TEST ANALYSIS & FAULT SIMULATION OF MICROFLUIDIC SYSTEMS

being a Thesis submitted for the Degree of

DOCTOR OF PHILOSOPHY

IN THE UNIVERSITY OF HULL

BY

THOMAS OLIVER MYERS

BEng (Hons)

OCTOBER 2010

ABSTRACT

This work presents a design, simulation and test methodology for microfluidic systems, with particular focus on simulation for test. A Microfluidic Fault Simulator (MFS) has been created based around COMSOL which allows a fault-free system model to undergo fault injection and provide test measurements. A post MFS test analysis procedure is also described.

A range of fault-free system simulations have been cross-validated to experimental work to gauge the accuracy of the fundamental simulation approach prior to further investigation and development of the simulation and test procedure.

A generic mechanism, termed a fault block, has been developed to provide fault injection and a method of describing a low abstraction behavioural fault model within the system. This technique has allowed the creation of a fault library containing a range of different microfluidic fault conditions. Each of the fault models has been cross-validated to experimental conditions or published results to determine their accuracy.

Two test methods, namely, impedance spectroscopy and Levich electro-chemical sensors have been investigated as general methods of microfluidic test, each of which has been shown to be sensitive to a multitude of fault. Each method has successfully been implemented within the simulation environment and each cross-validated by first-hand experimentation or published work.

A test analysis procedure based around the Neyman-Pearson criterion has been developed to allow a probabilistic metric for each test applied for a given fault condition, providing a quantitive assessment of each test. These metrics are used to analyse the sensitivity of each test method, useful when determining which tests to employ in the final system. Furthermore, these probabilistic metrics may be combined to provide a fault coverage metric for the complete system.

The complete MFS method has been applied to two system cases studies; a hydrodynamic “Y” channel and a flow cytometry system for prognosing head and neck cancer.

Decision trees are trained based on the test measurement data and fault conditions as a means of classifying the systems fault condition state. The classification rules created by the decision trees may be displayed graphically or as a set of rules which can be loaded into test instrumentation. During the course of this research a high voltage power supply instrument has been developed to aid electro-osmotic experimentation and an impedance spectrometer to provide embedded test.

ACKNOWLEDGMENTS

I would like to expressly thank my supervisors Dr Ian Bell and Dr Anthony Wilkinson and my PhD Panel Chair Dr Jim Gilbert, for their help, advice and occasional awkward questioning.

I would like to thank my colleagues in the Department of Chemistry, Professor Steve Haswell, Dr Pete Docker and Mrs. Jane Woods, for their “Microfluidic Science” help and advice, access to equipment and “live” applications.

Most importantly I would like to thank my Mum, Dad and Brother Scott, for without their continued support my PhD would not have been possible.

And to my friends who have provided light relief making the PhD process bearable!

CONTENTS

TABLE OF FIGURES

LIST OF TABLES

Science is a wonderful thing if one does not have to earn one's living at it. ~Albert Einstein

Chapter 1 Introduction

The miniaturisation of chemical and analytical biological systems has undergone significant advances in recent years driven primarily by superior performance in chemical terms; such as, processing efficiency, process control, combining of analytical steps (micro Total Analysis Systems μTAS), sensor integration and low sample consumption. Moreover, the term *microfluidics* is one which is beginning to enter popular science [1], [2]. There is little doubt that lab-on-a-chip (LOC) technology has potential in the most necessary parts of modern life, however relatively few applications are leaving the laboratory and entering the outside world [3]. This slow rate of wide exploitation and commercialisation is not unexpected [1], but recently Becker [4] has concluded that the outlook for development of the LOC market in the next few years is positive. More extensive deployment of LOC technology will require reliable and easy to use products, which in turn will depend on effective manufacture, test and quality assurance processes; of these tests is our research focus.

Wider deployment of LOC technology will only be achieved by decreasing the reliance on external equipment and operators with LOC expertise. A system design approach will help achieve this, integrating chemistry, fluidics, sensing, electronics and software to optimise performance, cost, manufacturability and reliability. This applies to both 'disposable-with-reader' LOC platforms (e.g. [2]) and systems employing microfluidic reactors in more continuous use (e.g. [5]). Test development will be part of this process.

Products must be tested during production where "perfect" quality cannot be guaranteed by manufacturing processes, or where regulation and certification specifies particular testing requirements. Faults caused by manufacturing variation and process or component failures will be tested for here. Any test method has to be efficient [6] to ensure economic viability.

For many systems LOC testing will not end post production, but continue into the deployment, with the need for self-test at power-up, on the insertion of a disposable microfluidic device, or periodically throughout runtime. Critical systems, for example in medicine or forensics, will have to deliver a correct analysis or report a fault. Autonomous monitoring systems, for example in environmental and security applications, will require regular self-test to detect problems such as fouling and degradation.

The need for more extensive work on LOC testing is predicated on increased commercialisation. Becker's [4] recent analysis of microfluidics using the "Gardner Hype Cycle", (Figure 1) placing current

microfluidic development on the *slope of enlightenment* approaching the *plateau of productivity*. It is on this plateau where commercialisation of microfluidics will begin, and development of effective testing will become a serious requirement.

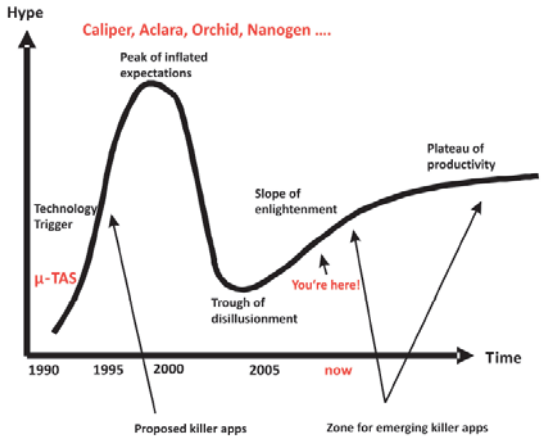


Figure 1 Gardner Hype Cycle [4]

Comparison with other technologies may also provide insight into the development path of LOC technology. The annual "MEMS Industry Report Card" (Roger Grace Associates) identifies most critical success factors governing MicroElectroMechanical Systems (MEMS) and their commercialisation. In a recent publications [7] the mean time for full commercialisation from discovery for a MEMS application is stated as 25 years. This would place full commercialisation of Microfluidic applications at 2015 if we consider Manz vision [8] in 1990 as the starting point. This is in line with Becker's analysis.

Similarly, for the semiconductor industry, the 2007 International Technology Roadmap for Semiconductors (ITRS) [9] states that the integration of emerging and non-digital technologies, specifically microfluidic systems, poses key challenges for testing and provides opportunities for innovative Design For Test (DFT). It highlights the need for integration of radically different test methods into a cost effective manufacturing process.

The type of platform which the microfluidic system is based on could determine its rate of commercialisation and the ease of testability. For example, continuous flow reactors which are typically etched using photolithography into a glass substrate are the most mature reactor design and have received the largest number of published papers. However, prompted by the requirement for high controllability, system diagnosis and reconfigurability, "digital" biochips have been receiving much recent attention, predominately from the test community.

The ability to manufacture a system could determine the overall success of the system, rather than success being based on the science alone. Gaβmann [10] reports on a complete flow injection system

based on PCB technology whereby 4 boards are stacked to create the microflow injection system, Figure 2. This system comprises two thermo-pneumatic pumps used for pumping the analyte and reagent, respectively. PCB technology is highly manufacturable and suited to volume production. Furthermore, this approach does not rely on any external connections, whether they be instrumentation or fluidic.

More recently 0.18μm CMOS technology is receiving much attention [11] for the complete fabrication of Microfluidic systems, currently making use of capacitive sensing, for such applications as DNA analysis. This technology has the advantage of being highly manufacturable and integratable, enabling commercial manufacture and following a mature technology path, with developed test techniques for the electrical domain.

Thus it seems reasonable to assume that there will be an increasing need for test to be considered as part of LOC system design. In line with Becker's call for more industry awareness amongst graduates there is a need for an awareness of test issues, fortunately there is some evidence that this is starting to happen [12].

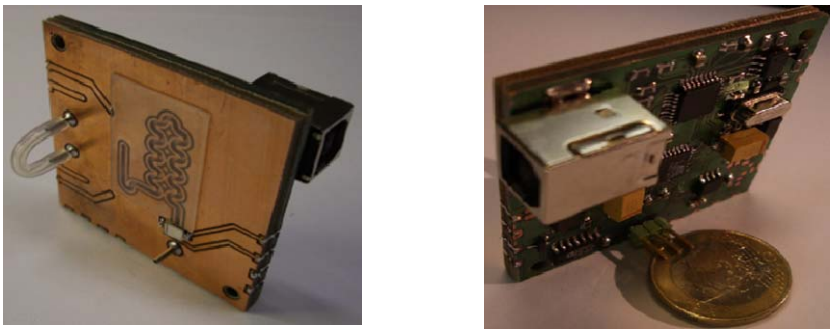


Figure 2 Gaβmann [3] fluidic PCB's

A survey among two popular literature databases (Web of Science and Scopus) showed that of all the literature published on microchemical systems[1] 10% mentioned test[2] (although this word could often be used in other contexts) fewer than 1 % referred or acknowledge faults[3], and fewer still <0.05% had considered faults as part of their system simulation[4]. Despite this there is a body of work concerning microfluidic and LOC system test.

The need for a complete system design workflow which allows for the integration and co-simulation of each constituent domains (chemistry, biology, fluidics, sensing/control electronics and software) to optimise performance, reduce cost (manufacturing and R&D) and improve reliability; test development will be part of this process. Heterogeneous simulation is not a trivial task, no one package

[1] microchemical, microfluidic, microfluidi*, microchem*
[2] test
[3] fault, failure
[4] fault + simulation

allows a system to be designed from geometry through to the fault classification algorithm. However "Multiphysics" modelling packages, such as COMSOL [13] go some of the way.

1.1 Aims & Objectives

- Determine an accurate method of microfluidic system simulation, through cross-validation.
- Investigate and generate a fault library of microfluidic fault conditions.
- Determine test methods capable of use on microfluidic systems and for providing the detection of the faults in the fault library.
- Produce microfluidic fault models capable of co-simulation with the fault-free system.
- Develop a Microfluidic Fault Simulator, MFS.
- Develop a test analysis algorithm to produce test metrics to determine the effectiveness of applied test methods.
- Develop practical classification rules which could be applied to system measurements to determine the operational condition of the system.

This thesis is organised as follows: Chapter 2 reviews the state-of-the-art for microfluidic fault simulation and test with a general review of fault classification techniques. Chapter 3 explores a FEM (Finite Element Method) using COMSOL "Multi-Physics" to simulate a range of Microfluidic system types and cross-validates the simulation data to experimental and published results to determine the methods accuracy. Chapter 4 investigates microfluidic faults and develops fault models to be used in system simulations; these models are cross-validated to experimental work. In this chapter test methods applicable to microfluidic systems are presented and investigated both experimentally and via simulation. Chapter 5 combines the work from Chapter 3 and Chapter 4 to develop a Microfluidic Fault Simulator, MFS to allow the automated exploration of nominal fault-free system measurements and faulty system measurements through the injection of fault conditions. The measured data from the MFS is analysed in Chapter 6 where a test analysis method is presented based on the traditional test hypotheses, showing that our approach from Chapters 3 to Chapter 5 may be integrated into current test analysis techniques without any abstraction of the system. Chapter 7 introduces dedicated instrumentation to aid system control and test, and presents decision tress as an automated method of generating test classification rules. Finally, Chapter 8 studies a new application, a flow cytometry application and the design, simulation and test methodology is applied. This work is concluded in Chapter 9 and further work discussed.

Chapter 2 Literature Review

2.1 Introduction

Lab-on-a-chip brings various testing issues, for example, testing without contamination (e.g. in one-use chips with reader systems), which like the vast complexity of digital systems may prevent simple functional tests from being used. For example, in a different technology area, cell phones have to be tested very quickly to be economically competitive [6]. To test a phone you could make a call, send an SMS, take a photograph, and so on, but this would take hundreds of times longer than is viable. Of course phones are not tested this way, and similarly, but for different reasons, innovative but effective testing methodologies and DfT techniques will be required for some LOC systems. In some cases, such as forensic and life-critical healthcare applications, effective quality validation may have strong legal and ethical implications.

There are tradeoffs in product quality and cost, influenced by the severity of consequences of failures, which can be addressed by appropriate decisions relating to design, manufacture and test. Which of these areas most effectively addresses a concern is not necessarily easy to determine. (e.g. see [14], which concerns, but is not limited to automotive electronics).

More recently similar approaches have been developed for analogue and mixed-signal electronics [15],[16]. Test of mixed-signal circuits is more difficult than purely digital circuits and recent rapid developments in more heterogeneous systems such as MEMS have provided yet more challenges. These trends are reflected in the IEEE Mixed-Signal Test Workshop (IMSTW), which was inaugurated as a forum focused on test and design for test issues related to systems encompassing digital and analogue electrical signals. In 2008 the name was changed to Mixed-Signals, Sensors and Systems Test Workshop (IMS3TW) to reflect the increasing importance of developments in heterogeneous systems; including Microfluidic and LOC technologies, and how these (increasingly multi-domain systems) will need to be tested

LOC systems may follow similar patterns to those other technologies which have emerged over recent decades. Digital electronic components test strategies have slowly evolved and standard methodologies can now be implemented as part of a normal design flow, supported by Computer Aided Design (CAD) tools [17]. There is a need for sophisticated electronics test techniques because i)

simplistic function test requires excessive time or provides insufficient coverage ii) potential defects are not economically detectable during production unless Design for Test (DFT) is employed iii) self-test is often required, either to make production test economical or provide in-field test. Although digital electronics testing is a mature area it remains challenging due to the continuous rapid advances in this technology.

In order to develop a methodology based on the requirements of design, simulation and test, there are four important topics; microfluidic (fault) modelling, microfluidic faults and microfluidic test and fault classification which need to be addressed.

2.2 Microfluidic Fault Modelling

This section reviews the state-of-the-art for microfluidic simulation. A recent review of microfluidic researchers implementing simulation in their work by the Royal Society of Chemistry [18] showed that of all papers published on microfluidic devices only 25% contained simulation. They stipulate one of the most important features of numerical analysis is the richness of data, in that significantly more parameters can be analysed in greater detail than empirical work. Implementation of these techniques will be more effective with multidisciplinary collaboration, as the concept is less alien to engineering researchers. Comparisons to the electronics community are made where system simulation is the initial step in the design flow, long before prototyping; allowing early optimization and removal of design flaws, in a low cost environment, rather than performing multiple high cost prototype iterations.

With computing power increasing and costs dropping, many heterogeneous modeling software packages, utilizing the Finite Element Method (FEM) approach are coming to the fore, such as, COMSOL [13] and Coventor [19]. Erickson [20] provides a review of FEM modeling approaches. These software packages simplify the mathematical model set-up by implementing a graphical user interface. However, there is general sceptism in the results they produce without experimental validation, hence the requirement for cross-validation.

The first work on microfluidic fault modeling was through the use equivalent circuit models, [21],[22] where Kerkhoff introduced a mature micro-flow sensor (heated beam principle), and applied approaches developed for fault modelling and simulation of analogue electronic circuits to predict the faulty behavior. Component substitution was applied to model faults occurring from processing defects. Whilst suffering from a lack of reliable defect statistics, possible fabrication defects in the fluidic flow sensor were identified (from experience). This enabled techniques developed for electronic circuit

assessment to be adapted to microfluidic systems. Use of abstract system models such as this, requires that the failure mechanisms are abstracted into the same domain to create appropriate fault models.

Berli [23] used electrical equivalent circuit models to model the fluidic transport mechanisms in an electrokinetic immunoassay chip. The geometry explored is reasonably modest, but the conversion approach is from first principles and is very complex. The channels are represented by a set of conductance coefficients and predict the fluid flows and corresponding currents in all the channels.

Generally, the approach of equivalent circuit modelling does not appear efficient as intermediatary packages have either to be used to aid the abstraction process or the system has to be studied from first principles, which is significantly prohibitive for microfluidic researchers.

Other less formal modelling methods, combining electrical circuits and algorithms have been proposed. Chatterjee [24] created a combined circuit and device model of an integrated microfluidic system. They identified 4 key areas of the microfluidic device; fluidic transport, mixing, reaction and separation. A second order reversible process, was modelled by ODE's, the mixing and separation processes were based on the charge of the ions or their electrophoretic mobility. These models were formed into an algorithm which described the system as a whole.

The natural progression from equivalent circuit modelling was the use of the modelling language VHDL-AMS (IEEE 1076.1). The current standard was introduced in 1999 and is aiding the creation of behavioural models and integration of individual domains in a single mixed-domain modelling environment. VHDL-AMS has the ability to describe mixed-signal domains at a much higher level while automatically maintaining the conservation of energy. Early adoption of this language was for investigating MEMS systems. Wilson [25] demonstrates the use of VHDL-AMS on an electro-magnetic and a MEMS system as a means of characterising their multiple domain behaviour. VHDL-AMS made its introduction to microfluidics in 1998 with Voigt [26] modelling a micropump at a low level of abstraction and incorporating this into an higher level abstraction system model.

Using the VHDL-AMS approach Kerkhoff [27] implements a multi-domain simulation of a DNA bio-sensor array. This allows fault simulation of the various parts of the system, and describes the behavior of the sensing technique, the fluidic transport mechanism and the coupling to the electronic domain. Kerkhoff [28] considered the simulation and test of microfluidic systems using electro-osmotic flow, controlled by FlowFETs, particularly for arrays of sensors. Faults modelled include shorts in the driver circuit and incorrect driver voltages, which produce incorrect flow rates. It is stated that in conventional fault simulations the fault models are usually static, however in systems containing biological fluids new time dependant defects can occur, such as partial or complete jamming of channels.

The typical causes of these problems are from proteins or peptides chains growing longer in the channels, cells may increase in volume due to osmosis, bacteria grow and blood may clot. The system has successfully been simulated with a range of new biological faults and conventional microfluidic faults. The simulation allows the design for test strategies to be implemented to detect, and where possible circumvent, the faulty part of the system. No details regarding the test strategy were reported.

Kerkhoff *et al.* [29] used a co-simulation technique for a flowFET system initially utilizing a FEM system model to model the system at a low level of abstraction then mapped it to the VHDL-AMS environment, providing a platform for fault injection. More recently, their approach was shown to work [30] for a two-dimensional biological MEF (MicroElectronic Fluidic) array. In this work protein and peptide chain growth are addeded to the previously described biological time dependant faults. A method of adding these faults to their previous approach using a hydrodynamic resistance parameter is described.

A similar abstraction technique has been used in MEMS modelling by Schlegal *et al.* [31]. They demonstrated a method for transposing a low level FEM model using an eigenmode reduced order modelling technique, followed by conversion to VHDL-AMS to form a "black box" component. This allows a low order FEM model to be efficiently used with higher level abstraction components, resulting in a heterogeneous system, with the accuracy of a FEM structure in a more computationally efficient form.

A variation on the FEM theme is the use of a finite volume technique. Chatterjee [32] investigates a unified model of electrophoretic transport phenomena and chemical behavior in weak analyte systems. A multi-block finite volume technique is used to describe the transport phenomena in a 3D micromixer. The results gained through this mathematically intensive method have become a standard and simplified feature of modern FEM modeling packages.

A more widely recognised method of high order behavioural modelling is through the use of SystemC, this has been cited by the International Technology Roadmap for Semiconductors (ITRS) as being one of the prominent heterogeneous system modeling languages [9]. It is based on C++, which allows the modeling interaction of multiple interacting physical domains along with software co-design. Zhang *et al.* [33] introduces SystemC as a single solution for hierarchical system modeling, suitable to MEFS (MicroElectroFluidic Systems). To strengthen their argument for systemC and droplet based biochips [34], they compare two approaches implementing a composite Polymersease Chain Rection (PCR) system; one approach is using conventional continuous flow and the other a droplet approach. The performance of both is evaluated for the ease of implementation in the SystemC design environment, the ease of design and integration complexity. A combination of low level component models utilising

ODE's and high level system behavioural models are used. They conclude that the droplet based approach is more efficient to model and has a lower design and integration complexity.

Wang proved that a system-oriented system model [35],[36] consisting of behavioural system components designed in Verilog-A of an electrokinetic immunoassay chip using a top down schematic approach could be composed and be used to analyse the system using DC and transient analysis techniques. The model was validated by numerical and experimental analysis and produced an error of less than 5%. More recent work [37] by Wang continues previous work on decomposing large complex systems into discrete components, however the interactions between adjacent components is facilitated using Fourier series coefficients in particular for analyte concentration profiles.

Work presented by Roman *et al*. [38] focuses on the development of a complete CAD tool for the development of a fault simulation environment for MEMS systems. While not directly compatible with microfluidics, the background and motivation has a similar foundation. For example, their work started with the identification of failure mechanisms and fault classes, which lead to the development of fault models, and embedded test. However, the lack of suitable CAD tools for microsystem fault simulation has resulted in a lack of quantitive evaluation of the suitability of self-testing approaches. Their approach has been to develope a Fault Model Description Language (FMDL) based upon CADENCE (CADENCE Design Systems Inc.) where a design may be broken down into well described modules, and then simulated and the test strategy assessed.

Reppa *et al.* [39],[40] implemented a circuit level model of a MEMS micro-spring in HSPICE coupled to MATLAB. They run multiple iterations (100+) of worst case simulations. A sliding orthotopic set membership identification scheme is used to identify the systems parameter vector. A recursive algorithm predicts the systems output interval based on the parameter bounds. A fault is detected if the actual systems outputs are not within the worst case interval output. It is shown that this scheme can be used to identify mechanical faults.

2.2.1 Modelling Summary

It has been shown that microfluidic fault modeling is slowly being investigated, predominately by those from an electrical background. It is important that microfluidic system modeling for the purposes of proof-of-concept and optimization are suitable for use without rework or abstraction, for the investigation of reliability and test strategy development.

One approach to analysing potential failure mechanisms and developing tests ahead of empirical testing is *Simulation before Test* (SbT) [41],[42] a technique which is used for electronic systems. This approach relies upon an accurate model of the system under test and appropriate fault models.

Widespread use of such approaches with LOC systems will require an increased use of simulation tools in LOC design.

2.3 Microfluidic Faults

The progress of *test* in engineering domains has been to recognize the faults and failure mechanisms associated with the processes and/or components which comprise that system. Furthermore, the ability to simulate that system, both in a "healthy" fault-free state and a faulted state allows data to be collected to facilitate test analysis. This process of fault simulation is now mature in the area of digital system design.

The general problem of simulation and fault predication becomes increasingly complex when more domains are added to the system. Therefore, heterogeneous systems such as mixed-signal and MEMS require much research effort to study their potential faults and subsequent fault simulation, to improve system test and subsequent reliability, which will come with streamlining the design and test workflow. The greatest differentiator between the classic digital domain and heterogeneous system is the continuous nature of the fault condition coupled with parametric variation. This is true of microfluidic systems which are highly heterogeneous.

Furthermore, microfluidics is largely a science based community; therefore the reporting of system failures and faults has negative connotations. This coupled with the low-volume production makes for a lack of manufacturing fault or failure statistics, which in turn leads to low-in field usage and resulting in-field failure types and Mean-Time before Failure (MTBF) statistics remaining unknown.

Faults may be classified as either parametric or catastrophic (sometimes also known as *soft* and *hard*, faults respectively). A parametric fault is one which moves a system parameter from its nominal value (e.g. an applied voltage shifted somewhat from the required value or a channel under or over expected width after etching). Parametric faults may result in faulty operation, depending upon the sensitivity of the system to this parameter and the values of other system and environmental parameters. A catastrophic fault would be a gross manufacturing defect e.g. foreign material deposited in a channel during manufacture, causing a full or partial blockage. Despite the name catastrophic faults may not always result in total system failure; but they do change some aspect of the systems structure or topology. This may have a strong impact on how such a fault is modelled.

There are no detailed reviews of microfluidic and LOC failure mechanisms, however Walraven [43] compiled a brief review on failure analysis in MEMS devices, including microfluidics as one of the

six categories reviewed. Walraven's main observation is that most MEM's test techniques have been leveraged from IC testing and therefore will be ineffective for fluidic diagnostics.

Presented in this section is a review of literature pertaining to microfluidic faults, this list is added to in Chapter 4 of the thesis, containing more faults from experience. This short review of microfluidic fault literature depicts the limited extent of research in this area. While it is anticipated that researchers face an array of faults daily most remain formally unreported.

2.3.1 Interconnect failures (world-to-chip)

Interconnects are one of the most important aspects of the microfluidic system. Without connection from the world-to-chip, no analysis would be possible. Gray [44] investigates mechanical and fluidic interconnect test apparatus and protocols for characterising both chip-to-chip and chip-to-world interconnects. Mechanical strength test apparatus were developed to test a range of interconnects pull-out forces, alongside fluidic testing where interconnects were tested for leakage under high pressure. Korivi [45] presents a generic chip-to-world interconnect system. A concluding remark is that with the standardisation of a method comes reliability and simplification of maintenance.

2.3.2 Channel Blockages - Partial and Complete

Kano *et. al.* [46] studies stacked microchemical reactors for use on an industrial scale and states that blockages are a critical problem to the successful and reliable operation of microfluidic systems. A mechanism which causes blockages in such a system is radical polymerization, a well-known technique in synthesis, when *bad* mixing conditions cause fouling. In the work presented, temperature sensors and the differences in temperature between them were offered as a method of detection. Two diagnosis techniques were trialed one a data-based system and the other a model-based system. Different degrees of blockage and location were investigated. Kano demonstrated that both techniques could detect a blockage location even when the blockage degree was less than 10%.

Particulate blockages occurring in cathedral chamber style biosensors have have been investigated by Chapman *et. al.* [46]. A capacitive detection method is used to measure the property of the biomaterial within the chamber, in an operational role, with sensitivity of this technique also shown towards fault measurement. The faults investigated were trapped bubbles and foreign particles within the cathedral chamber. The cathedral design allows some fluidic routing reconfigurability if a fault is detected. The impact of a fault on fluid flow was investigated in COMSOL by restructuring the geometry to create varying degree of blockage conditions. Based on this principle and through the implementation of a Monte Carlo analysis it was found that the cathedral design showed that 6 times more *particles* were required to cause a complete blockage over a conventional parallel array design.

Amador *et. al.* [47] have studied flow distributions in microfluidic reactor and manifold structures. They use an analytical model approach based on an equivalent electrical resistance network to facilate this. Further to using this modelling method to investigate how manufacturing tolerances can affect flow, they also study blockages. Blockages are introduced into their electrical resistance model as an infinite resistance.

Precipitate blockages are reported in the patent by Caliper Life Sciences [48]. Typically stored analytes on a microfluidic device are stored in concentration and diluted by the amount required when needed. This dilution process can cause precipitation, releasing salts and proteins which cause channels to become blocked.

2.3.3 Electrode Degradation

Urbanski [49] assess the application of AC voltages in electro-osmotic flow, as sustained DC voltages on electrodes cause Faradaic reactions which degrade the electrodes and cause concentration gradients. Further work by Liu *et al* [50] is presented in the microfluidic test section under through the use of impedance spectroscopy (Section 2.4.4).

2.3.4 Bubble Formation

Leung *et al.* [51] state that many organic reactions generate products in the gas phase, which produce bubbles. They demonstrate an optical method using the bubbles refractive index to provide a method of continuous real-time monitoring technique.

A patent by Caliper Life Sciences [52] recognises that trapped bubbles may cause problems in microfluidic channels and therefore they have developed a method for voltage and current monitoring using an array of electrodes. They state that one of the sources of bubbles may be from joints in channels. There have been no publications pertaining to the detailed cause of bubbles. Intuition prevails and names sources as electrolysis and leakage at air/fluid inlets.

2.3.5 Biological

Zhang *et. al.* [53], [54] uses a 9 chain amino acid peptide AMLDLLKSV in a system for the detection of cancer cells using an MEF system. Leucine is an amino acid which is known to have binding problems, resulting in the peptide AMDLLKSV, since the peptide is used as part of the detection mechanism then Leucine's failure to bind results in a lower quality of detection. The "purity" of the amino acid is proportional to its conductivity, therefore current tests of control electrodes or impedance spectroscopy tests using direct sensing electrodes can determine the "purity". Their system is an array of multiple sites, each allowing synthesis of the peptide. If a peptide quality is deemed insufficient then that

particular site is added to a database and is no longer used, greatly improving the purity of the peptides produced.

2.3.6 Design Considerations

In addition to the consideration of outright fault conditions, manufacturing and design tolerances, poor design or system operation outside normal limits can cause a system to behave out of specification.

Manufacturing defects and tolerances may be attributed directly to causing system faults and such have been reported for MEMS systems [55]. However, such faults for microfluidic reactor fabrication have so far been unreported, due to the low commercially manufactured quantities. Furthermore, in chemical separation systems, such as those using chromatography bead packing densities and irregularities can cause out of specification operation [56].

Instead, system operation *faults* are considered due to design oversights, for example, Sun *et. al.* [57] introduce a electro-osmotic switching design, whereby, due to reservoirs dimensions, after a certain system operation time an hydrodynamic back pressure is produced caused by the fluid height difference between the inlet and outlet reservoirs due to Electro-osomotic flow (EOF) flow, this back pressure overcomes the EOF flow, causing the system to fail (cease flowing in the desired direction).

Joule heating [58], pH Gradients [59] and Zeta potential variation or reversal [60] are all examples of circumstantial, system operation faults or out of specification behaviour. Therefore, it is desirable in a system that some reconfigurability is designed in, making the design defect tolerant [61] and capable of degrading gracefully; such possibilities are provided using the digital droplet based biochip system architecture. A microvalue is studied for robust design using the Taguchi methodology [62] and response surface used to explore the widest performance range. A hardware and software co-design principle is used to split the system design into non-reconfigurable and reconfigurable partitions.

2.3.6.1 Microfluidic Fault Summary

There is little published work on the range of potential faults which could be encountered in microfluidic systems, from experience and conversation the amount of published work is not proportional to the problems encountered daily. Some of the most frequently discussed work relates to blockages, which are common in most continuous flow microfluidic systems, regardless of transport mechanism. The potential range of failure mechanisms is probably infinite since any combination of chemical or biological process could produce a unique mechanism, however the resulting fault may be already defined i.e. blockage, electrode degradation or bubble.

Furthermore, the investigation into fault types has uncovered a range of patents from private companies (in particulary Caliper Life Sciences, Inc) patening fault detection technology, therefore these companies recognise that for realistic microfluidic commercialisations, fault conditions have to be accounted for, further highlighting the growing gap between microfluidic research (number of published papers) and microfluidic industry (number of published patents).

2.4 Current Testing of Microfluidic Systems

As with microfluidic fault modelling, consideration of the testing of microfluidic systems is beginning to occur. Microfluidic systems are the latest (and most complex) in a long line of Mixed-Signal Systems; the most recent to be investigated for commercialisation, being MEMS. Microfluidic systems are more complex to test than MEMS due to increased domain count. To date there has been relatively little activity in the area of testing continuous flow microfluidic and microchemical systems, particularly in comparison with mechanical Microsystems,[63],[64],[65],[66] and the Digital Biochip, which differs from continuous flow systems as it using discrete droplets which move around the surface of the device. This has gained interest from the test community, as these devices are re-configurable, tightly controlled and can be studied using well-known path and optimization algorithms, which is desirable. However, our primary interest is in continuous flow systems, as these are the most researched and therefore, the most likely to reach commercialization[67][1].

2.4.1 Optical Detection

Kerkhoff [68] has reported using optical detection methods to validate and test FlowFET's. This technique requires reactor transparency and employs fluorescent tracer beads, a laser and CCD camera to monitor the flow and path of the beads. A 3D perspective may also be achieved. This approach may be used to determine fluid velocity and detect obstructions in the channels.

2.4.2 Resistance Testing

De Venuto [69], [70] describes a novel optical DNA biosensor which uses fluorescence to measure DNA extraction. The biosensor is fabricated from silicon including the on-board photodiode for fluorescence detection. Faults relating to the fabrication of the photodiode are investigated using an Inductive Fault Analysis (IFA) technique. A simulation of the 5 possible faults (consisting of resistive open and bridge faults) is carried out, which is said to cover all possible faults. From this a critical resistance curve is determined and used by the test strategy when in deployment, by comparison to two

[1] An explicit survey among the most researched / published papers has not been possible to prove that continuous flow systems are the most likely to reach mass commercialisation.

local reference points. For the open and short fault conditions an analysis (impact analysis) is performed inductively to determine the erroroneous consequence on the system given this faulty sensor data.

2.4.3 Oscillatory

Liu [71] propose testing a DNA sensor array using an oscillation methodology. They use a generalised impedance model for the electrode, double layer and electrolyte but do not give details on the implementation of the oscillation method.

Oscillation methods have been used as a DNA detection method in a Bio-Sensor [27] using a simple RC network where the capacitance of the electrodes acts as the C in the time constant. A frequency increase from 45 kHz through to 85 kHz occurs during the fault free detection of DNA. It is expected that a fault would cause a significant decrease in capacitance resulting in a higher oscillation frequency. The system has been simulated using the previously reported method [28] of mixing FEM simulations with VHDL-AMS.

2.4.4 Impedance Spectroscopy

Impedance spectroscopy is a mature and widely used technique in applications ranging from corrosion analysis [72] to fuel cell evaluation [73]. Dumas [74] implemented an impedance test methodology for use on a Micro-Electrode array for a bio-sensor. The array consists of 60 elements and the impedance is measured for each one and compared to an arbitrary electrode within the array, the assumption is made that because each electrode should be identical, any variation in the impedance of one electrode to compared to another highlights a potentially faulty electrode. The electrodes are manufactured with a highly porous surface to increase the effective surface area and to decrease the interfacial impedance. During use the electrodes surface tends to degrade, thus increasing impedance and causing variations in measured results. Liu *et al* [50] investigated electrode degradation in a Micro Electrode Array (MEA) containing 60 electrodes and designed to investigate bead bio-assays. The work carried out demonstrated that electrode degradation could be modelled by an equivalent lumped RC circuit model, and by varying values of Z_p mimic degradation of the porus layer. They compared the fault simulation and experimental results and found a high degree of validation.

Electro-Impedance Spectroscopy was used by Ayliffe [75],[76] demonstrating its use as a functional flow sensor for ionic solutions and Zou [77] using interdigitated electrode arrays for dielectrophoretic manipulation and detection of microparticles. This is a good demonstration of how a functional technique and hardware can be utilised as detection and potential test hardware, with little increase in cost.

The detection of bubbles has been investigated using optical methods, in particular changes in refractive index [51], however, it is thought that bubble detection could also be achieved using an impedance spectroscopy method, as a bubble passing between two electrodes would perturb the impedance.

2.4.5 Capacitance

Ghafar-Zadeh [78],[79] investigates on-chip capacitive sensing using the CMOS based technology Direct-Write Microfluidic Fabrication Process (DWFP), which allows for highly integrated electrodes. Their first work experiments with these electrodes to detect and classify four different chemical solutions with known dielectric constants, with a sensitivity of 530mV/fF. Their later work uses a similar technique for monitoring the growth of bacteria, in particular *Escherichia coli* (Ecoli). It has been found that bacteria in the proximity of an electrode would cause a detectable change in double layer capacitance ($\Delta C < 60pF$). The importance of this technique is that it offers functional sensing and could be used as a test method to identify electrode degradation, bubbles, or simply reported an unknown capacitive state.

2.4.6 Digital Biochips

Su [80], [81] introduces digital biochip as an alternative to continuous flow biochips, offering the advantages of dynamic re-configurability and architecture scalability. Presented are classifications of catastrophic and parametric faults in droplet based microfluidic systems and indications of how faults could be detected by electrostatically controlling and tracking droplet motion. Tolerance analysis was also presented, based upon Monte Carlo simulations, in order to characterise the impact of parameter variations on system performances. Minimum detection deviations for each parameter considered along with linear search algorithms, were used in conjunction with Monte Carlo simulations. Re-configurability was demonstrated [82] through simulation of a chip containing faulty fluidic cells in which the system reconfigured itself in operation so as to no longer use this area of the chip. Test strategies are analysed further [83] and test droplets are introduced as monitorable test signals. Their movement around the system was analysed and, depending on the size of the chip used, one of two algorithms were used. If the chip is considered small, optimal solutions could be obtained through the use of an Inductive Logic Programming model (ILP). For larger systems heuristic algorithms were used. A parallel-scan test is implemented successfully for both off-line and on-line testing [84]. Su considers concurrent test [85]. Xu Considers DFT [86],[87]. Zhao considers BIST [88],[89] . Xu considers fault models and functional test [90], Mitra also considers functional test [91]. Datta Considers diagnosis [92],[93]. Zhang considers fault tolerance [94].

2.5 Fault Detection and Diagnostics

In Chapter 6 we review methods of test analysis, that is the probabilistic determination of whether a system (or circuit) should pass or fail a test based on a series of measured data (test sensors), and considers their associated testing errors; type I and type II. The reason for the postponement of the review is because at this juncture our interests are in the modeling, test and classification of a system, using the assumption that the measured data does not contain probabilistic errors.

Most system test in the electronics and mixed-signal communities focus on passing or failing a system post production and pre-deployment. Therefore, if a system is deemed faulty it is scrapped or sent for rework. However, the interest in test for many other systems, especially those found in safety critical roles, concerns continuous post-deployment (online) testing, with a preemptive action based upon the test outcome. For this reason fault classification or detection schemes need to be implemented. The basis of these schemes is very mature, dating back to the 1960's. A three part review on process fault detection and diagnosis is given by Venkatasubramanian *et al.* [95-97].

For decades, many systems across a broad range of industries have had fault or failure detection and correction strategies integrated into their design. In 1962 Bell Telephone Laboratories developed a safety analysis technique, known as Fault Tree Analysis (FTA), for use on the Minuteman Missile launch control system [98]. The fault tree is a predetermined route of propagation of how a fault at a low level within the system can propagate to cause catastrophic failure. Fault tree analysis became a standard fault monitoring and predication technique in the Nuclear Industry, the Civil Aviation Industry, and in many other safety critical systems. Fault Trees theory is detailed in the Fault Tree handbook [98].

Typically fault trees were created by experts using a deductive procedure assessing combinations of how hardware, software and human error could lead to an undesired event. Abu-Hakima [99] presents the case of the J85-CAN-15 Jet engine where the whole system is not fully understood (having been added to over the years), so interviews with experts to create a fault tree is not a viable option. Instead Abu-Hakima introduces the Hakeem algorithm which maybe used in the generation of behavioural models from fault based knowledge. In this approach the technique of Fault Based Reasoning (FBR) is introduced. Learning the device component model, its behaviour and functionality using the FBR knowledge provides the technician with a tool that can achieve model-based diagnosis. The author compares this approach to a purely model-based reasoning approach (MBR) undertaken by de Kleer [100] stating that MBR can lead to a combinatorial explosion in producing diagnostic information for a complex system.

Madden [101] introduced a fault diagnosis engine based on the authors Induction Fault Tree (IFT) algorithm. The algorithm learns from an examples database containing sensor data and expert knowledge. The algorithm uses induction to learn by analysing sensor data, which may be real or simulated data, that has been classified as normal operation or relating to one or more fault states. Madden [102] argues that while other techniques could have been used, such as, neural network approaches, the inductive method used produces clear fault rules which may be understood by the engineers, hence an approach which is more likely to be accepted.

Papadopoulos [103] takes these ideas further and develops a methodology called "HIP-HOPS" in order to develop and implement safety critical monitoring schemes into safety critical systems. During safety analysis the primary objective is to anticipate potential system failure scenarios. This is achieved by identifying low level faults and observing how they propagate through the system to a high level fault and give rise to system malfunctions. Papadopoulos argues whether the analyst has captured the real behaviour of the system, does the system really behave as predicted by the analyst in the course of the fault assessment? Do failures propagate as expected and do the corrective measures have the predicted effect? Papadopoulos states that his safety monitor can help address these issues by having a behavioural model of the system, in which low level faults can be injected and the effects of the system observed. This allows predictions encoded in the safety case to be analysed, and optimistic predictions found. Additional effects may be observed that were not predicted when originally assessing the system. These either allow a revision of the safety case, or a revision of the system to perform as expected. The safety monitor is broken down into three mechanisms; primary detection of abnormal conditions, the diagnosis of complex failures and the correction or control of such failures.

The requirement to remove the human element from data processing has existed since the 1950's through machine learning. Machine learning has become a widely implementable and deployable technique with the ever decreasing scale and economic cost of processing power, making it deployable into a wide range of applications. In 1986 Quinlan [104] summarised work carried out in the field of decision trees and describes the Iterative Dichotomiser 3 (ID3) algorithm in detail. Since then decision trees have been used extensively in market research, online databases and search engines, but it is their statistical power to classify faults which makes them of interest here. Over the years decision trees have found their way into many industrial and scientific applications. Lee [105] provided a concept on migrating fault trees to decision trees using historical test data to create the tree, which is then modifiable with real-time data. They conclude to say that the decision tree plays a role in knowledge discovery and therefore is continually updated with the latest faults and fault patterns.

Assaf [106] reviewed methods of how to select the next best test, which will provide the most "information" given a set of suspected components. Pattipati [107] introduced component failure probability into the next test selection process. Assaf implemented diagnostic decision trees to a dependable computer-based system, which continues their previous work on ADORA [108], a specialised monitoring methodology for systems which employ redundancy; where more than one component has to fail for a complete system failure. The aspect of their work which may prove useful in test strategies for microfluidics devices is how to implement system monitors and sensors in strategic and information rich sites within the system, while keeping diagnostic costs and test cost to a minimum. ADORA allows tests to be ordered using the Cost and Diagnostic Importance Factor (CDIF). The CDIF is the DIF (Diagnostic Importance Factor) measured from the Vesely-Fussel formula [98] divided by the cost of testing. Reay [109] describes an analysis strategy for converting a fault tree into a Binary Decicion Diagram (BDD) which provide a more efficient means for reliability analysis since they take the form of a Boolean equation.

Bobbio [110] compares Fault Tree analysis with Bayesian networks and concludes that Bayesian networks are more suitable for systems with uncertainty, time variant faults or complex dependencies; such as those systems which consist of hardware and software or in our case a chemical domain. Limbourg [111] introduces the Dempster-Shafer Theory of Evidence as an alternative to the afore mentioned probabilistic modelling techniques. DST is particularly useful when a high uncertainty and a conservative treatment of this uncertainty are necessary, for a safe design of the system.

Fault Tree Analysis is starting to be studied in MEMS and microfluidic systems. Batzias [112] studies the causes of noise in a biosensor and deduces that an improved SNR can be achieved through the implementation of a fuzzy fault tree technique that uses a top down and bottom up approach. This approach allows theoretical and empirical observations to meet to form the optimal solution. The methodology has a fault tree at its core and each node uses fuzzy reasoning to determine the information present in the noisy signal so that the data can be analysed and a decision made.

Myers [113] demonstrated how decision tree analysis could be coupled with a microfluidic fault model for a system comprising a fluidic, chemical and optical domain, to detect and classify system faults. Fault rules could be auto-generated which, when used with fault detection hardware, could be used to detect and diagnose system failures. Decision tree analysis is a popular tool in data mining and provides a useful statistical method for drawing out patterns in fault data.

Failure Modes and Effects Analysis (FMEA) is the technique to study a system to identify, prioritise and alleviate potential problems from the system, currently regulated as a British Standard

[114]. Generally a failure mode is a cause-effect chain that will cause the system to perform its erroroneously. This technique is popular in a wide range of industries from manufacturing to nuclear. Rosing [115] proposes FMEA^{+} with application to MEMS, as part of their simulation methodology. They use FMEA^{+} first proposed by Olbrich [116], as a means to provide quantitative simulation, they only simulate the most likely failure scenarios based upon the outcome of the assessment.

A range of methods from inductive fault analysis [117], pattern recognistion [118] through to probabilistic neural networks [119] for the determination of failure modes and mechanisms in MEMS.

2.6 Conclusions & Objectives

This chapter has introduced the state of the art for the microfluidic simulation and test in an effort to further the microfluidic design workflow and enable microfluidic systems to reach commercialsation. A review of the literature highlights that this area is very much unexplored (our web survey), but highly required ([4, 9]). History tends to repeat itself; and the workflow to commercialisation has been forged by digital, analog and more recently mixed-signal and MEM's. The common blocks of the workflow are design, simulation and test. The design is simulated to allow efficient proof-of-concept studies and optimization, then the simulation used in test analysis, through fault injection and test method implementation.

There is a basic requirement for microfluidic system simulation [18]. If met, this will lead to a greater understanding of microfluidic system design. As seen in section 2.3 many have begun to investigate disparate microfluidic fault simulation methods, with varying degrees of success and scalablility. In section 2.4 microfluidic test methods are reported, many of which have been born out of work on mixed-signal systems. This is no common theme or a single outstanding method, although electrical based test approaches, such as, impedance measurement seem to be gaining some momentum.

With this body of work in existence the work presented in this thesis aims to further these methods, in particular standardize a method of microfluidic fault simulation and test method adoption; and combine them into a single microfluidic design, simulation and test workflow. The now mature area of fault classification has been introduced and will be investigated.

Chapter 3 Microfluidic Modelling

3.1 Introduction

Microfluidic research continues to be very much an empirical science, despite encouragement from leading bodies for the wider adoption of numerical analysis and system simulation [18]. The advantages of simulation being the time reduction of the development cycle, the reduced cost of development, and the ability to provide a wealth of system performance data. One of the main reasons for this slow rate of adoption is the alien environment and distrust in simulated results by empirically trained researchers. The simulation environment and results are common place in most engineering disciplines, but remain somewhat novel to the microfluidic researcher. This chapter explores the use of a "Multi-physics" simulation environment for modeling microfluidic systems, cross-validating between simulation and experimental results.

While the empirical approach satisfies the "proof-of-concept" process, a simulation approach is mandatory for the consideration of microfluidic system fault analysis and test, allowing for multiple iterations, each cycle applying parameter variability, fault injection and test measurements. Microfluidic test is an important issue and a current stumbling block between proof-of-concept systems and commercial microfluidic systems. The subject of microfluidic faults and test will be covered from Chapter 4 onwards. As fault implementation and subsequent test methods are to be explored in the simulation environment, it is imperative that fault-free simulations provide an accurate representation of the real system, the subject explored in this chapter.

3.2 Microfluidic Simulation Environment

There are many simulation environments and methods available, detailed in Chapter 2 and recently reviewed [18]. Throughout this thesis COMSOL "Multi-Physics" (COMSOL AB) is used, which implements the FEM (Finite Element Method). COMSOL provides a "multi-physics" simulation environment, which supports heterogeneous multi-domain simulation, ideal for heterogeneous microfluidic system simulation. In addition to the basic COMSOL package, the MEMS module is used, which provides dedicated microfluidic physics equations, such as, Navier-Stokes and Electroosmotic velocity. Furthermore, COMSOL provides many configuration tools, the solver, global expression support and the scripting interface, which one used in later chapters.

3.3 COMSOL Introduction and Overview

COMSOL Multi-Physics is a FEM modeling package which is optimized for solving heterogeneous models. This section provides a brief overview and introduction to the terminology which is used in this thesis when COMSOL is considered. In depth details and philosophy of COMSOL and the FEM approach are out of the scope of this thesis, only a brief guide is given here.

COMSOL may use 2D or 3D geometries derived from within the COMSOL GUI or imported from a third party CAD file. Consider the 3D geometry in Figure 3, COMSOL considers this geometry the model "domain". This model comprises two individual geometries (one a long 3D cuboid, the second the cube on the side); therefore these are termed "sub-domains".

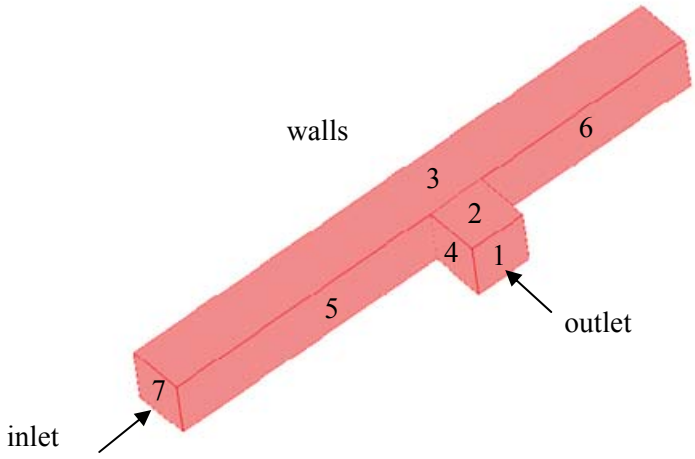


Figure 3 Introductory Geometry

Each face of the geometry is termed a "boundary" and each boundary is uniquely identified by an integer, shown in Figure 3.

COMSOL Multi-physics is specifically designed to simulate models comprising a wide range of physical disciplines. We continue this explanation through the introduction of a hydrodynamic flow problem using the Incompressible Navier-Stokes equation.

In order to solve the model and apply physical equations COMSOL requires the geometry to be "meshed", which means dividing it into segments. Equations are then applied to each segment, propagating the description throughout the entire geometry. An example of the meshed geometry may be found in Figure 4.

Now that the mesh and equations are applied some conditions are required, which govern the way the equations interact with the geometry. In COMSOL there are sub-domain and boundary conditions. In the example, the sub-domain conditions would describe the viscosity and density of the fluid for the two sub-domains. The boundary conditions would describe the inlet, outlet and walls; these may be seen applied to the geometry in Figure 3. The problem may now be solved using an appropriate solver.

More complex models may use multiple equations. Equations are organized into "application modes". Application modes allow different equations to act upon the same geometry (mesh) and then to be coupled to describe their interaction, Figure 4.

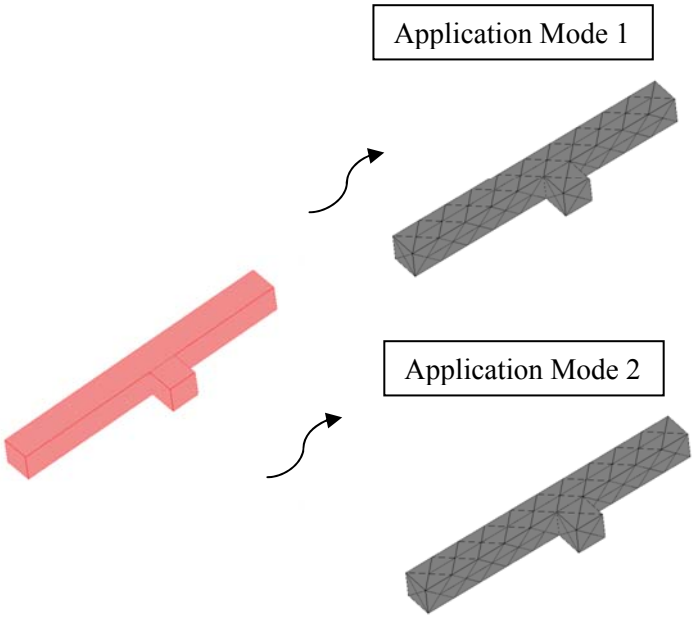


Figure 4 Application Modes

Extending the previous hydrodynamic flow to describe temperature distribution with flow would require the coupling of application mode 1 (Navier-Stokes) with application mode 2(heat transfer with convection). High aspect ratios (lateral and vertical) should be avoided, as some of the equations in the MEM's toolbox are optimized for geometries on the micro scale (mm to μm). For more information see COMSOL documentation MEM's Module User Documentation.

3.4 Introduction to Models

In this chapter the three most common microfluidic building blocks; hydrodynamic flow, diffusional mixing and electro-osmotic flow are presented. Each explores different application modes contained within COMSOL to construct a system model, and each model is cross-validated to experimentation, whether this is from our own experiments or from those found in the literature. Our interest throughout this thesis is continuous flow microfluidic systems.

3.5 Hydrodynamic Model

Hydrodynamic Flow is one of the most common transportation mechanisms in continuous flow microfluidic systems, achieved primarily through the use of positive pressure derived from a syringe or peristaltic pump.

Continuous flow labyrinth style reactors are usually etched using a photolithography technique, although alternates exist such as those based on medical manifolds which are machined from Acrylic and diffusion bonded (Carville Ltd). Here we use a "T" shaped reactor micro-milled from Polycarbonate used in such a way that it comprises two separable halves. The two halves were not diffusion bonded like medical manifolds, but sealed and bolted, allowing the two halves to be separated for subsequent fault studies described in Chapter 4. A further advantage is that the machine CAD file could be directly imported into COMSOL to form the model geometry, tightly aligning the geometries between simulation and experiment. The reactor CAD schematic is shown in Figure 5. The main channel dimensions were 65mm long; distance between inlets 25mm; channel depth and width were 500µm and 700µm, respectively.

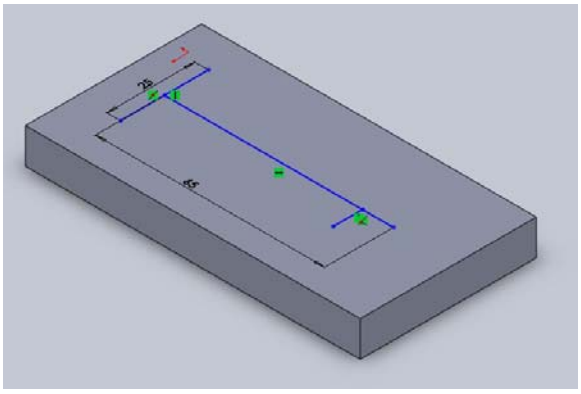


Figure 5 Solidworks Assembly of the "T" Reactor

3.5.1 Experimental Set-Up

Figure 6 describes the configuration of the apparatus for the hydrodynamic flow experimentation, including the "T" reactor for the validation of the hydrodynamic flow model. The experiment uses two Instech Peristaltic pumps (#P625) with tubing set (#P625/TS020S). The fluid (water) in the system enters from either inlet, each inlet containing a different dye to act as a mixing marker. The fluid is pumped in its laminar flow regime therefore no turbulent mixing should occur and any mixing will be due to diffusional-mixing, in this experiment we are only interested in the flow, not the mixing process.

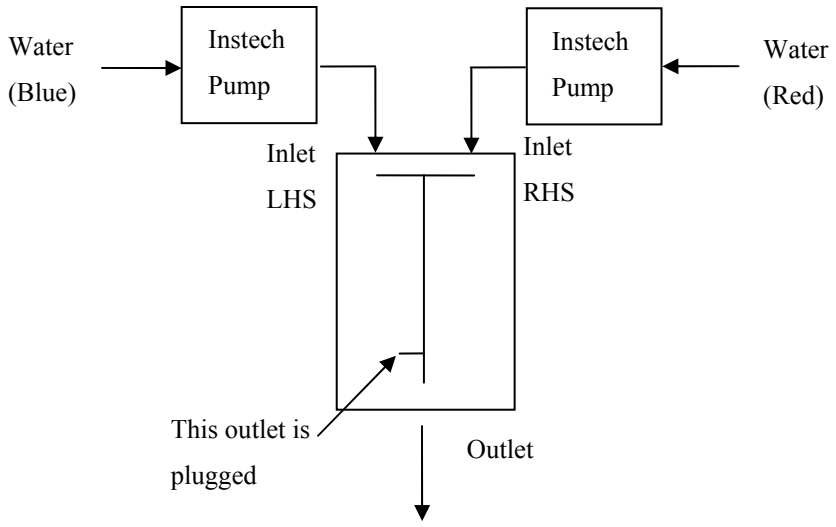


Figure 6 Hydrodynamic Flow Experimental Set-up

The fundamental parameter for validation is the volumetric flow. Since for predictable operation fluid flow needs to be laminar. The Reynolds number must be considered when determining the flow rate.

$$Re = \frac{\rho . v . d}{\eta}$$

Equation 1 Reynolds Equation for circular channels

When considering a non-circular channel

$$Re = \frac{\rho . v . \frac{4.A}{P}}{\eta}$$

Equation 2 Reynolds Equation for non-circular channels

Where ρ is the density 1000 (kg/m^3), v the velocity (m/s), d the diameter of the pipe (channel), A the cross-sectional area, P the perimeter and η the viscosity of the fluid 1x10^{-3} (kg/m.s)

For microfluidics laminar flow is considered when Re < 10. For this experiment Re ≈ 1 will be considered. Re-arranging Equation 2 to find the velocity:

$$v = \frac{\eta}{\rho\left(\frac{4.A}{P}\right)}$$

Equation 3 Finding Velocity when Re = 1

For a channel whose depth and width are 500μm and 700μm, respectively then the flow rate is $1.7x10^{-3}$ m/s. The equivalent volumetric flow rate for the peristaltic pumps is 8.3μL/Min.

3.5.2 Simulation Set-Up

Since this study is restricted to fluid flow, and the flow of fluid assumed to have a constant density, then only the Incompressible Navier-Stokes application mode is required. COMSOL uses the generalised Navier-Stokes equation, Equation 4, which allows for variable viscosity and solves for pressure, p and the velocity vector, x, y and z, when solving for a 3D model.

The first equation is the *momentum transport equation*

$$\rho\frac{\delta \boldsymbol{u}}{\delta t} - \nabla.[\eta(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T)] + \rho(\boldsymbol{u}.\nabla)\boldsymbol{u} + \nabla p = \boldsymbol{F}$$

Equation 4 Momentum Transport Equation

The second associated equation is the *equation of continuity* for incompressible fluids.

$$\nabla.\boldsymbol{u} = 0$$

Equation 5 Equation of Continuity

Where in Equation 4 and Equation 5; p is the pressure, η the dynamic viscosity, ρ the density, $\boldsymbol{u}$ is the velocity field and **F** is a volume force field.

The whole geometry is treated as a single sub-domain described by density and viscosity. Boundary conditions are described as "walls" for the whole geometry, each inlet is described as a "velocity inlet" and the outlet described as an "outlet" whose pressure is equivalent to 0 Pa.

3.5.3 Cross-Validating Simulation and Experimental Results

To obtain reasonably measurable quantities the experiment was run for a 30 minute period where 520μL of fluid was collected at the outlet. Velocity conditions were set in the COMSOL model and boundary integration was used to observe the volumetric flow rate at the outlet which produced $2.99x10^{-10}$

m^3/s which equates to 17.94µL/Min (each inlet approximately contributing 8.3µL/Min) or 538.2µL over a 30 minute period. The difference in the simulated and experimental flow rates were 3.35%.

Figure 7 shows that laminar flow conditions are observed and the steams under experimental conditions and simulation conditions are comparable.

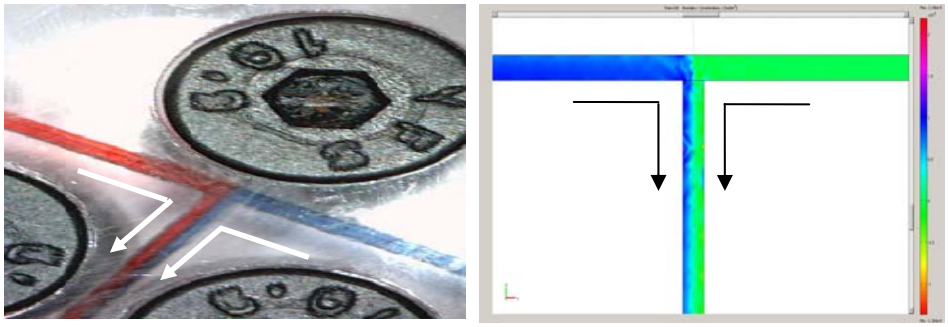


Figure 7 (Left) Experimental Flow Lines; (Right) Simulation Flow Stream Post Plot

Furthermore, Figure 8 shows the experimental and simulation results around the area of the side channel (the channel perpendicular to the main channel near the outlet). It is clear that both experiment and simulation show that the channels have combined; however, of more interest is how the flow has similarly entered the side channel in both cases. Complete protrusion is prohibited as this is a sealed channel therefore there is no through flow.

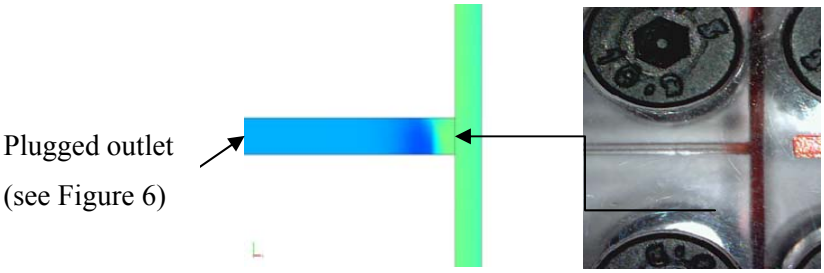


Figure 8 Side Channel Flow Observations (Left) Simulation Post Plot; (Right) Experimental Still

3.5.4 Hydrodynamic Summary

This section has introduced a very fundamental experiment and simulation model, to provide simple cross-validation and to gauge the accuracy of the modeling approach. The results show that the COMSOL model is within 3.35% of the experimental flow results. If this level of confidence continues for more complex scenarios then it demonstrates a suitable approach for fault analysis.

3.6 Diffusion Model

3.6.1 Introduction

The theory of diffusion is mature, and is one which is proving useful in microfluidic systems. The low Reynolds numbers found in microfluidic systems leads to laminar flow; therefore "mixing" of species and reagents by turbulent flow is not present. This has the advantage of being able to precisely control the mixing in microfluidic systems through diffusional mixing. Due to this predictable behaviour diffusional mixing can be used as a tool to characterize flow. Peyman *et al.*[121] use diffusional mixing to characterize flow streams in a reactor used for the assessment of biochemical assays, using magnetic particle technology. In this instance adjacent streams are used to carry reagents when performing immunoassays, therefore it is important that there is no cross-contamination through diffusion between adjacent streams.

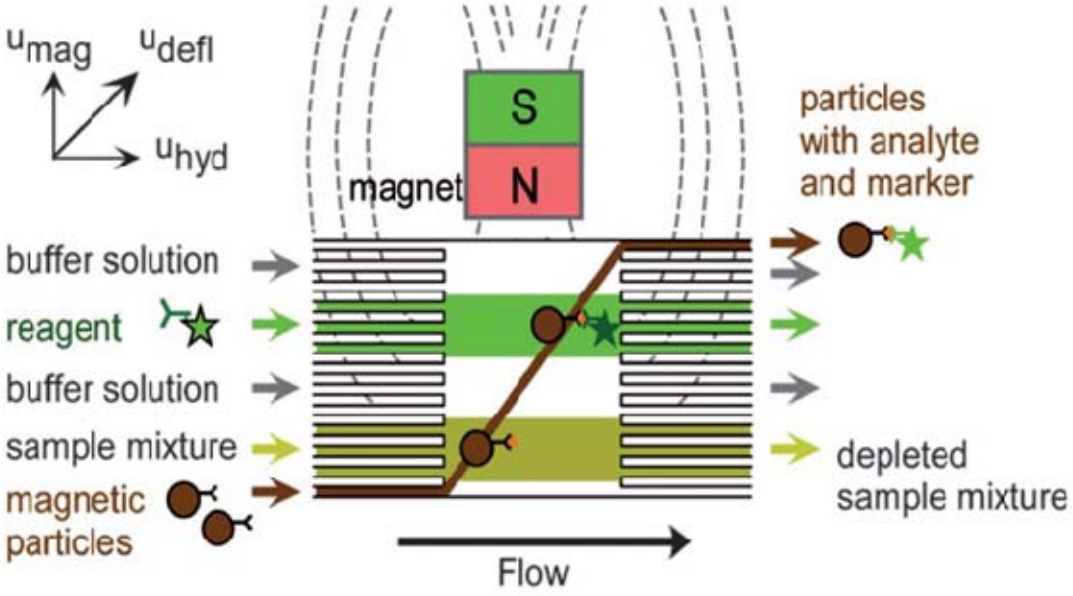


Figure 9 Peyman Application Schematic [121]

This is a good example of where empirical experiments are used to perform an assessment of a system performance, presenting a case where a simulation approach would have been a useful validation tool to optimize the reactor design. Therefore an equivalent simulation is performed and validated against experimental results.

3.6.2 Experimental Set-up

Figure 10 details the main reactor chamber geometry and dimensions as shown in the COMSOL geometry.

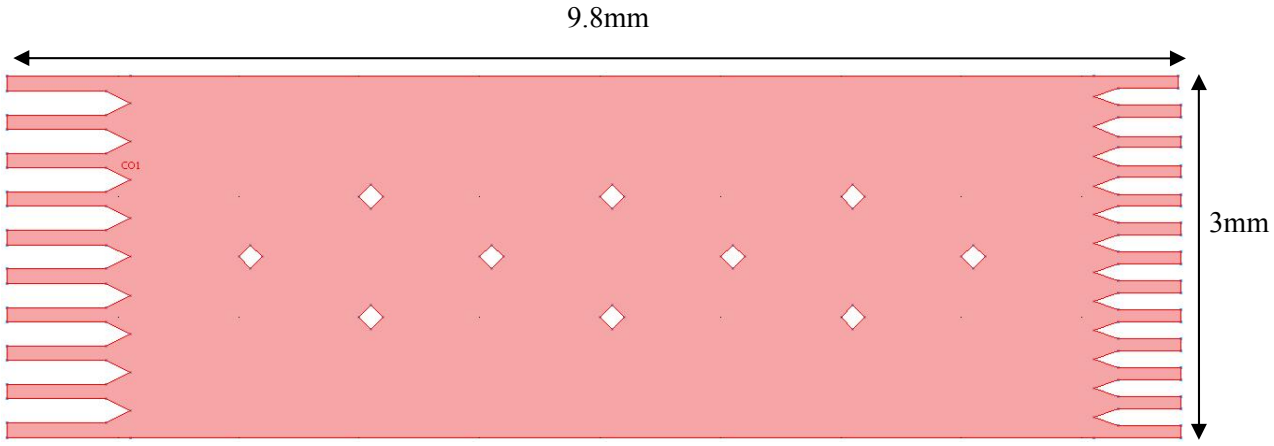


Figure 10 COMSOL Reactor Dimensions

The experimental set-up is shown in Figure 11. In order to visually observe the diffusional mixing between streams, alternating streams of Iron (III) Sulphate and Potassium Thiocyanate were used. When these two colourless reagents diffuse they produce a dark red complex (Iron Thiocyanate), resulting in a visible diffusion; a photographic still is taken for later analysis.

The depth of the reactor is 20µm, with each inlet and outlet channel width measuring 100µm. The diffusion coefficients for Fe^{3+} (Iron Sulphate) and for SCN^- (Thiocyanate) are $0.62x10^{-5}cm^2s^{-1}$ and $1.758x10^{-5}cm^2s^{-1}$, respectively [121].

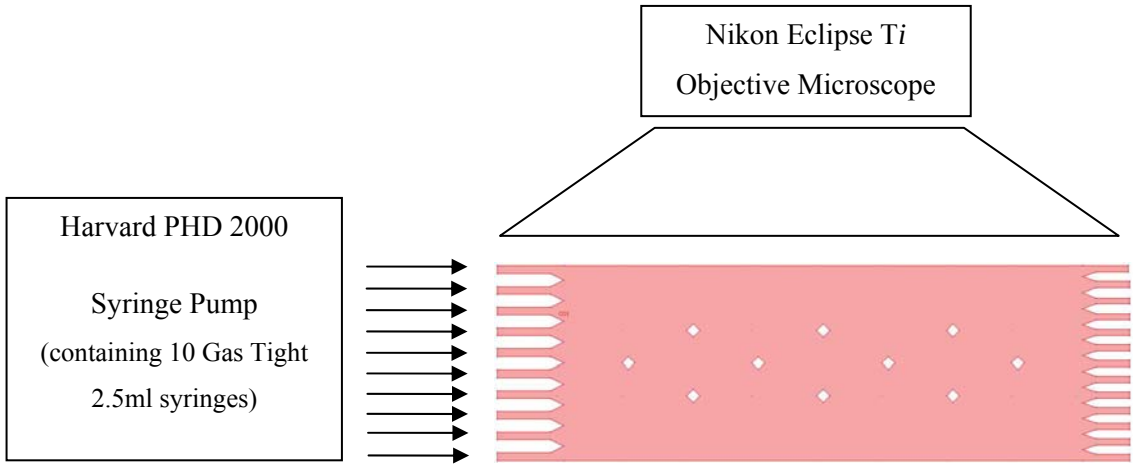


Figure 11 Diffusion Experimental Set-up

3.6.3 Experimental Results

The reactor was observed under a Nikon Eclipse T*i* Objective Microscope during the diffusion reactions, Figure 12 shows the initial results. The experimental chamber flow velocity was ~350 μm s^{-1} and takes 20.6 seconds to complete the crossing of the chamber, from inlet to outlet. Initially, it was observed that the streams "waved" or "dipped", towards the outlet stage. This was due to an unbalanced outlet stage (see Figure 15). There are less channel restrictions through the wider channels than the single narrow channel at the top of the reactor; by adding greater pressure restrictions to the lower stage, the flow lines were corrected. The corrected flow may be seen in Figure 12 (bottom).

Post experimental analysis was to study the "amount" of diffusional mixing. The strong colour differential between the single reagents and the diffused complex, allowed the colour intensity to be used as a suitable measure, Figure 13.

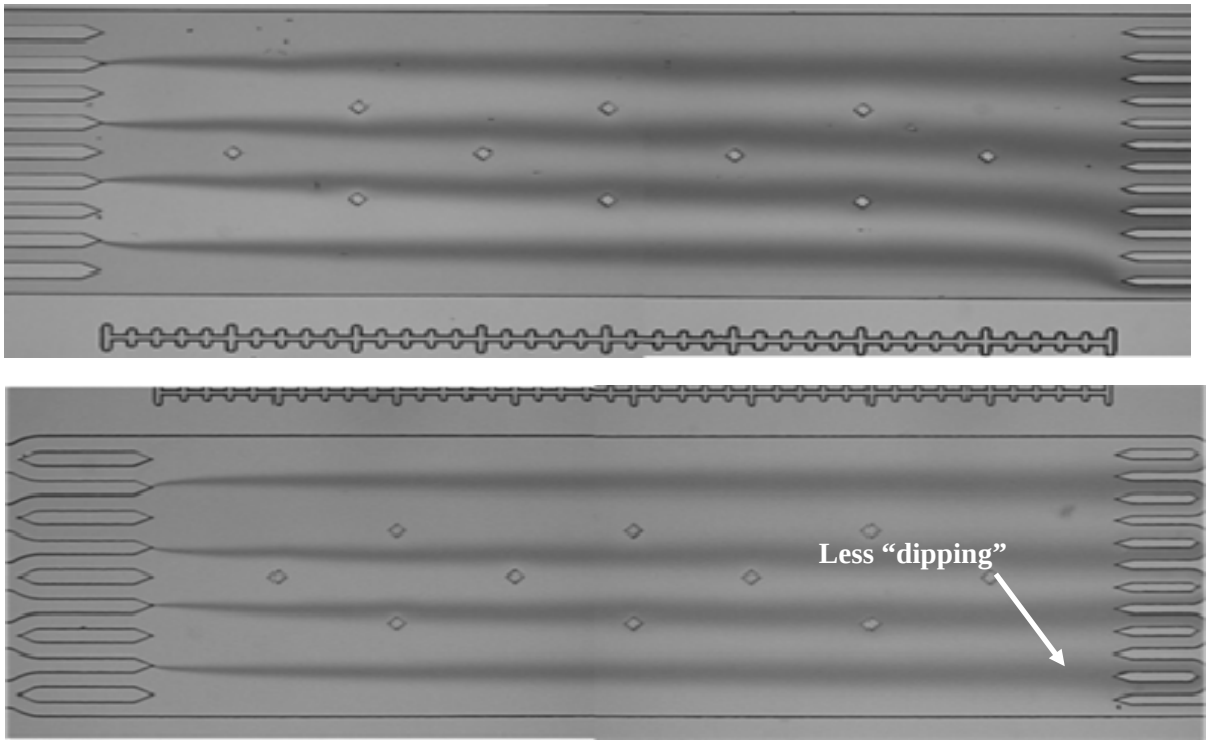


Figure 12 (Top) Original Diffusion Experiment; (Bottom) Corrected Outlet Diffusion Experiment

Peyman concludes that the zone of diffusion appeared dark and produced a lower greyscale value; therefore the amount of diffusion near the outlet could be measured from the plot to be 340μm on average [121].

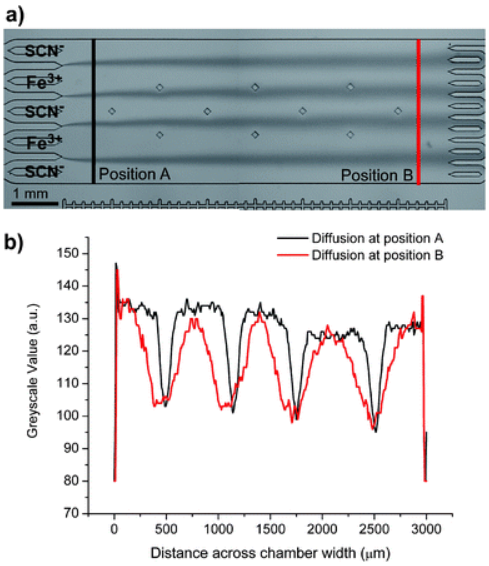


Figure 13 a) Grayscale Diffusion Plot; b) Measured Diffusion Intensity Plot [121].

The diffusion theoretically calculated value from the Einstein-Smoluchowski Equation 6 was found to be 420µm. The difference was stated as being temperature dependant, experimentally the temperature was 21˚C compared to the standard theoretical temperature of 25˚C, given for the published diffusion coefficients, a difference of 19%.

$$t = \frac{x^2}{2D}$$

Equation 6 Einstein-Smoluchowski

Where t is time (s), D is the diffusion coefficient (m^2s^{-1}) and x is the diffusion distance (m). The error between Peyman and theory was 19%.

3.6.4 Simulation Set-Up

The model geometry was imported into COMSOL from the photo-plot mask (.DXF) of the reactor which was used to etch the experimental reactor, this method ensures geometrical accuracy between the physical and simulated reactor. An Incompressible Navier-Stokes application layer was used to describe the fluid flow. It was determined through simulation that an inlet velocity of 850µm/s, resulted in a chamber velocity of 350µm/s matching the reported experimental velocity, verified by boundary integration. COMSOL provides a “massless” particle trace feature. This showed that for the given flow rate the particle took 21 seconds to flow between inlet and outlet, Payman recorded a value of 20.6 seconds, a difference of 1.9%.

The simulation model was extended to include two individual Convection and Diffusion application modes, one to describe Fe^{3+} (Iron Sulphate), $0.62x10^{-5}cm^2s^{-1}$ and one for SCN^- (Thiocyanate), $1.758x10^{-5}cm^2s^{-1}$.

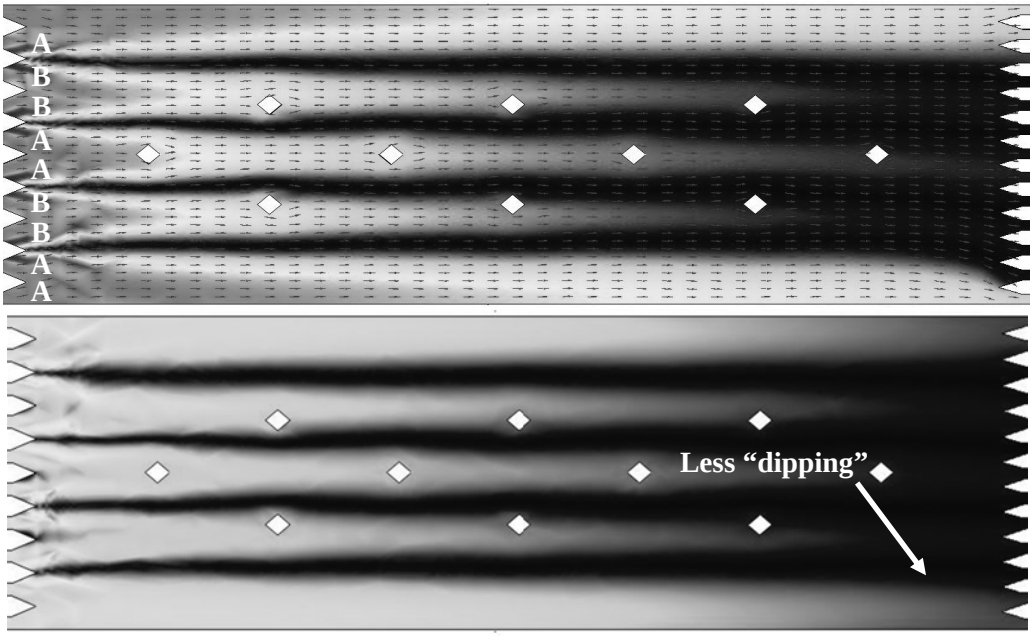


Figure 14 Diffusion Model Post-Plots; (Top) Original; abs(c-c2) (Bottom) Corrected; abs(c-c2).

Alternate inlets were assigned for each application layer, shown in Figure 14. For example, in one application layer AA = 10μM and BB = 0μM and then in the second application layer the assignment was AA = 0μM and BB = 10μM, while this is not true of the concentrations found in the experiment, it allowed the combined diffusion to be determined using, abs(c-c2) during post-plot, which was required for experimental comparison.

Figure 14 (Top) shows the simulated diffusion of the original reactor, without the corrected outlet stage. The plot is shown using a grayscale colour-map in COMSOL for abs(c-c2). The default range produces simulation artifacts, where in parts of the geometry the mesh density is too coarse, resulting in potentially marginally negative concentrations or concentrations that are outside the “possible” range. These artefacts could be attenuated at the expense of high computational overhead, namely increasing mesh density. In the simulation results described these regions occur in low quantities around the inlets, therefore, they may be worked around during analysis. An example of this is shown in Figure 14 (Bottom) where the range has been bounded to 0 (light) to 20 μM (dark). By bounding the range in such a way much of the surplus information to our analysis is removed and the diffused flow streams become more apparent. A further step, while maintaining these bounds is to show the analysis in grayscale, Figure 17.

