measuring resistor	50 Ω
C_{aa}	PD free insulation capacitor	0.128 pF

5.2.3 Cavity PD results

The PD pulse shown in Figure 5.5 is the detected output signal, and the occurrence also demonstrates the random behaviour due to the statistical time delay. Similarly to the experimental results, the magnitude of the main PD is higher than the magnitude of the reverse discharges, which is best explained by the space-charge decay phenomenon.

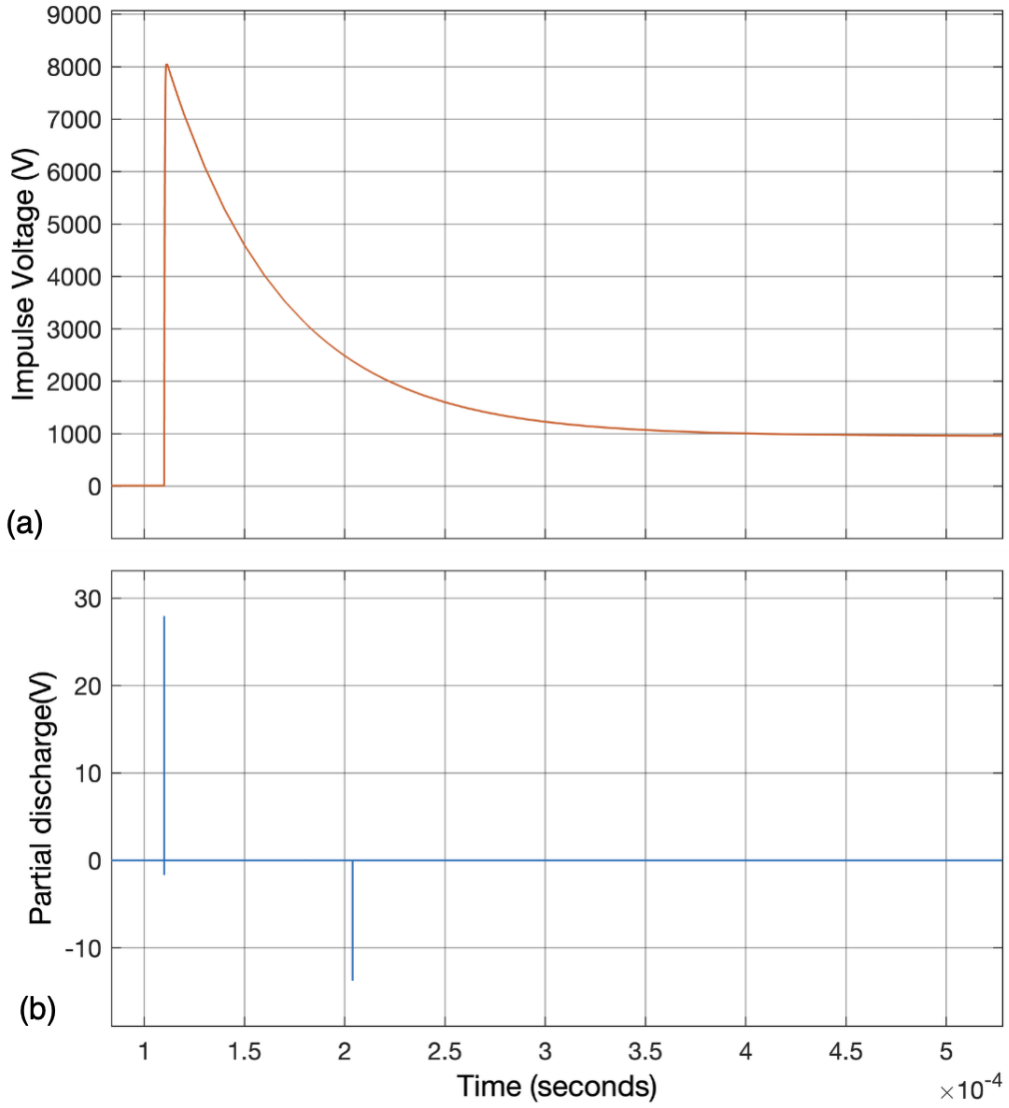


Figure 5.5: Simulated cavity PD pulses under impulse voltage. (a) Lightning impulse voltage, (b) Main PD and reverse discharge.

5.3 Corona

5.3.1 Simulation values

For the corona, in the beginning, free electrons in the air flow into neutral air molecules, creating positive ions and electrons [49]. The capacitor values are cal-

culated as $C_c = 0.0174\,\text{aF}$, $C_s = 0.0232\,\text{aF}$ using equations 5.4 and 5.5, respectively.

$$C_c = \frac{\varepsilon_0 \pi r_{point}^2}{L_d} \tag{5.4}$$

$$C_s = \frac{\varepsilon_0 \varepsilon_r \pi r_{point}^2}{L_{gap} - L_d} \tag{5.5}$$

Table 5.3 below shows the values of the calculation parameters [50, 51] in addition to Table 2.1.

Table 5.3: Parameter Values for corona model

Symbols and values		
Parameters	**Definition**	**Value**
r_{point}	Needle radius	5 µm
L_{gap}	Air gap length	10 mm
L_d	Length of conductive channel formed by corona	700 µm
C_c	capacitance of the region in front of the needle tip	0.993 aF
C_s	capacitance of the area between the HV electrode and the ground electrode	0.075 aF

5.3.2 Start-up free electron and switch algorithm (PD mechanism)

Figure 5.6 below is the implemented PD occurrence algorithm [47]. For simulating the model, the availability of free electrons can be modeled in two stages [48]: By setting the probability value that represents the availability of free electrons and having a random number between 0 and 1, generated by a random number function. The generated random number is compared to the set probability value for which it will indicate the availability of free electrons [47]. The value can be set as 0.25, 0.5, or 1.0 to suggest that 25 %, 50 %, or 100 % of free electrons are available.

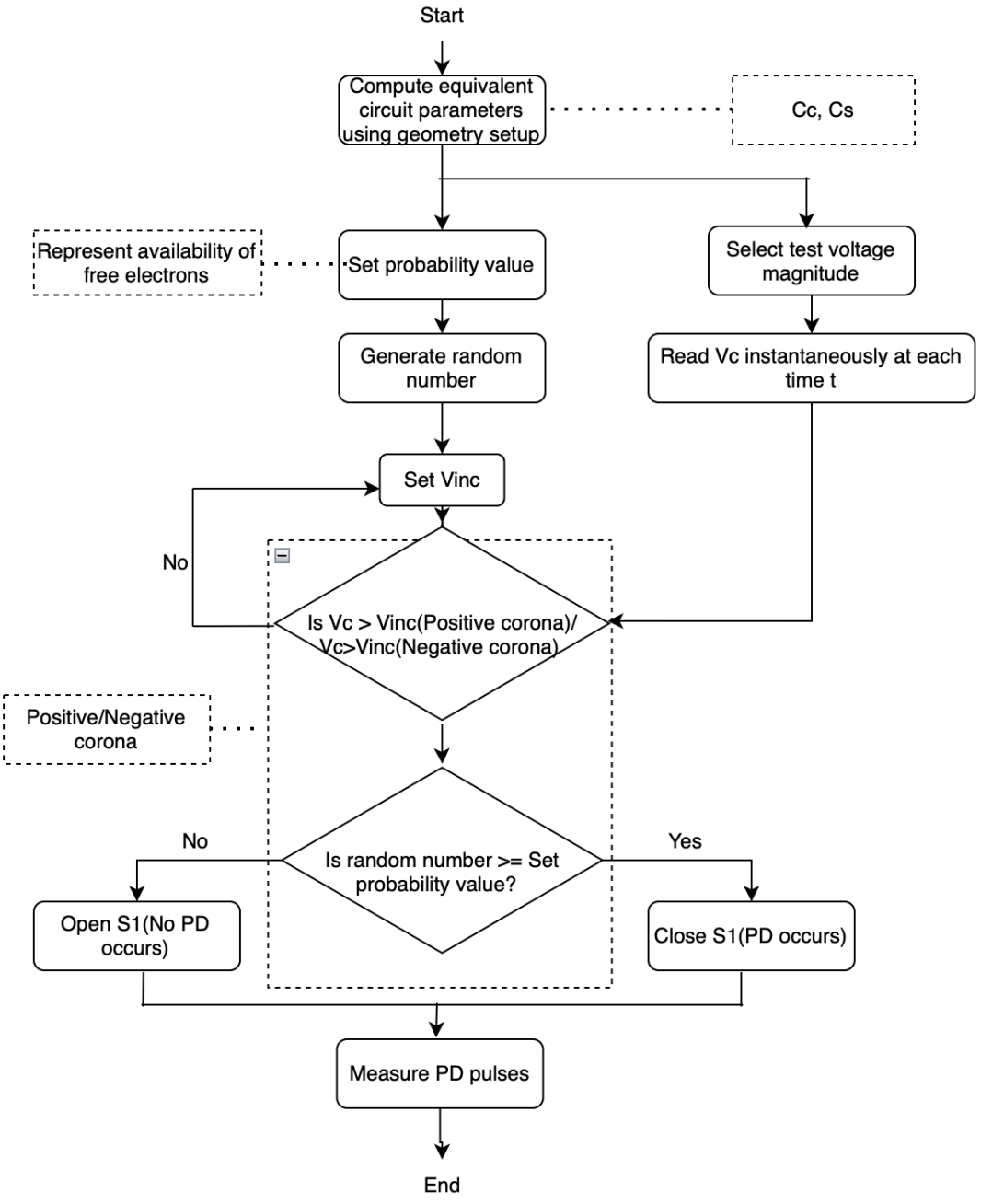


Figure 5.6: Corona PD occurrence algorithm.

Figure 5.7 shows that a switch is connected across the discharge capacitor (Cc) to simulate the inception voltage. The switch indication of ON means that there is a free electron for PD occurrence, and the switch indication of OFF implies that

there is no PD occurrence.

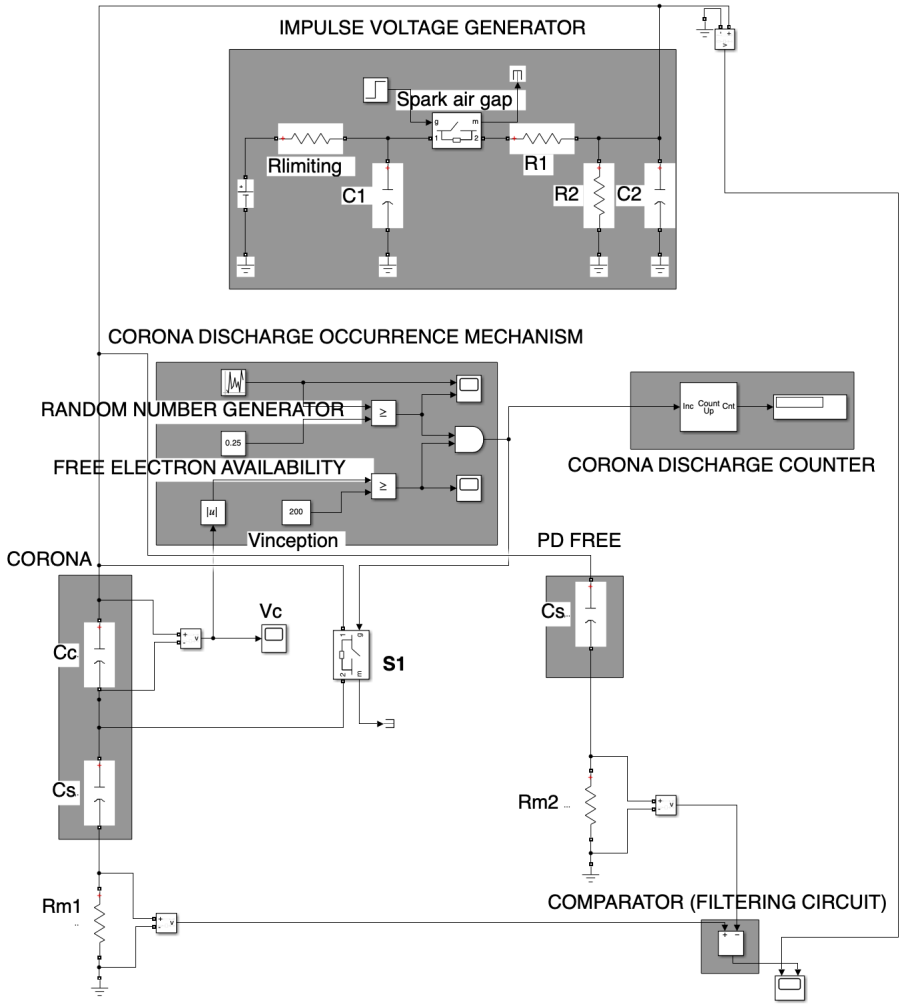


Figure 5.7: Model of corona characterization under impulse voltage. Adopted from [47]

5.3.3 Corona results

The PD pulse shown in Figure 5.8 is the output signal detected. Similarly to the experimental results, the magnitude of positive PDs is detected only during the rise time of the impulse voltage. The magnitude of the positive corona is always greater than that of the negative corona, as shown in Figure 5.9.

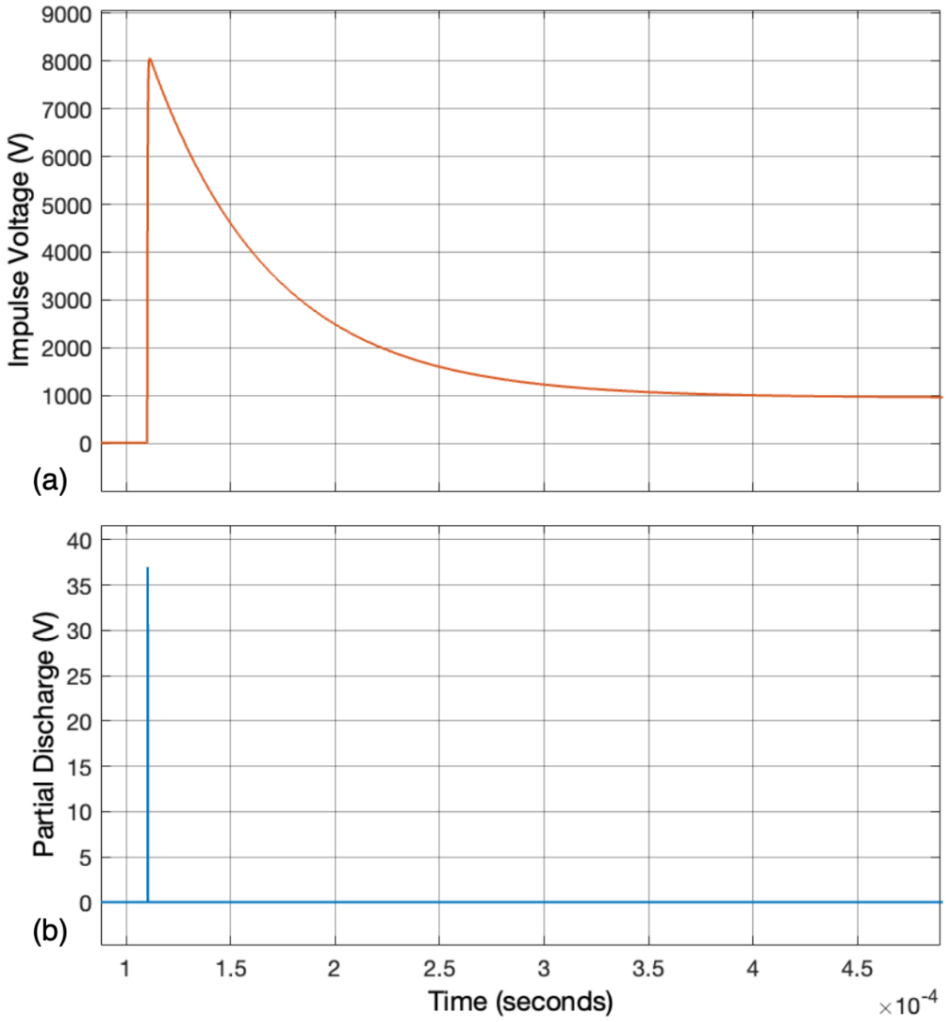


Figure 5.8: Simulated positive corona pulse under impulse voltage. (a) Lightning impulse voltage, (b) Positive corona pulse.

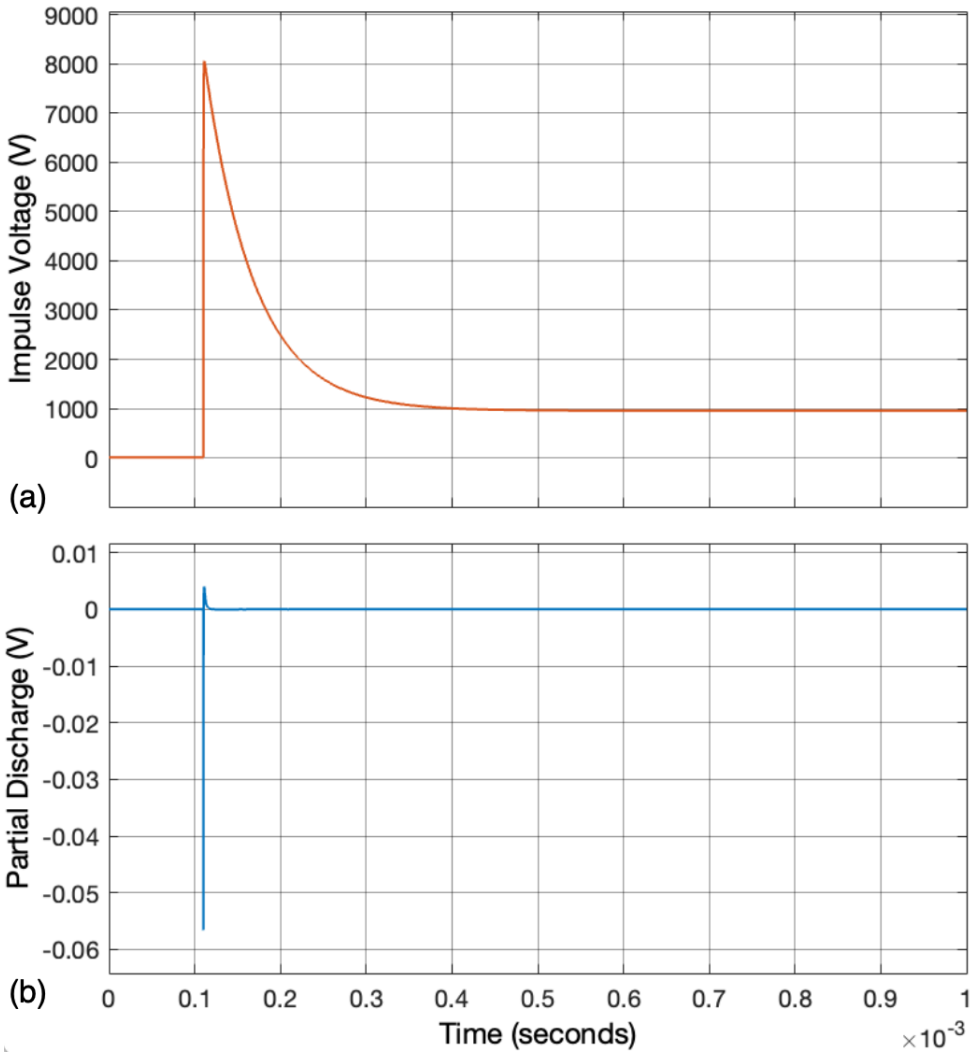


Figure 5.9: Simulated negative corona pulse under impulse voltage. (a) Lightning impulse voltage, (b) Negative corona pulse.

5.4 Comparison of Experimental and Simulated cavity PD and corona under LI voltage

The experimental and simulated cavity PD and corona discharge behaviour under LI voltages are very similar. The key findings noted:

- Both experimental and simulated cavity PD discharges have identical pulse positioning relative to the impulse voltage pulse, identical main discharge and reverse discharge magnitudes.
- Both experimental and simulated corona discharges have identical pulse positioning relative to the impulse voltage pulse, and similar positive and negative discharge magnitudes.
- The experimental cavity PD and corona discharge pulses have a ringing effect caused by reflections and resonances within the electrical circuit, which differs from the simulated smooth cavity PD and corona discharge pulses.

5.5 Summary and pointers to the next chapter

This chapter conducts simulations on cavity PD and corona under lightning impulse voltage, and their characteristics correlate with those obtained experimentally. The experimental results are reproduced with simulations, and a better understanding of the characterization of PD is obtained. The next chapter discusses the balanced PD detection circuit and the findings, followed by the conclusion and suggestions for future work.

Chapter 6

Conclusion

This work has delved into cavity PD and corona characteristics under LI voltage and analyzed the detected PD results under LI voltage. The key findings are:

- The role of the balanced PD detection circuit enabled PD detection under LI voltage application.
- The role of space charge in cavity PD enabled reverse PD pulses, which is not the case in corona.

The MATLAB® Simulink® model was used to simulate cavity PD and corona under LI voltage and reproduced the experimental measurement results. A comparison between the experimental and simulation data revealed a slight difference in the PD pulses due to the physical parameters. The results are as follows:

- The main PDs of the cavity under LI voltage occur near the voltage peak, and reverse PDs occur during the fall.
- The PDs of the positive corona occur only near the voltage peak, and the magnitude of the positive corona is higher than that of the main PDs of the cavity.
- The PDs of the negative corona under LI voltage occur near the voltage peak, and their magnitudes are smaller than those of the positive corona.

This work clearly explains how to characterize the cavity PD and corona under impulse voltage. The efficacy of the developed balanced PD detection circuit under LI impulse voltage deserves further study. The research has the potential to further extend knowledge in PD diagnosis by investigating the characterization of surface discharge defects under impulse voltages.

References

[1] Z. Sun, X. Zhao, J. Li, and Y. Li, "Experiment investigation of partial discharges under impulse voltage," in *2009 IEEE 9th International Conference on the Properties and Applications of Dielectric Materials*, 2009, pp. 525–529, DOI 10.1109/ICPADM.2009.5252374.

[2] C. Wadhwa, *High Voltage Engineering*. New Age International (P) Limited, 2006, Online at https://books.google.co.za/books?id=4rQu1M0sjRAC.

[3] G. Mitra, B. Salvage, and M. Sakr, "Detection and measurement of discharges in gaseous cavities in solid dielectrics under impulse voltage conditions," *Proc. IEE, Conference on Dielectric and Insulating Materials*, vol. 112, no. 5, pp. 1056–1060, 1965, DOI 10.1049/piee.1965.0180.

[4] V. Garcia-Colon, "Partial discharge measurement during impulse testing," in *IEEE Conference on Electrical Insulation and Dielectric Phenomena (CEIDP), Des Moines, Iowa, USA*, 2014.

[5] J. Wu, A. Rodrigo Mor, P. V. van Nes, and J. J. Smit, "Measuring method for partial discharges in a high voltage cable system subjected to impulse and superimposed voltage under laboratory conditions," *International Journal of Electrical Power & Energy Systems*, vol. 115, pp. 105–489, 2020, DOI https://doi.org/10.1016/j.ijepes.2019.105489.

[6] R. J. Densley and B. Salvage, "Partial discharges in gaseous cavities in solid dielectrics under impulse voltage conditions," *IEEE Transactions on Electrical Insulation*, vol. EI-6, no. 2, pp. 54–62, 1971, DOI 10.1109/TEI.1971.299155.

[7] IEC, *High-voltage test techniques - partial discharge measurements, (IEC 60270:2000).* IEC, 2015.

[8] J. Wu, A. Rodrigo Mor, and J. J. Smit, "The effects of superimposed impulse transients on partial discharge in XLPE cable joint," *International Journal of Electrical Power & Energy Systems*, vol. 110, pp. 497–509, 2019, DOI https://doi.org/10.1016/j.ijepes.2019.03.031.

[9] IEC, *High voltage test techniques-Measurements of partial discharge by electromagnetic and acoustic methods, (IEC 62478:2016).* IEC, 2016.

[10] G. W. Bowdler, *Measurements in high-voltage test circuits.* Pergamon Press, 1973, Online at https://nla.gov.au/nla.cat-vn1973337.

[11] F. Kreuger, *Discharge Detection in High Voltage Equipment.* A Heywood Book, 1964.

[12] R. Bartnikas, "Partial discharges. their mechanism, detection, and measurement," *IEEE Transactions on Dielectrics and Electrical Insulation*, vol. 9, no. 5, pp. 763–808, 2002, DOI 10.1109/TDEI.2002.1038663.

[13] K. Schon, *High Impulse Voltage and Current Measurement Techniques: Fundamentals – Measuring Instruments – Measuring Methods.* Springer, 08 2013, DOI 10.1007/978-3-319-00378-8.

[14] IEC, *High-voltage test techniques - Part 1: General definitions and test requirements, (IEC 60060-1:2010).* IEC, 2010.

[15] E. Kuffel, *High voltage engineering: fundamentals.* Oxford ; Boston : Butterworth-Heinemann, 2000.

[16] IEC, *High-voltage test techniques - Part 3: Definitions and requirements for on-site testing, (IEC 60060-3:2006).* IEC, 2006.

[17] L. Zhang, H. Xutao, and L. Junhao, "Partial discharge detection and analysis of needle-plane defect in SF_6 under negative oscillating lightning impulse voltage based on uhf method," *IEEE Transactions on Dielectrics and*

Electrical Insulation, vol. 24, pp. 296–303, 02 2017, DOI 10.1109/TDEI.2016.005979.

[18] K. Backhaus, L. Elspaß, and K. Pasche, "Detection technique of partial discharges at impulse voltage with appropriate filter settings for signal separation," *Energies*, vol. 12, no. 23, 2019, DOI 10.3390/en12234445.

[19] Y. Li, H.-B. Mu, J. Deng, G.-J. Zhang, and S.-H. Wang, "Partial discharge characteristics of oil/polypropylene film with a needle-plate electrode excited by impulse voltages," in *2013 Annual report conference on electrical insulation and dielectric phenomena*, Oct. 2013, pp. 1225–1228, DOI 10.1109/CEIDP.2013.6748256.

[20] N. Hayakawa, R. Yamaguchi, Y. Ukai, H. Kojima, F. Endo, and H. Okubo, "Partial discharge activities under ac/impulse superimposed voltage in LN2/polypropylene laminated paper insulation system for hts cables," *Journal of Physics: Conference Series*, vol. 234, p. 032020, 07 2010, DOI 10.1088/1742-6596/234/3/032020.

[21] L. Junhao, L. Zhang, J. Liang, X. Yao, and Y. Li, "Partial discharge characteristics over SF_6/epoxy interfaces under impulse voltage," *Dielectrics and Electrical Insulation, IEEE Transactions on*, vol. 20, pp. 2158–2164, Dec. 2013, DOI 10.1109/TDEI.2013.6678865.

[22] L. Gang, W. Guangning, and T. Demin, "The system architecture of selective frequency balance measurement for PD," in *Conference Record of the 2006 IEEE International Symposium on Electrical Insulation*, 2006, pp. 151–154, DOI 10.1109/ELINSL.2006.1665279.

[23] F. Kreuger, "Partial discharge measurement on three-core belted power cables," *IEEE Transactions on Power Delivery*, vol. 4, no. 2, pp. 927–931, 1989, DOI 10.1109/61.25571.

[24] S. Sitole, (2020) Email to Tibello Mphuti, 31 March.

[25] P. Morshuis and J. Smit, "Partial discharges at DC voltage: their mechanism, detection and analysis," *IEEE Transactions on Dielectrics and Electrical In-*

sulation, vol. 12, no. 2, pp. 328–340, 2005, DOI 10.1109/TDEI.2005.1430401.

[26] G. Callender and P. Lewin, "Modeling partial discharge phenomena," *IEEE Electrical Insulation Magazine*, vol. 36, no. 2, pp. 29–36, 2020.

[27] Y. Hasegawa, Y. Ohki, K. Fukunaga, M. Mizuno, and K. Sasaki, "Complex permittivity spectra of various insulating polymers at ultrawide-band frequencies," *Electrical Engineering in Japan*, vol. 198, no. 3, pp. 11–18, 2017.

[28] S. Z. Ahmed Dabbak, H. A. Illias, B. C. Ang, N. A. Abdul Latiff, and M. Z. H. Makmud, "Electrical properties of polyethylene/polypropylene compounds for high-voltage insulation," *Energies*, vol. 11, no. 6, p. 1448, 2018.

[29] T. Gora, "Investigating the effects of altitude (air density) on the HVDC breakdown voltage of small rod-plane air gaps," Master's thesis, University of the Witwatersrand, Johannesburg, 2016.

[30] M. K. Khangale, "A study of the effect of temperature on cavity partial discharges in polyethylene (PE) insulation," Master's thesis, University of the Witwatersrand, Johannesburg, 2024.

[31] Q. Zhang, J. Zhu, J. Jia, F. Tao, and L. Yang, "Design of a current transducer with a magnetic core for use in measurements of nanosecond current pulses," *Measurement Science and Technology*, vol. 17, p. 895, 03 2006, DOI 10.1088/0957-0233/17/4/039.

[32] X. Zhao, X. Yao, Z. Guo, Y. Wang, L. Junhao, and Y. Li, "Partial discharge characteristics and mechanism in voids at impulse voltages," *Measurement Science and Technology*, vol. 22, p. 035704, 02 2011, DOI 10.1088/0957-0233/22/3/035704.

[33] C. Karagiannopoulos, P. Bourkas, D. Agoris, and C. Psomopoulos, "Free electron energy during the aging of an organic dielectric under switching impulse voltages," in *Conference Record of the 1998 IEEE International Symposium on Electrical Insulation (Cat. No.98CH36239)*, vol. 2, 1998, pp. 499–502 vol.2, DOI 10.1109/ELINSL.1998.694842.

[34] L. Gui, Y. Mi, S. Deng, L. Liu, X. Ge, and W. Ouyang, "Partial discharge characteristics of an air gap defect in the epoxy resin of a saturable reactor under an exponential decay pulse voltage," *High Voltage*, vol. 5, 08 2020, DOI 10.1049/hve.2019.0121.

[35] Y. Xie, H. He, J. He, and C. Wu, "The effect of corona discharge on leader initiation in long air gaps," *Plasma Science, IEEE Transactions on*, vol. 42, pp. 890–895, 04 2014, DOI 10.1109/TPS.2014.2305446.

[36] L. Junhao, D. Hu, L. Zhang, and B. Che, "Dielectric breakdown characteristics of the rod-plane electrode system in SF_6 gas under oscillating impulse voltage," in *2016 IEEE International Power Modulator and High Voltage Conference (IPMHVC)*, 07 2016, pp. 38–311, DOI 10.1109/IPMHVC.2016.8012845.

[37] W. Wang, P. Wang, W. Gong, L. Wang, and Q. Li, "Characteristics and mechanism of ac-dc composite partial discharge of aging oil-paper insulation under switching impulse voltage in extremely uneven electric field," in *2020 IEEE International Conference on High Voltage Engineering and Application (ICHVE)*, 2020, pp. 1–4, DOI 10.1109/ICHVE49031.2020.9279555.

[38] R. Prochazka, "Simulation model of flashover process on insulator string exposed to the impulse voltages," in *2016 Conference on Diagnostics in Electrical Engineering (Diagnostika)*, 2016, pp. 1–4, DOI 10.1109/DIAGNOSTIKA.2016.7736485.

[39] D. Han, T. Lin, G. Zhang, Y. Liu, and Q. Yu, "SF_6 gas decomposition analysis under point-to-plane 50 Hz ac corona discharge," *IEEE Transactions on Dielectrics and Electrical Insulation*, vol. 22, no. 2, pp. 799–805, 2015, DOI 10.1109/TDEI.2015.7076778.

[40] X. Zhao, L. Pu, S. Ren, Z. Ju, and W. Zhao, "Partial discharge characteristics and mechanism in SF_6 gas under oscillating impulse voltages," in *2016 International Conference on Condition Monitoring and Diagnosis (CMD)*, 2016, pp. 839–842, DOI 10.1109/CMD.2016.7758000.

[41] P. Bunme, N. Khamsen, V. Kasemsuwan, K. Jitkajornwanich, A. Pichetjamroen, N. Teerakawanich, and S. Srisonphan, "Polarity effect of pulsed corona discharge plasma on seed surface modification," in *2017 International Electrical Engineering Congress (iEECON)*, 2017, pp. 1–4, DOI 10.1109/IEECON.2017.8075754.

[42] E. I. Bousiou and P. N. Mikropoulos, "Experimental investigation on corona charge-voltage characteristics in the coaxial configuration under lightning impulse voltages," in *2018 IEEE International Conference on High Voltage Engineering and Application (ICHVE)*, 2018, pp. 1–4, DOI 10.1109/ICHVE.2018.8641902.

[43] O. Gouda and A. El-Faraskoury, "Partial discharge measurements with internal artificial cavities defects for underground cables," in *Conference: IEEE-16th International Middle- East Power Systems Conference -MEPCON'2014 Ain Shams University, Cairo, Egypt, December 23 - 25, 2014At: Cairo*, Dec. 2014, DOI 10.13140/2.1.2027.8567.

[44] J. Wu, A. Rodrigo Mor, and J. J. Smit, "Partial discharges activated by impulses and superimposed voltages in a high voltage cable model," *International Journal of Electrical Power & Energy Systems*, vol. 120, p. 106027, 2020, DOI https://doi.org/10.1016/j.ijepes.2020.106027.

[45] P. Ghosh, B. Chakraborty, S. Dalai, and S. Chatterjee, "Simulation and real-time generation of non-standard lightning impulse voltage waveforms," in *2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, 2022, pp. 1–5, DOI 10.1109/ICDCECE53908.2022.9792719.

[46] A. Sabat and S. Karmakar, "Simulation of partial discharge in high voltage power equipment," *International Journal on Electrical Engineering and Informatics*, vol. 3, pp. 234–247, 06 2011, DOI 10.15676/ijeei.2011.3.2.8.

[47] C. N. Kgotso Hlapolosa, Victor Madonsela, "Simulating cavity and corona partial discharge (PD) under the impulse voltage. focus on cavity simulations

& simulation of cavity and corona partial discharges under lightning impulse voltage," *4th year project*, 2022.

[48] E. Haikali and C. Nyamupangedengu, "Measured and simulated time-evolution PD characteristics of typical installation defects in MV XLPE cable terminations," *SAIEE Africa Research Journal*, vol. 110, pp. 136–144, 09 2019, DOI 10.23919/SAIEE.2019.8732785.

[49] C. Wang, X. Chen, J. Ouyang, T. Li, and J. Fu, "Pulse current of multi-needle negative corona discharge and its electromagnetic radiation characteristics," *Energies*, vol. 11, no. 11, 2018, DOI 10.3390/en11113120.

[50] M. Gunawardana, A. Kanchana, P. Wijesingha, H. Perera, R. Samarasinghe, and R. Lucas, "A Matlab Simulink model for a partial discharge measuring system," *EECon*, 01 2015, Online at http://dl.lib.uom.lk/handle/123/20383.

[51] Y. Wang, Z. Huang, X. Zhang, and J. Shang, "Characteristics of electrical tree morphological in glass fibre reinforced epoxy resin under power frequency voltage," *IET Science, Measurement & Technology*, vol. 16, no. 4, pp. 274–282, 2022, DOI https://doi.org/10.1049/smt2.12102.

Appendix A

C code

A.1 C code used to simulate cavity PD and corona under lightning impulse voltage

The C code used to simulate the cavity PD and corona characteristics is given below .