As previously noted from Figure 12 (top) it was observed that the real outlet stage was unbalanced with unequal flow at the outlets. The same was true of the COMSOL model, demonstrating good validation between simulation and experiment since it too suffered from "dipped" flow when using the same CAD file, Figure 14. The same pressure restrictions were added to the COMSOL model, Figure 15.

The major advantage of simulation is the power of analysis; a cross-domain velocity analysis was performed at Position B (see Figure 13) of the reactor to measure velocity in the x-direction, the results of which are shown in Figure 16.

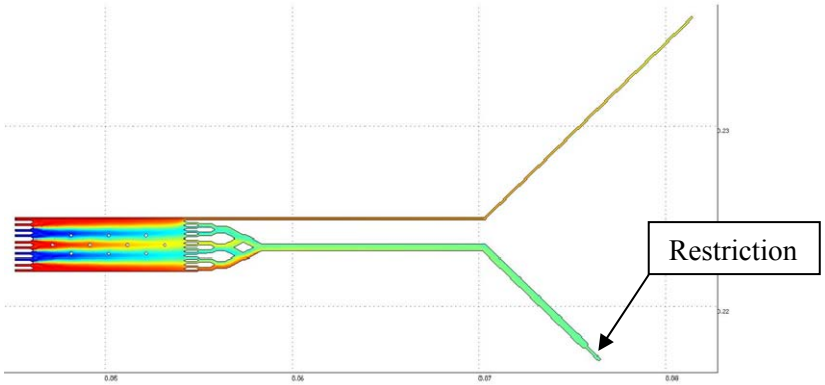


Figure 15 Corrected Outlet Stage

This provides useful visualization of the flow profile and quantifiable evidence of the effectiveness of the correction applied to the outlet stage, more so than can be determined from the diffusion plots in both the experimental and simulation case. This level of analysis is unachievable in an experimental environment, at least without significant modification to the reactor design.

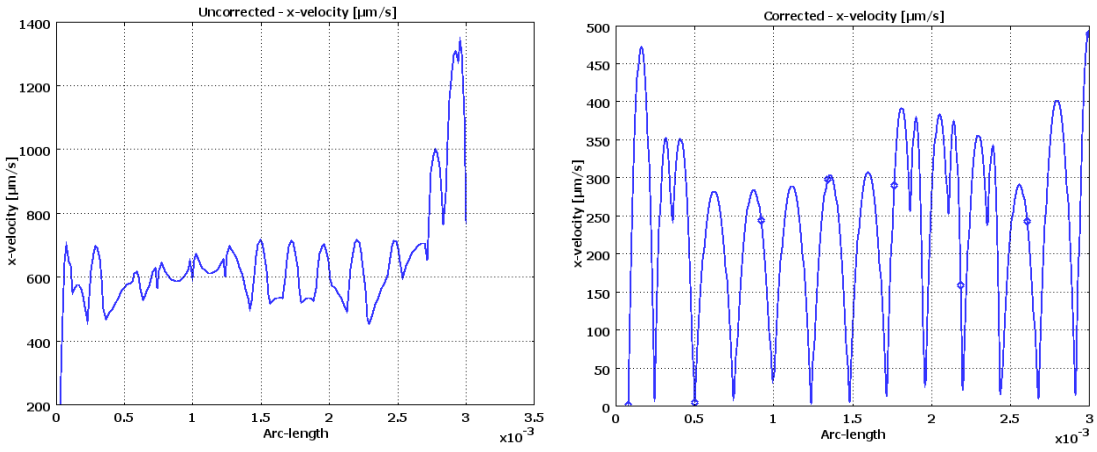


Figure 16 (Left) Un-Corrected outlet Stage Flow; (Right) Corrected Outlet Stage Flow

```
image1 = imread('stream4.jpg');
profile1 = improfile(image1, [81,81],[14,435],500);
profile2 = improfile(image1, [1.1270e+3,1.1270e+3],[14,435],500);
figure(1)
plot(1:1:size(profile1(:,:,1)),profile1(:,:,1));
hold on
plot(1:1:size(profile2(:,:,1)),profile2(:,:,1),'r');
hold off
```

Figure 17 MATLAB Image Analysis Code

Image Analysis was performed in MATLAB, Figure 17 on the grayscale plot (stream4.jpg), Figure 18 to produce Figure 19, a quantifiable measure of the diffusion for comparison.

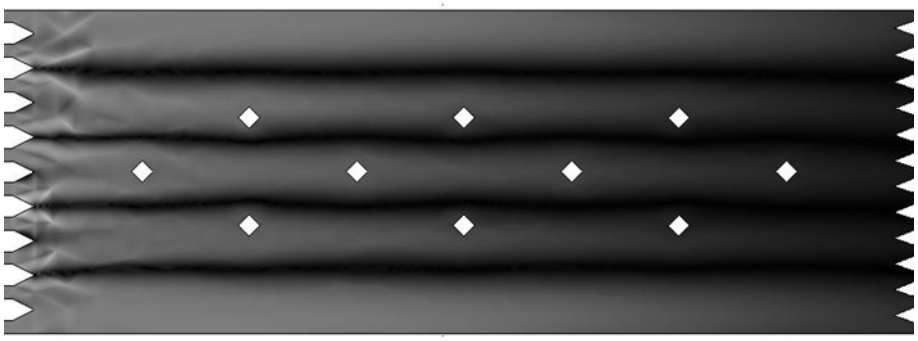


Figure 18 Grayscale Plot of Bounded Analysis

From comparisons between Figure 13 and Figure 19 it may be observed that the simulation results show higher diffusional mixing at Position B than experiment, shown by lower intensity values.This may be attributed to the "ideal" nature of the simulation, over experimental conditions.

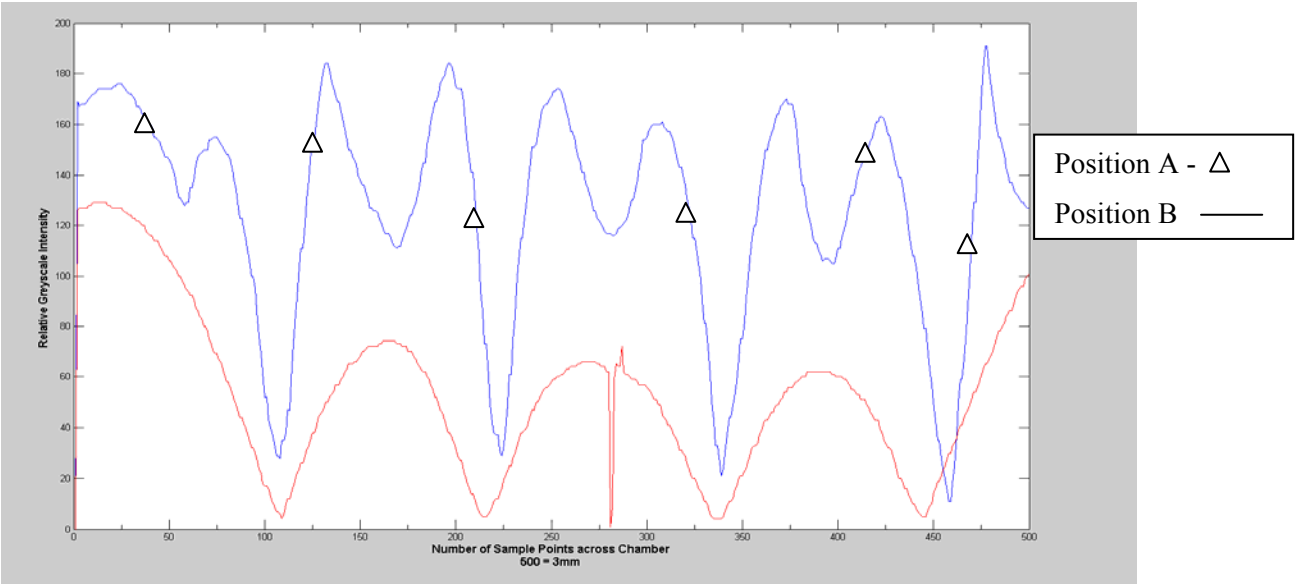


Figure 19 Simulation Greyscale Intensity Plot; Position A (top plot) and Position B (bottom plot).

The average pixel separation, between position A and position B, from our image analysis, for Position A is 45 pixels and the average pixel separation of Position B is 95 pixels, converting to distance is 270um and 570um, respectively. This provides an average diffusion distance of 420um which agrees exactly with the papers calculations [121].

3.6.5 Diffusional Mixing Summary

The import of the photolithography CAD file into COMSOL reduces simulation set-up work load of the microfluidic researcher. The coupling of the application modes shows that layer by layer more complex system descriptions can be modeled, and modeled with a high level of inherent accuracy. The "dipping" of the flow streams of both the experimental and simulation system represent a high degree of cross-validation, with such high confidence and an abundance of post-plotting analysis tools, the ease and ability of validating and optimizing system performance is being proven. Peyman (experimental) and COMSOL flow times for crossing the geometry chamber agreed to within 1.9%, furthermore, the measured COMSOL diffusion distance agreed exactly with Peyman's theoretical value, but varied by 19% with the experimentally measured value, all present a high degree of accuracy, subject to controlled experimental conditions.

3.7 Electro-osmotic Flow (EOF)

3.7.1 Introduction

Electro-osmotic flow applications are receiving much attention from the microfluidic community, due to their low dependence on mechanical pumping methods and high configurability. The Chemistry department in the University is exploring EOF as a transport mechanism in an application which will be investigated in Chapter 8. However, the published work by Sun *et al.*[58] is used here for cross-validation due to the level of detail and quantified results they published.

Sun *et al* [122] investigated two key design features of a microfluidic electro-osmotic cell sorting application, the velocities within the channels, and the optimum switching geometry. Their investigation explored the use of simulations as a means of assessing different design geometries allowing rapid parameter optimisation. They verified their simulations through cross-validation to experimentation, the results of which were reported in their paper and used here.

Our interest is initially to verify our simulations against their results, to further validate the COMSOL simulation approach on a different type of system before considering fault simulations.

3.7.2 System Overview

Different types of blood cell are mixed with fluorescent markers off-chip and are then presented to the inlet reservoir. Cells are transported around the reactor using EOF, which is controlled via Platinum electrodes at the inlet and outlet reservoirs. Sheath flow is implemented in the experimental system to pre-align the cells, one by one, prior to fluorescence excitation by the laser diode. An integrated laser diode coupled with a photo-detector, provides a compact optical detection system. This is not considered in the simulation. The fluorescent signal determines which outlet the cell is switched into. EOF switching is implemented in the experimental and simulation system and will be investigated in this section. EOF switching is achieved by switching the applied electrode voltage at the outlet reservoirs. The reactor design is based on a "Y" channel.

3.7.3 Electro-osmotic Flow Theory

Electro-osmotic flow is a complex science and an area of research in its own right, its theory and mathematics are out of the scope of this thesis. The reader is referred to the literature [123].

COMSOL aids the implementation of EOF for a given geometry through the use of the Conductive DC media application mode, which contains pre-built mathematical equations which may be applied transparently to the geometry. Advanced configuration of the underlying mathematics may be achieved and the reader is referred to the COMSOL documentation *MEMS User Guide.*

A basic understanding of EOF theory is required by the user. In the simplest implementation two coupled application modes are required; one to describe the distribution of the electric field (conductive DC media, emdc) and the second to describe the motion of flow (Stokes Flow, mmglf). Coupling is achieved through the boundary conditions. In the mmglf application mode, the "wall" boundary is assigned the "electro-osmotic velocity" vector; Ex_emdc, Ey_emdc and Ez_emdc. This provides the coupling between the two application modes to allow the Electric Double Layer (EDL) mathematics to be evaluated, resulting in the flow behavior description due to the applied electric field.

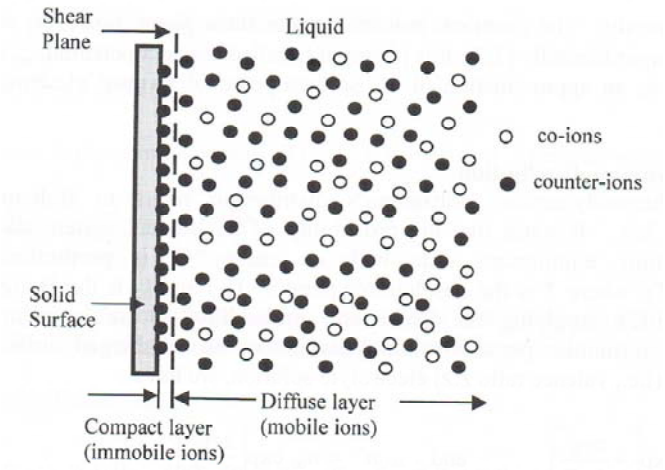


Figure 20 EDL ion diagram [124]

The EDL describes the formation of the solid-liquid charge interface, Figure 20. The immobile ions in the compact layer remain stationary under the application of an electric field, however the ions in the diffuse layer are mobile and move under the influence of an electric field, the movement may either be anodic or cathodic dependent upon the polarity of the Zeta potential. The movement of the diffuse ions results in bulk flow.

3.7.4 EOF Velocity Model[1]

Here the channel between the two reservoirs (inlet and outlet) is considered. The velocity of the electro-osmotic flow is initially investigated along the straight channel between an inlet and outlet reservoir. An overview of the model set-up is presented in Figure 21, and reflects the implementation by Sun *et al.*

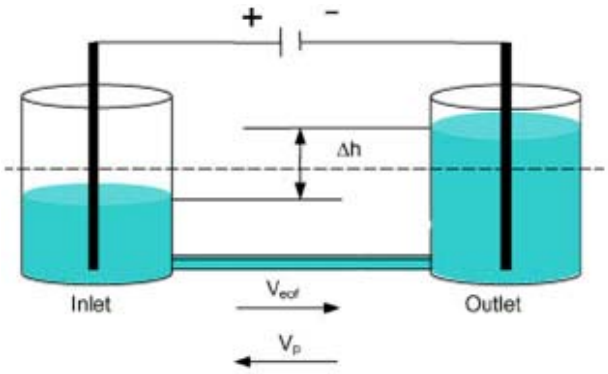


Figure 21 Sun Channel / Reservoir Geometry [58]

Where Δh is the height difference caused by flow, V_{eof} is the velocity due to electro-osmotic flow and V_p is the velocity due to back pressure.

[1] Model File: sunmodel.mph

The geometry studied for our simulation is further simplified by not modeling the geometry of the reservoirs, but modeling the effect of the reservoirs mathematically. The dimensions of the reservoirs are provided Table 1 and their effect mathematically calculated at each simulation time step, as though they physically exist. The geometry discussed is a 100µm wide and 40µm deep channel. FEM geometries become very inefficient when very different dimensions of scale are implemented; in this case the µm dimensions of the channel width and depth and the mm scales of the reservoirs diameters and height. Obviously the mm dimensions between the two reservoirs are unavoidable. The derived pressure is written to the outlet boundary condition, Figure 22.

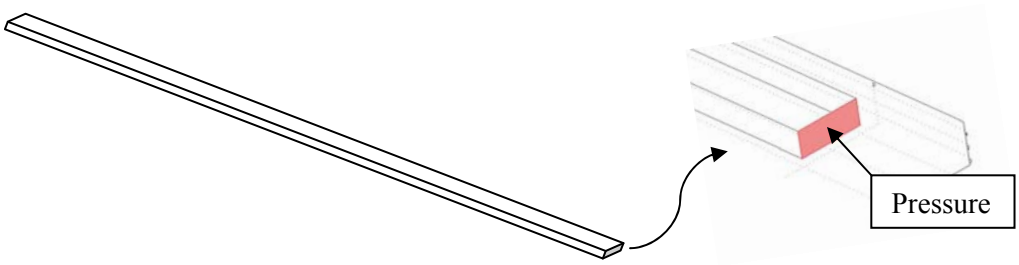


Figure 22 Simplified Channel Model (inset) Outlet Boundary

The EOF velocity relies upon multiple physical parameters along with the reservoirs dimensions. These are shown in Table 1. This demonstrates the flexibility of the modeling approach; FEM geometries, and separate but interacting equations, allows decreased computation time by reducing the number of non-essential elements. It may be observed from Figure 21 that the inlet and outlet reservoirs initially have their fluids at the same level.

Parameter	**Value**
Channel Width [µm]	100
Channel Depth [µm]	40
Channel Length [mm]	5
Reservoir Diameter [mm]	4
Reservoir Height [mm]	2
Electric Field [V/cm]	300
Buffer Density [kg/m^3] (H_2O)	1000
Buffer Viscosity [m^2/s]	10^{-9}
Relative Permittivity	78.5
Zeta Potential [V]	-0.0295

Table 1 Physical Parameters

EOF moves the liquid from the inlet to the outlet, causing a difference in height between the two reservoirs and introduces a head, resulting in a reverse pressure driven flow Equation 7, which perturbs the EOF, which is flowing in the opposing direction.

$$v_p = -\frac{\Delta h.\rho.g.D_h^2}{2.f.Re.\mu.L}$$

Equation 7 Linear Flow due to Height Differences

Where Δh is the difference in height, ρ the density of solution, g acceleration due to gravity, $f.Re$ the anning fraction multiplied by the Reynolds number, μ solution viscosity, L length of the channel and D_h the hydraulic diameter. Therefore the average flow in a channel is the summation of the flow due to EOF and the pressure driven back flow, Equation 8.

$$v_{ave} = \frac{-\zeta.\varepsilon.E_x}{\mu} - \frac{\Delta h.\rho.g.D_h^2}{2.f.Re.\mu.L}$$

Equation 8 Average Flow due to EOF and Pressure Driven Flow

The first collection of terms in Equation 8 describe EOF flow, ζ the zeta potential, ε the relative permittivity, E_X the applied electric field and μ the electro-osmotic mobility. Pressure driven back flow is calculated per simulation time step, using the equations desired, the resulting back pressure is written to the outlet boundary condition to model the effect of the reservoirs. The inset of Figure 22 shows the outlet of the simplified channel, where the outlet reservoir would be situated. The dimensions of the reservoirs are entered prior to the simulation, as "constants", along with other system constants, and are assigned; *Res_Radius* and *Res_Height*, in this case both take values $2x10^{-3}$[m]. From this the *Res_Vol* is calculated, Equation 9.

Res_Vol = ((pi*(Res_Radius^2))*Res_Height)

Equation 9 Calculating the Reservoir Volume as a COMSOL constant

The rate at which Δh changes should be proportional to the volumetric flow rate, *VolFlow*, calculated every simulation time step by performing a boundary integration over the outlet boundary for U_mmglf. Three Global Expressions are calculated each time step to calculate the Back Pressure due to the height difference. The first expression is *VolTotal* which calculates the total volume of solution which has left the outlet, this is the current simulation time step multiplied by *VolFlow,* Equation 10, assumes constant flow.

VolTotal = (VolFlow*t)

Equation 10 Calculating the Total Volume leaving the channel

delta_H is the calculated height difference between the reservoirs, Equation 11. The fraction (VolTotal/Res_Vol) determines for a given flow volume the rise in reservoir fluid height.

$$delta\_H = ((VolTotal / Res\_Vol)*(Res\_Height*2))$$

Equation 11 Calculating the Reservoir Height Difference

Finally, the back pressure is calculated, *Back_Pressure*, Equation 12.

$$Back\_Pressure = ((\rho*g*delta\_H)$$

Equation 12 Calculating the Back Pressure

Where $\rho = 1000kg/m^3$ is the density and g = 9.81m/s the gravitational constant. The value of pressure driven flow, *Back_Pressure*, is then applied to the "outlet" boundary.

3.7.5 Cross-Validation of Simulation & Experimentation

A simple single channel model was implemented using the geometry and physical properties previously discussed. In order to evaluate the model in its simplest form it was simulated neglecting the back pressure equations. Sun *et al* reported 695 µm/s for their simulation velocity; ours shows a little over 600 µm/s, for an electric field of 300V/m, ~13.6% difference.

With the addition of the back pressure, Back_Pressure, on boundary 2 (outlet) of the channel the velocity profile across the width of the channel (arc-length 100µm), is shown in Figure 23.

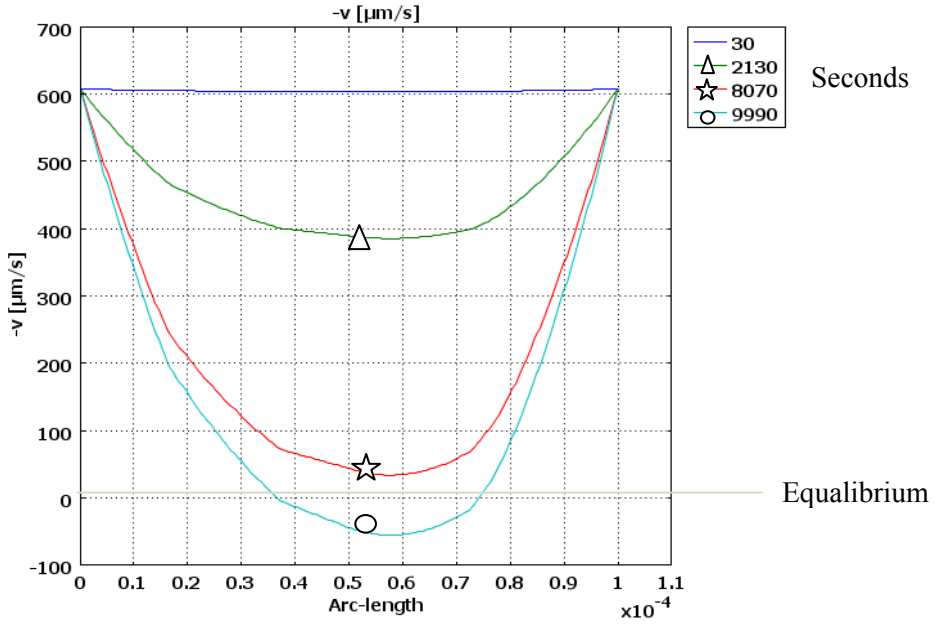


Figure 23 Velocity Profile with Hydrostatic Counter Pressure

Each plot on Figure 23 represents a simulation time step, for clarity the simulation is plotted at 30, 2130, 8070 and 9990 seconds. It may be observed that the velocity profile soon starts to be perturbed by the back pressure. This effect continues to increase, and at approximately 9000 seconds the two flows, EOF and flow due to back pressure, reach equilibrium (0 m/s). At 9990 the dominant flow is hydrodynamic flowing backwards; more precisely shown in Figure 25.

The velocity flow arrows of Figure 24 show the forming of the reverse flow parabola, and eventually reverse flow. The magnitude of flow at the walls remains largely unchanged, agreeing with Figure 23. To analyse back pressure the instantaneous flow is studied for the outlet, Figure 25. This determines when the flow rate reverses and enables a simulation time to be read off the x axis.

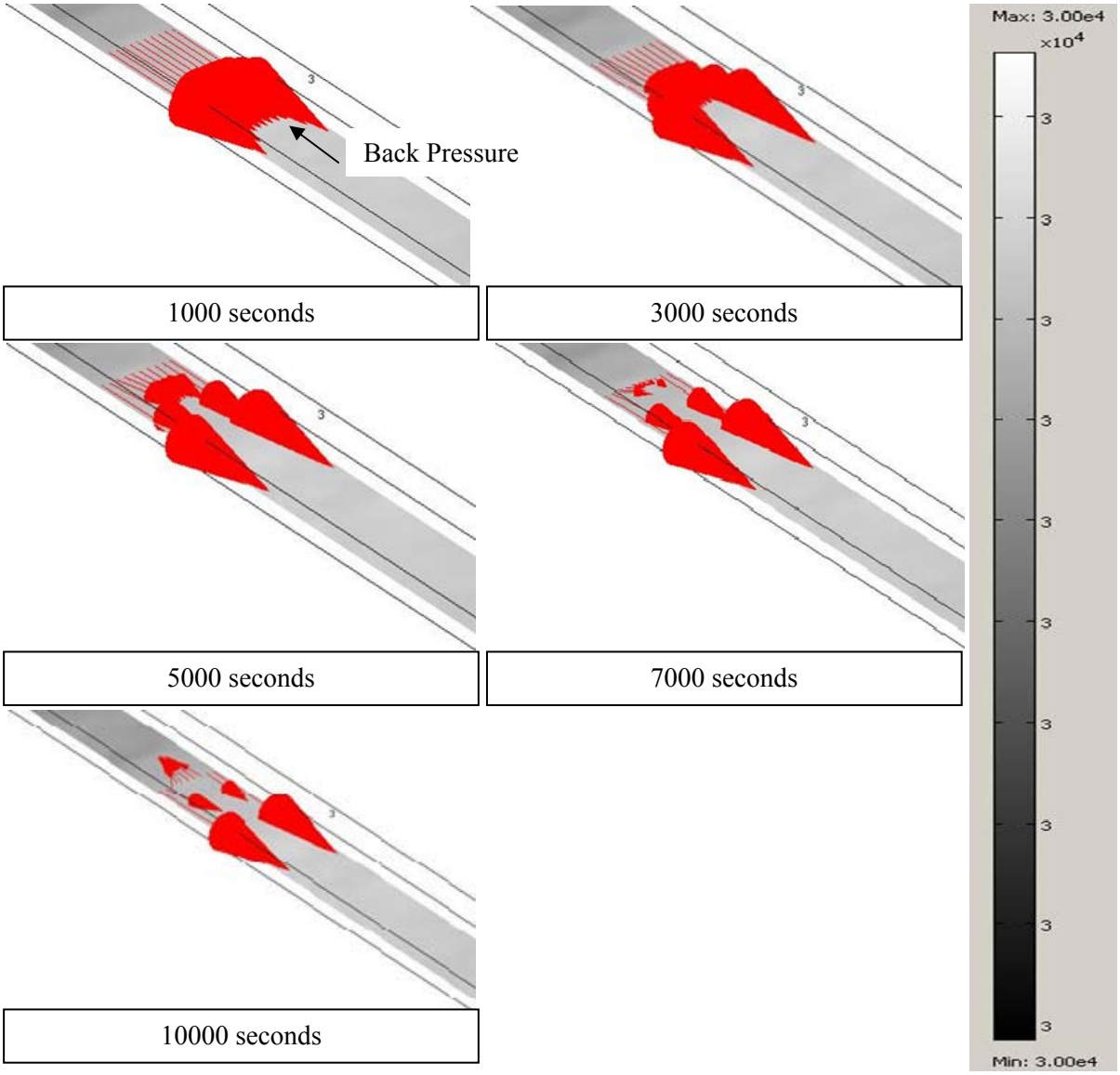


Figure 24 Velocity Slice Plot with Proportional Flow Arrows

In Figure 25 the flow rate starts at a little over 600 µm/s at t=0, closely approximating the results reported by experimentation at a value of 695µm/s a difference of 13.7%. The t value where zero flow occurs in our simulation is 0.83 x 10^4 seconds (Figure 25). From the back pressure plot in Figure 26, the time value, 0.83 x 10^4 seconds equates to a pressure of ~17Pa. the published where flow reversal occurs is at a pressure of 14.7 Pa, a difference of 13.5%. While our flow and pressure simulation results closely approximate the published values, the time at which this occurs in simulation is very different from the published experimental work, a factor of 34.5.

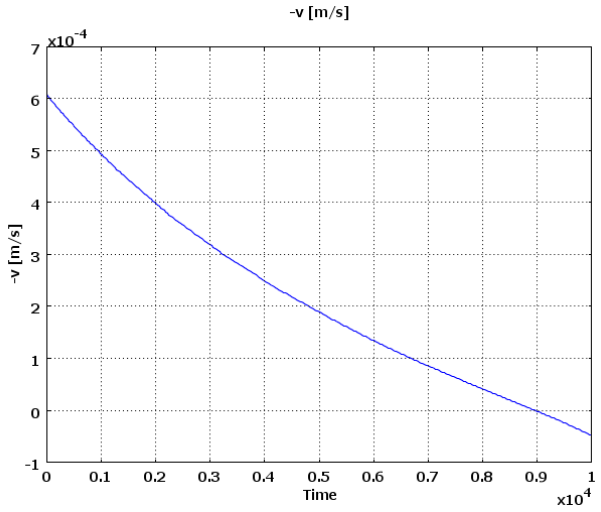


Figure 25 Simulation instantaneous flow (µL/s)

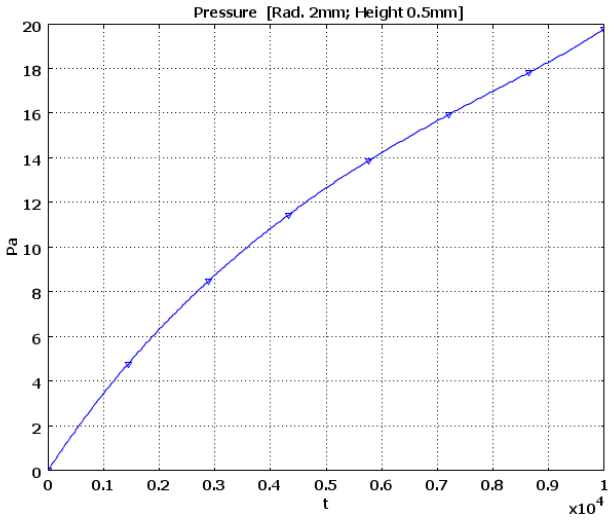


Figure 26 Simulated hydrostatic pressure with time (seconds)

Sun *et al.* state t = 240 seconds, and the COMSOL simulation predicts t = 0.83 x10^4 seconds, a factor of 34.5. Tthe difference is attributed to differences in transported volumes. Back pressure is caused

because of a height difference in the inlet and outlet reservoirs. In an ideal system this difference would be caused solely by transporting fluid from the inlet to the outlet. However, Sun *et. al.* report experimental non-idealities, namely, electrolysis and evaporation at the electrodes (inlet and outlet). Therefore, the pressure difference and resulting flow is no longer determined by the fluid transported, but by losses as well as transportation. This is the reason Sun *et al.* measure a collected volume of 0.305µL compared to the simulated outlet collected volume of 9.2 µL for the same pressure; and this is the reason for the increased time factor of 34.5. It takes longer to transport the fluid to cause the height difference than it does to evaporate the fluid.

In the simulation environment ideal conditions resulted in collected and transported volumes being equal. To aid cross-validation given the reported problem the Sun *et al.* transported volume was extrapolated over time, representing the conditions if electrolysis and evaporation were neglected, Figure 27.

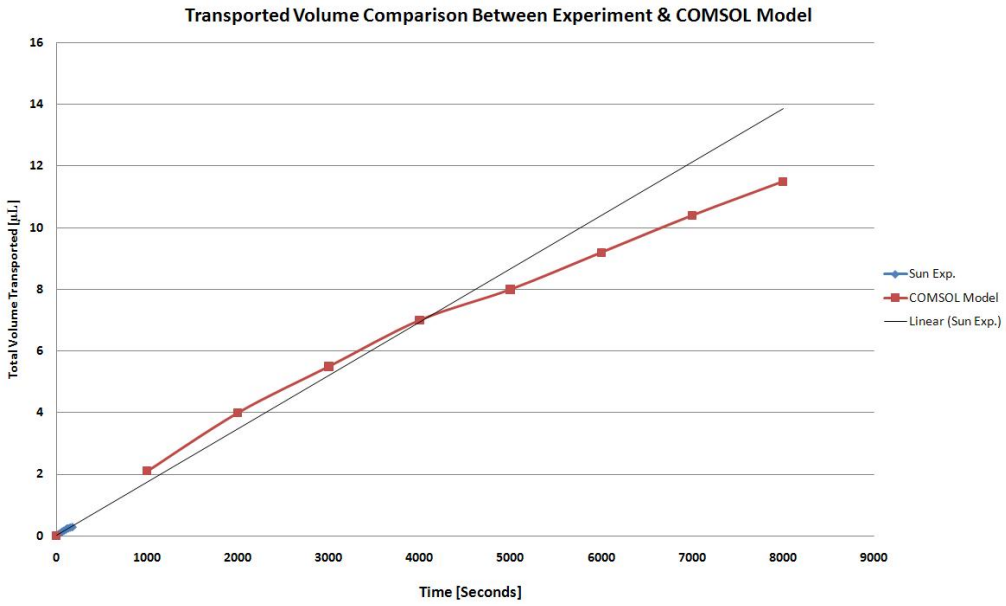


Figure 27 Experimental and COMSOL Instantaneous Flow

The trendline closely approximates the COMSOL simulation (15%) showing that the COMSOL simulation and experimentation would closely approximate.

3.7.6 EOF Summary

This section has shown that the features of COMSOL can be layered to provide an accurate description of a complex system, such as one which utilizes EOF as a transport mechanism. Furthermore in this case it has been shown how using a combination of intrinsic variables associated with the

application modes (velocity) and global expressions, the systems behavior can be extended to describe experimental phenomon such as hydrodynamic back pressure. Cross-validation of the velocity and velocity channel profile showed a high degree of accuracy ~13.6% difference.

This section has also demonstrated how experimentation and simulation can deviate due to poor experimental set-up and operation (evaporation and electrolysis), further highlighting the potential problems associated with practical microfluidic systems. This said when correction factors were applied the experimental and simulation difference was reduced to ~15%.

3.7.7 Switching Mechanism Simulation[2]

One of the aims of Sun *et al.'s* research was to optimise the cell sorting geometry for switching speed (efficiency) and accuracy (sorting). In approaching this task they investigated two angles of the "Y" structure; 90˚ and 180˚. The outcome of their study is that the optimum angle for switching speed and accuracy is 90˚. Since our interests here are not with optimisation, but with model validation and ultimately test, we chose to work with the most optimum.

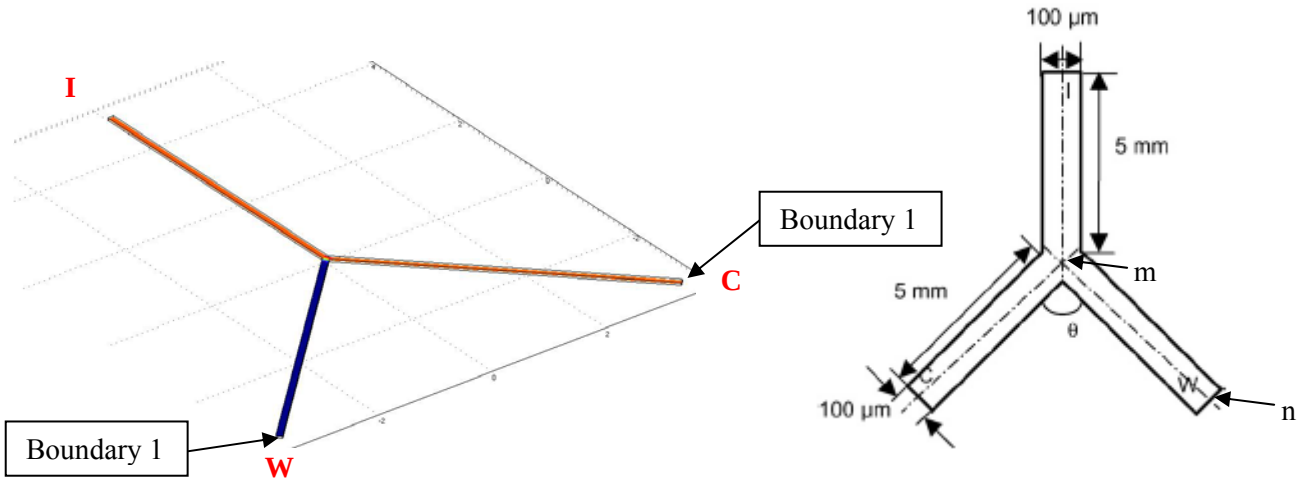


Figure 28 Switching Model Geometry (Left) COMSOL Geometry, (Right) Sun Geometry

Switching is achieved by diverting the EOF flow. The simulation approach consisted of maintaining I = 100V and switching W and C between ground and *floating potential* as described in the paper and shown here in Figure 28.

Switching is first considered "statically", that is where the switch conditions are set prior to the evaluation of the simulation. This allows the switched velocity and leakage distance to be measured, and to initially validate the basic simulation before increasing the complexity. An example of a switch would

[2] Model File: SunModelSwitching.mph

be with C at ground potential and W floating, the velocity was measured at C through point evaluation (point located at -3.425e-3,-3.425e-3,20e-6) the measured velocity in the COMSOL simulation was 204μm/s, compared to the Sun *et al.* simulation which produced 229 μm/s. Therefore, the accuracy between the COMSOL and Sun model is 7.3% for the switched velocity.

The leakage between the switched outlet and the un-switched outlet could lead to sorting inaccuracies, therefore the switching leakage distance is considered, along the dotted line (m to n) shown in Figure 28. Sun *et. al* investigate how the angle of the outlet channels influence the leakage distance, Figure 29.

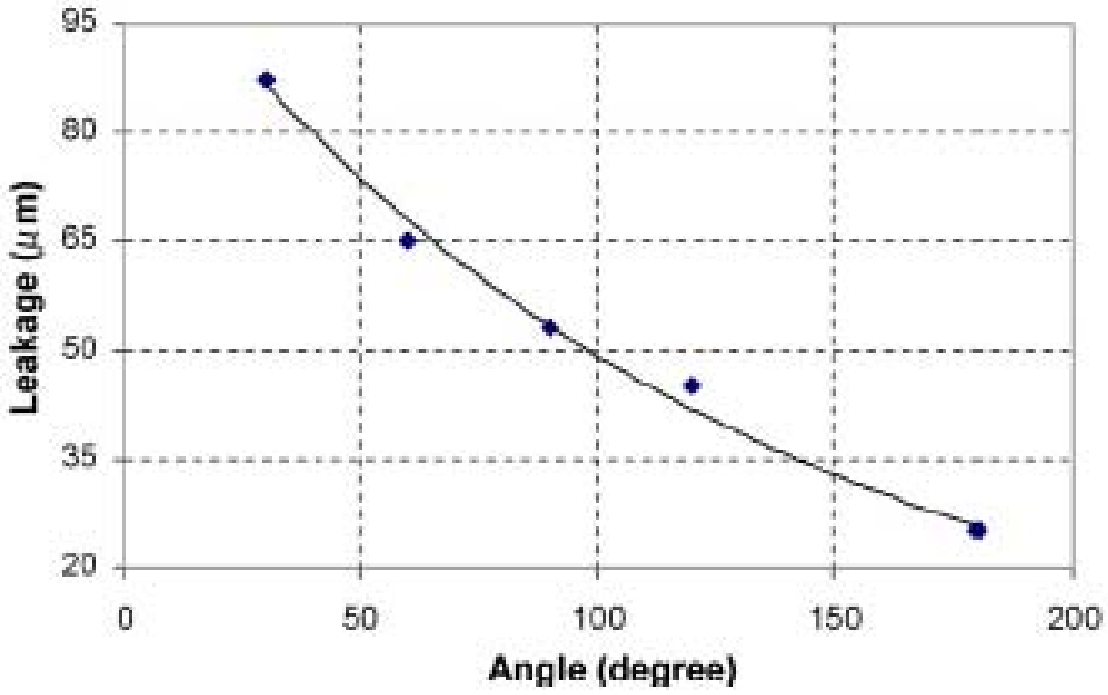


Figure 29 Relationship between Outlet Channel angle and leakage distance [58]

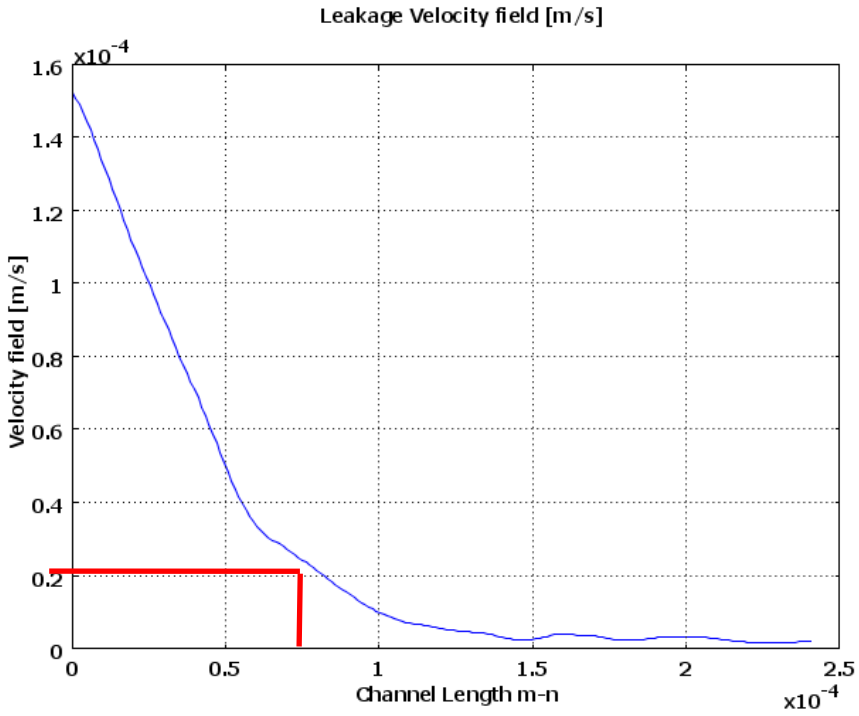


Figure 30 COMSOL Leakage Distance

Figure 30 shows the leakage distance obtained for the simulated geometry. Sun *et. al.* method of determining the leakage distance is to consider a channel's velocity *switched* when the velocity is reduced to 10% of the full flowing velocity, which in this case is 20µm/s. The distance is approximately 80 µm, compared to 53 µm in Sun simulation and 55.2 µm (Figure 29) for the un-switched channel in their experiment, an error of 33.8%.

3.7.8 Dynamic Switching Simulation[3]

Previously "static" switching conditions allowed for the assessment of switched velocity and leakage. Switching speed is also required, therefore the simulation must model the act of switching, requiring the dynamic assignment of C and W electrode boundary conditions. Switching determines the cell throughput of the system and affects the systems overall accuracy. An overview of switched simulation plots and measurement points is shown in Figure 31. Figure 31 (Left) shows the COMSOL switching junction and Figure 31 (Right) Sun *et. al.* simulation velocity plot.

The Conductive Media DC application mode governs the switching, therefore boundaries 1 (W) and 11 (C) are assigned the boundary condition *electric potential* in conjunction with the variables V1 and

[3] Model File: SunModelSwitching.mph

V2, respectively. In the physics settings > Global Equations we create two "states", V1 and V2 with initial conditions equal to 0[V] and associated weak constraints[4] *test(V1)*I1* and *test(V2)*I2*, respectively.

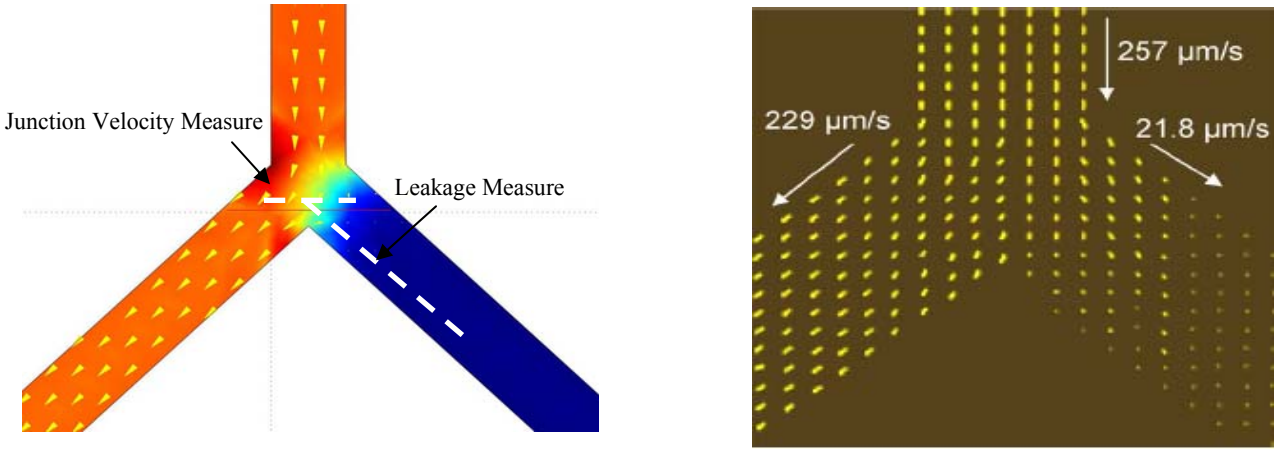


Figure 31 (Left) COMSOL Switched Slice Plot; (Right) Sun Switched Simulation

When an outlet leg is switched on it should be pulled to ground, therefore the *Electric Potential* should be 0V while the other leg should be *floating*. For the ground condition to occur the electrode (outlet) should have a low resistance (1Ω) which in turn pulls the otherwise floating voltage low. The "floating" condition is implemented by maintaining a high resistance (~10^{30}Ω) on the electrode (outlet).

The following global expressions were then assigned for the whole model. Our simulation theory governed by Equation 13 here is that the boundary resistances, R1 and R2 are controlled using Heaviside functions. R1 (the resistance of the electrode where V1 is applied) is 1 Ω for the first half of the simulation and then switches to 1x10^{30} Ω, R2 (the resistance of the electrode where V2 is applied) is the reverse, (see Figure 32). This in turn allows us to calculate the currents and V1 and V2 through the weak constraint assignment.

R1 = 1+1e30*flc1hs(t-0.5,0.01)

I1 = (V0-V1)/R1

R2 = 1e30-(1e30*flc1hs(t-0.5,0.01))+1

I2 = (V0-V2)/R2

V0 = 0

Equation 13 Global Expressions for Switching

[4] The use of weak constraints in this model is based on the COMSOL tutorial which utilises a similar approach. The theory and fundamental principles of weak constraints is complex and will not be described in this work.

The Heaviside function is a means of conditional testing and applying an alternative value in such a way that the solver is presented with a "smooth" transition, promoting convergence. The second term of the flc1hs function is the scale parameter which determines the resolution of the transition and should be at least 1 order of magnitude smaller than the maximum solver time step. It is 1ms here because this is the same order of magnitude as the switching time.

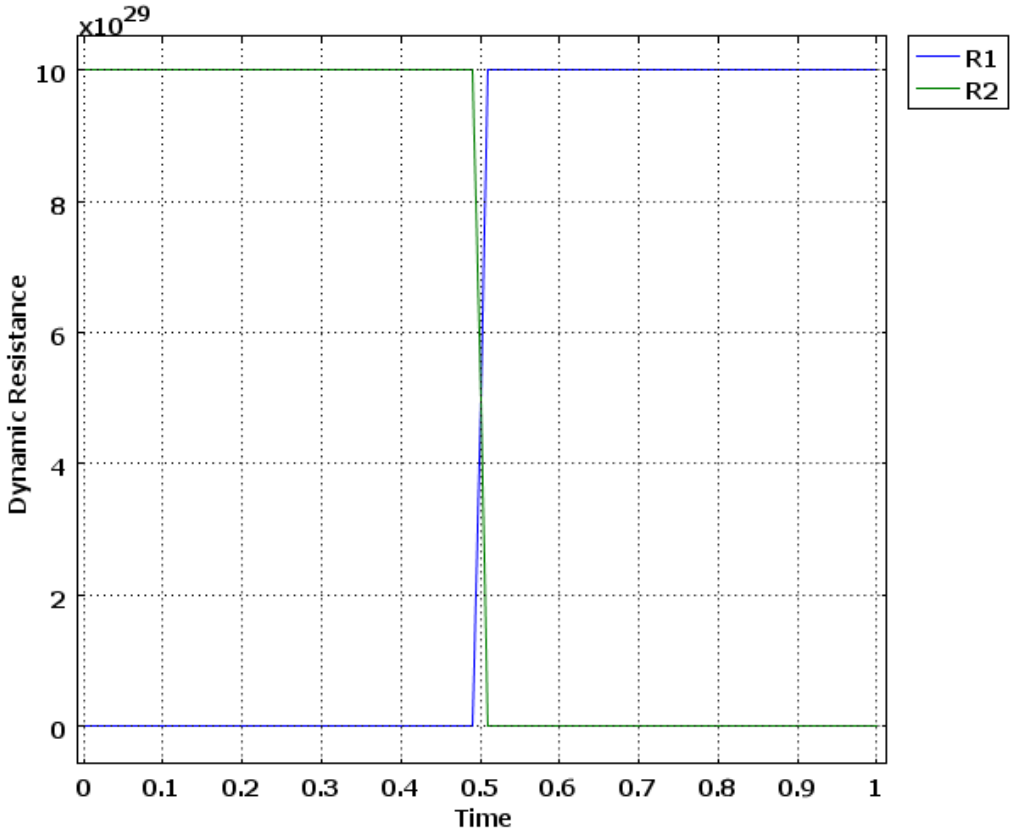


Figure 32 Switching of the Electrode Impedances (Times are shown in Seconds)

3.7.8.1 Dynamic Switching cross-validation

Sun *et al.* describe the point in the geometry where they measure the velocity switching and present the results in graphical form, Figure 33. The point of measurement is denoted "O". Figure 33 shows a minimum velocity magnitude of approximately 90µm/s, the switching time from this arbitrary point to full switched velocity is stated to be 8ms, there is an error on Sun's x-axis scale, shown in Figure 33. They have their x-axis units in seconds not mili-seconds.

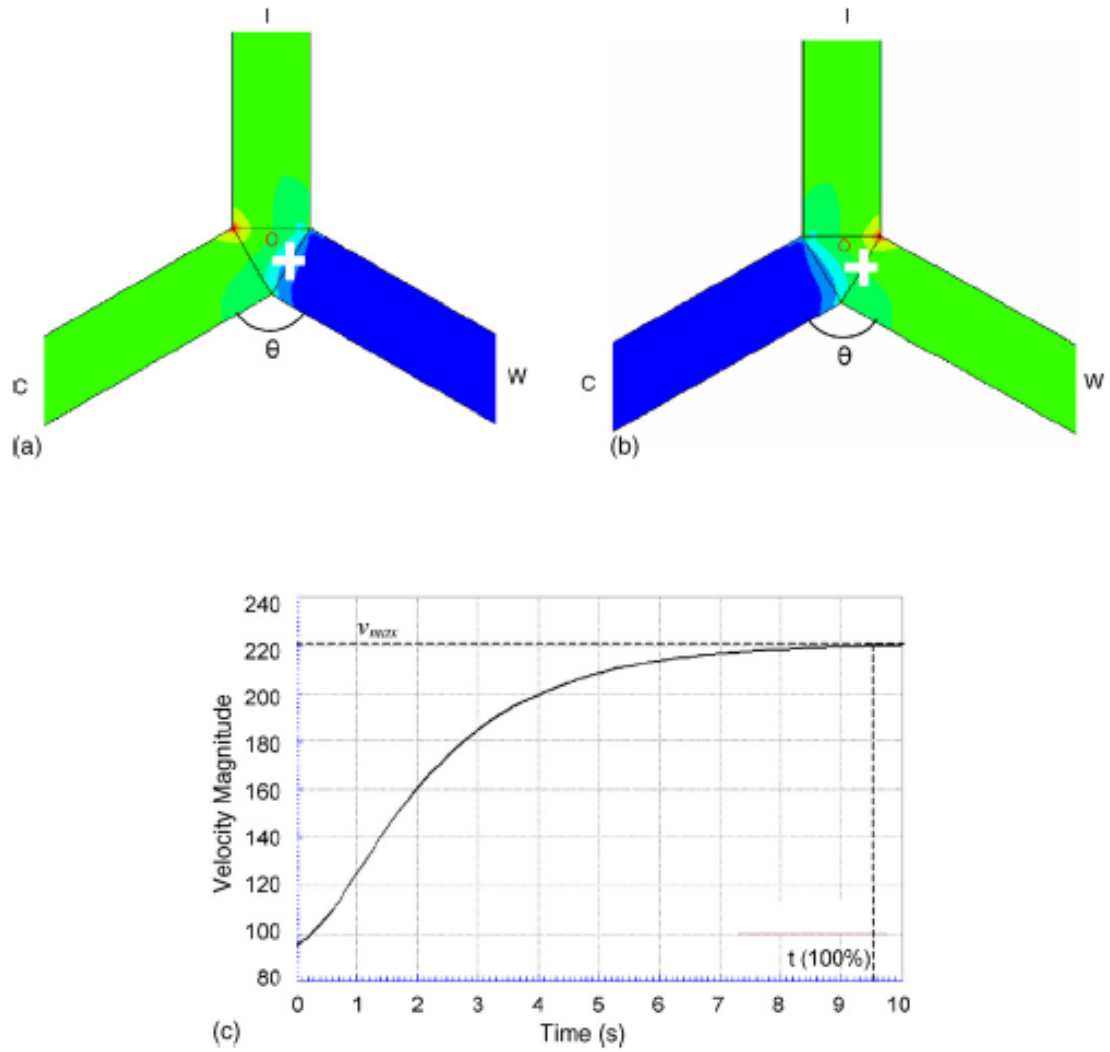


Figure 33 Sun Switching Speed Results [58]

Figure 34 presents the switching speed from the simulation. The total switching time from full velocity (~200µm/s) to the ~10% velocity is approximately 20ms. The switching simulation switched ON and OFF the opposite of the Sun *et al.* model, this was to determine if our simulation contained sufficient information even when used slightly differently, the simulation proved to be robust.

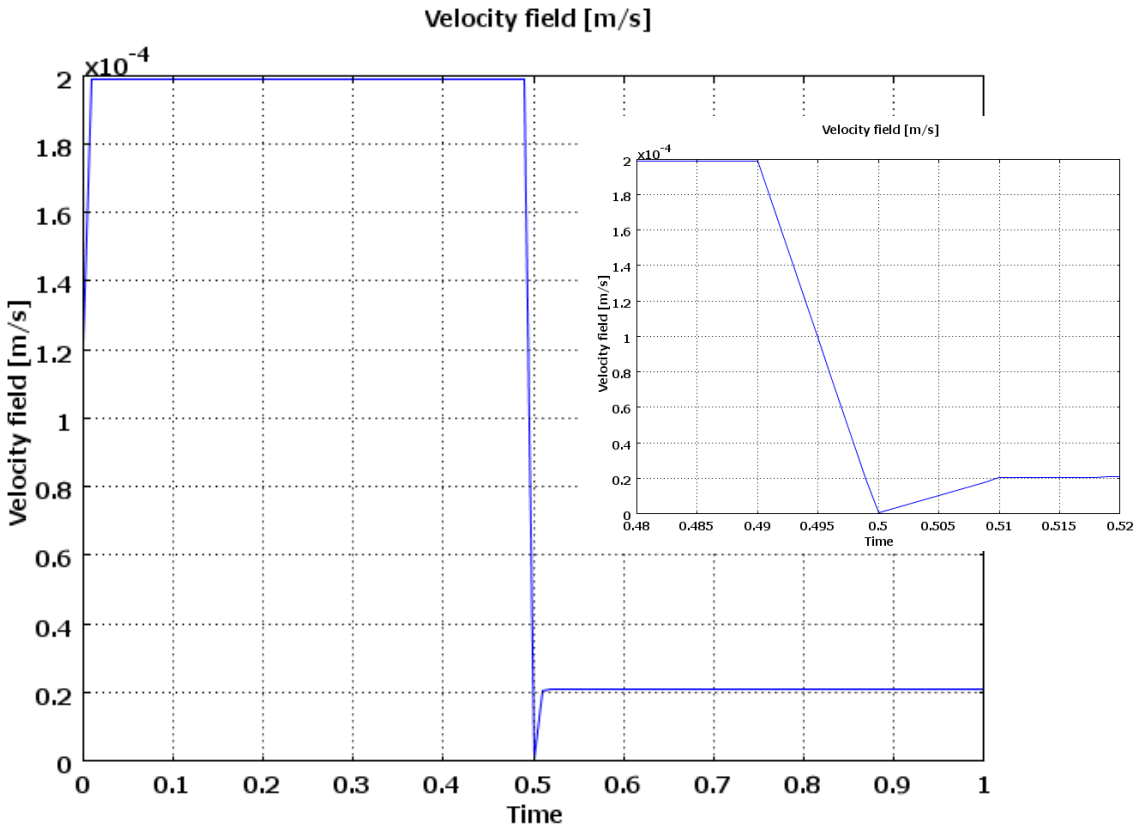


Figure 34 COMSOL Switching Speed (Main) Complete Switch, (Inset) 20ms Switch Close-up (Times in Seconds)

3.7.8.2 Switching Conclusion

Analysis of the switching mechanism continues to show a good degree of cross-validation from the switching leakages 53µm and 80µm (37.5%) to the switching speeds 8ms and 20ms (60%). Furthermore, our model was operated in the reverse direction and an accurate description was maintained. Un-switched velocities were shown to be more accurate having a small difference of 7.3%.

3.8 Conclusion & Discussion

This chapter has introduced a method of fault-free microfluidic system modeling, using COMSOL Multi-Physics, which has proved to be a very powerful and accurate tool. A simulation approach being recommended to microfluidic researchers as an efficient technique for system design and optimization. FEM modeling is the most likely simulation method that will be adapted for heterogeneous system simulation; therefore our fault simulation and test analysis techniques compliment this workflow.

If the proposed fault simulation and test analysis techniques are to be worthwhile, it is imperative that the foundation on which they are based is proven to represent the experimental system with sufficient

accuracy. An assessment carried out on a variety of systems in this chapter, provides a high degree of confidence.

A simple hydrodynamic system was initially simulated to assess simulation cross-validation to an experimental system prior to more complex models being generated. The cross-validation error between measured and simulated flow was determined to be 3.35%.

The diffusion model increased the complexity of the simulation model, by requiring a more intricate reactor design and three application modes. This simulation allowed design exploration, for example in the published work a chamber velocity of 350μm/s was reported, but no inlet velocity to achieve this given, our model calculated this to be 850μm/s. Furthermore, subtle conditions, such as stream "dipping" where described by the models behavior and supported its correction, demonstrating the detailed behavourial description. Our model agreed exactly with theory for the diffusion distance being 420μm.

Finally, the EOF model maintained cross-validation accuracy of 13.7% for EOF flow and 13.5% for calculated back pressure. This model demonstrated how COMSOL may be configured to describe dynamic behavior by using global expressions to determine boundary condition values. While the errors of switching increased to 37.5% for leakage distance and 60% for switching time, these models allow with a basic understanding of COMSOL and the FEM approach allowing the researcher to at least approximate their systems performance.

The models implemented in this chapter were not optimized for accuracy or modified to improve simulation performance since our interest was to capture and simulate the approximate bahaviour of the system for test purposes.

This chapter has proven COMSOL to be an accurate and flexible platform on which to build the fault simulator and test analysis methodology.

Chapter 4 Microfluidic Fault Modelling

4.1 Introduction

In this chapter microfluidic faults will be investigated. The majority of published work in microfluidics reports on developing the science, and the success achieved of breaking new ground. The describing of faults and failure modes in such a science based community is seen to have negative connotations, therefore many faults; parametric or catastrophic go unreported. With low volume commercial manufacturing of microfluidic systems, manufacturing defect statistics are not well-known, nor has the full spectrum of faults and failure modes been discovered. Furthermore, with few manufactured systems, few systems exist in the field, therefore, in-field failure statistics Mean-Time before Failure (MTBF) and time dependant faults are unknown. This lack of *a priori* statistics, which are commonly found in other test research areas, such as digital semiconductor manufacture and now MEMS, means that intuition, personal experience and conversations with those working in the community have to be relied upon to determine likely faults to begin development of the methodology.

A novel method of injecting faults within a FEM structure, the Fault Block, is presented and proves to be a generic method of fault injection, independent of fault type. This has the advantage of being able to describe fault behaviour at a low level of abstraction. Previous methods reported in the literature review have abstracted the entire system into a high-level behavioural model and then injected high-level faults into this already high-level model. Behavioural inaccuracies may result, and go unchallenged as cross-validation of high-level system fault models against experimentation is more difficult.

In this chapter a range of fault types are presented, modelled and cross-validated to determine the accuracy of the fault model. The fault model (fault behaviour) may be described in several ways; the movement of a parameter value out of a specified bounds, the injection of a fault block and parametric fault, or a change in function of a geometrical boundary condition or a combination of them all.

As with the simulation of the fault-free system the accuracy of the fault model is highly important to the success of the Simulation before Test, SbT approach, therefore careful cross-validation between simulation and experimentation is carried out for each proposed fault condition / model to determine the accuracy of the description to the experimental behaviour, its computational expense, and its suitability to the SbT method (injection).

Since system parameter measurements are required to be made in order to cross-validate, then it seems fitting to introduce the two proposed microfluidic test methods; impedance spectroscopy and Levich electro-chemical sensors, in this chapter.

4.2 Fault Modelling

The adopted modelling approach, as described in Chapter 3, is the FEM technique. This is not the most computational efficient, however, it is the most likely simulation method for the microfluidic research and system development community to adopt. Therefore the decision to develop a Microfluidic Fault Simulator, MFS based upon FEM, is that it aids in the development of the full system workflow, but this is not the exclusive reason. The FEM approach allows the implementation of microfluidic fault models at low-levels of abstraction, facilitating in-depth cross-validation work on a range of fault types. It also introduces a method of generic "static" fault injection and a vehicle by which to describe fault behaviour through application mode specific physics.

4.2.1 Evolutionary Fault Injection Discussion

The term static fault injection is used to denote that the fault condition is present and fixed from the beginning of the simulation; the evolution or development process of the fault condition is not modelled. For example, a path which describes an evolutionary fault injection process might be the application of an excessive electrode voltage in an EOF application, which results in electrolysis and the creation of hydrogen bubbles, a bubble could escape into the EOF channel and cause EOF transport to cease. In the static approach the bubble would be modelled at fixed dimensions, conditions and location from the start of the simulation. The fault block will be introduced in terms of static faults. The static fault approach will be used throughout this thesis.

4.2.2 Fault Block

The Fault Block, FB, is a novel and generic method of providing a "space" within the fault-free system in which to describe faulty behaviour. It is a geometric block which is injected at predetermined locations within the original fault free system geometry. The Fault Block may be used to describe a range of fault conditions and behaviour by describing the fault physics per application mode through the sub-domain and boundary conditions of the fault block, Figure 35 demonstrates the injection process. This method may be automated by controlling these conditions through the COMSOL scripting interface, driven by an external algorithm, Chapter 5.

Fault injection has either not been considered in such system simulations (either microfluidic or any FEM simulation) or has been ad-hoc with each fault uniquely implemented. Therefore, the strength of the fault block comes from being a common fault platform that can be customized and injected, allowing its use in fault injection automation.

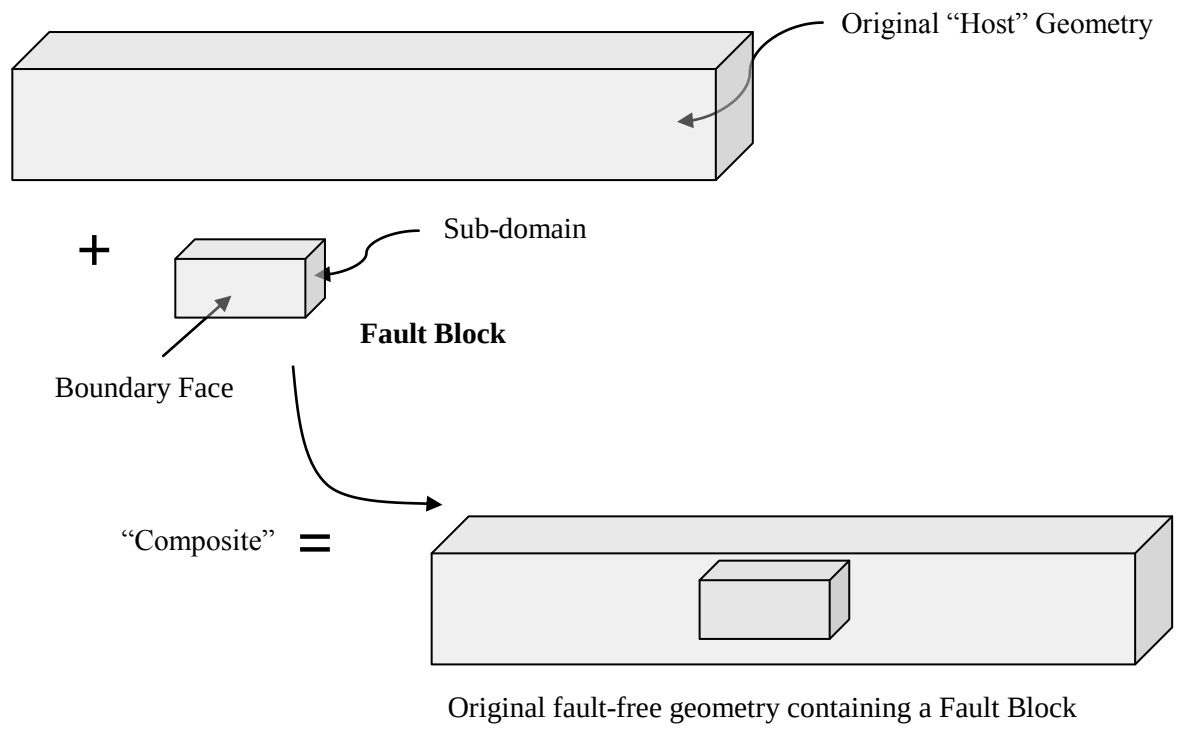


Figure 35 Fault Block Anatomy

4.2.3 “Static” Fault Injection Process

1. Determine the dimensions and location of the fault block.
2. Inject the fault block and create a composite geometry with the “Host” geometry.
3. Set the sub-domain and boundary conditions per application mode.

4.2.3.1 Fault Block Dimensions & Location

The dimensions (*x*, *y*, and *z*) of the fault block are used to describe the physical dimensions of the fault. For example, in the case of a blockage fault the x and z dimensions may be the same as the host channel, determining a complete blockage or some fraction of the host geometries dimensions for a partial blockage. The location of the fault block is used to describe where the fault is to occur in the “host” geometry; this may be pre-determined or governed by the *most likely* statistics.

4.2.3.2 Injection and Composite Formation

The injection of a fault block creates a composite object with the host geometry. A composite object may be formed in two ways; a union or subtraction, a simulation analysis of these two methods may be found in Figure 36. The subtraction of the fault block removes the block from simulation space, essentially it creates void, where no physics are described, limiting the usefulness of this method However, this is a very computational efficient method to describe a blockage in a hydrodynamic flow application. This method creates internal boundary walls which impede the flow;

however, due to the lack of physical equation descriptions errors occur when implementing detection methods. For this reason the union (a+b) is used for fault injection. This creates an additional sub-domain in the host geometry. COMSOL uses an associative technique to provide the additional sub-domain with the same physics settings as the sub-domain to which it is inserted, the boundary conditions are described as continuities. At this stage the model still remains fault free. It is an object within another object, both possessing the same physics. The physical sub-domain equations and boundary conditions have to be implemented to describe the desired fault behaviour.

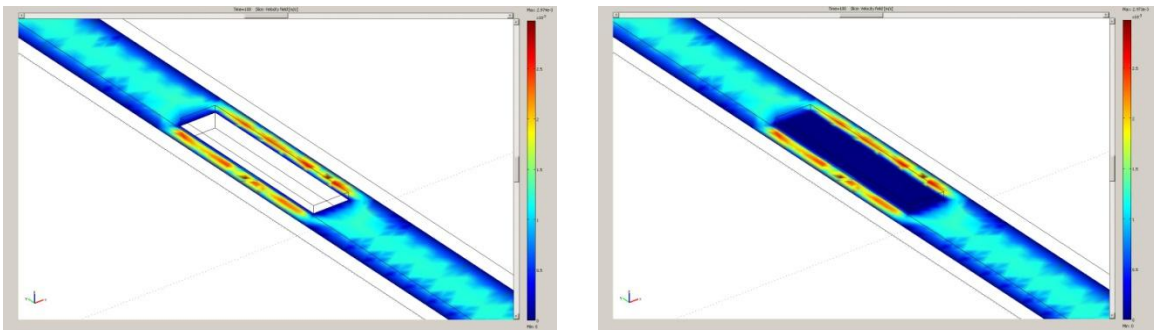


Figure 36 Composite Geometry with Fault Block (Left) Subtracted; (Right) Union

4.2.3.3 Setting Sub-domain & Boundary Conditions

COMSOL uses sub-domain and boundary condition auto-association with composite geometries for each application mode. Therefore, in order to describe the fault behaviour the sub-domain and boundary conditions require setting to those conditions described by the fault model.

4.3 Test Methods

In this thesis we investigate two methods of test; impedance spectroscopy and Levich electro-chemical Sensors. The reason for their introduction here, ahead of the fault behavioural models, is because the test methods are included in the faulty simulations as a means of facilitating cross-validation to the experimental faulted systems employing the same test methods.

4.3.1 Identifying Functional Sensors & Test Sensors

The most basic microfluidic functional sensors include pressure transducers, flow sensors, temperature measurement devices to name a few. An approach often adopted by the science community for using these as test sensors would be to set predetermined bounds (golden values) and calibration curves of operation. If the sensor values are within these bounds the system is functioning as expected, if these values deviate, there is “something wrong” with the system. This approach may not be sensitive to parametric faults, and provides poor discrimination between fault types. Optimum placement in the system may be in conflict between its normal functional role and its role as a test

sensor, potentially requiring an abundance of expensive (financial and real estate) sensors through-out the system.

Moreover, sensors which are sensitive to multiple parameters are sometimes avoided due to the lack of discrimination. In our test approach sensors which are sensitive to multiple parameters are adopted and discrimination patterns created using the Simulation before Test (SbT) approach.

Purposefully deployed test methods (test sensors) do not necessarily require additional "on-chip" hardware i.e. more electrodes, as chip designs already using electrodes in a functional role, for example EOF transport, then these electrodes may also be used for test electrodes implementing a technique such as impedance spectroscopy. Purposefully deployed test sensors and the use of functional sensors, becomes a case of semantics.

4.3.1.1 Impedance Spectroscopy

Impedance spectroscopy is a mature measurement technique [125] which has been used over many decades in a range of roles from electrical circuit measurement through to solid-liquid phase analysis. More recently it has been applied to microfluidic systems. It is to be investigated here as a possible test technique for microfluidic systems. Electrical Impedance Spectroscopy (EIS) has featured in many functional roles in microfluidics and some test roles. Ayliffe *et al.*[76] demonstrated that integrated gold electrodes could be fabricated within the microfluidic device for EIS measurements . Hywel *et al*. [126] uses micro-electrode pairs to measure the dielectric properties of single cells in a flow cytometer application, this method allows the analysis of some cell characteristics, but requires a multi-frequency band system to determine all the properties required. EIS has been used in particle detection and manipulation [127],[128] and for bio-behaviour studies and chemosensitivity of cancer cells [129].

In the test role impedance spectroscopy will be implemented using multiple electrodes, however, only one sink – source pair will be active at any given time. A frequency scanning approach will be adopted, in which an impedance magnitude and phase measurement will be acquired per frequency, at least in preliminary investigative work.

4.3.1.2 Electrochemical Sensor

The subject of electrochemistry is vast and out of the scope of this thesis. However, many microfluidic researchers have been investigating electrochemical phenomena using the integrated electrodes, as described in the impedance spectroscopy section 2.4.4. Much of this work is based on the classic Levich equation for the Rotating Disk Electrode (RDE), Equation 14 and is used to allow the observation of ultra-fast voltammetry reactions [130].

$$i_L = 0.62.n.F.[A]_{bulk}.D_A^{\frac{2}{3}}.\omega^{\frac{1}{3}}.v^{-\frac{1}{6}}.C_o$$

Equation 14 Classic Levich Equation

Where$n$ is the number of electrodes, F the Faraday constant, $[A]_{bulk}$ is the species "bulk" concentration, D_A the diffusion coefficient, ω is the angular velocity of rotation and v the kinematic viscosity and C_0 the initial concentration.

Collins *et al.*[131] introduces the concept of a flow transducer based around the modified Levich [132] mass transport limited current Equation 15. Under hydrodynamic conditions forced convection dominates the transport of ions. When the width of the microchannel is very small compared to its length, then lateral diffusion of ions is significant. The flow of ions produces a current, which is proportional to the chemical parameters described in Equation 15 and to the cube root of the velocity, Q of the ions.

$$i_L = 0.925.n.F.[A]_{bulk}.D_A^{\frac{2}{3}}.Q^{\frac{1}{3}}.w.\sqrt[3]{\frac{x_e^2}{h^2 d}}$$

Equation 15 Mass Transport Limited Current Equation

Where n is the number of electrodes, F the Faraday constant, x_e the electrode length, h the cell half-height, w the cell width. $[A]_{bulk}$ is the species "bulk" concentration, D_A the diffusion coefficient for the species and w the electrode width, the implementation of this non-rotary method may be found in Figure 78.

A numerical approach was taken by Compton *et al.*[130] to study the mass-transport limited current flowing at the micro-electrodes. Godino *et al.*[133] studied embedded microband electrodes through COMSOL simulation, investigating physical conditions which caused the current to deviate from the Levich equation. They used a range of 2D and 3D models to capture features like edge diffusion and the parabolic flow profile, and validated their results to published experimental work. They found that the main cause of deviation was due to axial diffusion; this is caused mainly by a "wide" channel, but also due to low flow rates. The deviation was found to be 5% from the theoretical Levich equation.

Ayliffe *et al.*[77] uses a combination of the EIS and Levich theory to achieve an impedance based flow sensor. They found the optimum EIS stimulation frequency to be 350 Hz and were able to detect flow rates as low as 2.4µl min^{-1}. When flow rates increase the counter balancing ions are replaced by bulk solution, lowering average ionic concentration and the ability to conduct current between the electrodes, resulting in an increased impedance.

4.4 Implementation of Test Methods in Simulation

4.4.1 Impedance Spectroscopy

In the literature impedance spectroscopy has been shown to be sensitive to electrode and fluidic parameters, having the advantage of low test hardware insertion, utilising discrete embedded electrodes or existing measurement electrodes. In this thesis we explore its sensitivity and use to detect parametric and catastrophic faults, in our experimental and simulation systems.

For our Simulation before Test approach it is imperative that impedance spectroscopy can be simulated in COMSOL. Therefore in this section we determine what the simulation requirements are to produce simulation models which accurately describe real experimental systems implementing impedance spectroscopy. This is demonstrated by cross-validating a simple microfluidic channel under both fault-free and with a fault (bubble) conditions, using impedance spectroscopy measurements, with its experimental equivalent. This cross-validation procedure highlights some experimental requirements for yielding high cross-validation accuracy. Once the most accurate simulation method and requirements have been determined more fault conditions are studied, resulting in a range of fault conditions from the fault library being mapped to their equivalent simulated models.

4.4.1.1 Preliminary Fault-Free Experimental

This section describes the initial fault-free cross-validation experimental apparatus, with an overview shown in Figure 37. This is based upon the simple “T” reactor described earlier in Chapter 3 for the hydrodynamic experiments (Figure 6), however, in this case we only study the two inlets and connecting channel, the simulation geometry provides a clear diagram of the section of reactor studied Figure 38.

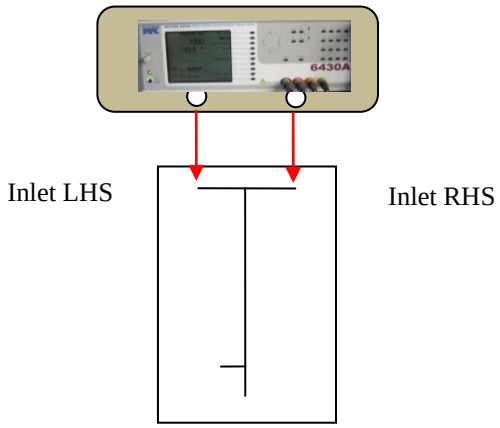


Figure 37 Fault-Free Experimental Apparatus

A Wayne Kerr 6403A Precision Component Analyzer was used for the experimental measurement of impedance. The probes were fixed to the reactor electrodes as shown in Figure 40, with the measurement signal having zero bias and 100mV amplitude. De-ionised water was loaded into the reactor via a syringe with an ambient temperature of 18˚C. The scan sweep is limited to 20Hz – 500 kHz

4.4.1.1.1 Preliminary Fault-Free Simulation

The simulation geometry reflects the experimental reactor section studied. In the simulation geometry we are able to describe only the section of reactor of interest, two inlets and the adjoining channel, in isolation from the rest of the reactor. Flow modeling is neglected since we are only interested in impedance measurements.

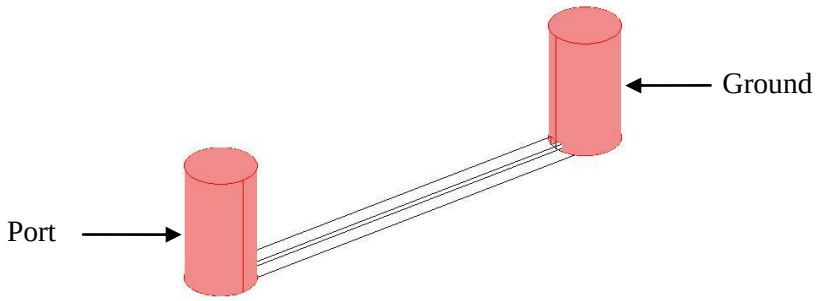


Figure 38 Simulation Reactor Geometry

The use of the electric currents application mode (emqvw) allows the computation of the impedance matrix, Z11 between a given port (source) and ground (sink). The frequency variable *nu_emqvw* is the frequency of the applied signal, which can be used with an incremental parametric solver to function as a frequency sweep. This allows the identification of frequency dependent features, should they exist. Furthermore, the application mode supports *small signal* analysis which allows biasing the signal around another application mode. One such useful example would be the Conductive Media DC (emdc) application mode used to perform Electro-osmotic Flow. This would allow simulation of impedance measurements taken during the operation of a system for continual fault monitoring. Sub-domain settings of conductivity σ =5.56 x10^{-6} S.m^{-1} with dielectric constant, ε_r = 80.