Listing A.1: Cavity PD

```
%This code performs a simulation of cavity PD

/* Block signals (default storage) */
B_CavityPD_T CavityPD_B;

/* Continuous states */
X_CavityPD_T CavityPD_X;

/* Disabled State Vector */
XDis_CavityPD_T CavityPD_XDis;

/* Block states (default storage) */
DW_CavityPD_T CavityPD_DW;

/* Real-time model */
static RT_MODEL_CavityPD_T CavityPD_M_;
RT_MODEL_CavityPD_T *const CavityPD_M = &CavityPD_M_;
static void rate_scheduler(void);

/*
```

```
 *         This function updates active task flag for each subrate.
 *         The function is called at model base rate, hence the
 *         generated code self-manages all its subrates.
 */
static void rate_scheduler(void)
{
  /* Compute which subrates run during the next base time step.
     Subrates
   * are an integer multiple of the base rate counter.  Therefore,
     the subtask
   * counter is reset when it reaches its limit (zero means run).
   */
  (CavityPD_M->Timing.TaskCounters.TID[2])++;
  if ((CavityPD_M->Timing.TaskCounters.TID[2]) > 9) {/* Sample time:
     [0.0001s, 0.0s] */
    CavityPD_M->Timing.TaskCounters.TID[2] = 0;
  }

  CavityPD_M->Timing.sampleHits[2] = (CavityPD_M->Timing.
    TaskCounters.TID[2] ==
    0) ? 1 : 0;
}

/* Projection for root system: '<Root>' */
void CavityPD_projection(void)
{
  /* Projection for S-Function (sfun_spid_contc): '<S30>/State-Space
     ' */
  /* Level2 S-Function Block: '<S30>/State-Space' (sfun_spid_contc)
     */
  {
    SimStruct *rts = CavityPD_M->childSfunctions[0];
    sfcnProjection(rts);
    if (ssGetErrorStatus(rts) != (NULL))
      return;
  }
}

/*
 * This function updates continuous states using the ODE3 fixed-step
 * solver algorithm
 */
static void rt_ertODEUpdateContinuousStates(RTWSolverInfo *si )
{
  /* Solver Matrices */
  static const real_T rt_ODE3_A[3] = {
    1.0/2.0, 3.0/4.0, 1.0
  };

  static const real_T rt_ODE3_B[3][3] = {
    { 1.0/2.0, 0.0, 0.0 },
```

```
    { 0.0, 3.0/4.0, 0.0 },

    { 2.0/9.0, 1.0/3.0, 4.0/9.0 }
};

time_T t = rtsiGetT(si);
time_T tnew = rtsiGetSolverStopTime(si);
time_T h = rtsiGetStepSize(si);
real_T *x = rtsiGetContStates(si);
ODE3_IntgData *id = (ODE3_IntgData *)rtsiGetSolverData(si);
real_T *y = id->y;
real_T *f0 = id->f[0];
real_T *f1 = id->f[1];
real_T *f2 = id->f[2];
real_T hB[3];
int_T i;
int_T nXc = 8;
rtsiSetSimTimeStep(si,MINOR_TIME_STEP);

/* Save the state values at time t in y, we'll use x as ynew. */
(void) memcpy(y, x,
              (uint_T)nXc*sizeof(real_T));

/* Assumes that rtsiSetT and ModelOutputs are up-to-date */
/* f0 = f(t,y) */
rtsiSetdX(si, f0);
CavityPD_derivatives();

/* f(:,2) = feval(odefile, t + hA(1), y + f*hB(:,1), args(:)(*)); */
hB[0] = h * rt_ODE3_B[0][0];
for (i = 0; i < nXc; i++) {
  x[i] = y[i] + (f0[i]*hB[0]);
}

rtsiSetT(si, t + h*rt_ODE3_A[0]);
rtsiSetdX(si, f1);
CavityPD_step();
CavityPD_derivatives();

/* f(:,3) = feval(odefile, t + hA(2), y + f*hB(:,2), args(:)(*)); */
for (i = 0; i <= 1; i++) {
  hB[i] = h * rt_ODE3_B[1][i];
}

for (i = 0; i < nXc; i++) {
  x[i] = y[i] + (f0[i]*hB[0] + f1[i]*hB[1]);
}

rtsiSetT(si, t + h*rt_ODE3_A[1]);
rtsiSetdX(si, f2);
```

```
  CavityPD_step();
  CavityPD_derivatives();

  /* tnew = t + hA(3);
     ynew = y + f*hB(:,3); */
  for (i = 0; i <= 2; i++) {
    hB[i] = h * rt_ODE3_B[2][i];
  }

  for (i = 0; i < nXc; i++) {
    x[i] = y[i] + (f0[i]*hB[0] + f1[i]*hB[1] + f2[i]*hB[2]);
  }

  rtsiSetT(si, tnew);
  CavityPD_step();
  CavityPD_projection();
  rtsiSetSimTimeStep(si,MAJOR_TIME_STEP);
}

real_T rt_urand_Upu32_Yd_f_pw_snf(uint32_T *u)
{
  uint32_T hi;
  uint32_T lo;

  /* Uniform random number generator (random number between 0 and 1)

     #define IA      16807                      magic multiplier =
         7^5
     #define IM      2147483647                 modulus = 2^31-1
     #define IQ      127773                     IM div IA
     #define IR      2836                       IM modulo IA
     #define S       4.656612875245797e-10      reciprocal of 2^31-1
     test = IA * (seed % IQ) - IR * (seed/IQ)
     seed = test < 0 ? (test + IM) : test
     return (seed*S)
   */
  lo = *u % 127773U * 16807U;
  hi = *u / 127773U * 2836U;
  if (lo < hi) {
    *u = 2147483647U - (hi - lo);
  } else {
    *u = lo - hi;
  }

  return (real_T)*u * 4.6566128752457969E-10;
}

/* Model step function */
void CavityPD_step(void)
{
  if (rtmIsMajorTimeStep(CavityPD_M)) {
    /* set solver stop time */
```

```
    if (!(CavityPD_M->Timing.clockTick0+1)) {
      rtsiSetSolverStopTime(&CavityPD_M->solverInfo,
                            ((CavityPD_M->Timing.clockTickH0 + 1) *
        CavityPD_M->Timing.stepSize0 * 4294967296.0));
    } else {
      rtsiSetSolverStopTime(&CavityPD_M->solverInfo,
                            ((CavityPD_M->Timing.clockTick0 + 1) *
        CavityPD_M->Timing.stepSize0 + CavityPD_M->Timing.
           clockTickH0 *
        CavityPD_M->Timing.stepSize0 * 4294967296.0));
    }
  }                                     /* end MajorTimeStep */

  /* Update absolute time of base rate at minor time step */
  if (rtmIsMinorTimeStep(CavityPD_M)) {
    CavityPD_M->Timing.t[0] = rtsiGetT(&CavityPD_M->solverInfo);
  }

  {
    real_T currentTime;
    if (rtmIsMajorTimeStep(CavityPD_M) &&
        CavityPD_M->Timing.TaskCounters.TID[1] == 0) {
      /* Constant: '<S17>/DC' */
      CavityPD_B.DC = CavityPD_P.DCVoltageSource1_Amplitude;
    }

    /* S-Function (sfun_spid_contc): '<S30>/State-Space' */

    /* Level2 S-Function Block: '<S30>/State-Space' (sfun_spid_contc
       ) */
    {
      SimStruct *rts = CavityPD_M->childSfunctions[0];
      sfcnOutputs(rts,0);
    }

    /* Gain: '<S12>/do not delete this gain' */
    CavityPD_B.donotdeletethisgain = CavityPD_P.
       donotdeletethisgain_Gain *
      CavityPD_B.StateSpace_o1[3];
    if (rtmIsMajorTimeStep(CavityPD_M) &&
        CavityPD_M->Timing.TaskCounters.TID[1] == 0) {
      /* Scope: '<Root>/Scope' */
      if (rtmIsMajorTimeStep(CavityPD_M)) {
        StructLogVar *svar = (StructLogVar *)CavityPD_DW.Scope_PWORK
           .LoggedData;
        LogVar *var = svar->signals.values;

        /* signals */
        {
          real_T up0[1];
          up0[0] = CavityPD_B.donotdeletethisgain;
          rt_UpdateLogVar((LogVar *)var, up0, 0);
```

```
      }
    }

    /* UniformRandomNumber: '<Root>/Uniform Random Number' */
    CavityPD_B.UniformRandomNumber =
      CavityPD_DW.UniformRandomNumber_NextOutput;

    /* RelationalOperator: '<S2>/Compare' incorporates:
     *  Constant: '<S2>/Constant'
     */
    CavityPD_B.Compare = (CavityPD_B.UniformRandomNumber >=
                          CavityPD_P.CompareToConstant1_const);
  }

  /* Abs: '<Root>/Abs' */
  CavityPD_B.Abs = fabs(CavityPD_B.donotdeletethisgain);

  /* RelationalOperator: '<S1>/Compare' incorporates:
   *  Constant: '<S1>/Constant'
   */
  CavityPD_B.Compare_k = (CavityPD_B.Abs >= CavityPD_P.
    CompareToConstant_const);

  /* Logic: '<Root>/AND' */
  CavityPD_B.AND = (CavityPD_B.Compare && CavityPD_B.Compare_k);
  if (rtmIsMajorTimeStep(CavityPD_M) &&
      CavityPD_M->Timing.TaskCounters.TID[1] == 0) {
  }

  /* Gain: '<Root>/Multiply' */
  CavityPD_B.Multiply = CavityPD_P.Multiply_Gain *
    CavityPD_B.donotdeletethisgain;
  if (rtmIsMajorTimeStep(CavityPD_M) &&
      CavityPD_M->Timing.TaskCounters.TID[1] == 0) {
  }

  /* RelationalOperator: '<S3>/Compare' incorporates:
   *  Constant: '<S3>/Constant'
   */
  CavityPD_B.Compare_a = (CavityPD_B.donotdeletethisgain <=
    CavityPD_P.CompareToConstant2_const);

  /* Logic: '<Root>/AND1' */
  CavityPD_B.AND1 = (CavityPD_B.Compare && CavityPD_B.Compare_a);

  /* DataTypeConversion: '<S18>/Data Type Conversion' */
  CavityPD_B.DataTypeConversion = CavityPD_B.AND;

  /* DataTypeConversion: '<S22>/Data Type Conversion' */
  CavityPD_B.DataTypeConversion_h = CavityPD_B.AND1;

  /* Gain: '<S13>/do not delete this gain' */
```

```
    CavityPD_B.donotdeletethisgain_p = CavityPD_P.
       donotdeletethisgain_Gain_a *
      CavityPD_B.StateSpace_o1[4];

    /* Gain: '<S9>/do not delete this gain' */
    CavityPD_B.donotdeletethisgain_a = CavityPD_P.
       donotdeletethisgain_Gain_d *
      CavityPD_B.StateSpace_o1[0];

    /* Sum: '<Root>/Balanced detection' */
    CavityPD_B.Balanceddetection = CavityPD_B.donotdeletethisgain_a
       -
      CavityPD_B.donotdeletethisgain_p;

    /* Gain: '<S11>/do not delete this gain' */
    CavityPD_B.donotdeletethisgain_d = CavityPD_P.
       donotdeletethisgain_Gain_k *
      CavityPD_B.StateSpace_o1[2];
    if (rtmIsMajorTimeStep(CavityPD_M) &&
        CavityPD_M->Timing.TaskCounters.TID[1] == 0) {
    }

    /* Gain: '<S4>/do not delete this gain' */
    CavityPD_B.donotdeletethisgain_k = CavityPD_P.
       donotdeletethisgain_Gain_o *
      CavityPD_B.StateSpace_o1[6];
    if (rtmIsMajorTimeStep(CavityPD_M) &&
        CavityPD_M->Timing.TaskCounters.TID[1] == 0) {
    }

    if (rtmIsMajorTimeStep(CavityPD_M) &&
        CavityPD_M->Timing.TaskCounters.TID[2] == 0) {
      /* Step: '<Root>/Step' */
      currentTime = (((CavityPD_M->Timing.clockTick2+
                       CavityPD_M->Timing.clockTickH2* 4294967296.0)
                          ) * 0.0001);
      if (currentTime < CavityPD_P.Step_Time) {
        /* Step: '<Root>/Step' */
        CavityPD_B.Step = CavityPD_P.Step_Y0;
      } else {
        /* Step: '<Root>/Step' */
        CavityPD_B.Step = CavityPD_P.Step_YFinal;
      }

      /* End of Step: '<Root>/Step' */
    }

    /* Gain: '<S10>/do not delete this gain' */
    CavityPD_B.donotdeletethisgain_j = CavityPD_P.
       donotdeletethisgain_Gain_c *
      CavityPD_B.StateSpace_o1[1];
    if (rtmIsMajorTimeStep(CavityPD_M) &&
```

```
        CavityPD_M->Timing.TaskCounters.TID[1] == 0) {
    }

    /* Gain: '<S14>/do not delete this gain' */
    CavityPD_B.donotdeletethisgain_m = CavityPD_P.
       donotdeletethisgain_Gain_c1 *
      CavityPD_B.StateSpace_o1[5];
    if (rtmIsMajorTimeStep(CavityPD_M) &&
        CavityPD_M->Timing.TaskCounters.TID[1] == 0) {
    }
  }

  if (rtmIsMajorTimeStep(CavityPD_M)) {
    /* Matfile logging */
    rt_UpdateTXYLogVars(CavityPD_M->rtwLogInfo, (CavityPD_M->Timing.
       t));
  }                                    /* end MajorTimeStep */

  if (rtmIsMajorTimeStep(CavityPD_M)) {
    real_T tmin;

    /* Update for S-Function (sfun_spid_contc): '<S30>/State-Space'
       */
    /* Level2 S-Function Block: '<S30>/State-Space' (sfun_spid_contc
       ) */
    {
      SimStruct *rts = CavityPD_M->childSfunctions[0];
      sfcnUpdate(rts,0);
      if (ssGetErrorStatus(rts) != (NULL))
        return;
    }

    if (rtmIsMajorTimeStep(CavityPD_M) &&
        CavityPD_M->Timing.TaskCounters.TID[1] == 0) {
      /* Update for UniformRandomNumber: '<Root>/Uniform Random
         Number' */
      tmin = CavityPD_P.UniformRandomNumber_Minimum;
      CavityPD_DW.UniformRandomNumber_NextOutput =
        (CavityPD_P.UniformRandomNumber_Maximum - tmin) *
        rt_urand_Upu32_Yd_f_pw_snf(&CavityPD_DW.RandSeed) + tmin;
    }
  }                                    /* end MajorTimeStep */

  if (rtmIsMajorTimeStep(CavityPD_M)) {
    /* signal main to stop simulation */
    {                                  /* Sample time: [0.0s, 0.0s]
       */
      if ((rtmGetTFinal(CavityPD_M)!=-1) &&
          !((rtmGetTFinal(CavityPD_M)-(((CavityPD_M->Timing.
             clockTick1+
               CavityPD_M->Timing.clockTickH1* 4294967296.0)) * 1.0E
                 -5)) >
```

```
            (((CavityPD_M->Timing.clockTick1+CavityPD_M->Timing.
                clockTickH1*
               4294967296.0)) * 1.0E-5) * (DBL_EPSILON))) {
        rtmSetErrorStatus(CavityPD_M, "Simulation finished");
      }

      if (rtmGetStopRequested(CavityPD_M)) {
        rtmSetErrorStatus(CavityPD_M, "Simulation finished");
      }
    }

    rt_ertODEUpdateContinuousStates(&CavityPD_M->solverInfo);

    /* Update absolute time for base rate */
    /* The "clockTick0" counts the number of times the code of this
        task has
     * been executed. The absolute time is the multiplication of "
        clockTick0"
     * and "Timing.stepSize0". Size of "clockTick0" ensures timer
        will not
     * overflow during the application lifespan selected.
     * Timer of this task consists of two 32 bit unsigned integers.
     * The two integers represent the low bits Timing.clockTick0 and
        the high bits
     * Timing.clockTickH0. When the low bit overflows to 0, the high
        bits increment.
     */
    if (!(++CavityPD_M->Timing.clockTick0)) {
      ++CavityPD_M->Timing.clockTickH0;
    }

    CavityPD_M->Timing.t[0] = rtsiGetSolverStopTime(&CavityPD_M->
        solverInfo);

    {
      /* Update absolute timer for sample time: [1.0E-5s, 0.0s] */
      /* The "clockTick1" counts the number of times the code of
          this task has
       * been executed. The resolution of this integer timer is 1.0E
          -5, which is the step size
       * of the task. Size of "clockTick1" ensures timer will not
          overflow during the
       * application lifespan selected.
       * Timer of this task consists of two 32 bit unsigned integers
          .
       * The two integers represent the low bits Timing.clockTick1
          and the high bits
       * Timing.clockTickH1. When the low bit overflows to 0, the
          high bits increment.
       */
      CavityPD_M->Timing.clockTick1++;
      if (!CavityPD_M->Timing.clockTick1) {
```

```
395         CavityPD_M->Timing.clockTickH1++;
          }
        }

        if (rtmIsMajorTimeStep(CavityPD_M) &&
400         CavityPD_M->Timing.TaskCounters.TID[2] == 0) {
          /* Update absolute timer for sample time: [0.0001s, 0.0s] */
          /* The "clockTick2" counts the number of times the code of
             this task has
           * been executed. The resolution of this integer timer is
             0.0001, which is the step size
           * of the task. Size of "clockTick2" ensures timer will not
             overflow during the
405        * application lifespan selected.
           * Timer of this task consists of two 32 bit unsigned integers
             .
           * The two integers represent the low bits Timing.clockTick2
             and the high bits
           * Timing.clockTickH2. When the low bit overflows to 0, the
             high bits increment.
           */
410        CavityPD_M->Timing.clockTick2++;
          if (!CavityPD_M->Timing.clockTick2) {
            CavityPD_M->Timing.clockTickH2++;
          }
        }
415
        rate_scheduler();
      }                                    /* end MajorTimeStep */
    }

420 /* Derivatives for root system: '<Root>' */
    void CavityPD_derivatives(void)
    {
      /* Derivatives for S-Function (sfun_spid_contc): '<S30>/State-
         Space' */
      /* Level2 S-Function Block: '<S30>/State-Space' (sfun_spid_contc)
         */
425   {
        SimStruct *rts = CavityPD_M->childSfunctions[0];
        real_T *sfcndX_fx = (real_T *) &((XDot_CavityPD_T *) CavityPD_M
           ->derivs)
          ->StateSpace_CSTATE[0];
        ssSetdX(rts, sfcndX_fx);
430     sfcndDerivatives(rts);
        if (ssGetErrorStatus(rts) != (NULL))
          return;
      }
    }
435
    /* Model initialize function */
    void CavityPD_initialize(void)
```

```
{
  /* Registration code */
440
  /* initialize real-time model */
  (void) memset((void *)CavityPD_M, 0,
                sizeof(RT_MODEL_CavityPD_T));

445  {
    /* Setup solver object */
    rtsiSetSimTimeStepPtr(&CavityPD_M->solverInfo,
                          &CavityPD_M->Timing.simTimeStep);
    rtsiSetTPtr(&CavityPD_M->solverInfo, &rtmGetTPtr(CavityPD_M));
450    rtsiSetStepSizePtr(&CavityPD_M->solverInfo, &CavityPD_M->Timing.
       stepSize0);
    rtsiSetdXPtr(&CavityPD_M->solverInfo, &CavityPD_M->derivs);
    rtsiSetContStatesPtr(&CavityPD_M->solverInfo, (real_T **)
                         &CavityPD_M->contStates);
    rtsiSetNumContStatesPtr(&CavityPD_M->solverInfo,
455      &CavityPD_M->Sizes.numContStates);
    rtsiSetNumPeriodicContStatesPtr(&CavityPD_M->solverInfo,
      &CavityPD_M->Sizes.numPeriodicContStates);
    rtsiSetPeriodicContStateIndicesPtr(&CavityPD_M->solverInfo,
      &CavityPD_M->periodicContStateIndices);
460    rtsiSetPeriodicContStateRangesPtr(&CavityPD_M->solverInfo,
      &CavityPD_M->periodicContStateRanges);
    rtsiSetContStateDisabledPtr(&CavityPD_M->solverInfo, (boolean_T
       **)
      &CavityPD_M->contStateDisabled);
    rtsiSetErrorStatusPtr(&CavityPD_M->solverInfo, (&
       rtmGetErrorStatus
465      (CavityPD_M)));
    rtsiSetRTModelPtr(&CavityPD_M->solverInfo, CavityPD_M);
  }

  rtsiSetSimTimeStep(&CavityPD_M->solverInfo, MAJOR_TIME_STEP);
470  rtsiSetIsMinorTimeStepWithModeChange(&CavityPD_M->solverInfo,
     false);
  rtsiSetIsContModeFrozen(&CavityPD_M->solverInfo, false);
  CavityPD_M->intgData.y = CavityPD_M->odeY;
  CavityPD_M->intgData.f[0] = CavityPD_M->odeF[0];
  CavityPD_M->intgData.f[1] = CavityPD_M->odeF[1];
475  CavityPD_M->intgData.f[2] = CavityPD_M->odeF[2];
  CavityPD_M->contStates = ((X_CavityPD_T *) &CavityPD_X);
  CavityPD_M->contStateDisabled = ((XDis_CavityPD_T *) &
     CavityPD_XDis);
  CavityPD_M->Timing.tStart = (0.0);
  rtsiSetSolverData(&CavityPD_M->solverInfo, (void *)&CavityPD_M->
     intgData);
480  rtsiSetSolverName(&CavityPD_M->solverInfo,"ode3");
  CavityPD_M->solverInfoPtr = (&CavityPD_M->solverInfo);

  /* Initialize timing info */
```

```
  {
485   int_T *mdlTsMap = CavityPD_M->Timing.sampleTimeTaskIDArray;
      mdlTsMap[0] = 0;
      mdlTsMap[1] = 1;
      mdlTsMap[2] = 2;