Boundary settings described one inlet as a port having a fixed current, with the second inlet at ground potential. The channel walls are described as electric insulators. The parametric scan sweep extended from 10Hz – 1MHz. COMSOL allows multiple solvers to be used in sequence for a given model. This allows a variety of solvers to be used, with each solver optimized for solving each application mode if necessary.

4.4.1.2 Results for Preliminary Experimental and Simulation

The results produced by the preliminary experimental and simulation work, using de-ionised water were very inaccurate and show that careful consideration in experimental set-up has to be taken when measuring impedance. Many external factors can affect the measurement, which may not be accounted for in the simulation, therefore perturbing the values used in cross-validation. Fluid conductivity, electrode placement and capacitive reactance are investigated.

4.4.1.2.1 Fluid Conductivity

The conductivity of the fluid used in the channels, was found to influence the accuracy between the experimental results and simulation results. De-ionized water with conductivity 5.56 $x10^{-6}$ $S.m^{-1}$ was initially considered.

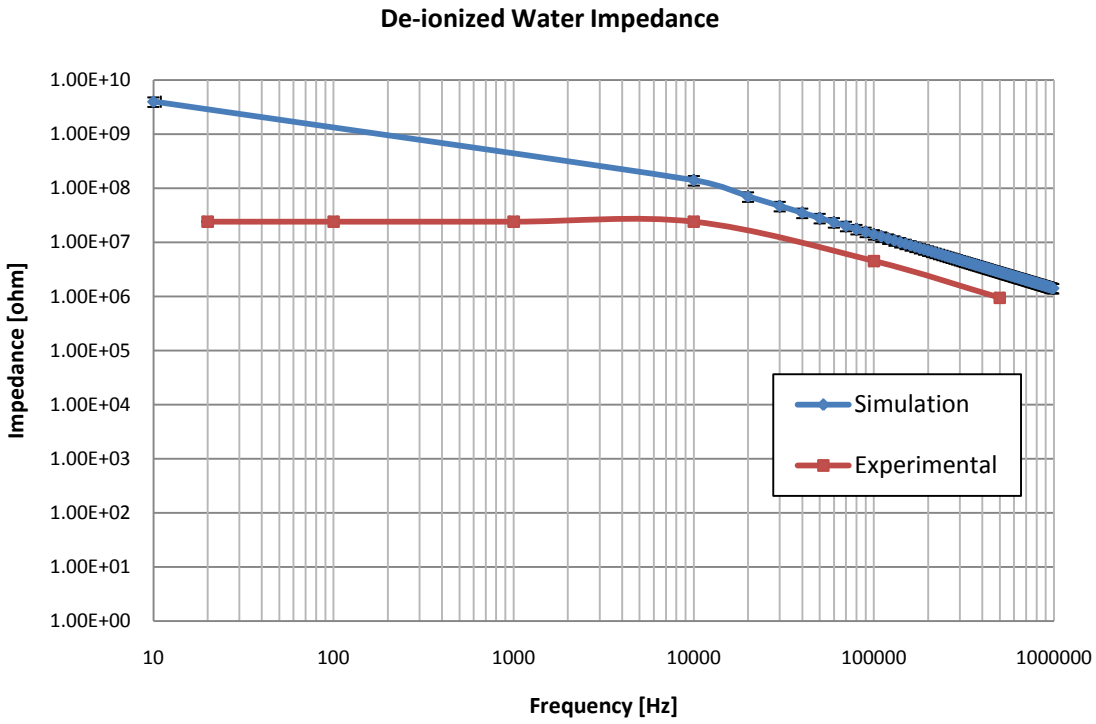


Figure 39 Impedance Magnitude Plot for De-ionized Water

Figure 39 shows two orders of magnitude difference between experimental and simulation results, with 20% error bars, when using de-ionised water as the fluid. The experimental result was lower than the simulation, indicating that a dominant experimental condition was not being sufficiently described in the simulation. Equivalent circuit analysis, section 4.4.1.2.3 of the geometry estimated a DC resistance of the geometry to be in the order of 4.5 $x10^{11}$ Ω, for the given geometry and fluid, approximating closer to COMSOL than experiment. Experimental measurements were consistently lower, suggesting that a dominant parameter was being measured, having lower impedance. De-ionized water by its nature (no ions) has a large impedance, therefore, other high

impedances in the experimental, could have a dominant effect on the measurements. Measurement electrode position and reactor capacitive reactance were also investigated.

4.4.1.2.2 Electrode Placement

Electrode placement was investigated in a "dry" reactor to determine the effect of placement. The extremes of electrode placement are shown in Figure 40.

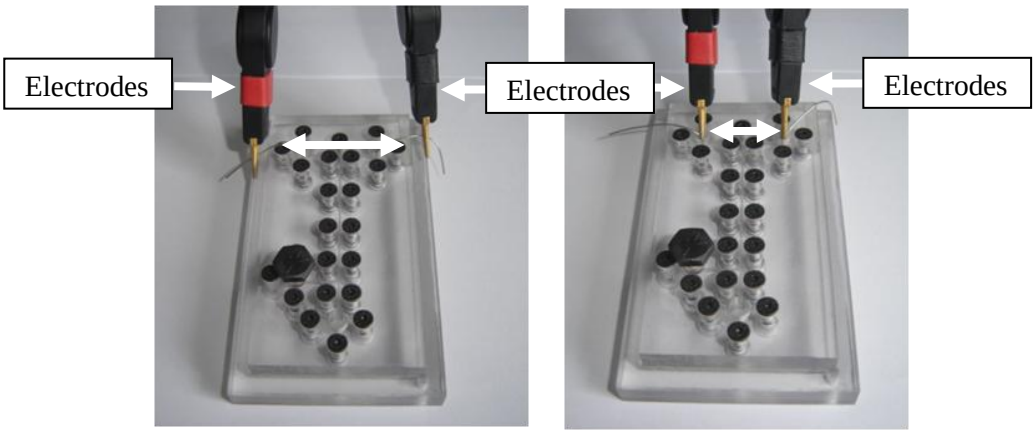


Figure 40 Two Electrode Placement Configurations showing the distance between electrodes

The impedance of the electrodes and measurement clips were measured to have the following impedance.

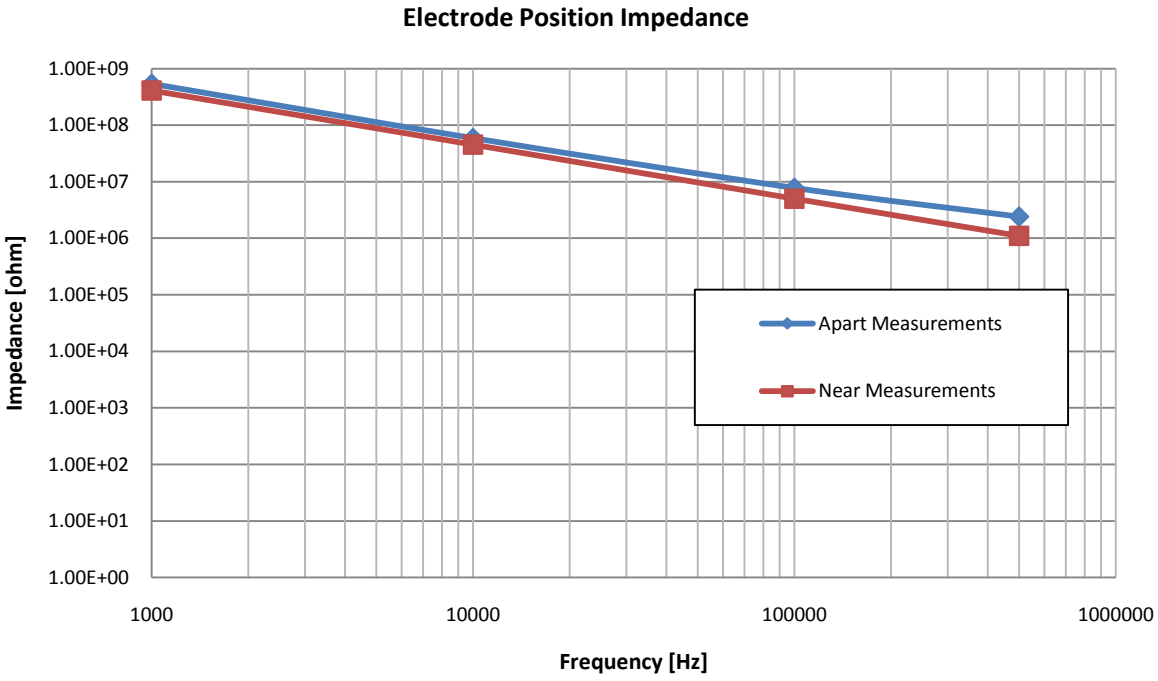


Figure 41 Impedance Magnitude plot for Electrode Placement

The measured impedance between "near" and "far" electrode placements, produce a similar impedance, in the order of magnitude, $1x10^9$ Ω. Therefore, there is little difference (24%) between the

two responses. The de-ionised water (Figure 39) and the dry (Figure 41) impedances at 1 kHz are within 15%, indicating electrode placement is not the source of error.

4.4.1.2.3 Capacitive Reactance

The capacitive reactance of the reactor acts in parallel with the fluidic impedance, therefore if this was dominate over the frequency range used, it would contribute to the lower impedance values experienced. The fluidic channel is shown with surrounding reactor material in Figure 42. The surrounding material was added in this case to include any possible dielectric affects.

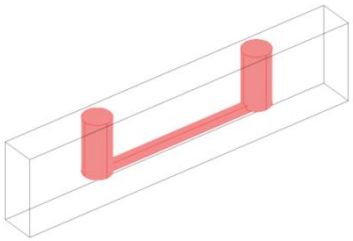


Figure 42 Test Reactor with surrounding material

The simulation was modified to include the quasi-electrostatic application layer (emes). The left inlet having the port input property "energy method". The right inlet was grounded and the connecting channel a continuity, while the surrounding reactor had zero charge symmetry. Using this technique the capacitance of the reactor could be measured. The capacitance between inlet and outlet was C = 1.2×10^{-13} F.

The experimental reactor underwent a frequency sweep with the channels dry, therefore the measured impedance was for the reactor geometry. The capacitive reactance Equation 16 was used to determine the capacitance of the geometry.

$$\boldsymbol{C = \frac{1}{(2\pi . f . R)}}$$

Equation 16 Capacitive Reactance (where f = 500 kHz and R = 1.1MΩ)

Therefore the capacitance of the experimental reactor was found to be C = 2.89×10^{-13} F. The measured and simulated results show a very similar capacitance (same order of magnitude) 58%. An equivalent circuit, Figure 43 may be used to determine if this capacitance should contribute to the discrepancies observed.

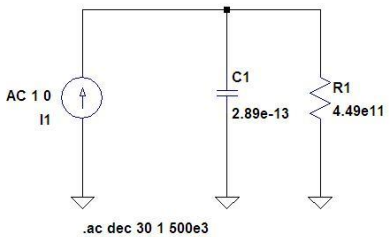


Figure 13 Equivalent Circuit

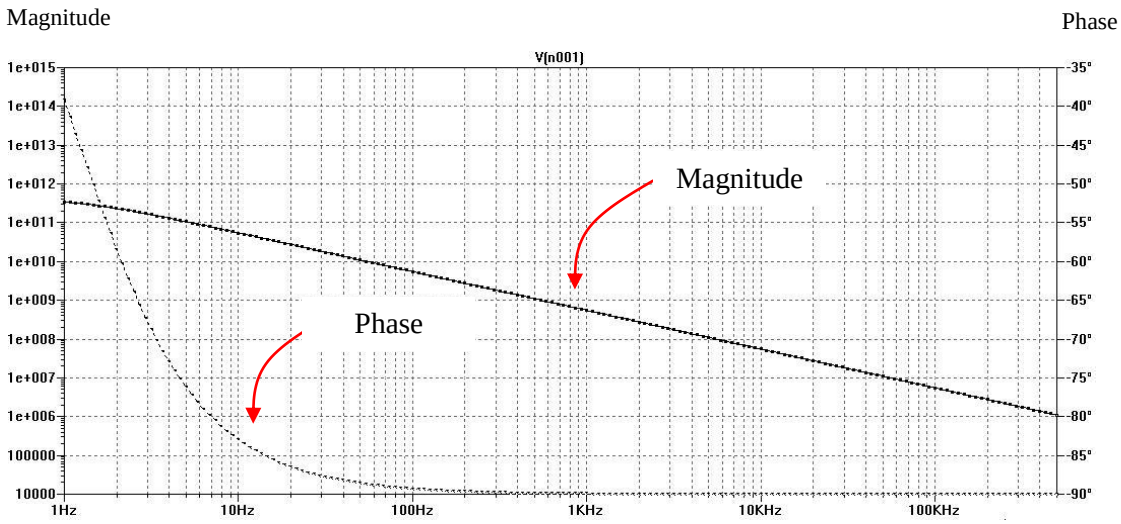


Figure 44 Equivalent Plot showing Capacitive Reactance

Figure 44 shows that the capacitive reactance, in parallel with the estimated DC resistance of the fluid channel, does not dominate the impedance response for this geometry, nor does it account for the discrepancy between experimental and simulation results.

4.5 Preliminary Fault-Free Experimental & Simulation Conclusion

Since large impedances are difficult to define and measure, and are naturally abundant in a system such as this, whether it be from electrode position or geometrical capacitive reactance. The use of de-ionized water with a low conductivity introduces further high impedance to the system, which may be of a similar order of magnitude to the surroundings, and therefore difficult to accurately measure. Furthermore, "ideal" de-ionized water should contain no ions resulting in a low electrical conductivity (5.56×10^{-6} $S.m^{-1}$), this being the case in the simulation, however, in experimentation the slightest contamination could largely affect this property, resulting in unexpected results.

The use of another fluid such as, KCl (Potassium Chloride) actively increases conductivity, increasing the conductivity of the fluid by 3 orders of magnitude, making the fluidic impedance more distinguishable from the surrounding impedances. The conductivity of KCl may be approximated using the ionic conductivity equation, Equation 17, for a given concentration.

$$S = \lambda_1 [X_1] + \lambda_2 [X_2] + \lambda_3 [X_3] + \lambda_4 [X_4] + \ldots$$

Equation 17 Equivalent Conductivity Equation

$K^+ = \lambda_1 = 7.35$ ($mS.m^2/mol$) at 25˚C and $Cl^- = \lambda_2 = 7.63$ ($mS.m^2/mol$) at 25˚C

Where λ is the molar conductivity and X the concentration. Therefore, for a concentration of 1mM the conductivity is stated as being 0.01498 $S.m^{-1}$

4.6 Continuation of Cross-Validation for Fault-Free Experiment & Simulation

The fault-free cross-validation process is continued using KCl as the measurement fluid. The experimental and simulation conditions remain as before, with the exception of the use of KCl, therefore in simulation the sub-domain conductivities are increased to 0.01498 $S.m^{-1}$.

4.6.1 Fault-Free Results

The cross-validation between experimentation and simulation, using 1mM KCl yields greater comparability, shown in Figure 45 and Figure 46. The difference in the two methods is approximately 15% on average for each data point. The experimental plot shows 9% error bars to indicate the maximum measurement deviation for each experimental run (10 experiments performed). Therefore the simulation error is around 6%, acceptable when many environmental factors affect the measured impedance as previously discussed.

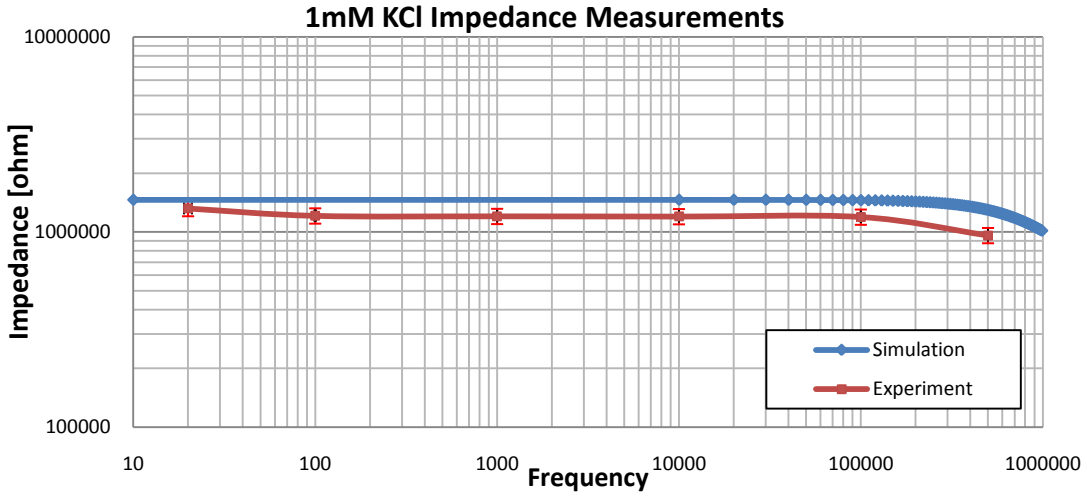


Figure 45 1mM Experimental & Simulation Impedance Plot

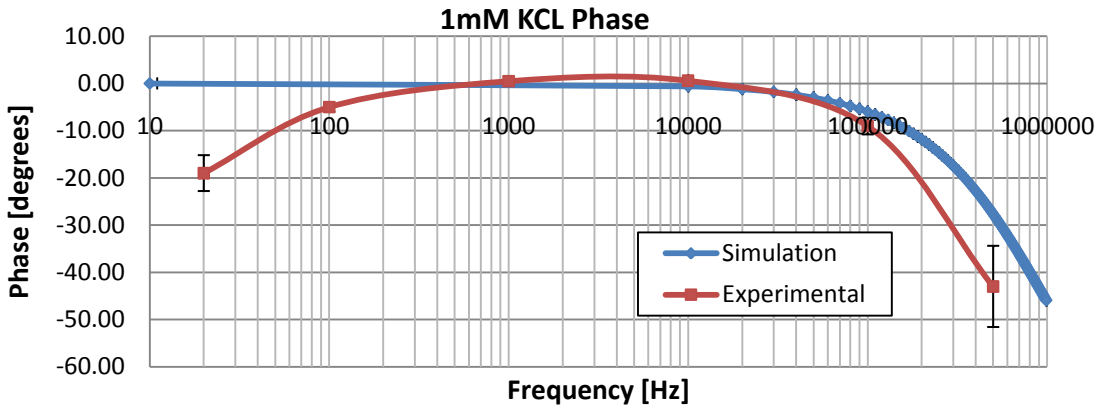


Figure 46 1mM Experimental & Simulation Phase Plot

4.7 Cross-Validation of Fault (Bubble) Experimental & Simulation Set-up

Before proceeding to investigate different fault types, a simple fault cross-validation experiment and simulation are carried out using a trapped bubble, to ensure that the Fault Block method is an accurate method to describe fault conditions.

4.7.1 Experimental

The same experimental set-up is maintained as in the fault-free experimental case; however, here a bubble is injected into the channel during the loading of the fluid. This is achieved by drawing a small volume of air into the syringe and expelling it along with the KCl into the reactor channel. The measurements probes were fixed to the reactor electrodes as shown previously in Figure 40, with a measurement signal having zero bias and 100mV amplitude.

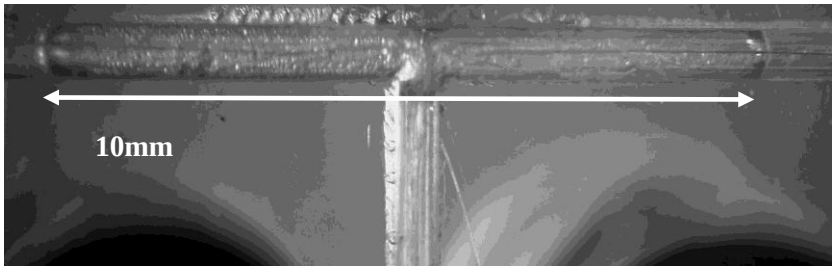


Figure 47 Experimental Trapped Bubble

The bubble measured 10mm and occupied the complete width and height of the channel.

4.7.2 Simulation

The simulation geometry used to describe the bubble is shown in Figure 48, having no surrounding reactor as previously discussed. The electric currents (emqvw) application mode was used along with a parametric solver to implement the scanning frequency.

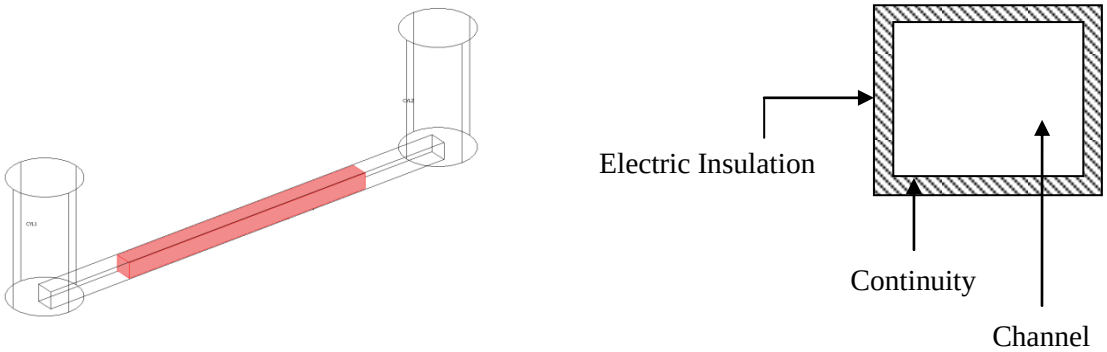


Figure 48 Bubble Fault Simulation Geometry

Figure 48 shows the simulation geometry containing a fault block, the cross-section highlights the limitation of this method. Since a bubble is a trapped gas, and gas expands so that the complete width and height of the channel is occupied by the bubble, therefore the fault block dimensions should do the same. The boundary condition of the fault block is forced to be described as a continuity and the boundary condition of the channel, electric insulation. Having two different boundary conditions with zero separation (extreme proximity) causes convergence errors because the residual error tends to become too large. Separation causes a gap (shown hatched in Figure 48 (Right) which in this example causes a fluidic path around the blockage. The fault block has dimensions (98µm x 92µm x 10mm). The sub-domain physics of the Fault Block were given to describe air; σ =0 $S.m^{-1}$ with dielectric constant, ε_r = 1. The remaining sub-domain settings were as before; σ =0.01498 $S.m^{-1}$ with dielectric constant, ε_r = 80, to describe 1mM KCL. The solution was solved with the parametric solver sweeping from 10Hz – 1MHz.

Figure 49 shows the impedance plot for the arrangement described. The primary feature to be observed is the difference between the simulation and experiment, which is 70% in the worst case. This represents a reasonable degree of accuracy considering the many influential factors that are present in such a system. The simulation shows little capacitive reactance, while the experimental measurements demonstrates a dramatic roll-off occurring >1 kHz. The reason for this is imposed by the "gap" between the Fault Block and the channel wall, due to the conflicting boundary conditions. In the simulating highly conductive 1mM KCl has a dominant resistive response created by the fluidic gap, over the capacitive response created by the fault block modelled as a bubble.

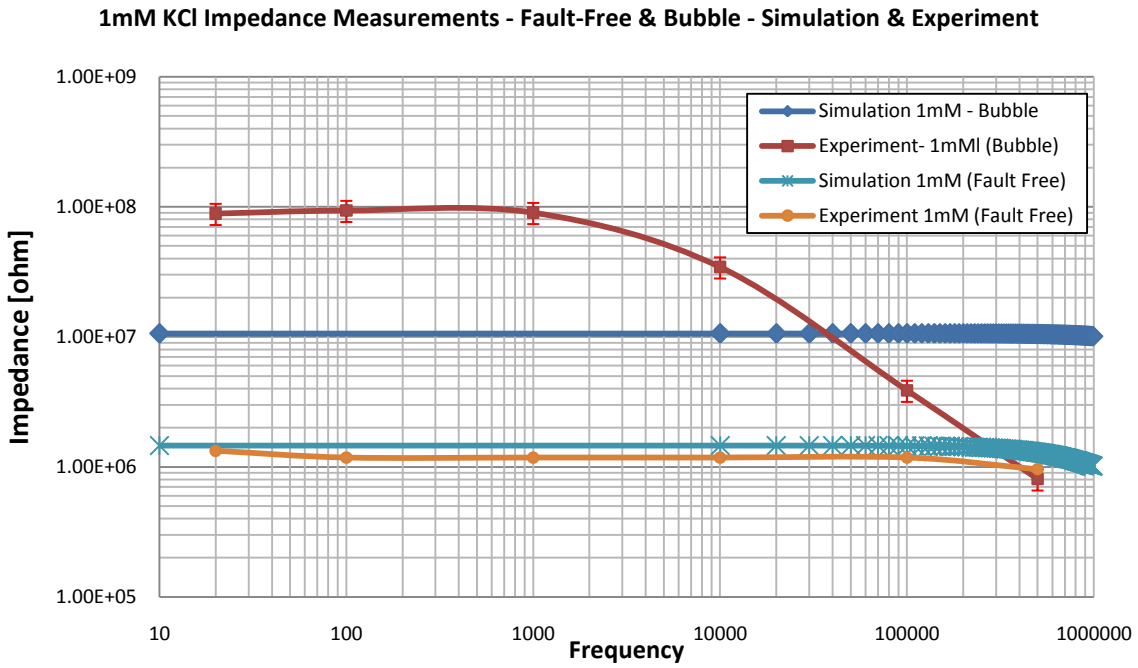


Figure 49 Bubble Cross-Validation Impedance Plot (no geometry)

4.7.2.1 Outer Geometry Inclusion

To overcome the limitation posed by two conflicting boundary types in close proximity, a third boundary is added, through the inclusion of the surrounding reactor material.

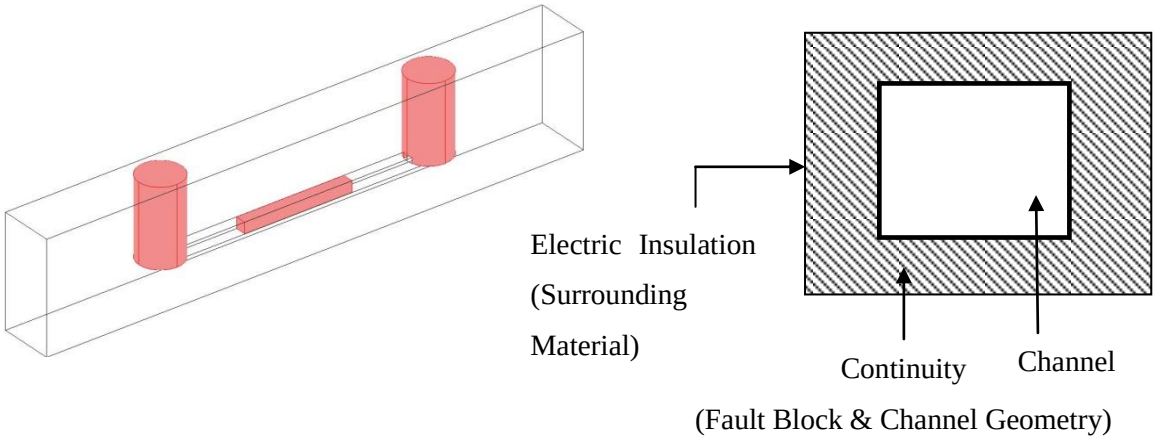


Figure 50 Inclusion of surrounding material

Figure 50 shows the cross-section of the boundary requirements. The surrounding material is provided in "excess" around the channel (shown hatched), its outer boundary described as "electric insulation", as the channel outer boundary was described in our previous example. The channel and fault block boundaries are to have the same height and width and both are described as "continuity" therefore, no conflicts exist. The sub-domain conditions completely describe the fluidic and bubble physics, which the simulation uses during solving. Figure 51 shows the improved impedance response; magnitudes are within 52%, and a capacitive roll-off is present.

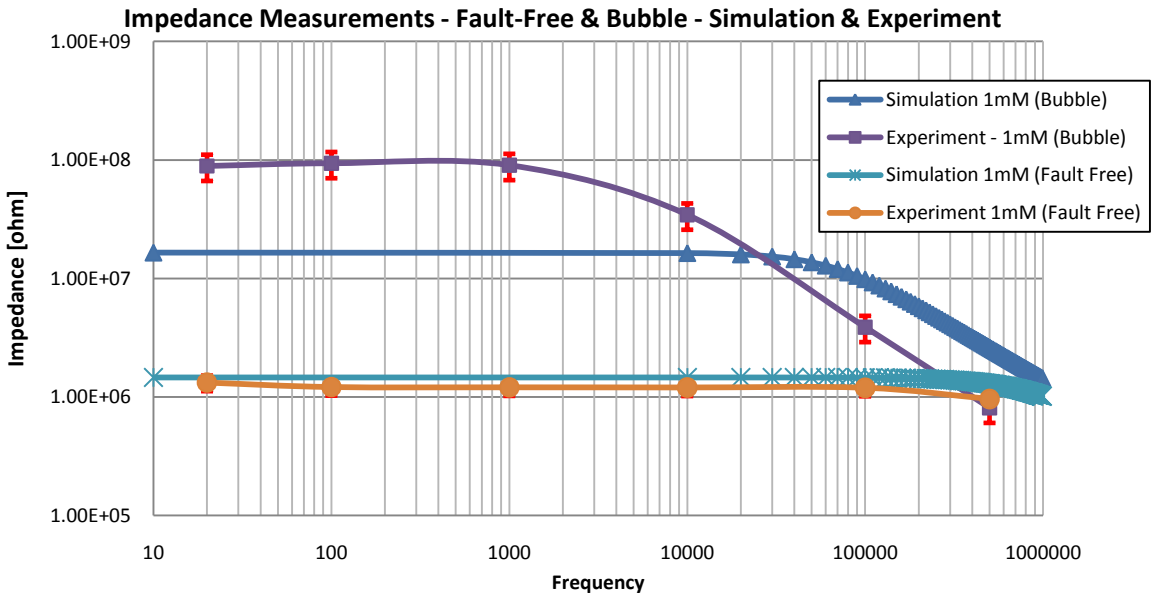


Figure 51 Impedance Spectroscopy Plot for inclusion of surrounding material

The inclusion of surrounding reactor material increases the computational overhead, by approximately 20% on average, as this material will be meshed, however, this seems the only viable solution to gain fault model accuracy using this approach. The amount of additional material which needs to be added has not been studied; this is the subject of further work. It has been observed that *any* additional material overcomes the problem of non-convergence.

4.8 Fault Experiments and Cross-Validation

In this section a range of fault models are investigated to determine whether the Fault Block or a combination of a fault block and parameter variance method can describe these conditions accurately when compared to their experimental counterparts. Some of the faults listed here are found in the Literature review, and some are investigated because they have been encountered through experience.

The faults investigated are:

Complete Blockage

Partial Blockage

Trapped Bubbles

Buffer Concentration Variation

Tubing Disconnection

Leakages

This list is by no means exhaustive of the faults which may occur in a microfluidic system. However, due to the low numbers of manufactured and deployed systems, few fault statistics are known. Furthermore, published work describes the science of microfluidics and fault reporting has negative connotations, therefore, faults are not commonly reported in the literature. In the scope of this thesis, these faults provide a means to demonstrate the method of how faults can be injected into a fault free simulation with low abstraction modelling and facilitate the further development of the workflow beyond this point.

This section focuses on using impedance spectroscopy as the test method; conventional sensors, such as those which measure flow and pressure, are used to aid validation. The Levich test method is not used in practical experiments; instead it is used in simulations later in this chapter and validated against published work.

4.8.1 Blockages

From our work on continuous flow microfluidic systems, blockages are one of the most frequently observed microfluidic faults. The term "observed" is used rather than "encountered", because it may be true that another fault type is more frequently encountered, but less observed, because a blockage tends to lead to a catastrophic failure, whereas other faults may cause an unnoticed parametric variation. Furthermore, blockages are not system specific and are common across all continuous flow systems types; hydrodynamic, electro-osmotic, electrophoretic etc.

4.8.1.1 Complete Blockage[1]

Complete blockages are the most catastrophic type of fault; they prevent fluid flow, disabling the function of a microfluidic system. Impedance spectroscopy is used as a test method to determine whether the blockage is detectable. In this section two approaches are used; an approach to investigate the affect on the hydrodynamic flow and an approach to investigate the affect on the electrical impedance. In order to optimise the simulation, two separate models are used, one using a geometry optimised to describe hydrodynamic flow (no large inlet ports) and one optimised for impedance measurements (detailed inlet ports with a truncated reactor geometry), Figure 52. This is not a limitation of the modelling approach, since either geometry could be extended to describe both cases, for example, the large inlet ports could be added to the complete reactor geometry. However, it is not computationally efficient, and not required for the validation we are performing here.

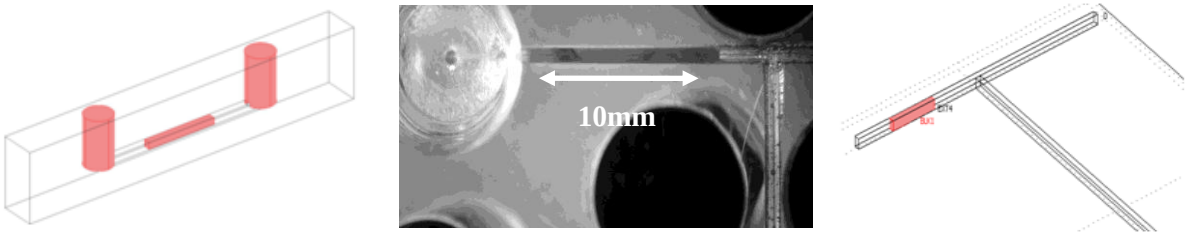


Figure 52 (Left) Model Geometry. (Centre) Experimental Blockage. (Right) Flow Geometry

A complete blockage is easily quantified, compared to a partial blockage. If the length of the blockage and position is known, which are relatively trivial to measure, the width and height are those of the host channel.

4.8.1.1.1 Complete Blockage Hydrodynamic Experiment

The experimental set-up (Figure 53), takes advantage of the "T" reactor described in the Hydrodynamic section section 3.5 of Chapter 3; recall that it may be separated in two halves to allow the placement of a blockage. The remaining experimental apparatus consists of a peristaltic pump connected to each inlet with flow rates of 5µL/Min. The fluid was 1mM KCl with a dye added for traceability. The blockage was 12mm in length and positioned 6mm from the inlet port, formed from

[1] Model File: Complete_Blockage.mph

a piece of PTFE plastic. A collection vessel at the outlet collected the fluid from the experiment for measurement. The experiment ran for 30 minutes in which time 150µL of fluid was collected.

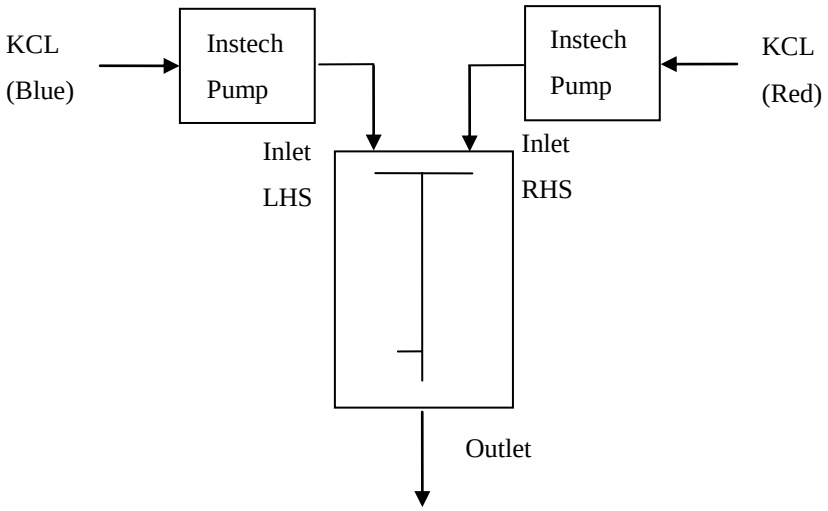


Figure 53 Blockage Experimental set-up

4.8.1.1.2 Complete Blockage Hydrodynamic Simulation

The experimental reactor geometry was imported into COMSOL from a CAD file. A Nevier-Stokes application mode described the hydrodynamic flow. The Fault Block geometry was added to create the inlet blockage, with the sub-domain set to describe PTFE and boundary conditions described as walls. The simulation time was set to run for 30 minutes. Boundary integration of the outlet for the velocity field yielded 9.628025e-11 [m^3/s] which is equivalent to 160.5 µL, a difference of 7% between experiment and simulation. This was a simple test to ensure initial validation of the experimental set-up and COMSOL simulation. The results showed close agreement between experiment and simulation. The differences may be attributed to the inaccuracies of the weighing approach to determine flow rate, not solely simulation inaccuracies.

4.8.1.1.3 Complete Blockage Impedance Experiment

The impedance experiment used 1mM KCl fluid. The Wayne Kerr 6430A was used as shown in Figure 54, with the settings previously described in the preliminary cross-validation study. The frequency scan ranged from 20Hz through to 500 kHz.

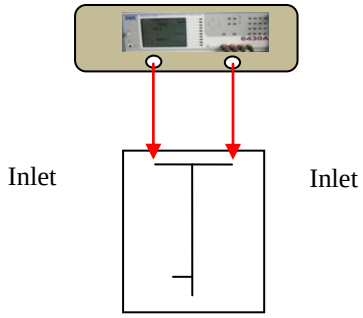


Figure 54 Complete Blockage Impedance Experimental Set-up

4.8.1.1.4 Complete Blockage Impedance Simulation

The simulation configuration followed the method as described in section 3.5. PTFE was described in the sub-domain settings as having conductivity σ =1 x10^{-12} S.m^{-1} with a dielectric constant, ε_r = 12.1. The remaining sub-domains were described as 1mM KCl.

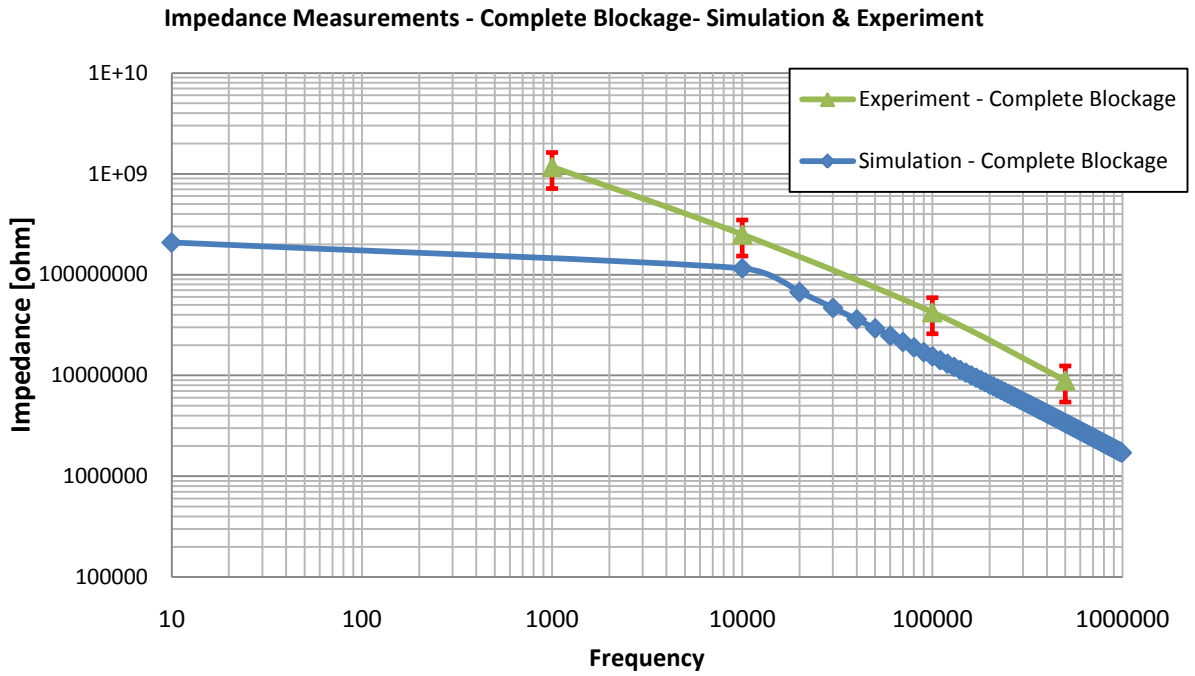


Figure 55 Bubble Simulation Impedance Plot (experimental results unstable below 1kHz)

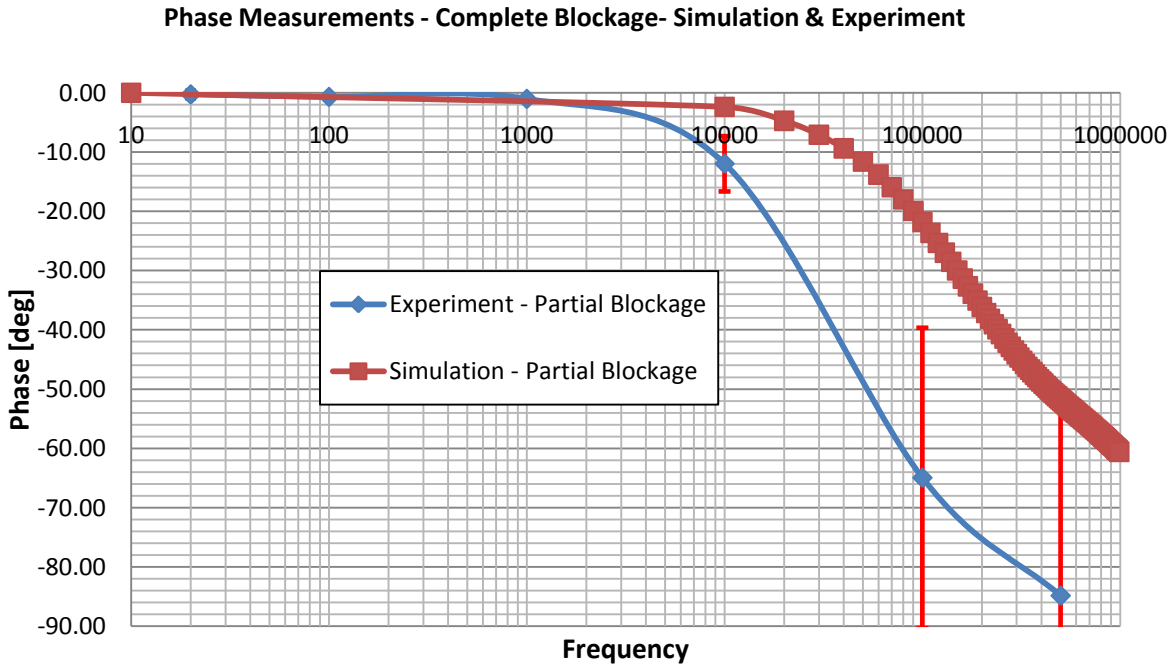


Figure 56 Bubble Simulation Phase Plot

The impedance magnitude response may be found in Figure 55 (11% error at 10 kHz), the phase plot Figure 56 (11% error at 10 kHz). These accurate impedance responses coupled with the high cross-validation of the hydrodynamic flow values (7%) demonstrates that the fault block accurately describes the behavior of a complete blockage fault.

4.8.2 Partial Blockage – Simple Block[2]

Partial blockages are probably one of the most likely blockage type, arising from the accidental or purposeful introduction of particulates, for example, due to a failed filter or through biological activity. Refer to the literature review chapter for more cases.

An experimental partial blockage may have a highly irregular, non-uniform shape making accuarate modeling difficult. The fault block approach is primarily investigated, together with more complex models, to determine if more detailed fault behavior can be implemented and provide any further information.

4.8.2.1 Partial Blockage Experimental Set-up

The same experimental set-up is used as for the complete blockage, except here we investigate an irregular shape blockage in the form of a "blob" of silicone epoxy inserted into the "T" junction of our experimental reactor. Its dimensions were approximated using bright field microscopy, width and height dimensions measured through the graticule and the overall depth (z-axis) using a focused point at the bottom of the channel and then re-focusing at the top of the blockage, Figure 57.

Figure 57 Blockage (Left) Blockage during experimentation. (Right) Close-up of blockage.

This provided a rough geometrical shape which could be translated into a Fault Block for assessment, approximately 0.5mm x 0.4mm x 0.6mm high.

4.8.2.2 Partial Blockage Simulation Set-up

The same simulation model is used as for the complete blockage, except that the Fault Block dimensions are changed to (500µm, 400µm, 600µm,) and it is positioned at the "T" junction. Figure 58 shows a single z slice plot of the velocity field, the grayscale shows dark 0 to light 1.9×10^{-3} m/s.

[2] Model File: T reactor centre blockage.mph

The plot is presented to show the flow around the *partial* blockage. The same experimental set-up was used as for the complete blockage and was run for 30 minutes, the outlet collection vessel collected 230µL.

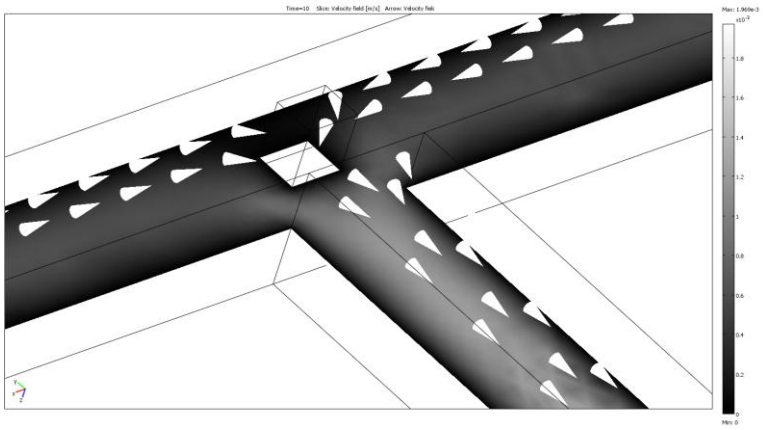


Figure 58 Velocity Field Post plot of the "T" Junction Blockage [m/s].

As before boundary integration was performed on the simulation outlet producing 1.9253e-10 [m^3/s], which equates to 264µL, a 15% difference between simulation and experiment, this discrepancy may be attribute to the measurement errors made in quantifying the partial blockage and measurement of the fluid, rather than reflecting upon the simulation model.

4.8.2.3 Impedance Measurements

The impedance aspect of the simulation remained the same from the hydrodynamic model, the difference between simulation and experiment for the magnitude being approximately 62%.

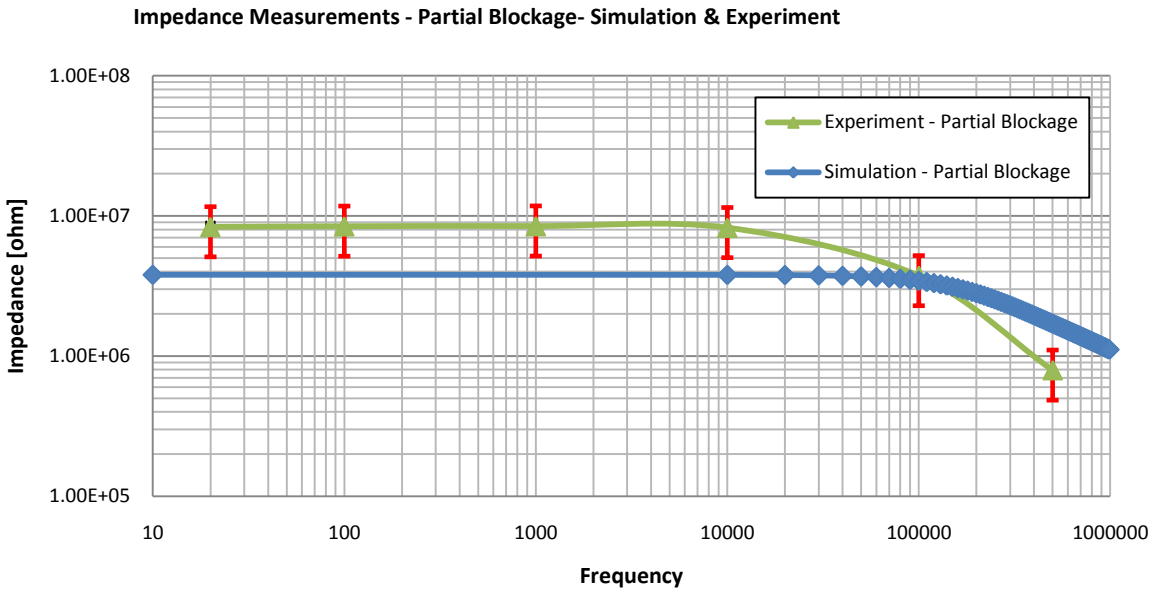


Figure 59 Partial Blockage - Experimentation & Simulation Impedance Plot

The simulation and experimental phase measurements are accurate < 5% up to 6 kHz, and the error becomes approximately 30% at 100 kHz. While this approximate method yielded accurate results, from flow 15% and impedance of 62% and phase <5% (around 6 kHz), a more accurate characterization of the partial blockage dimenstions should improve the simulations.

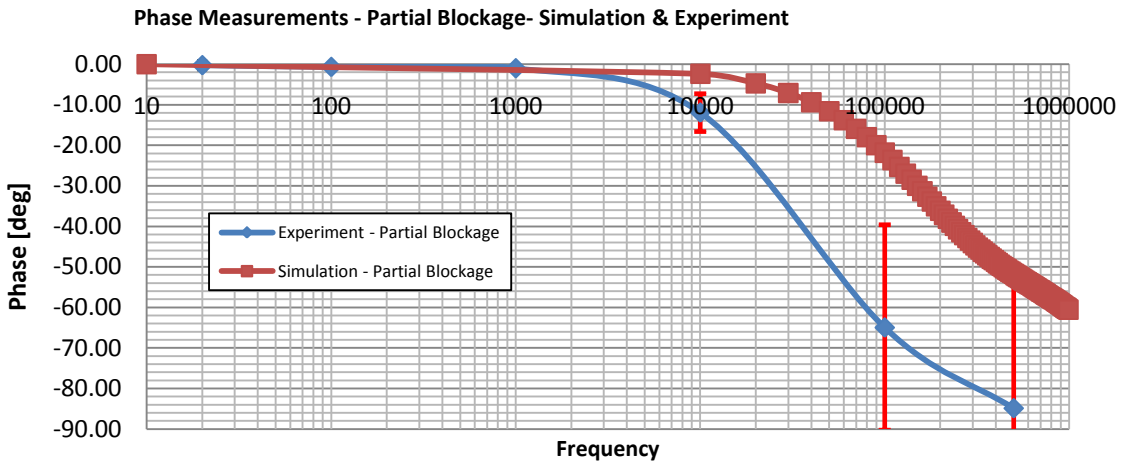


Figure 60 Partial Blockage - Experimentation & Simulation Phase Plot

To attempt to more accurately measure the experimental blockage a relative density X-ray (CAT scan) was used. The reactor was loaded into an X-Tek HMX 160 3D X-ray machine for a 4 hour scan. The results were analysed in ImageJ which allowed the partial blockage to be viewed in 3D, however, it was found that the densities between the blockage medium and the reactor medium were too similar to be able make any accurate measurements. An alternative blockage medium which provided high relative densities could have been investigated. However, due to the expsensive and lengthy CAT scans a new controllable method of partial blockage generation was required, and one which could be quantified, easily and rapidly implemented. This lead to work on the polymer monolith.

4.8.3 Partial Blockage – Polymerisation Monolith

Polymer monoliths may be produced with highly controlled interstitial volumes. Here a polymer monolith (Silica sol) is formed in a capillary; the porosity and therefore the interstitial volume of the monolith may be tightly controlled by varying the chemical formulae of the polymer used. Measuring the volume of the monolith (partial blockage) is important as this provides dimensional information for the implementation of the Fault Block in the simulation. The volume was attained by first measuring the weight of a monolith free capillary, and then measuring the dry and wet weight of the capillary containing the monolith. Advantages for using a capillary over complex reactor geometries in this case, include simplification of the model geometry for validation and the

ability to easily analyse the cross-section of the capillary to visually analyse the monolith structure, using a SEM (Scanning Electron Microscope) as shown in Figure 62.

4.8.3.1 Monolith Experimental Set-up

In this validation experiment, various flow rates of 50mM of Sodium Phosphate Buffer (Na_2HPO_4) pH 7 were used (monolith friendly), the pressure and impedance were measured. The experimental set-up is shown in Figure 61. 3 different volumes were used.

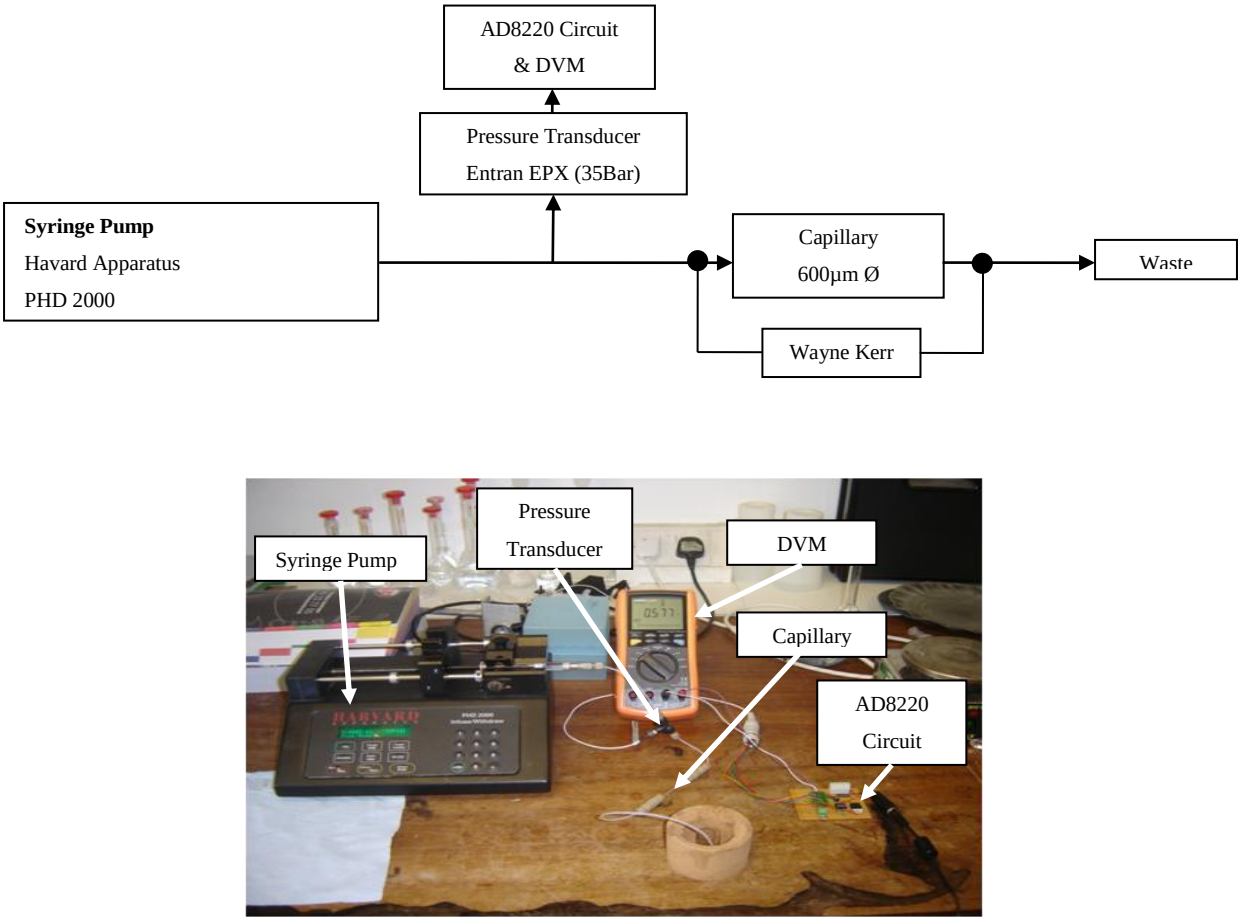


Figure 61 (Top) Schematic Overview of Experiment; (Bottom) Photograph

The Entran EPX is a stainless steel diaphragm pressure transducer with a dynamic range of 35 bar, producing a 125mV FSO (full scale output). Therefore a differential amplification circuit was constructed (schematic Appendix I) based around the Analog Devices AD8220 differential low drift amplifier. The output voltage was proportional to the applied pressure and read by a Digital Volt Meter (DVM). The pressure transducer and amplification circuit were calibrated against a range of known pressures and the corresponding voltages measured , resulting in a calibration curve used to derive the pressures in the experiment.

Generally, the higher the content of Silica Sol the smaller the pore size and therefore higher the back pressure. A monolith free capillary provided comparative results. The Scanning Electron

Microscope (SEM) images in Figure 62 show the porosity differences of the 3 capillaries. It should be noted that the large void shown to the left of each image is where the measurement electrodes were placed, which are explained in a later section. In Figure 62 the monlith porosity descreases from top to bottom of the page. Therefore Capillary III monolith volume is greater than Capillary I.

Table 2 shows the measured pressure and impedance for the given flow rates and interstitial volumes. Flow rates 100 µL/min, 200 µL/min and 400 µL/min were used with each volume and their corresponding pressure measured. For added detail the dry and wet weights were recorded, along with the raw (unprocessed) pressure voltages. Pressure was recorded as a cross-validation metric for the forthcoming fault models. A single impedance value was recorded for each flow rate as the measurement was found not to be dependent on the flow rate.

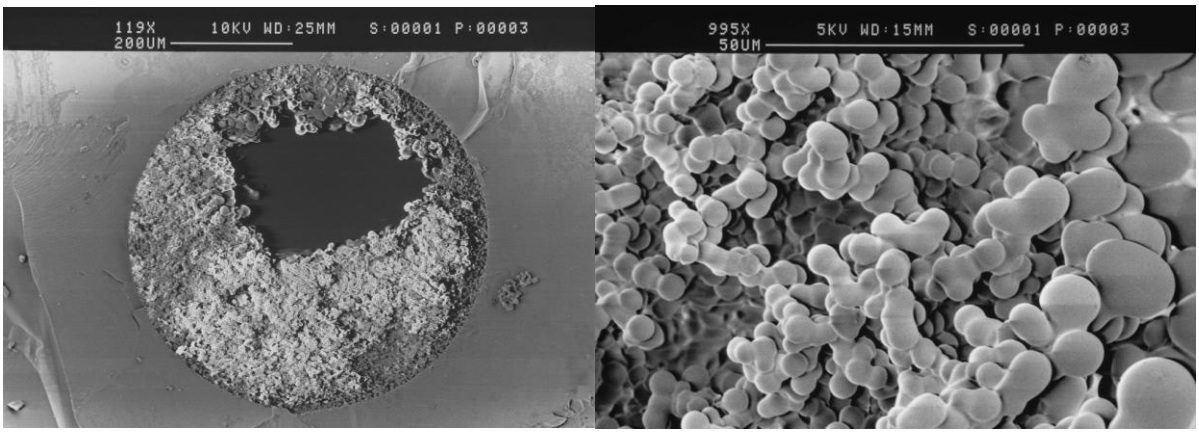
Results

Capillary	Dry Weight [g]	Wet Weight [g]	Interstitial Volume [m^3]	Monolith [%]	Flow Rate [µl/min]	Pressure Offset Voltage [Vdc]	Pressure Raw Voltage [Vdc]	Pressure [bar]	Impedance (1KHz)	Conductivity [µS]
No Monolith	-	-	$2.2 \times 10^{-8} m^3$	0	100	0.964	0.972	-	309 KΩ -0.5◦	3.3µS -4.5◦
Capillary I	0.3113	0.3327	$2.14 \times 10^{-8} m^3$	3%	100	0.853	0.950	0	446 KΩ -1.05◦	2.4 µS -8◦
					200	0.853	1.001	0		
					400	0.853	1.2	0.01		
Capillary II	0.3212	0.3395	$1.8 \times 10^{-8} m^3$	18%	100	0.805	1.01	0.03	475 KΩ -0.4◦	2.2µS -8◦
					200	0.800	1.050	0.05		
					400	0.815	1.270	0.17		
Capillary III	0.3341	0.3432	$9.1 \times 10^{-9} m^3$	58%	100	0.510	1.45	0.447	545 KΩ -0.7◦	2.0µS -8◦
					200	0.608	1.82	0.60		
					400	0.571	3.01	1.32		

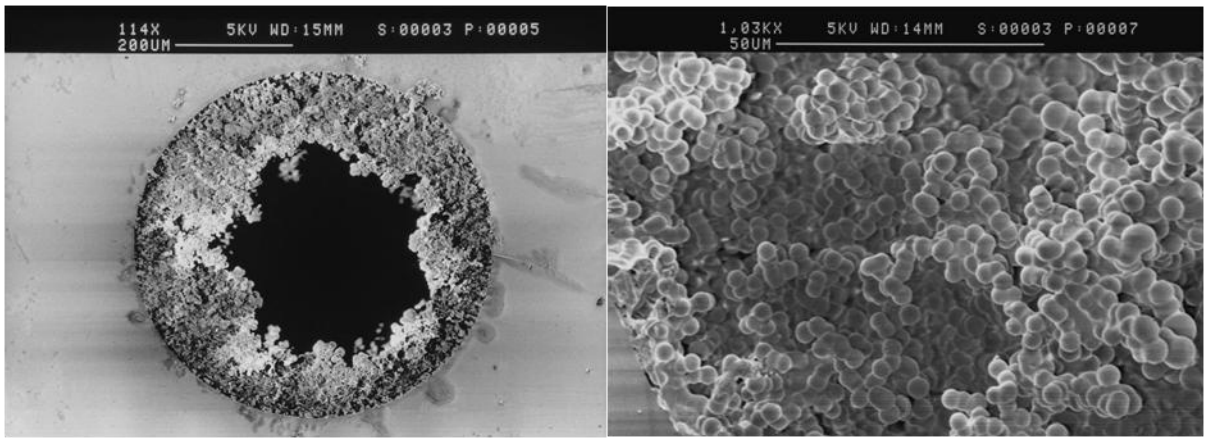
Table 2 Impedance and Pressure Results

NOTE: Impedance Spectroscopy was performed scanning the frequency range from 1kHz through to 500kHz; Figure 63 shows the relationship between monolith volume and impedance.

Capillary I



Capillary II



Capillary III

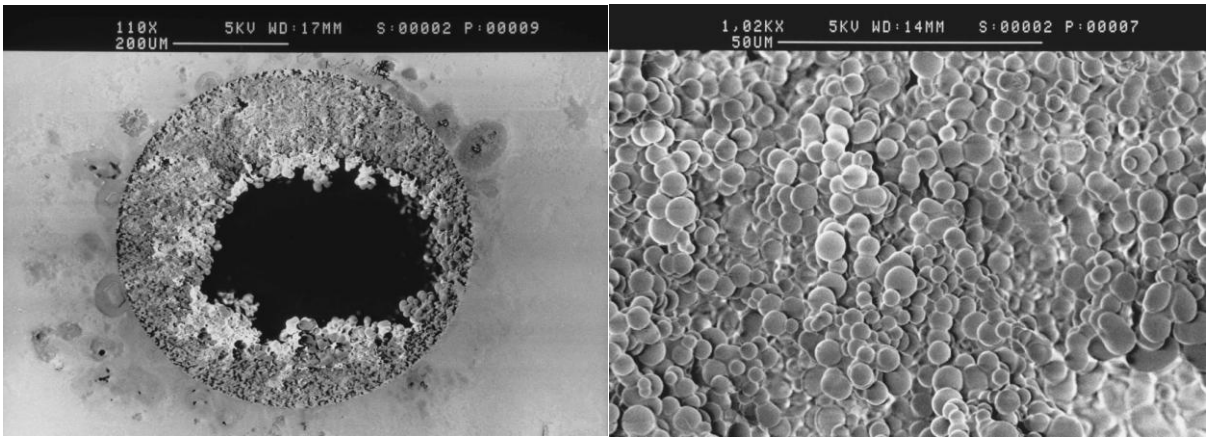


Figure 62 SEM of Capillary Cross-Section

Table 2 details many attributes of each capillary, but of particular interest here is the percentage of partial blockage; 3%, 18% and 58%, respectively.

4.8.3.2 Experimental Results

From the experimental results it was observed that the pressure is proportional to the flow rate, as the flow rate increases so does the pressure. Impedance magnitude is independent (at a fixed frequency) of the flow rate but increases with monolith density, Figure 61.

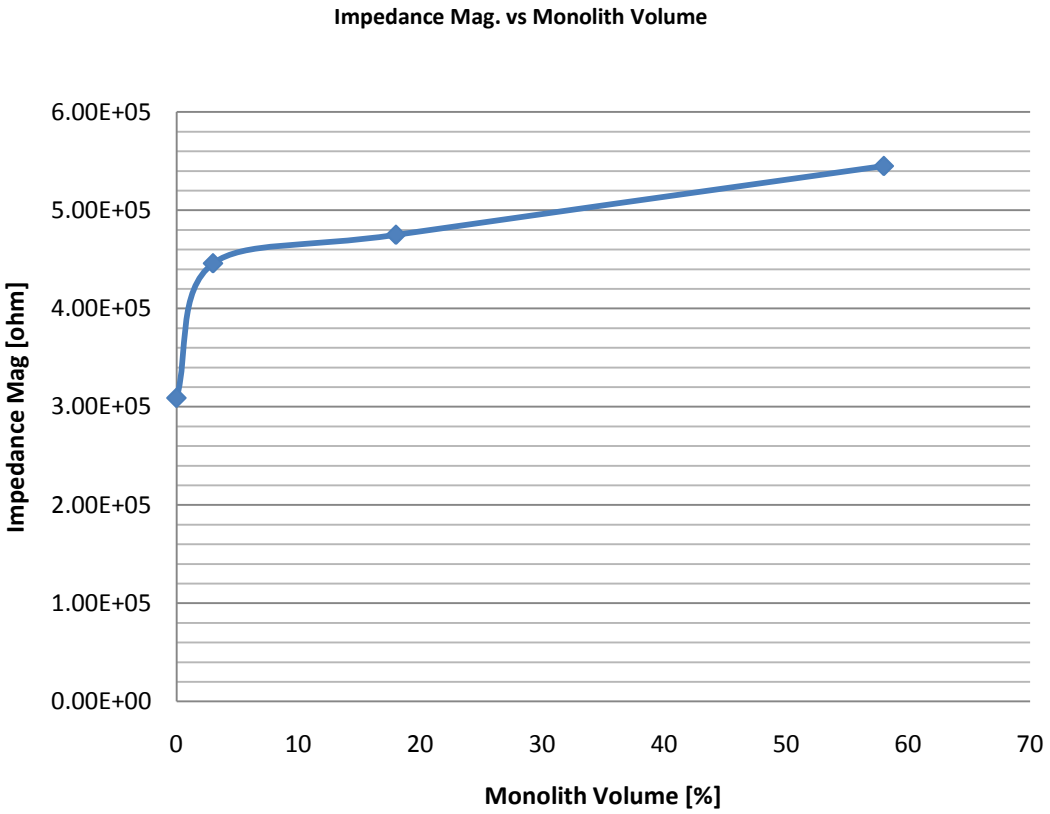


Figure 63 Impedance Magnitude vs Monolith Percentage

Therefore, impedance is shown to be sensitive to the monolith volume, which follows that impedance spectroscopy may be a suitable test method for partial blockages. Furthermore, from the impedance magnitude one may be able to estimate the size of the blockage. From these experiments we have obtained metrics for cross-validation (pressure and impedance), and have shown impedance only to be sensitive to blockage magnitude not flow rate.

4.8.3.3 Simulation of the Monolith Blockage

The complete system is shown diagrammatically in Figure 64. The FEM technique does not favour systems having different aspect ratios, as mentioned earlier. Therefore, the capillary alone is studied in this simulation section, neglecting of the connecting tubing from the simulation will affect the resulting pressure value, resulting in differences from the experimental value.

The simulation model used a single application mode; Incompressible Navier-Stokes. The modelling of the fault-free capillary was deemed trivial and validation of incompressible fluid flow models has been determined elsewhere. The capillary length was 80mm and 600 µm diameter (internal).

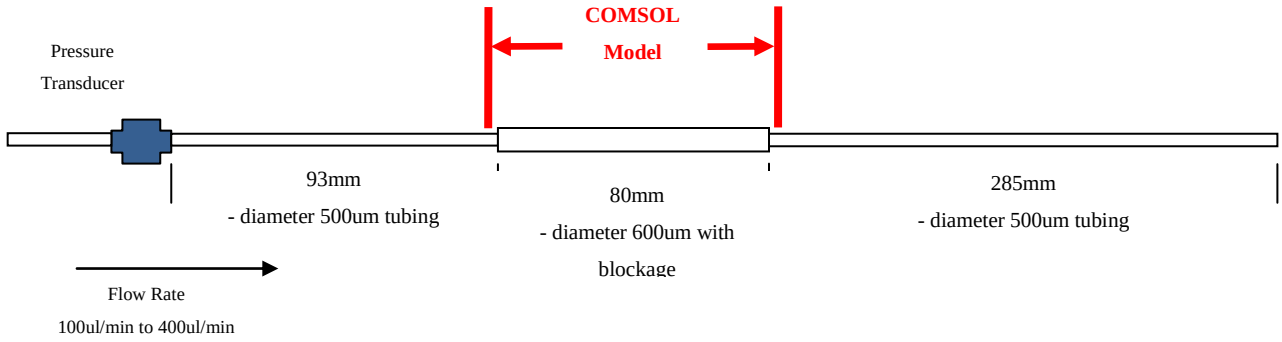


Figure 64 Conceptual Diagram of Capillary system

Two fault models are investigated in this section to describe the behavior of the monolith, they are the *complex monolith model* and the *cross-sectional model.*

4.8.4 Complex Monolith Model[3]

From work on the simple rectangular model and study of the monolith SEM plots it was decided that continued and repeated flow obstacles were required, shown in Figure 65.

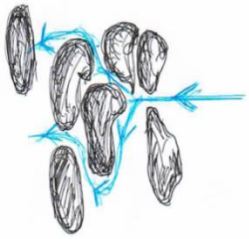


Figure 65 Diagram demonstrating repeated flow blockages

[3] Model File: 59PercentComplexModel

This presented the problem of how to create a complex monolith while maintaining the correct volumetric percentage between monolith volume and interstitial space.

The model was created in a 2D work-plane, Figure 66 (Top). This had the advantage of being able to tightly control the interstitial volume of the monolith by applying the rectangular blocks as percentages to one another per section. The second advantage was that this 2D geometry could be revolved into 3D. The revolving process not only revolves the geometry but also the mesh (a revolved mapped mesh), therefore a very computationally efficient "mapped" (blocks as opposed to tetrahedral) mesh can be applied in 2D and then revolved. Mapped meshing also alleviates problems associated with high aspect ratios in FEM models. For example, consider Figure 64 the entire length of the model is 458mm while the diameter of the model is 600µm maximum, a ratio of 1:763.

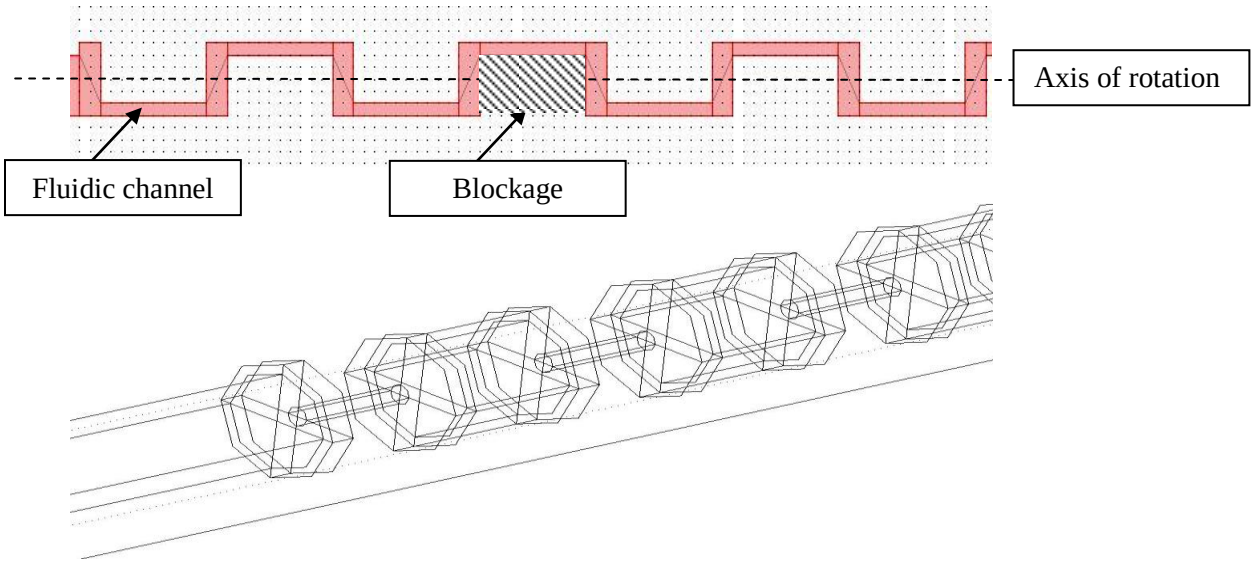


Figure 66 (Top) Complex Geometry Design 2D; (Bottom) 3D geometry after Revolution Process

Figure 66 is an example of the 59% monolith volume. Once revolved the "Geometric Properties" feature in COMSOL can be applied to determine the interstitial volume which equated to 1.3862 x $10^{-10}m^3$, the total capillary volume was $3.67x10^{-10}m^3$, therefore the percentage was 62.2% as opposed to 59% measured in the experimental set-up, an error of 5.1%.

Table 3 presents critical FEM parameters, the simulation solve time entry shows that this particular implementation could not be solved due to a "Memory Error", this could be due to the abundance of low quality elements, leading to memory errors in the solver.

Parameter	Value
Simulation Solve Time	Memory Error
Degress of Freedom	44751
No. of Elements	1776
Element Quality	0.0047
Element Volume Ratio	0.0022
Solver Type	Transient
Solver	SPOOLES
Simulation Step Time	0:0.1:1 seconds

Table 3 Complex Monolith Model Statistics

The quality measure is related to the models aspect ratio, meaning that the anisotropic elements can have a low quality even though they may have a reasonable shape. Quality is measured on a scale of 0 to 1, where 1 is perfect quality. COMSOL recommend [user guide] that the mesh element quality is greater than 0.005 to avoid a singularity in the matrix and the better the quality of the element the better the quality of the solution.

4.8.5 Cross-Sectional Model[4]

This model is simpler than the previously introduced complex model, and is based on multiple fault block (cylinders) presenting multiple flow impeding paths, but with improved element quality.

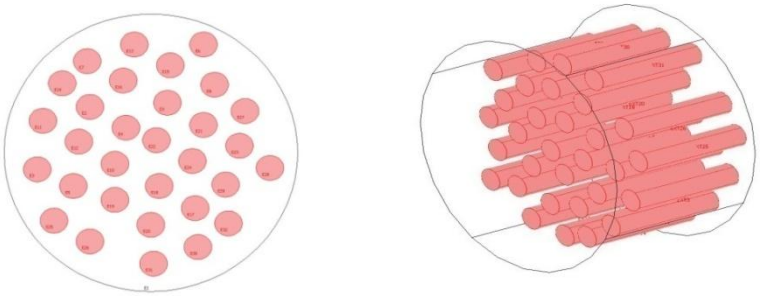


Figure 2(Left) Cross-Section showing multiple cylinders; (Right) 3D Extrusion Model

Table 4 shows that the cross-sectional cylinder model improves upon mesh element quality, when compared to the *complex monolith model*, at the expense of increased degree of freedom and element count.

[4] Model File: Cylinder_Model.mph

Parameter	Value
Simulation Solve Time	Memory Error
Degrees of Freedom	176653
No. of Elements	40679
Element Quality	0.0908
Element Volume Ratio	6.29e-5
Solver Type	Transient
Solver	SPOOLES
Simulation Step Time	0:0.1:1 seconds

Table 4 Cross-Sectional Cylinder Model Simulation Parameters

Therefore, a simulation memory error occurs not due to the element quality as in the *complex monolith model*, but due to the number of elements. This model's geometry and cylinder implementation could be manually optimised to reach a solution without a memory overrun, but we are interested in a generic automated injection approach.

These two models have failed to provide any cross-validation to the measured pressure and impedance.

4.9 Blockage Modelling Discussion

In this section a range of blockage mechanisms have been introduced experimentally. The complete blockage has been shown to be the simplest and most accurate to model. Partial blockage dimensional quantification inhibits the accuracy of mapping the experimental fault into simulation model, even in the simplest case. Moreover, the complex irregular nature of recurring flow obstacles, produce low mesh element quality, which results in a non-converging simulation.

Simulation and experimental findings have shown impedance spectroscopy to be sensitive to blockage detection. The polymerisation experiments have shown that partial blockages and flow rate directly affect channel pressure, while impedance magnitude is proportional to blockage dimensional magnitude, but independent of flow rate given the electrode arrangement, Table 2.

Fault blockage modelling has highlighted the fact there may be a "standard" fault model which sufficiently replicates an experimental fault condition, given the variability and trade-off parameters; mesh quality, convergence and cross-validation accuracy, but it just has not been found. It should be recalled that the purpose of the fault model is to provide a low abstraction means of obtaining a sufficient

representation of a fault condition in an otherwise fault-free system model. With this we should be able to analyse suitability and sensitivity for a given test method. Furthermore, given the automated nature of fault injection it is important to use a generic method which through simple parameter customisation can be manipulated to describe a range of approximate fault conditions.