490   /* polyspace +2 MISRA2012:D4.1 [Justified:Low] "CavityPD_M
         points to
         static memory which is guaranteed to be non-NULL" */
      CavityPD_M->Timing.sampleTimeTaskIDPtr = (&mdlTsMap[0]);
      CavityPD_M->Timing.sampleTimes = (&CavityPD_M->Timing.
         sampleTimesArray[0]);
      CavityPD_M->Timing.offsetTimes = (&CavityPD_M->Timing.
         offsetTimesArray[0]);
495
      /* task periods */
      CavityPD_M->Timing.sampleTimes[0] = (0.0);
      CavityPD_M->Timing.sampleTimes[1] = (1.0E-5);
      CavityPD_M->Timing.sampleTimes[2] = (0.0001);
500
      /* task offsets */
      CavityPD_M->Timing.offsetTimes[0] = (0.0);
      CavityPD_M->Timing.offsetTimes[1] = (0.0);
      CavityPD_M->Timing.offsetTimes[2] = (0.0);
505   }

  rtmSetTPtr(CavityPD_M, &CavityPD_M->Timing.tArray[0]);

  {
510   int_T *mdlSampleHits = CavityPD_M->Timing.sampleHitArray;
      mdlSampleHits[0] = 1;
      mdlSampleHits[1] = 1;
      mdlSampleHits[2] = 1;
      CavityPD_M->Timing.sampleHits = (&mdlSampleHits[0]);
515   }

  rtmSetTFinal(CavityPD_M, 0.001);
  CavityPD_M->Timing.stepSize0 = 1.0E-5;

520   /* Setup for data logging */
  {
      static RTWLogInfo rt_DataLoggingInfo;
      rt_DataLoggingInfo.loggingInterval = (NULL);
      CavityPD_M->rtwLogInfo = &rt_DataLoggingInfo;
525   }

  /* Setup for data logging */
  {
      rtliSetLogXSignalInfo(CavityPD_M->rtwLogInfo, (NULL));
530   rtliSetLogXSignalPtrs(CavityPD_M->rtwLogInfo, (NULL));
      rtliSetLogT(CavityPD_M->rtwLogInfo, "tout");
      rtliSetLogX(CavityPD_M->rtwLogInfo, "");
```

```
    rtliSetLogXFinal(CavityPD_M->rtwLogInfo, "");
    rtliSetLogVarNameModifier(CavityPD_M->rtwLogInfo, "rt_");
    rtliSetLogFormat(CavityPD_M->rtwLogInfo, 4);
    rtliSetLogMaxRows(CavityPD_M->rtwLogInfo, 0);
    rtliSetLogDecimation(CavityPD_M->rtwLogInfo, 1);
    rtliSetLogY(CavityPD_M->rtwLogInfo, "");
    rtliSetLogYSignalInfo(CavityPD_M->rtwLogInfo, (NULL));
    rtliSetLogYSignalPtrs(CavityPD_M->rtwLogInfo, (NULL));
  }

  CavityPD_M->solverInfoPtr = (&CavityPD_M->solverInfo);
  CavityPD_M->Timing.stepSize = (1.0E-5);
  rtsiSetFixedStepSize(&CavityPD_M->solverInfo, 1.0E-5);
  rtsiSetSolverMode(&CavityPD_M->solverInfo,
     SOLVER_MODE_SINGLETASKING);

  /* block I/O */
  (void) memset(((void *) &CavityPD_B), 0,
                sizeof(B_CavityPD_T));

  /* states (continuous) */
  {
    (void) memset((void *)&CavityPD_X, 0,
                  sizeof(X_CavityPD_T));
  }

  /* disabled states */
  {
    (void) memset((void *)&CavityPD_XDis, 0,
                  sizeof(XDis_CavityPD_T));
  }

  /* states (dwork) */
  (void) memset((void *)&CavityPD_DW, 0,
                sizeof(DW_CavityPD_T));

  /* child S-Function registration */
  {
    RTWSfcnInfo *sfcnInfo = &CavityPD_M->NonInlinedSFcns.sfcnInfo;
    CavityPD_M->sfcnInfo = (sfcnInfo);
    rtssSetErrorStatusPtr(sfcnInfo, (&rtmGetErrorStatus(CavityPD_M))
       );
    CavityPD_M->Sizes.numSampTimes = (3);
    rtssSetNumRootSampTimesPtr(sfcnInfo, &CavityPD_M->Sizes.
       numSampTimes);
    CavityPD_M->NonInlinedSFcns.taskTimePtrs[0] = (&rtmGetTPtr(
       CavityPD_M)[0]);
    CavityPD_M->NonInlinedSFcns.taskTimePtrs[1] = (&rtmGetTPtr(
       CavityPD_M)[1]);
    CavityPD_M->NonInlinedSFcns.taskTimePtrs[2] = (&rtmGetTPtr(
       CavityPD_M)[2]);
    rtssSetTPtrPtr(sfcnInfo,CavityPD_M->NonInlinedSFcns.taskTimePtrs
```

```
      );
    rtssSetTStartPtr(sfcnInfo, &rtmGetTStart(CavityPD_M));
    rtssSetTFinalPtr(sfcnInfo, &rtmGetTFinal(CavityPD_M));
    rtssSetTimeOfLastOutputPtr(sfcnInfo, &rtmGetTimeOfLastOutput(
       CavityPD_M));
    rtssSetStepSizePtr(sfcnInfo, &CavityPD_M->Timing.stepSize);
    rtssSetStopRequestedPtr(sfcnInfo, &rtmGetStopRequested(
       CavityPD_M));
    rtssSetDerivCacheNeedsResetPtr(sfcnInfo, &CavityPD_M->
       derivCacheNeedsReset);
    rtssSetZCCacheNeedsResetPtr(sfcnInfo, &CavityPD_M->
       zCCacheNeedsReset);
    rtssSetContTimeOutputInconsistentWithStateAtMajorStepPtr(
       sfcnInfo,
      &CavityPD_M->CTOutputIncnstWithState);
    rtssSetSampleHitsPtr(sfcnInfo, &CavityPD_M->Timing.sampleHits);
    rtssSetPerTaskSampleHitsPtr(sfcnInfo, &CavityPD_M->Timing.
       perTaskSampleHits);
    rtssSetSimModePtr(sfcnInfo, &CavityPD_M->simMode);
    rtssSetSolverInfoPtr(sfcnInfo, &CavityPD_M->solverInfoPtr);
  }

  CavityPD_M->Sizes.numSFcns = (1);

  /* register each child */
  {
    (void) memset((void *)&CavityPD_M->NonInlinedSFcns.
       childSFunctions[0], 0,
                  1*sizeof(SimStruct));
    CavityPD_M->childSfunctions =
      (&CavityPD_M->NonInlinedSFcns.childSFunctionPtrs[0]);
    CavityPD_M->childSfunctions[0] =
      (&CavityPD_M->NonInlinedSFcns.childSFunctions[0]);

    /* Level2 S-Function Block: CavityPD/<S30>/State-Space (
       sfun_spid_contc) */
    {
      SimStruct *rts = CavityPD_M->childSfunctions[0];

      /* timing info */
      time_T *sfcnPeriod = CavityPD_M->NonInlinedSFcns.Sfcn0.
         sfcnPeriod;
      time_T *sfcnOffset = CavityPD_M->NonInlinedSFcns.Sfcn0.
         sfcnOffset;
      int_T *sfcnTsMap = CavityPD_M->NonInlinedSFcns.Sfcn0.sfcnTsMap
         ;
      (void) memset((void*)sfcnPeriod, 0,
                    sizeof(time_T)*1);
      (void) memset((void*)sfcnOffset, 0,
                    sizeof(time_T)*1);
      ssSetSampleTimePtr(rts, &sfcnPeriod[0]);
      ssSetOffsetTimePtr(rts, &sfcnOffset[0]);
```

```
      ssSetSampleTimeTaskIDPtr(rts, sfcnTsMap);

      {
        ssSetBlkInfo2Ptr(rts, &CavityPD_M->NonInlinedSFcns.blkInfo2
            [0]);
      }

      _ssSetBlkInfo2PortInfo2Ptr(rts,
        &CavityPD_M->NonInlinedSFcns.inputOutputPortInfo2[0]);

      /* Set up the mdlInfo pointer */
      ssSetRTWSfcnInfo(rts, CavityPD_M->sfcnInfo);

      /* Allocate memory of model methods 2 */
      {
        ssSetModelMethods2(rts, &CavityPD_M->NonInlinedSFcns.
            methods2[0]);
      }

      /* Allocate memory of model methods 3 */
      {
        ssSetModelMethods3(rts, &CavityPD_M->NonInlinedSFcns.
            methods3[0]);
      }

      /* Allocate memory of model methods 4 */
      {
        ssSetModelMethods4(rts, &CavityPD_M->NonInlinedSFcns.
            methods4[0]);
      }

      /* Allocate memory for states auxilliary information */
      {
        ssSetStatesInfo2(rts, &CavityPD_M->NonInlinedSFcns.
            statesInfo2[0]);
        ssSetPeriodicStatesInfo(rts,
          &CavityPD_M->NonInlinedSFcns.periodicStatesInfo[0]);
      }

      /* inputs */
      {
        _ssSetNumInputPorts(rts, 2);
        ssSetPortInfoForInputs(rts,
          &CavityPD_M->NonInlinedSFcns.Sfcn0.inputPortInfo[0]);
        ssSetPortInfoForInputs(rts,
          &CavityPD_M->NonInlinedSFcns.Sfcn0.inputPortInfo[0]);
        _ssSetPortInfo2ForInputUnits(rts,
          &CavityPD_M->NonInlinedSFcns.Sfcn0.inputPortUnits[0]);
        ssSetInputPortUnit(rts, 0, 0);
        ssSetInputPortUnit(rts, 1, 0);
        _ssSetPortInfo2ForInputCoSimAttribute(rts,
          &CavityPD_M->NonInlinedSFcns.Sfcn0.inputPortCoSimAttribute
```

```
            [0]);
        ssSetInputPortIsContinuousQuantity(rts, 0, 0);
        ssSetInputPortIsContinuousQuantity(rts, 1, 0);

        /* port 0 */
        {
          real_T const **sfcnUPtrs = (real_T const **)
            &CavityPD_M->NonInlinedSFcns.Sfcn0.UPtrs0;
          sfcnUPtrs[0] = &CavityPD_B.DC;
          ssSetInputPortSignalPtrs(rts, 0, (InputPtrsType)&sfcnUPtrs
            [0]);
          _ssSetInputPortNumDimensions(rts, 0, 1);
          ssSetInputPortWidthAsInt(rts, 0, 1);
        }

        /* port 1 */
        {
          real_T const **sfcnUPtrs = (real_T const **)
            &CavityPD_M->NonInlinedSFcns.Sfcn0.UPtrs1;
          sfcnUPtrs[0] = &CavityPD_B.DataTypeConversion;
          sfcnUPtrs[1] = &CavityPD_B.Step;
          sfcnUPtrs[2] = &CavityPD_B.DataTypeConversion_h;
          ssSetInputPortSignalPtrs(rts, 1, (InputPtrsType)&sfcnUPtrs
            [0]);
          _ssSetInputPortNumDimensions(rts, 1, 1);
          ssSetInputPortWidthAsInt(rts, 1, 3);
        }
      }

      /* outputs */
      {
        ssSetPortInfoForOutputs(rts,
          &CavityPD_M->NonInlinedSFcns.Sfcn0.outputPortInfo[0]);
        ssSetPortInfoForOutputs(rts,
          &CavityPD_M->NonInlinedSFcns.Sfcn0.outputPortInfo[0]);
        _ssSetNumOutputPorts(rts, 2);
        _ssSetPortInfo2ForOutputUnits(rts,
          &CavityPD_M->NonInlinedSFcns.Sfcn0.outputPortUnits[0]);
        ssSetOutputPortUnit(rts, 0, 0);
        ssSetOutputPortUnit(rts, 1, 0);
        _ssSetPortInfo2ForOutputCoSimAttribute(rts,
          &CavityPD_M->NonInlinedSFcns.Sfcn0.
            outputPortCoSimAttribute[0]);
        ssSetOutputPortIsContinuousQuantity(rts, 0, 0);
        ssSetOutputPortIsContinuousQuantity(rts, 1, 0);

        /* port 0 */
        {
          _ssSetOutputPortNumDimensions(rts, 0, 1);
          ssSetOutputPortWidthAsInt(rts, 0, 7);
          ssSetOutputPortSignal(rts, 0, ((real_T *) CavityPD_B.
            StateSpace_o1));
```

```
    }

    /* port 1 */
    {
      _ssSetOutputPortNumDimensions(rts, 1, 1);
      ssSetOutputPortWidthAsInt(rts, 1, 6);
      ssSetOutputPortSignal(rts, 1, ((real_T *) CavityPD_B.
         StateSpace_o2));
    }
  }

  /* states */
  ssSetContStates(rts, &CavityPD_X.StateSpace_CSTATE[0]);
  ssSetContStateDisabled(rts, &CavityPD_XDis.StateSpace_CSTATE
     [0]);

  /* path info */
  ssSetModelName(rts, "State-Space");
  ssSetPath(rts, "CavityPD/powergui1/EquivalentModel1/State-
     Space");
  ssSetRTModel(rts,CavityPD_M);
  ssSetParentSS(rts, (NULL));
  ssSetRootSS(rts, rts);
  ssSetVersion(rts, SIMSTRUCT_VERSION_LEVEL2);

  /* parameters */
  {
    mxArray **sfcnParams = (mxArray **)
      &CavityPD_M->NonInlinedSFcns.Sfcn0.params;
    ssSetSFcnParamsCount(rts, 10);
    ssSetSFcnParamsPtr(rts, &sfcnParams[0]);
    ssSetSFcnParam(rts, 0, (mxArray*)CavityPD_P.
       StateSpace_P1_Size);
    ssSetSFcnParam(rts, 1, (mxArray*)CavityPD_P.
       StateSpace_P2_Size);
    ssSetSFcnParam(rts, 2, (mxArray*)CavityPD_P.
       StateSpace_P3_Size);
    ssSetSFcnParam(rts, 3, (mxArray*)CavityPD_P.
       StateSpace_P4_Size);
    ssSetSFcnParam(rts, 4, (mxArray*)CavityPD_P.
       StateSpace_P5_Size);
    ssSetSFcnParam(rts, 5, (mxArray*)CavityPD_P.
       StateSpace_P6_Size);
    ssSetSFcnParam(rts, 6, (mxArray*)CavityPD_P.
       StateSpace_P7_Size);
    ssSetSFcnParam(rts, 7, (mxArray*)CavityPD_P.
       StateSpace_P8_Size);
    ssSetSFcnParam(rts, 8, (mxArray*)CavityPD_P.
       StateSpace_P9_Size);
    ssSetSFcnParam(rts, 9, (mxArray*)CavityPD_P.
       StateSpace_P10_Size);
  }
```

```
    /* work vectors */
    ssSetRWork(rts, (real_T *) &CavityPD_DW.StateSpace_RWORK[0]);
    ssSetIWork(rts, (int_T *) &CavityPD_DW.StateSpace_IWORK[0]);
    ssSetPWork(rts, (void **) &CavityPD_DW.StateSpace_PWORK[0]);

    {
      struct _ssDWorkRecord *dWorkRecord = (struct _ssDWorkRecord
          *)
        &CavityPD_M->NonInlinedSFcns.Sfcn0.dWork;
      struct _ssDWorkAuxRecord *dWorkAuxRecord = (struct
          _ssDWorkAuxRecord *)
        &CavityPD_M->NonInlinedSFcns.Sfcn0.dWorkAux;
      ssSetSFcnDWork(rts, dWorkRecord);
      ssSetSFcnDWorkAux(rts, dWorkAuxRecord);
      ssSetNumDWorkAsInt(rts, 4);

      /* MODE */
      ssSetDWorkWidthAsInt(rts, 0, 4);
      ssSetDWorkDataType(rts, 0,SS_INTEGER);
      ssSetDWorkComplexSignal(rts, 0, 0);
      ssSetDWork(rts, 0, &CavityPD_DW.StateSpace_MODE[0]);

      /* RWORK */
      ssSetDWorkWidthAsInt(rts, 1, 2);
      ssSetDWorkDataType(rts, 1,SS_DOUBLE);
      ssSetDWorkComplexSignal(rts, 1, 0);
      ssSetDWork(rts, 1, &CavityPD_DW.StateSpace_RWORK[0]);

      /* IWORK */
      ssSetDWorkWidthAsInt(rts, 2, 23);
      ssSetDWorkDataType(rts, 2,SS_INTEGER);
      ssSetDWorkComplexSignal(rts, 2, 0);
      ssSetDWork(rts, 2, &CavityPD_DW.StateSpace_IWORK[0]);

      /* PWORK */
      ssSetDWorkWidthAsInt(rts, 3, 22);
      ssSetDWorkDataType(rts, 3,SS_POINTER);
      ssSetDWorkComplexSignal(rts, 3, 0);
      ssSetDWork(rts, 3, &CavityPD_DW.StateSpace_PWORK[0]);
    }

    ssSetModeVector(rts, (int_T *) &CavityPD_DW.StateSpace_MODE
        [0]);

    /* registration */
    sfun_spid_contc(rts);
    sfcnInitializeSizes(rts);
    sfcnInitializeSampleTimes(rts);

    /* adjust sample time */
    ssSetSampleTime(rts, 0, 0.0);
```

```
      ssSetOffsetTime(rts, 0, 0.0);
      sfcnTsMap[0] = 0;

      /* set compiled values of dynamic vector attributes */
      ssSetNumNonsampledZCsAsInt(rts, 0);

      /* Update connectivity flags for each port */
      _ssSetInputPortConnected(rts, 0, 1);
      _ssSetInputPortConnected(rts, 1, 1);
      _ssSetOutputPortConnected(rts, 0, 1);
      _ssSetOutputPortConnected(rts, 1, 1);
      _ssSetOutputPortBeingMerged(rts, 0, 0);
      _ssSetOutputPortBeingMerged(rts, 1, 0);

      /* Update the BufferDstPort flags for each input port */
      ssSetInputPortBufferDstPort(rts, 0, -1);
      ssSetInputPortBufferDstPort(rts, 1, -1);
    }
  }

  /* Matfile logging */
  rt_StartDataLoggingWithStartTime(CavityPD_M->rtwLogInfo, 0.0,
     rtmGetTFinal
    (CavityPD_M), CavityPD_M->Timing.stepSize0, (&rtmGetErrorStatus(
       CavityPD_M)));

  /* SetupRuntimeResources for Scope: '<Root>/Scope' */
  {
    RTWLogSignalInfo rt_ScopeSignalInfo;
    static int_T rt_ScopeSignalWidths[] = { 1 };

    static int_T rt_ScopeSignalNumDimensions[] = { 1 };

    static int_T rt_ScopeSignalDimensions[] = { 1 };

    static void *rt_ScopeCurrSigDims[] = { (NULL) };

    static int_T rt_ScopeCurrSigDimsSize[] = { 4 };

    static const char_T *rt_ScopeSignalLabels[] = { "" };

    static char_T rt_ScopeSignalTitles[] = "";
    static int_T rt_ScopeSignalTitleLengths[] = { 0 };

    static boolean_T rt_ScopeSignalIsVarDims[] = { 0 };

    static int_T rt_ScopeSignalPlotStyles[] = { 0 };

    BuiltInDTypeId dTypes[1] = { SS_DOUBLE };

    static char_T rt_ScopeBlockName[] = "CavityPD/Scope";
    static int_T rt_ScopeFrameData[] = { 0 };
```

```
    static RTWPreprocessingFcnPtr
        rt_ScopeSignalLoggingPreprocessingFcnPtrs[] =
      {
      (NULL)
    };

    rt_ScopeSignalInfo.numSignals = 1;
    rt_ScopeSignalInfo.numCols = rt_ScopeSignalWidths;
    rt_ScopeSignalInfo.numDims = rt_ScopeSignalNumDimensions;
    rt_ScopeSignalInfo.dims = rt_ScopeSignalDimensions;
    rt_ScopeSignalInfo.isVarDims = rt_ScopeSignalIsVarDims;
    rt_ScopeSignalInfo.currSigDims = rt_ScopeCurrSigDims;
    rt_ScopeSignalInfo.currSigDimsSize = rt_ScopeCurrSigDimsSize;
    rt_ScopeSignalInfo.dataTypes = dTypes;
    rt_ScopeSignalInfo.complexSignals = (NULL);
    rt_ScopeSignalInfo.frameData = rt_ScopeFrameData;
    rt_ScopeSignalInfo.preprocessingPtrs =
      rt_ScopeSignalLoggingPreprocessingFcnPtrs;
    rt_ScopeSignalInfo.labels.cptr = rt_ScopeSignalLabels;
    rt_ScopeSignalInfo.titles = rt_ScopeSignalTitles;
    rt_ScopeSignalInfo.titleLengths = rt_ScopeSignalTitleLengths;
    rt_ScopeSignalInfo.plotStyles = rt_ScopeSignalPlotStyles;
    rt_ScopeSignalInfo.blockNames.cptr = (NULL);
    rt_ScopeSignalInfo.stateNames.cptr = (NULL);
    rt_ScopeSignalInfo.crossMdlRef = (NULL);
    rt_ScopeSignalInfo.dataTypeConvert = (NULL);
    CavityPD_DW.Scope_PWORK.LoggedData = rt_CreateStructLogVar(
      CavityPD_M->rtwLogInfo,
      0.0,
      rtmGetTFinal(CavityPD_M),
      CavityPD_M->Timing.stepSize0,
      (&rtmGetErrorStatus(CavityPD_M)),
      "ScopeData2",
      0,
      0,
      1,
      1.0E-5,
      &rt_ScopeSignalInfo,
      rt_ScopeBlockName);
    if (CavityPD_DW.Scope_PWORK.LoggedData == (NULL))
      return;
  }

  /* Start for Constant: '<S17>/DC' */
  CavityPD_B.DC = CavityPD_P.DCVoltageSource1_Amplitude;

  /* Start for S-Function (sfun_spid_contc): '<S30>/State-Space' */
  /* Level2 S-Function Block: '<S30>/State-Space' (sfun_spid_contc)
     */
  {
    SimStruct *rts = CavityPD_M->childSfunctions[0];
```

```
    sfcnStart(rts);
    if (ssGetErrorStatus(rts) != (NULL))
      return;
  }
905
  {
    real_T tmin;
    int32_T r;
    int32_T t;
910 uint32_T tseed;

    /* InitializeConditions for S-Function (sfun_spid_contc): '<S30
       >/State-Space' */
    /* Level2 S-Function Block: '<S30>/State-Space' (sfun_spid_contc
       ) */
    {
915   SimStruct *rts = CavityPD_M->childSfunctions[0];
      sfcnInitializeConditions(rts);
      if (ssGetErrorStatus(rts) != (NULL))
        return;
    }
920
    /* InitializeConditions for UniformRandomNumber: '<Root>/Uniform
        Random Number' */
    tmin = floor(CavityPD_P.UniformRandomNumber_Seed);
    if (rtIsNaN(tmin) || rtIsInf(tmin)) {
      tmin = 0.0;
925 } else {
      tmin = fmod(tmin, 4.294967296E+9);
    }

    tseed = tmin < 0.0 ? (uint32_T)-(int32_T)(uint32_T)-tmin : (
       uint32_T)tmin;
930 r = (int32_T)(tseed >> 16U);
    t = (int32_T)(tseed & 32768U);
    tseed = ((((tseed - ((uint32_T)r << 16U)) + (uint32_T)t) << 16U)
        + (uint32_T)
          t) + (uint32_T)r;
    if (tseed < 1U) {
935   tseed = 1144108930U;
    } else if (tseed > 2147483646U) {
      tseed = 2147483646U;
    }

940 CavityPD_DW.RandSeed = tseed;
    tmin = CavityPD_P.UniformRandomNumber_Minimum;
    CavityPD_DW.UniformRandomNumber_NextOutput =
      (CavityPD_P.UniformRandomNumber_Maximum - tmin) *
      rt_urand_Upu32_Yd_f_pw_snf(&CavityPD_DW.RandSeed) + tmin;
945
    /* End of InitializeConditions for UniformRandomNumber: '<Root>/
       Uniform Random Number' */
```

```
  }
}

/* Model terminate function */
void CavityPD_terminate(void)
{
  /* Terminate for S-Function (sfun_spid_contc): '<S30>/State-Space'
     */
  /* Level2 S-Function Block: '<S30>/State-Space' (sfun_spid_contc)
     */
  {
    SimStruct *rts = CavityPD_M->childSfunctions[0];
    sfcnTerminate(rts);
  }
}
```