The evolutionary approach to fault modeling could use the concept of a fault block. One possible method is to evolve the fault block physics and dimensions as the fault evolves. The fault mechanism would be described by a set of global expressions.

The scope of this thesis is to provide a methodology which supports fault injection using low abstraction models to provide a simulation before test approach which allows test analysis to be performed. As such it is felt that the fault block approach provides sufficient accuracy in describing complete and partial blockages to meet the requirements of this goal. Further work could investigate the modification of the fault block geometry and physics to yield higher cross-validation accuracy, should this be deemed necessary depending on the test methods deployed.

4.10 Leakages[5]

The problem of fluidic leakages are not explicitly discussed in the literature, however, it is intuitive that leakages could occur and are a common occurance in experimental work. Leakages may occur as manufacturing defects, where two substrates are insufficiently bonded or adjacent channels have a break through between them. They may occur from world-to-chip interconnects, where tubing is connected to the microfluidic reactor, if a poor connection is made or an interconnect becomes faulty then this could lead to a leakage.

4.10.1 Leakage Experiment

In this section we consider a leakage caused by a failed (absent) un-used channel plug. The original "T" reactor described in the hydrodynamic experimental section of Chapter 3 has an un-used channel near to the outlet and perpendicular to the main channel. This has a plug to stop fluid escaping the outlet. In this experimental case this plug is removed, Figure 68.

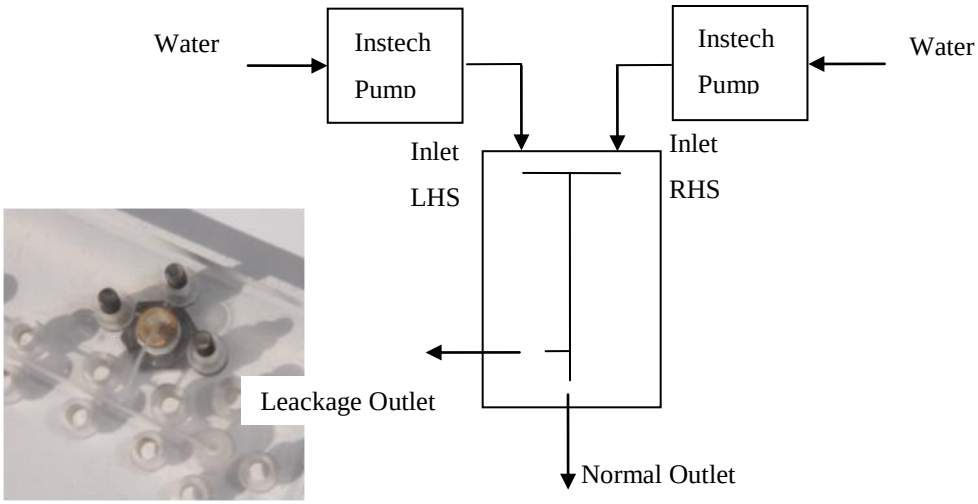


Figure 68 Leakage Experimental Set-up (Inset) Experimental Reactor

A diagram of the complete experimental set-up is shown in Figure 68. The flow rate of the inlets was 8.3µL/min and the experiment was left to run for a 30 minute period. The fluid was collected at the normal and leakage outlet, the collected volumes were 280 µL and 225 µL, respectively. The total of 505µL is as expected given that a similar volume of 538 µL was measured in the fault free case (section 3.5).

[5] Model File: Milled Reactor Leakage.mph

4.10.2 Leakage Simulation

Leakage fault models may be implemented in one of two ways. An existing boundary condition may be modified depending upon the application mode used; for example, in a hydrodynamic mode a "wall" boundary may be modified to an "open" condition. Alternatively, a fault block may be inserted and a composite geometry formed. COMSOL auto-associates the sub-domain and boundary conditions across all application modes, therefore, the fault block is transparent.

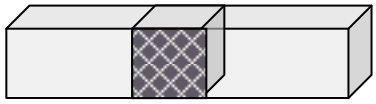


Figure 69 Fault Block insertion for leakage boundary condition

In the example in Figure 69 one of the boundary faces of the fault block (shown hatched) has been modified to provide an "open" condition to describe a leak, in the middle of a wall boundary condition in the host geometry.

In the simulation described here, the hydrodynamic "T" reactor flow model is used, with the perpendicular channel end boundary changed from "wall" to "outlet" to simulate the leakage. The same 8.3µL/min flow rates were implemented as in the experiment and the simulation period was set-up for 30 minutes.

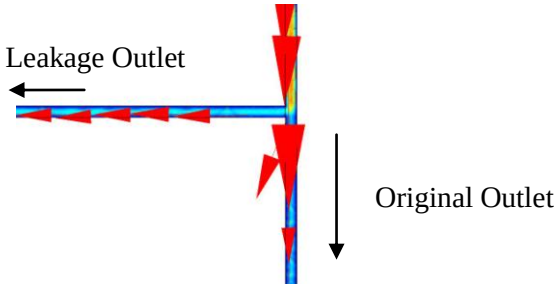


Figure 70 Post Plot showing proportional flow arrows entering the open boundary and the outlet.

Figure 70 shows how the flow is split between the existing outlet and the faulty open boundary. compared to 264 µL and 213 µL in the simulation. Producing overall differences of 5.7% and 5.3%, respectively.

4.10.3 Leakage Conclusion

This model cross-validates that this method of leakage fault model is highly descriptive of the experimental leak, and an acceptable method of leakage modelling. We have not presented a method of detecting this fault, which will be described in section 4.14 using Levich electrochemical sensors.

4.11 Channel Disconnect Fault

Microfluidic systems require a world-to-chip interconnect. The failure of these connections would generally cause catastrophic system operation, resulting from lack of sample or analyte supply. World-to-chip interconnect failure has been studied [44]. Fault modelling and fault detection are investigated here.

A cross-sectional schematic of a channel disconnection is shown in Figure 71 (top). The intuitive description would be to have a zero flow condition entering the reactor inlet. If one thinks ahead to the type of test method that could be applied to detect this condition, assuming that functional sensors are to be neglected due to their cost, then this may be treated as an electrical problem and investigated using impedance spectroscopy Figure 71 (bottom). If it is considered that the fluid in the channel (either side of the disconnection) will have a resistance; the disconnection (gap) may be described as a capacitance.

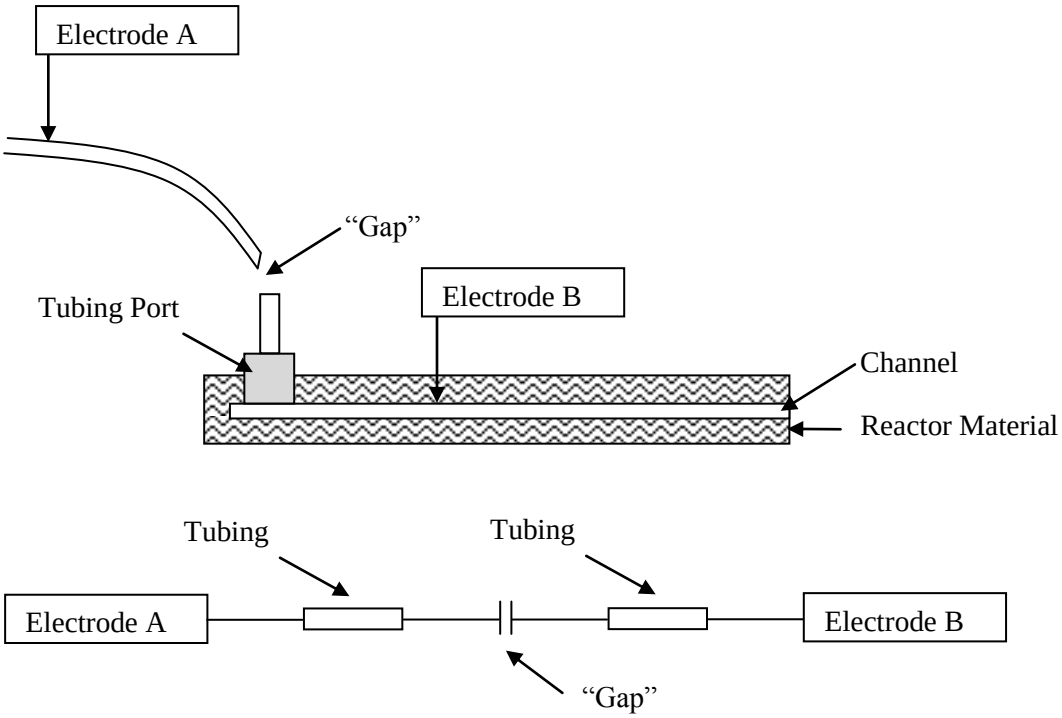


Figure 71 Diagram of Channel Disconnection Fault

4.11.1 Disconnection Experimental

The hydrodynamic "T" reactor is used as before, however, in this section a length of PEEK tubing (300mm) is attached to the left hand side inlet of the reactor. One impedance electrode is attached to the free end of the tubing and the other electrode placed in the right hand side reactor inlet. There is no flow used in this experiment, only enough to fill the channels and tubing. The 1mM KCl fluid is loaded into the reactor and tubing. The Wayne Kerr 6430A was used as shown in Figure 72, with the settings previously described. The frequency scan ranged from 20 Hz through to 500 kHz

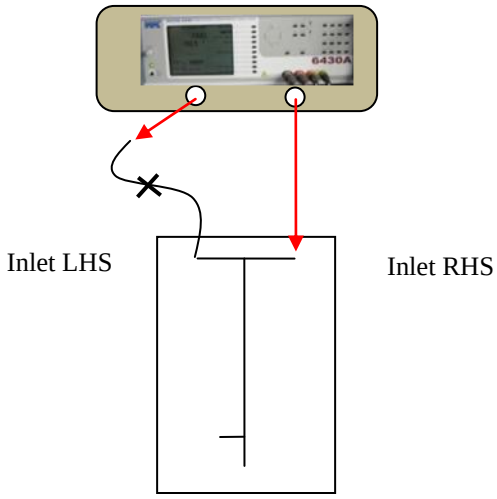


Figure 72 Experimental Disconnection Diagram

The impedance of the fault-free implementation of this experimental arrangement has already been determined in section 4.4. The fault condition (disconnection) was implemented by unscrewing the inlet port (Up-Church ¼" UNF) connection from the reactor left hand side inlet. Figure 74 and Figure 75 present the impedance and phase for the fault-free and fault condition.

4.11.2 Channel Disconnect Fault Model[6]

The disconnect fault model is implemented in much the same way as the bubble model; air is inserted between the source and sink, plus flow is ceased. In the case of the bubble the actual fault is limited by the bounding geometry, physical equations can only be applied to a valid geometry and not "off" the geometry, as in the actual experimental case. For example, in the experiment the actual fault is the "gap" between the source of the fluid and the reactor inlet. However, the fault model has to be implemented within the reactor geometry in the simulation environment. The simulated implementation of the fault is bounded by the reactor, therefore, the accuracy of the model will dependent upon the

[6] Model File: Impedance_Disconnect.mph

dimensions of the reactor geometry and the size of the inserted fault block. The inserted fault block has to describe the “air” disconnection “gap”. In this case no fault block is inserted, instead the port geometry object is used as the fault block object and described as air having a zero flow condition, Figure 73.

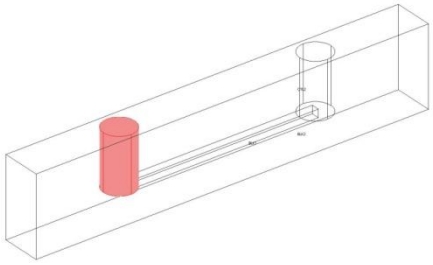


Figure 73 Simulation Geometry for a Channel Disconnection

The results for the experimental and simulated impedance are shown in Figure 74 and Figure 75.

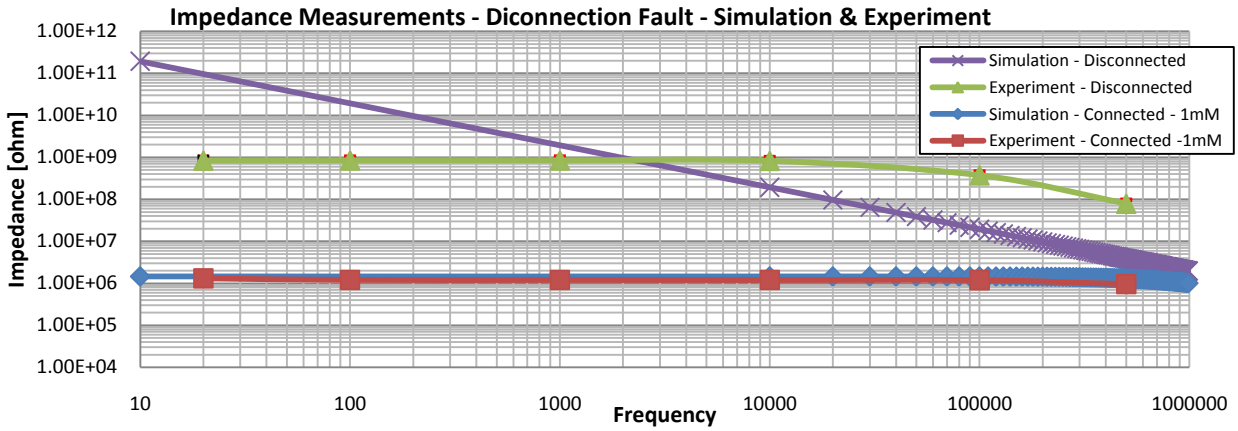


Figure 74 Impedance Magnitude Plot - Channel Disconnection

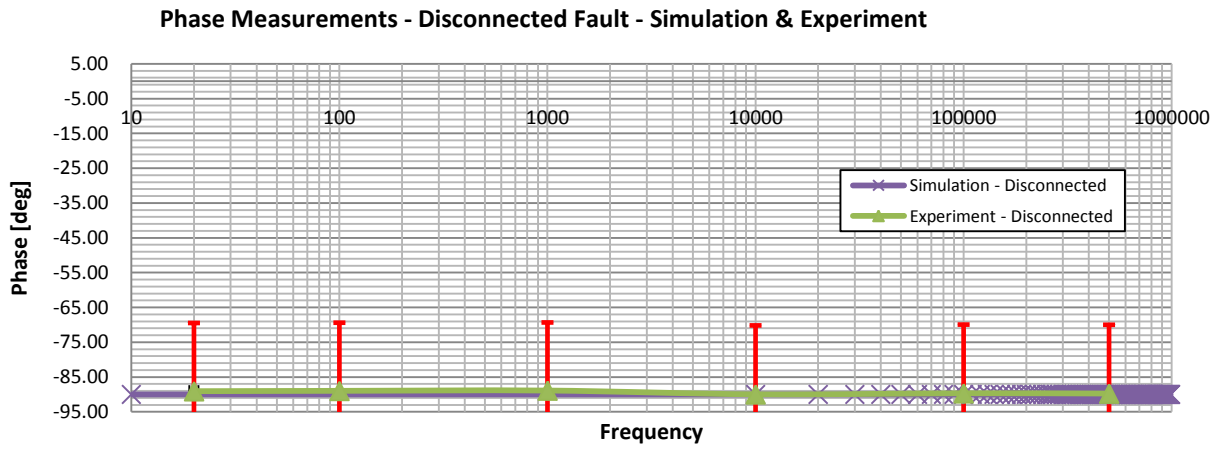


Figure 75 Phase Plot - Channel Disconnection

The impedance results show a large change between channel and tubing connection, and disconnection, approximately 3 orders of magnitude for the real part of the impedance, Figure 74 and Figure 75. The simulated disconnection produces a higher impedance (another 3 orders of magnitude) than the experimental results.

4.11.3 Channel Disconnection Summary

While one may think this is a high inaccuracy it follows from the restrictions to limit the model to the reactor geometry. Therefore the accuracy of this method is a question of fault block dimension control. Here the existing geometry was used. However, a separate geometry of sufficient size to describe the fault, in addition to the reactor geometry could be implemented. This could produce geometry far larger than the reactor itself, resulting in low quality FEM elements and poor simulation results for the microfluidic part of the system. Furthermore, what is the magnitude of a disconnection problem? This fault condition could have a multitude of behavioural models for the same system, and in a design environment, one would not strive to match exact experimental conditions. Our interest is in the development of fault models that describe sufficient behavior for test analysis. The fault block method achieves this, and supports a method of auto-generating this fault condition.

4.12 Reagent / Concentration Faults (Variations)

The monitoring and measurement of reagent and analyte concentrations is an important requirement in microfluidic systems, since these are often critical for correct chemical and biological performance. We therefore investigate the use of impedance spectroscopy to monitor the concentration of KCl, although based on Equation 17, any fluid or reagent could be used with this method, only its concentration and equivalent conductivity are required.

4.12.1 Concentration Experiment

The experimental set-up for measuring the impedance of varying concentrations is shown in Figure 76 and is the same as for the hydrodynamic experiment, Chapter 3. Here three different KCl concentrations which are measured; 1mM, 10mM and 100mM are used. The resulting impedance magnitude measurements are shown in Figure 77.

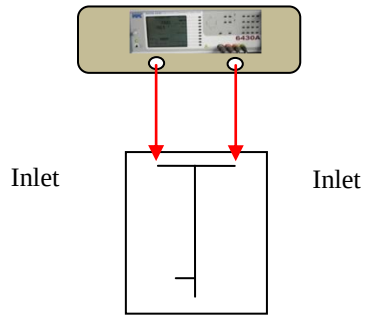


Figure 76 Concentration Experiment

4.12.2 Concentration Simulation

The simulation model is identical to the preliminary cross-validation simulation model, however, here the concentration values are set at 1mM, 10mM and 100mM. Figure 77 offers a comparison between experimental and simulation magnitudes and show the sensitivity of impedance spectroscopy to various concentrations. It may be observed from Figure 77 that the greatest discrimination between conditions occurs below 10 kHz, the creation of this frequency is trivial in modern embedded electronics. These results demonstrate a linear relationship between reagent concentration and real impedance.

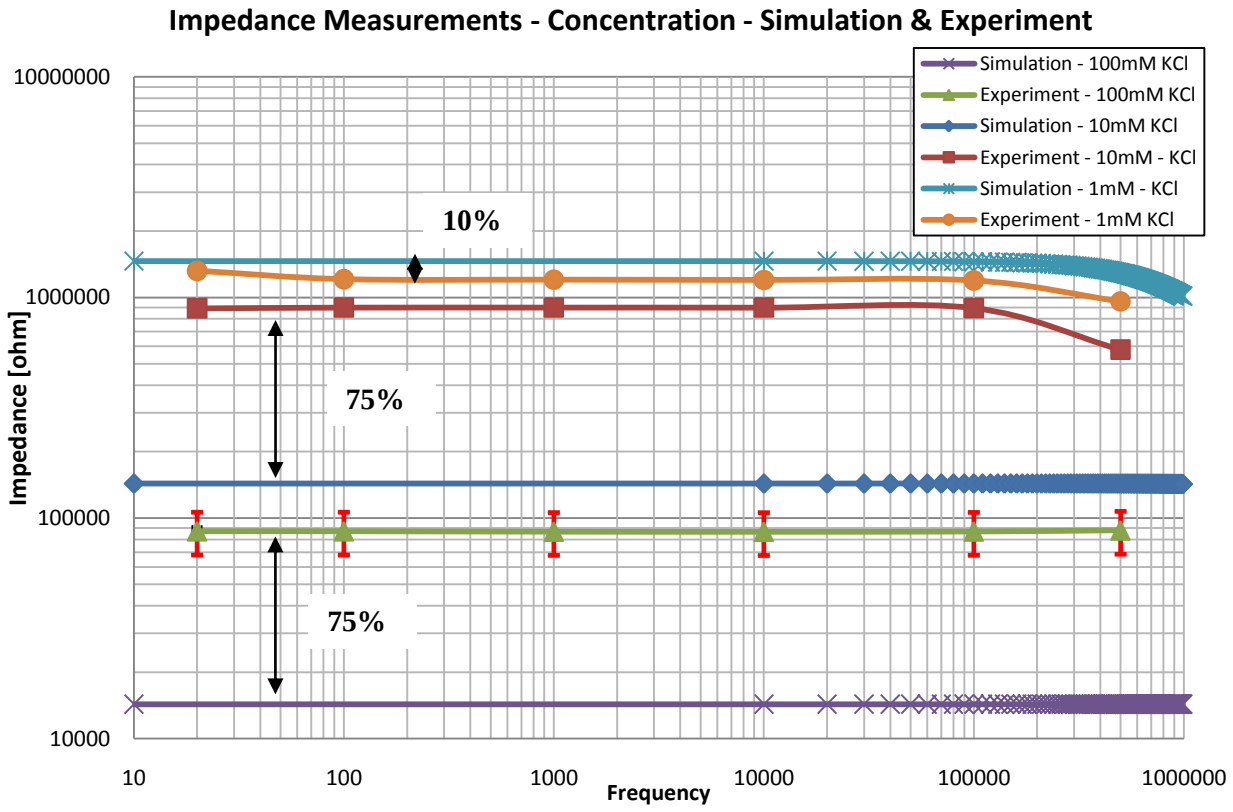


Figure 77 Concentration Impedance Magnitudes

The impedance magnitudes associated with concentration simulation and experiments were 75% different for the higher concentrations 10mM and 100mM, more accuracy was obtained with the lower concentration of 1mM where 10% error was found. Generally the simulation magnitude is lower, providing a conservative estimate of the magnitude.

4.13 Fault Models using Impedance Spectroscopy Detection

A range of models using impedance spectroscopy have been described. The fault block method has been investigated as a generic behavioural fault model template and has shown to be adaptable to all fault conditions presented where this provides a valid method (i.e. not concentration degradation).

While the fault block method might not perfectly describe all conditions, in its simplest implementation, such as partial blockages and channel disconnections, it provides a sufficient behavioural model for producing measured data for the later test analysis. The fault block method has proven to be highly accurate in other fault instances, providing a high degree of cross-validation. The advantage of the fault block method is that it is generic and lends itself to automation of fault injection.

4.14 Levich Sensors

Levich sensors were introduced alongside impedance spectroscopy at the beginning of this chapter. Impedance pectroscopy has shown to be sensitive to wide range of faults and system parameters; however, fluid velocity is a parameter fundamental to microfluidics and requires a separate monitoring approach.

Collins *et. al* [131] provides a useful overview of recent microfluidic flow transducer technology. Many of these designs require specific reactor design to integrate them and largely depend on their surrounding environment to determine their accuracy and sensitivity. The approach which Collins describes utilizes the re-distribution of ions created by the parabolic velocity flow profile, this redistribution causes ions to flow faster mid-channel than those near the walls, within the electric double layer, the rate of flow of these ions effects the electrical admittance when measured with an ac voltage across the channel. This approach is based on the Levich mass transport current limited equation.

4.14.1 Levich Sensor Topology

The Levich mass transport current limited equation was applied by Rees *et al*. [134] to “micro” channels for the measurement of ultrafast voltammetry, Figure 78 provides an overview of the Levich sensor topology.

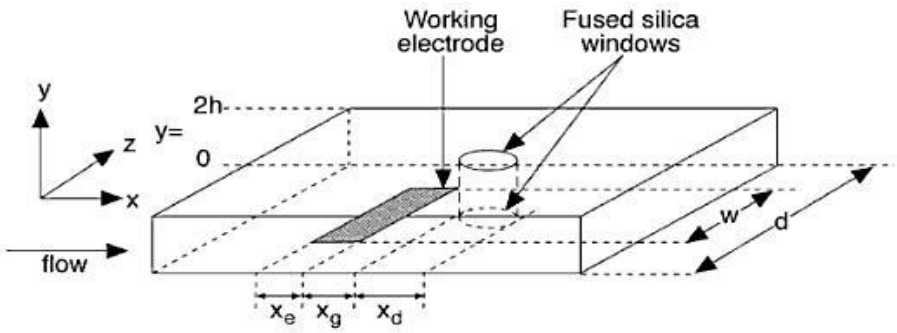


Fig. 8.19 A modified channel flow cell for *in situ* UV/vis spectroscopy. The marked lengths represent the electrode length, the distance between the electrode and the window and the diameter of the window itself. Reproduced from Ref. [16] with permission from Wiley.

Figure 78 Levich Sensor Electrode Topology [134]

An analysis of the published results made by Collins *et al.*[131] is performed to yield a metric by which to compare our simulation results in the absence of direct experimental cross-validation. From the literature the following parameters are deduced (some are not reported explicitly); Concentration, c = 0.8M, x_e = 5mm, w = 200µm, D = $1x10^{-6}$ m^2/s, h = 250µm and d = 500µm. A zero flow current offset of 0.0338mA was reported, the same offset was added to the inferred response shown in Figure 79. The values above the peaks in Figure 79 correspond to the x-axis of the theoretical implementation of the same figure.

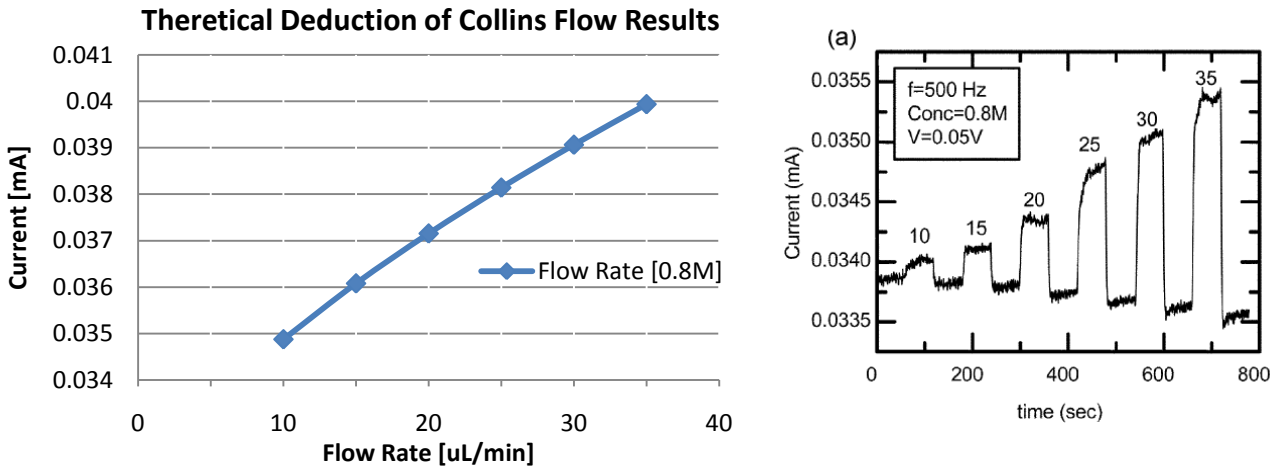


Figure 79 (Left) Theoretical Implementation; (Right) Collins Experimental Observation

The deduction was carried out using the Levich current limited mass-transport equation, the equation was solved using fixed parameters for each flow rate. This theoretical approach forms the basis of the simulation approach, and shows high agreement with experimental findings, Table 5.

Flow Rate [µL/min]	Theoretical [mA]	Collins [mA]	Error [%]
10	0.0349	0.0340	2.6
15	0.0361	0.0342	5.3
20	0.0372	0.0344	7.5
25	0.0382	0.0347	9
30	0.039	0.0352	9.7
35	0.0398	0.0254	11

Table 5 Comparing Theoretical and Experimental Levich Currents

4.14.2 Levich Sensor Simulation[7]

The mass transport current limited approach was investigated in the simulation environment. The geometry studied was on the micro scale, rather than pseudo micro as reported by Collins. For example, the channel width is 100µm (5 times smaller than Collins), the electrode length is 40µm (125 times smaller than Collins), a list of geometrical parameters are found in Table 6. Geometrical block electrodes were inserted into a channel geometry, the channel having width d and height 2h, the electrodes having width x_e, length w and separation, x_g, as shown in Figure 80 (Left) hydrodynamic flow conditions were created using a stokes flow application mode, using the conductive DC application mode to determine the electrical current.

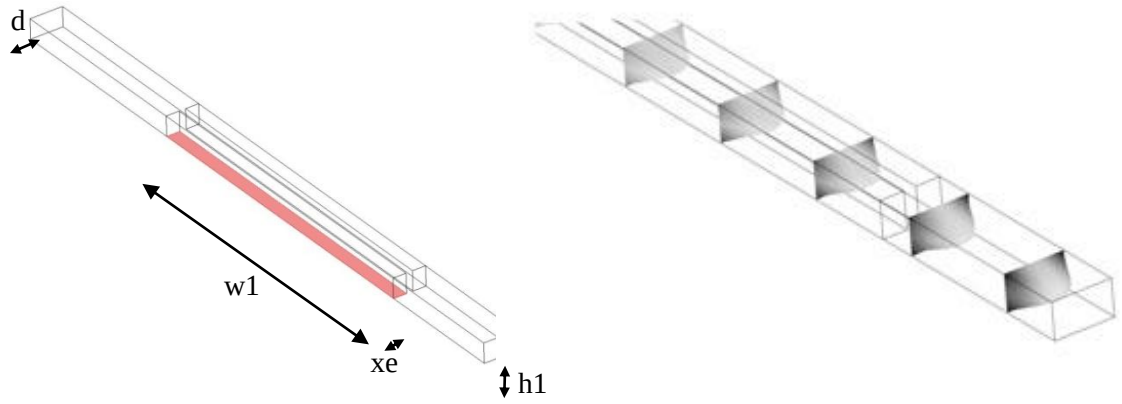


Figure 80 (Left) Simulation channel showing block electrodes; (Right) hydrodynamic flow profile.

The hydrodynamic flow profile shown Figure 80 (Right) demonstrates the reduced wall velocity, compared to the mid-channel (the dimensions are given in accordance with Table 6 and Equation 18). The grayscale shows concentration, the darkest is 300 mol/m^3 and the lightest 878mol/m^3. The plot is shown as a 9 slice deformation plot, where the slices show the velocity profile. The flow is measured using

[7] Model File: Flow Sensor.mph

boundary integration at the outlet, *Flow* (m^3/s). The concentration is integrated over the lower boundary of the electrode block (shown in Figure 80), *ConC* (mol/m).

4.14.3 Simplified Levich Sensors

The implementation of Levich sensors was simplified[8], Figure 81, to aid simulation efficiency and potential auto-deployment of test sensors. Instead of creating the physical electrodes as geometrical shapes, which have to be meshed and solved, an evaluation point is placed in the geometry which is evaluated at each simulation time step using Equation 18 through the global expression technique in COMSOL. In the previous method the flowing concentration was integrated over the electrode boundary, here the dimensions of the electrode are applied numerically.

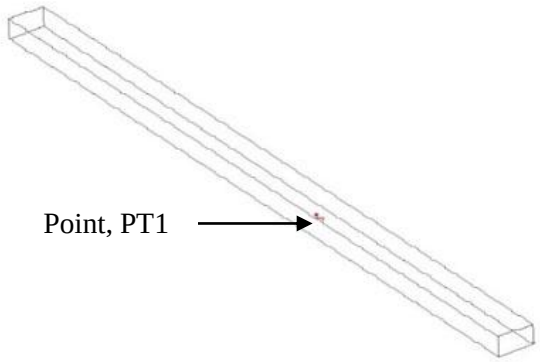


Figure 81 Levich Sensor Geometry

The physical electrode dimensions are provided as "constants" to reduce the computational expense. The following parameters are assigned:

Parameter	Value	Notes
n	1	Assumed only a single electron passed
F	9.65e4[C/mol]	Faraday's constant
D	9.1e-6[cm^2/s]	Diffusion coefficient of species
w1	40e-6[m]	Electrode length
xe	1e-3[m]	Electrode Width
h1	20e-6[m	Height of Channel
d1	100e-6[m]	Width of Channel

Table 6 Levich Sensor Parameters

[8] Model File: Flow Sensor Simplified.mph

In the model configuration the integration variable, *ConC*, [mol/m] is assigned to the measurement point and evaluated for each simulation step. *Flow* is integrated across the channel's outlet flow boundary, per step and the modified Levich mass-transport current equation is solved as a global expression, Equation 18.

$$iCurr = (0.925*n*F*ConC*(D^(2/3))*(Flow^(1/3))*w1*((xe^2)/(h1^2*d1))^(1/3))$$

Equation 18 Levich Simulation Current Expression

4.14.3.1 Simulation Results

The flow rate simulations using the global expression approach demonstrates a cubic root response (see Figure 82), the absolute mass transport current is less than that reported by Collins, given the electrode dimensions are in some cases several orders of magnitude less than Collins.

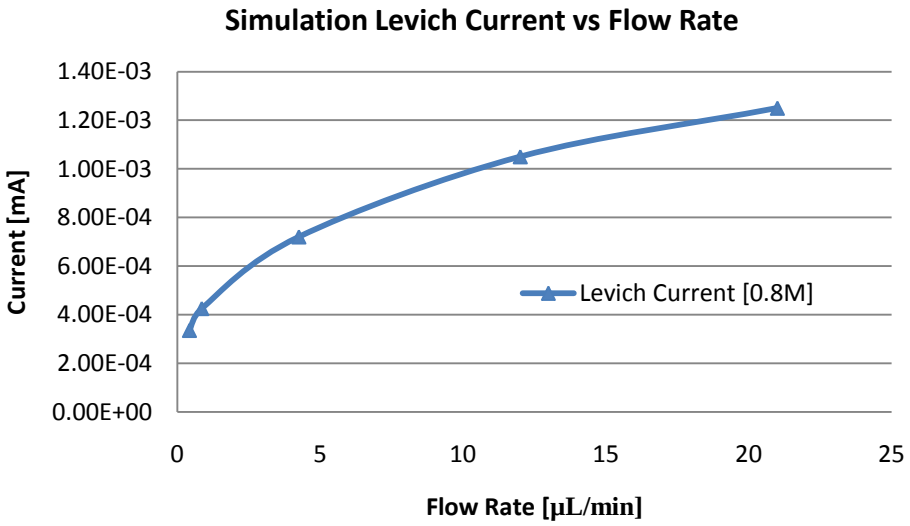


Figure 82 Levich Current vs Flow Rate Plot

4.14.4 Fault Models using Levich Detection

While no fault models have been studied directly using the Levich approach, the fault conditions pertaining to flow can now be detected from measurement of the induced current in the Levich electrode.

The simulated response from the Levich sensor has not directly been cross-validated to experiment, largely because of the unavailability of embedded electrode reactors to this research. However, studying "similar" responses from published work, and through consideration of electrochemical theory, the results produced by simulation show acceptable beheviour; enough evidence to proceed.

4.15 Fault Library Summary

The fault models previously presented are summarised here to describe their implementation in the FEM simulation environment.

4.15.1 Complete Blockage

Model Method		Dimensions	Sub-domain	Boundary Cond's
Fault Block	X	Approx. Full width & Height of channel	Physics as blockage material.	As wall depending on application mode.
Parametric				
Fault Block – Parametric				

4.15.2 Partial Blockage

Model Method		Dimensions	Sub-domain	Boundary Cond's
Fault Block	X	As blockage dimensions	Physics as blockage material.	As wall depending on application mode.
Parametric				
Fault Block – Parametric				

4.15.3 Bubble

Model Method		Dimensions	Sub-domain	Boundary Cond's
Fault Block	X	Length dependent upon condition, width and height as channel	Gas description	walls
Parametric				
Fault Block – Parametric				

4.15.4 Leakage

Model Method		Dimensions	Sub-domain	Boundary Cond's
Fault Block	X	N/A – depending on whether fault block is required	N/A (associative)	As Outlet depending on application mode.
Parametric				
Fault Block – Parametric				

4.15.5 Channel Disconnection

Model Method		Dimensions	Sub-domain	Boundary Cond's
Fault Block		As complete blockage or use existing sub-geometry	Physics as air	As wall depending on application mode.
Parametric				
Fault Block – Parametric	X			

4.15.6 Reagent / Concentration Variation

Model Method		Dimensions	Sub-domain	Boundary Cond's
Fault Block		N/A	N/A	N/A
Parametric	X			
Fault Block – Parametric				

Concentration parameter varied to describe fault condition

4.16 Discussion and Conclusion

In this chapter a range of microfluidic fault conditions have been described, experimented and simulated; an important step in microfluidic test research. The generic method of fault modeling, the fault block, has been introduced and developed to describe a wide range of fault conditions. The strength of the fault block is that a single method is used to inject a range of faulty behavior, into an existing fault-free system model without modification and to describe those faults. This generic method approaches injection and behavior automation, the subject of the next chapter.

The fault block method is not the only method of fault description. Parameteric faults have been described, along with fault block – parameteric faults (a combination of both approaches).

Impedance spectroscopy and Levich electro-chemical sensors have been investigated as two potential methods of microfluidic test. For each test method a simulation technique has been developed to implement the method within the system simulation to obtain test measurement data. Impedance spectroscopy is the most computational expensive, requiring the solving of an additional application layer. Levich sensors only require several additional global expressions.

The concept described in this chapter offers flexibility, accuracy and expandability of fault injection, description and range. The fault library presented may be expanded upon as additional fault types are discovered.

Chapter 5 Simulation Methodology

5.1 Introduction

This chapter introduces an algorithm to automate the simulation work described in the previous two chapters. The purpose of the algorithm is to form a Microfluidic Fault Simulator, MFS. Our research has independently validated the accuracy of the FEM modeling approach using COMSOL for describing the behavior of fault-free microfluidic systems. In addition to this, the previous chapter demonstrated that low abstraction microfluidic fault models can be implemented using the generic fault block method and parametric variation, or a combination of the two.

The use of a fault simulator is a typical starting point of any test investigatory work, examples of which have been found in the Chapter 2 and Milor [14]. Since our interest in this thesis is to investigate suitable test methods and the testability of microfluidic systems, and with no commercially available microfluidic fault simulation software, then the design of the MFS is mandatory for our further work.

The MFS is implemented in MATLAB and uses the COMSOL scripting interface for interaction and manipulation of the system model. The main advantage of this approach is that it requires no abstraction of the system model; the model (FEM structure) is directly exported from COMSOL. This increases the transparency of the MFS and enables its use by multidisciplinary teams and for integration into the wider system workflow.

The MFS determines a nominal fault-free system parameter space by applying a Monte Carlo analysis to achieve parametric variation. Fault injection is achieved via manipulation of the FEM model through the hierarchical scripting interface, based on fault models, within a fault library. Monte Carlo analysis is performed concurrently to provide parametric variation to determine the faulty system parameter space. Test method simulations are applied and measurement data is stored for later test analysis.

5.2 FEM Model Structure

The implementation of the algorithm is made possible through the COMSOL FEM model structure. The FEM model is represented as a FEM structure which is a *scalar structure array*, an array of containers for a set of fields, which are accessible via a scripting interface using an hierarchical dot notation. The data structures in the fields of the FEM structure define different aspects of the simulation, from the geometry objects, the PDE's, the solver type and details of the mesh. A complete description of the FEM structure can be found in COMSOL MATLAB Interface Guide (3.5a) Chapter 1. A brief overview of the fields most used in the algorithm is given here.

5.2.1 Fem.appl

This field denotes the application modes which constitute the model; the class field specifies the physics associated with the application mode. The application mode governs the boundary and sub-domain conditions.

5.2.2 Fem.geom

The geometry field associates the draw field, which describes the collection of objects which constitute the model geometry, with the mathematics to describe the simulation model.

5.2.3 Fem.mesh

The mesh field stores the mesh description for the whole geometry for each application mode.

5.2.4 Fem.sol

The solution field is associated with `femsol` , the most important component of the solution is the matrix `fem.sol.u`, whose columns are *solution vectors* containing values for the degrees of freedom.

The proposed methodology is designed to be integrated into a single workflow. This begins with the ability to export the FEM model (fault-free) from within the COMSOL workspace into the fault simulator. COMSOL supports a FEM structure "export" feature, which allows the simulation structure to be exported to MATLAB to allow interaction through a scripting approach. The compilation of the structure is a background task and transparent to the user, there is no requirement for model abstraction.

5.3 Overview of the Fault Simulation Algorithm

The MFS algorithm serves two purposes; nominal fault-free simulation and fault simulation. The initial COMSOL simulation in the COMSOL workspace is used for proof-of-concept studies and validating the design. It is imperative that the nominal fault-free design space is known to aid later test metric determination. Therefore, the fault simulation algorithm allows multiple fault-free simulations using nominal input parameters and their associated tolerances (Monte Carlo analysis). The complete simulation algorithm is provided as a flowchart in Figure 83.

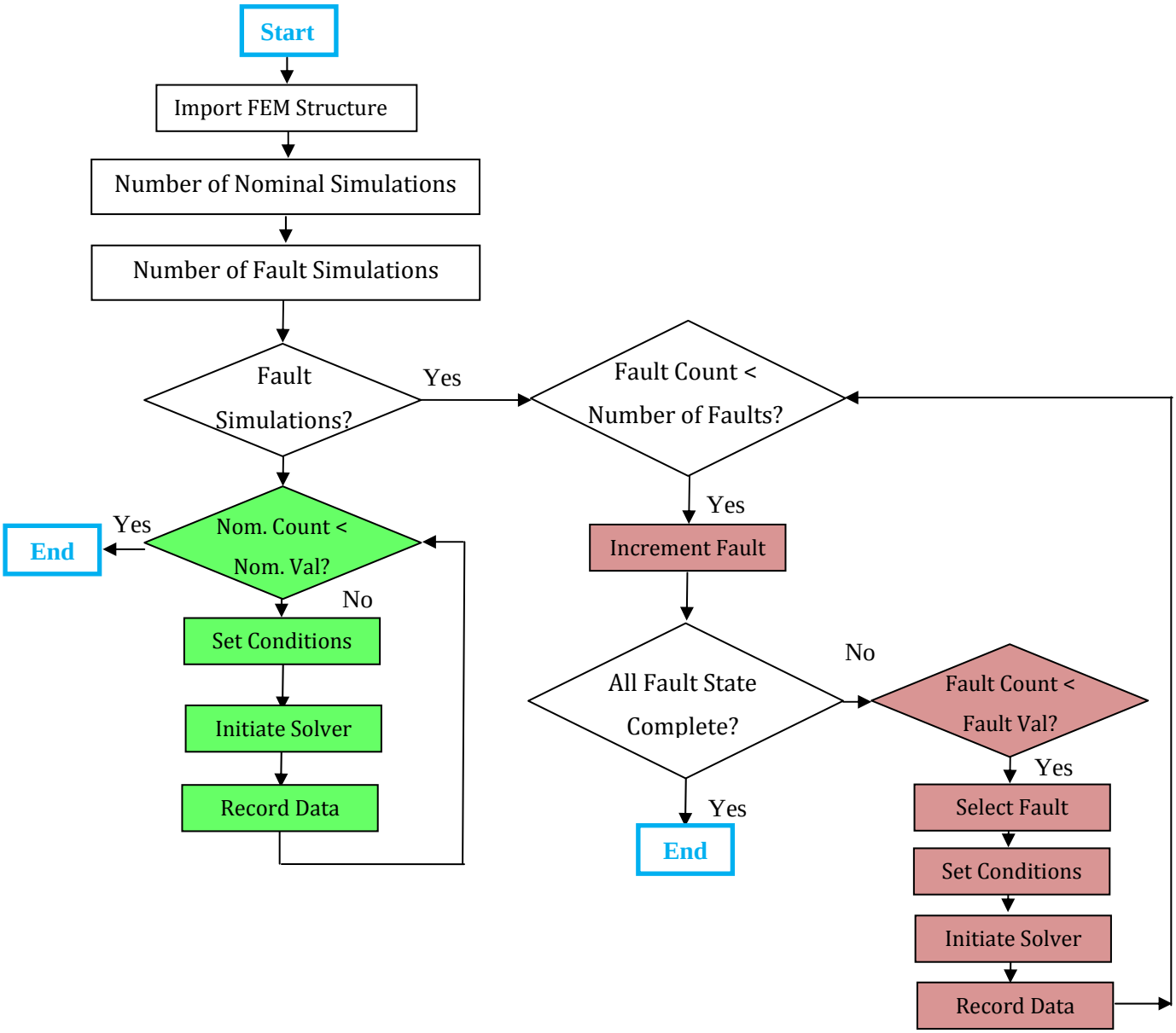


Figure 83 MFS Algorithm

The fault conditions (fault dictionary) which are injected by the MFS are determined prior to simulation throughout this thesis. However, it is envisaged that the MFS could parse the FEM structures application modes to gauge the type of system and from that build a fault dictionary dynamically from the available fault models in the fault library (the subject of further work). Fault conditions from the fault dictionary are systematically injected and simulated for the predetermined number of nominal simulations. Once the predetermined number of nominal simulations has been solved for a given fault condition the next condition is injected and the process repeated.

5.4 Simulation Procedure

The fault-free and faulty system simulation algorithm paths share a common procedure to allow the system model to be prepared and solved, the only addition being fault injection when the faulty path is considered.

5.4.1 Fault-Free Nominal Procedure

1. Nominal Parameter variance
2. FEM Compilation
3. Mesh
4. Time Dependant Solve
5. Parametric Solve (system dependant)
6. Functional & Test Sensor Evaluation
7. Data Record

5.4.2 Fault Nominal Procedure

1. Nominal Parameter variance
2. Fault Injection and Mapping
3. FEM Compilation
4. Mesh
5. Time Dependant Solve
6. Parametric Solve (system dependant)
7. Functional & Test Sensor Evaluation
8. Data Record

Each section of the *nominal* simulation procedure will be described next in detail. Some of the steps, such as, meshing and solving are intrinsic to FEM simulation and the reader is referred to COMSOL documentation.

5.4.3 Nominal Parameter variance

This is the first step in the *nominal* procedure. It implements distributed (bounded) parameter values, based on the systems specification, using Monte Carlo analysis. Monte Carlo analysis provides system variance based upon nominal parameter values and their tolerances.

```
parameter1 = normrnd(300e-6, (300e-6*0.2)); %velocity1 - +/-20%
parameter2 = normrnd(300e-6, (300e-6*0.2)); %velocity2 - +/-20%
 parameter3 = normrnd(297, (297*0.1));       %Operating Temp. +/-10%
 parameter4 = normrnd(0.8, (0.8*0.01));      %Concentration 1 +/-0.01%
 parameter5 = normrnd(0.2, (0.2*0.01));      %Concentration 2 +/-0.01%
```

Figure 84 Parameter Monte Carlo Analysis (MATLAB code)

Figure 84 shows how nominal system parameters, such as, velocities, concentrations and temperature can be "statistically" adjusted based on their tolerance. In MATLAB this is coded using the *normrnd* function to generate a pseudo random number based on the value provided and its bounds. The newly generated value is then written back into the corresponding FEM constant which represent that parameter within the system.

```
 s = sprintf('%10.2d[m/s]',parameter1);
 femt.const{2} = s;
```

Figure 85 Writing a new parameter value into the FEM structure (MATLAB code)

5.4.4 FEM Compilation

The FEM model requires re-compilation due to the modification of the constants, or in the case of the faulty system path, the injection of the fault block. The *multiphysics* function compiles the application mode and adds to the application mode for the system FEM structure.

```
%Compile Model
femt = multiphysics(femt);
```

Figure 86 MFS Multiphysics Script

5.4.5 Mesh

Meshing of FEM models is out of the scope of this thesis. The mesh generation throughout this thesis, unless stated otherwise, has been the default mesh. It is worth noting at this point that alongside the model in the COMSOL workspace, a *history* (. m file, a MATLAB file) is automatically generated which provides the script equivalent of the activities in the COMSOL GUI. This is useful for extracting set routines for use in the MFS procedure, such as using the automatically generated mesh script, (See Figure 87).

```
femt.xmesh=     meshextend(femt, ...
                'geoms',[1], ...
                'eqvars','on', ...
                'cplbndeq','on', ...
                'cplbndsh','off', ...
                'linshape',[1], ...
                'linshapetol',0.1);
```

Figure 87 MFS Mesh Script

5.4.6 Time Dependant Solve

A time dependant solver is required for the majority of simulations in this thesis, since the original use of the fault-free FEM model is to validate the systems operation and performance for a specified “run”. In this case the `femtime` solver is used. For a complete list of function arguments refer to the COMSOL User Manual.

```
femt.sol= femtime(femt, ...
          'u',0, ...
          'method','eliminate', ...
          'conjugate','off', ...
          'symmetric','auto', ...
          'solcomp',{'u2','p2','c','v2','w2'}, ...
          'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c'}, ...
          'odesolver','bdf_ida', ...
          'tlist',[colon(0,1,50)], ...
          'rtol',0.01, ...
          'masssingular','maybe', ...
          'consistent','bweuler', ...
          'tout','tlist', ...
          'tsteps','free', ...
          'linsolver','spooles', ...
          'mcase',0);
```

Figure 88 MFS Solver Script

From Figure 88 the key arguments will be introduced here. The parameter `u` determines the initial solution for the solver to use, since this is the first call of the solver, the solution is 0. `tlist` is the initial, incremental and final simulation time step; in this case the system is studied for the period 0 to 50 seconds

at 1 seconds intervals. `linsolver` determines the type of solver to use, this is optimized dependent upon the application mode used, the solver selected in the COMSOL GUI may be extracted for use in the MFS, Figure 88.

5.4.7 Parametric Solve

The parametric solver is used to solve the electric currents application layer (emqvw) which is used for the impedance spectroscopy test method. For systems were the IS test method is not required, then, this step of the procedure may be omitted.

```
femt.sol= femstatic(femt, ...
            'init',femt0.sol, ...
            'u',femt0.sol, ...
            'method','eliminate', ...
            'nullfun','auto', ...
            'complexfun','off', ...
            'matherr','on', ...
            'solfile','off', ...
            'conjugate','off', ...
            'symmetric','auto', ...
            'solcomp',{'Vportconstr1_g1_emqvw','V'}, ...

            'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'},
            ...
            'pname','nu_emqvw', ...
            'plist',[colon(100,100000,1000000)], ...
            'porder',1, ...
            'oldcomp',{}, ...
            'ntol',1.0E-6, ...
            'maxiter',25, ...
            'nonlin','auto', ...
            'damping','on', ...
            'hnlin','on', ...
            'linsolver','pardiso', ...
            'uscale','none', ...
            'mcase',0);
```

Figure 89 MFS Parametric Script

These parameters are explained in detail in the COMSOL User Manual. Noteworthy parameters for discussion here are; femstatic, init, pname, plist and linsolver. *femstatic* calls the parametric solver used for solving the model for impedance data, using the following parameters. *init* provides the solver with the initial solution, which is `femt0.sol`, the solution to the time dependant simulation. *pname* is the parametric variable which is incrementally given the values listed in *plist*. In this case the variable *nu_emqvw* (the frequency variable) is incremented from 100 to 1000000 in 10000 steps. *linsolver* specifies the solver type for the parametric solver to use, (Figure 89).

5.5 Functional & Test Sensor Evaluation

Functional sensor values are the results of system expressions, evaluated at pre-determined points or boundaries, based upon the system application mode physics, expressed at discrete simulation steps. Therefore, the computational overhead of these sensors is minimal. Test sensor evaluation in the case of Levich sensors has a fixed overhead which is derived from additional expressions used to determine the current based on system conditions. Impedance spectroscopy requires the use of a parametric solver which has significant computational overhead compared to functional and Levich sensor evaluation, as a complete re-solve using a parametric solver is required per scan.

5.5.1 Functional Sensor

Functional sensing points may be declared during the scripting phase. An evaluation point may be established at any valid location within the geometry. The `postinterp` function is used with the prototypes of the solved FEM structure, the variable to be evaluated and the geometry co-ordinates.

```
var = postinterp(femt, 'v2', [5e-5; 4.95e-3; 20e-6]);
```

Figure 90 MFS Functional Sensor Script

The *postinterp* function evaluates the variable at the current simulation time step. Since in the algorithm this is applied post-solver then the functional sensor values are evaluated for the final time step of the simulation, (Figure 90).

5.5.2 Test Sensor Evaluation

5.5.2.1 Levich Sensor Script

The Levich sensor data is calculated as part of the time-dependant simulation, through the processing of the global expressions. During each simulation cycle the expressions are evaluated using the `postglobaleval` expression, the prototypes required are the solved FEM structure and the global variable.

```
var = postglobaleval(femt,{'iOut'});
```

Figure 91 MFS Levich Sensor Script

The *postglobaleval* function evaluates the variable for all time steps. For data storage and later assessment only the final time step value is recorded, which is in keeping with the functional sensor data, Figure 91.

5.5.2.2 Impedance Spectroscopy

The impedance magnitude and phase are stored in the global variable, Z11_emqvw. The magnitude data may be extracted by evaluating `abs(Z11_emqvw)` and the phase by `180*arg(Z11_emqvw)/pi`.

The `postglobaleval` function is used to evaluate the impedance matrix; the prototypes required are the solved FEM structure and the global variable, a typical example:

```
mag = postglobaleval(femt,{'abs(Z11_emqvw)'});

phi = postglobaleval(femt,{'180*arg(Z11_emqvw)/pi'});
```

Figure 92 MFS Impedance Spectroscopy Script

Using this technique the impedance and phase magnitudes are recorded for each frequency applied during the frequency sweep, Figure 92.

5.5.3 Data Record

The simulation data generated by the MFS is stored in these structures for later test analysis (Chapter 6). For test analysis the nominal fault-free parameter space must be determined, the “FFParaLog” is an m by n array where m is the nominal simulation number and n the parameter to be recorded, shown in Figure 93.

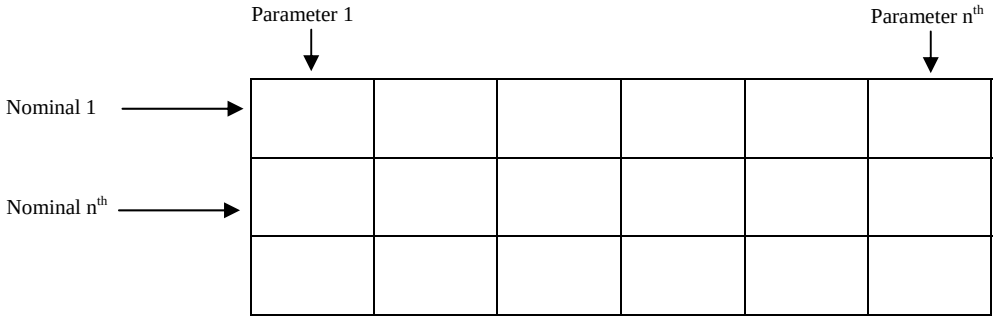


Figure 93 Fault-Free Parameter Storage Structure

The faulty system parameters are recorded in the “FaultParaLog” , shown in Figure 94, which is a m by n table by z where m is Fault applied, n the nominal simulation number and z is the parameter of interest.

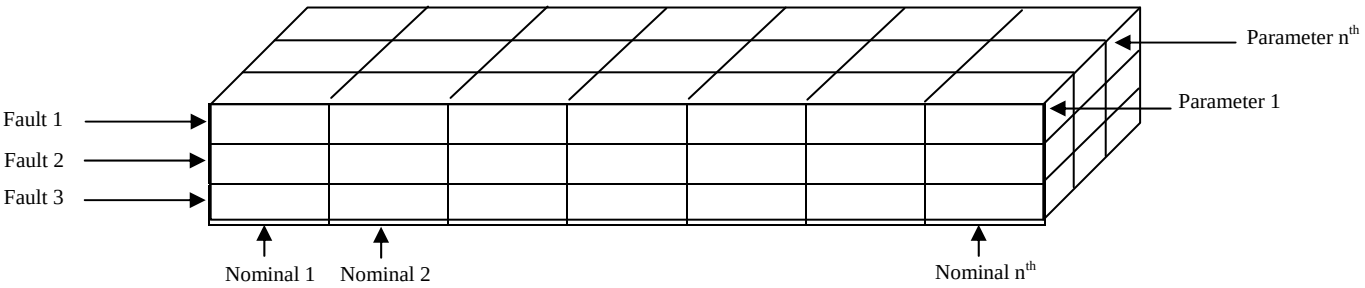


Figure 94 Fault Parameter Storage Structure

A separate array, FaultCond, Figure 95 stores the applied Fault Condition, when the algorithm is operated in Fault Mode.

```
FaultCond(u,i,6)={'Right Pump Fail'};
```

Figure 95 MFS Fault Condition Record Script

The impedance magnitude and phase are stored for all frequency responses.

```
FFImpedanceMag1(u,1) = postglobaleval(femt,{'abs(Z11_emqvw)'});
FFImpedancePhi1(u,1) = postglobaleval(femt,{'180*arg(Z11_emqvw)/pi'});

FImpedanceMag1(u,i)  = postglobaleval(femt,{'abs(Z11_emqvw)'});
FImpedancePhi1(u,i)  = postglobaleval(femt,{'180*arg(Z11_emqvw)/pi'});
```

Figure 96 MFS Impedance Data Record Script

An example of the impedance data array for a faulty system simulation, `FImpedanceMag1(u,i)` is shown in Figure 97.

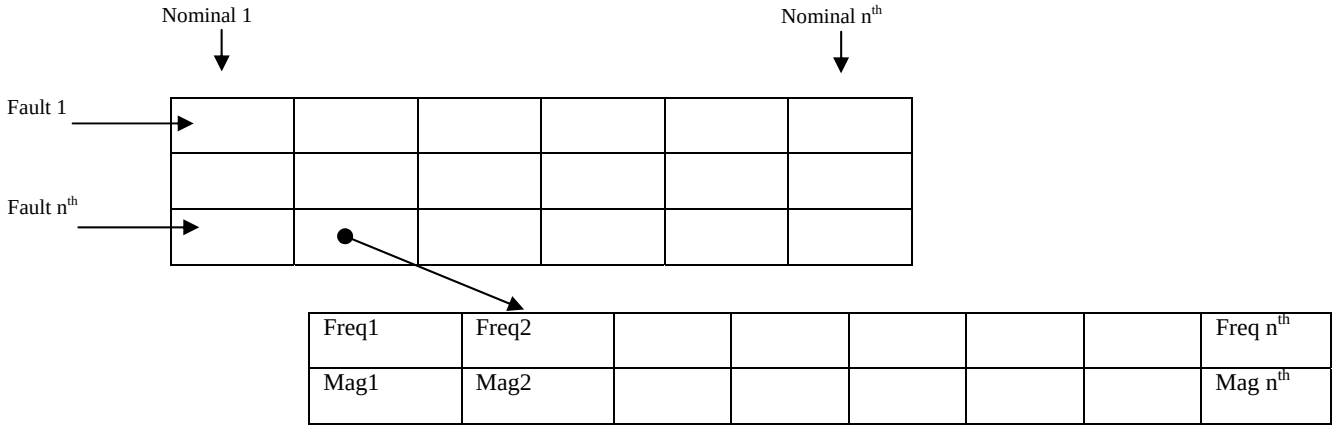


Figure 97 Impedance Data Storage Structure

5.6 Fault Injection and Mapping

Fault conditions and their models were introduced in Chapter 4. Described here is the procedure to determine what faults to use, how frequently to use them, and where in the system to implement them.

5.6.1 Fault Injection Procedure

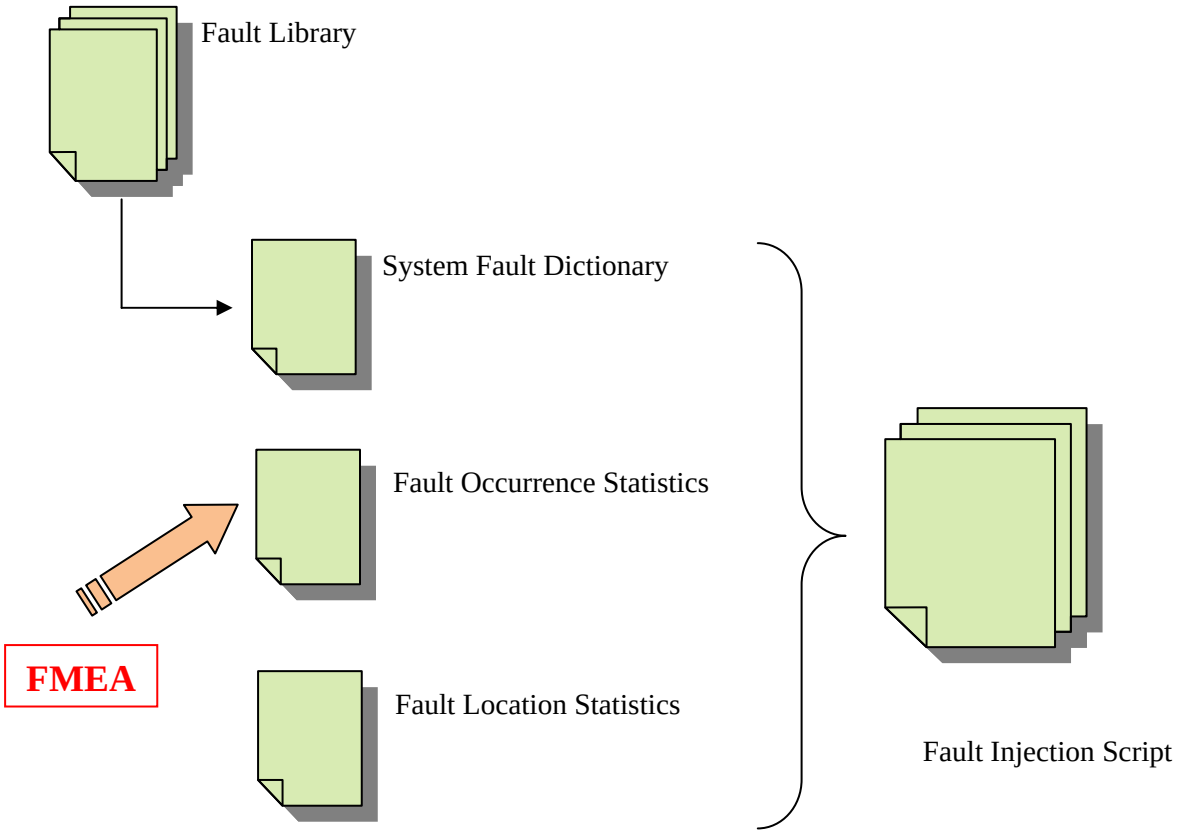


Figure 98 Fault Injection Procedure Diagram

Figure 98 overviews how faults are implemented in the fault simulation algorithm. A Fault Library is a compilation of all possible (known) microfluidic system faults and possible manufacturing defect faults. As faults are discovered and their behavior modelled, they are added to the Fault Library. A system fault dictionary is a finite selection of faults from the fault library, appropriate to the system under test. The algorithm will cycle through these faults injecting them in turn. However, a possible modification to the algorithm would be to parse the application mode namespaces in the FEM model

structure to determine the most likely selection of faults, based on the application mode and the therefore the type of system that is being simulated (subject of further work, here this is carried out manually).

The expected frequency of a fault to be injected is determined by the “Occurrence Statistics” file, due to the unavailability of fault occurrence statistics during this work all faults are treated equal, and each is injected the same number of times. However, provision in the algorithm has been made to allow occurrence statistics to be implemented as they become available.

Finally, the fault injection location in the system geometry (the most likely location of occurrence given the system type and geometry) is determined by the “Location Statistics” file. Again these statistics are not known, so predetermined locations are used in this work, based upon intuition and experience. Provision for this data has been implemented in the algorithm.

An alternative approach would be to use Failure Mode Effect Analysis (FMEA), to select the most likely fault, or the one with the most consequences, its frequency and location. A similar technique has been reviewed in the Chapter 2.

The outcome of this process is the fault description, which is implemented in the system model via the fault injection script.

5.6.2 Fault Injection Script

There are three techniques which the fault algorithm uses to inject and describe the fault behaviour in the system simulation. Fault Block, Parametric variation and Fault Block–Parametric variation. These were described in Chapter 4.

5.6.2.1 Fault Block

The Fault Block technique uses the script shown in Figure 99 to inject the Fault Block into the geometry at a pre-determined location (which could be governed by the location file); the physics of the Fault Block are then set to describe the desired fault behaviour.

The function `block3` creates a 3D block (the Fault Block) and contains the prototypes to customise the size, location and orientation of the block. The size of the block varies according to the type of fault being described (see chapter 4). For example, the size (width and height) of the block my assume the width and height of the channel in which it is placed to describe a complete blockage, and less if a partial blockage is to be described. The location is selected depending on where the user would like to implement the fault condition. The orientation prototype may be used if the channels are non-

perpendicular to either the x or y-axis. The return variable from the `block3` function takes the results of the block structure, in this case `g6`.

```
% Fault Block Geometry
g6=block3('50e-6','0.25e-3','30e-6','base','corner','pos',{'25e-6','4.5e-
3','0'},'axis',{'0','0','1'},'rot','0');

% Forming a Composite Geometry

g7=geomcomp({g1,g6},'ns',{'EXT1','BLK1'},'sf','EXT1+BLK1','face','none','edge
','all');

% Analyzed geometry
clear p s

femt.draw.p.objs={g2,g5,g4,g3};
femt.draw.p.name={'PT1','PT4','PT3','PT2'};
femt.draw.p.tags={'g2','g5','g4','g3'};

femt.draw.s.objs={g7};
femt.draw.s.name={'CO1'};
femt.draw.s.tags={'g7'};

femt.draw=struct('p',femt.draw.p,'s',femt.draw.s);

femt = geomanalyze(femt);

% Application Mode Manipulation to describe model behaviour
femt.appl{1}.equ.ind = [1 2];
femt.appl{1}.equ.sigma = {'conductivityL' '0'};
femt.appl{1}.equ.epsilonr = {'80' '1'};

femt.appl{2}.equ.ind = [1,2];
femt.appl{2}.equ.D = {'D','0'};
femt.appl{2}.equ.w = {'w2','0'};
femt.appl{2}.equ.v = {'v2','0'};
femt.appl{2}.equ.u = {'u2','0'};

femt.appl{3}.equ.ind = [1,2];
femt.appl{3}.equ.eta = {'mat1_eta(T[1/K])[Pa*s]' ,'1.98[Pa*s]'};
femt.appl{3}.equ.rho = {'mat1_rho(T[1/K])[kg/m^3]' ,'1.3[kg/m^3]'};
```

Figure 99 Fault Injection Script

The `geomcomp` function creates a composite geometry. The Fault Block must form a "union" with the fault-free geometry. The prototypes for the `geomcomp` function permit such a composite to be formed. `{g1,g6}` selects the two geometrical objects to become composite, in this case g1 is the original geometry and g6 the fault block previously created. The prototype `'ns'{'EXT1','BLK1'}` assigns the namespaces `EXT1` and `BLK1`, respectively. The act of performing a "union" is designated by `'sf','EXT1+BLK1'.`

`femt.draw` is required to perform mandatory draw compilation when handling geometrical objects, with `struct('p',femt.draw.p,'s',femt.draw.s)`updating the femt structure with the new geometry. Finally `femt = geomanalyze(femt)`analyses the geometry to ensure that all the changes are legal and compiles the geometry for use.

To complete some fault models manipulation of the sub-domain and boundary conditions is required. Sub-domain and boundary conditions are accessed through the application namespace `.appl`. Sub-domain indexes are described by `.equ.ind`. This namespace does not provide sufficient information to exclusively describe the sub-domain, other fields are required and depend upon the application mode. For example, in the electric currents application mode `.equ.sigma` and `.equ.epsilonr` are used to complete the description of the sub-domain conductivity. Figure 100 shows the sub-domain indexes 1 and 2, their location in the field represents the sub-domain number and their value determines the assignment of the variables in the other associated fields. Therefore, in this case sub-domain 1 has the value 1 which refers to a conductivity (signma) *conductivityL* and permitivitty (epsilon) *80*.

```
femt.appl{1}.equ.ind = [1 2];
femt.appl{1}.equ.sigma = {'conductivityL' '0'};
femt.appl{1}.equ.epsilonr = {'80' '1'};
```

Figure 100 MFS Sub-domain Script

The same procedure is followed for boundary condition assignment, Figure 101. The element type field array `.bnd.eltype` states the possible conditions, these conditions are applied to boundaries in the same way conditions are applied to sub-domains. The element type index is applied to the boundary index `.bnd.ind` where location in the boundary index field corresponds with the boundary in the geometry.

```
femt.appl{1}.bnd.eltype = {'nJ0','port','V0'};
femt.appl{1}.bnd.ind = [2,1,1,1,1,3];
```

Figure 101 MFS Boundary Condition Script

5.6.2.2 Parametric Script

In the nominal simulation path parameter variance is introduced using the `normrnd` function in MATLAB. A parametric fault simply expands the pseudo bounds (bounded randomness). For example, the nominal fault-free velocity parameter would be described as having +/-20% variation as shown in Figure 102.

```
parameter1 = normrnd(300e-6, (300e-6*0.2)); %velocity1 - +/-20%
```

Figure 102 MFS Parameter Script

For a velocity flow fault the generated value might have bounds +/-100%.

5.6.2.3 Fault Block - Parametric Script

Some faults descriptions require a combination of parametric variance and the use of a Fault Block. One such example would be a flow inlet channel disconnection. The flow parameter would be reduced to zero and a fault block would be inserted described as air, to simulate the air between the tubing electrode and the channel electrode.

5.7 Fault Injection & Mapping Summary

This section has shown that the simulation procedure may be adapted to include the injection of faults, based on the fault models described in Chapter 4. Furthermore, while the type of fault, its location and frequency of injection are hard coded in the algorithm presented in this thesis, the algorithm remains flexible enough to add fault occurrence, location and type determination in the future, with little modification. One suggested mechanism is the use of an Failure Mode Effect Analysis (FMEA) approach.

5.8 Discussion & Conclusion

In this chapter a MFS algorithm has been described. The creation of the MFS was prompted by the lack of commercially available simulators, and the requirement to perform fault simulations or test evaluation.

The simulator supports the use of the FEM structure, available via export from the COMSOL workspace without abstraction, a key benefit for any tool wishing to sit within an existing workflow. The simulator allows multiple nominal fault free system simulations to be performed, followed by fault simulations using the same process.

The scope of the fault simulations is limited at present by the lack of fault statistics. For example, the selection of fault types, their frequency of occurrence and their likely locations within a system, could all be used by the algorithm, but this information is not available due to the immature nature of the field. Future development may include parsing the FEM structures application modes to select faults to inject and using FMEA to determine usage criteria of fault models.

Our interest in the MFS in this thesis is to further develop the test methodology by providing a means of combining fault-free systems and fault models and simulating test methods to generate measurement data for further test analysis.

Chapter 6 Test Analysis

6.1 Introduction

The previously described Microfluidic Fault Simulator (MFS) workflow, is structured in such a way that the stored measurement data from the simulation may be fed into existing mixed-signal test analysis techniques.

Analogue and Mixed-signal system parameters are continuous in nature with an element of uncertainty due to process variations, aging, surrounding environment etc Figure 103. Therefore, their fault spectrum cannot be enumerated as in the classical digital domain where there are 2n stuck at faults for a circuit with n interconnections. Heterogeneous systems with multiple interacting domains, even when those domains are electrical variations still present significant test challenges; test therefore becomes more complex when other than electrical domains are present, such as MEMS and microfluidic systems.

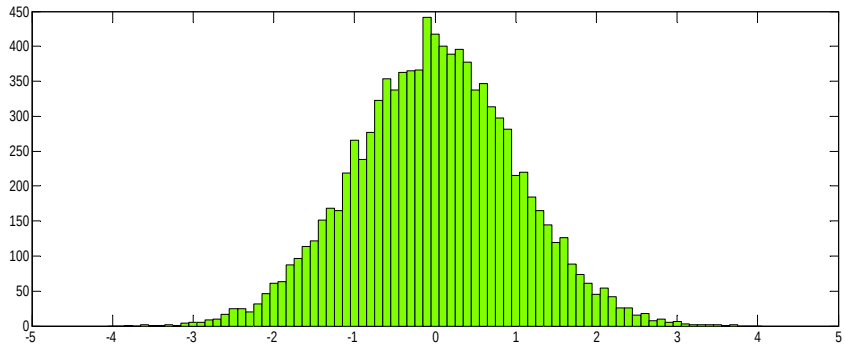


Figure 103 Normal Distribution of a System Parameter

Test strives to classify the system into a Boolean outcome; pass or fail, or a binning designation. Commonly this is done by directly verifying functionality, which is time-consuming and has been proven not to be sensitive to some parametric faults [14]. For some proposed microfluidic systems, such as DNA processing cartridges, functional test methods will be prohibited by cross-contamination in their subsequent use as legal evidence and for other systems the sheer complexity of the system architecture will inhibit this approach. While this research does not explicitly derive a structural test method, it does examine the most effective tests and provides a metric of their ability to classify the system outcome correctly.

This chapter builds on the simulation work previously described for obtaining fault-free and faulty system data. Now that data is used to determine the effectiveness of the proposed test method. Well known analogue and mixed-signal probabilistic test metrics are investigated, to demonstrate that the simulation and test methodology presented in this thesis provides a means of integrating highly heterogeneous systems into an established workflow, within the mixed-signal test community.

Mixed-signal circuits, MEM's and microfluidic systems are subject to continuous parameter variability, process and component tolerances, which, along with the measurement errors, cause ambiguity in the measurement data for both a fault-free and faulty systems. Bayesian frameworks methods have recently been developed to aid the decision making of the test outcome based on an assumed *a priori* probability [15]. This assumed *a priori* statistic is not known in many circumstances, therefore the more conventional approach is the *frequentist* (or classical) approach popularized in 1933 by Neyman-Pearson [16] where no *a priori* data is required to determine a classification; a null hypothesis is proposed and tested. This is pertinent to the determination of the classification of a system as fault-free or faulty based on a particular test in isolation of other factors, such as the manufacturing procedure. This method will be considered in this chapter.

6.2 Neyman-Pearson Review

Zjajo *et. al.* [17] use the Neyman-Pearson decision criteria to assist their structural fault modeling and detection by analyzing DC node voltages to detect process variations. Khouas [18] recognizes the high cost of iterative analog fault simulation and uses a runtime probability assessment to determine if a fault is detectable, if so stopping the simulation to reduce overhead. Later they proposed [19] the FDP (Fault Detection Probability), and Abderrahman *et al.*[20, 21] extended this approach of *probability of detection*, by identifying an upper and lower *guaranteed detectability region* (Figure 104) , an *uncertain region* (partial detectability) and a *tolerance range*. They use a Constraint Logic Programming (CLP) based optimization method, derived from the applied test frequency to determine the upper and lower bounds of the tolerance window, X_{il} and X_{iu}, respectively.

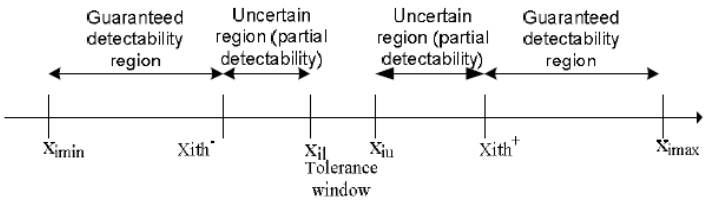


Figure 104 Guaranteed & Uncertain Detectability Regions [21]

Milor [14] provides a useful review of many early analog and mixed-signal simulation and test methods, however, no single technique has become dominant, due to the differing requirements of each

system, but the metrics of establishing the "effectiveness" of a particular test have become common place. The most fundamental being the probability of detection.

6.3 Probabilistic Test Approach

The probabilistic approach to test outcome is not desirable, there is no clear pass or fail. However, it is the most realistic, given the continuous nature of the system parameters and their associated variances. Figure 105 provides an example of how random parameter variation influences test outcome. If parameter, X is to be considered the test parameter then a value of 4 would correctly classify the system as faulty and a value of -1 would correctly classify the system as fault-free, however, what if X = 1.5?

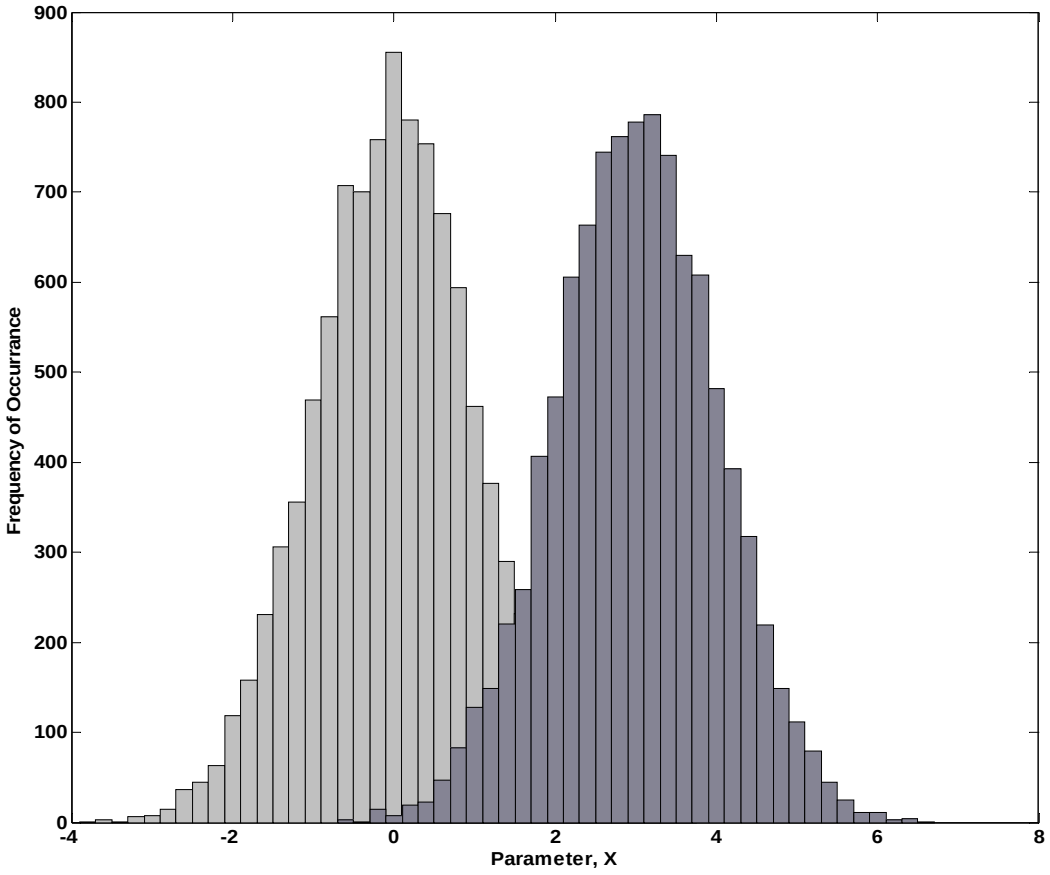


Figure 105 Histogram of Fault-free and Fault Distributions

6.3.1 Probability of Detection Definition

The determination of the *probability of detection* metric requires a large statistical sample of system data, which may be achieved through manufacturing measurements, or more likely from a Simulation before Test, SbT approach using Monte Carlo fault simulation. From system simulation (fault-free and faulty) parameter distributions, (see Figure 105), are determined.

For a given measurement, ϕ, the probability density functions are derived for the fault-free case, G, and faulty cases, F_x and the system split into $p(\phi|G)$ and $p(\phi|F_x)$ resulting in Figure 106.

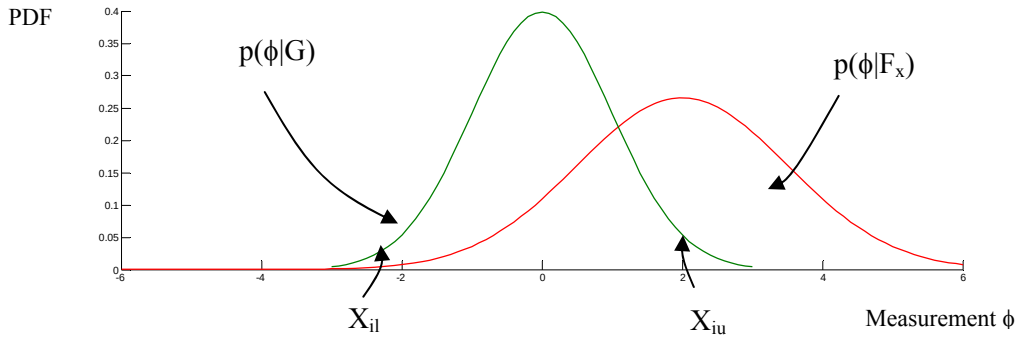


Figure 106 Probability Density Functions Fault-Free and Faulty

The region of acceptability, ROA, is represented in Figure 106 by the upper and lower bounds, X_{il} and X_{iu}, respectively. The limits of the ROA have been given a number of definitions in the literature. The most common is a multiple of the standard deviation, typically, 3σ as this has its origins in manufacturing process quality [22].

6.3.2 Hypothesis Testing

In general our interest is to determine the decision, pass or fail, for a system based on a series of test measurements, ϕ_i. In the general case, Equation 19 our decision could be based on the discrimination function derived from the *probability of detection*.

$$g(\phi) = \begin{cases} 1 \text{ (pass) if } X_{il} < \phi < X_{iu} \\ 0 \text{ (fail) otherwise} \end{cases}$$

Equation 19 Probability of Detection General Case

The probability of detection approach is incomplete due to the potential for partial overlap between parameter values for a fault-free and faulty system. This will result in errors in the classification;

if a system is faulty and the measurement takes a value; $X_{il} < \phi < X_{iu.}$ Then the discrimination function would incorrectly pass the system.

Hypothesis testing is a well known statistical technique which rejects or accepts a prior hypothesis about a sample, based on observations. In the case presented the Null Hypothesis, H_o, is that the system is fault-free and the alternative hypothesis, H_1, is that the system is faulty.

The measured value ϕ and the discrimination function $g(\phi)$, determine whether the null hypothesis is accepted or rejected, leading to four possible outcomes, Table 7.

	Fault Free	Faulty
Fault-Free System	✓	Type I error
Faulty System	Type II error	✓

Table 7 Probabilistic Outcomes

The two errors, Type I and Type II relate to the error in classification because of the probabilistic nature of actual system conditions based on a single test measurement, ϕ. A Type I error could be referred to as a Yield Loss and a Type II error as Test Escape.

Based on the confidence value (standard deviation) used to determine the ROA (the discrimination function) for the fault-free system response, then the probability of a fault-free system is known, since it is independent of the faulty response, Table 8.

Standard Deviation	Probability [%]
σ	68.26
2σ	95.44
3σ	99.74

Table 8 Confidence Values

In the general fault-free case:

$$PD_x = \int_{Xil}^{Xiu} p(\phi \mid G_x) d\phi$$

Equation 20 General Fault-Free Case [23]

In the general faulty case:

$$PD_x = \int_{-\infty}^{\infty} p(\phi \mid F_x)(1 - g(\phi))d\phi$$

Equation 21 General Faulty Case [23]

In the specific faulty case shown in Figure 106:

$$PD_x = \int_{-\infty}^{Xil} p(\phi \mid F_x)d\phi + \int_{Xiu}^{\infty} p(\phi \mid F_x)d\phi$$

Equation 22 Specific Faulty Case [23]

The general cases for the two errors:

Type I error **:** H_o is true but is rejected.

This corresponds to the case where a good system is failed. The probability of a type I error (α) can be calculated as:

$$\alpha = \int_{-\infty}^{\infty} p(\phi|G)(1 - g(\phi))d\phi$$

Equation 23 Type I Error [23]

Type II error **:** H_o is false but is accepted.

This corresponds to the case when a faulty system is passed. The probability associated with this (β_x) is:

$$\beta_x = \int_{-\infty}^{\infty} p(\phi \mid F_x)g(\phi)d\phi$$

Equation 24 Type II Error [23]

The fault-free case is only dependent upon the fault-free distribution and its association with the ROA as shown in Table 7. Conversely, the fault probability is dependent upon the fault distribution and its relationship to the ROA bounds. The type I error probability (α) is independent of the fault distribution and depends only on the probability distribution of the fault-free system and the discrimination function $g(\phi)$. The type II error probability (β) provides no additional confidence measure as it is equal to 1-α.

In this chapter the probabilistic hypotheses are used as a quality metric to assess the quality of each applied test for each given fault condition.

6.4 Simulation Data Prior to Test Analysis

The probabilistic analysis (test analysis) algorithm utilizes the simulation test data from the fault simulation algorithm as a post-processing algorithm to generate the probabilistic test metrics. Furthermore, an hydrodynamic "Y" channel case study is presented, demonstrating the complete workflow. Measurement data generated and stored from the simulation algorithm is organized into fault-free and faulty structures, to facilitate easier manipulation and search of data in the test analysis phase. Figure 107 shows this implementation.

Fault Free Structures

```
FaultFree = struct('TestPoint',{},'Measurements',[]);

FFImpedanceMagFreqGrp = struct('Frequency', [], 'Magnitude', []);
FFImpedancePhiFreqGrp = struct('Frequency', [], 'Phase', []);
```

Fault Data Structures

```
FaultData = struct('FaultCondition',{},'TestPoint',{},'Measurements',[]);

FImpedanceMagFreqGrp = struct('Frequency', [], 'Magnitude', []);
FImpedancePhiFreqGrp = struct('Frequency', [], 'Phase', []);
```

Figure 107 Data Structure Coding

In each structure the nominal simulations are grouped in accordance to the "TestPoint", where "TestPoint" refers to a system measurement, but is termed test because both functional sensors and dedicated test sensors form part of the systems test analysis. A further grouping is designated in the faulty structrure, the "FaultCondition". Data is organized by fault condition and then by test point, grouping all nominal simulation data.

```
>> FaultFree(1)

ans =

       TestPoint: {'Outlet Velocity (v)'}

       Measurements: [0.0014 0.0012 0.0013]

>> FaultData(1)

ans =

       FaultCondition: {'Inlet Left Blockage'}

       TestPoint: {'Outlet Velocity (v)'}

       Measurements: [9.2428e-004 9.7977e-004 7.8556e-004]
```

Figure 108 Data Structure Search Results

An example of how the data structure may be used to visually analyse data may be found in Figure 108. Impedance spectroscopy data is stored in its own dedicated structure, this is due to the amount of data initially returned from a frequency sweep per fault condition. The storage of impedance data is revisited in Figure 109.

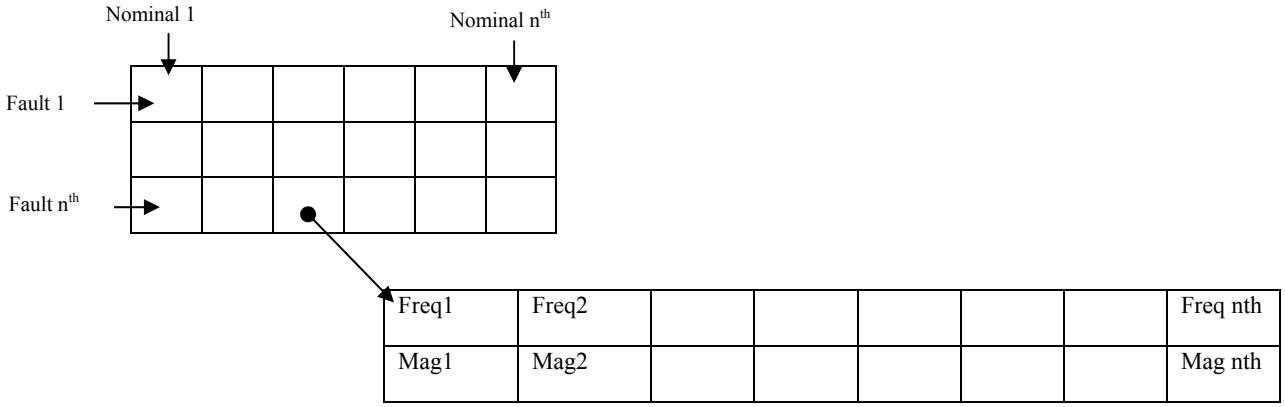


Figure 109 Simulation Impedance Data Storage

Impedance data is re-structured into frequency groups for test analysis purposes (the previous frequency sweeps are broken down). Each fault condition has f frequency groups, where f is the number of frequencies in the sweep. Each frequency group contains all the nominal magnitudes or phases (depending on group) for each simulation run for that fault condition, Figure 110.

```
FImpedanceMag1FreqGrp  = struct('Magnitude', []);
FImpedancePhi1FreqGrp  = struct('Phase', []);

for fault = 1:9
    for x = 1:10      %frequency
        for j = 1:10  %nominal
            if x > length(ImpedanceMag1(fault,j).y)

                disp('Nan')

            else

                FImpedanceMag1FreqGrp(fault,x,1).Magnitude(j) =
            ImpedanceMag1(fault,j).y(x);
                FImpedancePhi1FreqGrp(fault,x,1).Phase(j) =
            ImpedancePhi1(fault,j).y(x);

            end;
        end;
    end;
end;
```

Figure 110 Impedance Frequency Group Structure

6.5 Simulation Runs

Simulation time is computational expensive using the FEM approach, therefore the minimum number of system simulation iterations is desired when using a Monte Carlo approach. In the later test analysis section (6.6) the Guassian response curve for the faulty and fault-free system is used. Due to the computational expense of multiple simulation runs using the Monte Carlo method, the minimum number of simulation runs is desired without adversely affecting the measurement data and ultimately perturbing curve fitting. In contrast to the computational expense, the higher the sample density (the more simulations) the more accurate the curve fitting. Curve fitting "Goodness" of fit metrics allow the determination of the optimal number of simulation runs to be investigated.

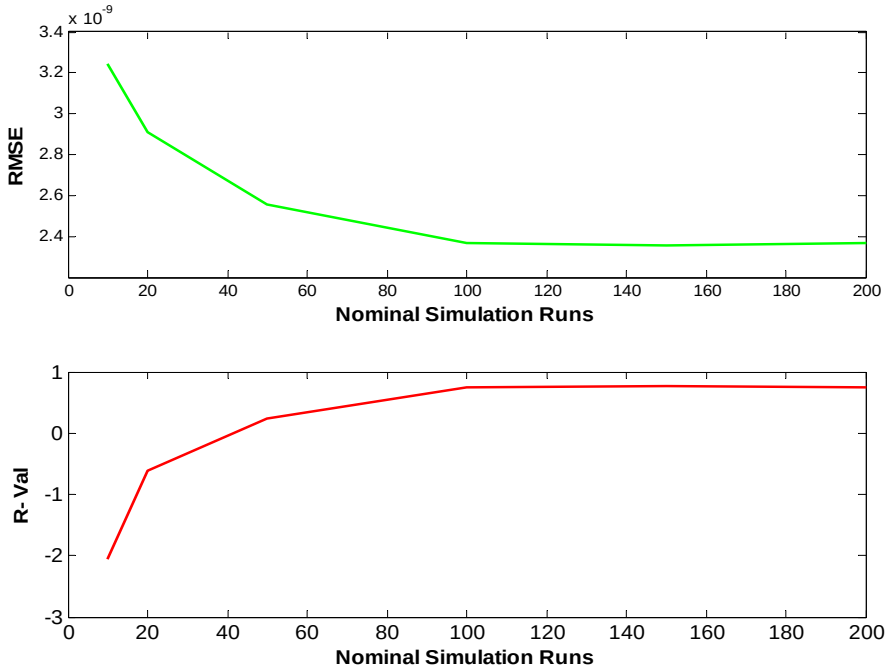


Figure 111 Curve Fitting Accuracy using Nominal Simulation

It is undesirable to perform excessive simulation runs; therefore, one may ask; what is the minimum number of runs which yield enough data for accurate curve fitting whilst keeping computational overhead to a minimum. Chapter 3 demonstrated the accuracy of the FEM approach, and it is this accuracy which forms the basis of the SbT methodology presented in this thesis, therefore it is important that the data produced by an accurate system simulation, does not mitigate accurate Test Analysis by poor curve fitting.

The impedance simulation data (magnitude) of a simple channel is analyzed through the curve fitting metrics; Root Mean Squared Error (RMSE), Equation 25 and R-value for increasing simulation runs, shown in Figure 111. The RMSE is the standard error of the regression with a value range of between 0 and 1, where a value approaching 0 is indicative of a more useful prediction. From Figure 111 the parameter RMSE is the most informative and converges on $2.5x10^{-9}$ around 100 simulation runs, indicating that further runs do not improve the accuracy of the curve fit and only contribute to computational overhead.

$$RMSE = \sqrt{MSE}$$

Equation 25 Root Mean Squared Error

The Residual, R value provides further evidence that 100 simulation cycles provides a "Good enough" fit between simulated and estimated data. The R value indicates the best predication terms when approaching 1 with negative values indicating terms which do not help the data fit, from Figure 111 a negative value exist from nominal simulations 1 through to approximately 60.

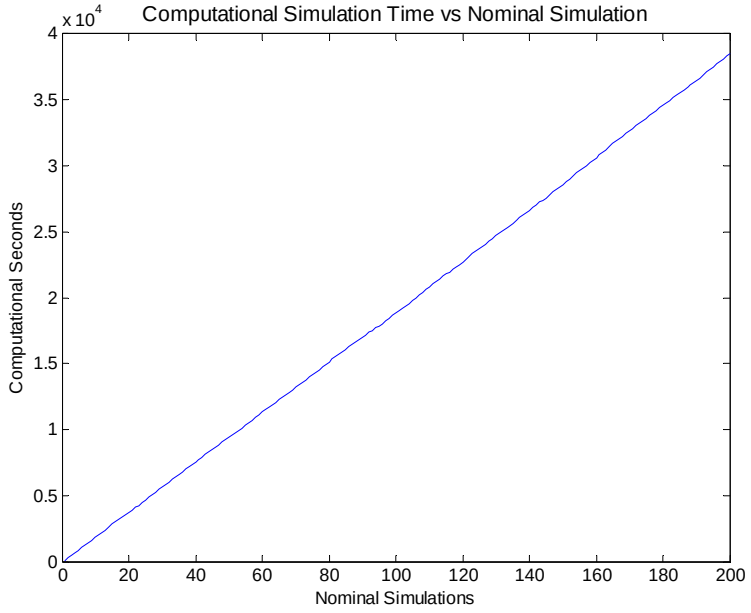


Figure 112 Computation Time

The simulation model for this analysis was a rectangular fluidic channel having a free mesh with 4023 degrees of freedom, over a single application layer. All simulations presented in this thesis are run on a workstation having XP Professional Service Pack 3, AMD Phenom Quad Core 2.2GHz Processor, 3.25GB RAM. The computational cost figure, Figure 112 shows how simulation time is directly proportional to the number of simulation runs.

6.6 Test Analysis Procedure

The aim of the test analysis procedure is to analyse simulation measurements to produce test hypotheses to determine the effectiveness of the applied test method(s) for a given system, as explained in the introduction to this chapter. The test analysis algorithm is given in Figure 113.

The input data to the algorithm is the fault-free and faulty measurement structures described earlier in this section 6.4.

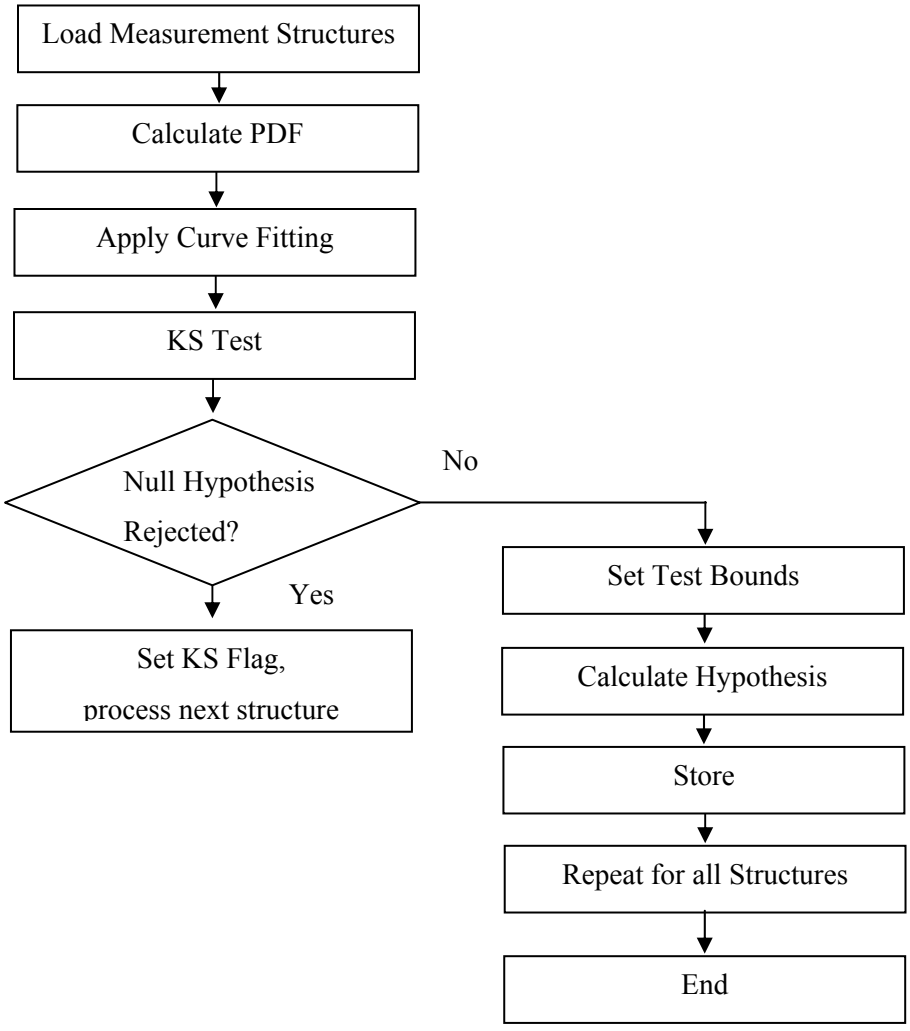


Figure 113 Test Analysis Algorithm

6.6.1 Calculate PDF & Curve Fitting

The aim of this step of the algorithm is to determine the PDF (probability density function) for the fault-free and faulty simulated (measured) data and then find the best fit PDF function. Two methods are explored:

1) The first method directly uses the measurement data and the *normpdf* function to calculate the PDF response for the limited sample set of measurements, Figure 114.

```
PDFPostFaultFFData = setfield(PDFPostFaultFFData, {i},
'PostFaultFFMeas', normpdf(sort(FaultFree(i).Measurements),
mean(sort(FaultFree(i).Measurements)),std(FaultFree(i).Measurements)));
```

Figure 114 PDF Code Example

The MATLAB *normpdf* function has prototypes of the measurements data (the distribution), the mean measurement data and the standard deviation of the measurement data.

Gaussian curve fitting is then applied to this response, Figure 115. The assumption that the measurement data is Gaussian arises from the use of Monte Carlo analysis to apply system variance, therefore, the assumption is made that the data is normally distributed, unless non-linearities are introduced. This assumption is tested in the next step of the algorithm.

```
% Single Gaussian Fit assumption
f = fittype('gauss1');

% Fault-Free Impedance Fitting
cFF = fit(sort(FFImpedanceMag1FreqGrp(frequency,1).Magnitude)',
PDFPostFFImp1MagFreqGrp(frequency,1).PostFaultMeas',f);

% Saving the Fit Coefficients
FFcoeff = coeffvalues(cFF);

% Producing Fitted Response
FFGaussianImp1Mag(frequency,1).FFGaussY = FFcoeff(1)*exp(-
((FFGaussianImp1Mag(frequency,1).FFGaussX-
FFcoeff(2)).^2)/(2*FFcoeff(3).^2));
```

Figure 115 Curve Fitting Code Example

Since the measurement data is range limited (simulation runs), then so is the fitted response between the measurement data x bounds, therefore the response may not reach its asymptotes. Asymptotes are required for later response integration in order to determine the probability between two points. In order to facilitate this data is extrapolated between multiples *sigma* (standard deviation), Figure 116. Examples of poor extrapolation may be found in Figure 117.

The multiple may be arbitrarily chosen, or based around a system such as a quality system (six-sigma) or related to a manufacturing processes tolerance, for example, three-sigma [22].

```
%Create high resolution x range
minofrange = (-sigma*std(FaultFree(x).Measurements)+
mean(FaultFree(x).Measurements));
maxofrange = (sigma*std(FaultFree(x).Measurements) +
mean(FaultFree(x).Measurements));
resofrange = (abs(minofrange)+ maxofrange)/1000;

FFData(x).FFGaussX = minofrange: resofrange: maxofrange;
```

Figure 116 Extrapolation Code

This method has the advantage of using the measured data to find the probability density function directly; however, a major disadvantage is the inaccuracy after extrapolation. The *normpdf* function weights the PDF response based on the limited distribution data originally provided. Therefore, when the PDF fitted line function is extrapolated beyond its original bounds the integrated area is greater than 1 (the maximum probability). Furthermore, the fitted line function was determined over the limited range, therefore the coefficients were generated to describe accurately the response to the limited data, not the extrapolated data range.

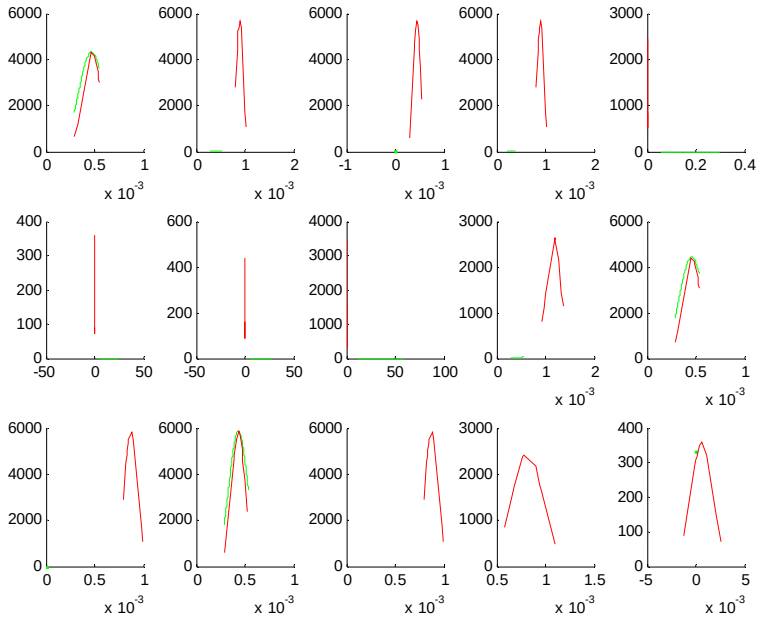


Figure 117 several examples of poor extrapolation

2) The alternative method uses the *norminv* and *normpdf* functions to generate the ideal Gaussian response (including asymptotes) based on the measured data mean and standard deviation. The *norminv* function is used to find the inverse normal distribution based on a high resolution probability distribution, for the mean and standard deviation of the measured data. This distribution is then used in the *normpdf*, Figure 118. The *norminv* function describes the probability over an ideal range; because this forms the x value argument input of the function.

```
    prob = (0.0002:0.0004:0.9998)';
    xFF =
norminv(prob,mean(FaultFree(x).Measurements),std(FaultFree(x).Measurements));
    yFF =
normpdf(xFF,mean(FaultFree(x).Measurements),std(FaultFree(x).Measurements));
```

Figure 118 Method 2: norminv and normpdf

Curve fitting this then simplified, Figure 119, removing the need for extrapolation and having a line function whose coefficients describe the response correctly weighted for the PDF to the asymptotes. The disadvantage is that the assumption (from which everything is based) that the measured data is Gaussian, this is tested next using a KS-Test.

```
FFDatacFF = fit(xFF, yFF,f);
FFDatacoeff = coeffvalues(FFDatacFF);
```

Figure 119 Method 2: Curve Fitting

Then complete "extrapolated" responses are formed, Figure 120.

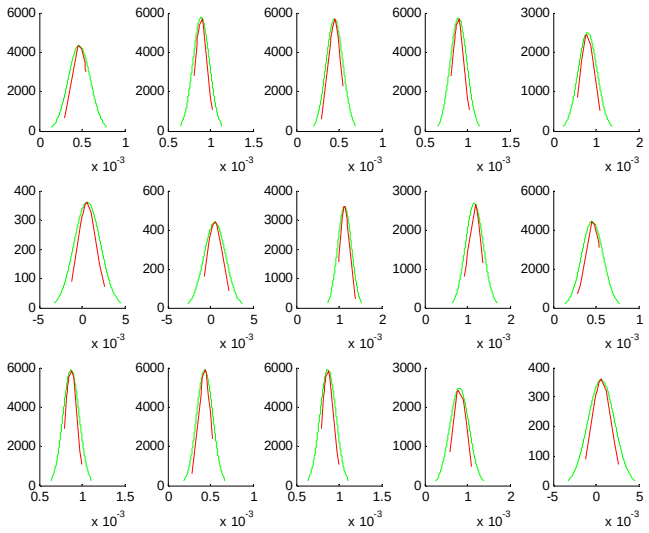


Figure 120 several examples of improved extrapolated responses

6.6.2 KS (Kolmogorov-Smirnov) Test

Earlier in this chapter the optimum number of nominal simulations was estimated to be 100 determined by the curve fitting metrics for this type of simulation and desired accuracy, RMSE and R value. The assumption was made that the measured data is Gaussian, based on the nominal simulation inputs being normally distributed. This was an "offline" observation, made on a limited set of data. A dynamic test is required to determine if the assumption to fit a Gaussian response is valid before analysis is made on the fitted response, for this the KS-Test is performed.

To gain this knowledge a Gaussian fit is made to the simulation data (previous step) and then the KS-Test performed to determine if the hypothesis that the simulated data has Gaussian properties i.e. do the measured data and the predicted data fit.

A two-sided KS test function is provided in MATLAB, Figure 121. Let X be the cumulative distribution function for the simulation data measurements and F be the hypothesized distribution function with a normal distribution where the mean and standard deviation is based on the measured data. Where *alpha* is the desired significance level. A 95% significance level is used throughout this research (alpha = 0.05).

[H, P, KS_Stat, cv] = kstest2(X,F,alpha)

Figure 121 MATLAB KS Function

The function returns the hypothesized result, H (reject / accept), the p-value and the KS test statistic and cv the cut-off value to determine if KS_Stat is significant. The two sided KS test statistic KS_Stat is defined as the greatest distance between X and F, Equation 26.

$$T = \max|F - X|$$

Equation 26 KS Test Statistic

The null hypothesis that X and F are from the same distribution is rejected if T is greater than the tabulated KS Test Statistic value for the given significance level. The null hypothesis will be rejected should the p-value equal or exceed the significance level, *alpha.*

An example of a KS Test is shown for an impedance measurement, Figure 122 (both responses are from a limited data set, therefore, assymptotes do not approach zero). The KS statistics, T is 0.325; the cut-off value for the given significance level is 0.535. Therefore, in this case the null hypothesis is not rejected. In the test analysis algorithm MATLAB provides this analysis and only the H parameter

(hypothesis reject or accept) is monitored. If in the test analysis algorithm a simulation run fails to pass this test, a flag is set and no further processing occurs since determining test metric data based on an ill fitting response could invalidate the test scheme.

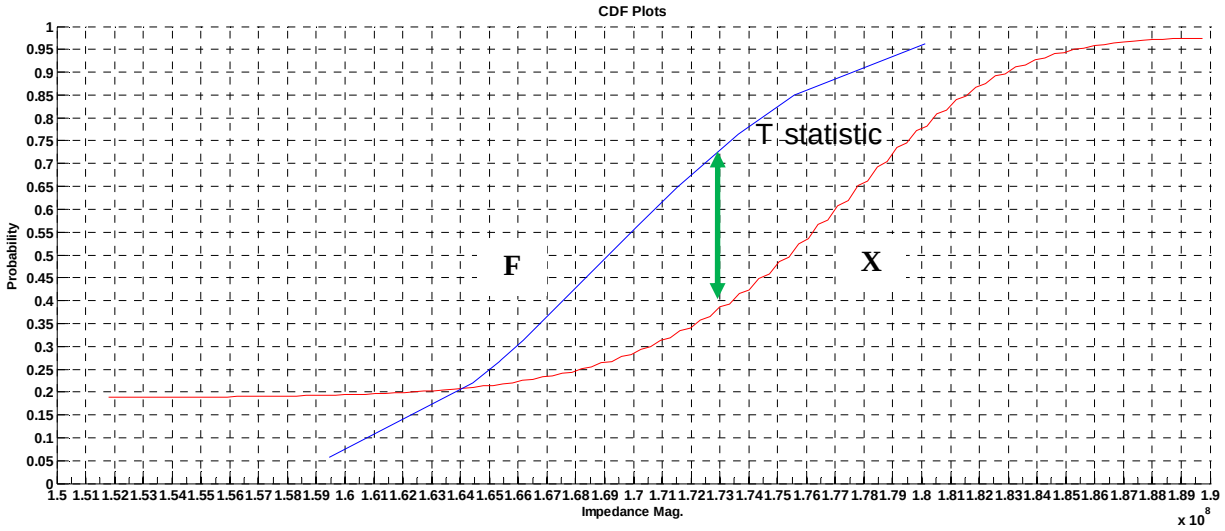


Figure 122 KS Testing an Impedance Measurement

6.6.3 Region of Acceptability (Test Bounds)

A common selection in the literature is 3σ to determine the region of acceptability. The ROA value of 3σ will be used throughout this thesis, this yields 99.74% fault-free probability.

```
ROAmin = (-3*std(FFData(x).FFGaussX)) + mean(FFData(x).FFGaussX);
ROAmax = (3*std(FFData(x).FFGaussX))  + mean(FFData(x).FFGaussX);
```

Figure 123 Region of Acceptability Code

6.6.4 Hypothesis

At this stage of the algorithm it is known that the data approximates a Gaussian with a sufficient level of confidence. The fitted curves for p(ϕ|G) and p(ϕ|F) along with a region of acceptability are available. Therefore, the test hypotheses may be calculated. Integration may be performed on the PDF functions to determine the four probability outcomes using the four equations given at the beginning of this chapter, Table 9. The code for each outcome is shown in Figure 124 to Figure 127.

The code shown in Figure 124 describes how the fault-free system probability is calculated and stored. The fault condition and measurement type are stored in the structure. The ROA's are determined and the line equation for the fault-free line is integrated between the upper and lower ROA bounds.

```
FF_FFData(fault).FaultCondition = FaultData(fault,x).FaultCondition;
FF_FFData(fault).TestPoint(x) = FaultData(fault,x).TestPoint;

ROAmin = (-3*std(FFData(x).FFGaussX)) + mean(FFData(x).FFGaussX);
ROAmax = (3*std(FFData(x).FFGaussX)) + mean(FFData(x).FFGaussX);

F = @(x1) (FFData(x).FFCoeff(1)*exp(-((x1-
FFData(x).FFCoeff(2))/FFData(x).FFCoeff(3)).^2));

Q = quad(F,ROAmin,ROAmax);
FF_FFData(fault).Probability(x) = abs(Q);
```

Figure 124 Test Hypothese Code: Fault-Free

The type I error is calculated as shown in Figure 125, the fault-free line function is used with two integrations performed, one between the lower ROA and the lower limit of the response and the second between the upper ROA and upper limit of response.

```
FF_FData(fault).FaultCondition = FaultData(fault,x).FaultCondition;
FF_FData(fault).TestPoint(x) = FaultData(fault,x).TestPoint;

F = @(x1) (FFData(x).FFCoeff(1)*exp(-((x1-
FFData(x).FFCoeff(2))/FFData(x).FFCoeff(3)).^2));
Q1 = quad(F,min(FFData(x).FFGaussX),ROAmin);
Q2 = quad(F,ROAmax,max(FFData(x).FFGaussX));
FF_FData(fault).Probability(x) = abs(Q1 + Q2);
```

Figure 125 Test Hypothese Code: Type I error

The faulty system hypothesis, Figure 126 uses the faulty response and is integrated between the upper ROA and the upper faulty response limit and a between the lower ROA and lower faulty response limit.

```
F_FData(fault).FaultCondition = FaultData(fault,x).FaultCondition;
F_FData(fault).TestPoint(x) = FaultData(fault,x).TestPoint;

F = @(x1) (FData(fault,x).FCoeff(1)*exp(-((x1-
FData(fault,x).FCoeff(2))/FData(fault,x).FCoeff(3)).^2));
Q1 = quad(F,ROAmax,max(FData(fault,x).FGaussX));
Q2 = quad(F,min(FData(fault,x).FGaussX),ROAmin);
F_FData(fault).Probability(x) = abs(Q1+Q2);
```

Figure 126 Test Hypothese Code: Faulty

The type II error is calculated using the faulty response, between the upper and lower ROA limits.

```
F_FFData(fault).FaultCondition = FaultData(fault,x).FaultCondition;
F_FFData(fault).TestPoint(x) = FaultData(fault,x).TestPoint;

F = @(x1) (FData(fault,x).FCoeff(1)*exp(-((x1-
FData(fault,x).FCoeff(2))/FData(fault,x).FCoeff(3)).^2));
Q2 = quad(F,ROAmin, ROAmax);
F_FFData(fault).Probability(x) = abs(Q2);
```

Figure 127 Test Hypothese Code: Type II error

This process is repeated for all measured data and fault conditions. Impedance spectroscopy magnitude and phase measurements follow the same procedure albeit using different nomenclature.

6.6.5 Test Analysis Storage

For each fault condition and test point the probabilities of each test outcome are stored, Table .

	Fault Free	Faulty
Fault-Free System	`FF_FF(frequency).Probability`	`FF_F(frequency).Probability`
Faulty System	`F_FF(frequency). Probability`	`F_F(frequency).Probability`

Table 9 Probabilistic Outcomes Code

From these outcomes, decisions about the effectiveness of the deployed tests can be made, in order to determine the most efficient tests and test locations and the overall fault coverage of the system.

6.7 Test Assessment

Each test for each fault condition can be assessed to find the “best” performing test for each fault condition, achieved through a trivial search of the outcome probability structures, Table 9. To find those tests (and locations) which yield the highest fault-free and faulty outcomes and the lowest test escape and yield loss errors.

Furthermore, Total Fault Coverage, (TFC) [19] of the system may then be determined by summing the most discriminatory test for each fault condition represented to the system. If T = {T_1, T_2, …

$T_n\}$ represents a set of n tests and F = $\{F_1, F_2 \ldots F_m\}$ a set of m fault conditions, then to determine FC of the test set T the sum of the best performing FDP functions, Equation 27.

$$FC = \frac{\sum_{j=1}^{m} Max\{FDP_{i,j} | T_i \in T\}}{m}$$

Equation 27 Fault Coverage Equation

where $FDPi_{,}j$ represents the test probability, T_i for a given fault condition, F_j and $Max\{FDP_{i,j} | T_i \in T\}$ represents the best test probability for the given fault condition.

6.8 Adaption of Algorithm for Statistical Plots

The test analysis algorithm plots the measurement data probabilistic fault-free and faulty test outcomes for visual analysis, alongside populating the outcome structures, Table 9. A window, shown in Figure 128 is produced for each fault condition. This is not mandatory to the analysis process but provides a visual check to the researcher that the simulation data and calculated test outcome is sensible.

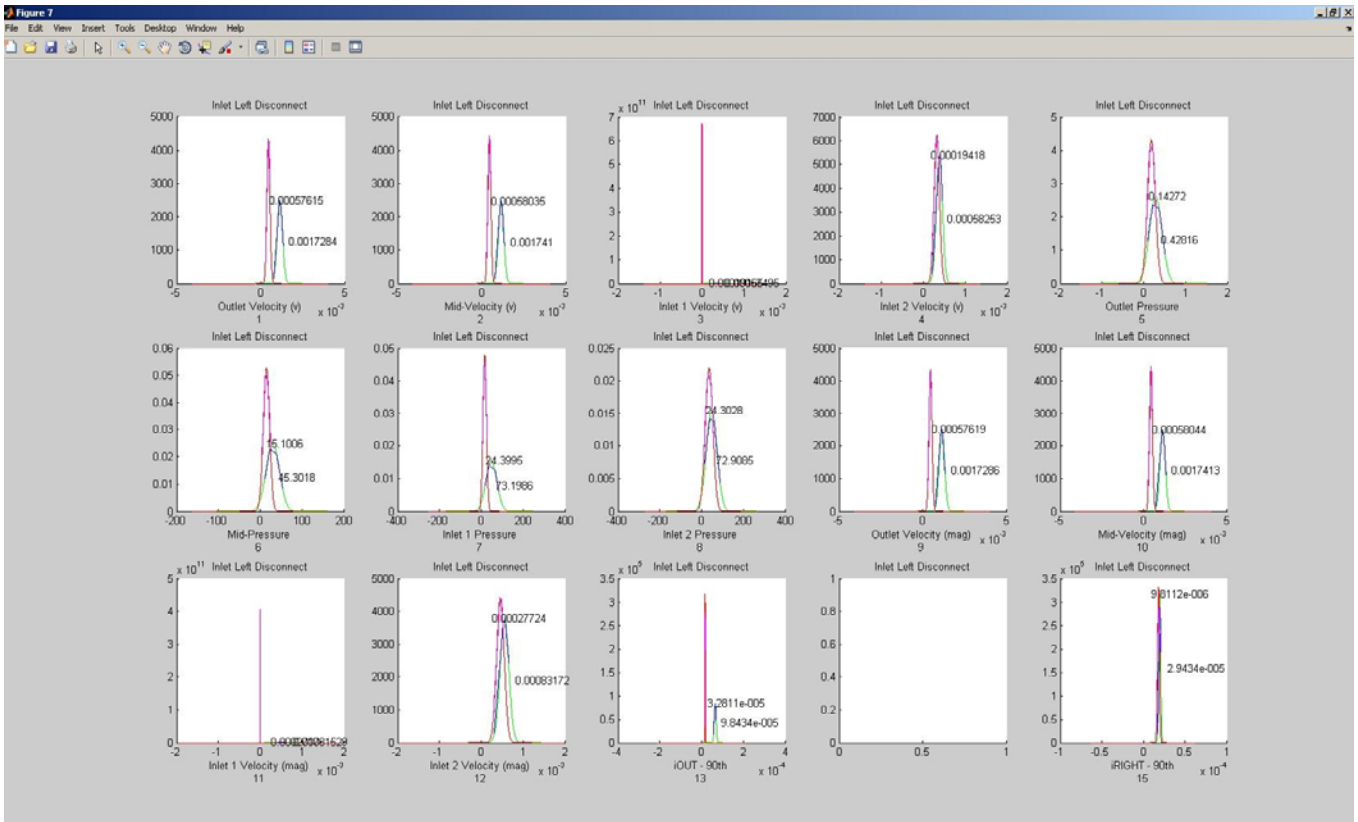


Figure 128 Probabilistic Outcome Plot

6.9 "Y" Channel Application Case Study

6.9.1 Introduction

This is a simple "Y" channel hydrodynamic diffusional mixing system. The purpose of this case study is to demonstrate the design, simulation and test methodology, and in particularly to demonstrate the operation of the previously described test analysis procedure in order to determine the most effective test method and locations for this system.

6.9.2 General System Simulation Description: Hydrodynamic "Y" Channel

The "Y" channel model comprises the physical geometry shown in Figure 129. The geometry is based on a work by Sun *et. al* [24], but uses hydrodynamic flow and the reactor in reverse. The channel width is 100µm; height 40 µm; channel lengths 5mm. The inlet channels are 45° to the main channel.

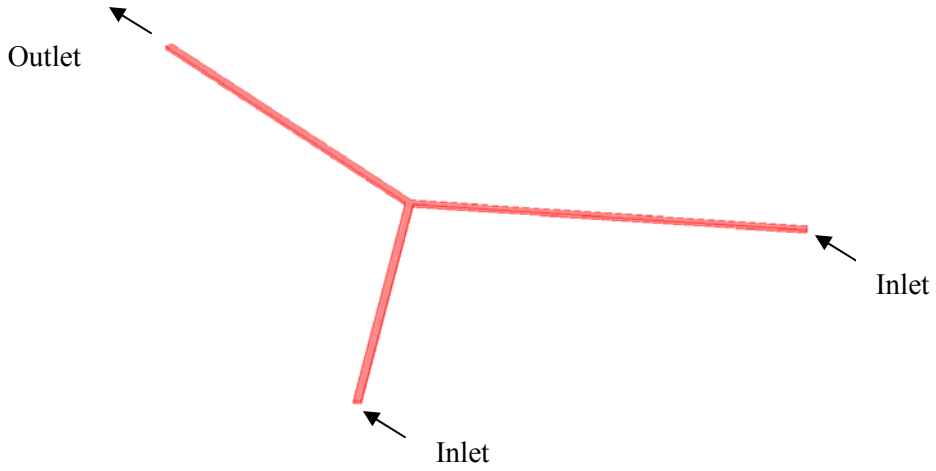


Figure 129 "Y" Channel Geometry

The functionality of the application is based on hydrodynamic flow to mix two analytes through diffusion. The two arms of the "Y" form inlets where a positive hydrodynamic flow is present and analytes with known concentrations are introduced. The outlet is at the end of the main channel, where the mixed fluid emerges.

This system simulation comprises three applications modes, (see Table 10). The ruling application mode is Stokes Flow (mmglf) which describes the fluid flow within the channels from the inlets to the outlets. The Convection and Diffusion (chdh) is coupled to the Stokes flow application mode to describe the diffusion of the analytes. The test application mode for impedance spectroscopy (emqvw) is the electric currents application mode, this application mode uses a parametric solver.

Application Mode #	Description	Abbreviation
1	Stokes Flow	mmglf2
2	Convection and Diffusion	chcd
3	Electric Currents	emqvw

Table 10 Application Modes

Sodium Hydroxide (NaOH) having density (rho(T[1/K])[kg/m^3]) and viscosity (eta(T[1/K])[Pa*s]); 1000[kg/m^3] and 1e-3[Pa*s], respectively is used as the fluid. The temperature dependency of solution is modeled, T. Diffusion is described by the diffusion coefficient (9.1e-6[cm^2/s]) and the convection governed from the velocity field, u2, v2 and w2 where the nominal flow rate is 320μm/s. To aid the convergence of inconsistent initial values a flc1hs(t[1/s]-0.1,10) smoothed Heaviside function with a continuous first derivative without overshoot is used on the inlet velocities, Equation 28.

Velocity Inlet: Velo1*flc1hs(t[1/s]-0.1,10)

Equation 28 Velocity Heaviside Expression

The right, c0 and left, c1 inlets have concentrations of 0.8[mol/kg] and 0.2[mol/kg], respectively. The analysis type (solver) was Transient for both the mmglf2 and chcd application modes using a direct SPOOLES solver. The simulation time range was 0:1:100 seconds. The two application modes were solved concurrently using the initial value expressions, and the electric currents application mode solved afterwards using the current solution settings.

6.10 Test Simulation Description: Hydrodynamic "Y" Channel

Impedance spectroscopy and Levich sensors will be assessed as test methods for this system using the simulation described. The implementation of impedance spectroscopy has previously been described using the electric currents application mode and a parametric solver. The solver is implemented to sweep a frequency range from 100Hz to 990 kHz. The electric currents application mode allowed electrodes to be modeled at the positions shown in Figure 130.

Custom simulation script may be found in Appendix II which describes the inlet and outlet boundary conditions in the electric current application mode by switching the source (forced current) between B and C while maintaining A as the sink (ground) electrode. As was the case for the Sun model, however here the boundary condition describes the impedance of the electrodes. When B is the source, C is assigned electric insulation and vice versa. Each arrangement is modeled per simulation cycle. The two impedance measurement paths are shown in Figure 130 indicated by the two different line types.

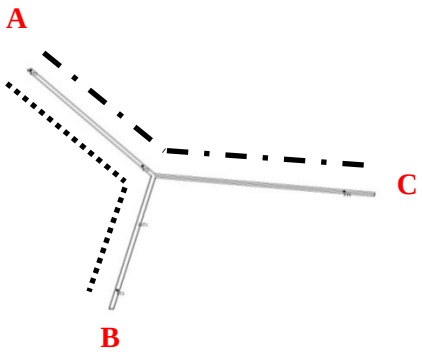


Figure 130 Impedance Spectroscopy Electrode Configuration

In the equivalent experimental system the measured impedance would be sensitive to the conductivity of the fluid in the channels, as well as being sensitive to fault conditions. Since conductivity of the fluid is related to the concentration of the fluid then this coupling or relationship has to be made explicitly in the simulation environment Equation 29.

```
AveConcL = abs(((ConcO+ConcM+ConcL)/3)/1000)

AveConcR = abs(((ConcO+ConcM+ConcR)/3)/1000)

conductivityRAWL = AveConcL*Inf_dilution*activity_coeff

conductivityRAWR = AveConcR*Inf_dilution*activity_coeff

conductivityL = 1e-9+(conductivityRAWL*flc1hs(t-0.1,1))

conductivityR = 1e-9+(conductivityRAWR*flc1hs(t-0.1,1))
```

Equation 29 Conductivity Modification

This simulation geometry comprises three sub-domains (main and two inlet channels), each specifying an individual conductivity, Equation 29 shows how the two inlet conductivities are calculated.

The conductivity is based on the simulated concentration and is calculated for right and left channels. An arbitrary point at the outlet, in the main, left and right channels evaluate the concentration, ConcO, ConcM, ConcL and ConcR, respectively per simulation step.

For example, AveConcL approximates the rough average for the path from the outlet to the left inlet. This is the then multiplied by the infinite dilution and activity coefficient to determine the conductivity contributed to the system due to the analyte concentration, conductivityRAWL, this is then added to the base conductivity to produce conductivityL.

Three Levich sensors are implemented in this system; one at the outlet, one in the right inlet and one in the left inlet. These are implemented as global expressions as previously described in Chapter 4 to determine the mass-transport current for each position, Equation 30. The flow rate values; FlowL, FlowR and FlowO are calculated using boundary integration on the respective boundaries and are made available to the global expressions when required.

iLeft = (0.925*n*F*ConcL*(D^(2/3))*(FlowL^(1/3))*w1*((xe^2)/(h1^2*d1))^(1/3))

iRight = (0.925*n*F*ConcR*(D^(2/3))*(FlowR^(1/3))*w1*((xe^2)/(h1^2*d1))^(1/3)

iOut = (0.925*n*F*ConcO*(D^(2/3))*(FlowO^(1/3))*w1*((xe^2)/(h1^2*d1))^(1/3))

Equation 30 Levich Sensor Equations

6.10.1 Post-Processing

The left inlet shows a concentration of 0.8M and the right inlet shows a concentration of 0.2M. With equal flow rates 320μm/s and an atmospheric pressure at the outlet then the perfectly diffused profile is approximately 0.5M.

Performing boundary integration on the outlet boundary 7 shows a flow rate of 0.00146 [uL/s]. The flow rate at each inlet is given as 6.033978e-4 [uL/s], these parameters form the specification of the system.

6.10.2 Fault Dictionary

Faults were selected for implementation in this system from the available fault library described in Chapter 4. Fault selection was not based on any prior probability, since such statistics are unknown. Therefore, all faults are injected with equal probability, and at pre-determined locations. The MFS script for this "Y" case can be found in Appendix II. This actions the fault-free nominal simulation runs, the fault injections and the recorded measurement data.

In this "Y" channel system 9 fault conditions were injected, described in Table 11, along with their nominal parameter range where applicable.

Fault Number	Parameter	Nominal Value	Fault Conditions
1	Right Inlet Concentration	0.8μM +/- 1%	+/- 20%
2	Left Inlet Concentration	0.2μM +/- 1%	+/- 20%
3	Right Inlet Velocity	320μm/s +/- 20%	Pump Fail +/- 200%
4			Disconnect – No Flow (capacitance)
5			Partial Blockage
6	Left Inlet Velocity	320μm/s +/- 20%	Pump Fail +/- 200%
7			Disconnect – No Flow (capacitance)
8			Partial Blockage
9	System Temperature	297 K +/- 10%	

Table 11 "Y" Channel Fault Dictionary

6.10.3 Test Analysis

Fault simulation and subsequent test analysis was performed on the system in accordance with the MFS method described in Chapter 5 and test analysis described earlier in this chapter, the test analysis script for this "Y" channel case can be found Appendix III.

Test outcomes were calculated for each fault condition for functional and Levich sensors, Table 11. Each probability (1-15) refers to a particular measurement given in the footer of the table.

Analysis of the results in Table 12 shows that all sensors (functional and Levich) have a high probability of determining a fault-free system as fault-free with a small Type I error. Of particular interest is the determination of a faulty system as being faulty, therefore, those sensors yielding a high faulty as faulty probability with a small Type II error demonstrates their potential strength as a test sensor.

The NaN value produced for the Left and Right Inlet Disconnect fault conditions indicated that a probability could not be determined. The reason is the measured data returned a single non-zero value for each nominal simulation i.e. there was no variation and therefore no distribution. Therefore, the PDF calculation and subsequent integration returned non-numeric result. A NaN in this case is indicative of a probability of 1, therefore demonstrating a strong test sensor.

System blockages are shown to be generally erroneously classified when considering functional sensors, however, it is later demonstrated that impedance spectroscopy can be used to detect these conditions. A contributing factor to low hypothese acceptance could be that generally in the fault simulation, blockage conditions were positoned between sensing points.

Results for the Levich test sensors provide tests which often satisfy the null hypothesis, H_1 that the faulty system is faulty, but their detection ability is sensitive to the position within the system. For example, for the fault condition left inlet disconnect, the iOUT and iLEFT Levich sensors show a high probability (0.9979, NaN), respectively, whereas, the iRIGHT Levich strongly rejects the H_1 hypotheses, producing a Type II error (0.8268). A similar pattern may be observed for the right inlet disconnect fault. Furthermore, Levich sensors are shown to be very sensitive to concentration variation, whereas, all functional sensors demonstrate high Type II error probabilities.

Table 12 presents the probabilistic outcomes for the full range of functional sensors. The impedance magnitude and phase results are presented in tables Table 13 to Table 16.

	Fault-Free as Fault-Free	Faulty as Faulty	Yield Loss	Test Escape
1 (Inlet Left Disconnect)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9957 0.9957 0.8258	0.5353 0.5942 0.0000 0.0004 0.0013 0.0003 0.0003 0.0019 0.5351 0.5945 0.0000 0.0004 0.9979 NaN 0.0000	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0022 0.0022 0.0001	0.4642 0.4054 0 1.0000 0.9983 0.9993 0.9999 0.9977 0.4645 0.4051 0 1.0000 0.0000 0.0000 0.8268
2(Inlet Left Blockage)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9957 0.9957 0.8258	0.0004 0.0004 0.0000 0.0004 0.0038 0.0014 0.0022 0.0013 0.0004 0.0004 0.0002 0.0004 0.0002 0.0014 0.0000	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0022 0.0022 0.0001	1.0000 1.0000 0.9996 1.0000 0.9958 0.9982 0.9974 0.9983 1.0000 1.0000 0.9994 1.0000 0.9628 0.9617 0.8294
3(Inlet Right Disconnect)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9957 0.9957 0.8258	0.6577 0.7585 0.0018 0.0000 0.0002 0.0003 0.0009 0.0003 0.6571 0.7582 0.0019 0.0000 0.6498 0.0013 NaN	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0022 0.0022 0.0001	0.3419 0.2411 0.9978 0 0.9994 0.9993 0.9987 0.9999 0.3425 0.2414 0.9977 0 0.3498 0.9616 0.0000
4(Inlet Right Blockage)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9957 0.9957 0.8258	0.0004 0.0004 0.0003 0.0004 0.0053 0.0019 0.0025 0.0016 0.0004 0.0004 0.0003 0.0004 0.0026 0.0008 0.0000	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0022 0.0022 0.0001	1.0000 1.0000 0.9993 1.0000 0.9943 0.9977 0.9971 0.9980 1.0000 1.0000 0.9993 1.0000 0.9954 0.9623 0.8292
5(Outlet Blockage)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9957 0.9957 0.8258	0.0193 0.0191 0.0159 0.0000 0.0562 0.1677 0.1095 0.1073 0.0192 0.0191 0.0161 0.0001 0.5300 0.0333 0.0001	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0022 0.0022 0.0001	0.9803 0.9805 0.9837 0.9996 0.9434 0.8319 0.8901 0.8922 0.9804 0.9805 0.9835 0.9995 0.4696 0.9647 0.8269
6(Right Pump Fail)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9957 0.9957 0.8258	0.5984 0.5984 0.0099 0.7092 0.3396 0.3396 0.2268 0.4270 0.4825 0.4825 0.0099 0.5378 0.6370 0.0017 0.8646	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0022 0.0022 0.0001	0.4012 0.4012 0.9897 0.2904 0.6600 0.6600 0.7727 0.5712 0.5171 0.5171 0.9897 0.4618 0.3626 0.9963 0.1350
7(Left Pump Fail)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9957 0.9957 0.8258	0.5676 0.5676 0.8025 0.0001 0.1979 0.1979 0.3416 0.0450 0.3619 0.3619 0.6218 0.0001 0.6906 0.8085 0.0000	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0022 0.0022 0.0001	0.4320 0.4320 0.1971 0.9995 0.8017 0.8017 0.6579 0.9546 0.6377 0.6377 0.3778 0.9995 0.3090 0.1911 0.8262
8(Right Conc. Degradation)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9957 0.9957 0.8258	0.0004 0.0004 0.0003 0.0003 0.0015 0.0015 0.0022 0.0010 0.0004 0.0004 0.0003 0.0003 0.9608 0.0003 0.9996	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0022 0.0022 0.0001	1.0000 1.0000 0.9993 0.9999 0.9981 0.9981 0.9974 0.9986 1.0000 1.0000 0.9993 0.9999 0.0371 0.9977 0.0000
9(Left Conc. Degradation)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9957 0.9957 0.8258	0.0004 0.0004 0.0624 0.0004 0.0359 0.0359 0.0465 0.0261 0.0004 0.0004 0.0624 0.0004 0.9980 0.9980 0.0000	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0022 0.0022 0.0001	1.0000 1.0000 0.9372 1.0000 0.9637 0.9637 0.9531 0.9735 1.0000 1.0000 0.9372 1.0000 0.0000 0.0000 0.8281

'Outlet Velocity (v)' 'Mid-Velocity (v)' Inlet 1 Velocity (v) Inlet 2 Velocity (v) 'Outlet Pressure' 'Mid-Pressure' 'Inlet 1 Pressure' 'Inlet 2 Pressure' Outlet Velocity (mag) 'Mid-Velocity (mag)' Inlet 1 Velocity (mag) Inlet 2 Velocity (mag) 'iOUT - 90th' 'iLEFT - 90th' 'iRIGHT - 90th'

Table 12 Test Outcomes

6.10.3.1 Impedance Magnitude 1

	Fault-Free as Fault-Free	Faulty as Faulty	Yield Loss	Test Escape
1 (Inlet Left Disconnect)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	1.0e-005 *0.2097 0.1490 0.0639 0.0160 0.0022 0.0002 0.0000 0.0000 0.0000 0.0000
2(Inlet Left Blockage)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	1.0e-009 *0.6748 0.6745 0.6740 0.6731 0.6718 0.6703 0.6686 0.6667 0.6645 0.6623	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799
3(Inlet Right Disconnect)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.0643 0.0644 0.0644 0.0645 0.0646 0.0647 0.0648 0.0650 0.0652 0.0654	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	0.9155 0.9155 0.9155 0.9154 0.9153 0.9152 0.9150 0.9149 0.9147 0.9144
4(Inlet Right Blockage)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	1.0e-009 *0.2490 0.2485 0.2471 0.2450 0.2422 0.2387 0.2345 0.2297 0.2243 0.2184	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799
5(Outlet Blockage)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.2414 0.2413 0.2412 0.2409 0.2405 0.2400 0.2395 0.2388 0.2380 0.2372	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	0.7385 0.7385 0.7387 0.7390 0.7394 0.7398 0.7404 0.7411 0.7418 0.7427
6(Right Pump Fail)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.7464 0.7462 0.7457 0.7450 0.7440 0.7427 0.7412 0.7394 0.7374 0.7351	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	0.2335 0.2337 0.2342 0.2349 0.2359 0.2372 0.2387 0.2405 0.2425 0.2447
7(Left Pump Fail)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.9704 0.9503 0.9497 0.9486 0.9473 0.9456 0.9436 0.9413 0.9387 0.9359	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	0.0095 0.0296 0.0302 0.0312 0.0326 0.0343 0.0363 0.0386 0.0411 0.0440
8(Right Conc. Degradation)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.9750 0.9750 0.9750 0.9750 0.9751 0.9751 0.9752 0.9752 0.9753 0.9754	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	0.0049 0.0049 0.0049 0.0048 0.0048 0.0048 0.0047 0.0046 0.0046 0.0045
9(Left Conc. Degradation)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	1.0e-023 *0.3609 0.2927 0.1760 0.0785 0.0260 0.0064 0.0012 0.0002 0.0000 0.0000

Frequency = {1 2 3 4 5 6 7 8 9 10} = 10Khz, 60Khz, 110Khz, 160Khz, 210Khz, 260Khz, 310Khz, 360Khz, 410Khz, 460Khz

Table 13 Impedance Magntide 1 Test Outcomes

6.10.3.2 Impedance Magnitude 2

	Fault-Free as Fault-Free	Faulty as Faulty	Yield Loss	Test Escape
1 (Inlet Left Disconnect)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	1.0e-007 *0.9872 0.8840 0.6760 0.4414 0.2462 0.1173 0.0478 0.0167 0.0050 0.0013
2(Inlet Left Blockage)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	1.0e-003 *0.1183 0.1179 0.1171 0.1159 0.1142 0.1121 0.1096 0.1068 0.1037 0.1004	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	0.9798 0.9798 0.9798 0.9798 0.9798 0.9798 0.9798 0.9798 0.9798 0.9798
3(Inlet Right Disconnect)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.5236 0.5233 0.5224 0.5211 0.5193 0.5170 0.5143 0.5112 0.5077 0.5038	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	0.4562 0.4566 0.4574 0.4588 0.4606 0.4628 0.4655 0.4687 0.4722 0.4761
4(Inlet Right Blockage)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.0093 0.0093 0.0092 0.0092 0.0092 0.0092 0.0091 0.0091 0.0090 0.0090	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	0.9706 0.9706 0.9706 0.9706 0.9707 0.9707 0.9707 0.9708 0.9708 0.9709
5(Outlet Blockage)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.6132 0.6131 0.6127 0.6120 0.6111 0.6100 0.6087 0.6071 0.6054 0.6035	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	0.3666 0.3668 0.3672 0.3679 0.3688 0.3699 0.3712 0.3727 0.3745 0.3764
6(Right Pump Fail)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.8014 0.8003 0.7976 0.7935 0.7880 0.7812 0.7734 0.7647 0.7553 0.7451	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	0.1785 0.1796 0.1823 0.1864 0.1919 0.1986 0.2064 0.2151 0.2246 0.2347
7(Left Pump Fail)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.9544 0.8982 0.8961 0.8929 0.8887 0.8837 0.8780 0.8717 0.8649 0.8575	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	0.0255 0.0817 0.0838 0.0870 0.0912 0.0962 0.1019 0.1082 0.1150 0.1223
8(Right Conc. Degradation)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	1.0e-022 *0.3001 0.3033 0.3113 0.3245 0.3435 0.3694 0.4035 0.4478 0.5049 0.5785
9(Left Conc. Degradation)	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799 0.9799	1.0e-008 *0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944 0.1944	1.0e-035 *0.9151 0.7230 0.4086 0.1658 0.0486 0.0104 0.0016 0.0002 0.0000 0.0000

Frequency = {1 2 3 4 5 6 7 8 9 10} = 10Khz, 60Khz, 110Khz, 160Khz, 210Khz, 260Khz, 310Khz, 360Khz, 410Khz, 460Khz

Table 14 Impedance Magntide 2 Test Outcomes

6.10.3.3 Impedance Phase 1

	Fault-Free as Fault-Free	Faulty as Faulty	Yield Loss	Test Escape
1 (Inlet Left Disconnect)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.9968 0.9970 0.9972 0.9976 0.9980 0.9985 0.9988 0.9991 0.9993 0.9995	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023	0.0028 0.0026 0.0024 0.0020 0.0015 0.0011 0.0008 0.0005 0.0003 0.0001
2(Inlet Left Blockage)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.0299 0.0299 0.0298 0.0298 0.0298 0.0298 0.0297 0.0297 0.0296 0.0295	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023	0.9697 0.9697 0.9698 0.9698 0.9698 0.9698 0.9699 0.9699 0.9700 0.9701
3(Inlet Right Disconnect)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.5122 0.5123 0.5123 0.5123 0.5124 0.5124 0.5125 0.5126 0.5127 0.5128	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023	0.4874 0.4873 0.4873 0.4873 0.4872 0.4872 0.4871 0.4870 0.4869 0.4868
4(Inlet Right Blockage)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.0474 0.0474 0.0474 0.0474 0.0473 0.0473 0.0472 0.0472 0.0471 0.0470	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023	0.9522 0.9522 0.9522 0.9522 0.9523 0.9523 0.9524 0.9524 0.9525 0.9526
5(Outlet Blockage)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.5222 0.5222 0.5221 0.5219 0.5217 0.5215 0.5212 0.5208 0.5204 0.5200	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023	0.4774 0.4774 0.4775 0.4777 0.4779 0.4781 0.4784 0.4788 0.4792 0.4796
6(Right Pump Fail)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.8670 0.8670 0.8668 0.8665 0.8662 0.8657 0.8651 0.8645 0.8637 0.8629	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023	0.1326 0.1326 0.1328 0.1331 0.1334 0.1339 0.1345 0.1351 0.1359 0.1367
7(Left Pump Fail)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.9916 0.9790 0.9786 0.9780 0.9773 0.9764 0.9754 0.9743 0.9732 0.9720	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023	0.0080 0.0206 0.0210 0.0216 0.0223 0.0232 0.0242 0.0253 0.0264 0.0276
8(Right Conc. Degradation)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.9906 0.9906 0.9906 0.9906 0.9906 0.9906 0.9906 0.9906 0.9906 0.9906	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023	0.0090 0.0090 0.0090 0.0090 0.0090 0.0090 0.0090 0.0090 0.0090 0.0090
9(Left Conc. Degradation)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.9996 0.9996 0.9996 0.9996 0.9996 0.9996 0.9996 0.9996 0.9996 0.9996	0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023 0.0023	1.0e-006 *0.1100 0.1059 0.0966 0.0834 0.0684 0.0532 0.0394 0.0277 0.0186 0.0120

Frequency = {1 2 3 4 5 6 7 8 9 10} = 10Khz, 60Khz, 110Khz, 160Khz, 210Khz, 260Khz, 310Khz, 360Khz, 410Khz, 460Khz

Table 15 Impedance Phase 1 Test Outcomes

6.10.3.4 Impedance Phase 2

	Fault-Free as Fault-Free	Faulty as Faulty	Yield Loss	Test Escape
1 (Inlet Left Disconnect)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.9955 0.9955 0.9957 0.9960 0.9963 0.9967 0.9970 0.9974 0.9978 0.9981	0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021	0.0041 0.0041 0.0039 0.0036 0.0033 0.0029 0.0026 0.0022 0.0018 0.0015
2(Inlet Left Blockage)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.0328 0.0328 0.0328 0.0327 0.0326 0.0325 0.0324 0.0322 0.0320 0.0318	0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021	0.9668 0.9668 0.9668 0.9669 0.9670 0.9671 0.9672 0.9674 0.9676 0.9678
3(Inlet Right Disconnect)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.6725 0.6723 0.6720 0.6715 0.6709 0.6701 0.6691 0.6679 0.6667 0.6653	0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021	0.3271 0.3273 0.3276 0.3281 0.3287 0.3295 0.3305 0.3317 0.3329 0.3343
4(Inlet Right Blockage)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.0704 0.0704 0.0703 0.0702 0.0700 0.0698 0.0696 0.0693 0.0690 0.0686	0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021	0.9292 0.9292 0.9293 0.9294 0.9296 0.9298 0.9300 0.9303 0.9306 0.9310
5(Outlet Blockage)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.5793 0.5791 0.5788 0.5783 0.5775 0.5766 0.5755 0.5743 0.5729 0.5713	0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021	0.4203 0.4205 0.4208 0.4213 0.4221 0.4230 0.4241 0.4253 0.4267 0.4283
6(Right Pump Fail)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.8849 0.8844 0.8832 0.8813 0.8788 0.8758 0.8723 0.8684 0.8642 0.8597	0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021	0.1147 0.1152 0.1164 0.1183 0.1208 0.1238 0.1273 0.1312 0.1354 0.1399
7(Left Pump Fail)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.9617 0.9386 0.9374 0.9357 0.9334 0.9308 0.9279 0.9247 0.9215 0.9181	0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021	0.0379 0.0610 0.0622 0.0639 0.0662 0.0688 0.0717 0.0749 0.0781 0.0815
8(Right Conc. Degradation)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.9996 0.9996 0.9996 0.9996 0.9996 0.9996 0.9996 0.9996 0.9996 0.9996	0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021	1.0e-006 *0.3407 0.3413 0.3429 0.3455 0.3491 0.3537 0.3594 0.3662 0.3743 0.3836
9(Left Conc. Degradation)	0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973 0.9973	0.9995 0.9996 0.9996 0.9996 0.9996 0.9996 0.9996 0.9996 0.9996 0.9996	0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021 0.0021	1.0e-009 *0.2451 0.2354 0.2135 0.1830 0.1486 0.1144 0.0839 0.0587 0.0393 0.0253

Frequency = {1 2 3 4 5 6 7 8 9 10} = 10Khz, 60Khz, 110Khz, 160Khz, 210Khz, 260Khz, 310Khz, 360Khz, 410Khz, 460Khz

Table 16 Impedance Phase 2 Test Outcomes

6.11 Search for Best Test / Fault Coverage

For large complex systems where there may be many sensors to be assessed then a search algorithm may be required to find the "Best" (most discriminating) test sensor. Due to the organization of the probabilistic results in the outcome structures then the search becomes trivial. Figure 131 demonstrates the simplicity of such an algorithm for searching the results for the mixed functional and Levich sensors.

```
for fault = 1:9
    [value, index] = max(F_FData(fault).Probability);
    BestTestPoints(fault,1) = FaultData(fault,index).TestPoint;
end;
```

Figure 131 Search Code

The resulting output of such a search would be a list of sensors, with each item respresenting a fault condition, (Figure 132). In this example the Outlet Pressure measurement sensor is shown to be the most sensitive to both inlet blockages, and combinations of Levich sensors for the other faults.

Fault Number	Best Test Sensor
1	'iOUT - 90th'
2	'Outlet Pressure'
3	'iLEFT - 90th'
4	'Outlet Pressure'
5	'iOUT - 90th'
6	'iRIGHT - 90th'
7	'iLEFT - 90th'
8	'iRIGHT - 90th'
9	'iOUT - 90th'

Figure 132 Search Output Results

Note: the 90th references in Figure 132 are the measured results taken in the 90th simulation second.

Fault Coverage, FC may be analysed for the functional, Levich and Impedance measurements according to the method proposed in [19] as reviewed earlier in this chapter.

Table 17 presents the fault coverage based on the probabilities shown in Table 12. To calculate FC the *probabilities of detection* are used. The description of FC in Equation 23 states that it is the probability that test T detects fault F, however, here we consider the probabilities associated with classifying a faulty system faulty. Therefore, in Table 12 we consider the FC for the combined set of functional sensors (1 to 15) for each fault condition and the combined Levich (iOut, iLeft and iRight) sensors for each fault condition. Additionally the end column shows the FC if both the functional and Levich sensors were to be used together.

Fault	**Functional**	**Levich**	**Combined TFC / Fault [%]**
1	0.1886	0.6659	**0.8545**
2	0.0009	0.0005	**0.0014**
3	0.1185	0.5504	**0.6689**
4	0.0011	0.0010	**0.0021**
5	0.0457	0.1881	**0.2338**
6	0.3968	0.3968	**0.7936**
7	0.3388	0.4997	**0.8385**
8	0.0007	0.6537	**0.6544**
9	0.0226	0.6653	**0.6879**
TFC / Test [%]	**0.1238**	**0.4024**	

Table 17 Functional & Levich Fault Coverage

The Levich sensors are shown to have more FC (3.25 times) than the whole set of functional sensors (TFC / Test [%]). The table also highlights the fact that even using combined sensors produces a low FC for faults 2, 4 and 5, which are all blockage faults.

Table 18 follows the same analysis however, this time using the impedance magnitude and phase measurements. The complete frequency scan (all 10 frequencies) is considered and averaged to produce the probability value. The end column of Table 18 does not combine the probabilities for each sensing type as in Table 17 since magnitude and phase 1 and magnitude and phase 2 are derived from different fault-free system responses (recall left and right channels).

The table shows weakness still in the coverage for faults 2 and 4, but improved coverage of fault 5 using impedance spectroscopy compared to functional and Levich sensors. Furthermore, the phase measurements show better coverage than magnitude measurements.

Fault	Magnitude 1	Magnitude 2	Phase 1	Phase 2	TFC / Fault [%]
1	0.9799	0.9799	0.9982	0.9966	**0.9886**
2	0.67×10^{-9}	0.1116×10^{-3}	0.0298	0.0325	**0.0155**
3	0.0647	0.5164	0.5124	0.6698	**0.4408**
4	0.2377×10^{-9}	0.0090	0.0473	0.0698	**0.0315**
5	0.2398	0.6097	0.5214	0.5764	**0.4868**
6	0.7423	0.7800	0.8655	0.8753	**0.8157**
7	0.9471	0.8886	0.9776	0.9330	**0.9365**
8	0.9751	0.9799	0.9906	0.9996	**0.9863**
9	0.9799	0.9799	0.9996	0.9996	**0.9897**
TFC / Test [%]	**0.5476**	**0.6381**	**0.6602**	**0.6836**	

Table 18 Impedance Magnitude & Phase Fault Coverage

While impedance spectroscopy is shown to have weak fault coverage of faults 2 and 4, improved coverage is gained using functional and Levich sensors. This study shows that using low test cost (real estate) impedance spectroscopy sensors provide comparable fault coverage as an abundance of functional sensors. If the total coverage is summed for each case then functional and Levich have a value of 4.735 and Impedance 5.6918, therefore 16.8% more coverage using less sensors and lower test cost.

6.12 Discussion & Conclusion

This chapter has introduced a method of test analysis based on the Neyman-Pearson method, which is widely used in the electronics test community. A test analysis algorithm has been developed based around this approach which couples the data from the MFS and produces test metrics for each sensor based on measured data from the simulation environment. The coupling process uses Gaussian curve fitting and KS testing to achieve an accurate transposition. Fault coverage analysis is also applied to the system to provide a single coverage metric.

The complete process MFS and test analysis has successfully been applied to a “Y” hydrodynamic case study in which impedance spectroscopy and Levich electro-chemical sensors were deployed to determine their sensitivity to a range of fault conditions. A summary of the case study is that a combination of impedance and Levich measurements provides more fault coverage than an abundance of functional sensors.

Chapter 7 Test Instrumentation & Fault Classification

7.1 Introduction

In this chapter we describe the instrumentation used to provide control and test for the experimental stages, with a view to further integration for system deployment through the introduction of decision trees. The novelty of the instrumentation presented is from its connectivity to the Simulation before Test and environment and its leaning towards an embedded single board solution. In this case we present a high voltage power supply used in electro-osmotic flow systems and an impedance spectrometer to provide embedded system test. Test schemes may be created automatically from the output of the SbT environment, by using decision tree analysis to provide fault classification schemes, the rules of which may then be loaded into the test instrumentation, in this case the impedance spectrometer.

7.2 Instrumentation Requirement

Microfluidic systems are highly multidisciplinary; therefore, personnel working on a single system have many different skills and specialities. It is important for progress that team members can speak a common language to share ideas and succeed in the tasks presented to them by the project. The integration of different workflows is important to this end. From experience the greatest language barrier is between engineers and scientists (chemists and biologists); often the scientists know what they want the system to do, but have problems describing it sufficiently to engineers for them to implement. Test (in the engineering product sense) is also often not familiar to scientists and there may be a problem with how to implement the often complex test rules for a given test instrument or system environment.

This next section addresses some of these issues through the introduction of dedicated instrumentation.

7.2.1 High Voltage Power Supply Instrument

The high voltage power supply, Figure 133 was developed for EOF applications and can be controlled manually or via a PC using an easy to use GUI, Figure 135. The instrument is designed to provide a wide range of voltages, depending upon the HV modules used inside.

Figure 133 High Voltage Power Supply

The instrument block diagram, Figure 134, shows the hardware implementation. The instrument provides communication via USB for individual computer control or via CAN should the instrument be used in a system with other instruments, controlled via LabView.

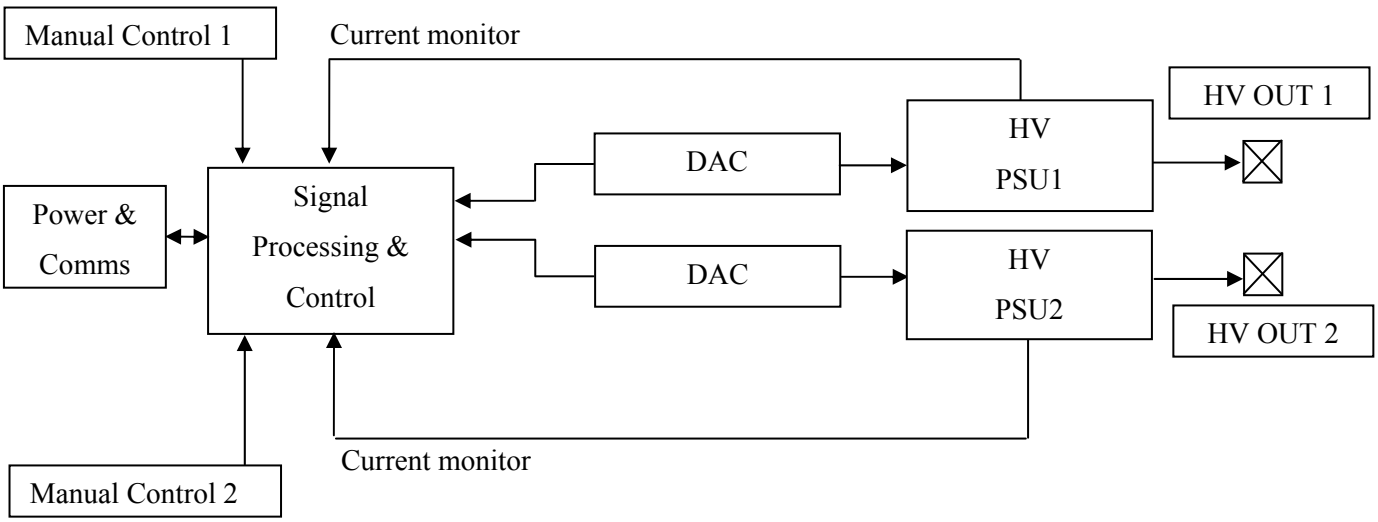


Figure 134 PSU Block Diagram

Off-the-shelf HV modules (UltraVolt) are used to provide two independent unipolar outputs (0 - +2Kv) or a single bipolar output (0 - +/-2Kv). These modules provide a low voltage current monitor which is recorded via the 10-bit on-board ADC's on the processor. Two 16-bit DAC's provide proportional HV control. Each channel has a manual control option.

7.2.1.1 HV PSU GUI

The HV PSU provides a simple control interface and safety interlock. The instrument is designed to work in three modes, which are selected from this interface. Manual mode (using the instrument mounted voltage control knobs) this is ideal if the researcher is wishing to perform a reaction or task where an original voltage estimate or incremental step value is not known. The second mode is "remote" mode where control is achieved through the GUI shown in Figure 135. The voltage value may be directly entered or varied using the slider control. Live current monitoring and fold-back protection may be implemented.