Listing A.2: Corona

```
%This code performs a simulation of corona

/* Block signals (default storage) */
B_Corona_T Corona_B;

/* Continuous states */
X_Corona_T Corona_X;

/* Disabled State Vector */
XDis_Corona_T Corona_XDis;

/* Block states (default storage) */
DW_Corona_T Corona_DW;

/* Real-time model */
static RT_MODEL_Corona_T Corona_M_;
RT_MODEL_Corona_T *const Corona_M = &Corona_M_;
static void rate_scheduler(void);

/*
 *         This function updates active task flag for each subrate.
 *         The function is called at model base rate, hence the
 *         generated code self-manages all its subrates.
 */
static void rate_scheduler(void)
{
  /* Compute which subrates run during the next base time step.
     Subrates
   * are an integer multiple of the base rate counter.  Therefore,
     the subtask
   * counter is reset when it reaches its limit (zero means run).
   */
  (Corona_M->Timing.TaskCounters.TID[2])++;
```

```
  if ((Corona_M->Timing.TaskCounters.TID[2]) > 9) {/* Sample time:
      [0.0001s, 0.0s] */
    Corona_M->Timing.TaskCounters.TID[2] = 0;
  }

  Corona_M->Timing.sampleHits[2] = (Corona_M->Timing.TaskCounters.
     TID[2] == 0) ?
    1 : 0;
}

/* Projection for root system: '<Root>' */
void Corona_projection(void)
{
  /* Projection for S-Function (sfun_spid_contc): '<S22>/State-Space
     ' */
  /* Level2 S-Function Block: '<S22>/State-Space' (sfun_spid_contc)
     */
  {
    SimStruct *rts = Corona_M->childSfunctions[0];
    sfcnProjection(rts);
    if (ssGetErrorStatus(rts) != (NULL))
      return;
  }
}

/*
 * This function updates continuous states using the ODE3 fixed-step
 * solver algorithm
 */
static void rt_ertODEUpdateContinuousStates(RTWSolverInfo *si )
{
  /* Solver Matrices */
  static const real_T rt_ODE3_A[3] = {
    1.0/2.0, 3.0/4.0, 1.0
  };

  static const real_T rt_ODE3_B[3][3] = {
    { 1.0/2.0, 0.0, 0.0 },

    { 0.0, 3.0/4.0, 0.0 },

    { 2.0/9.0, 1.0/3.0, 4.0/9.0 }
  };

  time_T t = rtsiGetT(si);
  time_T tnew = rtsiGetSolverStopTime(si);
  time_T h = rtsiGetStepSize(si);
  real_T *x = rtsiGetContStates(si);
  ODE3_IntgData *id = (ODE3_IntgData *)rtsiGetSolverData(si);
  real_T *y = id->y;
  real_T *f0 = id->f[0];
  real_T *f1 = id->f[1];
```

```
80  real_T *f2 = id->f[2];
    real_T hB[3];
    int_T i;
    int_T nXc = 5;
    rtsiSetSimTimeStep(si,MINOR_TIME_STEP);
85
    /* Save the state values at time t in y, we'll use x as ynew. */
    (void) memcpy(y, x,
                  (uint_T)nXc*sizeof(real_T));

90  /* Assumes that rtsiSetT and ModelOutputs are up-to-date */
    /* f0 = f(t,y) */
    rtsiSetdX(si, f0);
    Corona_derivatives();

95  /* f(:,2) = feval(odefile, t + hA(1), y + f*hB(:,1), args(:)(*));
       */
    hB[0] = h * rt_ODE3_B[0][0];
    for (i = 0; i < nXc; i++) {
      x[i] = y[i] + (f0[i]*hB[0]);
    }
100
    rtsiSetT(si, t + h*rt_ODE3_A[0]);
    rtsiSetdX(si, f1);
    Corona_step();
    Corona_derivatives();
105
    /* f(:,3) = feval(odefile, t + hA(2), y + f*hB(:,2), args(:)(*));
       */
    for (i = 0; i <= 1; i++) {
      hB[i] = h * rt_ODE3_B[1][i];
    }
110
    for (i = 0; i < nXc; i++) {
      x[i] = y[i] + (f0[i]*hB[0] + f1[i]*hB[1]);
    }

115 rtsiSetT(si, t + h*rt_ODE3_A[1]);
    rtsiSetdX(si, f2);
    Corona_step();
    Corona_derivatives();

120 /* tnew = t + hA(3);
       ynew = y + f*hB(:,3); */
    for (i = 0; i <= 2; i++) {
      hB[i] = h * rt_ODE3_B[2][i];
    }
125
    for (i = 0; i < nXc; i++) {
      x[i] = y[i] + (f0[i]*hB[0] + f1[i]*hB[1] + f2[i]*hB[2]);
    }
```

```
  rtsiSetT(si, tnew);
  Corona_step();
  Corona_projection();
  rtsiSetSimTimeStep(si,MAJOR_TIME_STEP);
}

real_T rt_urand_Upu32_Yd_f_pw_snf(uint32_T *u)
{
  uint32_T hi;
  uint32_T lo;

  /* Uniform random number generator (random number between 0 and 1)

     #define IA      16807                      magic multiplier =
         7^5
     #define IM      2147483647                 modulus = 2^31-1
     #define IQ      127773                     IM div IA
     #define IR      2836                       IM modulo IA
     #define S       4.656612875245797e-10      reciprocal of 2^31-1
     test = IA * (seed % IQ) - IR * (seed/IQ)
     seed = test < 0 ? (test + IM) : test
     return (seed*S)
   */
  lo = *u % 127773U * 16807U;
  hi = *u / 127773U * 2836U;
  if (lo < hi) {
    *u = 2147483647U - (hi - lo);
  } else {
    *u = lo - hi;
  }

  return (real_T)*u * 4.6566128752457969E-10;
}

/* Model step function */
void Corona_step(void)
{
  if (rtmIsMajorTimeStep(Corona_M)) {
    /* set solver stop time */
    if (!(Corona_M->Timing.clockTick0+1)) {
      rtsiSetSolverStopTime(&Corona_M->solverInfo,
                            ((Corona_M->Timing.clockTickH0 + 1) *
        Corona_M->Timing.stepSize0 * 4294967296.0));
    } else {
      rtsiSetSolverStopTime(&Corona_M->solverInfo, ((Corona_M->
         Timing.clockTick0
        + 1) * Corona_M->Timing.stepSize0 + Corona_M->Timing.
           clockTickH0 *
        Corona_M->Timing.stepSize0 * 4294967296.0));
    }
  }                                    /* end MajorTimeStep */
```

```
  /* Update absolute time of base rate at minor time step */
  if (rtmIsMinorTimeStep(Corona_M)) {
    Corona_M->Timing.t[0] = rtsiGetT(&Corona_M->solverInfo);
  }

  {
    real_T currentTime;
    if (rtmIsMajorTimeStep(Corona_M) &&
        Corona_M->Timing.TaskCounters.TID[1] == 0) {
      /* Constant: '<S12>/DC' */
      Corona_B.DC = Corona_P.DCVoltageSource1_Amplitude;
    }

    /* S-Function (sfun_spid_contc): '<S22>/State-Space' */

    /* Level2 S-Function Block: '<S22>/State-Space' (sfun_spid_contc
       ) */
    {
      SimStruct *rts = Corona_M->childSfunctions[0];
      sfcnOutputs(rts,0);
    }

    if (rtmIsMajorTimeStep(Corona_M) &&
        Corona_M->Timing.TaskCounters.TID[1] == 0) {
      /* UniformRandomNumber: '<Root>/RABDOM NUMBER GENERATOR' */
      Corona_B.RABDOMNUMBERGENERATOR = Corona_DW.
        RABDOMNUMBERGENERATOR_NextOutpu;

      /* RelationalOperator: '<Root>/GreaterThanOrEqual'
         incorporates:
       *  Constant: '<Root>/FREE ELECTRON AVAILABILITY'
       */
      Corona_B.GreaterThanOrEqual = (Corona_B.RABDOMNUMBERGENERATOR
        >=
        Corona_P.FREEELECTRONAVAILABILITY_Value);
    }

    /* Gain: '<S8>/do not delete this gain' */
    Corona_B.donotdeletethisgain = Corona_P.donotdeletethisgain_Gain
      *
      Corona_B.StateSpace_o1[3];

    /* Abs: '<Root>/Abs' */
    Corona_B.Abs = fabs(Corona_B.donotdeletethisgain);

    /* RelationalOperator: '<Root>/GreaterThanOrEqual1' incorporates
       :
     *  Constant: '<Root>/CORONA INCEPTION VOLTAGE'
     */
    Corona_B.GreaterThanOrEqual1 = (Corona_B.Abs >=
      Corona_P.CORONAINCEPTIONVOLTAGE_Value);
    if (rtmIsMajorTimeStep(Corona_M) &&
```

```
225         Corona_M->Timing.TaskCounters.TID[1] == 0) {
    }

    /* Logic: '<Root>/AND' */
    Corona_B.AND = (Corona_B.GreaterThanOrEqual && Corona_B.
       GreaterThanOrEqual1);
230 if (rtmIsMajorTimeStep(Corona_M) &&
        Corona_M->Timing.TaskCounters.TID[1] == 0) {
      /* Scope: '<Root>/VOLTAGE AT THE POINT' */
      if (rtmIsMajorTimeStep(Corona_M)) {
        StructLogVar *svar = (StructLogVar *)
235       Corona_DW.VOLTAGEATTHEPOINT_PWORK.LoggedData;
        LogVar *var = svar->signals.values;

        /* time */
        {
240       double locTime = (((Corona_M->Timing.clockTick1+
                              Corona_M->Timing.clockTickH1*
                                 4294967296.0)) *
                            1.0E-5)
            ;
          rt_UpdateLogVar((LogVar *)svar->time, &locTime, 0);
245     }

        /* signals */
        {
          real_T up0[1];
250       up0[0] = Corona_B.donotdeletethisgain;
          rt_UpdateLogVar((LogVar *)var, up0, 0);
        }
      }
    }
255
    /* DataTypeConversion: '<S13>/Data Type Conversion' */
    Corona_B.DataTypeConversion = Corona_B.AND;

    /* Gain: '<S1>/do not delete this gain' */
260 Corona_B.donotdeletethisgain_j = Corona_P.
       donotdeletethisgain_Gain_n *
      Corona_B.StateSpace_o1[5];
    if (rtmIsMajorTimeStep(Corona_M) &&
        Corona_M->Timing.TaskCounters.TID[1] == 0) {
    }
265
    /* Gain: '<S6>/do not delete this gain' */
    Corona_B.donotdeletethisgain_k = Corona_P.
       donotdeletethisgain_Gain_m *
      Corona_B.StateSpace_o1[1];
    if (rtmIsMajorTimeStep(Corona_M) &&
270     Corona_M->Timing.TaskCounters.TID[1] == 0) {
    }
```

```
    /* Gain: '<S5>/do not delete this gain' */
    Corona_B.donotdeletethisgain_b = Corona_P.
       donotdeletethisgain_Gain_k *
275   Corona_B.StateSpace_o1[0];

    /* Sum: '<Root>/Vpd' */
    Corona_B.Vpd = Corona_B.donotdeletethisgain_b -
      Corona_B.donotdeletethisgain_k;

    /* Gain: '<S9>/do not delete this gain' */
    Corona_B.donotdeletethisgain_d = Corona_P.
       donotdeletethisgain_Gain_o *
      Corona_B.StateSpace_o1[4];
    if (rtmIsMajorTimeStep(Corona_M) &&
        Corona_M->Timing.TaskCounters.TID[1] == 0) {
    }

    /* Gain: '<S7>/do not delete this gain' */
    Corona_B.donotdeletethisgain_l = Corona_P.
       donotdeletethisgain_Gain_l *
      Corona_B.StateSpace_o1[2];
    if (rtmIsMajorTimeStep(Corona_M) &&
        Corona_M->Timing.TaskCounters.TID[1] == 0) {
    }

    if (rtmIsMajorTimeStep(Corona_M) &&
        Corona_M->Timing.TaskCounters.TID[2] == 0) {
      /* Step: '<Root>/Step' */
      currentTime = (((Corona_M->Timing.clockTick2+Corona_M->Timing.
         clockTickH2*
                      4294967296.0)) * 0.0001);
      if (currentTime < Corona_P.Step_Time) {
        /* Step: '<Root>/Step' */
        Corona_B.Step = Corona_P.Step_Y0;
      } else {
        /* Step: '<Root>/Step' */
        Corona_B.Step = Corona_P.Step_YFinal;
      }

      /* End of Step: '<Root>/Step' */
    }
  }

  if (rtmIsMajorTimeStep(Corona_M)) {
    /* Matfile logging */
    rt_UpdateTXYLogVars(Corona_M->rtwLogInfo, (Corona_M->Timing.t));
  }                                    /* end MajorTimeStep */

  if (rtmIsMajorTimeStep(Corona_M)) {
    real_T tmin;

    /* Update for S-Function (sfun_spid_contc): '<S22>/State-Space'
```

```
      */
    /* Level2 S-Function Block: '<S22>/State-Space' (sfun_spid_contc
       ) */
    {
      SimStruct *rts = Corona_M->childSfunctions[0];
      sfcnUpdate(rts,0);
      if (ssGetErrorStatus(rts) != (NULL))
        return;
    }

    if (rtmIsMajorTimeStep(Corona_M) &&
        Corona_M->Timing.TaskCounters.TID[1] == 0) {
      /* Update for UniformRandomNumber: '<Root>/RABDOM NUMBER
         GENERATOR' */
      tmin = Corona_P.RABDOMNUMBERGENERATOR_Minimum;
      Corona_DW.RABDOMNUMBERGENERATOR_NextOutpu =
        (Corona_P.RABDOMNUMBERGENERATOR_Maximum - tmin) *
        rt_urand_Upu32_Yd_f_pw_snf(&Corona_DW.RandSeed) + tmin;
    }
  }                                    /* end MajorTimeStep */

  if (rtmIsMajorTimeStep(Corona_M)) {
    /* signal main to stop simulation */
    {                                  /* Sample time: [0.0s, 0.0s]
       */
      if ((rtmGetTFinal(Corona_M)!=-1) &&
          !((rtmGetTFinal(Corona_M)-(((Corona_M->Timing.clockTick1+
               Corona_M->Timing.clockTickH1* 4294967296.0)) * 1.0E
                  -5)) >
            (((Corona_M->Timing.clockTick1+Corona_M->Timing.
               clockTickH1*
               4294967296.0)) * 1.0E-5) * (DBL_EPSILON))) {
        rtmSetErrorStatus(Corona_M, "Simulation finished");
      }

      if (rtmGetStopRequested(Corona_M)) {
        rtmSetErrorStatus(Corona_M, "Simulation finished");
      }
    }

    rt_ertODEUpdateContinuousStates(&Corona_M->solverInfo);

    /* Update absolute time for base rate */
    /* The "clockTick0" counts the number of times the code of this
       task has
     * been executed. The absolute time is the multiplication of "
       clockTick0"
     * and "Timing.stepSize0". Size of "clockTick0" ensures timer
       will not
     * overflow during the application lifespan selected.
     * Timer of this task consists of two 32 bit unsigned integers.
     * The two integers represent the low bits Timing.clockTick0 and
```

```
        the high bits
     * Timing.clockTickH0. When the low bit overflows to 0, the high
        bits increment.
     */
    if (!(++Corona_M->Timing.clockTick0)) {
      ++Corona_M->Timing.clockTickH0;
    }

    Corona_M->Timing.t[0] = rtsiGetSolverStopTime(&Corona_M->
       solverInfo);

    {
      /* Update absolute timer for sample time: [1.0E-5s, 0.0s] */
      /* The "clockTick1" counts the number of times the code of
         this task has
       * been executed. The resolution of this integer timer is 1.0E
         -5, which is the step size
       * of the task. Size of "clockTick1" ensures timer will not
         overflow during the
       * application lifespan selected.
       * Timer of this task consists of two 32 bit unsigned integers
         .
       * The two integers represent the low bits Timing.clockTick1
         and the high bits
       * Timing.clockTickH1. When the low bit overflows to 0, the
         high bits increment.
       */
      Corona_M->Timing.clockTick1++;
      if (!Corona_M->Timing.clockTick1) {
        Corona_M->Timing.clockTickH1++;
      }
    }

    if (rtmIsMajorTimeStep(Corona_M) &&
        Corona_M->Timing.TaskCounters.TID[2] == 0) {
      /* Update absolute timer for sample time: [0.0001s, 0.0s] */
      /* The "clockTick2" counts the number of times the code of
         this task has
       * been executed. The resolution of this integer timer is
         0.0001, which is the step size
       * of the task. Size of "clockTick2" ensures timer will not
         overflow during the
       * application lifespan selected.
       * Timer of this task consists of two 32 bit unsigned integers
         .
       * The two integers represent the low bits Timing.clockTick2
         and the high bits
       * Timing.clockTickH2. When the low bit overflows to 0, the
         high bits increment.
       */
      Corona_M->Timing.clockTick2++;
      if (!Corona_M->Timing.clockTick2) {
```

```
        Corona_M->Timing.clockTickH2++;
      }
    }

    rate_scheduler();
  }                                    /* end MajorTimeStep */
}

/* Derivatives for root system: '<Root>' */
void Corona_derivatives(void)
{
  /* Derivatives for S-Function (sfun_spid_contc): '<S22>/State-
     Space' */
  /* Level2 S-Function Block: '<S22>/State-Space' (sfun_spid_contc)
     */
  {
    SimStruct *rts = Corona_M->childSfunctions[0];
    real_T *sfcndX_fx = (real_T *) &((XDot_Corona_T *) Corona_M->
       derivs)
      ->StateSpace_CSTATE[0];
    ssSetdX(rts, sfcndX_fx);
    sfcnDerivatives(rts);
    if (ssGetErrorStatus(rts) != (NULL))
      return;
  }
}

/* Model initialize function */
void Corona_initialize(void)
{
  /* Registration code */

  /* initialize real-time model */
  (void) memset((void *)Corona_M, 0,
                sizeof(RT_MODEL_Corona_T));

  {
    /* Setup solver object */
    rtsiSetSimTimeStepPtr(&Corona_M->solverInfo, &Corona_M->Timing.
       simTimeStep);
    rtsiSetTPtr(&Corona_M->solverInfo, &rtmGetTPtr(Corona_M));
    rtsiSetStepSizePtr(&Corona_M->solverInfo, &Corona_M->Timing.
       stepSize0);
    rtsiSetdXPtr(&Corona_M->solverInfo, &Corona_M->derivs);
    rtsiSetContStatesPtr(&Corona_M->solverInfo, (real_T **)
                         &Corona_M->contStates);
    rtsiSetNumContStatesPtr(&Corona_M->solverInfo,
      &Corona_M->Sizes.numContStates);
    rtsiSetNumPeriodicContStatesPtr(&Corona_M->solverInfo,
      &Corona_M->Sizes.numPeriodicContStates);
    rtsiSetPeriodicContStateIndicesPtr(&Corona_M->solverInfo,
      &Corona_M->periodicContStateIndices);
```

```
    rtsiSetPeriodicContStateRangesPtr(&Corona_M->solverInfo,
      &Corona_M->periodicContStateRanges);
    rtsiSetContStateDisabledPtr(&Corona_M->solverInfo, (boolean_T**)
      &Corona_M->contStateDisabled);
    rtsiSetErrorStatusPtr(&Corona_M->solverInfo, (&rtmGetErrorStatus
       (Corona_M)));
    rtsiSetRTModelPtr(&Corona_M->solverInfo, Corona_M);
  }

  rtsiSetSimTimeStep(&Corona_M->solverInfo, MAJOR_TIME_STEP);
  rtsiSetIsMinorTimeStepWithModeChange(&Corona_M->solverInfo, false)
     ;
  rtsiSetIsContModeFrozen(&Corona_M->solverInfo, false);
  Corona_M->intgData.y = Corona_M->odeY;
  Corona_M->intgData.f[0] = Corona_M->odeF[0];
  Corona_M->intgData.f[1] = Corona_M->odeF[1];
  Corona_M->intgData.f[2] = Corona_M->odeF[2];
  Corona_M->contStates = ((X_Corona_T *) &Corona_X);
  Corona_M->contStateDisabled = ((XDis_Corona_T *) &Corona_XDis);
  Corona_M->Timing.tStart = (0.0);
  rtsiSetSolverData(&Corona_M->solverInfo, (void *)&Corona_M->
     intgData);
  rtsiSetSolverName(&Corona_M->solverInfo,"ode3");
  Corona_M->solverInfoPtr = (&Corona_M->solverInfo);

  /* Initialize timing info */
  {
    int_T *mdlTsMap = Corona_M->Timing.sampleTimeTaskIDArray;
    mdlTsMap[0] = 0;
    mdlTsMap[1] = 1;
    mdlTsMap[2] = 2;

    /* polyspace +2 MISRA2012:D4.1 [Justified:Low] "Corona_M points
       to
       static memory which is guaranteed to be non-NULL" */
    Corona_M->Timing.sampleTimeTaskIDPtr = (&mdlTsMap[0]);
    Corona_M->Timing.sampleTimes = (&Corona_M->Timing.
       sampleTimesArray[0]);
    Corona_M->Timing.offsetTimes = (&Corona_M->Timing.
       offsetTimesArray[0]);

    /* task periods */
    Corona_M->Timing.sampleTimes[0] = (0.0);
    Corona_M->Timing.sampleTimes[1] = (1.0E-5);
    Corona_M->Timing.sampleTimes[2] = (0.0001);

    /* task offsets */
    Corona_M->Timing.offsetTimes[0] = (0.0);
    Corona_M->Timing.offsetTimes[1] = (0.0);
    Corona_M->Timing.offsetTimes[2] = (0.0);
  }
```

```
  rtmSetTPtr(Corona_M, &Corona_M->Timing.tArray[0]);
495
  {
    int_T *mdlSampleHits = Corona_M->Timing.sampleHitArray;
    mdlSampleHits[0] = 1;
    mdlSampleHits[1] = 1;
500   mdlSampleHits[2] = 1;
    Corona_M->Timing.sampleHits = (&mdlSampleHits[0]);
  }

  rtmSetTFinal(Corona_M, 0.001);
505 Corona_M->Timing.stepSize0 = 1.0E-5;

  /* Setup for data logging */
  {
    static RTWLogInfo rt_DataLoggingInfo;
510   rt_DataLoggingInfo.loggingInterval = (NULL);
    Corona_M->rtwLogInfo = &rt_DataLoggingInfo;
  }

  /* Setup for data logging */
515 {
    rtliSetLogXSignalInfo(Corona_M->rtwLogInfo, (NULL));
    rtliSetLogXSignalPtrs(Corona_M->rtwLogInfo, (NULL));
    rtliSetLogT(Corona_M->rtwLogInfo, "tout");
    rtliSetLogX(Corona_M->rtwLogInfo, "");
520   rtliSetLogXFinal(Corona_M->rtwLogInfo, "");
    rtliSetLogVarNameModifier(Corona_M->rtwLogInfo, "rt_");
    rtliSetLogFormat(Corona_M->rtwLogInfo, 4);
    rtliSetLogMaxRows(Corona_M->rtwLogInfo, 0);
    rtliSetLogDecimation(Corona_M->rtwLogInfo, 1);
525   rtliSetLogY(Corona_M->rtwLogInfo, "");
    rtliSetLogYSignalInfo(Corona_M->rtwLogInfo, (NULL));
    rtliSetLogYSignalPtrs(Corona_M->rtwLogInfo, (NULL));
  }

530 Corona_M->solverInfoPtr = (&Corona_M->solverInfo);
  Corona_M->Timing.stepSize = (1.0E-5);
  rtsiSetFixedStepSize(&Corona_M->solverInfo, 1.0E-5);
  rtsiSetSolverMode(&Corona_M->solverInfo, SOLVER_MODE_SINGLETASKING
     );

535 /* block I/O */
  (void) memset(((void *) &Corona_B), 0,
                sizeof(B_Corona_T));

  /* states (continuous) */
540 {
    (void) memset((void *)&Corona_X, 0,
                  sizeof(X_Corona_T));
  }
```

```
545   /* disabled states */
      {
        (void) memset((void *)&Corona_XDis, 0,
                      sizeof(XDis_Corona_T));
      }
550
      /* states (dwork) */
      (void) memset((void *)&Corona_DW, 0,
                    sizeof(DW_Corona_T));

555   /* child S-Function registration */
      {
        RTWSfcnInfo *sfcnInfo = &Corona_M->NonInlinedSFcns.sfcnInfo;
        Corona_M->sfcnInfo = (sfcnInfo);
        rtssSetErrorStatusPtr(sfcnInfo, (&rtmGetErrorStatus(Corona_M)));
560     Corona_M->Sizes.numSampTimes = (3);
        rtssSetNumRootSampTimesPtr(sfcnInfo, &Corona_M->Sizes.
           numSampTimes);
        Corona_M->NonInlinedSFcns.taskTimePtrs[0] = (&rtmGetTPtr(
           Corona_M)[0]);
        Corona_M->NonInlinedSFcns.taskTimePtrs[1] = (&rtmGetTPtr(
           Corona_M)[1]);
        Corona_M->NonInlinedSFcns.taskTimePtrs[2] = (&rtmGetTPtr(
           Corona_M)[2]);
565     rtssSetTPtrPtr(sfcnInfo,Corona_M->NonInlinedSFcns.taskTimePtrs);
        rtssSetTStartPtr(sfcnInfo, &rtmGetTStart(Corona_M));
        rtssSetTFinalPtr(sfcnInfo, &rtmGetTFinal(Corona_M));
        rtssSetTimeOfLastOutputPtr(sfcnInfo, &rtmGetTimeOfLastOutput(
           Corona_M));
        rtssSetStepSizePtr(sfcnInfo, &Corona_M->Timing.stepSize);
570     rtssSetStopRequestedPtr(sfcnInfo, &rtmGetStopRequested(Corona_M)
           );
        rtssSetDerivCacheNeedsResetPtr(sfcnInfo, &Corona_M->
           derivCacheNeedsReset);
        rtssSetZCCacheNeedsResetPtr(sfcnInfo, &Corona_M->
           zCCacheNeedsReset);
        rtssSetContTimeOutputInconsistentWithStateAtMajorStepPtr(
           sfcnInfo,
          &Corona_M->CTOutputIncnstWithState);
575     rtssSetSampleHitsPtr(sfcnInfo, &Corona_M->Timing.sampleHits);
        rtssSetPerTaskSampleHitsPtr(sfcnInfo, &Corona_M->Timing.
           perTaskSampleHits);
        rtssSetSimModePtr(sfcnInfo, &Corona_M->simMode);
        rtssSetSolverInfoPtr(sfcnInfo, &Corona_M->solverInfoPtr);
      }
580
      Corona_M->Sizes.numSFcns = (1);

      /* register each child */
      {
585     (void) memset((void *)&Corona_M->NonInlinedSFcns.childSFunctions
           [0], 0,
```

```
                1*sizeof(SimStruct));
    Corona_M->childSfunctions = (&Corona_M->NonInlinedSFcns.
       childSFunctionPtrs[0]);
    Corona_M->childSfunctions[0] = (&Corona_M->NonInlinedSFcns.
       childSFunctions[0]);

    /* Level2 S-Function Block: Corona/<S22>/State-Space (
       sfun_spid_contc) */
    {
      SimStruct *rts = Corona_M->childSfunctions[0];

      /* timing info */
      time_T *sfcnPeriod = Corona_M->NonInlinedSFcns.Sfcn0.
         sfcnPeriod;
      time_T *sfcnOffset = Corona_M->NonInlinedSFcns.Sfcn0.
         sfcnOffset;
      int_T *sfcnTsMap = Corona_M->NonInlinedSFcns.Sfcn0.sfcnTsMap;
      (void) memset((void*)sfcnPeriod, 0,
                    sizeof(time_T)*1);
      (void) memset((void*)sfcnOffset, 0,
                    sizeof(time_T)*1);
      ssSetSampleTimePtr(rts, &sfcnPeriod[0]);
      ssSetOffsetTimePtr(rts, &sfcnOffset[0]);
      ssSetSampleTimeTaskIDPtr(rts, sfcnTsMap);

      {
        ssSetBlkInfo2Ptr(rts, &Corona_M->NonInlinedSFcns.blkInfo2
           [0]);
      }

      _ssSetBlkInfo2PortInfo2Ptr(rts,
        &Corona_M->NonInlinedSFcns.inputOutputPortInfo2[0]);

      /* Set up the mdlInfo pointer */
      ssSetRTWSfcnInfo(rts, Corona_M->sfcnInfo);

      /* Allocate memory of model methods 2 */
      {
        ssSetModelMethods2(rts, &Corona_M->NonInlinedSFcns.methods2
           [0]);
      }

      /* Allocate memory of model methods 3 */
      {
        ssSetModelMethods3(rts, &Corona_M->NonInlinedSFcns.methods3
           [0]);
      }

      /* Allocate memory of model methods 4 */
      {
        ssSetModelMethods4(rts, &Corona_M->NonInlinedSFcns.methods4
           [0]);
```

```
    }

    /* Allocate memory for states auxilliary information */
    {
      ssSetStatesInfo2(rts, &Corona_M->NonInlinedSFcns.statesInfo2
          [0]);
      ssSetPeriodicStatesInfo(rts,
        &Corona_M->NonInlinedSFcns.periodicStatesInfo[0]);
    }

    /* inputs */
    {
      _ssSetNumInputPorts(rts, 2);
      ssSetPortInfoForInputs(rts,
        &Corona_M->NonInlinedSFcns.Sfcn0.inputPortInfo[0]);
      ssSetPortInfoForInputs(rts,
        &Corona_M->NonInlinedSFcns.Sfcn0.inputPortInfo[0]);
      _ssSetPortInfo2ForInputUnits(rts,
        &Corona_M->NonInlinedSFcns.Sfcn0.inputPortUnits[0]);
      ssSetInputPortUnit(rts, 0, 0);
      ssSetInputPortUnit(rts, 1, 0);
      _ssSetPortInfo2ForInputCoSimAttribute(rts,
        &Corona_M->NonInlinedSFcns.Sfcn0.inputPortCoSimAttribute
          [0]);
      ssSetInputPortIsContinuousQuantity(rts, 0, 0);
      ssSetInputPortIsContinuousQuantity(rts, 1, 0);

      /* port 0 */
      {
        real_T const **sfcnUPtrs = (real_T const **)
          &Corona_M->NonInlinedSFcns.Sfcn0.UPtrs0;
        sfcnUPtrs[0] = &Corona_B.DC;
        ssSetInputPortSignalPtrs(rts, 0, (InputPtrsType)&sfcnUPtrs
          [0]);
        _ssSetInputPortNumDimensions(rts, 0, 1);
        ssSetInputPortWidthAsInt(rts, 0, 1);
      }

      /* port 1 */
      {
        real_T const **sfcnUPtrs = (real_T const **)
          &Corona_M->NonInlinedSFcns.Sfcn0.UPtrs1;
        sfcnUPtrs[0] = &Corona_B.DataTypeConversion;
        sfcnUPtrs[1] = &Corona_B.Step;
        ssSetInputPortSignalPtrs(rts, 1, (InputPtrsType)&sfcnUPtrs
          [0]);
        _ssSetInputPortNumDimensions(rts, 1, 1);
        ssSetInputPortWidthAsInt(rts, 1, 2);
      }
    }

    /* outputs */
```

```
    {
      ssSetPortInfoForOutputs(rts,
        &Corona_M->NonInlinedSFcns.Sfcn0.outputPortInfo[0]);
      ssSetPortInfoForOutputs(rts,
        &Corona_M->NonInlinedSFcns.Sfcn0.outputPortInfo[0]);
      _ssSetNumOutputPorts(rts, 2);
      _ssSetPortInfo2ForOutputUnits(rts,
        &Corona_M->NonInlinedSFcns.Sfcn0.outputPortUnits[0]);
      ssSetOutputPortUnit(rts, 0, 0);
      ssSetOutputPortUnit(rts, 1, 0);
      _ssSetPortInfo2ForOutputCoSimAttribute(rts,
        &Corona_M->NonInlinedSFcns.Sfcn0.outputPortCoSimAttribute
            [0]);
      ssSetOutputPortIsContinuousQuantity(rts, 0, 0);
      ssSetOutputPortIsContinuousQuantity(rts, 1, 0);

      /* port 0 */
      {
        _ssSetOutputPortNumDimensions(rts, 0, 1);
        ssSetOutputPortWidthAsInt(rts, 0, 6);
        ssSetOutputPortSignal(rts, 0, ((real_T *) Corona_B.
            StateSpace_o1));
      }

      /* port 1 */
      {
        _ssSetOutputPortNumDimensions(rts, 1, 1);
        ssSetOutputPortWidthAsInt(rts, 1, 4);
        ssSetOutputPortSignal(rts, 1, ((real_T *) Corona_B.
            StateSpace_o2));
      }
    }

    /* states */
    ssSetContStates(rts, &Corona_X.StateSpace_CSTATE[0]);
    ssSetContStateDisabled(rts, &Corona_XDis.StateSpace_CSTATE[0])
       ;

    /* path info */
    ssSetModelName(rts, "State-Space");
    ssSetPath(rts, "Corona/powergui1/EquivalentModel1/State-Space"
       );
    ssSetRTModel(rts,Corona_M);
    ssSetParentSS(rts, (NULL));
    ssSetRootSS(rts, rts);
    ssSetVersion(rts, SIMSTRUCT_VERSION_LEVEL2);

    /* parameters */
    {
      mxArray **sfcnParams = (mxArray **)
        &Corona_M->NonInlinedSFcns.Sfcn0.params;
      ssSetSFcnParamsCount(rts, 10);
```

```
      ssSetSFcnParamsPtr(rts, &sfcnParams[0]);
      ssSetSFcnParam(rts, 0, (mxArray*)Corona_P.StateSpace_P1_Size
          );
      ssSetSFcnParam(rts, 1, (mxArray*)Corona_P.StateSpace_P2_Size
          );
      ssSetSFcnParam(rts, 2, (mxArray*)Corona_P.StateSpace_P3_Size
          );
      ssSetSFcnParam(rts, 3, (mxArray*)Corona_P.StateSpace_P4_Size
          );
      ssSetSFcnParam(rts, 4, (mxArray*)Corona_P.StateSpace_P5_Size
          );
      ssSetSFcnParam(rts, 5, (mxArray*)Corona_P.StateSpace_P6_Size
          );
      ssSetSFcnParam(rts, 6, (mxArray*)Corona_P.StateSpace_P7_Size
          );
      ssSetSFcnParam(rts, 7, (mxArray*)Corona_P.StateSpace_P8_Size
          );
      ssSetSFcnParam(rts, 8, (mxArray*)Corona_P.StateSpace_P9_Size
          );
      ssSetSFcnParam(rts, 9, (mxArray*)Corona_P.
          StateSpace_P10_Size);
    }

    /* work vectors */
    ssSetRWork(rts, (real_T *) &Corona_DW.StateSpace_RWORK[0]);
    ssSetIWork(rts, (int_T *) &Corona_DW.StateSpace_IWORK[0]);
    ssSetPWork(rts, (void **) &Corona_DW.StateSpace_PWORK[0]);

    {
      struct _ssDWorkRecord *dWorkRecord = (struct _ssDWorkRecord
          *)
        &Corona_M->NonInlinedSFcns.Sfcn0.dWork;
      struct _ssDWorkAuxRecord *dWorkAuxRecord = (struct
          _ssDWorkAuxRecord *)
        &Corona_M->NonInlinedSFcns.Sfcn0.dWorkAux;
      ssSetSFcnDWork(rts, dWorkRecord);
      ssSetSFcnDWorkAux(rts, dWorkAuxRecord);
      ssSetNumDWorkAsInt(rts, 4);

      /* MODE */
      ssSetDWorkWidthAsInt(rts, 0, 3);
      ssSetDWorkDataType(rts, 0,SS_INTEGER);
      ssSetDWorkComplexSignal(rts, 0, 0);
      ssSetDWork(rts, 0, &Corona_DW.StateSpace_MODE[0]);

      /* RWORK */
      ssSetDWorkWidthAsInt(rts, 1, 2);
      ssSetDWorkDataType(rts, 1,SS_DOUBLE);
      ssSetDWorkComplexSignal(rts, 1, 0);
      ssSetDWork(rts, 1, &Corona_DW.StateSpace_RWORK[0]);

      /* IWORK */
```

```
        ssSetDWorkWidthAsInt(rts, 2, 23);
765     ssSetDWorkDataType(rts, 2,SS_INTEGER);
        ssSetDWorkComplexSignal(rts, 2, 0);
        ssSetDWork(rts, 2, &Corona_DW.StateSpace_IWORK[0]);

        /* PWORK */
770     ssSetDWorkWidthAsInt(rts, 3, 22);
        ssSetDWorkDataType(rts, 3,SS_POINTER);
        ssSetDWorkComplexSignal(rts, 3, 0);
        ssSetDWork(rts, 3, &Corona_DW.StateSpace_PWORK[0]);
      }
775
      ssSetModeVector(rts, (int_T *) &Corona_DW.StateSpace_MODE[0]);

      /* registration */
      sfun_spid_contc(rts);
780   sfcnInitializeSizes(rts);
      sfcnInitializeSampleTimes(rts);

      /* adjust sample time */
      ssSetSampleTime(rts, 0, 0.0);
785   ssSetOffsetTime(rts, 0, 0.0);
      sfcnTsMap[0] = 0;

      /* set compiled values of dynamic vector attributes */
      ssSetNumNonsampledZCsAsInt(rts, 0);
790
      /* Update connectivity flags for each port */
      _ssSetInputPortConnected(rts, 0, 1);
      _ssSetInputPortConnected(rts, 1, 1);
      _ssSetOutputPortConnected(rts, 0, 1);
795   _ssSetOutputPortConnected(rts, 1, 1);
      _ssSetOutputPortBeingMerged(rts, 0, 0);
      _ssSetOutputPortBeingMerged(rts, 1, 0);

      /* Update the BufferDstPort flags for each input port */
800   ssSetInputPortBufferDstPort(rts, 0, -1);
      ssSetInputPortBufferDstPort(rts, 1, -1);
    }
  }

805 /* Matfile logging */
  rt_StartDataLoggingWithStartTime(Corona_M->rtwLogInfo, 0.0,
     rtmGetTFinal
    (Corona_M), Corona_M->Timing.stepSize0, (&rtmGetErrorStatus(
       Corona_M)));

  /* SetupRuntimeResources for Scope: '<Root>/VOLTAGE AT THE POINT'
     */
810 {
    RTWLogSignalInfo rt_ScopeSignalInfo;
    static int_T rt_ScopeSignalWidths[] = { 1 };
```

```
    static int_T rt_ScopeSignalNumDimensions[] = { 1 };
815
    static int_T rt_ScopeSignalDimensions[] = { 1 };

    static void *rt_ScopeCurrSigDims[] = { (NULL) };

820 static int_T rt_ScopeCurrSigDimsSize[] = { 4 };

    static const char_T *rt_ScopeSignalLabels[] = { "" };

    static char_T rt_ScopeSignalTitles[] = "";
825 static int_T rt_ScopeSignalTitleLengths[] = { 0 };

    static boolean_T rt_ScopeSignalIsVarDims[] = { 0 };

    static int_T rt_ScopeSignalPlotStyles[] = { 0 };
830
    BuiltInDTypeId dTypes[1] = { SS_DOUBLE };

    static char_T rt_ScopeBlockName[] = "Corona/VOLTAGE AT THE POINT
       ";
    static int_T rt_ScopeFrameData[] = { 0 };
835
    static RTWPreprocessingFcnPtr
       rt_ScopeSignalLoggingPreprocessingFcnPtrs[] =
      {
      (NULL)
    };
840
    rt_ScopeSignalInfo.numSignals = 1;
    rt_ScopeSignalInfo.numCols = rt_ScopeSignalWidths;
    rt_ScopeSignalInfo.numDims = rt_ScopeSignalNumDimensions;
    rt_ScopeSignalInfo.dims = rt_ScopeSignalDimensions;
845 rt_ScopeSignalInfo.isVarDims = rt_ScopeSignalIsVarDims;
    rt_ScopeSignalInfo.currSigDims = rt_ScopeCurrSigDims;
    rt_ScopeSignalInfo.currSigDimsSize = rt_ScopeCurrSigDimsSize;
    rt_ScopeSignalInfo.dataTypes = dTypes;
    rt_ScopeSignalInfo.complexSignals = (NULL);
850 rt_ScopeSignalInfo.frameData = rt_ScopeFrameData;
    rt_ScopeSignalInfo.preprocessingPtrs =
      rt_ScopeSignalLoggingPreprocessingFcnPtrs;
    rt_ScopeSignalInfo.labels.cptr = rt_ScopeSignalLabels;
    rt_ScopeSignalInfo.titles = rt_ScopeSignalTitles;
855 rt_ScopeSignalInfo.titleLengths = rt_ScopeSignalTitleLengths;
    rt_ScopeSignalInfo.plotStyles = rt_ScopeSignalPlotStyles;
    rt_ScopeSignalInfo.blockNames.cptr = (NULL);
    rt_ScopeSignalInfo.stateNames.cptr = (NULL);
    rt_ScopeSignalInfo.crossMdlRef = (NULL);
860 rt_ScopeSignalInfo.dataTypeConvert = (NULL);
    Corona_DW.VOLTAGEATTHEPOINT_PWORK.LoggedData =
       rt_CreateStructLogVar(
```

```
      Corona_M->rtwLogInfo,
      0.0,
      rtmGetTFinal(Corona_M),
      Corona_M->Timing.stepSize0,
      (&rtmGetErrorStatus(Corona_M)),
      "coronavoltage",
      1,
      0,
      1,
      1.0E-5,
      &rt_ScopeSignalInfo,
      rt_ScopeBlockName);
    if (Corona_DW.VOLTAGEATTHEPOINT_PWORK.LoggedData == (NULL))
      return;
  }

  /* Start for Constant: '<S12>/DC' */
  Corona_B.DC = Corona_P.DCVoltageSource1_Amplitude;

  /* Start for S-Function (sfun_spid_contc): '<S22>/State-Space' */
  /* Level2 S-Function Block: '<S22>/State-Space' (sfun_spid_contc)
     */
  {
    SimStruct *rts = Corona_M->childSfunctions[0];
    sfcnStart(rts);
    if (ssGetErrorStatus(rts) != (NULL))
      return;
  }

  {
    real_T tmin;
    int32_T r;
    int32_T t;
    uint32_T tseed;

    /* InitializeConditions for S-Function (sfun_spid_contc): '<S22
       >/State-Space' */
    /* Level2 S-Function Block: '<S22>/State-Space' (sfun_spid_contc
       ) */
    {
      SimStruct *rts = Corona_M->childSfunctions[0];
      sfcnInitializeConditions(rts);
      if (ssGetErrorStatus(rts) != (NULL))
        return;
    }

    /* InitializeConditions for UniformRandomNumber: '<Root>/RABDOM
       NUMBER GENERATOR' */
    tmin = floor(Corona_P.RABDOMNUMBERGENERATOR_Seed);
    if (rtIsNaN(tmin) || rtIsInf(tmin)) {
      tmin = 0.0;
    } else {
```

```
910       tmin = fmod(tmin, 4.294967296E+9);
      }

      if (tmin < 0.0) {
        tseed = (uint32_T)-(int32_T)(uint32_T)-tmin;
915   } else {
        tseed = (uint32_T)tmin;
      }

      r = (int32_T)(tseed >> 16U);
920   t = (int32_T)(tseed & 32768U);
      tseed = ((((tseed - ((uint32_T)r << 16U)) + (uint32_T)t) << 16U)
          + (uint32_T)
              t) + (uint32_T)r;
      if (tseed < 1U) {
        tseed = 1144108930U;
925   } else if (tseed > 2147483646U) {
        tseed = 2147483646U;
      }

      Corona_DW.RandSeed = tseed;
930   tmin = Corona_P.RABDOMNUMBERGENERATOR_Minimum;
      Corona_DW.RABDOMNUMBERGENERATOR_NextOutpu =
        (Corona_P.RABDOMNUMBERGENERATOR_Maximum - tmin) *
        rt_urand_Upu32_Yd_f_pw_snf(&Corona_DW.RandSeed) + tmin;

935   /* End of InitializeConditions for UniformRandomNumber: '<Root>/
         RABDOM NUMBER GENERATOR' */
    }
  }

  /* Model terminate function */
940 void Corona_terminate(void)
  {
    /* Terminate for S-Function (sfun_spid_contc): '<S22>/State-Space'
       */
    /* Level2 S-Function Block: '<S22>/State-Space' (sfun_spid_contc)
       */
    {
945   SimStruct *rts = Corona_M->childSfunctions[0];
      sfcnTerminate(rts);
    }
  }
```