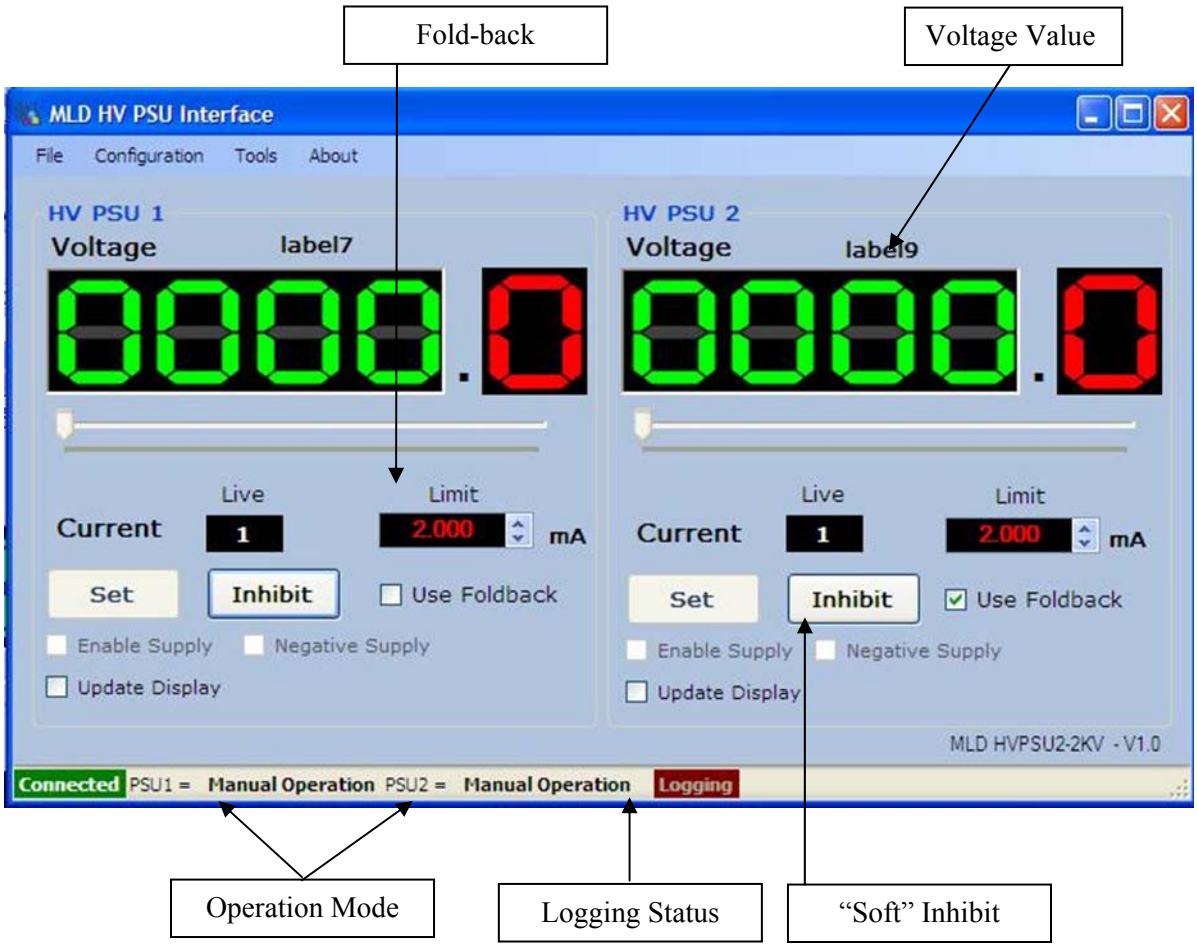


Figure 135 High Voltage PSU GUI

The third mode is custom sequence mode, (see Figure 136) where voltage sequences may be loaded in .CSV format from third party software, such as Microsoft Excel. This allows the researcher to create custom control sequences without requiring engineering support or the use of a programming

language. It is thought that control and switching equations, as previously presented in Chapter 4 & 5 can be used to implement real control and switching sequences for this instrument.

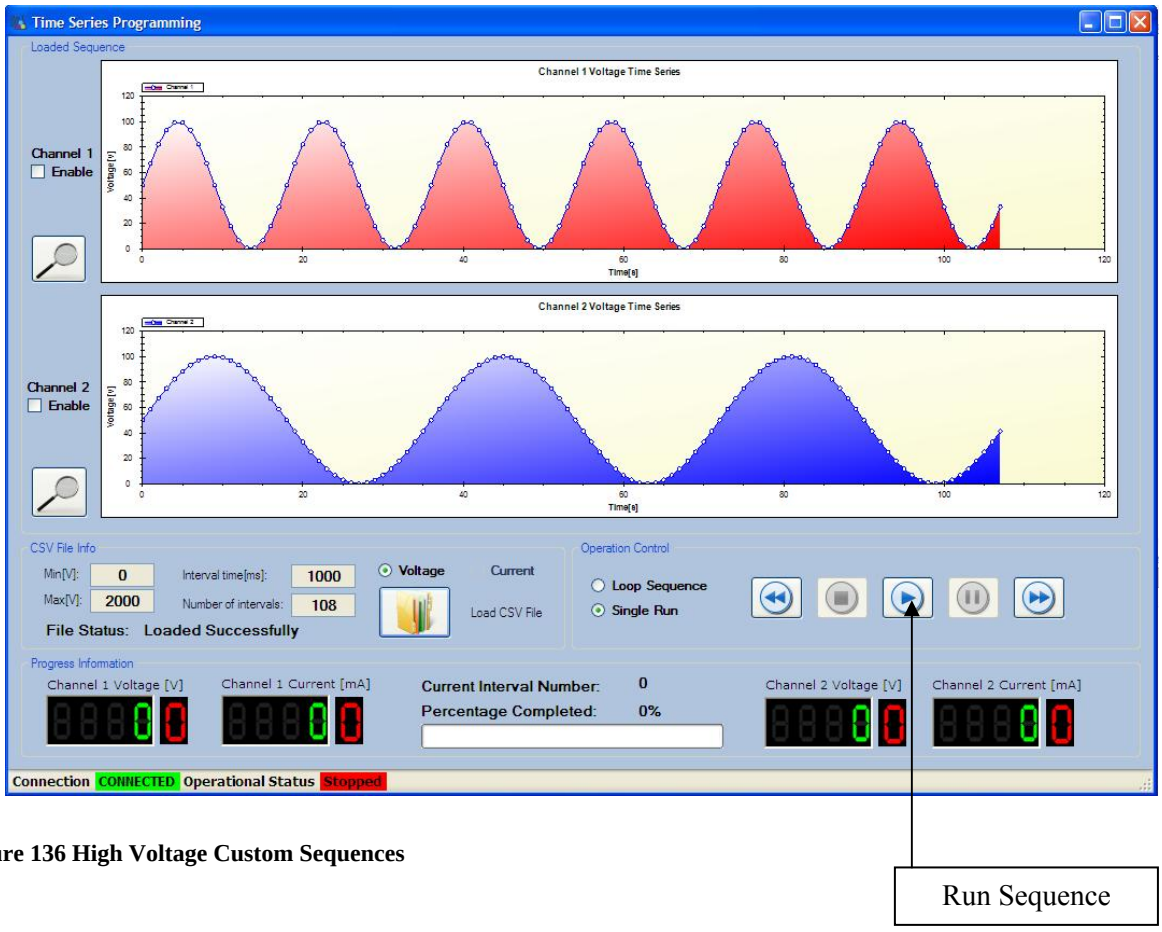


Figure 136 High Voltage Custom Sequences

In all three modes logging and charting may be performed on the voltage and current values and saved to a .CSV file for later analysis. A programmable sample period allows the researcher to capture fast value changes.

7.2.2 Embedded Control & Test Board

A microfluidics development board; Figure 137, was created to allow experiments to be carried out in an embedded control environment, to compliment the high voltage instrument. The board comprises such features as, pump speed control, valve control, a real-time clock, storage EEPROM, USB connectivity and a Bluetooth module for data download (in a deployable situation).

This board incorporates an Analog Devices AD5933 providing on-board impedance spectroscopy. There are other features such as pump and valve controllers along with Bluetooth communications.

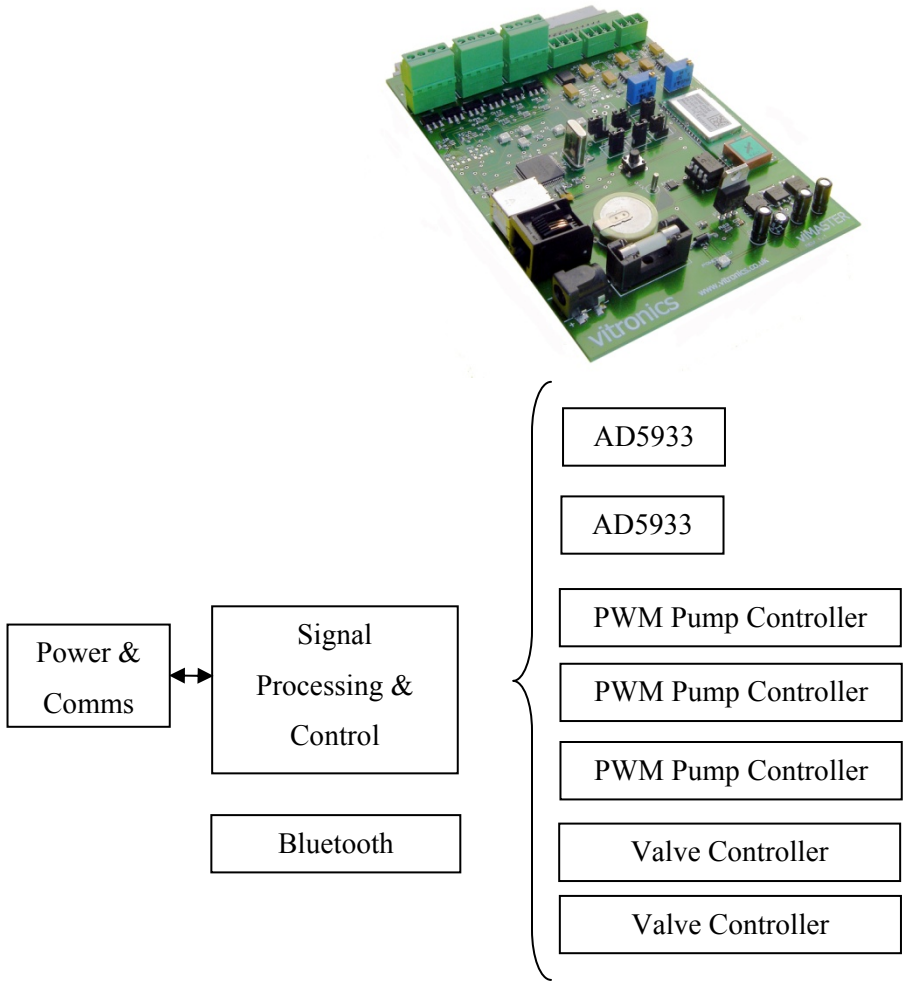


Figure 137 Photograph of Microfluidics Development Board created for experimental work

The AD5933 is a 1 MSPS 12-bit impedance converter, its small package size 16 SSOP and large impedance range 100Ω to 10MΩ provides a useful single chip solution for microfluidic impedance assessment. A block diagram of the connection details is shown in Figure 138, where Rfb is the "feedback" resistor for the device (the reader is referred to the AD5933 data sheet Rev 0).

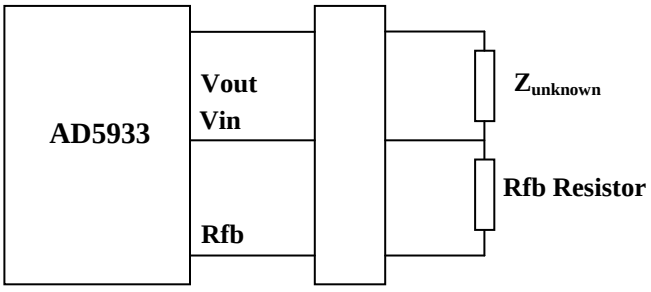


Figure 138 Schematic of Impedance Connections

Analog Devices provide a JAVA based design assistant for the AD5933 and AD5934 Devices, Figure 139. This interface allows the assessment of the device for a range of parameters such as start, increment frequency, voltage output level and settling cycles. An equivalent circuit is provided to mimic the application load.

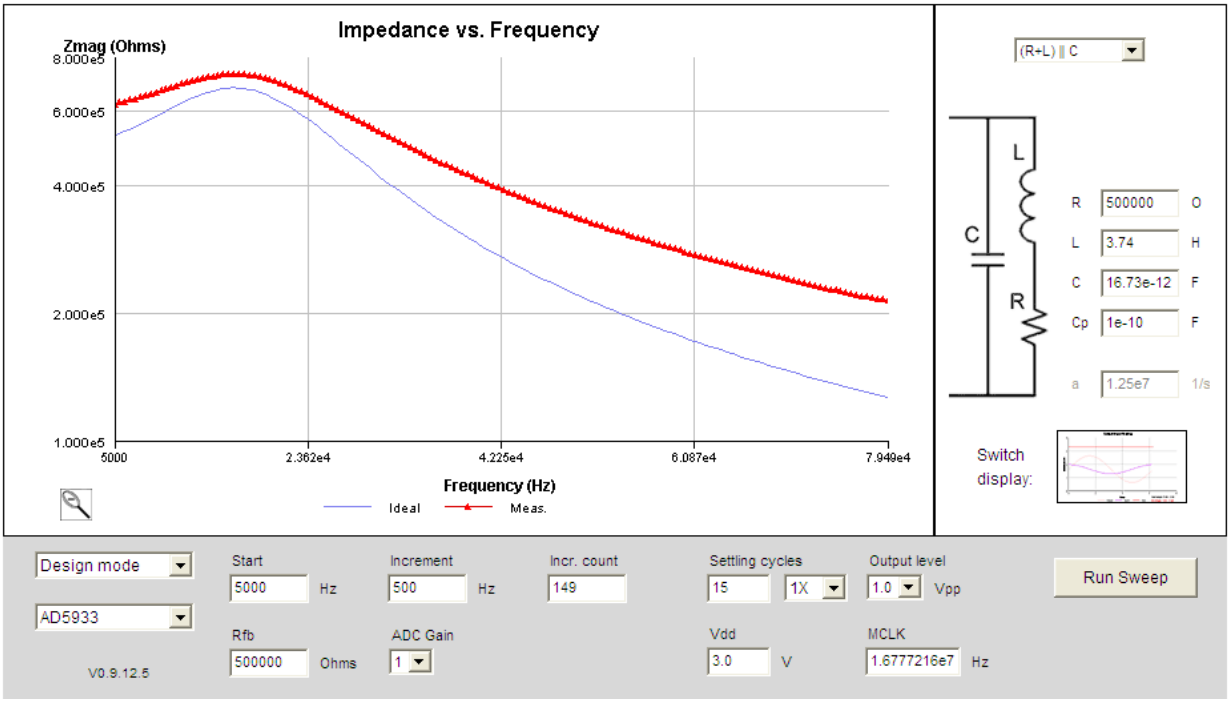


Figure 139 Analog Devices Online AD5933 Design Assistant

The AD5933 has two related limitations which prevent is use as a test sensor in microfluidic applications, witnessed experimentally and via the design assistant.

7.2.2.1 AD5933 Limitations for Microfluidics

1. The AD5933 requires calibration to the impedance in which it is to be operated "near", to establish a gain factor. Analog Devices do not specify the value of how much the $Z_{unknown}$ impedance can deviate from the calibrated impedance to remain accurate.

2. The "feedback" resistor has to be "near" to the $Z_{unknown}$ magnitude. Analog Devices do not specify a ratio, however, from our experimental investigation (which the design assistant cross-validates) it has to be around 1:1 (the exact tolerance has not been investigated nor has been published by Analog Devices Inc.) otherwise large measurement inaccuracies occur, Figure 140.

Post-calibration of an impedance value (resistance) close to the calibration resistance produces an accurate output where the deviation is less then <1% for the 500 KΩ resistor ($z_{unknown}$).

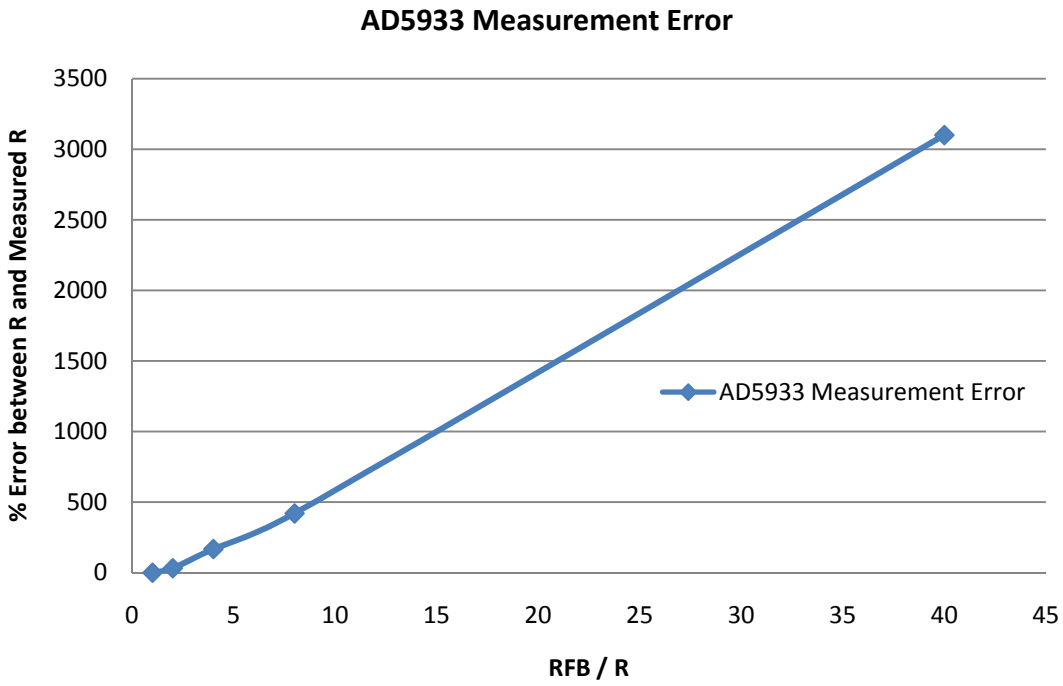


Figure 140 AD5933 Measurement Error zunknown to Rfb Ratio

However, Figure 140 demonstrates how significant errors can be produced depending upon the ratio between the unknown impedance and the feedback resistor, rfb. In one case the unknown impedance is 8 times greater than the feedback resistor, which produces a measurement error of 420%.

From the fault detection work in Chapter 4 it may be expected that the impedance magnitude may change by several orders of magnitude between a fault-free and faulty system. Therefore, it is imperative that the impedance test sensor is capable of measuring accurately over these ranges. One option would be to switch in different Rfb's to maintain accuracy over several ranges of impedances. This implies that the AD5933 device is unsuitable as an impedance test sensor for microfluidic applications, therefore in our research other approaches are considered.

7.2.3 Dedicated Impedance Spectrometer

The limitations presented by the AD5933 were overcome by the design of a dedicated impedance spectrometer, Figure 141.

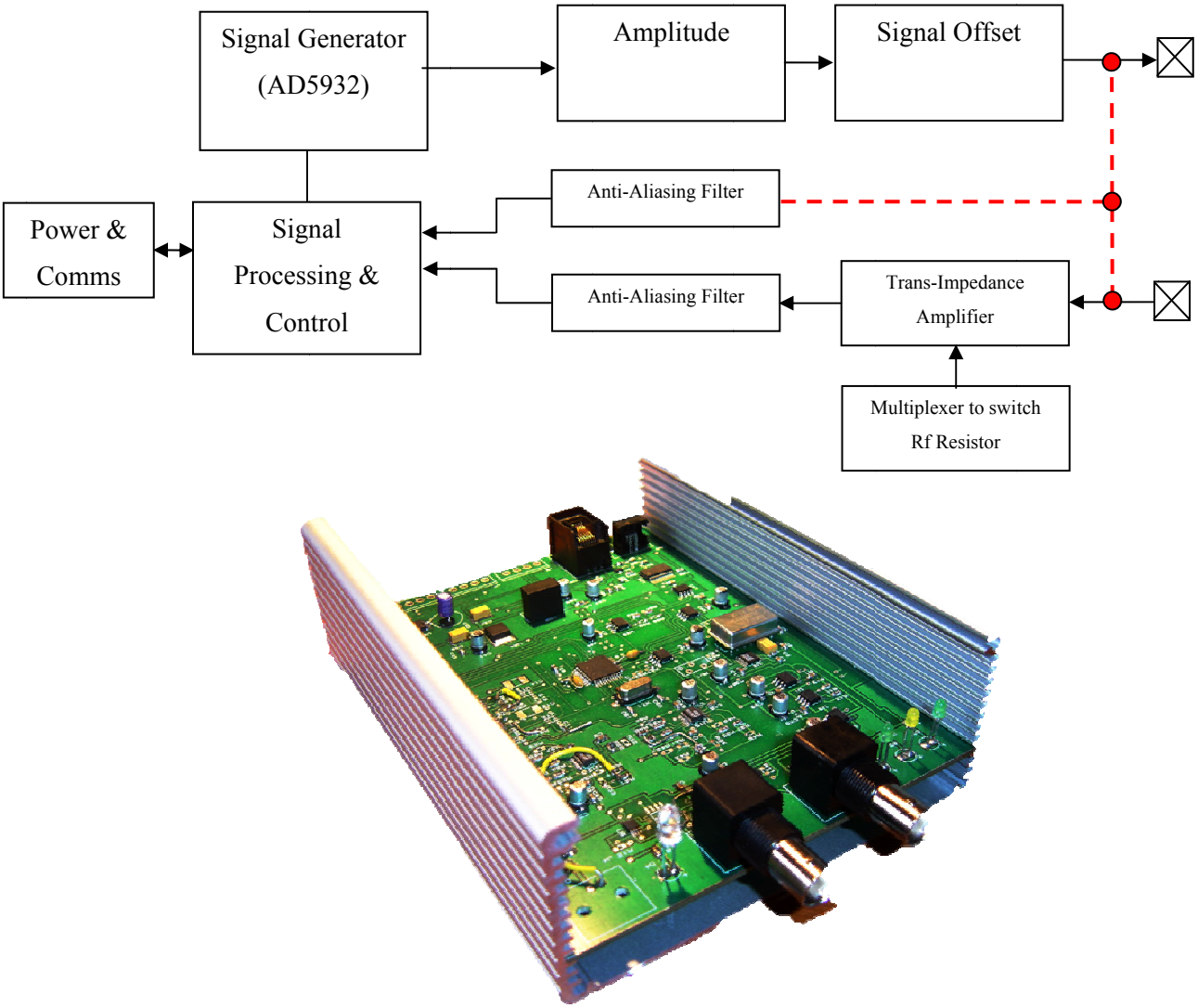


Figure 141 (Top) Spectrometer Block Diagram; (Bottom) Photograph of the Instrument

This device is USB powered and is based around a Microchip dsPIC33FJ128MC804. The signal is generated by an Analog Devices AD5932 DDS providing a working signal frequency of 1Hz to 10MHz with a resolution of 0.1Hz. The signal amplitude is controlled via a bipolar amplifier stage achieving a peak-to-peak signal of +/-5V, followed by a signal offset stage to bias the excitation signal if required; the excitation signal is then applied to the DUT via a BNC connector. The return signal from the DUT enters

the instrument via the second BNC connector and is presented to a current measurement stage. The current measurement stage employs a multiplexer to switch in a range of precision resistors to maximise the current signal from the trans-impedance amplifier for measurement by the microcontrollers 10-bit ADC. Anti-aliasing filters filter the measured current signal and the applied voltage signal before processing. A dynamic sampling frequency is applied depending on the set excitation frequency to improve the captured signals time resolution.

Processing involves determining the impedance magnitude of DUT achieved by measuring the voltage difference between the excitation signal voltage level and the returned DUT voltage level, and then dividing this by the measured current value, which is proportional to the current measuring resistor on the trans-impedance amplifier. The impedance phase is calculated by measuring the time shift between the voltage and current sine waves. This time shift is converted to phase by knowing the sampling frequency of the ADC (the time between samples).

The instrument contains a state-machine which responds to commands issued from the PC, this instrument acts as a slave. The three commands are to initialise the instrument with the start, increment frequency, signal amplitude and offset; another command to increment the frequency and a command to measure the impedance (magnitude and phase).

10KV galvanic isolation is provided for the power supply and USB communications to allow operation on ground or DUT's at potential.

7.2.3.1 Impedance Spectrometer GUI

The user interface, Figure 142 allows the instrument to be configured graphically. Provision to either implement a frequency sweep or work at a fixed frequency is provided. A sweep is beneficial where investigative work is being performed and a fixed frequency when the presence of a particular fault is being monitored.

In sweep mode a start and increment frequency are provided, the number of increments and the time period between increments. The signal amplitude and offset are fixed in this release of software to +/-1V and 0V, respectively.

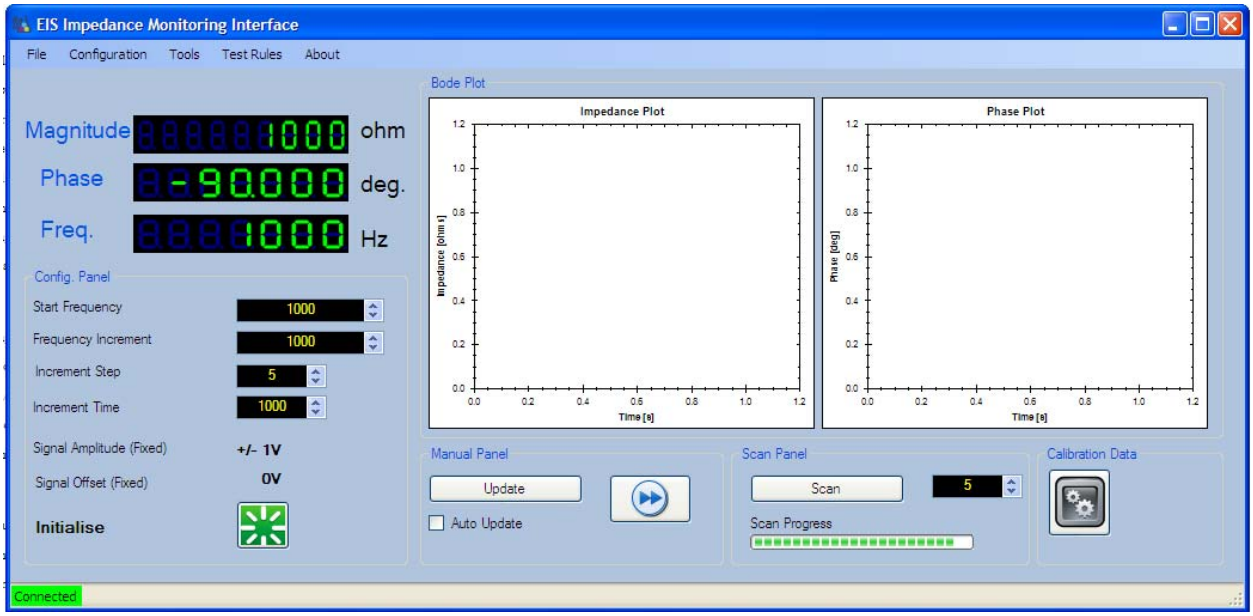


Figure 142 Impedance Spectrometer User Interface

Bode plots are populated per frequency sweep, and may be downloaded in .CSV format for analysis in third party software if the logging option is selected. In fixed frequency mode the "live" impedance magnitude and phase is plotted and may be logged using the logging feature.

Test Diagnosis

The test diagnosis feature, Figure 143 converts the impedance spectrometer into a microfluidic diagnostic tool. This feature allows the loading of test rules, which are monitored during the operation of the microfluidic system to provide test diagnostics to the user about the systems behaviour. The example in Figure 143 shows the display of a healthy system and then a system where a bubble condition has occurred. This has applications in areas such as buffer loading in a production environment, or in deployment if this feature was to be implemented in a purely embedded system.

Figure 143 Impedance Spectrometer Test Diagnostic Screen

The rules for this approach can be implemented through intuition or by classification schemes created by decision trees. The data to derive these schemes is the data from the afore mentioned SbT approach. This provision is in place, but has not been fully ratified.

7.3 Decision Trees

In the previous chapter system simulation (functional and test sensor) data was analysed to determine the probability of detection and the test hypothesis. This analysis provided metrics on the quality of the tests. In this chapter this data is further analysed using decision tree classification to develop a scheme of fault classification.

Decision Tree analysis constructs a classification scheme (based on the previously recommended test sensors) to allow the faults to be classified. MATLAB provides a classification tree function, the advantage of using such an approach is that this function systematically assesses large data volumes, such as those output from the SbT, and derives the optimum scheme, which is presented in human readable form. This allows the data produced by the SbT to be practically implemented and used to detect and classify microfluidic faults, without the workflow user having to decode the SbT outcomes into a test scheme, although that option remains should the user wish.

The decision tree objective is to eliminate (in order to classify a system) as many possible conditions (fault conditions) as possible with the least number of branches, through the application of simple rules; if A is greater than B then branch X or otherwise branch Y. The decision tree algorithm is nonparametric (no assumptions are made on the data) the data is presented and branches made to the next test or the data classified. The construction of the tree (set of rules) is acheieved through training data. A set of data n by m is presented, where n is the test data and m the classification of that that data set. The data is parsed and split into unique tests. Three types of decision split criterion may be implemented in MATLAB. These are the Gini splitting rule, the twoing rule and the deviance rule.

The Gini rule strives to split the largest class from the data group, and then subsequently larger classes. The twoing rule splits the data into two groups which represent 50% of the data set [144]. The deviance rule is based on the information content of the instance of data; its entropy, determined by Equation 31.

$$\text{entropy}(p1, p2, \ldots pn) = -p1 \times \log p1 - p2 \times \log p2 \ldots - pn \times \log pn$$

Equation 31 Entropy Calculation

This determines how much information (bits) a new instance of data must contain in order for a particular node to correctly classify it.

7.3.1 Decision Tree Implementation

The *classregtree* function found in the MATLAB statistics toolbox which may be used to create decision (classification) or regression trees. The function requires an array of predictor values (TestSensors) and a corresponding array of classifiers (FaultConditions). An example is shown in Figure 144.

```
T = classregtree(TestSensors,FaultConditions,'names',SensorNames);
```

Figure 144 Classification Function Code

The additional prototype, *names*, may be used to label the predicator variables for clear identification of the terminal nodes. The resulting decision tree, T consists of a set of rules based around decisions made at each terminal node against a predetermined value. If new data is greater than the value a branch to the right is taken, if less then to the left, this process continues until the data is classified. These rules are visually represented in the form of a decision tree, Figure 145. The decision tree graphs for all sensor groups can be found in Appendix IV.

Throughout this thesis the Gini split criterion is used. This is the default split criterion in MATLAB and strives to isolate the largest class. For example, if a class A represented 40% of all the classes then the

Gini criterion would determine the rules first to isolate this class. Not only does this present itself as the most logical over twoing and deviance, but when considering fault classication the most frequently (or likely) fault would present itself as the largest class, therefore the classification rule would classify this condition at the beginning of the rule set, providing a quicker response if the condition presented itself to the system.

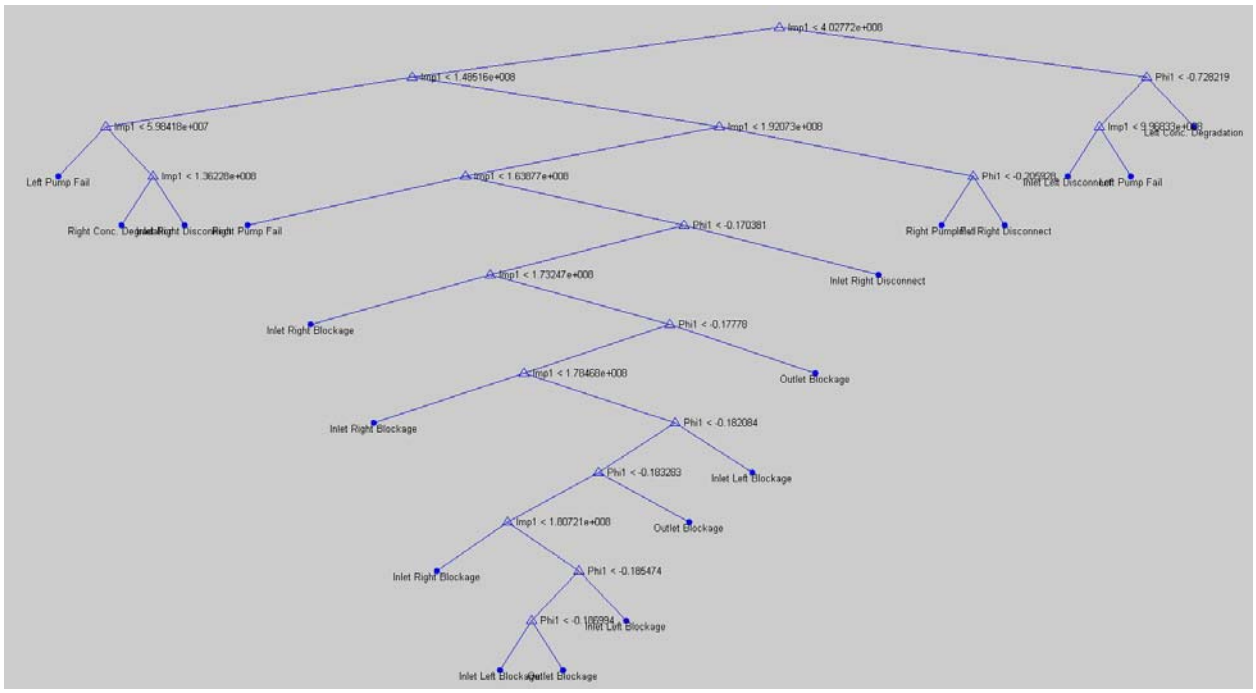


Figure 145 Decision Tree Graph (All Sensors)

The output of the decision tree algorithm is a set of if … then rules. They are generally displayed visually as a decision tree, however, the test instrumentation may be loaded with these rules and processed as text based macros.

7.3.2 Decision Tree Cost and Classification Rate

Chapter 6 determined probabilistically the most efficient tests based on their discrimination between the fault-free and faulty response. In this chapter our interest is in using the measurement data to create a practical fault detection and classification scheme, using the decision tree technique.

While the probabilistic metrics previously determined provide a good indicator of potential sensor performance, the decision tree (classification tree) provides further information about the ability of that sensor to correctly classify a particular condition and how many steps (nodes) have to be processed to do so.

In this section it is investigated whether a limited number of test sensors, can provide the same (or better) classification rate as an abundance of sensors (functional and test). Therefore, the question is posed "what is the minimal test hardware required without increasing misclassification?"

To answer that question the measurement data from the "Y" channel case study (Chapter 6) is used and the following sensor groups are investigated; All Sensors (test and functional), Test Sensors (Impedance and Levich), Impedance Magnitude Only and All Impedance (Magnitude & Phase).

Our interest here is to find out how do these sensors groups perform at correctly classifying a fault condition. Taking this further, how many tests (sensors) does the system require to reach an *acceptable* classification rate? The metic which helps answer these questions is the misclassication cost. Misclassification cost is the probability of misclassification based for each number of of terminal nodes (tests applied) Figure 146, where cost (y axis) is the probability of misclassification based on the number of terminal nodes (x axis); in general the more terminal nodes (test points) the lower the rate of misclassification. The standard error markers show the misclassification probability for the minimum number of terminal nodes required to achieve that probability, identified by a square marker.

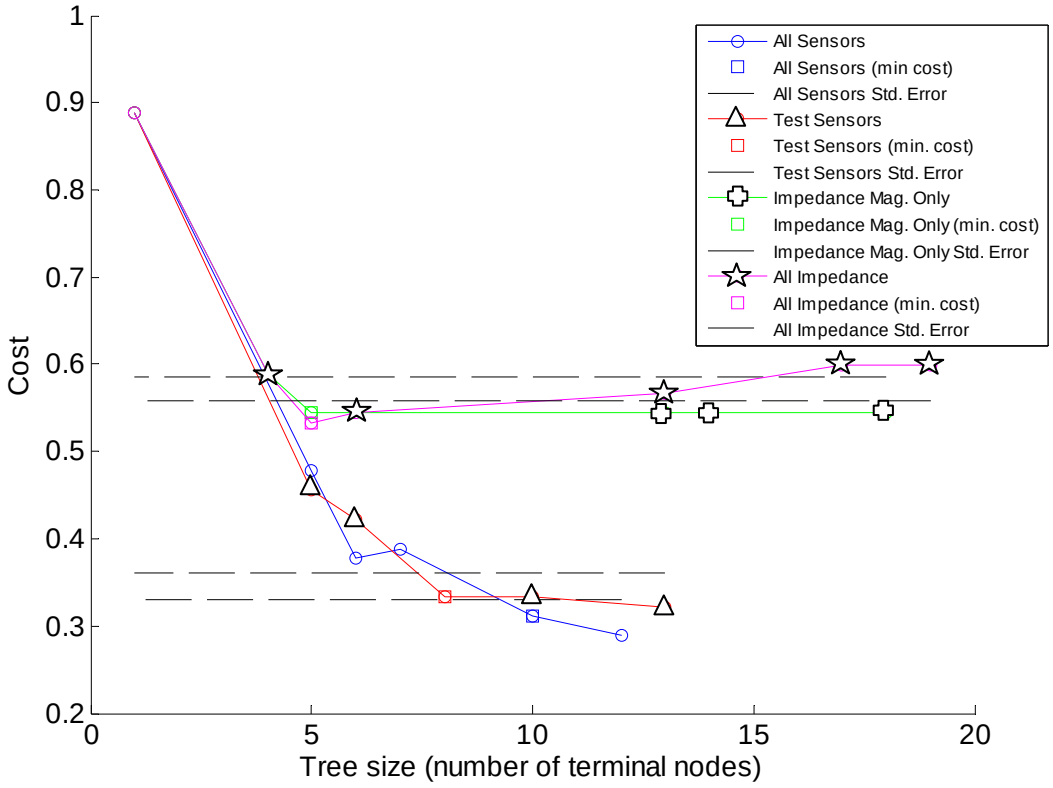


Figure 146 Analysing the Diagnostic Cost for different schemes

Table 19 summaries Figure 146 by analysing the minimum number of terminal nodes to achieve the standard error the impedance sensor groups require the fewest nodes, 5, however, they have a high misclassification rate of just over half.

Group	Min. Terminal Node	Min. Terminal Probability
All Sensors	10	0.32
Test Sensors	8	0.34
Impedance Mag.	5	0.54
All Impedance	5	0.54

Table 19 Misclassification Costs

The most profound observation is that the deployment of functional sensors and test sensor (All Sensors) only yields a misclassification improvement of 0.02, compared to only deploying dedicated test sensors (lower cost and inregrated into the design) with a probability of 0.34. Furthermore, the test sensor scheme requires fewer terminal nodes, 8 compared to the All Sensor approach. While the misclassification rate is high using this method, this is not our interest here since a larger dataset may by required to improve upon this, and would be the subject of further work. Our interest here is in the observation that decision tree analysis verifies our findings from Chapter 6 in that a combination of test sensors (Levich and Impedance) is required to determine and classify all fault conditions, which is reflected in Table 19 along with the observation that an abundance of functional sensors does not provide a significant improvement in classification over test sensors alone.

7.4 Discussion & Conclusion

Microfluidics is highly multi-disciplinary; therefore, instrumentation is required to be used by engineers and scientists. The instrumentation presented at the simplest level operates out-of the box with little training however, its advanced features allow for customisation for those requiring more advance interaction.

Impedance spectroscopy instrumentation is required to be able to operate over many impedance magnitudes in a single sweep, negating the use of the only dedicated network analyser IC on the market, the AD5933. Our impedance spectrometer has been shown to be robust and easy to operate, capable of measuring the impedances required in a microfluidic test environment. Furthermore, the dedicated GUI allows the loading of classification schemes from decision tree analysis, making this test instrument useable and programmable by non-test engineers.

Decision tree analysis provides an efficient means of classification scheme creation, having the advantage of it being in human readable form. The auto-generation of the decision rules which may be directly loaded into the test instrumentation makes this approach useable by non-test engineers, ideal for use in the multidisciplinary environment of microfluidics.

The analysis of misclassification costs revealed that while the test sensors showed a high degree of discrimination in chapter 6, they provide a high degree of misclassification, 0.34, even though impedance sensors have been shown to have a low error (Type I or Type II) when studying test hypothesis. This further promotes the decision tree technique as not only a useful tool for classification scheme creation, but for post-test analysis to determine the potential misclassification rate.

The high voltage power supply and impedance spectrometer are used in the experimental stages of the next chapter.

Chapter 8 Case Study: Microfluidic Flow Cytometry

8.1 Introduction

This chapter presents a second test case acting as complete trial of the developed methodology, covering design, simulation and test. The methodology has been developed asynchronously to active microfluidic projects within the University, therefore, timing does not permit that a system design is optimised through simulation, instead, an existing experimental system design is utilised for simulation and test purposes. At the end of this chapter the next generation of system design is considered.

The aim of the methodology is to provide a rapid, design and test assessment of a proposed microfluidic system, without laborious cross-validation and development of test metrics. Therefore, in this chapter the methodology will be used as would be expected by a microfluidic system developer. Previous work indicates confidence in the simulated system can be high, therefore, in this chapter only the flow parameter is cross-validated to ensure some correlation between the two approaches.

The system studied in this chapter is a flow cytometry sorting system, used in the assessment of cancer cells. The system is capable of disaggregating cells from cancerous tissue, transporting them using electro-osmotic flow, attaching a death marker, and then classifying the cell using fluorescence. These cells are subsequently sorted into live, dead and waste classes using an electro-osmotic switching technique.

A limited number of likely fault conditions will be considered; blockage due to cell lysing and electrode degradation. In the later case, this fault condition is based on applying ohms law within the system and will not be cross-validated.

The next generation of the system design is investigated through heterogeneous FEM simulation, allowing the use of Electro-osmotic Pumps (EOP's) to be assessed before committing the design to manufacture.

8.2 Flow Cell Cytometry

Flow cytometry is a well established method for counting cells having a particular expression marker; generally they are used in clinical research and prognosis of cancer cells. These instruments are typically large, expensive and require expert training. *FACS* (Fluorescence Activated Cell Sorting) is an extension of flow cytometry which sorts the cells based on a fluorescent marker. In this research FACS is implemented on a microfluidic device for the study of head and neck cancer cell lines. Head and neck cancer is highly heterogeneous in location and origin and the metastatic behavior differs from patient to patient, making prognosis (determining a benign tumor from one with metastatic potential) difficult. There has been no improvement in morality survival rate in decades with a 5 year survival rate of 50%.

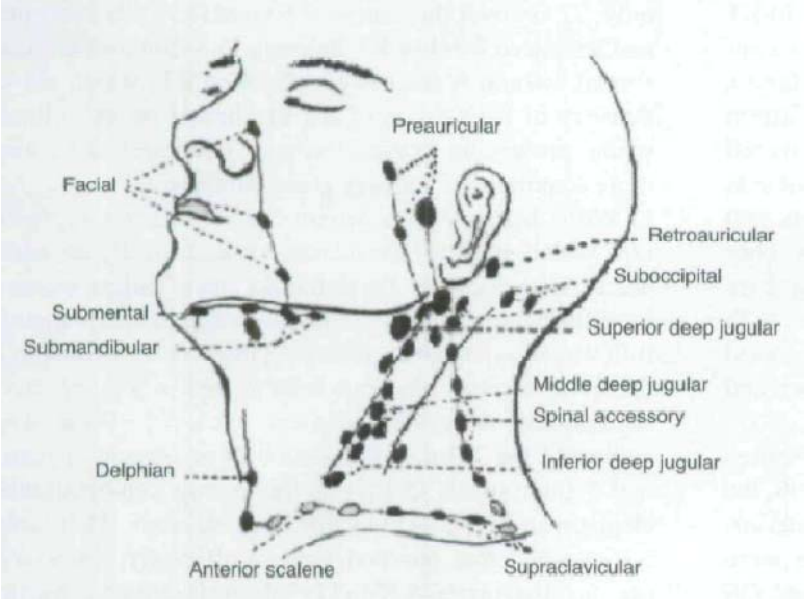


Figure 147 Head and Neck Tumor Sites (Marur 2008)

Flow cytometry can aid prognosis by being able to identify metastatic markers in a variety of cell lines, therefore there is a strong advantage of having highly deployable and easy to use devices. Such a device could be based around a microfluidic reactor.

The potential for highly deployable integrated devices for medical prognosis, with potential use by *non-experts* results in a strong requirement for test and on-line testability when in deployment. The subject of test is explored in this chapter. Experimental and simulation analysis of potential faults are cross-validated for later use in fault simulation and subsequent test analysis to determine the effectiveness of the deployed test strategy. Furthermore, the design of the next generation device is explored using the same simulation method.

8.3 Experimental Apparatus

The experimental apparatus shown in Figure 148 was used to perform EOF and cell switching, and also to create faulty system conditions for assessment.

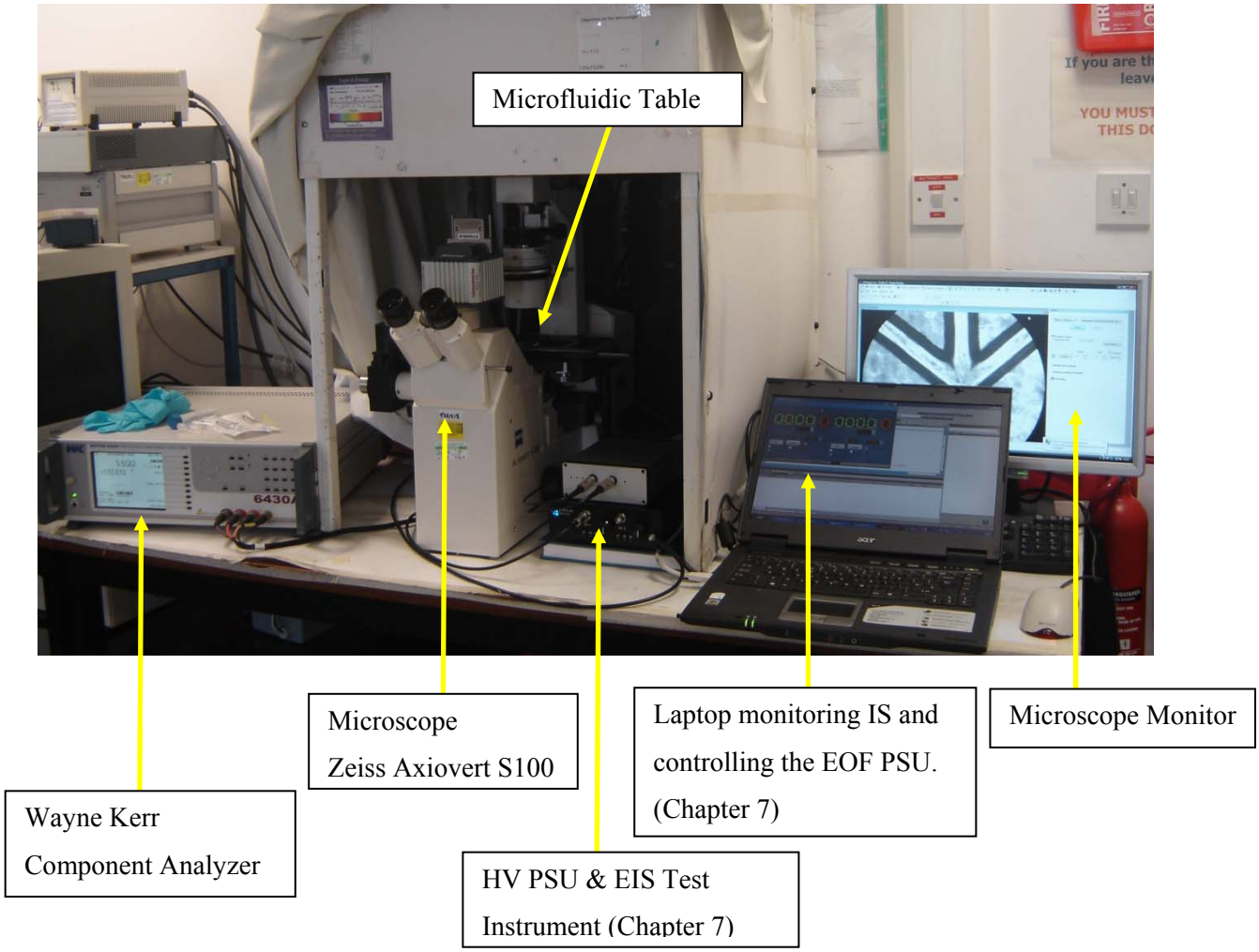


Figure 148 Cell Sorting Experimental Apparatus

The Zeiss Axiovert S100 microscope is used for observing EOF flow rates during cross-validation and for observing the effects of blockage fault conditions. The same microscope is also used for observing the fluorescence of the system during cell sorting, a topic out of scope for this thesis. The Wayne Kerr impedance spectrometer is used in conjunction with the impedance spectrometer described in Chapter 7, as a checking aid to the new instrument. The HV PSU (described in Chapter 7) is used to perform switching and to cause catastrophic cell lysing blockage conditions . The experimental schematic

is shown in Figure 149. The HV and impedance spectrometer both share the same electrodes. Only one instrument is connected at a time.

Figure 149 Experimental Schematic (PSU and IS)

A custom built microfluidic reactor holder has been constructed to aid reproducible measurement results and to aid the connection of control and test instrumentation, Figure 150. The holder is formed from three Polycarbonate substrates; the base to locate the reactor, a lid and then clamps with integrated spring loaded gold coated probes. The reactor is highlighted in the box shown in Figure 150.

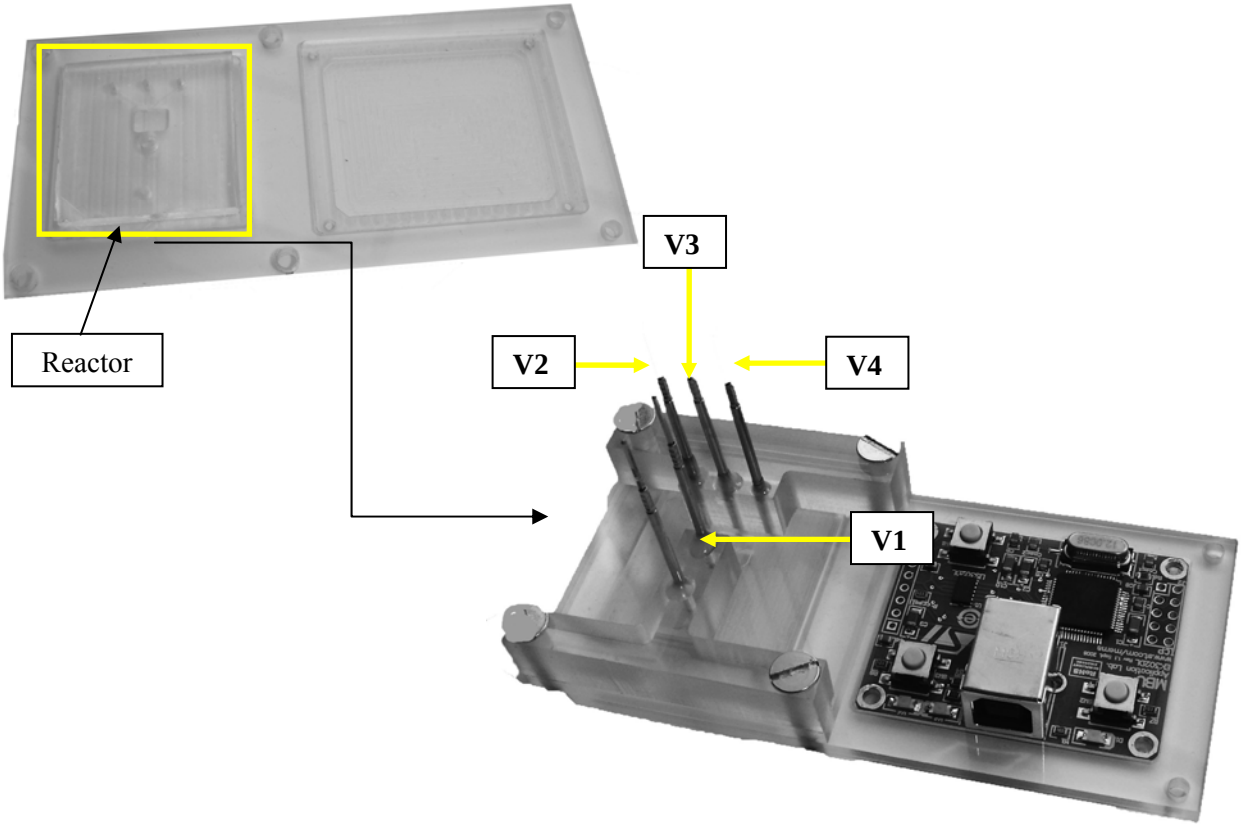


Figure 150 Microfluidic Reactor Holder

The probes labelled V1 – V4 are used as functional EOF voltage electrodes and as test electrodes for impedance spectroscopy measurements. The board on the right is an ST accelerometer which is not used in the research presented in this thesis.

This concludes the experimental apparatus; the next section will describe the simulation implementation of this system and the dimensions of the reactor before proceeding to the investigation of how to test it.

8.4 System Simulation

A complete reactor schematic is shown in Figure 151 (Left). The reactor features a large tissue entry port where tissue may be inserted to be disaggregated. For the purposes of experimentation this port is ignored, as the cells are supplied disaggregated from tissue, therefore the simulation geometry may be reduced to the section shown in Figure 151.

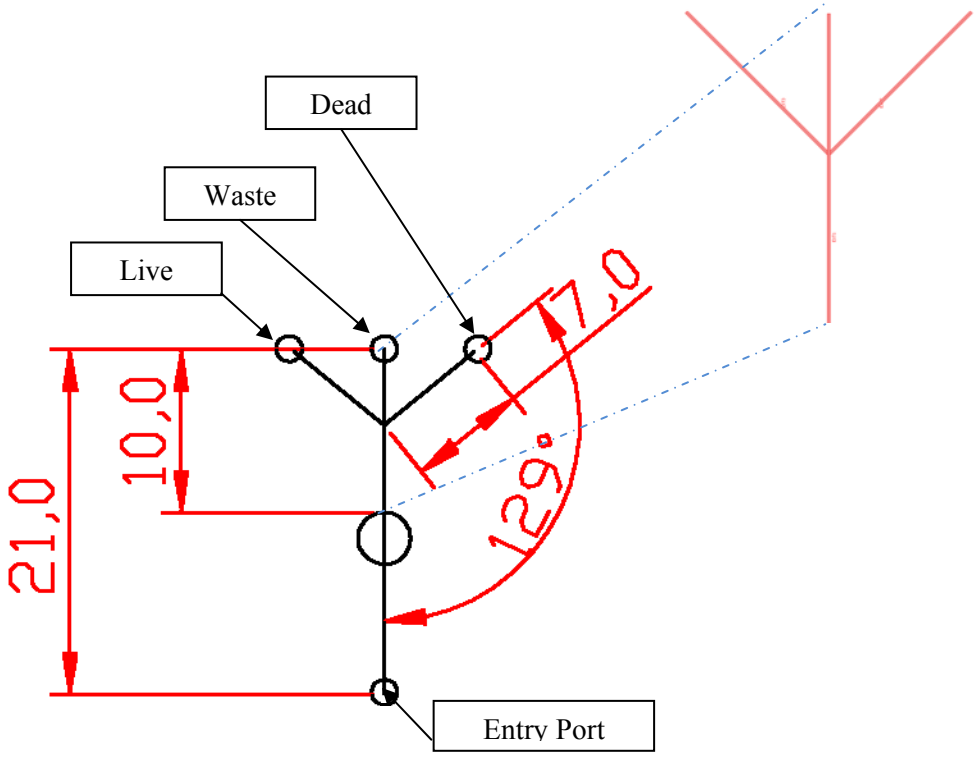


Figure 151 (Left) Complete Reactor Diagram (Inset/Right) COMSOL Geometry

The system comprises 3 application modes, Table 20; Stokes Flow, Conductive Media DC and the Electric Currents application mode for impedance spectroscopy. The ruling application mode is Stokes Flow.

Application Mode #	**Description**	**Abbreviation**
1	Electric Currents	Emqvw
2	Conductive Media DC	emdc
3	Stokes Flow	mmglf

Table 20 Application Modes

The Analysis Type was Transient for both the mmglf and emdc application modes using a direct SPOOLES solver. The simulation time range was 0:1:2 seconds. The two application modes were solved concurrently using the initial value expressions. The Electric Currents application used a parametric solver to implement the frequency sweep, by modifying the variable *nu_emqvw*.

All boundary conditions are described as walls having an electro-osmotic velocity or having electric insulation. The sub-domain parameters are such as required to describe the experimental conditions 0.5% Agarose gel, derived from HEPES/Sucrose with concentrations 25mM and 250mM, respectively. For the purposes of the simulation the density and viscosity have assumed values 1000[kg/m^3] and 1e-9[Pa*s], respectively and an electrical conductivity of 0.22[S/m], derived from a previous experiment. The Zeta potential is positive and fixed at 13mV resulting in anodic EOF flow.

8.4.1 Flow Cross-Validation

For the purposes of cross-validation a neutral dye is used experimentally to determine the flow rate for given electric field strengths. A neutral dye is used instead of the Agarose gel containing beads or cells to prevent the measuring of their EOF mobilities due to electrophoresis in the field, instead of the required velocity due to electro-osmotic flow.

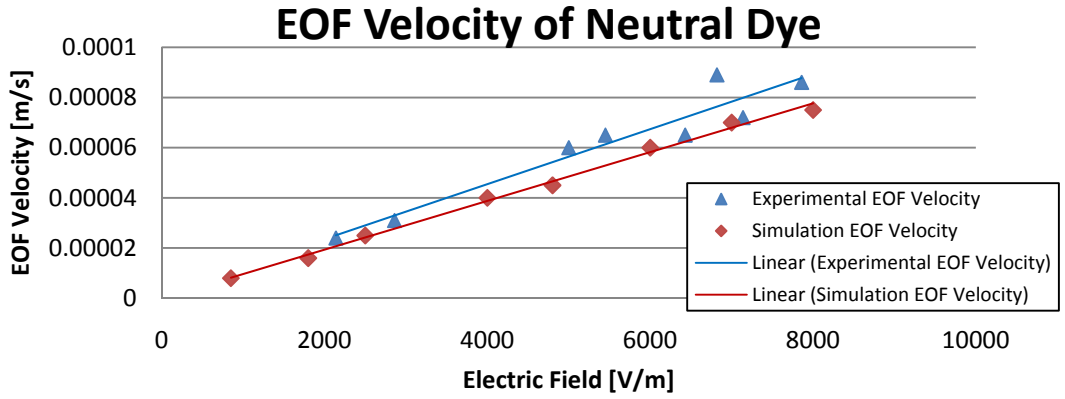


Figure 152 EOF Cross-Validation Plot

From the EOF plot, Figure 152 the greatest error between experiment and simulation is 9.2% this represents a high degree of confidence in the simulation system, and remains consistent with the cross-validation bounds of simulated systems in previous chapters.

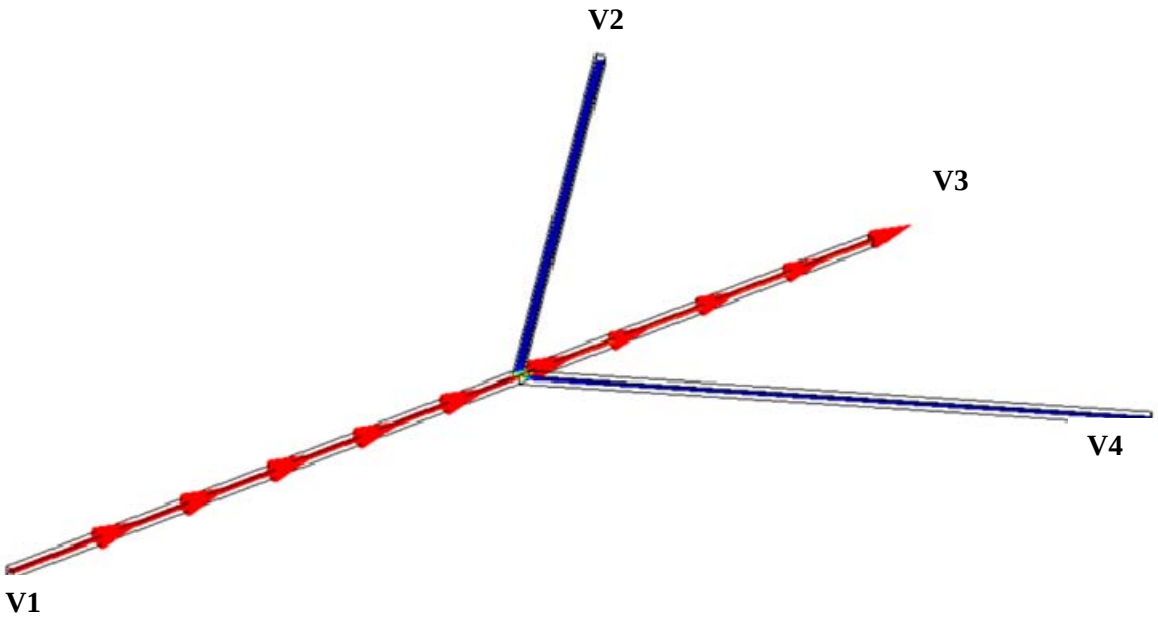


Figure 153 EOF Velocity Slice Plot

The Velocity Slice plot, Figure 153 shows anodic EOF flow for the conditions given in Table 21 causing flow in the main channel only.

Variable	Value
V1	0V (ground)
V2	Insulation
V3	60V
V4	Insulation
Zeta	0.013[V]
Conductivity	0.22[S/m]

Table 21 Example EOF Flow Condition Variables Main Channel

The electric field strength for the described electrode placement / potentials is approximately 5000 V/m along the main channel (V1 to V3) and 0 V/m otherwise. The volumetric flow rate at V3 is 2.908703e-24 [m^3/s], the corresponding velocity is 40µm/s.

8.4.1.1 Flow Switching

For the sake of brevity the simulation may be used to provide detailed information about the switching characteristics of the system. The analysis performed is similar to that for the Sun *et al.* model described in Chapter 3. Switching is the most important performance of a sorting system; however, no specification exists for the system at this time.

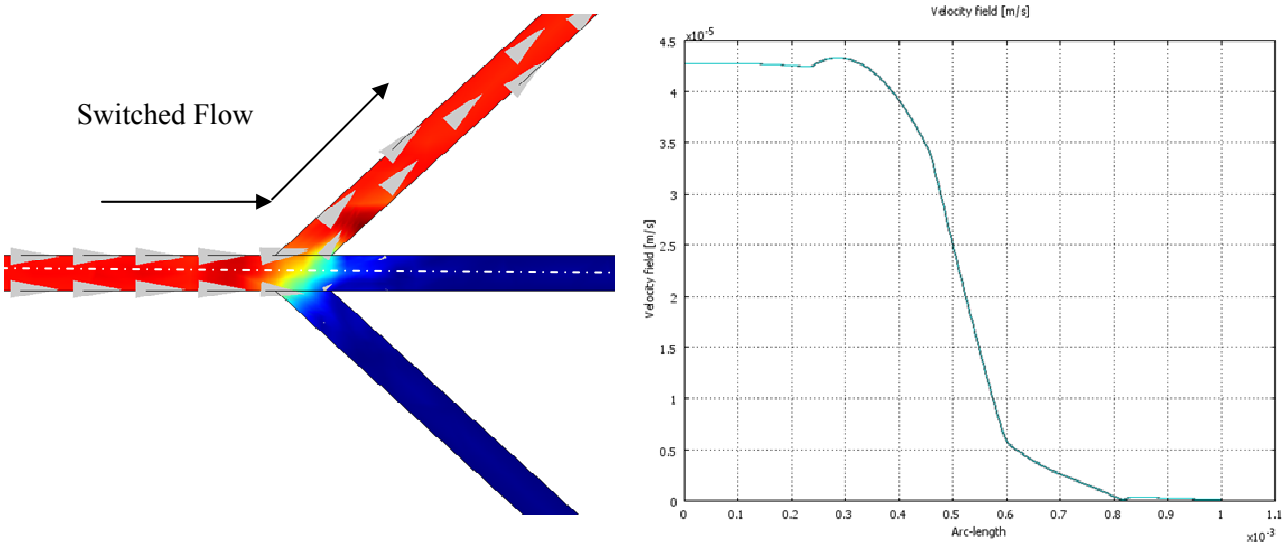


Figure 154 Switching Slice Plot

Figure 154 provides a visual analysis of the switching towards V2 electrode, the white dotted line represents the arc-length from where the velocity plots, Figure 154 (right) is taken across the junction. The velocity plot shows very little leakage into the main channel (waste) when V2 channel is switched. The velocity is shown to go from full flow to zero flow with leakage of only 500µm.

This simulation overview highlights the fact that the two main features of the system are sufficiently described to begin investigating how these performance parameters are affected by fault conditions.

8.5 Cell Sorter Fault Conditions

This system introduces two new fault conditions to the existing fault library. This demonstrates the flexibility of the library and mapping method to include new fault conditions are they are reported. In this case cell lysed blockage and electrode degradation are described.

8.5.1 Cell Lysed Blockage

This is a continuation of biological activity blockages described in Chapter 4. In this instance cells are sorted in this system using EOF flow. Through experimentation it was observed that an excessive electric field applied to K562 blood cell precursor cancer cells resulted in them lysing and the resulting cell debris would travel along the channel and begin blocking the channel, the more cells that lysed the greater the problem, until eventually the blockage would act as an insulator and EOF would cease.

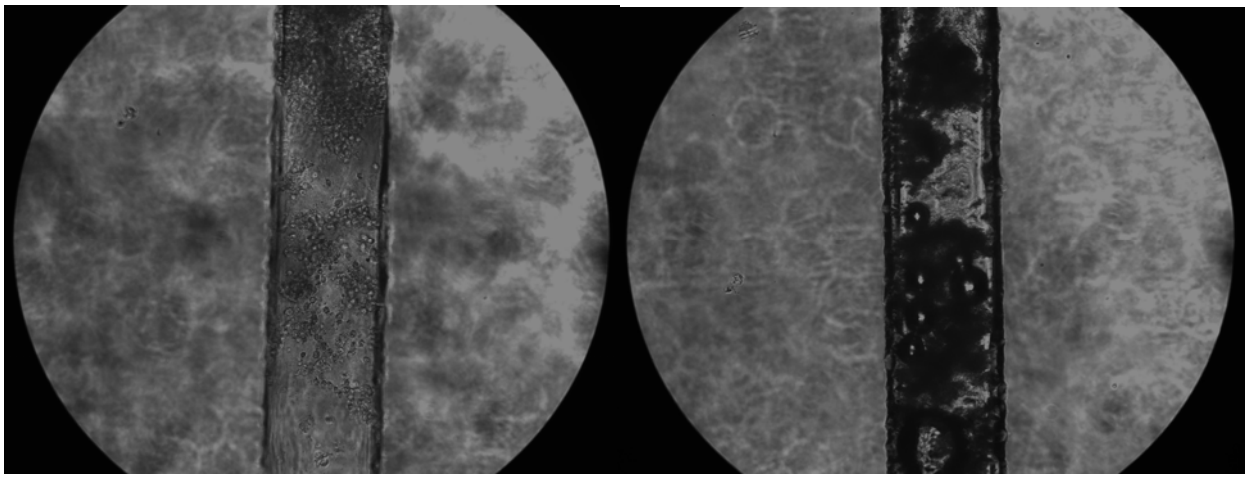


Figure 155 Photograph of Cell Lysing Blockage

Figure 155 (Left) shows early accumulation of K562 cell debris, as flow continues more debris collects until a complete blockage results (Right). This fault condition was repeated several times and each time the blockage occurred in roughly the same place, between the Y junction and electrode V3. In all cases an electric field in the region of 700 V/m was required.

8.5.1.1 Lysed Cell Detection Method

Impedance spectroscopy is investigated as a test method for detecting lysed cells, Figure 156 and Figure 157 show that magnitude is most discriminatory between the faulty and fault-free conditions, changing two orders of magnitude compared to the phase measurements which show a very similar response, between the fault-free and faulty experimental condition.

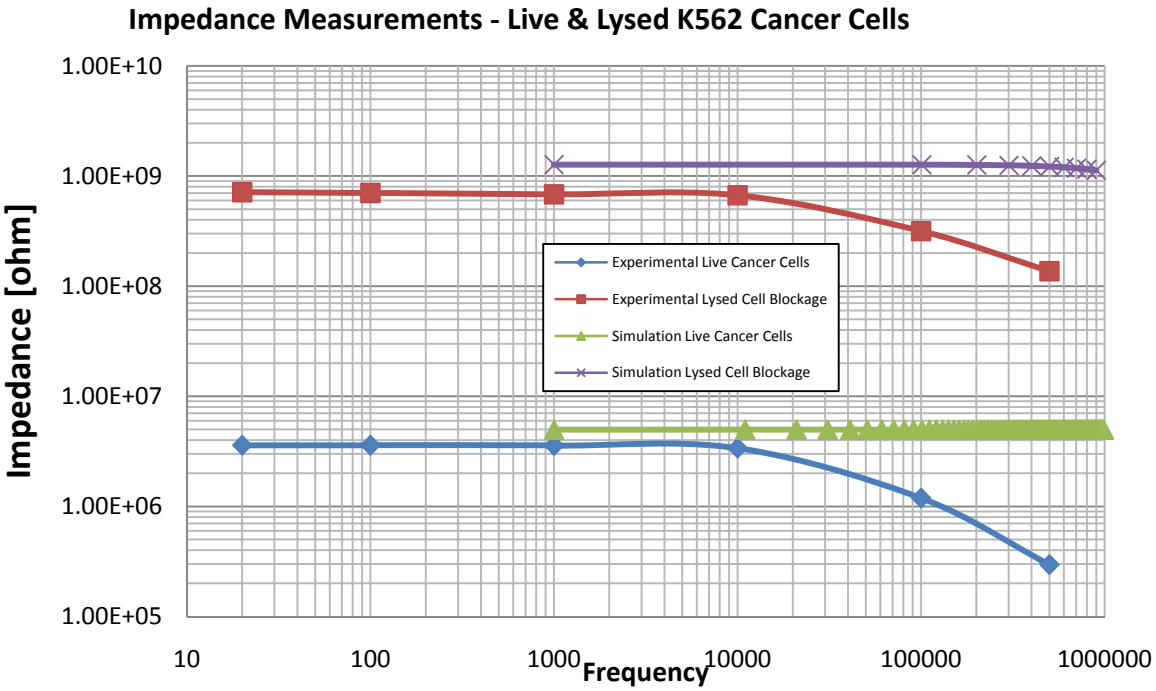


Figure 156 Fault-Free & Faulty Cancer Cell Impedance

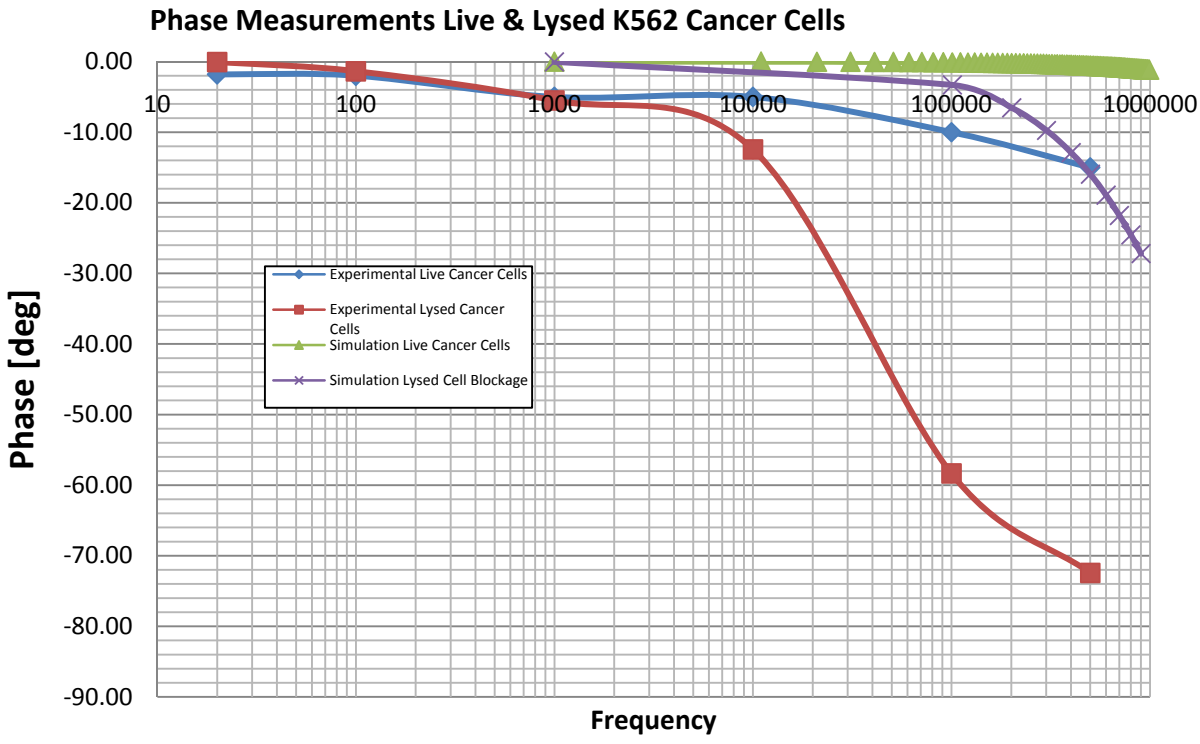


Figure 157 Fault-Free & Faulty Cancer Cell Phase

8.5.1.2 Lysed Cell Fault Model

In Chapter 4 the concept of the Fault Block, which could be configured to describe various fault types, was introduced. It may be assumed, backed by evidence from lysed cell experimentation that this blockage may be described as a complete blockage, given enough time this fault will continue to build until the channel is completely blocked and EOF ceases.

Chapter 4 described how a section of surrounding reactor material had to be placed around the channel for the impedance response to remain consistent with experimentation. However, using this collection of application modes, such an approach seems to be limited by the capabilities of COMSOL. One may recall that the inner channel forms a union with the outer surrounding material, this results in the inner boundary conditions been described as "walls" with is perfectly adequate for hydrodynamic applications, however, for describing electro-osmotic flow the "electro-osmotic velocity" boundary condition must be applied, but is not available when using this approach. Therefore, the surrounding material approach is not implemented (COMSOL have been informed this is a problem). It should be recalled, that this technique was used to accurately describe the impedance over the full frequency sweep to maintain accuracy at higher frequencies reaching the capacitive reactance of the geometry impedance.

Since in this chapter our interest is in using the design, simulation and test methodology for practical implementation, a fixed frequency by which to measure the impedance magnitude and phase will be used, simplifying the test hardware. From previous chapters and the results shown in Figure 156 and Figure 157 a frequency of 10 kHz shall be implemented. This permits that the surrounding material may be excluded and obviates the requirement for a parametric solver. Since limited information is known about the mapping of a lysed cell blockage to a Fault Block, then the fault model physics will be that of air (as a starting assumption). Cross-validation between this assumption and experimental results will determine whether this is a valid assumption.

Figure 158 Lysed Cell Fault Block located in most common location

8.5.2 Lysed Cell Simulation Cross-Validation

The experimental impedance measurements for live cancer cells and lysed cancer cells are cross-validated to fault block simulation results, Figure 156 and Figure 157. As previously described the surrounding reactor material has been neglected in this series of simulations, therefore, capacitive roll-off is not described, shown for both the faulty and fault-free impedance results. Furthermore, our interest is a measurement at a fixed frequency, 10 kHz. The fault block magnitude results are within 10% of the experimental measurements for both fault-free and faulty conditions, at 10 kHz.

The phase measurements and simulation results show similar characteristics, Figure 157 and remain the same order of magnitude. The fault-free phase results show little roll-off and at 10 kHz the difference between experiment and simulation is 75%. The fault responses show roll-off, with the experimental results demonstrating greater roll-off attributed to the capacitance of the system, not described in the simulation, other than through the blockage capacitance alone. Despite this the difference at 10 kHz is 83%, again maintaining the same order of magnitude.

8.6 Electrode Degradation

Electrode degradation in is a common problem in many electro-chemistry systems (Chapter 2). Given this evidence, microfluidic systems using electrodes, such as in electro-osmotic flow would be susceptible to the same issues. In this sorting application, switching is the key mechanism which constitutes its performance. Poor switching or incorrect switching greatly degrades the system performance.

In this system simulation switching is implemented as described for the Sun model in Chapter 3, however, here the global equations are extended to include the three electrodes; V2, V3 and V4.

```
R1 = R1on+R1off*flc1hs(t-0.5,0.01)

I1 = (V0-Volt1)/R1

V0 = 0

R2 = R2off-(R2off*flc1hs(t-0.5,0.01))+R2on

I2 = (V0-Volt2)/R2

R3 = R3off-(R3off*flc1hs(t-0.5,0.01))+R3on

I3 = (V0-Volt3)/R3
```

Figure 159 Global Expressions for Switching

The dynamic electrode ON and OFF resistance of each electrode is provided as a variable. Since this system switches only one channel at any one time, the electrodes may be considered as pairs, for example, the switching voltages Volt1 and Volt2 will be implemented on the boundary conditions, while Volt3 will be made insulation. Then Volt1 and Volt3 will be implemented while Volt2 is made an electrical insulation, this simplifies the Heaviside switching functions.

The success of a switch is determined by how quickly and completely the velocity is re-directed from one channel to the other, a slow switch allows cells to continue flowing into the incorrect channel for a given time, since the switching speed has not been specified for this application, the velocity alone is investigated. A switch may be considered complete, when for the ON condition the velocity has reached 90% and for the OFF condition 10%, similar to the specification proposed by Sun *et al.*

Figure 160 (Left) shows a clean velocity switch from the side channel (left) to the main channel; grey scale map darkest colour 0 velocity and lightest colour full velocity. Figure 160 (Right) shows how through electrode degradation the On and Off resistances of the electrodes are very similar, therefore, even with one electrode on (the main channel) and the other electrode off (the side channel) the velocity is split between the main and side channel. In a practical application this would cause the approximately 50% of the cells to flow into the waste and 50% into the side channel collection reservoir. This would be deemed a catastrophic failure.

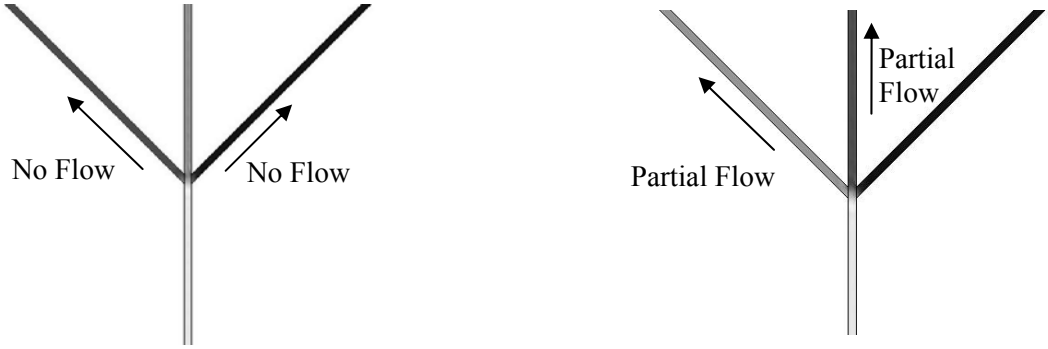


Figure 160 Simulation results; (Left) Good Main Channel Switch, (Right) Faulty Main Channel Switch

In the case described, the On resistance has increased from 1 Ω to 100 MΩ, a comparable value to the dynamic off resistance of the electrode. A study of the velocity profile, Figure 161 along the main channel shows how, when the electrode correctly switches a transition from full velocity, (80µm/s) to zero flow occurs; however in the faulty case the OFF flow is only reduced to 40µm/s.

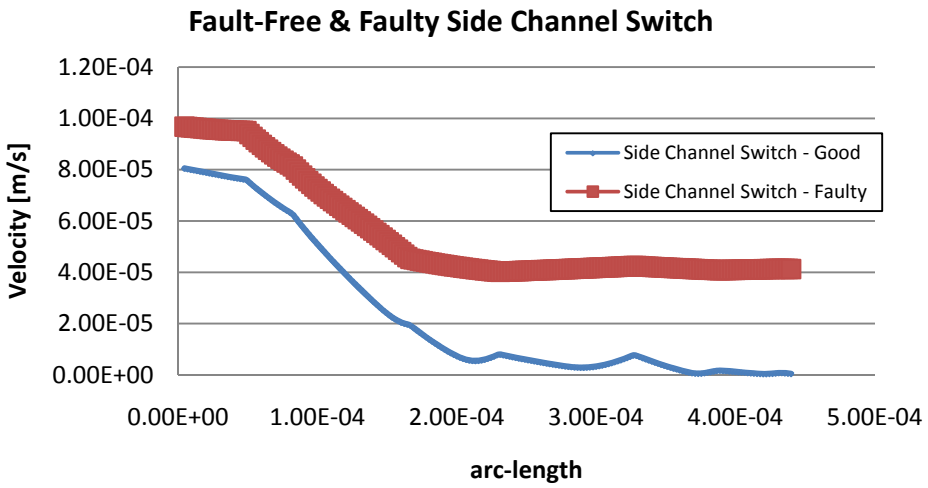


Figure 161 Velocity Profile plot of the side channel with Faulty and Fault-Free Switching

The same is true of the main channel, Figure 162, when the electrode has sufficiently degraded then it is unable to reach full flow velocity.

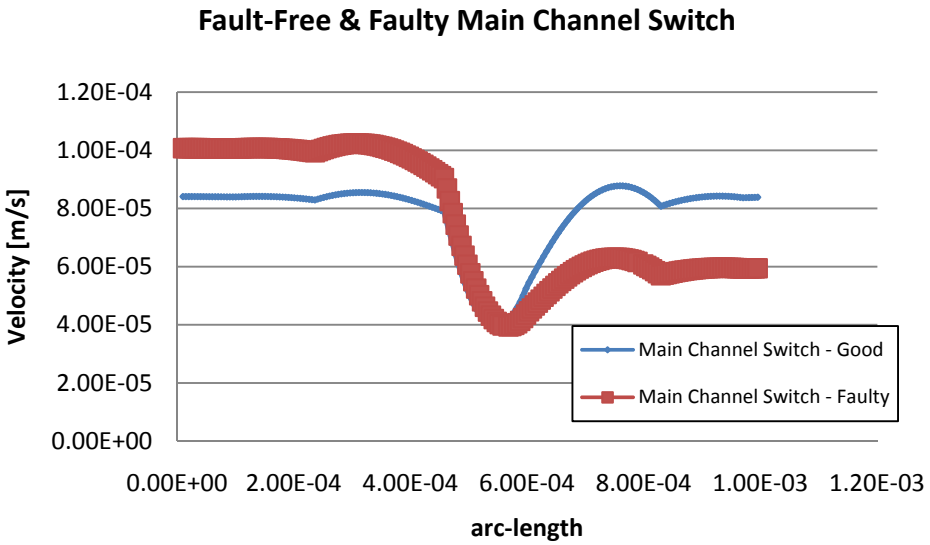


Figure 162 Profile plot of the main channel with Faulty and Fault-Free Switching

The controlling resistances Ron and Roff are constants within the simulation environment.

8.6.1 Electrode Degradation Fault Model & Detection Technique

The fault model for electrode degradation utilises an electrode fault block and manipulates the conductivity value of this fault block through the hierarchical namespace, Figure 163.

```
parameter1 = normrnd(1e10, (1e10*0.25));      %Electrode 1 Degradation ON
s = sprintf('%10.2d',parameter1);
femt.const{2} = s;
FFParaLog(FF,1) = parameter1;
FFCond(FF,1) = {'Electrode 1 Resistance'};
```

Figure 163 Electrode Degradation Fault Model Script

A fault block (0.1mm) thick is placed at each electrode point; V2, V3 and V4, Figure 164 and mimics the electrical properties of a real system electrode, namely its resistivity. Resistivity is given by, Equation 32 where R is the electrode resistance, A the cross-sectional area of the electrode and L the thickness (length) of the electrode. In the theoretical electrode implementation described in this chapter, the area of the electrode is that of the channel 100µm^2 and the thickness is 0.1mm.

$$\rho = \frac{R.A}{L}$$

Equation 32 Resistivity Equation

Impedance spectroscopy has been used to measure and monitor electrode degradation [125]Therefore, it is proposed that such a technique should be investigated in this application. The Electric Currents application mode is used to facilitate this test method in the simulation environment implemented using a parametric solver as previously described in earlier chapters.

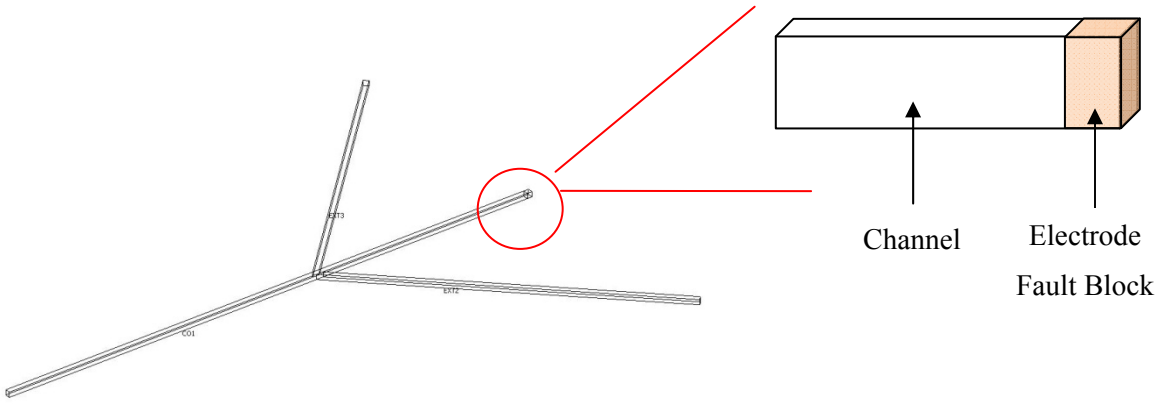


Figure 164 Electrode Fault Block Diagram

The common “ground” electrode is in position V1 for impedance spectroscopy measurements; electrodes V2, V3 and V4 may then be scanned in turn to test the system.

The coupling between the electrode resistances in the other two application modes, mmglf and emdc, and the electric currents application mode is by calculating the resistivity of the electrode based on

its resistance, Figure 165. The reciprocal of resistivity is conductivity, therefore the reciprocal may be added to the sub-domain conductivity for that electrode fault block, thus when the electric currents application mode is solved for the Z11 matrix (impedance) between a port (V2, V3 or V4) with a fixed current and the ground electrode the resulting measurements will be influenced by the resistance of the electrode.

```
ResistivityV1 = (R1*1e-8)/1e-4

ResistivityV2 = (R2*1e-8)/1e-4

ResistivityV3 = (R3*1e-8)/1e-4
```

Figure 165 Resistivity Calculations

8.6.2 Electrode Degradation Simulation Results

Impedance simulation for a "healthy" electrode having an ON resistance of 1Ω and an OFF resistance of 1^{10} Ω is simulated and compared to the "Degraded" electrode having an ON and OFF resistance of 1^{10} Ω. The magnitude bode plots are shown in Figure 166.

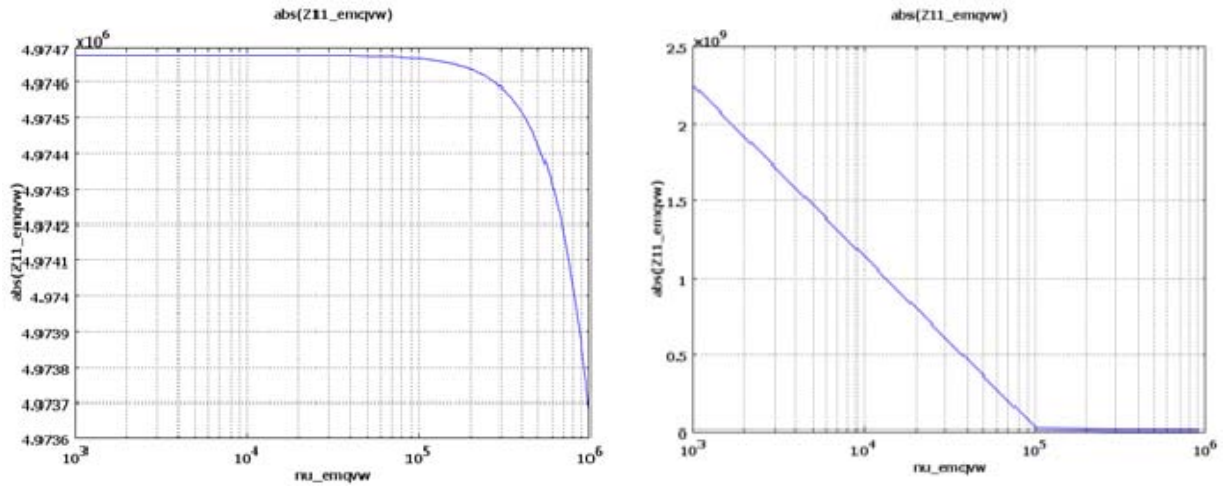


Figure 166 Impedance Magnitude Plots; (Left) Healthy Electrode (Right) Degraded

The bode plots in Figure 166 show 3 orders of magnitude between the healthy and degraded electrode conditions, at the lower frequencies (<10 kHz). The healthy electrode maintains an impedance of 4.9M Ω for all frequencies (DC – 1MHz). The degraded electrode measures $1x10^9$ Ω at the DC frequencies and drops down 2 orders of magnitude to $5x10^6$ Ω when the frequency is above 500 kHz. This shows that impedance spectroscopy is a valuable test method for measuring electrode degradation and is most sensitive at lower frequencies.

8.7 Fault Model and Test Method Conclusion

Two new fault models have been investigated and mapped to the fault library; the lysed cell blockage and electrode degradation, further demonstrating the flexibility and preliminary accuracy of the fault block method. These models have been cross-validated, but with not enough detail to fully ratify their behviour at this stage. This chapter is a demonstration of the methodologies ability to be applied to new systems, and the fault library extended. Further work would be required in mapping the behavior of the complex scenarios which the lysed cell could exist within and describing its environmental dependancies. Moreover, the electrode degradation behavior has only been described at either of its extremes; a healthy electrode and an electrode so severly degraded it may be considered an open circuit. Impedance spectroscopy has continued to show be a senstitive measuring technique for microfluidic system behavior.

8.8 System Fault Library

The system comprises 3 key important parameters to enable the system to perform to "specification", described in Table 21. Table 22 presents the fault library for the system which will be injected and simulated.

System Parameter	Value
Electric Field Strength	700 [V/m]
Electrode ON resistance [nom.]	1 [ohm]
Electrode OFF resistance [nom.]	1e10 [ohm]
Drive Electrode Voltage	100 [Vdc]

Table 22 System Nominal Parameters

Impedance spectroscopy test hardware will be deployed to measure the system response at a fixed frequency of 10 kHz. Impedance measurements will be made sequentially between V1 and V2, V3 and V4 in turn. This will cover all the complete system architecture and provides an example of using functional electrodes as a means of implementing test. For the sake of completeness it may be assumed that the impedance measurements are not biased upon the high voltage DC electro-osmotic voltage but measured when this voltage is not present.

Fault Number	Fault Type	Fault Model	Specification Tolerance
1	Electrode 1 Degradation ON	R1ON = 1e10	+/- 25%
2	Electrode 2 Degradation ON	R2ON = 1e10	+/- 25%
3	Electrode 3 Degradation ON	R3ON = 1e10	+/- 25%
4	Drive Voltage	V1 = 0V	+/- 5%
5	Drive Voltage	V1 = 200V	+/- 5%
6	Cell Blockage Ch.1	Fault Block Ch.1	N/A
7	Cell Blockage Ch.2	Fault Block Ch.2	N/A
8	Cell Blockage Ch.3	Fault Block Ch.3	N/A

Table 23 Fault Conditions & Fault Model

8.9 Implementing the simulation algorithm

The simulation algorithm was implemented as described using the nominal system parameters in accordance with the specification shown in Table 23. Impedance measurements were made between electrodes (V1 and V2, V3 and V4) with the frequency fixed at 10 kHz. Faults were injected as described in Table 23 their fault models have previously been described. 100 simulation iterations were carried out per fault condition and for the fault-free condition. Impedance measurement results were stored and test analysis performed as described in Chapter 5 and demonstrated in detail in Chapter 6, the test outcome probabilities are shown in Table 24. Full simulation algorithm script may be found in Appendix V.

	Fault-Free as Fault-Free	Faulty as Faulty	Yield Loss	Test Escape
1 (Electrode 1 Degradation ON)	0.9973 0.9973 0.9973	0.935 $1x10^{-9}$ $0.8x10^{-9}$	0.0023 0.0023 0.0023	0.06 0.983 0.984
2(Electrode 2 Degradation ON)	0.9973 0.9973 0.9973	$1.1x10^{-9}$ 0.921 $0.71x10^{-9}$	0.0023 0.0023 0.0023	0.994 0.074 0.984
3(Electrode 3 Degradation ON)	0.9973 0.9973 0.9973	$1.8x10^{-9}$ $2.1x10^{-9}$ 0.978	0.0023 0.0023 0.0023	0.983 0.982 0.01
4(Drive Voltage ON)	0.9973 0.9973 0.9973	$1x10^{-4}$ $2x10^{-3}$ $4x10^{-3}$	0.0023 0.0023 0.0023	0.98 0.985 0.981
5(Drive Voltage OFF)	0.9973 0.9973 0.9973	$3x10^{-4}$ $4x10^{-3}$ $4x10^{-3}$	0.0023 0.0023 0.0023	0.983 0.979 0.984
6(Cell Blockage 1)	0.9973 0.9973 0.9973	0.91 $3.7x10^{-5}$ $8x10^{-6}$	0.0023 0.0023 0.0023	0.085 0.984 0.98
7(Cell Blockage 2)	0.9973 0.9973 0.9973	$5x10^{-3}$ 0.97 $1x10^{-4}$	0.0023 0.0023 0.0023	0.978 0,002 0.98
8(Cell Blockage 3)	0.9973 0.9973 0.9973	$8x10^{-4}$ $4.2x10^{-5}$ 0.967	0.0023 0.0023 0.0023	0.981 0.985 0.07

'mag 1 (abs(Z11_emqvw))' 'mag 2 (abs(Z11_emqvw))' 'mag 3 (abs(Z11_emqvw))'

Table 24 Test Outcomes

8.10 Test Analysis

Test analysis for this system is much simplified over the "Y" channel system in Chapter 6, because only 3 impedance test sensors have been implemented at a fixed frequency. From the results in Table 24 it may be observed that impedance spectroscopy is very good at detecting electrode degradation and cell blockage faults, but only by the test method associated with the channel where the fault has occurred. Impedance spectroscopy has low probability of detecting faults occurring in adjacent channels. As such impedance spectroscopy has to be implemented per channel. Conversely, electrode drive voltage faults are not detected by this test method, resulting in a high test escape, however, a simple voltage monitoring method could be implemented to detect such faults, complimenting the impedance spectroscopy method for other faults.

8.11 Discussion & Conclusion

This chapter has demonstrated the methodologies ability to adapt and operate with various *unseen* systems; in this case a microfluidic flow cytometry system. This system is based on the electro-osmotic flow transport mechanism previously described in Chapters 3 and 4, however, in this system two new faults are considered; a lysed cell and electrode degradation. The fault library is shown to be able to facilitate the addition of these two new fault mappings and the fault block method shown to be ever adaptable to new conditions, further highlighting its universatality and strength.

Impedance spectroscopy is the only test method considered due to its sensitivity to blockage conditions and intrinsic ability to measure resistance (electrode degradation). Test analysis shows impedance spectroscopy to be sensitive to electrode degradation and cell blockages (>90% correct determination), with little sensitivity to electrode drive voltage variation.

8.12 Design Optimisation – Assessing the next generation of device

8.12.1 Introduction

This section presents work carried out in COMSOL to analyse a new design of reactor and method of operation for this application. The new method is based on an electro-osmotic pump (EOP). The idea of the EOP is that the electric field to create EOF remains localized across a chamber containing firits (to increase surface area) and therefore the electric field is not across entire channels, which can potentially degrade or destroy biological samples, has have been shown by the lysed cell fault condition. A further advantage is that much smaller electric voltages can be applied to generate the same electric field.

While this work uses the EOP based chip it does not implement frits in the simulation. It was felt that the basic performance, pumps and switching would be analysed before improving the EOP performance using frits. The purpose of this study is to ensure the basic design produces hydrodynamic flow, and any design flaws or further design or control considerations identified.

The section addresses the design aspect of the methodology, and is described for completion rather than a mandatory part of our test methodology.

8.12.2 Geometry

Figure 167 is the proposed EOP cell sorting geometry produced by the microfluidic design team working on this project. The CAD plot (Figure 167) masked has been marked up to show identifiable features.

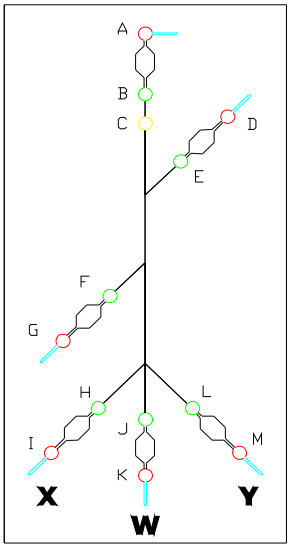


Figure 167 New EOP Reactor Geometry

The centre channel has been marked “W” to include “waste” or un-switched cells, “X” and “Y” are possible channel switching options.

8.12.2.1 Simulation Geometry

The simulation is implemented in 2D, and simplified by the removal of the side channels. The complimentary electrode to each pair has been removed to increase simulation efficiency. Since the design is going to be etched to 50µm deep, then planer dimensions have been increased by 100µm. This makes the channel width approximately 150µm wide, Figure 168.

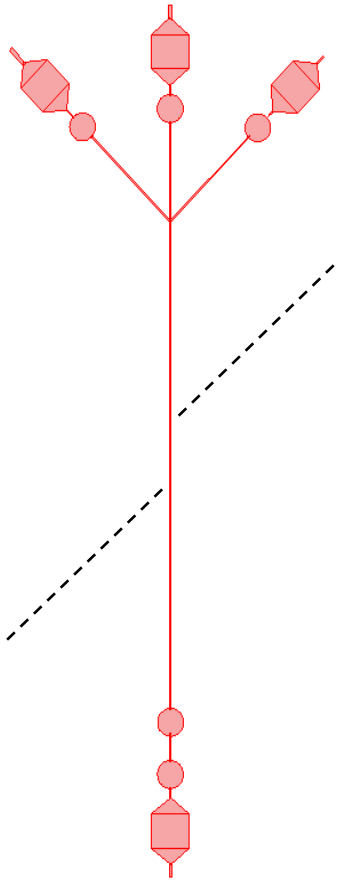


Figure 168 EOP COMSOL Geometry (dotted lines show removed channels)

The simulation of this geometry is the same implementation as for the Sun *et. al.* system, using the same two application modes, Stokes Flow (mmglf) and the Conductive Media DC (emdc). The impedance spectroscopy application mode is not used in this section. The parameters (constants) describing this simulation system are presented in Table 25.

The conductivity and zeta potential have changed value to those which the microfluidic design team would like to test for this new system.

Parameter	Value
Density	1000 [Kg/m^3]
Viscosity	1e-3 [Pa*s]
Conductivity	175 µS/cm
Zeta Potential	0.1 [V]
Positive Potential Voltage, V1	100 [V]
Temperature	293 [K]

Table 25 EOP Constants

8.12.2.2 Electric Field Simulation

In this simulation the Electric Field [V/m] is studied. V1 is applied to electrode A and electrode B is connected to ground (Figure 167). All other boundaries are electrically insulated. From the plot it may be observed that an electric field is developed across the EOP.

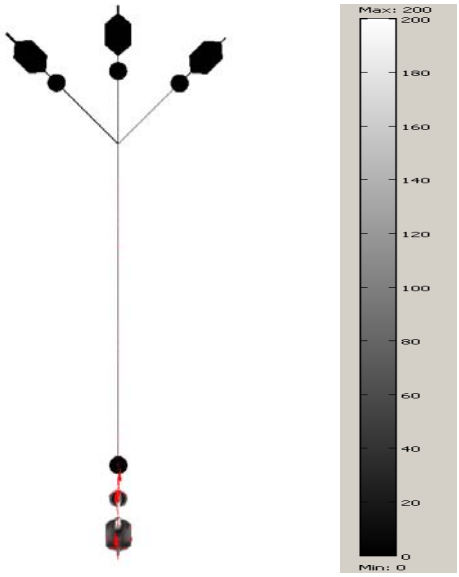


Figure 169 EOP Electric Field Simulation Slice Plot (V/m)

The electric field slice plot, Figure 169 shows that the electric field is localized only across the EOP chamber and not across the reactor channels.

8.12.2.3 EOF Simulation

This simulation proves that an electro-osmotic flow is produced from the EOP. In this case the flow is shown between C and the switching junction, shown in the velocity profile, Figure 170. The flow is towards the switching junction.

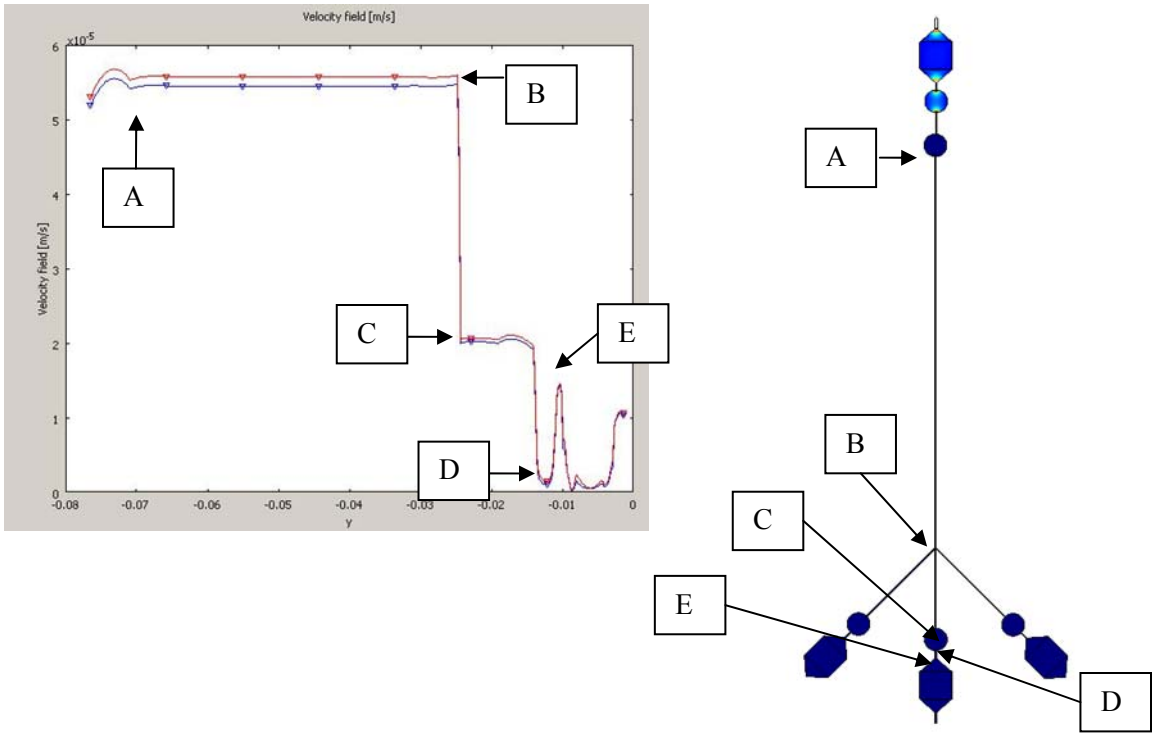


Figure 170 EOF Flow Profile for EOP

The boundary conditions for the X and Y switching channels were modeled as "open" for the flow profile shown in Figure 170. A complimentary flow profile is shown in Figure 172 when these boundaries modeled as "walls". The reason for these two approaches is that electrodes will be placed in these reservoirs in the experimental system. However, at this stage it is not known what seal they will form so both extreme conditions have been modeled to observe the effects on this system is flow performance.

The volumetric flow rate at electrode C (boundary 35), straight after the EOP outlet, and at the switching junction (boundaries 160,168) was obtained using boundary integration to determine the change in volumetric flow rate, Figure 171.

Value of integral: 3.146951e-7 [m^2/s], Expression: U_mmglf, Boundary: 35 **(electrode C)**
Value of integral: 9.268899e-9 [m^2/s], Expression: U_mmglf, Boundaries: 160, 168 **(switching junction)**

Figure 171 EOP Volumetric Flow Rates

The velocity profile for when X and Y are wall conditions is shown in Figure 172. The same feature marking scheme is used.

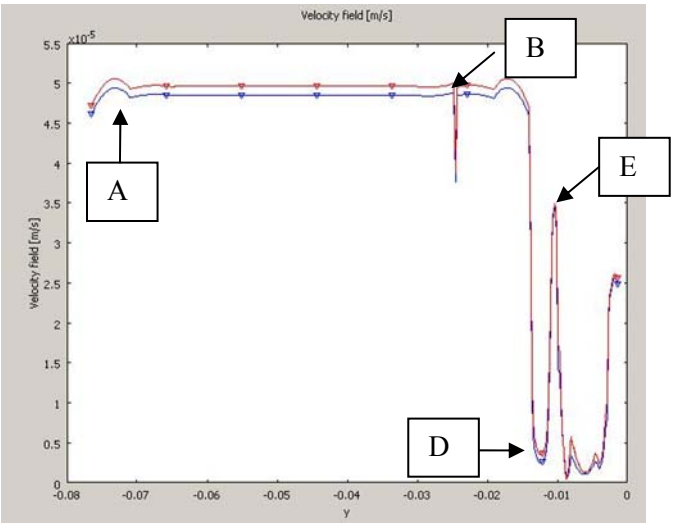


Figure 172 EOP Velocity Profile (Walls)

From Figure 172 there is no C feature because due to the walls there is no drop in velocity as flow escapes to the open boundaries.

The two plots share similar features. The first noticeable feature is the flow rate drop at the switching junction. For the wall boundary condition this is momentary, but in the open boundary simulation the flow rate permanently drops to a new level as some of the flow escapes. The flow then dips again as it crosses electrode J, recovers in the short channel to the EOP chamber, drops through the EOP chamber and then recovers in the channel to the outlet.

8.12.2.4 Dual EOP Simulation

In this simulation the inlet EOP is operated as before alongside the outlet EOP on the main channel. Figure 173 uses flow arrows to show that both EOP's work in the same direction with the following electrode configuration; A & J (+V) and B & K (0V). The idea behind this configuration is to pump from the inlet and pull to the outlet, the reasoning is that the design team were not sure whether sufficient hydrodynamic driving velocity would be created with just the inpet EOP pumping.

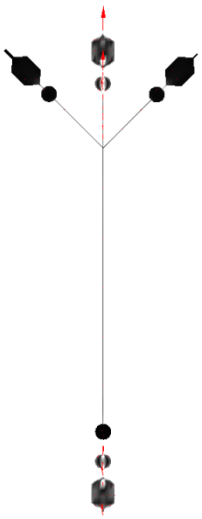


Figure 173 Dual EOP EF Slice Plot

The same flow profile analysis as performed in simulation 2 was performed here, to observe the effect of the dual EOP. The same features as in simulation 2 can be identified, Figure 174.

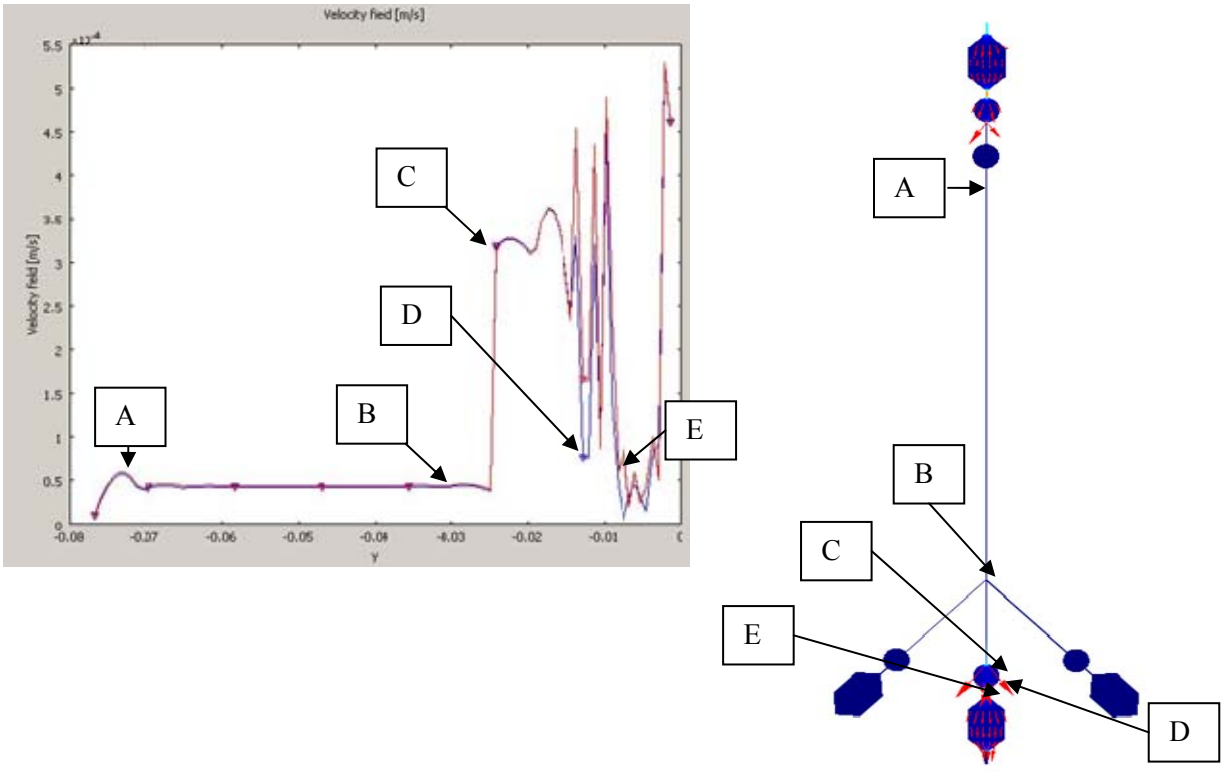


Figure 174 Dual EOP Velocity Profiles

8.12.3 EOP Flow Summary

The EOP generates flow. However, the dual EOP does not increase the flow in the main channel just at the outlet. One of the reasons this may be the case is that the outlet EOP may draw fluid from the X and Y switch channels, rather than drawing fluid from the main channel and increasing the total through flow. The open and wall boundary simulations determine the best and worst case scenarios for flow, depending on the physical experimental electrode implementation.

8.13 Un-Switched Leakage Simulations

This simulation investigates the leakage from the main channel into either of the switched channel X and Y. This is important in sorting applications, as it helps determine possible sorting errors. The simulation analysis discussed is when only the inlet EOP is on.

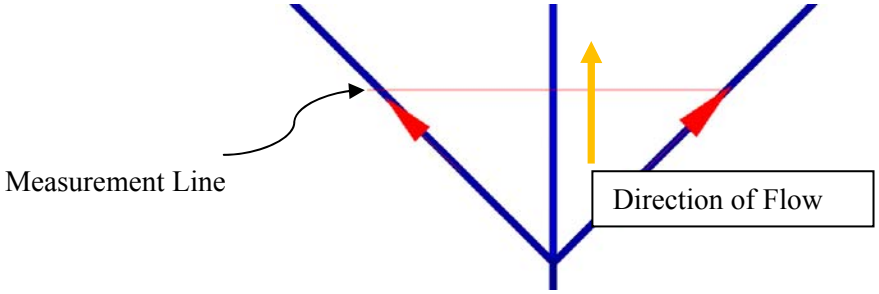


Figure 175 Un-Switched Leakage Slice Profile

The measurement line in Figure 175 indicates where the cross-sectional flow measurement is taken.

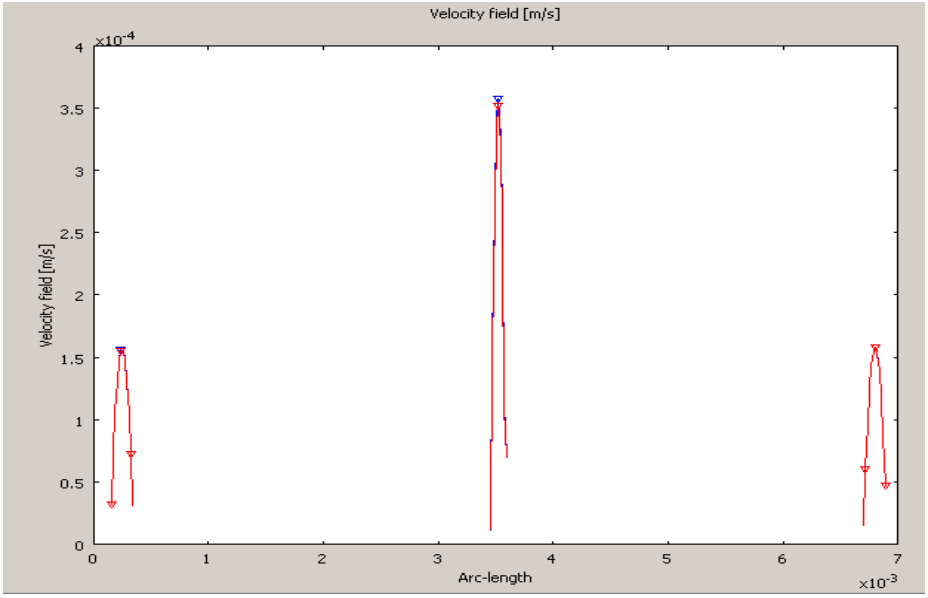


Figure 176 Un-Switched Flow Profile (Open)

Figure 176 shows that the majority of flow is down the main channel towards the outlet, approximately 3.5 $x10^{-4}$ m/s peak. However, with open boundary conditions at the X and Y electrodes then flow occurs in both un-switched channels, at 1.6 $x10^{-4}$ m/s peak, this is approximately half the main channel flow. This leakage situation may be improved with closed (wall) boundary conditions as shown in Figure 177. Observing this plot shows a decrease in main channel velocity, this is due to increased pressure within the system, with fewer open boundaries.

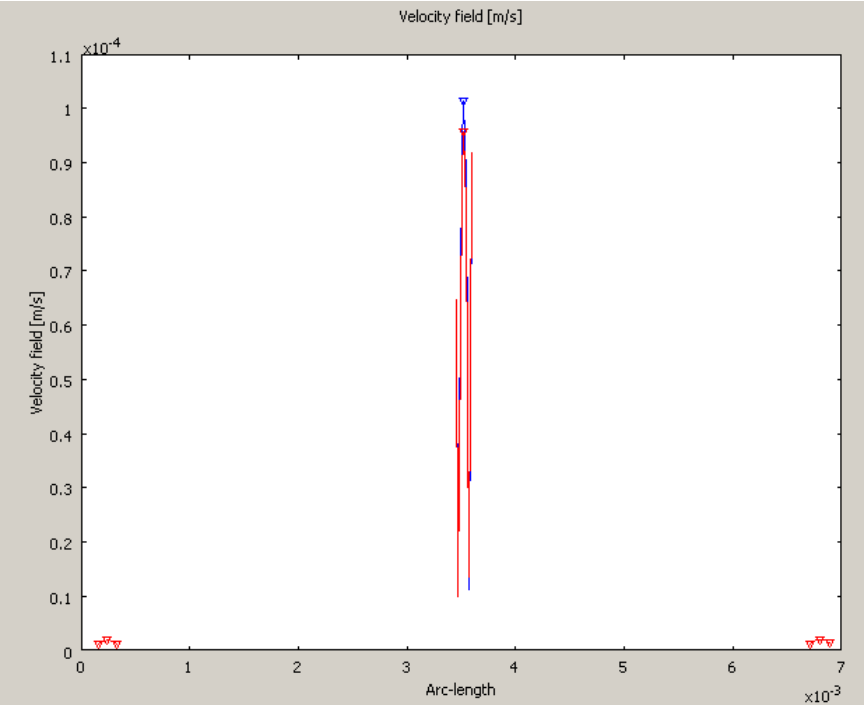


Figure 177 Un-Switched Flow Profile (Walls)

8.13.1 Switched Leakage Simulations

This simulation investigates the effectiveness of the reactor's ability to switch the flow to implement a sorting mechanism. For this the inlet EOP is used to cause main channel flow and sorting channel Y EOP is used to create a suction (L = +V, M = 0V).

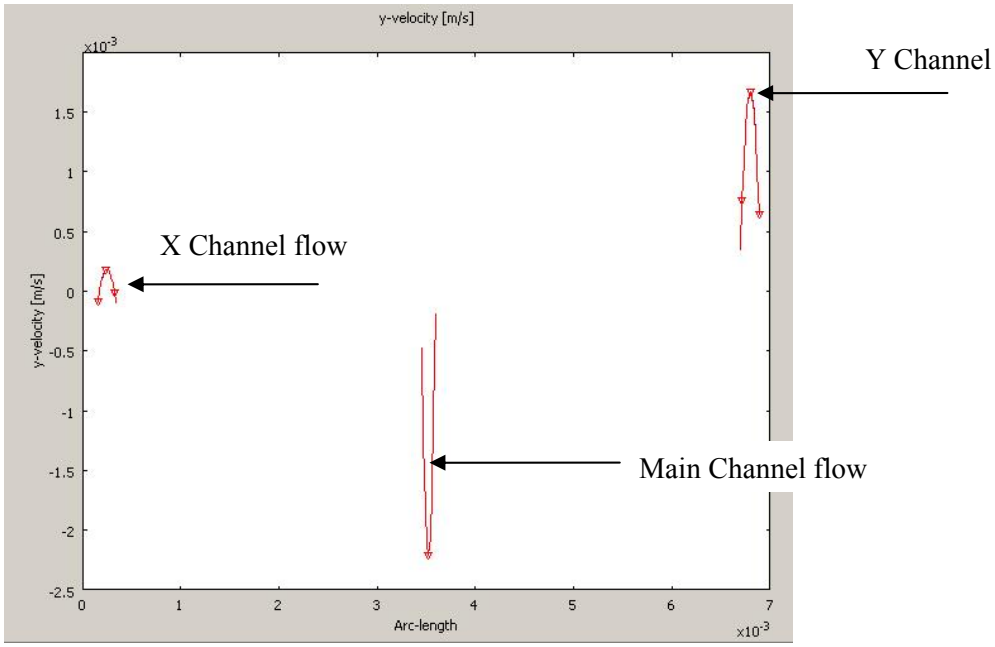


Figure 178 Switched Flow Profile

Study of the velocity plot shows that the EOP does create a flow in the Y channel as desired, however, the main source of this fluid is from the outlet reservoir. This can further be seen in the surface arrow plot below.

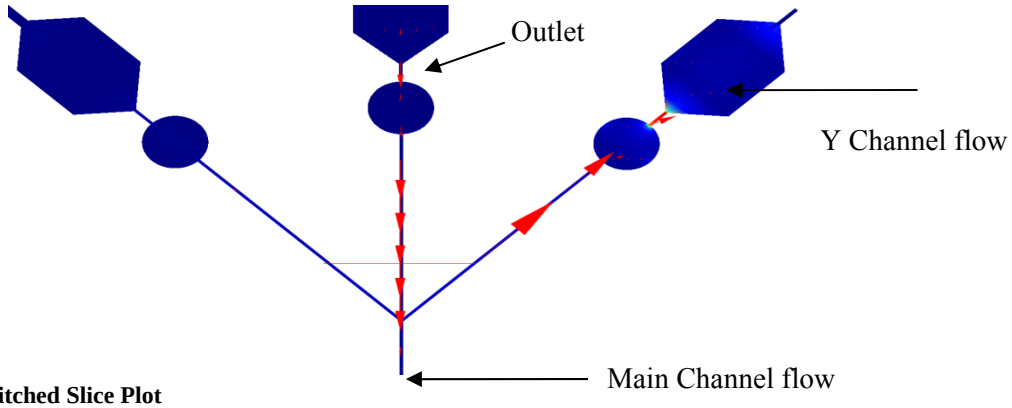


Figure 179 Switched Slice Plot

The proportional arrows from the main channel are negligible compared to the reverse flow arrows from the outlet.

8.14 Switching Conclusions

There may be leakage problems depending on how likely the electrodes are to creating open or wall conditions, as shown in simulation 4. The main concern is how successful the EOP's will be at sorting the passing cells. From the results of simulation 5, the EOP for X and Y channels have greatest suction effect from the outlet than the main channel, which means that this method may not be effective for switching cells transiting the main channel, and worst still could draw "waste" cells into either of the switched reservoirs creating an erroneous count.

8.15 Discussion and Conclusion

This section of the chapter investigates a brief overview of the potential design work of a new reactor. The approach demonstrates the level of investigation COMSOL provides in assessing a new design such as velocity profiles and electric field distribution, far superior to the level of investigation offered by experimental work alone.

Chapter 9 Conclusion & Further Work

9.1 Introduction

Microfluidics systems (Lab-on-a-Chip) are set to address some of the most important issues of modern life, reflected by growing research activity into the science. The volume of research is currently disproportionate to the commercial exploitation of microfluidic systems, as researchers hunt for the "Killer Application" [27] unaided by the ever increasing "gap" between microfluidic research and microfluidic industry [28]. One of the contributing factors to this lack of commercial system is the lack of Design for X, where X is currently Manufacture and Test.

The work presented in this thesis details the requirements of a Microfluidic Design, Simulation and Test Methodology. The work presented has resulted in a functioning Microfluidic Fault Simulator and Test Analysis methodology with application to several case studies.

Microfluidic simulation is widely underutilized in the research community and has received encouragement for its adoption by the Royal Society of Chemistry [29]. Contributing factors to the low rate of adaption are the alien work environment and distrust in the simulated results. In this thesis we have targeted three application areas; hydrodynamic flow, diffusional mixing and electro-osmotic flow, using a combination of experimentation and published work to cross-validate the results. The FEM simulation is an accurate tool by which to investigate Microfluidic system design and functionality.

For the first time microfluidic faults have been modelled using a low abstraction FEM technique using a generic approach, the "Fault Block", with support from parametric faults or a combination of the two depending on the fault condition. An evolving list of microfluidic faults covering a wide range of applications has been compiled and low abstraction behavioural fault models developed and cross-validated to experimental and published work, with a high degree of accuracy, typically less than 14% and 80% for test methods.

Two microfluidic test methods have been proposed, impedance spectroscopy and Levich electro-chemical sensors, based on their usefulness as discrete, highly integrated functional sensors in the literature. Methods of how to implement them in a system simulation have been determined. Impedance spectroscopy has been found to require a separate Electric Currents application mode and parametric solver, whereas Levich sensors make use of the Convection and Diffusion application mode and their

current values are evaluated from global expressions. Their simulations have been cross-validated against experimental and published work, respectively.

With a high degree of confidence in the fault-free system simulations and a range of mapped fault models and a Microfluidic Fault Simulator, MFS, algorithm has been developed to provide fault-free simulations, and fault simulations by fault model injection into the fault-free simulation model. This algorithm requires no system abstraction, and uses the COMSOL scripting interface to manipulate the original system model, making it highly integratable into the design workflow. Furthermore, the MFS evaluates applied test methods and records measurement data for analysis by the test analysis scheme. Since no fault statistics are available for microfluidic faults the algorithm implements faults on a fixed basis, type, location and likelihood, however, the MFS has provision when such statistics become available.

The test analysis algorithm uses measurement data from the MFS to assess the applied test methods and assign each a probabilistic outcome, based on the traditional test hypotheses, however, here for the first time applied to microfluidic systems. This highlights the methodologies objective of being integratable into existing workflows.

While the test analysis algorithm provides test metrics based upon the measured data from the MFS, this data is able to automatically generate practical test schemes through the use of decision tree analysis. Dedicated test instrumentation in the form of an impedance spectrometer have been developed to support the practical implementation of the classification rule set and provide an embedded means of measuring the wide range of impedances experienced between fault-free and faulty microfluidic systems. With the same aim a dedicated user friendly high voltage power supply was developed and has been used in the experimental investigative work. The hope for this power supply is that control sequences could be extracted from the simulation environment so that no additional programming would be required. This would allow the simulation methodology not only to optimise the reactor design, but to provide test analysis, generate test classification schemes and program control instrumentation.

In the final chapter the methodology is used on a new application to demonstrate its effectiveness. The application is a microfluidic cell sorting device for use with head and neck cancer cells, based on EOF transportation and switching. This not only demonstrates the methodologies ability to work effectively with new systems, but also demonstrates the continued flexibility of the fault block and fault library concept with the addition of two new fault conditions; the lysed cell and electrode degradation models.

9.2 Further Work

In order to improve on the work presented in this thesis and to advance the subject further, the following further work is proposed:

9.2.1 Observability & Controllability

In the work presented simple flow paths have been studied, that is each channel has had an inlet and outlet providing complete observability to the entire channel in which to implement a test method. Microfluidic systems are set to become hugely complex, a single device containing thousands of interconnecting channels and intra-connecting channels, especially for reactors designed for production scale tasks.

Therefore each channel will not have an accessible inlet and outlet by when to implement a test. This is not a new problem and has existed in system test from the beginnig of digital system test [30]. The beginning of a solution would be the parsing of the FEM geometry objects and boundary conditions to determine how many channels existed (geometry objects) and how many available inlets and outlets where available (boundary conditions), some form of test vector and routing algorithm would have to be applied to determine the most optimum positions for test sensors (electrodes) and controllability would have to be used to switched the correct routing to enable intra-channels to be tested. It is envisaged that impedance spectroscopy would provide an excellent test method due to the multiplexing of electrodes and channel paths in order to obtain observability and controllability of the system.

9.2.2 Fault Library

Important further work is to continue building the fault model library as more faults are discovered, more fault library entries made. It is important to keep a consistient mapping approach, such as using the Fault Block to simply the MFS algorithm; however, the use of the Fault Block should not limit the progression of the MFS. Another mapping method or unique fault models for particular faults may be required, in order to maintain fault description accuracy. More detailed studies of faults are required to describe the low order behavior which may result in more *physics* being added to the fault block description.

9.2.3 Simulation Efficiency

As previously discussed FEM simulation is not the most computational efficient form of system simulation, one of the reasons for the move to high order behavioural models. However, it is the most likely simulation technique for microfluidic researchers. In this thesis simple device models with a couple (three at the most) application modes have been implemented and even in these cases simulation cycles

could take in the order of minutes. If the MFS is to be used to accurately gauge statistical distributions for analysis then a large sample set is required (> 1, 000). Therefore the length of each simulation cycle or the number of required cycles would have to be reduced. FMEA or concurrent test analysis would be two possible methods and are discussed in this section. One other method would be to abstract the model to create a high level behavioural model. A similar approach was investigated by Kerkhoff [ref], however, in this work the faul-free system model was abstracted and then faults applied, here the suggestion is to either abstract the faulty system model or the fault itself.

9.2.4 Application mode parsing

A system model would be loaded into the MFS and depending upon the application modes which it comprised the most likely fault conditions would be injected. For example, a system solely comprising Navier-Stokes equations would be assumed to be a flow system, therefore the most disruptive or likely faults would be considered to be blockages and as such fault models would be injected and described accordinging to the mapping of the fault library. Locations for the faults could be considered based on the most likely location or occurrence statistics (if known).

9.2.5 FMEA (Failure Mode Effect Analysis)

The implementation of FMEA is interesting and thought provoking for further work into fault statistics and simulation efficiency. FMEA could be used to assess the system to determine the most likely fault conditions and only the most likely fault condition(s) loaded into the MFS. This topic was briefly reviewed in Chapter 2.

9.2.6 Fault and Manufacturing Statistics

As manufacturing and Mean Time Before Failure statistics (MTBF) become available through commercial and volume manufacture; then these statistics would better guide the MFS to be more accurate and efficient in the fault conditions which are studied and simulated.

9.2.7 Simulation Concurrent Test Analysis

Implementing a simulation concurrent test analysis would analysis the simulation data per cycle to determine if a decision could be made with reasonable confidence. It would be used to serve two purposes, the potential reduction of computational simulation overhead and preliminary assessment of a test method. Test analysis was described Chapter 6; the procedure is to calculate the discrimination of a test method for a particular fault condition, post-simulation. Here it is proposed that the statistical discrimination (test outcomes) are calculated per simulation iteration. Therefore a test method which has a high discrimination for a particular fault could halt the simulation iteration count for that fault condition

and move onto the next condition. Conversely, if a particular test method was proving to have no or little discrimination and was producing high Type I errors (test escapes) then there would be little point in proceeding with another 1000+ cycles, therefore the simulation could be halted and the suggestion to try another test method or modify the observability and controllability scheme.

9.2.8 Reaction Engineering Lab

The detailed description of the chemistry or biology within the reactor has not been investigated in the scope of this thesis, although an add-on module by COMSOL exists to make these descriptions – "Reaction Engineering Lab". This allows the description of space dependant reactions to be simulated within the FEM system model. This would provide the ability to completely asses a microfluidic system. This would have links to the evolutionary fault modelling approach.

9.2.9 Evolutionary Fault Modelling

As described in Chapter 4 static fault modelling is used in this thesis, and in most published work on system fault injection. The problem with static fault injection is two-fold. Even in a low abstraction implementation, some form of abstraction has to be made and some assumptions about how to implement that fault as a model have to be made, therefore, some detail will be lost and the model is a result of the users interpretation. A more realistic route would be to set the system parameters which would cause this fault. For example, consider the failure path in an EOF system – electrode degradation causes electrolysis which produces hydrogen bubbles, these bubbles leave the electrode and enter the fluid stream, as more bubbles enter the stream they begin to become trapped eventually breaking the electric field and ceasing EOF flow. The static approach will be to insert that bubble fault model and observe system behaviour and make test measurements. The evolutionary approach would be to model electrode degradation and have coupled the electric field / current to an electrolysis equation which would form bubble models. While this approach would yield possible more information and be more true to an experimental system, the simulation overhead would potentially be huge and it still would suffer from abstraction at some stage of the failure path description.

9.2.10 Decision Tree Analysis

Decision tree analysis requires further work to improve on the misclassification rate. More simulation data may be required in order to improve training or the use of a more advanced algorithm. Decision tree analysis is a powerful and useful tool and should be investigated further as a post analysis tool to the MFS and test analysis procedures.

9.3 Publications & Presentations

9.3.1 Publications & Presentations

Myers, T.O. and I.M. Bell, Fault modelling and test development for continuous flow microchemical sensor systems, in Mixed-Signals, Sensors, and Systems Test Workshop, 2008. IMS3TW 2008. IEEE 14th International, IEEE, Editor. 2008, IEEE: Vancouver, BC. p. 1 - 6.

Myers, T.O. and I.M. Bell, Test Metric Assessment of Microfluidic Systems through Heterogeneous Fault Simulation, in Mixed-Signals, Sensors, and Systems Test Workshop, 2010. IMS3TW 2010. IEEE 16th International, IEEE, Editor. 2010, IEEE: Montpellier, France. p.1-6.

9.3.2 Posters

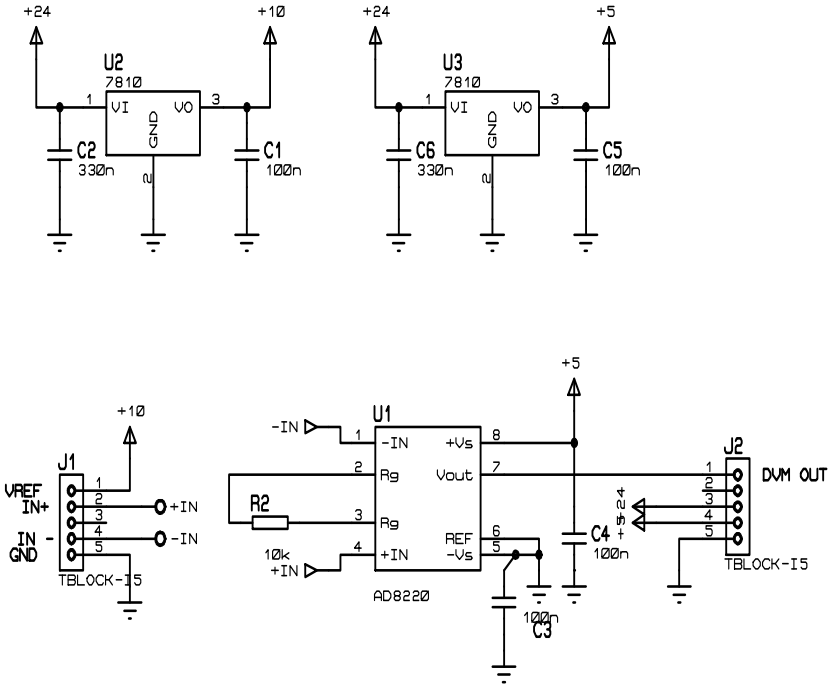
Bell, I.M., et al. An Automated micro-TAS detection system for the assesment of the aquatic pheromone Glutathione using ortho-phthaladehyde as a fluorescent dye. in Analytical Research Forum. 2008. University of Hull, UK: Royal Society of Chemistry. (Poster).

9.3.3 Unpublished work & Work under consideration

Microfluidic Test Review - Journal considered Lab-on-a-Chip - (in writing)

Microfluidic System Test Metric Assessment through Fault Simulation – JETTA (under review)