Appendix B

Balanced PD detection circuit

B.1 Balanced PD detection circuit sensitivity tests

Balanced PD detection circuit feasibility

PD calibrator pulse measured with a digital oscilloscope

Figure B.1.1 below shows a PD calibrator connected directly to the oscilloscope to measure the PD pulse for reference purposes.

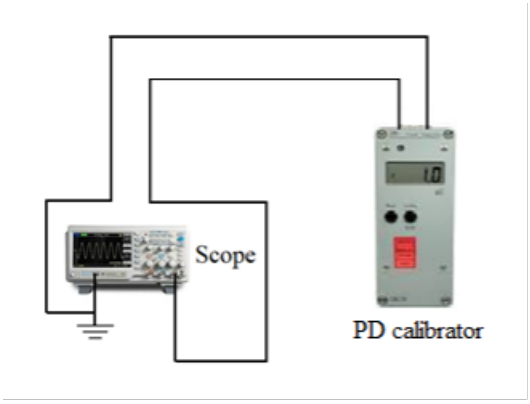


Figure B.1.1: PD calibrator connected to an oscilloscope

Figure B.1.2 below shows the Output waveform of the PD calibrator, 1.1 V(peak) at 100 pC.

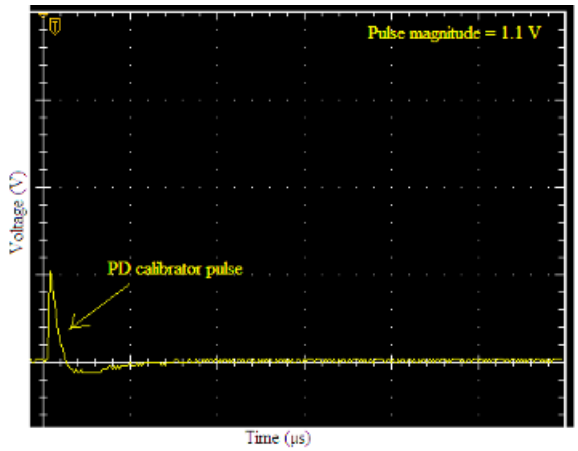


Figure B.1.2: PD calibrator output waveform

Figure B.1.3 below shows an optimized circuit with a PD calibrator connected to measure the output waveform compared to the waveform obtained with a PD calibrator directly connected to the oscilloscope.

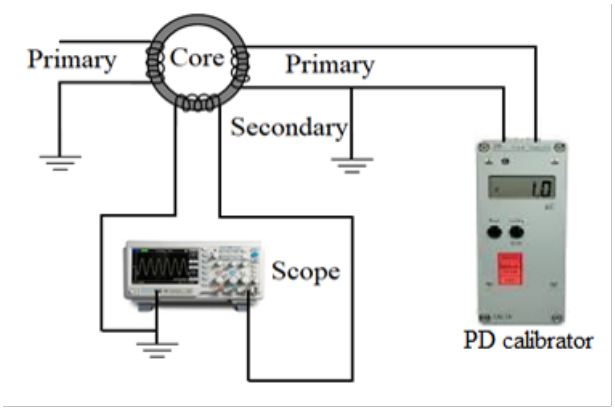


Figure B.1.3: PD calibrator signal connected to the optimized circuit

Figure B.1.4 below shows the optimized response from the PD detection circuit with the PD calibrator connected, as shown in Figure B.1.3. The measured output signal from the toroidal core is 1060 mV at 100 pC.

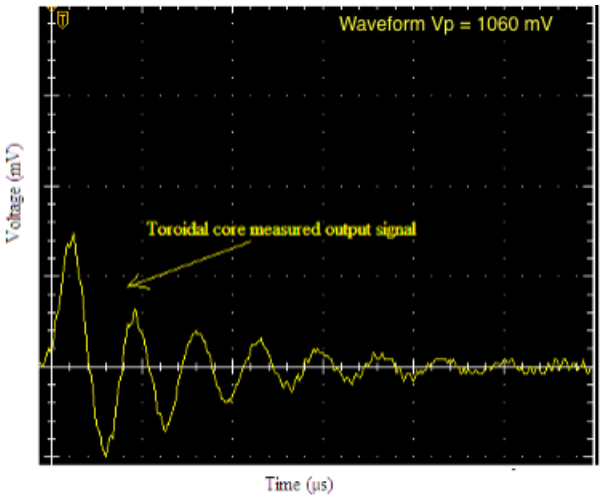
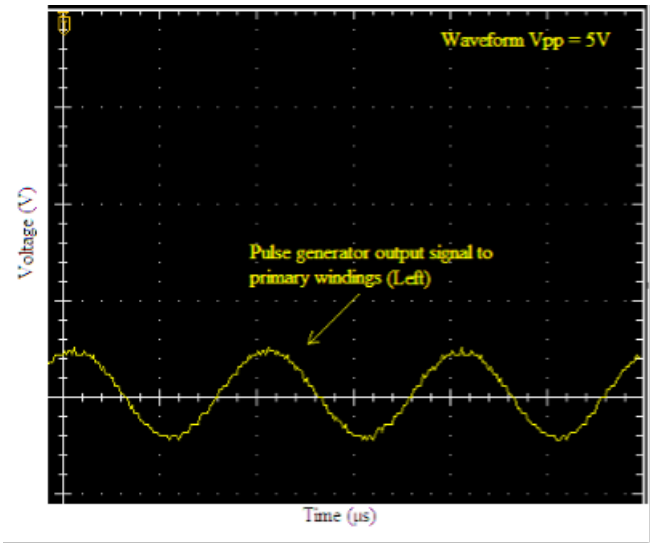


Figure B.1.4: PD detection circuit optimized response waveform

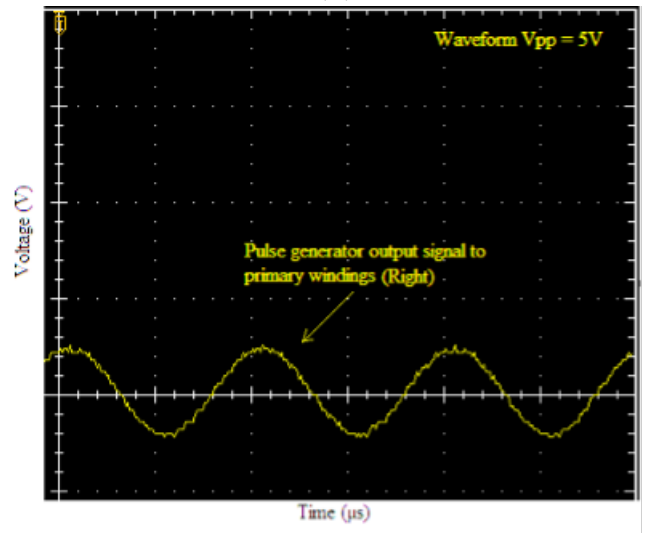
Toroidal core signal cancelation confirmation using a pulse generator

Figure B.1.5 below shows the PD detection circuit with a pulse generator connected to the primary windings with an identical pulse wave from the pulse generator to confirm signal cancelation.

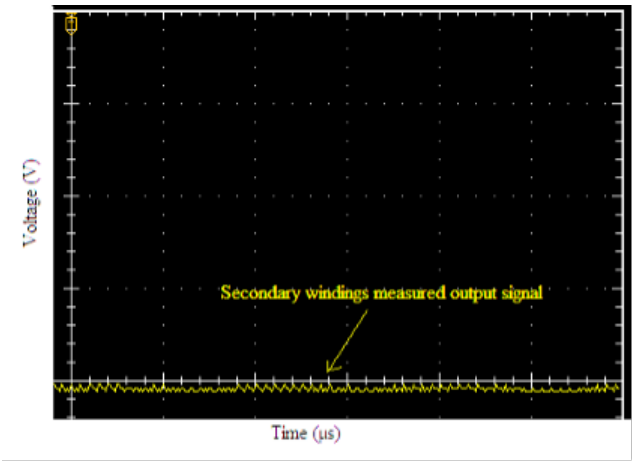
Figure B.1.5 also shows primary input signals (a) and (b) from a pulse generator and a resultant waveform (c). The consequent waveform proves the circuit.



(a)



(b)



(c)

Figure B.1.5: PD detection circuit identical input signal (a and b) and measured output signal (c)

B.2 Toroidal core datasheet

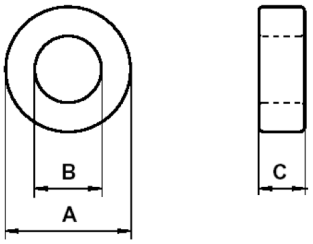
MAGNETICS ®
A Division of Spang & Company

Specification for:

ZW43615TC

110 Delta Drive
Pittsburgh, PA 15238
Phone: 412/696-1333
Fax: 412/696-0333
Email:magnetics@spang.com

DIMENSIONS



(mm)	Uncoated Nominal:	Coated Min:	Coated Max:
O.D. (A)	36	35.36	37.14
I.D. (B)	23	21.86	23.64
Ht. (C)	15	14.75	15.75

Eff. Parameters		
A_e mm^2	l_e mm	V_e mm^3
95.9	89.6	8596

INDUCTANCE

AL value (nH/T^2)	**Test conditions**
13400 ± 30%	10 kHz, 0.5 mT (For N = 5, use 0.42 mA), 25°C

ELECTRICAL LOSSES

tan δ / μ_i	**Production lot limit Average**	**Test conditions**	
≤ 4.1·10^{-6}	≤ 3.5 ·10^{-6}	10 kHz	0.5 mT, 25ºC
≤ 60·10^{-6}	≤ 50·10^{-6}	100 kHz	

COATING

Epoxy rated for 200°C continuous operation.
Voltage breakdown rating 2000 V Min Wire-to-Wire.

NOTE

Spec. Modifications	**Previous**	**Revised**
2005.11.08	Breakdown voltage > 1,000 V LF: General W material	Breakdown voltage > 2,000 V LF: Detail as indicated

Appendix C

Cavity PD

C.1 Cavity PD Experimental Results

These are the experimental cavity PD pulses obtained during impulse voltage shorts. On the rise time of the impulse voltage shorts, main discharges ((a) - (d)) were observed; however, on the fall time, the reverse discharges were only observed on the oscillographs ((e) - (h)) together with the main discharges on a small scale due to small PD magnitudes of the reverse discharges.

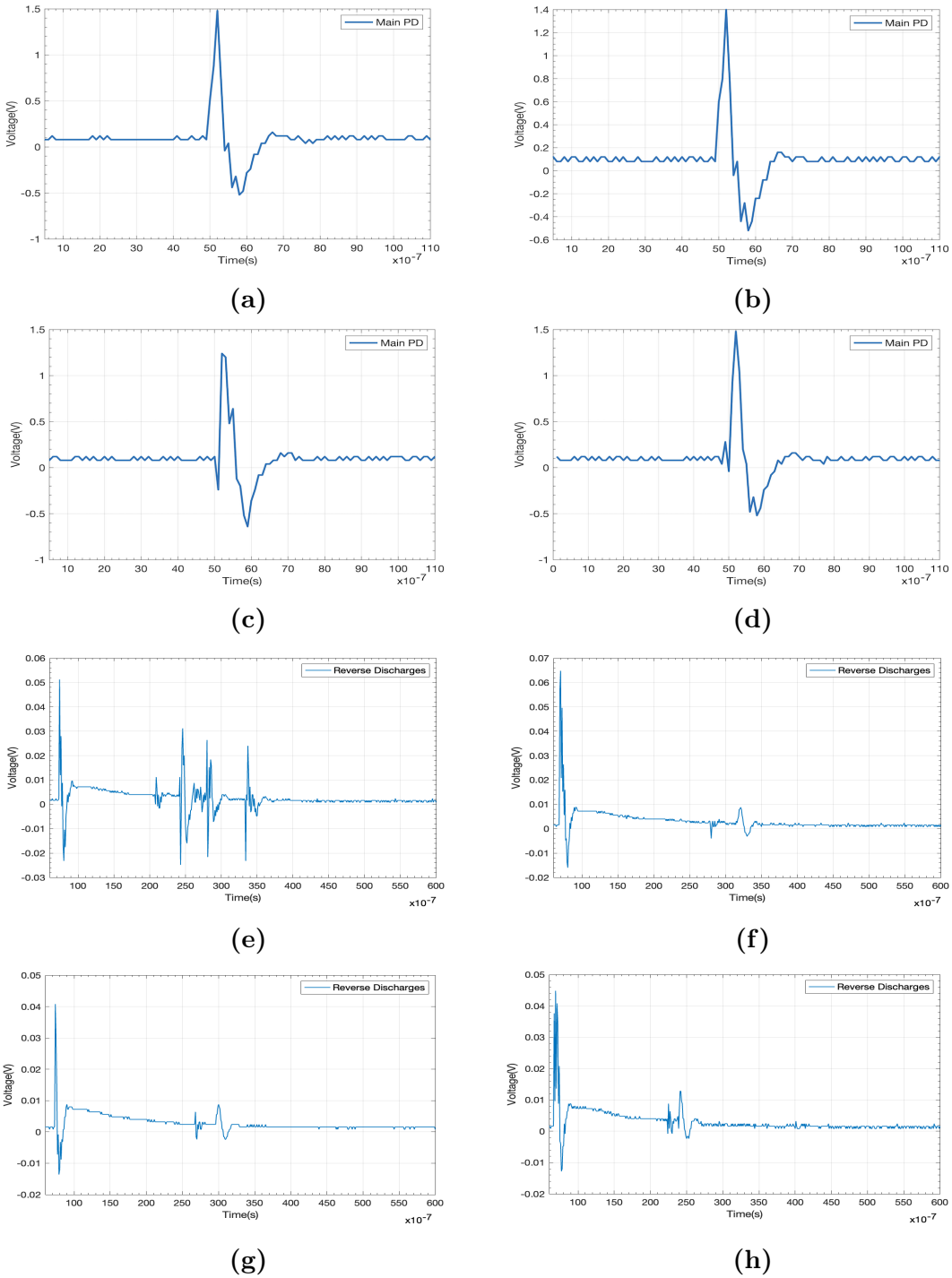


Figure C.1.1: Cavity PD Experimental Results

Appendix D

Corona discharges

D.1 Corona PD Experimental Results

These are the experimental corona pulses observed during impulse voltage shorts. Corona discharges ((a) - (d)) are the positive corona pulses, and corona discharges ((e) - (h)) are the negative corona pulses.

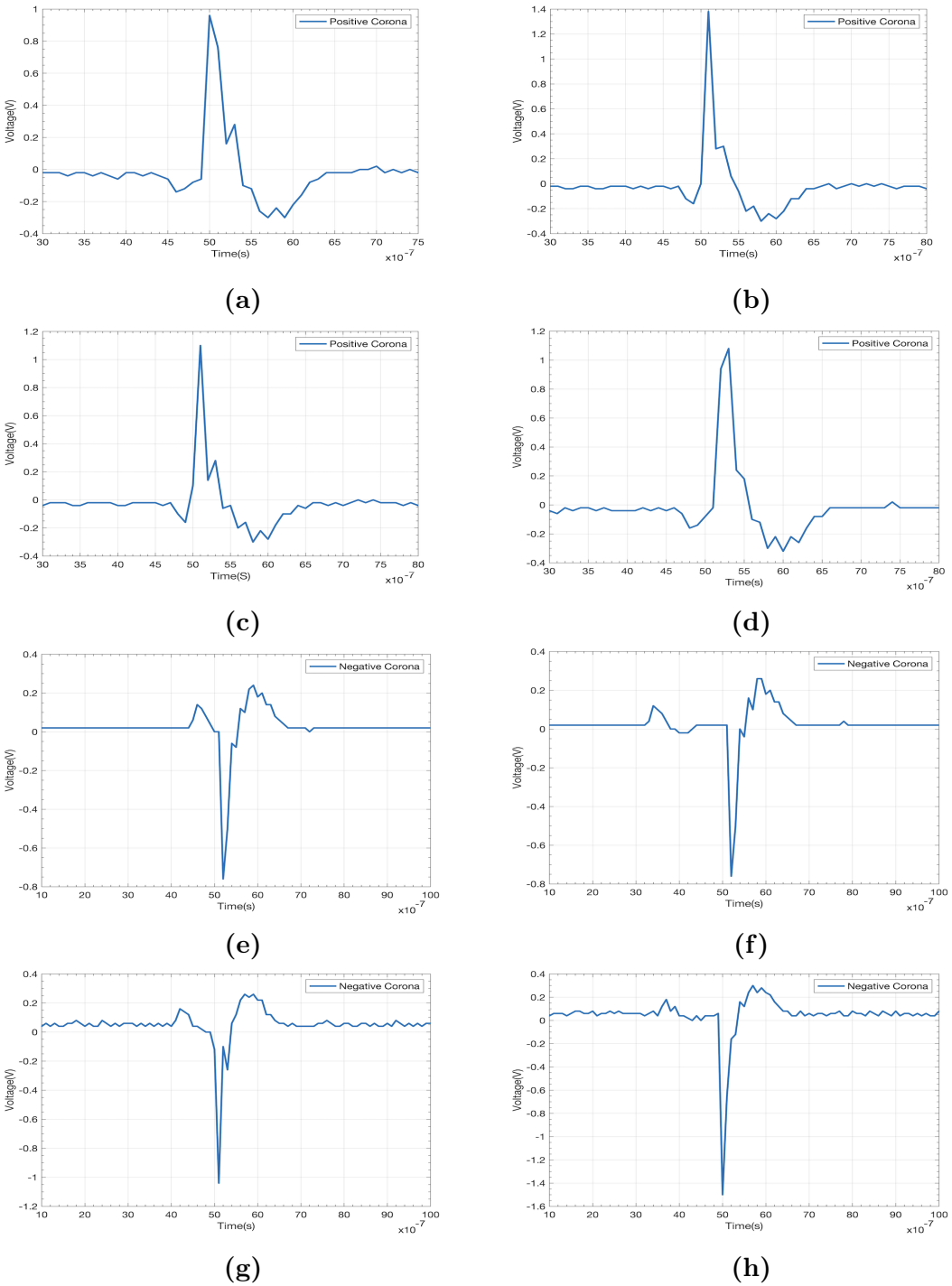


Figure D.1.1: Corona Experimental Results

Appendix E

SAUPEC conference paper

E.1 This appendix comprises a conference paper presented at the 31st Southern African Universities Power Engineering Conference (SAUPEC), January 24-26, 2023

2023 SAUPEC Conference, Johannesburg, South Africa, January 24-26, 2023

Characterization of Cavity PD Under Impulse Voltage in Polyethylene Insulation

Tibello Vincent Mphuti
School of Electrical & Information Engineering
University of the Witwatersrand
Johannesburg, South Africa
1501593@students.wits.ac.za

Cuthbert Nyamupangedengu
School of Electrical & Information Engineering
University of the Witwatersrand
Johannesburg, South Africa
cuthbert.nyamupangedengu@wits.ac.za

2023 31st Southern African Universities Power Engineering Conference (SAUPEC) | 979-8-3503-3186-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/SAUPEC57889.2023.10057881

***Abstract*— Understanding PD (Partial Discharge) phenomena under impulse voltage is important knowledge for the improvement of reliability of HV (High Voltage) equipment. PD in HV equipment can cause severe insulation deterioration resulting in breakdown, production loss and high cost of repairs. HV equipment insulation can fail due to PD activities which are initiated by impulse voltages such as lightning and switching. The latter has been exacerbated by load shedding switching operations in South Africa. PD detection under AC (Alternating Current) frequency voltage conditions is well developed, however there is inadequate knowledge in PD detection and diagnosis under non-power frequencies. The challenge is to enable PD detection during impulse voltage, due to the overlaps between the impulse voltage and the PD frequencies. In the present work, balanced PD detection circuit is implemented to suppress impulse voltage interference for the purpose of detecting cavity PD under impulse voltage. The measurement results show that PD pulses occur on specific locations of the impulse test voltage waveform and is consistent with similar results in the literature.**

Keywords — Cavity PD, Impulse voltage, Balanced PD detection circuit

I. INTRODUCTION

PD is localized electrical discharge activity that partly span punctured insulation between conductors. PD occurs if the dielectric strength of an insulation material is exceeded under normal voltage application and abnormal voltages due to the intensity of the localized ionization, which can lead to insulation breakdown [1].

A PD detection system is an important tool for diagnosis and analysis of the dielectric condition of insulation material in HV power systems. Typical partial discharges are corona (air ionization) due to non-uniform field on sharp end of insulation or connection, surface discharges due to interfacing of different insulation material, cavity discharges due to voids formed in solid or liquid insulation material and treeing as a result of electrical stress and moisture in insulation material. PD can continuously degrade the insulation material [2]. Power plant equipment do not only experience operating voltages, but also encounter lightning impulse and switching impulses [1]. PD detection under AC frequency voltage is well developed, however there is limited information on the efficacy of PD detection and characteristics under impulse voltages [1, 3]. PD measurement system that can effectively detect and measure PD's on incipient defects under impulse voltage is of benefit in the understanding of PD phenomena. The knowledge is necessary to make informative decision with regards to HV equipment insulation diagnosis. Catastrophic failure in power plant can be mitigated in that regard [4].

It's evident that HV equipment in operation, are not only subjected to AC operating voltage, but also transient voltages such as lightning and switching impulse. The transient voltages can superimpose on the normal AC voltage supply to HV equipment in-service. This phenomenon can result in hidden PD activity which may continue under normal AC operating voltage due to the degradation of stressed insulation caused by nature of PD process initiated by transient voltages [5].

PD measurements under impulse voltage can provide useful information about incipient defects in HV equipment. Therefore, it is considered important to diagnose insulation condition of HV equipment under impulse voltages [6]. The present paper represents a balanced PD detection system for cavity PD detection under impulse voltage.

II. THE EXPERIMENTAL MEASUREMENTS

A. PD defect model

The test specimen is a parallel plate electrode set-up for creating cavity PD, which consists of copper electrodes and sandwiching polyethylene sheets with one sheet having a cavity as shown in Fig 1. One of the common dielectric materials used to fabricate voids is High Density Polyethylene (HDPE) due to its high dielectric strength [7], and is of interest in this study as it simulates the power cable insulation system [8]. The complex relative permittivity is 2.3 on frequency band 1.0 kHz to 30 MHz for HDPE [9]. HDPE has advantages in that is sufficiently flexible and has high impact strength [10]. Fig 2 shows the schematic of the test cell. The test cell is immersed in oil to avoid unwanted surface discharges and flashovers.

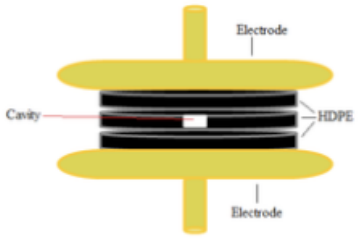


Fig 1: Parallel plate electrode set-up with cavity

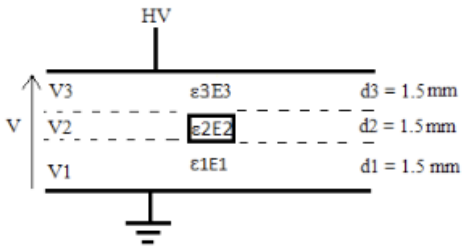


Fig 2: Schematic of test cell

The cavity thickness is 1.5 mm (thickness of a single sheet) with a diameter of 3 mm. At 0.8 bar air pressure, from Paschen curve minimum breakdown voltage is 4.5 kV. With reference to the 3-layer model in Fig 2, and (1) together with the cavity sparkover voltage from the Paschen curve, the voltage across the test object that would initiate PD in the cavity can be determined.

$$E_2 = \frac{V}{\frac{\varepsilon_2}{\varepsilon_1} d1 + \frac{\varepsilon_2}{\varepsilon_2} d2 + \frac{\varepsilon_2}{\varepsilon_3} d3} \quad (1)$$

Where:

E_2: Magnitude of field intensity in air cavity

d_1: Polyethylene thickness

d_2: Air cavity thickness

d3: Polyethylene thickness

ε_1: Polyethylene relative permittivity = 2.3

ε_2: Air relative permittivity = 1

ε_3: Polyethylene relative permittivity = 2.3

V: Applied voltage to the test specimen

The minimum required externally applied voltage to initiate PD in the cavity is therefore calculated using (1) as 8,4 kV

B. The balanced PD detection circuit

PD detection through measurement of the energy emitted by the PD event can give information about its location and source. PD detection circuits are based on the principles of processing energies emitted by PD events. Various PD detection techniques have been exploited with their ability to detect PD activity under impulse voltage. However, PD detection under impulse voltage is relatively challenging. In various ultrawideband PD detection methods such as inductive probe, electromagnetic radiation detection using antenna, capacitive probe and acoustic sensors, the efficacy of the measurements is limited.

Wu et al. [5, 6] investigated the effects of transient voltages on HV cable with an artificial defect using two High Frequency Current Transformer (HFCT) under AC, impulse voltages of different polarity and superimposed voltages. PD activities are reported under longer impulse front time and switching impulse.

Mitra et al. [3] and Densley et al. [11] reported smallest discharges detected, using a bridge PD detection circuit. PD activity behaviour on artificial air-filled cavities of different dimensions in solid insulation under lightning and switching impulse voltage was detected. Kreuger and Shihab [12] investigated PD measurement on a three-core belted power cable using balanced bridge method and reported that different PD results are obtained due to the sensitivity limitations of the measuring circuit.

Hayakawa et al. [13] investigated electrical insulation characteristics of a High Temperature Superconducting cable insulation model. The test object was subjected to AC voltages superimposed with lightning impulse voltage. PD signal was not detectable under ac voltage before the impulse voltage application. PD signal was detected under the subsequent AC voltage after the application of impulse voltage.

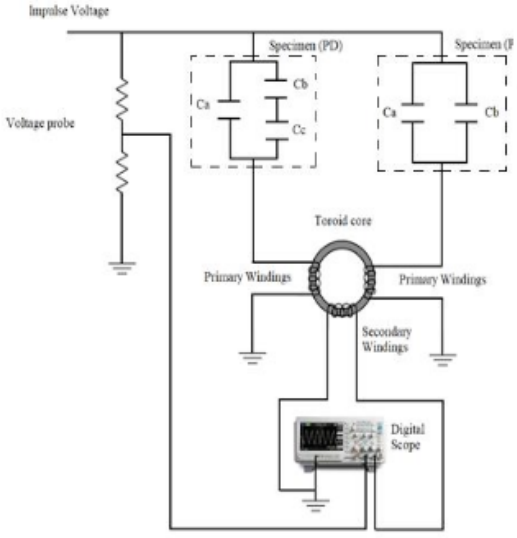
Balanced PD detection circuit is adopted in the present research paper for characterization of cavity PD under impulse voltage. The setup to investigate PD detection under impulse is shown in Fig 3. Table I below shows various PD detection methods and each methods advantage and disadvantages

TABLE I. PD detection methods

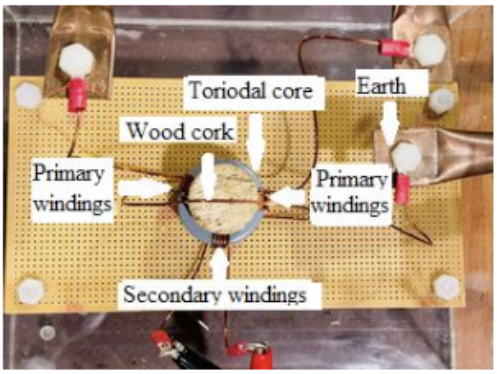
PD detection methods		
Technique	***Pros***	***Cons***
PD current detection through coupling capacitor	Suitable for conventional narrow band PD detection	Not suitable for ultrawideband PD detection
PD current magnetic field through inductive probe	Suitable for ultrawideband PD detection	-Susceptible to impulse voltage interference -Require a filter (suppress PD)
PD transient electric field using capacitive probe	Suitable for ultrawideband PD detection	-Susceptible to impulse voltage interference -Require a filter (suppress PD)
PD event acoustic energy through acoustic sensor	Suitable for ultrawideband PD detection	-Susceptible to impulse voltage interference and external noise -Require an amplifier (PD not origin)

PD detection methods		
Technique	***Pros***	***Cons***
PD electromagnetic radiation through antenna	Suitable for ultrawideband PD detection	-Susceptible to impulse voltage interference and external noise -Require an amplifier (PD not origin)
Balanced bridge	-Suitable for ultrawideband PD detection -Noise cancelation	Need to be balanced to allow cancelation

Generally, the balanced bridge method for PD detection consists of resistors and identical adjustable capacitors [14]. The noise can only be suppressed when the electric bridge is in the balanced state, however the distributed stray parameters can cause the bridge not to be balanced [14]. An RLC filtering circuit in the form of a toroidal core for PD detection can be used. The concept of a balanced bridge method is used in this paper. The proposed balanced PD detection circuit is made of a high permeability ferrite toroid which is for high frequency application. The toroid core comprises of common primary windings on which the test specimen is connected to. PD test specimen and PD free specimen are used. The toroid core consists of an optimized secondary winding on which PD signal is measured using a Tektronix 6 GHz digital scope and displaying both PD magnitudes and applied lightning impulse voltage which is measured via the voltage probe. Fig 3 presents the equivalent circuit and the picture of the balanced PD detection setup.



(a)

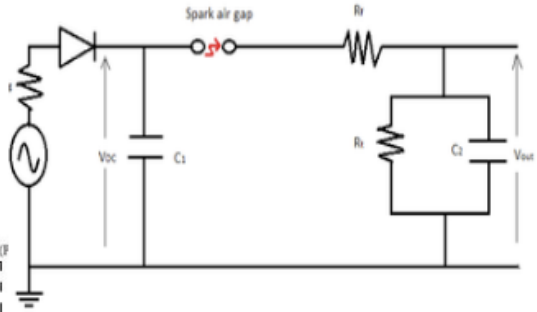


(b)

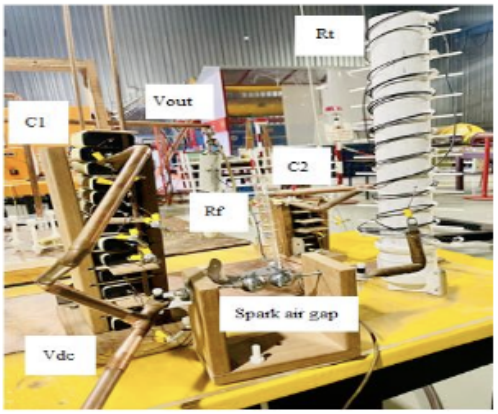
Fig 3: (a) Balanced PD detection equivalent circuit and (b) Balanced PD detection circuit picture.

C. Experimental Procedure

A single stage lightning impulse voltage source used in this research experiment is as shown in Fig 4. The components of the impulse generator are adjustable to produce the desired standard waveforms in accordance with the IEC 60060-1 [15].



(a)



(b)

Fig 4: (a) Single stage impulse voltage generator equivalent circuit diagram, (b) Single stage impulse voltage generator picture.

Impulse voltage shots of incremental peak magnitude are applied on the test specimen until the PD activity is observed. The same impulse voltage magnitude is repeated in applying stress shots to the specimen, while recording PD signals in every event. There are considerable wait period intervals between shots to allow for space charge dispersion in the cavity that would influence the next PD activity. To ensure that the obtained signals are indeed from the defected test specimen, a control experiment was conducted where two defect-free specimens were tested in a balanced mode. The obtained measurement results are presented in Fig 5, where there is no evidence of PD in the defect free setup. The test impulse voltage is sufficiently cancelled out in the balanced bridge.

III. RESULTS AND DISCUSSION

The lightning impulse voltage shown in Fig 5 was measured using a voltage probe connected to a digital oscilloscope. PD detection circuit integrity and efficiency is proved as shown by the balanced PD (free) detection circuit output signal in Fig 5 below. Fig 5 is the background noise benchmark for cavity-free test specimen.

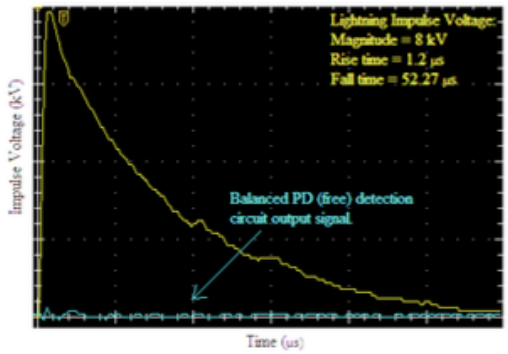


Fig 5: Measured lightning impulse voltage and Balanced PD free detection circuit output signal

A. PD detection and extraction

PD detection and extraction under lightning impulse voltage is achieved using the balanced PD detection circuit and the results are presented below.

Under the same condition of the lightning impulse voltage specification as shown in Fig 5, parallel plate electrode set-up with cavity as shown in Fig 1, was introduced on one leg on the balanced PD detection circuit and the other leg with PD free specimen (HDPE sheets with no cavity). As shown in Fig 6, 7, 8 and 9, PD pulses are detected and recorded using a digital oscilloscope during the application of lighting impulse voltage stress.

Fig 7 also presents typical measured PD relative to the test impulse voltage. Observed are the main discharges during the rise time of the lightning impulse voltage and reverse discharges during the fall time of the lightning impulse voltage. It is evident that the reverse discharges obtained in Fig 7, 8 and 9 during the fall time are influenced by the residual charge because of the space charge developed on the cavity wall due to PD activity during the rising edge of the impulse voltage [16].

From Niemeyer's [17] generalised PD model and the theory of cavity discharge mechanisms, space charges are in the opposite polarity to the initial field, however with the fall time of the lightning impulse voltage being significantly shorter than the dispersion time of the space charge, the resultant voltage established by the space charge can exceed the cavity sparkover voltage and therefore, initiate the reverse PD [16]. Reverse discharges are dependent on the cavity surface conductivity which change with aging condition [16]. The obtained results shown in Figs 6, 7, 8 and 9 are representative of typical measurements obtained. Because of the stochastic nature of PD measurement, event may give slightly different results but within the statistical insignificant variations.

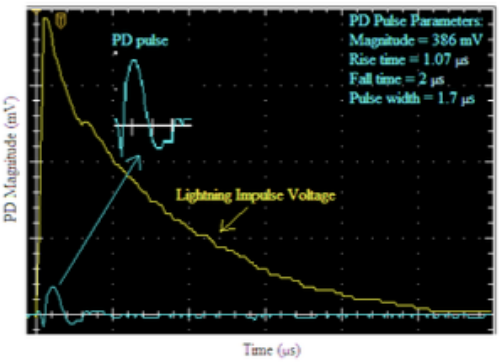


Fig 6: Measured PD pulse obtained during lightning impulse voltage shots

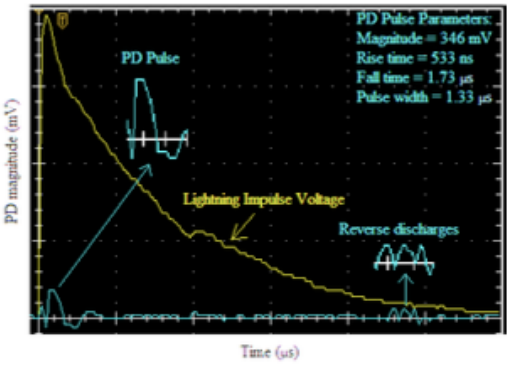


Fig 7: Measured PD pulse and positive reverse discharges because of polarity inversion under lightning impulse voltage

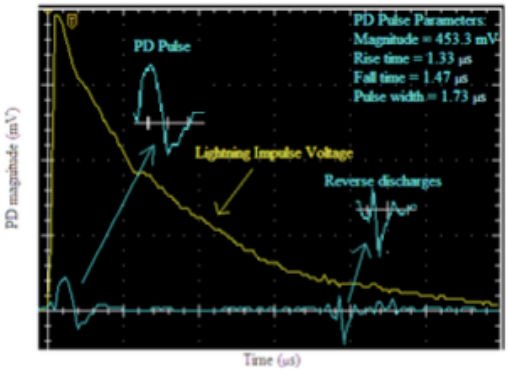


Fig 8: Measured PD pulse and negative reverse discharges because of polarity inversion under lightning impulse voltage

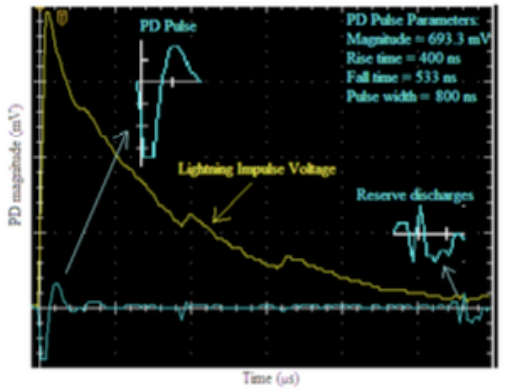


Fig 9: Space charge effect on measured PD pulse and reverse discharge after multiple lightning impulse voltage shots

IV. CONCLUSION

The measurements result for cavity PD under lightning impulse voltage are presented in this paper. Balanced PD detection circuit enabled measurement of PD pulses on the front and tail of the impulse voltage. The observed main discharges and reverse discharges are consistent with similar results in the literature. The role of PD generated space charge in the cavity in influencing discharges, subsequent discharges on the tail of the impulse voltage is evident in the obtained results.

Balanced PD detection circuit is viable for characterization of cavity PD under impulse voltage. Future work will entail comparing of cavities and corona under impulse test voltage.

REFERENCES

[1] Z. Sun, X. Zhao, J. L, Yanming L.I. Experiment Investigation of Partial Discharges under Impulse Voltage. IEEE 9th International Conference on the Properties and Applications of Dielectric Materials, 19-23 July 2009, pp 525-529.

[2] Wadhwa C.L. High Voltage Engineering. New Age International (P) Ltd., Publishers, New Techno Press, third edition, 2010.

[3] G. Mitra., M.M. Sakr, B. Salvage. Detection and measurement of discharges in gaseous cavities in solid dielectrics under impulse voltage conditions. Proc. IEE, Conference on Dielectric and Insulating Materials, Vol 112, No 5, May 1965, pp 1056-1060.

[4] V.R. Garcia-Colon. Partial Discharge Measurement During Impulse Testing. IEEE Conference on Electrical Insulation & Dielectric Phenomena, October 2014

[5] J. Wu, P.V.M. Van Nes, A.R. Mor, J.J. Smit. Partial Discharges in XLPE Insulated Cable under Superimposed Transient Voltages. Electrical Insulation Conference (EIC), 16-19 June 2019, pp 221-225.

[6] J. Wu, A. Rodrigo Mor, P. Van Nes, J.J. Smit. Measuring method for partial discharge in a high voltage cable system subjected to impulse and superimposed voltage under laboratory. Electrical Power and Energy Systems 115 (2020) 105489.2019.

[7] P. H. F. Morshuis. Partial discharges mechanisms. PhD thesis, Delft University of Technology, 1993.

[8] G. Callender, P. L. Lewin. Modeling Partial Discharge Phenomena. IEEE Electrical Insulation Magazine, University of Southampton, Vol 36, No 2, March/April, pp 29-36.

[9] Y. Hasegawa, Y. Ohki, K. Fukunaga, M. Mizuno, K. Sasaki. Complex Permittivity Spectra of Various Insulating Polymers at Ultrawide-Band Frequencies. Electrical Engineering in Japan, Vol 198, No 3, 2017 Translated from Denki Gakkai Ronbunshi, Vol. 135-A, No 2, February 2015, pp. 63–68.

[10] S. Ziad, A. Dabbak, H. A. Illias, B. C. Ang, N. A. A. Latiff, M. Z. H. Makmud. Electrical Properties of Polyethylene/Polypropylene Compounds for High-Voltage Insulation. Energies 2018, pp 1-13, www.mdpi.com/journal/energies. Accessed on 2020/08/22

[11] R.J. Densley, B. Salvage. Partial Discharges in Gaseous Cavities in Solid Dielectric Under Impulse Voltage Conditions. IEEE Transaction on Electrical Insulation, Vol EI-6, No 2, June 1971, pp 54-62.

[12] F.H. Kreuger, S Shihab. Partial discharge measurement on three-core belted power cables. IEEE Transaction on power delivery, Vol 4, No 2, April 1989, pp 927-931.

[13] N. Hayakawa, R. Yamaguchi, Y. Ukai, H. Kojima1, F. Endo, H. Okubo. Partial Discharge Activities under AC/Impulse Superimposed Voltage in LN2/Polypropylene Laminated Paper Insulation System for HTS Cables. 9th European Conference on Applied Superconductivity (EUCAS 09), Journal of Physics: Conference Series 234, 2010.

[14] L. Gang, W. Guangning, T. Demin. The System Architecture of Selective Frequency Balance Measurement for PD. Conference Record of the 2006 IEEE International Symposium on Electrical Insulation, pp 151- 154.

[15] IEC 60060-1. High-voltage test techniques Part 1: General Definitions and Test Requirements. 2010

[16] L. Gui, Y. Mi, S. Deng, L. Liu, X. Ge, W. Ouyang. Partial Discharge Characteristics of an Air Gap Defect in the EpoxyResin of a Saturable Reactor under an Exponential Decay Pulse Voltage. IET Review Copy Only, pp 1-18.

[17] L. Niemeyer. A Generalized Apporach to Partial Discharge Modeling. IEEE Transaction on Dielectric and Electrical Insulation, Vol. 2, No. 4, August 1995.