APPENDIX I – Entran Amplifier Schematic



APPENDIX II – "Y" Channel Simulation Script

```
%-------------------------------------------------------------------------
% Project:      Microfluidic Fault Simulator Code
% Author:       Tom Myers
% Date Created: 20/07/10
%
% Description:  Latest MFS code for PhD Thesis
%                For the "Y" Channel Simulation
%-------------------------------------------------------------------------
flbinaryfile = 'Y channel_IS_Flow.mph';

n = 10;            %How many runs for the fault free simulation
FaultStates = 3;   %How many faults to increment through

FaultNumber = 8;   %Starting Fault Number

TotalSimCycles = n*FaultStates;

disp(TotalSimCycles)

%FaultInjection = 0; %if 1 inject faults
                     %if 0 fault-free model

Nominal = 0;         %if 1 simulate with fixed value parameters
                     %if 0 simulate with nominal input parameters

FaultFreeFirst = 1; %if 1 then simulate nominal fault-free first
                    %if 0 then go straight into fault simulations

FaultFreeSims  = 50; %Number of fault-free runs


%=========================================================================
%Outer Simulation Loop
%NOTE:
%       One loop of the outer loop for each Fault State
%=========================================================================
if(FaultFreeFirst == 1)
    %=====================================================================
    % Fault-Free Simulations
    %=====================================================================
    for FF = 1:FaultFreeSims

        disp('Iteration Number')
        disp(FF)                 %Iteration loop number for sanity check
        disp(clock)             %[year month day hour minute seconds]

        %Record Start of Cycle Time
        FaultFreeCycleTime(FF,:) = clock;

                %=============================================================================
        %Save the FEM Structure
        %=================================================================
        % Geometry
        clear draw
        g2=flbinary('g2','draw',flbinaryfile);
        g5=flbinary('g5','draw',flbinaryfile);
        g4=flbinary('g4','draw',flbinaryfile);
        g3=flbinary('g3','draw',flbinaryfile);
        draw.p.objs = {g2,g5,g4,g3};
        draw.p.name = {'PT1','PT4','PT3','PT2'};
        draw.p.tags = {'g2','g5','g4','g3'};
        g1=flbinary('g1','draw',flbinaryfile);
        draw.s.objs = {g1};
        draw.s.name = {'EXT1'};
        draw.s.tags = {'g1'};
        fem.draw = draw;
        fem.mesh = flbinary('m1','mesh',flbinaryfile);

        femt = fem; %work on a copy of the structure - re-load purposes
```

```
%=========================================================================
% User Defined specification
%=========================================================================
%The 'normrnd' function is used to obtain a random value from within
%the normal distribution. This is created from the user defined
%sepcification, in the presence of no inductive fault statistics.
%=========================================================================

%Need to parse constants to find values, or variable names!
parameter1 = normrnd(300e-6, (300e-6*0.2)); %velocity1 - +/-20%
parameter2 = normrnd(300e-6, (300e-6*0.2)); %velocity2 - +/-20%
parameter3 = normrnd(297, (297*0.1));       %Operating Temp. +/-10%
parameter4 = normrnd(0.8, (0.8*0.01));      %Concentration 1 +/-0.01%
parameter5 = normrnd(0.2, (0.2*0.01));      %Concentration 2 +/-0.01%

s = sprintf('%10.2d[m/s]',parameter1);
femt.const{2} = s;
FFParaLog(FF,1) = parameter1;
FFCond(FF,1) = {'velocity1'};

s = sprintf('%10.2d[m/s]',parameter2);
femt.const{4} = s;
FFParaLog(FF,2) = parameter2;
FFCond(FF,2) = {'velocity2'};

s = sprintf('%10.2d[K]',parameter3);
femt.const{10} = s;
FFParaLog(FF,3) = parameter3;
FFCond(FF,3) = {'system temperature'};

s = sprintf('%10.2d[mol/kg]',parameter4);
femt.const{14} = s;
FFParaLog(FF,4) = parameter4;
FFCond(FF,4) = {'Concentration1'};

s = sprintf('%10.2d[mol/kg]',parameter5);
femt.const{16} = s;
FFParaLog(FF,5) = parameter5;
FFCond(FF,5) = {'Concentration2'};

%Compile Model
femt = multiphysics(femt);

%=========================================================================
% Time-Dependant Solver - (mmglf2 & chdh)
%=========================================================================
% Extend mesh
femt.xmesh=meshextend(femt, ...
    'geoms',[1], ...
    'eqvars','on', ...
    'cplbndeq','on', ...
    'cplbndsh','off', ...
    'linshape',[1], ...
    'linshapetol',0.1);

% Solve problem
femt.sol=femtime(femt, ...
    'u',0, ...
    'method','eliminate', ...
    'nullfun','auto', ...
    'complexfun','off', ...
    'matherr','on', ...
    'solfile','off', ...
    'conjugate','off', ...
    'symmetric','auto', ...
    'solcomp',{'u2','p2','c','v2','w2'}, ...
    'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
    'rowscale','on', ...
    'blocksize','auto', ...
    'reacf','on', ...
    'odesolver','bdf_ida', ...
    'tlist',[colon(0,1,50)], ...
    'rtol',0.01, ...
    'masssingular','maybe', ...
    'consistent','bweuler', ...
    'estrat',1, ...
    'tout','tlist', ...
    'tsteps','free', ...
    'complex','on', ...
    'atol',{'0.0010'}, ...
    'maxorder',5, ...
```

```
            'minorder',1, ...
            'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:                Concentration,                c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.3237763717777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739199,'camprojection'
,'orthographic','transparency',1.0}, ...
            'linsolver','spooles', ...
            'thresh',0.1, ...
            'preorder','nd', ...
            'uscale','none', ...
            'mcase',0);

        femt0=femt;
        %=====================================================================
        %Parametric Solver - (emqvw)
        %Note: Default when model loaded is Left
        %=====================================================================
        % Multiphysics
        femt=multiphysics(femt);

        % Extend mesh
        femt.xmesh=meshextend(femt, ...
                              'geoms',[1], ...
                              'eqvars','on', ...
                              'cplbndeq','on', ...
                              'cplbndsh','off', ...
                              'linshape',[1], ...
                              'linshapetol',0.1);

        % Solve problem
        femt.sol=femstatic(femt, ...
            'init',femt0.sol, ...
            'u',femt0.sol, ...
            'method','eliminate', ...
            'nullfun','auto', ...
            'complexfun','off', ...
            'matherr','on', ...
            'solfile','off', ...
            'conjugate','off', ...
            'symmetric','auto', ...
            'solcomp',{'Vportconstr1_g1_emqvw','V'}, ...
            'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
            'rowscale','on', ...
            'blocksize','auto', ...
            'reacf','on', ...
            'pname','nu_emqvw', ...
            'plist',[colon(10000,50000,500000)], ...
            'porder',1, ...
            'oldcomp',{}, ...
            'ntol',1.0E-6, ...
            'maxiter',25, ...
            'nonlin','auto', ...
            'damping','on', ...
            'hnlin','on', ...
            'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:                Concentration,                c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.3237763717777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739201,'camprojection'
,'orthographic','transparency',1.0}, ...
            'linsolver','pardiso', ...
            'pardreorder','nd', ...
            'pardrreorder','on', ...
            'pivotstrategy','off', ...
            'pivotperturb','1.0E-8', ...
            'itol',1.0E-6, ...
            'rhob',20, ...
            'errorchk','on', ...
            'uscale','none', ...
            'mcase',0);
```

```
%All Impedance Data for the First Pass (Conductivity Left)
FFImpedanceMag1(FF,1) = postglobaleval(femt,{'abs(Z11_emqvw)'});
FFImpedancePhi1(FF,1) = postglobaleval(femt,{'180*arg(Z11_emqvw)/pi'});

%=========================================================================
% Switch IS to scan Right Channel
% Note: this occurs for both the fault-free and fault model
%       boundary 1 is loaded as ground, boundary 11 needs to be ground
%=========================================================================
%Only a single subdomain
femt.appl{1}.equ.sigma = {'conductivityR'};

%No rubber-band function required here as the geometry remains
%fixed from when it was designed in the COMSOL GUI
femt.appl{1}.bnd.ind = [2 2 2 2 2 2 3 2 2 2 1];

%Compile Model
femt = geomanalyze(femt);
%Compile Model
femt = multiphysics(femt);

%=========================================================================
% Time-Dependant Solver - (mmglf2 & chdh)
%=========================================================================
% Extend mesh
femt.xmesh=meshextend(femt, ...
    'geoms',[1], ...
    'eqvars','on', ...
    'cplbndeq','on', ...
    'cplbndsh','off', ...
    'linshape',[1], ...
    'linshapetol',0.1);

% Solve problem
femt.sol=femtime(femt, ...
    'u',0, ...
    'method','eliminate', ...
    'nullfun','auto', ...
    'complexfun','off', ...
    'matherr','on', ...
    'solfile','off', ...
    'conjugate','off', ...
    'symmetric','auto', ...
    'solcomp',{'u2','p2','c','v2','w2'}, ...
    'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
    'rowscale','on', ...
    'blocksize','auto', ...
    'reacf','on', ...
    'odesolver','bdf_ida', ...
    'tlist',[colon(0,1,50)], ...
    'rtol',0.01, ...
    'masssingular','maybe', ...
    'consistent','bweuler', ...
    'estrat',1, ...
    'tout','tlist', ...
    'tsteps','free', ...
    'complex','on', ...
    'atol',{'0.0010'}, ...
    'maxorder',5, ...
    'minorder',1, ...
    'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:               Concentration,               c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739199,'camprojection'
,'orthographic','transparency',1.0}, ...
    'linsolver','spooles', ...
    'thresh',0.1, ...
    'preorder','nd', ...
    'uscale','none', ...
    'mcase',0);

femt0=femt;

%=========================================================================
```

```
% Parametric Solver - (emqvw)
% Note: Second Parametric Solve with Right Channel
%=====================================================================
% Solve problem
femt.sol=femstatic(femt, ...
    'init',femt0.sol, ...
    'u',femt0.sol, ...
    'method','eliminate', ...
    'nullfun','auto', ...
    'complexfun','off', ...
    'matherr','on', ...
    'solfile','off', ...
    'conjugate','off', ...
    'symmetric','auto', ...
    'solcomp',{'Vportconstr1_g1_emqvw','V'}, ...
    'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
    'rowscale','on', ...
    'blocksize','auto', ...
    'reacf','on', ...
    'pname','nu_emqvw', ...
    'plist',[colon(10000,50000,500000)], ...
    'porder',1, ...
    'oldcomp',{}, ...
    'ntol',1.0E-6, ...
    'maxiter',25, ...
    'nonlin','auto', ...
    'damping','on', ...
    'hnlin','on', ...
    'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:              Concentration,              c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.3237763717770758E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739201,'camprojection'
,'orthographic','transparency',1.0}, ...
    'linsolver','pardiso', ...
    'pardreorder','nd', ...
    'pardrreorder','on', ...
    'pivotstrategy','off', ...
    'pivotperturb','1.0E-8', ...
    'itol',1.0E-6, ...
    'rhob',20, ...
    'errorchk','on', ...
    'uscale','none', ...
    'mcase',0);

%=====================================================================
%= Measuring the Velocity at "Strategic" Locations
% In a potential GUI version the user will be able to determine what
% parameter is measured and where the sensor is located.
% For example, u = velocity field, p = pressure ....
%              and then the co-ordinates for the point on the geometry
%=====================================================================
% Velocity Measurements (y - direction)
%=====================================================================
FFParaLog(FF,6) = postinterp(femt, 'v2', [5e-5; 4.95e-3; 20e-6]);         %Outlet velo (PT1)
FFParaLog(FF,7) = postinterp(femt, 'v2', [5e-5; 5e-4; 20e-6]);           %mid-channel velo (PT2)
FFParaLog(FF,8) = postinterp(femt, 'v2', [-3e-3; -3e-3; 20e-6]);         %velo1 inlet (PT3)
FFParaLog(FF,9) = postinterp(femt, 'v2', [3.11e-3; -3e-3; 20e-6]);       %velo2 inlet (PT4)

%=====================================================================
% Pressure Measurements
%=====================================================================
FFParaLog(FF,10) = postinterp(femt, 'p2', [5e-5; 4.95e-3; 20e-6]);        %Outlet velo (PT1)
FFParaLog(FF,11) = postinterp(femt, 'p2', [5e-5; 5e-4; 20e-6]);           %mid-channel velo (PT2)
FFParaLog(FF,12) = postinterp(femt, 'p2', [-3e-3; -3e-3; 20e-6]);         %velo1 inlet (PT3)
FFParaLog(FF,13) = postinterp(femt, 'p2', [3.11e-3; -3e-3; 20e-6]);       %velo2 inlet (PT4)

%=====================================================================
% Velocity Vector Measurements
%=====================================================================
FFParaLog(FF,14) = postinterp(femt, 'U_mmglf2', [5e-5; 4.95e-3; 20e-6]);  %Outlet velo (PT1)
FFParaLog(FF,15) = postinterp(femt, 'U_mmglf2', [5e-5; 5e-4; 20e-6]);     %mid-channel velo (PT2)
FFParaLog(FF,16) = postinterp(femt, 'U_mmglf2', [-3e-3; -3e-3; 20e-6]);   %velo1 inlet (PT3)
FFParaLog(FF,17) = postinterp(femt, 'U_mmglf2', [3.11e-3; -3e-3; 20e-6]); %velo2 inlet (PT4)

%=====================================================================
```

```
        % Current Sensor Measurements (Levich)
        %=========================================================================
        % Continuous Data Recorded for in-depth sensor analysis, this does
        % represent the test sensor, but can be used in the analysis of the
        % design.
        % n records data each simulation cycle.
        %*************************************************************
        %***** Could some data mining algorithm be let loose on these
        %continuous recording to find data for fault classification? ******
        FFiOut(FF,1) = postglobaleval(femt,{'iOut'});
        FFiLeft(FF,1) = postglobaleval(femt,{'iLeft'});
        FFiRight(FF,1) = postglobaleval(femt,{'iRight'});

        %All Impedance Data for the Second Pass (Conductivity Right)
        FFImpedanceMag2(FF,1) = postglobaleval(femt,{'abs(Z11_emqvw)'});
        FFImpedancePhi2(FF,1) = postglobaleval(femt,{'180*arg(Z11_emqvw)/pi'});

        %=======================================
        % Fixed-Time Recorded (90th second) for test generation, this would be
        % the sample time of the test sensor, for example.

         %Error Trap added 10/11/09
        if((length(FFiOut(FF,1).y))<10)
            disp('Error')
        end;

        %Added some error detection, if solver not solved for 10 simulation
        %cycles then this section falls over because there is no 90 sample
        FFParaLog(FF,18) = FFiOut(FF,1).y(length(FFiOut(FF,1).y));
        FFParaLog(FF,19) = FFiLeft(FF,1).y(length(FFiLeft(FF,1).y));
        FFParaLog(FF,20) = FFiRight(FF,1).y(length(FFiRight(FF,1).y));
        %=======================================

    end;

else
    %=========================================================================
    % Fault Simulations
    %=========================================================================
    for u = 1:FaultStates

    %=========================================================================
    %Inner Simulation Loop
    %NOTE:
    %       One loop of the inner loop per nominal parameter simulation
    %=========================================================================
        for i = 1:n

        disp('Iteration Number')
        disp(i)                 %Iteration loop number for sanity check
        disp('Fault Number')
        disp(u)                 %Iteration loop number for sanity check
        disp(clock)             %[year month day hour minute seconds]

        %Record Start of Cycle Time
        FaultCycleTime(u,i,:) = clock;

        %allows monitoring of simulation efficieny

        %=========================================================================
        %Save the FEM Structure
        %=========================================================================
        % Geometry
        clear draw
        g2=flbinary('g2','draw',flbinaryfile);
        g5=flbinary('g5','draw',flbinaryfile);
        g4=flbinary('g4','draw',flbinaryfile);
        g3=flbinary('g3','draw',flbinaryfile);
        draw.p.objs = {g2,g5,g4,g3};
        draw.p.name = {'PT1','PT4','PT3','PT2'};
        draw.p.tags = {'g2','g5','g4','g3'};
        g1=flbinary('g1','draw',flbinaryfile);
        draw.s.objs = {g1};
        draw.s.name = {'EXT1'};
        draw.s.tags = {'g1'};
        fem.draw = draw;
        fem.mesh = flbinary('m1','mesh',flbinaryfile);

        femt = fem; %work on a copy of the structure - re-load purposes

        %=========================================================================
```

```
% User Defined specification
%=========================================================================
%The 'normrnd' function is used to obtain a random value from within
%the normal distribution. This is created from the user defined
%sepcification, in the presence of no inductive fault statistics.
%=========================================================================

%Need to parse constants to find values, or variable names!
parameter1 = normrnd(300e-6, (300e-6*0.2)); %velocity1 - +/-20%
parameter2 = normrnd(300e-6, (300e-6*0.2)); %velocity2 - +/-20%
parameter3 = normrnd(297, (297*0.1));       %Operating Temp. +/-10%
parameter4 = normrnd(0.8, (0.8*0.01));      %Concentration 1 +/-0.01%
parameter5 = normrnd(0.2, (0.2*0.01));      %Concentration 2 +/-0.01%

%=========================================================================
%Determine whether fixed or statistical nominal values are used
%Note:
%=========================================================================
if(Nominal == 0)

    s = sprintf('%10.2d[m/s]',parameter1);
    femt.const{2} = s;
    FaultParaLog(u,i,1) = parameter1;
    FaultCond(u,i,1) = {'velocity1'};

    s = sprintf('%10.2d[m/s]',parameter2);
    femt.const{4} = s;
    FaultParaLog(u,i,2) = parameter2;
    FaultCond(u,i,2) = {'velocity2'};

    s = sprintf('%10.2d[K]',parameter3);
    femt.const{10} = s;
    FaultParaLog(u,i,3) = parameter3;
    FaultCond(u,i,3) = {'system temperature'};

    s = sprintf('%10.2d[mol/kg]',parameter4);
    femt.const{14} = s;
    FaultParaLog(u,i,4) = parameter4;
    FaultCond(u,i,4) = {'Concentration1'};

    s = sprintf('%10.2d[mol/kg]',parameter5);
    femt.const{16} = s;
    FaultParaLog(u,i,5) = parameter5;
    FaultCond(u,i,5) = {'Concentration2'};

else
    %fixed values
    disp('fixed values')
end;

%=========================================================================
%Determine Fault Simulation Type
%Note:
%=========================================================================
%This implements each possible fault (no probability of occurrance)
%and runs the nominal values along with the fault n number of times
%depending upon the simulation cycle number.
switch FaultNumber

    % Left Inlet Concentration Degradation
    case 8
        FaultCond(u,i,6)={'Left Conc. Degradation'};

        disp('Left Conc. Degradation')

        parameter4 = normrnd(0.2, (0.2*0.2));      %Concentration 2 +/-20%

        s = sprintf('%10.2d[mol/kg]',parameter4);
        femt.const{14} = s;
        FaultParaLog(u,i,4) = parameter4;
        FaultCond(u,i,4) = {'Concentration1'};

        TwoDomains = 0;

        % Right Inlet Concentration Degradation
    case 7
        FaultCond(u,i,6)={'Right Conc. Degradation'};

        disp('Right Conc. Degradation')

        parameter5 = normrnd(0.8, (0.8*0.2));      %Concentration 1 +/-20%
```

```
            s = sprintf('%10.2d[mol/kg]',parameter5);
            femt.const{16} = s;
            FaultParaLog(u,i,4) = parameter5;
            FaultCond(u,i,4) = {'Concentration2'};

            TwoDomains = 0;

            %Left Inlet Pump Fail
        case 6

            FaultCond(u,i,6)={'Left Pump Fail'};

            disp('Left Pump Fail')

            parameter1 = normrnd(300e-6, (300e-6*2)); %velocity1 - +/-200%

            s = sprintf('%10.2d[m/s]',parameter1);
            femt.const{2} = s;
            FaultParaLog(u,i,1) = parameter1;
            FaultCond(u,i,1) = {'velocity1'};

            TwoDomains = 0;

            %Right Inlet Pump Fail
        case 5

            FaultCond(u,i,6)={'Right Pump Fail'};

            disp('Right Pump Fail')

            parameter2 = normrnd(300e-6, (300e-6*2)); %velocity1 - +/-200%

            s = sprintf('%10.2d[m/s]',parameter2);
            femt.const{4} = s;
            FaultParaLog(u,i,1) = parameter2;
            FaultCond(u,i,1) = {'velocity2'};

            TwoDomains = 0;

            %Bloackage 1 (outlet)
        case 4

            FaultCond(u,i,6)={'Outlet Blockage'};

            disp('Outlet Blockage')

            % Geometry
            g6=block3('50e-6','0.25e-3','30e-6','base','corner','pos',{'25e-6','4.5e-
3','0'},'axis',{'0','0','1'},'rot','0');
            g7=geomcomp({g1,g6},'ns',{'EXT1','BLK1'},'sf','EXT1+BLK1','face','none','edge','all');

            % Analyzed geometry
            clear p s
            femt.draw.p.objs={g2,g5,g4,g3};
            femt.draw.p.name={'PT1','PT4','PT3','PT2'};
            femt.draw.p.tags={'g2','g5','g4','g3'};

            femt.draw.s.objs={g7};
            femt.draw.s.name={'CO1'};
            femt.draw.s.tags={'g7'};

            femt.draw=struct('p',femt.draw.p,'s',femt.draw.s);

            femt = geomanalyze(femt);

            % Initialize mesh
            femt.mesh=meshinit(femt, ...
                'hauto',6, ...
                'hpnt',20, ...
                'xscale',1.0, ...
                'yscale',1.0, ...
                'zscale',1.0, ...
                'jiggle','on', ...
                'methodfac','tri');

            %Compile Model
            femt = multiphysics(femt);

            %NOTE: boundary conditions for the inserted fault block
            %do not have to be determined as they are automatically
```

```
    %assigned to continuity (COMSOL associativity).

    % ================================================Physics of the blockage
    %Application Mode 1 (emqvw)
    femt.appl{1}.equ.ind = [1 2];
    femt.appl{1}.equ.sigma = {'conductivityL' '0'};
    femt.appl{1}.equ.epsilonr = {'80' '1'};

    femt.appl{2}.equ.ind = [1,2];
    femt.appl{2}.equ.D = {'D','0'};
    femt.appl{2}.equ.w = {'w2','0'};
    femt.appl{2}.equ.v = {'v2','0'};
    femt.appl{2}.equ.u = {'u2','0'};

    femt.appl{3}.equ.ind = [1,2];
    femt.appl{3}.equ.eta = {'mat1_eta(T[1/K])[Pa*s]' ,'1.98[Pa*s]'};
    femt.appl{3}.equ.rho = {'mat1_rho(T[1/K])[kg/m^3]' ,'1.3[kg/m^3]'};

    TwoDomains = 1;

    %Bloackage 2 (Inlet Right)
case 3

    FaultCond(u,i,6)={'Inlet Right Blockage'};

    disp('Inlet Right Blockage')

    % Geometry
    g6=block3('50e-6','0.25e-3','30e-6','base','corner','pos',{'3.45e-3','-3.4e-
3','0'},'axis',{'0','0','1'},'rot','45');
    g7=geomcomp({g1,g6},'ns',{'EXT1','BLK1'},'sf','EXT1+BLK1','face','none','edge','all');

    % Analyzed geometry
    clear p s
    femt.draw.p.objs={g2,g5,g4,g3};
    femt.draw.p.name={'PT1','PT4','PT3','PT2'};
    femt.draw.p.tags={'g2','g5','g4','g3'};

    femt.draw.s.objs={g7};
    femt.draw.s.name={'CO1'};
    femt.draw.s.tags={'g7'};

    femt.draw=struct('p',femt.draw.p,'s',femt.draw.s);

    femt = geomanalyze(femt);

    % Initialize mesh
    femt.mesh=meshinit(femt, ...
        'hauto',6, ...
        'hpnt',20, ...
        'xscale',1.0, ...
        'yscale',1.0, ...
        'zscale',1.0, ...
        'jiggle','on', ...
        'methodfac','tri');

    %Compile Model
    femt = multiphysics(femt);

    %NOTE: boundary conditions for the inserted fault block
    %do not have to be determined as they are automatically
    %assigned to continuity (COMSOL associativity).

    % ================================================Physics of the blockage
    %Application Mode 1 (emqvw)
    femt.appl{1}.equ.ind = [1 2];
    femt.appl{1}.equ.sigma = {'conductivityL' '0'};
    femt.appl{1}.equ.epsilonr = {'80' '1'};

    femt.appl{2}.equ.ind = [1,2];
    femt.appl{2}.equ.D = {'D','0'};
    femt.appl{2}.equ.w = {'w2','0'};
    femt.appl{2}.equ.v = {'v2','0'};
    femt.appl{2}.equ.u = {'u2','0'};

    femt.appl{3}.equ.ind = [1,2];
    femt.appl{3}.equ.eta = {'mat1_eta(T[1/K])[Pa*s]' ,'1.98[Pa*s]'};
    femt.appl{3}.equ.rho = {'mat1_rho(T[1/K])[kg/m^3]' ,'1.3[kg/m^3]'};

    TwoDomains = 1;
```

```
                %Disconnect (Inlet Right)
            case 2
                %This is is the exact same conditions as for the inlet
                %blockage, apart from the inlet velocity is set to 0
                %m/s

                s = sprintf('%10.2d[m/s]',0);
                femt.const{4} = s;
                FaultParaLog(u,i,1) = parameter2;
                FaultCond(u,i,1) = {'velocity2'};

                FaultCond(u,i,6)={'Inlet Right Disconnect'};

                disp('Inlet Right Disconnect')

                % Geometry
                %figure(1)
                g6=block3('40e-6','0.25e-3','25e-6','base','corner','pos',{'3.25e-3','-3.2e-
3','0'},'axis',{'0','0','1'},'rot','45')
                g7=geomcomp({g1,g6},'ns',{'EXT1','BLK1'},'sf','EXT1+BLK1','face','none','edge','all')
                %geomplot(g7)

                % Analyzed geometry
                clear p s
                femt.draw.p.objs={g2,g5,g4,g3};
                femt.draw.p.name={'PT1','PT4','PT3','PT2'};
                femt.draw.p.tags={'g2','g5','g4','g3'};

                femt.draw.s.objs={g7};
                femt.draw.s.name={'CO1'};
                femt.draw.s.tags={'g7'};

                femt.draw=struct('p',femt.draw.p,'s',femt.draw.s);

                femt = geomanalyze(femt);

                % Initialize mesh
                femt.mesh=meshinit(femt, ...
                    'hauto',6, ...
                    'hpnt',20, ...
                    'xscale',1.0, ...
                    'yscale',1.0, ...
                    'zscale',1.0, ...
                    'jiggle','on', ...
                    'methodfac','tri');

                %Compile Model
                femt = multiphysics(femt);

                %NOTE: boundary conditions for the inserted fault block
                %do not have to be determined as they are automatically
                %assigned to continuity (COMSOL associativity).

                % ================================================Physics of the blockage
                %Application Mode 1 (emqvw)
                femt.appl{1}.equ.ind = [1 2];
                femt.appl{1}.equ.sigma = {'conductivityL' '0'};
                femt.appl{1}.equ.epsilonr = {'80' '1'};

                femt.appl{2}.equ.ind = [1,2];
                femt.appl{2}.equ.D = {'D','0'};
                femt.appl{2}.equ.w = {'w2','0'};
                femt.appl{2}.equ.v = {'v2','0'};
                femt.appl{2}.equ.u = {'u2','0'};

                femt.appl{3}.equ.ind = [1,2];
                femt.appl{3}.equ.eta = {'mat1_eta(T[1/K])[Pa*s]' ,'1.98[Pa*s]'};
                femt.appl{3}.equ.rho = {'mat1_rho(T[1/K])[kg/m^3]' ,'1.3[kg/m^3]'};

                TwoDomains = 1;

                %Bloackage 2 (Inlet Left)
            case 1

                FaultCond(u,i,6)={'Inlet Left Blockage'};

                disp('Inlet Left Blockage')

                %Create Physical Blockage
                % Geometry
```

```
                g6=block3('50e-6','0.25e-3','30e-6','base','corner','pos',{'-3.45e-3','-3.4e-
3','0'},'axis',{'0','0','1'},'rot','-45');
                g7=geomcomp({g1,g6},'ns',{'EXT1','BLK1'},'sf','EXT1+BLK1','face','none','edge','all');

                % Analyzed geometry
                clear p s
                femt.draw.p.objs={g2,g5,g4,g3};
                femt.draw.p.name={'PT1','PT4','PT3','PT2'};
                femt.draw.p.tags={'g2','g5','g4','g3'};

                femt.draw.s.objs={g7};
                femt.draw.s.name={'CO1'};
                femt.draw.s.tags={'g7'};

                femt.draw=struct('p',femt.draw.p,'s',femt.draw.s);
                %femt.geom=geomcsg(femt);

                femt = geomanalyze(femt);

                % Initialize mesh
                femt.mesh=meshinit(femt, ...
                    'hauto',6, ...
                    'hpnt',20, ...
                    'xscale',1.0, ...
                    'yscale',1.0, ...
                    'zscale',1.0, ...
                    'jiggle','on', ...
                    'methodfac','tri');

                %Compile Model
                femt = multiphysics(femt);

                %NOTE: boundary conditions for the inserted fault block
                %do not have to be determined as they are automatically
                %assigned to continuity (COMSOL associativity).

                % ================================================Physics of the blockage
                %Application Mode 1 (emqvw)
                femt.appl{1}.equ.ind = [1,2];
                femt.appl{1}.equ.sigma = {'conductivityL','0'};
                femt.appl{1}.equ.epsilonr = {'80','1'};

                %Application Mode 2 (chdh)
                femt.appl{2}.equ.ind = [1,2];
                femt.appl{2}.equ.D = {'D','0'};
                femt.appl{2}.equ.w = {'w2','0'};
                femt.appl{2}.equ.v = {'v2','0'};
                femt.appl{2}.equ.u = {'u2','0'};

                %Application Mode 3 (mmglf2)
                femt.appl{3}.equ.ind = [1,2];
                femt.appl{3}.equ.eta = {'mat1_eta(T[1/K])[Pa*s]' ,'1.98[Pa*s]'};
                femt.appl{3}.equ.rho = {'mat1_rho(T[1/K])[kg/m^3]' ,'1.3[kg/m^3]'};

                TwoDomains = 1;

                %Disconnect (Inlet Left)
            case 0
                %This is is the exact same conditions as for the inlet
                %blockage, apart from the inlet velocity is set to 0
                %m/s

                s = sprintf('%10.2d[m/s]',0);
                femt.const{2} = s;
                FaultParaLog(u,i,2) = parameter1;
                FaultCond(u,i,2) = {'velocity1'};

                FaultCond(u,i,6)={'Inlet Left Disconnect'};

                disp('Inlet Left Disconnect')

                %Create Physical Blockage
                % Geometry
                g6=block3('50e-6','0.25e-3','30e-6','base','corner','pos',{'-3.45e-3','-3.4e-
3','0'},'axis',{'0','0','1'},'rot','-45');
                g7=geomcomp({g1,g6},'ns',{'EXT1','BLK1'},'sf','EXT1+BLK1','face','none','edge','all');

                % Analyzed geometry
                clear p s
                femt.draw.p.objs={g2,g5,g4,g3};
                femt.draw.p.name={'PT1','PT4','PT3','PT2'};
```

```
        femt.draw.p.tags={'g2','g5','g4','g3'};

        femt.draw.s.objs={g7};
        femt.draw.s.name={'CO1'};
        femt.draw.s.tags={'g7'};

        femt.draw=struct('p',femt.draw.p,'s',femt.draw.s);
        %femt.geom=geomcsg(femt);

        femt = geomanalyze(femt);

        % Initialize mesh
        femt.mesh=meshinit(femt, ...
            'hauto',6, ...
            'hpnt',20, ...
            'xscale',1.0, ...
            'yscale',1.0, ...
            'zscale',1.0, ...
            'jiggle','on', ...
            'methodfac','tri');

        %Compile Model
        femt = multiphysics(femt);

        %NOTE: boundary conditions for the inserted fault block
        %do not have to be determined as they are automatically
        %assigned to continuity (COMSOL associativity).

        % ================================================Physics of the blockage
        %Application Mode 1 (emqvw)
        femt.appl{1}.equ.ind = [1,2];
        femt.appl{1}.equ.sigma = {'conductivityL','0'};
        femt.appl{1}.equ.epsilonr = {'80','1'};

        %Application Mode 2 (chdh)
        femt.appl{2}.equ.ind = [1,2];
        femt.appl{2}.equ.D = {'D','0'};
        femt.appl{2}.equ.w = {'w2','0'};
        femt.appl{2}.equ.v = {'v2','0'};
        femt.appl{2}.equ.u = {'u2','0'};

        %Application Mode 3 (mmglf2)
        femt.appl{3}.equ.ind = [1,2];
        femt.appl{3}.equ.eta = {'mat1_eta(T[1/K])[Pa*s]' ,'1.98[Pa*s]'};
        femt.appl{3}.equ.rho = {'mat1_rho(T[1/K])[kg/m^3]' ,'1.3[kg/m^3]'};

        TwoDomains = 1;
end;

%Compile Model
femt = multiphysics(femt);

%=========================================================================
% Time-Dependant Solver - (mmglf2 & chdh)
%=========================================================================
% Extend mesh
femt.xmesh=meshextend(femt, ...
    'geoms',[1], ...
    'eqvars','on', ...
    'cplbndeq','on', ...
    'cplbndsh','off', ...
    'linshape',[1], ...
    'linshapetol',0.1);

% Solve problem
femt.sol=femtime(femt, ...
    'u',0, ...
    'method','eliminate', ...
    'nullfun','auto', ...
    'complexfun','off', ...
    'matherr','on', ...
    'solfile','off', ...
    'conjugate','off', ...
    'symmetric','auto', ...
    'solcomp',{'u2','p2','c','v2','w2'}, ...
    'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
    'rowscale','on', ...
    'blocksize','auto', ...
    'reacf','on', ...
    'odesolver','bdf_ida', ...
    'tlist',[colon(0,1,50)], ...
```

```
            'rtol',0.01, ...
            'masssingular','maybe', ...
            'consistent','bweuler', ...
            'estrat',1, ...
            'tout','tlist', ...
            'tsteps','free', ...
            'complex','on', ...
            'atol',{'0.0010'}, ...
            'maxorder',5, ...
            'minorder',1, ...
            'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:               Concentration,               c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739199,'camprojection'
,'orthographic','transparency',1.0}, ...
            'linsolver','spooles', ...
            'thresh',0.1, ...
            'preorder','nd', ...
            'uscale','none', ...
            'mcase',0);

        femt0=femt;

        %Data taken here because time dependant data is no longer available
        %after the parametric solver!

        %=========================================================================
        % Current Sensor Measurements (Levich)
        %=========================================================================
        % Continuous Data Recorded for in-depth sensor analysis, this does
        % represent the test sensor, but can be used in the analysis of the
        % design.
        % n records data each simulation cycle.
        %*************************************************************
        %***** Could some data mining algorithm be let loose on these
        %continuous recording to find data for fault classification? ******
        iOut(u,i) = postglobaleval(femt,{'iOut'});
        iLeft(u,i) = postglobaleval(femt,{'iLeft'});
        iRight(u,i) = postglobaleval(femt,{'iRight'});

        %=========================================================================
        %Parametric Solver - (emqvw)
        %Note: Default when model loaded is Left
        %=========================================================================
        % Multiphysics
        femt=multiphysics(femt);

        % Extend mesh
        femt.xmesh=meshextend(femt, ...
            'geoms',[1], ...
            'eqvars','on', ...
            'cplbndeq','on', ...
            'cplbndsh','off', ...
            'linshape',[1], ...
            'linshapetol',0.1);

        % Solve problem
        femt.sol=femstatic(femt, ...
            'init',femt0.sol, ...
            'u',femt0.sol, ...
            'method','eliminate', ...
            'nullfun','auto', ...
            'complexfun','off', ...
            'matherr','on', ...
            'solfile','off', ...
            'conjugate','off', ...
            'symmetric','auto', ...
            'solcomp',{'Vportconstr1_g1_emqvw','V'}, ...
            'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
            'rowscale','on', ...
            'blocksize','auto', ...
            'reacf','on', ...
            'pname','nu_emqvw', ...
            'plist',[colon(10000,50000,500000)], ...
            'porder',1, ...
```

```
            'oldcomp',{}, ...
            'ntol',1.0E-6, ...
            'maxiter',25, ...
            'nonlin','auto', ...
            'damping','on', ...
            'hnlin','on', ...
            'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:               Concentration,               c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739201,'camprojection'
,'orthographic','transparency',1.0}, ...
            'linsolver','pardiso', ...
            'pardreorder','nd', ...
            'pardrreorder','on', ...
            'pivotstrategy','off', ...
            'pivotperturb','1.0E-8', ...
            'itol',1.0E-6, ...
            'rhob',20, ...
            'errorchk','on', ...
            'uscale','none', ...
            'mcase',0);

        %All Impedance Data for the First Pass (Conductivity Left)
        ImpedanceMag1(u,i) = postglobaleval(femt,{'abs(Z11_emqvw)'});
        ImpedancePhi1(u,i) = postglobaleval(femt,{'180*arg(Z11_emqvw)/pi'});

        conductivityleft = postglobaleval(femt,{'conductivityRAWL'});
        conductivityright = postglobaleval(femt,{'conductivityRAWR'});

        %======================================================================
        % Switch IS to scan Right Channel
        % Note: this occurs for both the fault-free and fault model
        %       boundary 1 is loaded as ground, boundary 11 needs to be ground
        %======================================================================
        if(TwoDomains == 0)
            %Only a single subdomain
            femt.appl{1}.equ.sigma = {'conductivityR'};
            %No rubber-band function required here as the geometry remains
            %fixed from when it was designed in the COMSOL GUI
            femt.appl{1}.bnd.ind = [2 2 2 2 2 2 3 2 2 2 1];

        else

            %Two subdomain's the second faulty
            femt.appl{1}.equ.sigma = {'conductivityR', '0' };
            %femt.appl{1}.bnd.eltype = {'V0','nJ0','cont','port'};

            %Find the boundary number for the Left Boundary using this
            %rubber-band function
            result1 = face(femt.geom,[0.0036365 0.0035655],[-0.0035355 -0.0034655],[0 40e-6],80e-6,1);
            femt.appl{1}.bnd.ind(result1{1}) = 1;    %ground

            %Find the boundary number for the Left Boundary using this
            %rubber-band function
            result2 = face(femt.geom,[-0.003465 -0.003536],[-0.003465 -0.003535],[0 40e-6],100e-6,2);
            femt.appl{1}.bnd.ind(result2{1}) = 2;    %electric insulation

            %femt = geomanalyze(femt);
        end;

        %Compile Model
        femt = geomanalyze(femt);
        %Compile Model
        femt = multiphysics(femt);

         %======================================================================
        % Time-Dependant Solver - (mmglf2 & chdh)
        %======================================================================
        % Extend mesh
        femt.xmesh=meshextend(femt, ...
            'geoms',[1], ...
            'eqvars','on', ...
            'cplbndeq','on', ...
            'cplbndsh','off', ...
            'linshape',[1], ...
```

```
        'linshapetol',0.1);

    % Solve problem
    femt.sol=femtime(femt, ...
        'u',0, ...
        'method','eliminate', ...
        'nullfun','auto', ...
        'complexfun','off', ...
        'matherr','on', ...
        'solfile','off', ...
        'conjugate','off', ...
        'symmetric','auto', ...
        'solcomp',{'u2','p2','c','v2','w2'}, ...
        'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
        'rowscale','on', ...
        'blocksize','auto', ...
        'reacf','on', ...
        'odesolver','bdf_ida', ...
        'tlist',[colon(0,1,50)], ...
        'rtol',0.01, ...
        'masssingular','maybe', ...
        'consistent','bweuler', ...
        'estrat',1, ...
        'tout','tlist', ...
        'tsteps','free', ...
        'complex','on', ...
        'atol',{'0.0010'}, ...
        'maxorder',5, ...
        'minorder',1, ...
        'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:               Concentration,               c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739199,'camprojection'
,'orthographic','transparency',1.0}, ...
        'linsolver','spooles', ...
        'thresh',0.1, ...
        'preorder','nd', ...
        'uscale','none', ...
        'mcase',0);

    femt0=femt;

    %=====================================================================
    % Parametric Solver - (emqvw)
    % Note: Second Parametric Solve with Right Channel
    %=====================================================================
    % Solve problem
    femt.sol=femstatic(femt, ...
        'init',femt0.sol, ...
        'u',femt0.sol, ...
        'method','eliminate', ...
        'nullfun','auto', ...
        'complexfun','off', ...
        'matherr','on', ...
        'solfile','off', ...
        'conjugate','off', ...
        'symmetric','auto', ...
        'solcomp',{'Vportconstr1_g1_emqvw','V'}, ...
        'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
        'rowscale','on', ...
        'blocksize','auto', ...
        'reacf','on', ...
        'pname','nu_emqvw', ...
        'plist',[colon(10000,50000,500000)], ...
        'porder',1, ...
        'oldcomp',{}, ...
        'ntol',1.0E-6, ...
        'maxiter',25, ...
        'nonlin','auto', ...
        'damping','on', ...
        'hnlin','on', ...
        'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
```

```
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:              Concentration,              c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739201,'camprojection'
,'orthographic','transparency',1.0}, ...
            'linsolver','pardiso', ...
            'pardreorder','nd', ...
            'pardrreorder','on', ...
            'pivotstrategy','off', ...
            'pivotperturb','1.0E-8', ...
            'itol',1.0E-6, ...
            'rhob',20, ...
            'errorchk','on', ...
            'uscale','none', ...
            'mcase',0);

        %=====================================================================
        %= Measuring the Velocity at "Strategic" Locations
        % In a potential GUI version the user will be able to determine what
        % parameter is measured and where the sensor is located.
        % For example, u = velocity field, p = pressure ....
        %              and then the co-ordinates for the point on the geometry
        %=====================================================================
        % Velocity Measurements (y - direction)
        %=====================================================================
        FaultParaLog(u,i,6) = postinterp(femt, 'v2', [5e-5; 4.95e-3; 20e-6]);         %Outlet velo (PT1)
        FaultParaLog(u,i,7) = postinterp(femt, 'v2', [5e-5; 5e-4; 20e-6]);            %mid-channel velo (PT2)
        FaultParaLog(u,i,8) = postinterp(femt, 'v2', [-3e-3; -3e-3; 20e-6]);          %velo1 inlet (PT3)
        FaultParaLog(u,i,9) = postinterp(femt, 'v2', [3.11e-3; -3e-3; 20e-6]);        %velo2 inlet (PT4)

        %=====================================================================
        % Pressure Measurements
        %=====================================================================
        FaultParaLog(u,i,10) = postinterp(femt, 'p2', [5e-5; 4.95e-3; 20e-6]);        %Outlet velo (PT1)
        FaultParaLog(u,i,11) = postinterp(femt, 'p2', [5e-5; 5e-4; 20e-6]);           %mid-channel velo (PT2)
        FaultParaLog(u,i,12) = postinterp(femt, 'p2', [-3e-3; -3e-3; 20e-6]);         %velo1 inlet (PT3)
        FaultParaLog(u,i,13) = postinterp(femt, 'p2', [3.11e-3; -3e-3; 20e-6]);       %velo2 inlet (PT4)

        %=====================================================================
        % Velocity Vector Measurements
        %=====================================================================
        FaultParaLog(u,i,14) = postinterp(femt, 'U_mmglf2', [5e-5; 4.95e-3; 20e-6]);  %Outlet velo (PT1)
        FaultParaLog(u,i,15) = postinterp(femt, 'U_mmglf2', [5e-5; 5e-4; 20e-6]);     %mid-channel velo (PT2)
        FaultParaLog(u,i,16) = postinterp(femt, 'U_mmglf2', [-3e-3; -3e-3; 20e-6]);   %velo1 inlet (PT3)
        FaultParaLog(u,i,17) = postinterp(femt, 'U_mmglf2', [3.11e-3; -3e-3; 20e-6]); %velo2 inlet (PT4)

        %All Impedance Data for the Second Pass (Conductivity Right)
        ImpedanceMag2(u,i) = postglobaleval(femt,{'abs(Z11_emqvw)'});
        ImpedancePhi2(u,i) = postglobaleval(femt,{'180*arg(Z11_emqvw)/pi'});

        %=======================================
        % Fixed-Time Recorded (9th scan cycle) for test generation, this would be
        % the sample time of the test sensor, for example.

        %Error Trap added 10/11/09
        if((length(iOut(u,i).y))<10)
            disp('Error')
        end;

        FaultParaLog(u,i,18) = iOut(u,i).y(length(iOut(u,i).y));
        FaultParaLog(u,i,19) = iLeft(u,i).y(length(iOut(u,i).y));
        FaultParaLog(u,i,20) = iRight(u,i).y(length(iOut(u,i).y));
        %=======================================

        %Record End of Cycle Time
        InnerCycleTimeEnd(u,i,:) = clock;
        end;

        %Move to the Next Fault
        FaultNumber = FaultNumber+1;
        %Record End of Cycle Time
        OuterCycleTimeEnd(u,:) = clock;

    end;

end;

disp('Finished - Enjoy Analysing Faulty Microchemical Reactors!');
```

APPENDIX III – "Y" Channel Test Analysis

```
%=========================================================================
%============================ Test Analysis  =============================
%-------------------------------------------------------------------------
% Declare the Fault-Free Measurement & Impedance Group Structures
%-------------------------------------------------------------------------
FaultFree = struct('TestPoint',{},'Measurements',[]);
FFImpedanceMag1FreqGrp = struct('Magnitude', []);
FFImpedancePhi1FreqGrp = struct('Phase', []);
FFImpedanceMag2FreqGrp = struct('Magnitude', []);
FFImpedancePhi2FreqGrp = struct('Phase', []);

%-------------------------------------------------------------------------
% Declare the Faulty Measurement & Impedance Group Structures
%-------------------------------------------------------------------------
FaultData = struct('FaultCondition',{},'TestPoint',{},'Measurements',[]);
FImpedanceMag1FreqGrp  = struct('Magnitude', []);
FImpedancePhi1FreqGrp  = struct('Phase', []);
FImpedanceMag2FreqGrp  = struct('Magnitude', []);
FImpedancePhi2FreqGrp  = struct('Phase', []);

FiLeftStruc =  struct('Value', []);
FiRightStruc = struct('Value', []);
FiOutStruc =   struct('Value', []);

pointer = 0;

%=========================================================================
%====================== PROCESSING STRUCTURES ============================

% FAULT-FREE
%-------------------------------------------------------------------------
% Processing Fault-Free Measurements
%-------------------------------------------------------------------------
for i = 1:15          %Number of Test Points
    for j =1:10       %Number of Nomminal Parameters
            %the 5 offset is used to step over the nominal parameters
            FaultFree(i,1).TestPoint = FaultParaLogDesc1(5+i);
            FaultFree(i,1).Measurements = FFParaLog(:,5+i);
    end;
end;

%-------------------------------------------------------------------------
% Seperate out the Fault-Free Impedance into groups
%-------------------------------------------------------------------------
for x = 1:10      %frequency grp
    for j = 1:10  %nominal pointer
        FFImpedanceMag1FreqGrp(x,1).Magnitude(j)  = FFImpedanceMag1(j,1).y(x);
        FFImpedancePhi1FreqGrp(x,1).Phase(j) = FFImpedancePhi1(j,1).y(x);
        FFImpedanceMag2FreqGrp(x,1).Magnitude(j)  = FFImpedanceMag2(j,1).y(x);
        FFImpedancePhi2FreqGrp(x,1).Phase(j) = FFImpedancePhi2(j,1).y(x);
    end;
end;

pointer=0;

% FAULTY
%-------------------------------------------------------------------------
% Seperate out the Faulty Sensor Measurements
%-------------------------------------------------------------------------
for i = 1:9             %- Number of Faults
    for j =1:15         %- Number of Fault Test Parameters
        FaultData(i,j).FaultCondition = FaultCond(i,1,6);
        FaultData(i,j).TestPoint = FaultParaLogDesc1(5+j);
        FaultData (i,j).Measurements = FaultParaLog(i,:,5+j);
    end;
end;

pointer=0;
%-------------------------------------------------------------------------
% Seperate out the Faulty Levich Sensor Data
```

```
for i = 1: 9 %number of fault conditions
    %for z = 1:length(iLeft(1,1).x) %number of simulation time steps
        for j = 1:10 %number of nominal runs
            %(i,:) used only to form a vertical list of results
            FiLeftStruc(i,:).Value(j,1)   = iLeft(i,j).y(50);
            FiRightStruc(i,:).Value(j,1)  = iRight(i,j).y(50);
            FiOutStruc(i,:).Value(j,1)    = iOut(i,j).y(50);
        end;
    %end;
end;
%-------------------------------------------------------------------------
% Seperate out the Faulty Impedance into groups
%-------------------------------------------------------------------------
for fault = 1:9
    for x = 1:10      %frequency
        for j = 1:10  %nominal
            if x > length(ImpedanceMag1(fault,j).y)
                disp('Nan')
            else
                FImpedanceMag1FreqGrp(fault,x,1).Magnitude(j) = ImpedanceMag1(fault,j).y(x);
                FImpedancePhi1FreqGrp(fault,x,1).Phase(j) = ImpedancePhi1(fault,j).y(x);
                FImpedanceMag2FreqGrp(fault,x,1).Magnitude(j) = ImpedanceMag2(fault,j).y(x);
                FImpedancePhi2FreqGrp(fault,x,1).Phase(j) = ImpedancePhi2(fault,j).y(x);
            end;
        end;
    end;
end;
%%
fault = 1;
frequency = 1;

%=========================================================================
%========================== CURVE FITTING ================================
FFData = struct('FFGaussX', [], 'FFGaussY', [], 'FFCoeff', []);
FFImpMag1Data = struct('FFGaussX', [], 'FFGaussY', [], 'FFCoeff', []);
FFImpMag2Data = struct('FFGaussX', [], 'FFGaussY', [], 'FFCoeff', []);
FFImpPhi1Data = struct('FFGaussX', [], 'FFGaussY', [], 'FFCoeff', []);
FFImpPhi2Data = struct('FFGaussX', [], 'FFGaussY', [], 'FFCoeff', []);

FData = struct('FGaussX', [], 'FGaussY', [], 'FCoeff', []);
FImpMag1Data = struct('FGaussX', [], 'FGaussY', [], 'FCoeff', []);
FImpMag2Data = struct('FGaussX', [], 'FGaussY', [], 'FCoeff', []);
FImpPhi1Data = struct('FGaussX', [], 'FGaussY', [], 'FCoeff', []);
FImpPhi2Data = struct('FGaussX', [], 'FGaussY', [], 'FCoeff', []);

% FiRightData = struct('FFGaussX', [], 'FFGaussY', [], 'FFCoeff', []);
% FiLeftData = struct('FFGaussX', [], 'FFGaussY', [], 'FFCoeff', []);
% FiOutData = struct('FFGaussX', [], 'FFGaussY', [], 'FFCoeff', []);

f = fittype('gauss1');
% These might need to be tweaked for fitting purposes if the fit function
% fails, or the resulting fit is poor
% Err: Try using or tightening upper and lower bounds on coefficients.
% options = fitoptions('gauss1');
% options.Lower = [5.9e-8 1.8e8 2.3e7];
% options.Upper = [6.2e-8 1.9e8 2.5e7];
coeffnames(f);
formula(f);

sigma = 5;

%-------------------------------------------------------------------------
% Fault Free Curves
%-------------------------------------------------------------------------
for x = 1:length(PDFPostFaultFreeData)

    %Added 01/07/10
    prob = (0.0002:0.0004:0.9998)';
    xFF = norminv(prob,mean(FaultFree(x).Measurements),std(FaultFree(x).Measurements));
    yFF = normpdf(xFF,mean(FaultFree(x).Measurements),std(FaultFree(x).Measurements));

    FFDatacFF = fit(xFF, yFF,f);
    FFDatacoeff = coeffvalues(FFDatacFF);

    FFData(x).FFGaussX = xFF;
    %FFData(x).FFGaussY = FFDatacoeff(1)*exp(-((FFData(x).FFGaussX-FFDatacoeff(2)).^2)/(2*FFDatacoeff(3).^2));
    FFData(x).FFGaussY = FFDatacoeff(1)*exp(-((FFData(x).FFGaussX-FFDatacoeff(2))/FFDatacoeff(3)).^2);
    FFData(x).FFCoeff = FFDatacoeff;
    end;
```

```
%tighter fit required for impedance curves
f = fittype('gauss1');

for frequency = 1:10

    %Added 01/07/10
    prob = (0.0002:0.0004:0.9998)';
    xFFmag1                                                                                    =
norminv(prob,mean(FFImpedanceMag1FreqGrp(frequency,1).Magnitude),(std(FFImpedanceMag1FreqGrp(frequency,1).Magni
tude)*1.5));
    yFFmag1                                                                                    =
normpdf(xFFmag1,mean(FFImpedanceMag1FreqGrp(frequency,1).Magnitude),(std(FFImpedanceMag1FreqGrp(frequency,1).Ma
gnitude)*1.5));

    FFImpedanceMag1cFF = fit(xFFmag1, yFFmag1,f);
    FFImp1Magcoeff = coeffvalues(FFImpedanceMag1cFF);

    FFImpMag1Data(frequency).FFGaussX = xFFmag1;
    FFImpMag1Data(frequency).FFGaussY       =       FFImp1Magcoeff(1)*exp(-((FFImpMag1Data(frequency).FFGaussX-
FFImp1Magcoeff(2))/FFImp1Magcoeff(3)).^2);
    FFImpMag1Data(frequency).FFCoeff = FFImp1Magcoeff;

end;


for frequency = 1:10

    %Added 01/07/10
    prob = (0.0002:0.0004:0.9998)';
    xFFmag2                                                                                    =
norminv(prob,mean(FFImpedanceMag2FreqGrp(frequency,1).Magnitude),std(FFImpedanceMag2FreqGrp(frequency,1).Magnit
ude));
    yFFmag2                                                                                    =
normpdf(xFFmag2,mean(FFImpedanceMag2FreqGrp(frequency,1).Magnitude),std(FFImpedanceMag2FreqGrp(frequency,1).Mag
nitude));

    FFImpedanceMag2cFF = fit(xFFmag2, yFFmag2,f);
    FFImp2Magcoeff = coeffvalues(FFImpedanceMag2cFF);

    FFImpMag2Data(frequency).FFGaussX = xFFmag2;
    FFImpMag2Data(frequency).FFGaussY       =       FFImp2Magcoeff(1)*exp(-((FFImpMag2Data(frequency).FFGaussX-
FFImp2Magcoeff(2))/FFImp2Magcoeff(3)).^2);
    FFImpMag2Data(frequency).FFCoeff = FFImp2Magcoeff;

end;

for frequency = 1:10

    %Added 01/07/10
    prob = (0.0002:0.0004:0.9998)';
    xFFphi1                                                                                    =
norminv(prob,mean(FFImpedancePhi1FreqGrp(frequency,1).Phase),std(FFImpedancePhi1FreqGrp(frequency,1).Phase));
    yFFphi1                                                                                    =
normpdf(xFFphi1,mean(FFImpedancePhi1FreqGrp(frequency,1).Phase),std(FFImpedancePhi1FreqGrp(frequency,1).Phase))
;

    FFImpedancePhi1cFF = fit(xFFphi1, yFFphi1,f);
    FFImp1Phicoeff = coeffvalues(FFImpedancePhi1cFF);

    FFImpPhi1Data(frequency).FFGaussX = xFFphi1;
    FFImpPhi1Data(frequency).FFGaussY       =       FFImp1Phicoeff(1)*exp(-((FFImpPhi1Data(frequency).FFGaussX-
FFImp1Phicoeff(2))/FFImp1Phicoeff(3)).^2);
    FFImpPhi1Data(frequency).FFCoeff = FFImp1Phicoeff;

end;

for frequency = 1:10

    %Added 01/07/10
    prob = (0.0002:0.0004:0.9998)';
    xFFphi2                                                                                    =
norminv(prob,mean(FFImpedancePhi2FreqGrp(frequency,1).Phase),std(FFImpedancePhi2FreqGrp(frequency,1).Phase));
    yFFphi2                                                                                    =
normpdf(xFFphi2,mean(FFImpedancePhi2FreqGrp(frequency,1).Phase),std(FFImpedancePhi2FreqGrp(frequency,1).Phase))
;
```

```
    FFImpedancePhi2cFF = fit(xFFphi2, yFFphi2,f);
    FFImp2Phicoeff = coeffvalues(FFImpedancePhi2cFF);

    FFImpPhi2Data(frequency).FFGaussX = xFFphi2;
    FFImpPhi2Data(frequency).FFGaussY       =       FFImp2Phicoeff(1)*exp(-((FFImpPhi2Data(frequency).FFGaussX-
FFImp2Phicoeff(2))/FFImp2Phicoeff(3)).^2);
    FFImpPhi2Data(frequency).FFCoeff = FFImp2Phicoeff;
end;

f = fittype('gauss1');

%-------------------------------------------------------------------------
% Faulty Curves
%-------------------------------------------------------------------------
for fault = 1:9
    for x = 1:length(PDFPostFaultData)

        %Added 01/07/10
        prob = (0.0002:0.0004:0.9998)';
        xF = norminv(prob,mean(FaultData(fault,x).Measurements),std(FaultData(fault,x).Measurements));
        yF = normpdf(xF,mean(FaultData(fault,x).Measurements),std(FaultData(fault,x).Measurements));

        if(isnan(yF))
            disp('NaN')
        else
            FDatacF = fit(xF, yF,f);
            FDatacoeff = coeffvalues(FDatacF);
        end;

        FData(fault,x).FGaussX = xF;
         FData(fault,x).FGaussY           =           FDatacoeff(1)*exp(-((FData(fault,x).FGaussX           -
         FDatacoeff(2))/FDatacoeff(3)).^2);
        FData(fault,x).FCoeff = FDatacoeff;
    end;
end;

f = fittype('gauss1');

for fault = 1:9
    for frequency = 1:10

        %Added 01/07/10
        prob = (0.0002:0.0004:0.9998)';
        xFmag1                                                                                                  =
norminv(prob,mean(FImpedanceMag1FreqGrp(fault,frequency,1).Magnitude),std(FImpedanceMag1FreqGrp(fault,frequency
,1).Magnitude));
        yFmag1                                                                                                  =
normpdf(xFmag1,mean(FImpedanceMag1FreqGrp(fault,frequency,1).Magnitude),std(FImpedanceMag1FreqGrp(fault,frequen
cy,1).Magnitude));

        FImpedanceMag1cF = fit(xFmag1, yFmag1,f);
        FDatacoeff = coeffvalues(FImpedanceMag1cF);

        FImpMag1Data(fault,frequency,1).FGaussX = xFmag1;
        FImpMag1Data(fault,frequency,1).FGaussY = FDatacoeff(1)*exp(-((FImpMag1Data(fault,frequency,1).FGaussX-
FDatacoeff(2))/FDatacoeff(3)).^2);
        FImpMag1Data(fault,frequency,1).FCoeff = FDatacoeff;


        %Added 01/07/10
        prob = (0.0002:0.0004:0.9998)';
        xFmag2                                                                                                  =
norminv(prob,mean(FImpedanceMag2FreqGrp(fault,frequency,1).Magnitude),std(FImpedanceMag2FreqGrp(fault,frequency
,1).Magnitude));
        yFmag2                                                                                                  =
normpdf(xFmag2,mean(FImpedanceMag2FreqGrp(fault,frequency,1).Magnitude),std(FImpedanceMag2FreqGrp(fault,frequen
cy,1).Magnitude));

        FImpedanceMag2cF = fit(xFmag2, yFmag2,f);
        FDatacoeff = coeffvalues(FImpedanceMag2cF);

        FImpMag2Data(fault,frequency,1).FGaussX = xFmag2;
        FImpMag2Data(fault,frequency,1).FGaussY = FDatacoeff(1)*exp(-((FImpMag2Data(fault,frequency,1).FGaussX-
FDatacoeff(2))/FDatacoeff(3)).^2);
        FImpMag2Data(fault,frequency,1).FCoeff = FDatacoeff;

        %Added 01/07/10
```

```
        prob = (0.0002:0.0004:0.9998)';
        xFphi1                                                                                                                                  =
norminv(prob,mean(FImpedancePhi1FreqGrp(fault,frequency,1).Phase),std(FImpedancePhi1FreqGrp(fault,frequency,1).
Phase));
        yFphi1                                                                                                                                  =
normpdf(xFphi1,mean(FImpedancePhi1FreqGrp(fault,frequency,1).Phase),std(FImpedancePhi1FreqGrp(fault,frequency,1
).Phase));

        FImpedancePhi1cF = fit(xFphi1, yFphi1,f);
        FDatacoeff = coeffvalues(FImpedancePhi1cF);

        FImpPhi1Data(fault,frequency,1).FGaussX = xFphi1;
        FImpPhi1Data(fault,frequency,1).FGaussY = FDatacoeff(1)*exp(-((FImpPhi1Data(fault,frequency,1).FGaussX-
FDatacoeff(2))/FDatacoeff(3)).^2);
        FImpPhi1Data(fault,frequency,1).FCoeff = FDatacoeff;

        %Added 01/07/10
        prob = (0.0002:0.0004:0.9998)';
        xFphi2                                                                                                                                  =
norminv(prob,mean(FImpedancePhi2FreqGrp(fault,frequency,1).Phase),std(FImpedancePhi2FreqGrp(fault,frequency,1).
Phase));
        yFphi2                                                                                                                                  =
normpdf(xFphi2,mean(FImpedancePhi2FreqGrp(fault,frequency,1).Phase),std(FImpedancePhi2FreqGrp(fault,frequency,1
).Phase));

        FImpedancePhi2cF = fit(xFphi2, yFphi2,f);
        FDatacoeff = coeffvalues(FImpedancePhi2cF);

        FImpPhi2Data(fault,frequency,1).FGaussX = xFphi2;
        FImpPhi2Data(fault,frequency,1).FGaussY = FDatacoeff(1)*exp(-((FImpPhi2Data(fault,frequency,1).FGaussX-
FDatacoeff(2))/FDatacoeff(3)).^2);
        FImpPhi2Data(fault,frequency,1).FCoeff = FDatacoeff;

    end;
end;


%=========================================================================
% Probability Structures
%=========================================================================
FF_FFData = struct('FaultCondition', {}, 'TestPoint', {}, 'Probability', []);
FF_FData = struct('FaultCondition', {}, 'TestPoint', {}, 'Probability', []);
F_FFData = struct('FaultCondition', {}, 'TestPoint', {}, 'Probability', []);
F_FData = struct('FaultCondition', {}, 'TestPoint', {}, 'Probability', []);

FF_FFMag1 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);
FF_FMag1 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);
F_FFMag1 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);
F_FMag1 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);

FF_FFMag2 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);
FF_FMag2 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);
F_FFMag2 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);
F_FMag2 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);

FF_FFPhi1 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);
FF_FPhi1 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);
F_FFPhi1 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);
F_FPhi1 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);

FF_FFPhi2 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);
FF_FPhi2 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);
F_FFPhi2 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);
F_FPhi2 = struct('FaultCondition', {}, 'Frequency', {}, 'Probability', []);

%=========================================================================
% Functional Sensor Probability Calculations
%=========================================================================
for fault = 1:9         %- Number of Faults
    for x =1:15         %- Number of Fault Test Parameters

        %-------------------------------------------------------------------
        % Fault-Free as Fault-Free
        FF_FFData(fault).FaultCondition = FaultData(fault,x).FaultCondition;
        FF_FFData(fault).TestPoint(x) = FaultData(fault,x).TestPoint;

        ROAmin = (-3*std(FFData(x).FFGaussX)) + mean(FFData(x).FFGaussX);
```

```
        ROAmax = (3*std(FFData(x).FFGaussX)) + mean(FFData(x).FFGaussX);

        F = @(x1) (FFData(x).FFCoeff(1)*exp(-((x1-FFData(x).FFCoeff(2))/FFData(x).FFCoeff(3)).^2));

        Q = quad(F,ROAmin,ROAmax);
        FF_FFData(fault).Probability(x) = abs(Q);


        %-------------------------------------------------------------------
        % Fault-Free as Faulty
        FF_FData(fault).FaultCondition = FaultData(fault,x).FaultCondition;
        FF_FData(fault).TestPoint(x) = FaultData(fault,x).TestPoint;

        F = @(x1) (FFData(x).FFCoeff(1)*exp(-((x1-FFData(x).FFCoeff(2))/FFData(x).FFCoeff(3)).^2));
        Q1 = quad(F,min(FFData(x).FFGaussX),ROAmin);
        Q2 = quad(F,ROAmax,max(FFData(x).FFGaussX));
        FF_FData(fault).Probability(x) = abs(Q1 + Q2);

        %-------------------------------------------------------------------
        % Faulty as Faulty
        F_FData(fault).FaultCondition = FaultData(fault,x).FaultCondition;
        F_FData(fault).TestPoint(x) = FaultData(fault,x).TestPoint;

        F = @(x1) (FData(fault,x).FCoeff(1)*exp(-((x1-FData(fault,x).FCoeff(2))/FData(fault,x).FCoeff(3)).^2));
        Q1 = quad(F,ROAmax,max(FData(fault,x).FGaussX));

        F = @(x1) (FData(fault,x).FCoeff(1)*exp(-((x1-FData(fault,x).FCoeff(2))/FData(fault,x).FCoeff(3)).^2));
        Q2 = quad(F,min(FData(fault,x).FGaussX),ROAmin);
        F_FData(fault).Probability(x) = abs(Q1+Q2);

        %-------------------------------------------------------------------
        % Faulty as Fault-Free
        F_FFData(fault).FaultCondition = FaultData(fault,x).FaultCondition;
        F_FFData(fault).TestPoint(x) = FaultData(fault,x).TestPoint;

        F = @(x1) (FData(fault,x).FCoeff(1)*exp(-((x1-FData(fault,x).FCoeff(2))/FData(fault,x).FCoeff(3)).^2));
        Q2 = quad(F,ROAmin, ROAmax);
        F_FFData(fault).Probability(x) = abs(Q2);

    end;
end;

%=========================================================================
% Impedance Magnitude 1 – Probability Calculations
%=========================================================================
for fault = 1:9             %- Number of Faults
    for frequency =1:10     %- Number of test frequencies

        %-------------------------------------------------------------------
        % Fault-Free as Fault-Free
        FF_FFMag1(fault).FaultCondition = FaultData(fault,1).FaultCondition;
        FF_FFMag1(fault).Frequency(frequency) = frequency;

        ROAmin = (-3*std(FFImpMag1Data(frequency).FFGaussX)) + mean(FFImpMag1Data(frequency).FFGaussX);
        ROAmax = (3*std(FFImpMag1Data(frequency).FFGaussX)) + mean(FFImpMag1Data(frequency).FFGaussX);

        F                =               @(x1)                (FFImpMag1Data(frequency).FFCoeff(1)*exp(-((x1-
FFImpMag1Data(frequency).FFCoeff(2))/FFImpMag1Data(frequency).FFCoeff(3)).^2));

        Q = quad(F,ROAmin,ROAmax);

        Q = Q*1.6;
        FF_FFMag1(fault).Probability(frequency) = abs(Q);

        %-------------------------------------------------------------------
        % Fault-Free as Faulty
        FF_FMag1(fault).FaultCondition = FaultData(fault,1).FaultCondition;
        FF_FMag1(fault).Frequency(frequency) = frequency;

        F                =               @(x1)                (FFImpMag1Data(frequency).FFCoeff(1)*exp(-((x1-
FFImpMag1Data(frequency).FFCoeff(2))/FFImpMag1Data(frequency).FFCoeff(3)).^2));

        Q1 = quad(F,min(FFImpMag1Data(frequency).FFGaussX),ROAmin);
        Q2 = quad(F,ROAmax,max(FFImpMag1Data(frequency).FFGaussX));
        FF_FMag1(fault).Probability(frequency) = abs(Q1 + Q2)*1.6;
```

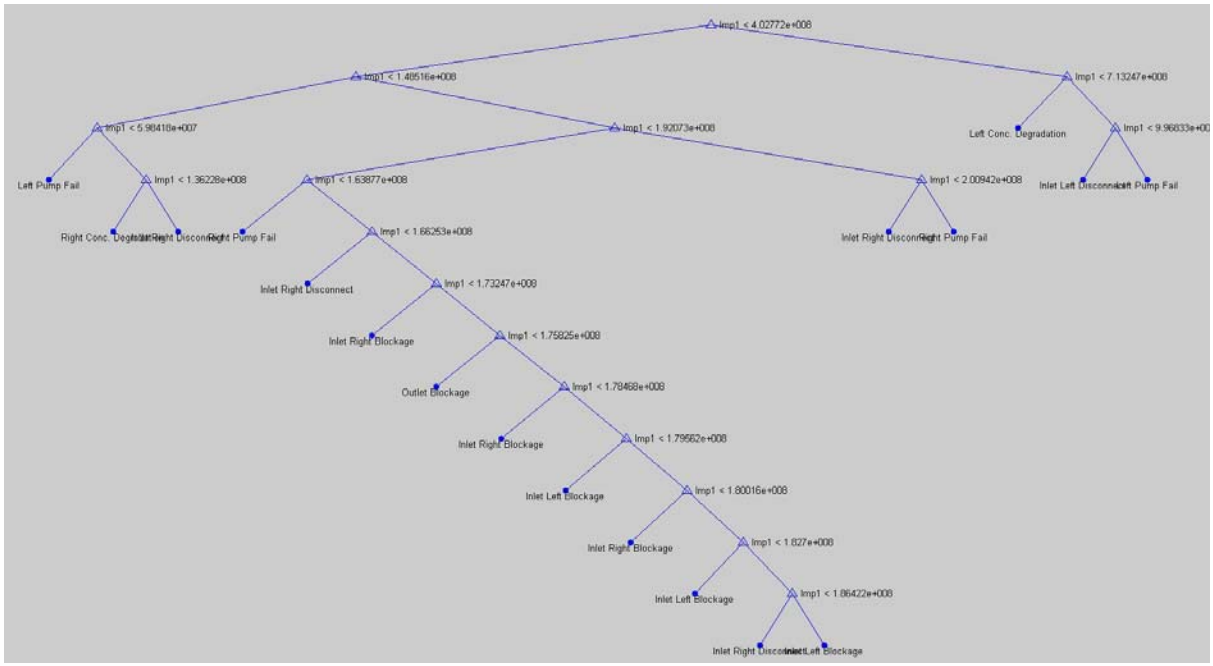
```
        %-------------------------------------------------------------------
        % Faulty as Faulty
        F_FMag1(fault).FaultCondition = FaultData(fault,1).FaultCondition;
        F_FMag1(fault).Frequency(frequency) = frequency;

        %Added the "other" magnitudes FF response "Good" boundaries
        ROAmin = (-3*std(FFImpMag2Data(frequency).FFGaussX)) + mean(FFImpMag2Data(frequency).FFGaussX);
        ROAmax = (3*std(FFImpMag2Data(frequency).FFGaussX)) + mean(FFImpMag2Data(frequency).FFGaussX);

        F               =               @(x1)               (FImpMag1Data(fault,frequency).FCoeff(1)*exp(-((x1-
FImpMag1Data(fault,frequency).FCoeff(2))/FImpMag1Data(fault,frequency).FCoeff(3)).^2));
        Q1 = quad(F,ROAmax,max(FImpMag1Data(fault,frequency).FGaussX));
        Q2 = quad(F,min(FImpMag1Data(fault,frequency).FGaussX),ROAmin);
        F_FMag1(fault).Probability(frequency) = abs(Q1+Q2)*1.6;

        %-------------------------------------------------------------------
        % Faulty as Fault-Free
        F_FFMag1(fault).FaultCondition = FaultData(fault,x).FaultCondition;
        F_FFMag1(fault).TestPoint(x) = FaultData(fault,x).TestPoint;

        F               =               @(x1)               (FImpMag1Data(fault,frequency).FCoeff(1)*exp(-((x1-
FImpMag1Data(fault,frequency).FCoeff(2))/FImpMag1Data(fault,frequency).FCoeff(3)).^2));
        Q2 = quad(F,ROAmin, ROAmax);
        F_FFMag1(fault).Probability(frequency) = abs(Q2)*1.6;

    end;
end;


%=========================================================================
% Impedance Magnitude 2 – Probability Calculations
%=========================================================================
for fault = 1:9             %- Number of Faults
    for frequency =1:10     %- Number of test frequencies

        %-------------------------------------------------------------------
        % Fault-Free as Fault-Free
        FF_FFMag2(fault).FaultCondition = FaultData(fault,1).FaultCondition;
        FF_FFMag2(fault).Frequency(frequency) = frequency;

        ROAmin = (-3*std(FFImpMag2Data(frequency).FFGaussX)) + mean(FFImpMag2Data(frequency).FFGaussX);
        ROAmax = (3*std(FFImpMag2Data(frequency).FFGaussX)) + mean(FFImpMag2Data(frequency).FFGaussX);

        F                =                @(x1)                (FFImpMag2Data(frequency).FFCoeff(1)*exp(-((x1-
FFImpMag2Data(frequency).FFCoeff(2))/FFImpMag2Data(frequency).FFCoeff(3)).^2));

        Q = quad(F,ROAmin,ROAmax);

        Q = Q*1.6;
        FF_FFMag2(fault).Probability(frequency) = abs(Q);

        %-------------------------------------------------------------------
        % Fault-Free as Faulty
        FF_FMag2(fault).FaultCondition = FaultData(fault,1).FaultCondition;
        FF_FMag2(fault).Frequency(frequency) = frequency;

        F                =                @(x1)                (FFImpMag2Data(frequency).FFCoeff(1)*exp(-((x1-
FFImpMag2Data(frequency).FFCoeff(2))/FFImpMag2Data(frequency).FFCoeff(3)).^2));

        Q1 = quad(F,min(FFImpMag2Data(frequency).FFGaussX),ROAmin);
        Q2 = quad(F,ROAmax,max(FFImpMag2Data(frequency).FFGaussX));
        FF_FMag2(fault).Probability(frequency) = abs(Q1 + Q2)*1.6;

        %-------------------------------------------------------------------
        % Faulty as Faulty
        F_FMag2(fault).FaultCondition = FaultData(fault,1).FaultCondition;
        F_FMag2(fault).Frequency(frequency) = frequency;

        %Added the "other" magnitudes FF response "Good" boundaries
        ROAmin = (-3*std(FFImpMag1Data(frequency).FFGaussX)) + mean(FFImpMag1Data(frequency).FFGaussX);
        ROAmax = (3*std(FFImpMag1Data(frequency).FFGaussX)) + mean(FFImpMag1Data(frequency).FFGaussX);

        F               =               @(x1)               (FImpMag2Data(fault,frequency).FCoeff(1)*exp(-((x1-
FImpMag2Data(fault,frequency).FCoeff(2))/FImpMag2Data(fault,frequency).FCoeff(3)).^2));
        Q1 = quad(F,ROAmax,max(FImpMag2Data(fault,frequency).FGaussX));
        Q2 = quad(F,min(FImpMag2Data(fault,frequency).FGaussX),ROAmin);
```

```
        F_FMag2(fault).Probability(frequency) = abs(Q1+Q2)*1.6;

        %-------------------------------------------------------------------
        % Faulty as Fault-Free
        F_FFMag2(fault).FaultCondition = FaultData(fault,x).FaultCondition;
        F_FFMag2(fault).TestPoint(x) = FaultData(fault,x).TestPoint;

        F               =               @(x1)               (FImpMag2Data(fault,frequency).FCoeff(1)*exp(-((x1-
FImpMag2Data(fault,frequency).FCoeff(2))/FImpMag2Data(fault,frequency).FCoeff(3)).^2));
        Q2 = quad(F,ROAmin, ROAmax);
        F_FFMag2(fault).Probability(frequency) = abs(Q2)*1.6;

    end;
end;
%=========================================================================
% Impedance Phase 1 – Probability Calculations
%=========================================================================
for fault = 1:9             %- Number of Faults
    for frequency =1:10     %- Number of test frequencies

        %-------------------------------------------------------------------
        % Fault-Free as Fault-Free
        FF_FFPhi1(fault).FaultCondition = FaultData(fault,1).FaultCondition;
        FF_FFPhi1(fault).Frequency(frequency) = frequency;

        ROAmin = (-3*std(FFImpPhi1Data(frequency).FFGaussX)) + mean(FFImpPhi1Data(frequency).FFGaussX);
        ROAmax = (3*std(FFImpPhi1Data(frequency).FFGaussX)) + mean(FFImpPhi1Data(frequency).FFGaussX);

        F                =                @(x1)                (FFImpPhi1Data(frequency).FFCoeff(1)*exp(-((x1-
FFImpPhi1Data(frequency).FFCoeff(2))/FFImpPhi1Data(frequency).FFCoeff(3)).^2));

        Q = quad(F,ROAmin,ROAmax);

        Q = Q;
        FF_FFPhi1(fault).Probability(frequency) = abs(Q);

        %-------------------------------------------------------------------
        % Fault-Free as Faulty
        FF_FPhi1(fault).FaultCondition = FaultData(fault,1).FaultCondition;
        FF_FPhi1(fault).Frequency(frequency) = frequency;

        F                =                @(x1)                (FFImpPhi1Data(frequency).FFCoeff(1)*exp(-((x1-
FFImpPhi1Data(frequency).FFCoeff(2))/FFImpPhi1Data(frequency).FFCoeff(3)).^2));

        Q1 = quad(F,min(FFImpPhi1Data(frequency).FFGaussX),ROAmin);
        Q2 = quad(F,ROAmax,max(FFImpPhi1Data(frequency).FFGaussX));
        FF_FPhi1(fault).Probability(frequency) = abs(Q1 + Q2);

        %-------------------------------------------------------------------
        % Faulty as Faulty
        F_FPhi1(fault).FaultCondition = FaultData(fault,1).FaultCondition;
        F_FPhi1(fault).Frequency(frequency) = frequency;

        %Added the "other" magnitudes FF response "Good" boundaries
        ROAmin = (-3*std(FFImpPhi2Data(frequency).FFGaussX)) + mean(FFImpPhi2Data(frequency).FFGaussX);
        ROAmax = (3*std(FFImpPhi2Data(frequency).FFGaussX)) + mean(FFImpPhi2Data(frequency).FFGaussX);

        F               =               @(x1)               (FImpPhi1Data(fault,frequency).FCoeff(1)*exp(-((x1-
FImpPhi1Data(fault,frequency).FCoeff(2))/FImpPhi1Data(fault,frequency).FCoeff(3)).^2));
        Q1 = quad(F,ROAmax,max(FImpPhi1Data(fault,frequency).FGaussX));
        Q2 = quad(F,min(FImpPhi1Data(fault,frequency).FGaussX),ROAmin);
        F_FPhi1(fault).Probability(frequency) = abs(Q1+Q2);

        %-------------------------------------------------------------------
        % Faulty as Fault-Free
        F_FFPhi1(fault).FaultCondition = FaultData(fault,x).FaultCondition;
        F_FFPhi1(fault).TestPoint(x) = FaultData(fault,x).TestPoint;

        F               =               @(x1)               (FImpPhi1Data(fault,frequency).FCoeff(1)*exp(-((x1-
FImpPhi1Data(fault,frequency).FCoeff(2))/FImpPhi1Data(fault,frequency).FCoeff(3)).^2));
        Q2 = quad(F,ROAmin, ROAmax);
        F_FFPhi1(fault).Probability(frequency) = abs(Q2);

    end;
end;
```

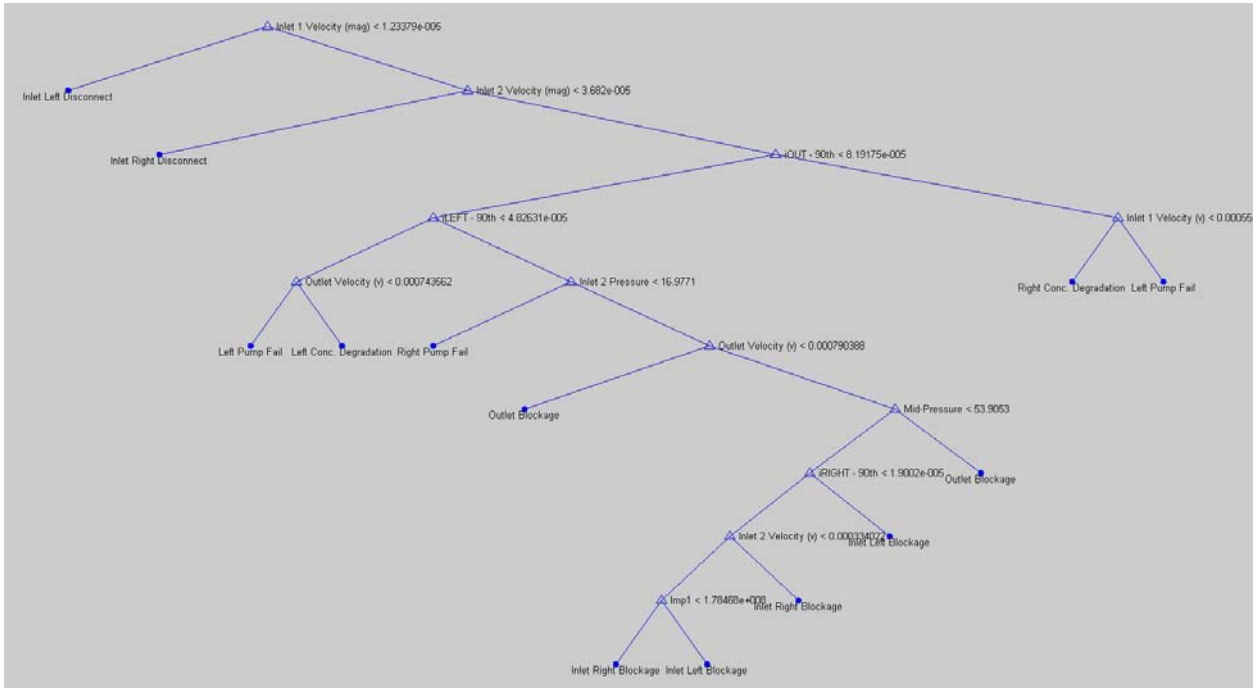
```
%=========================================================================
% Impedance Phase 2 – Probability Calculations
%=========================================================================
for fault = 1:9             %- Number of Faults
    for frequency =1:10     %- Number of test frequencies


        %-------------------------------------------------------------------
        % Fault-Free as Fault-Free
        FF_FFPhi2(fault).FaultCondition = FaultData(fault,1).FaultCondition;
        FF_FFPhi2(fault).Frequency(frequency) = frequency;

        ROAmin = (-3*std(FFImpPhi2Data(frequency).FFGaussX)) + mean(FFImpPhi2Data(frequency).FFGaussX);
        ROAmax = (3*std(FFImpPhi2Data(frequency).FFGaussX)) + mean(FFImpPhi2Data(frequency).FFGaussX);

        F                =                @(x1)                (FFImpPhi2Data(frequency).FFCoeff(1)*exp(-((x1-
FFImpPhi2Data(frequency).FFCoeff(2))/FFImpPhi2Data(frequency).FFCoeff(3)).^2));

        Q = quad(F,ROAmin,ROAmax);

        FF_FFPhi2(fault).Probability(frequency) = abs(Q);

        %-------------------------------------------------------------------
        % Fault-Free as Faulty
        FF_FPhi2(fault).FaultCondition = FaultData(fault,1).FaultCondition;
        FF_FPhi2(fault).Frequency(frequency) = frequency;

        F                =                @(x1)                (FFImpPhi2Data(frequency).FFCoeff(1)*exp(-((x1-
FFImpPhi2Data(frequency).FFCoeff(2))/FFImpPhi2Data(frequency).FFCoeff(3)).^2));

        Q1 = quad(F,min(FFImpPhi2Data(frequency).FFGaussX),ROAmin);
        Q2 = quad(F,ROAmax,max(FFImpPhi2Data(frequency).FFGaussX));
        FF_FPhi2(fault).Probability(frequency) = abs(Q1 + Q2);

        %-------------------------------------------------------------------
        % Faulty as Faulty
        F_FPhi2(fault).FaultCondition = FaultData(fault,1).FaultCondition;
        F_FPhi2(fault).Frequency(frequency) = frequency;

        %Added the "other" magnitudes FF response "Good" boundaries
        ROAmin = (-3*std(FFImpPhi1Data(frequency).FFGaussX)) + mean(FFImpPhi1Data(frequency).FFGaussX);
        ROAmax = (3*std(FFImpPhi1Data(frequency).FFGaussX)) + mean(FFImpPhi1Data(frequency).FFGaussX);

        F              =              @(x1)              (FImpPhi2Data(fault,frequency).FCoeff(1)*exp(-((x1-
FImpPhi2Data(fault,frequency).FCoeff(2))/FImpPhi2Data(fault,frequency).FCoeff(3)).^2));
        Q1 = quad(F,ROAmax,max(FImpPhi2Data(fault,frequency).FGaussX));
        Q2 = quad(F,min(FImpPhi2Data(fault,frequency).FGaussX),ROAmin);
        F_FPhi2(fault).Probability(frequency) = abs(Q1+Q2);

        %-------------------------------------------------------------------
        % Faulty as Fault-Free
        F_FFPhi2(fault).FaultCondition = FaultData(fault,x).FaultCondition;
        F_FFPhi2(fault).TestPoint(x) = FaultData(fault,x).TestPoint;

        F              =              @(x1)              (FImpPhi2Data(fault,frequency).FCoeff(1)*exp(-((x1-
FImpPhi2Data(fault,frequency).FCoeff(2))/FImpPhi2Data(fault,frequency).FCoeff(3)).^2));
        Q2 = quad(F,ROAmin, ROAmax);
        F_FFPhi2(fault).Probability(frequency) = abs(Q2);

    end;
end;

%=========================================================================
% Searching for the "Best" Sensor
%=========================================================================
for fault = 1:9
    [value, index] = max(F_FData(fault).Probability);
    BestTestPoints(fault,1) = FaultData(fault,index).TestPoint;
end;

disp('end of cycle')
```

APPENDIX IV - Decision Trees

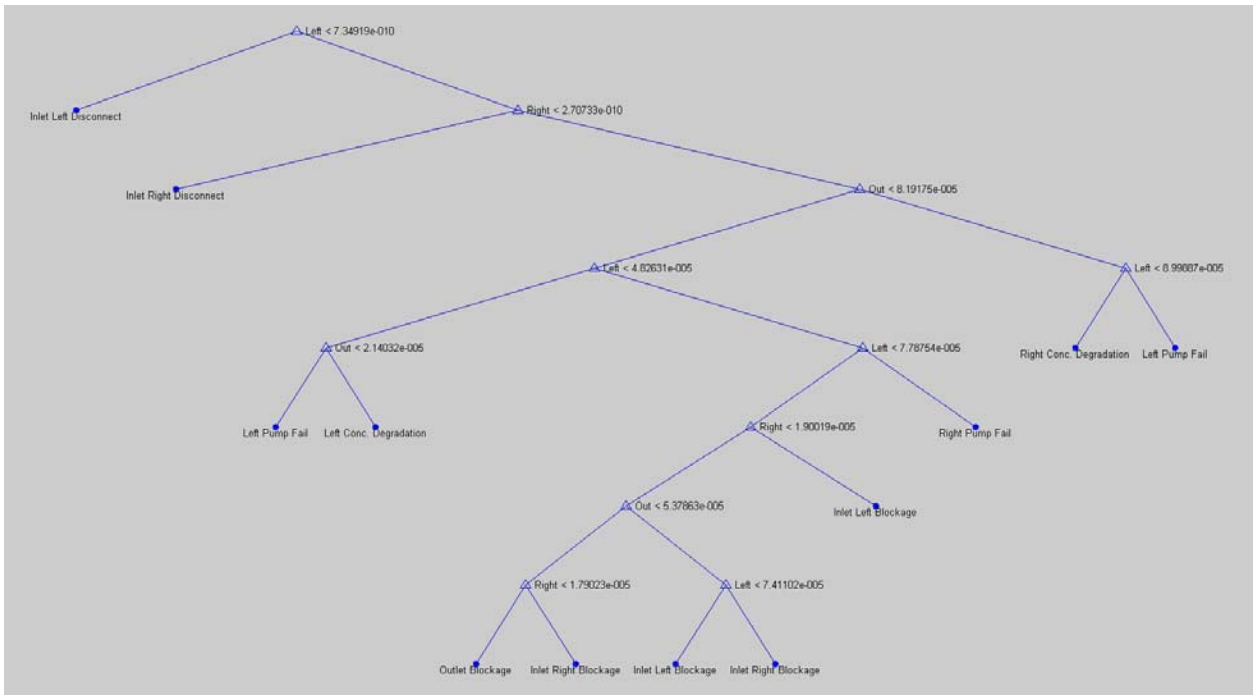
Gini Split Criterion



Twoing Split Criterion



Deviance Split Criterion



All Sensors Gini classification Rule

1 if x11<1.23379e-005 then node 2 else node 3

2 class = Inlet Left Disconnect

3 if x12<3.682e-005 then node 4 else node 5

4 class = Inlet Right Disconnect

5 if x13<8.19175e-005 then node 6 else node 7

6 if x14<4.82631e-005 then node 8 else node 9

7 if x3<0.000554279 then node 10 else node 11

8 if x1<0.000743562 then node 12 else node 13

9 if x8<16.9771 then node 14 else node 15

10 class = Right Conc. Degradation

11 class = Left Pump Fail

12 class = Left Pump Fail

13 class = Left Conc. Degradation

14 class = Right Pump Fail

15 if x1<0.000790388 then node 16 else node 17

16 class = Outlet Blockage

17 if x6<53.9053 then node 18 else node 19

18 if x15<1.9002e-005 then node 20 else node 21

19 class = Outlet Blockage

20 if x4<0.000334022 then node 22 else node 23

21 class = Inlet Left Blockage

22 if x14<7.41102e-005 then node 24 else node 25

23 class = Inlet Right Blockage

24 class = Inlet Left Blockage

25 class = Inlet Right Blockage

CODE FOR COST CALCULATION

```
[cLev,sLev,nLev,bestLev] = test(tLev,'cross',Levich,FaultConditions);

tminLev = prune(tLev,'level',bestLev)

[mincostLev,minlocLev] = min(cLev);
```

APPENDIX V – Cell Sorter Simulation Script

```
%-------------------------------------------------------------------------
% Project:      Cell Sorter Simulation Script
% Author:       Tom Myers
% Date Created: -
%
% Description:  Latest MFS code for PhD Thesis
%                For the cell sorter
%-------------------------------------------------------------------------
flbinaryfile = 'Chapter_8.mph';

n = 10;            %How many runs for the fault free simulation
FaultStates = 3;   %How many faults to increment through

FaultNumber = 8;   %Starting Fault Number

TotalSimCycles = n*FaultStates;

disp(TotalSimCycles)

%FaultInjection = 0; %if 1 inject faults
                     %if 0 fault-free model

Nominal = 0;         %if 1 simulate with fixed value parameters
                     %if 0 simulate with nominal input parameters

FaultFreeFirst = 1; %if 1 then simulate nominal fault-free first
                     %if 0 then go straight into fault simulations

FaultFreeSims  = 50; %Number of fault-free runs


%=========================================================================
%Outer Simulation Loop
%NOTE:
%       One loop of the outer loop for each Fault State
%=========================================================================
if(FaultFreeFirst == 1)
    %=====================================================================
    % Fault-Free Simulations
    %=====================================================================
    for FF = 1:FaultFreeSims

        disp('Iteration Number')
        disp(FF)                 %Iteration loop number for sanity check
        disp(clock)             %[year month day hour minute seconds]

        %Record Start of Cycle Time
        FaultFreeCycleTime(FF,:) = clock;

               %=================================================================
        %Save the FEM Structure
        %=====================================================================
        % Geometry
        clear draw
        g2=flbinary('g2','draw',flbinaryfile);
        g5=flbinary('g5','draw',flbinaryfile);
        g4=flbinary('g4','draw',flbinaryfile);
        draw.p.objs = {g2,g5,g4};
        draw.p.name = {'PT1','PT4','PT3'};
        draw.p.tags = {'g2','g5','g4','g3'};
        g1=flbinary('g1','draw',flbinaryfile);
        draw.s.objs = {g1};
        draw.s.name = {'EXT1'};
        draw.s.tags = {'g1'};
        fem.draw = draw;
        fem.mesh = flbinary('m1','mesh',flbinaryfile);

        femt = fem; %work on a copy of the structure - re-load purposes

        %=====================================================================
        % User Defined specification
        %=====================================================================
        %The 'normrnd' function is used to obtain a random value from within
        %the normal distribution. This is created from the user defined
```

```
%sepcification, in the presence of no inductive fault statistics.
%=====================================================================

%Need to parse constants to find values, or variable names!
parameter1 = normrnd(1e10, (1e10*0.25));      %Electrode 1 Degradation ON
parameter2 = normrnd(1e10, (1e10*0.25));      %Electrode 2 Degradation ON
parameter3 = normrnd(1e10, (1e10*0.25));      %Electrode 1 Degradation ON
parameter4 = normrnd(0, (0*0.05));          %Drive OFF Voltage
parameter5 = normrnd(200, (200*0.05));      %Drive ON Voltage

s = sprintf('%10.2d',parameter1);
femt.const{2} = s;
FFParaLog(FF,1) = parameter1;
FFCond(FF,1) = {'Electrode 1 Resistance'};

s = sprintf('%10.2d',parameter2);
femt.const{4} = s;
FFParaLog(FF,2) = parameter2;
FFCond(FF,2) = {'Electrode 2 Resistance'};

s = sprintf('%10.2d',parameter3);
femt.const{10} = s;
FFParaLog(FF,3) = parameter3;
FFCond(FF,3) = {'Electrode 3 Resistance};

s = sprintf('%10.2d[V]',parameter4);
femt.const{14} = s;
FFParaLog(FF,4) = parameter4;
FFCond(FF,4) = {'Drive OFF Voltage'};

s = sprintf('%10.2d[V]',parameter5);
femt.const{16} = s;
FFParaLog(FF,5) = parameter5;
FFCond(FF,5) = {'Drive ON Voltage'};

%Compile Model
femt = multiphysics(femt);

 %=====================================================================
 %CENTRE CHANNEL SIMULATION
%=====================================================================
% Time-Dependant Solver
%=====================================================================
% Extend mesh
femt.xmesh=meshextend(femt, ...
    'geoms',[1], ...
    'eqvars','on', ...
    'cplbndeq','on', ...
    'cplbndsh','off', ...
    'linshape',[1], ...
    'linshapetol',0.1);

% Solve problem
femt.sol=femtime(femt, ...
    'u',0, ...
    'method','eliminate', ...
    'nullfun','auto', ...
    'complexfun','off', ...
    'matherr','on', ...
    'solfile','off', ...
    'conjugate','off', ...
    'symmetric','auto', ...
    'solcomp',{'u2','p2','c','v2','w2'}, ...
    'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
    'rowscale','on', ...
    'blocksize','auto', ...
    'reacf','on', ...
    'odesolver','bdf_ida', ...
    'tlist',[colon(0,1,50)], ...
    'rtol',0.01, ...
    'masssingular','maybe', ...
    'consistent','bweuler', ...
    'estrat',1, ...
    'tout','tlist', ...
    'tsteps','free', ...
    'complex','on', ...
    'atol',{'0.0010'}, ...
    'maxorder',5, ...
    'minorder',1, ...
    'callback','postcallback', ...
```

```
'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:                Concentration,                c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.3237763717770 58E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739199,'camprojection'
,'orthographic','transparency',1.0}, ...
            'linsolver','spooles', ...
            'thresh',0.1, ...
            'preorder','nd', ...
            'uscale','none', ...
            'mcase',0);

        femt0=femt;
        %=========================================================================
        %Parametric Solver - (emqvw)
        %Note: Default when model loaded is Left
        %=========================================================================
        % Multiphysics
        femt=multiphysics(femt);

        % Extend mesh
        femt.xmesh=meshextend(femt, ...
                        'geoms',[1], ...
                        'eqvars','on', ...
                        'cplbndeq','on', ...
                        'cplbndsh','off', ...
                        'linshape',[1], ...
                        'linshapetol',0.1);

        % Solve problem
        femt.sol=femstatic(femt, ...
            'init',femt0.sol, ...
            'u',femt0.sol, ...
            'method','eliminate', ...
            'nullfun','auto', ...
            'complexfun','off', ...
            'matherr','on', ...
            'solfile','off', ...
            'conjugate','off', ...
            'symmetric','auto', ...
            'solcomp',{'Vportconstr1_g1_emqvw','V'}, ...
            'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
            'rowscale','on', ...
            'blocksize','auto', ...
            'reacf','on', ...
            'pname','nu_emqvw', ...
            'plist',[colon(10000,50000,500000)], ...
            'porder',1, ...
            'oldcomp',{}, ...
            'ntol',1.0E-6, ...
            'maxiter',25, ...
            'nonlin','auto', ...
            'damping','on', ...
            'hnlin','on', ...
            'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:                Concentration,                c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.3237763717770 58E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739201,'camprojection'
,'orthographic','transparency',1.0}, ...
            'linsolver','pardiso', ...
            'pardreorder','nd', ...
            'pardrreorder','on', ...
            'pivotstrategy','off', ...
            'pivotperturb','1.0E-8', ...
            'itol',1.0E-6, ...
            'rhob',20, ...
            'errorchk','on', ...
            'uscale','none', ...
            'mcase',0);

        %Impedance Data Record
```

```
FFImpedanceMag1(FF,1) = postglobaleval(femt,{'abs(Z11_emqvw)'});
FFImpedancePhi1(FF,1) = postglobaleval(femt,{'180*arg(Z11_emqvw)/pi'});


 %=========================================================================
 %RIGHT CHANNEL SIMULATION
 %=========================================================================
% Switch IS to scan Right Channel
% Note: this occurs for both the fault-free and fault model
%       boundary 1 is loaded as ground, boundary 11 needs to be ground
%=================================================================
%Only a single subdomain
femt.appl{1}.equ.sigma = {'conductivityR'};

%No rubber-band function required here as the geometry remains
%fixed from when it was designed in the COMSOL GUI
femt.appl{1}.bnd.ind = [2 2 2 2 2 2 3 2 2 2 1];

%Compile Model
femt = geomanalyze(femt);
%Compile Model
femt = multiphysics(femt);

%=========================================================================
% Time-Dependant Solver
%=========================================================================
% Extend mesh
femt.xmesh=meshextend(femt, ...
    'geoms',[1], ...
    'eqvars','on', ...
    'cplbndeq','on', ...
    'cplbndsh','off', ...
    'linshape',[1], ...
    'linshapetol',0.1);

% Solve problem
femt.sol=femtime(femt, ...
    'u',0, ...
    'method','eliminate', ...
    'nullfun','auto', ...
    'complexfun','off', ...
    'matherr','on', ...
    'solfile','off', ...
    'conjugate','off', ...
    'symmetric','auto', ...
    'solcomp',{'u2','p2','c','v2','w2'}, ...
    'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
    'rowscale','on', ...
    'blocksize','auto', ...
    'reacf','on', ...
    'odesolver','bdf_ida', ...
    'tlist',[colon(0,1,50)], ...
    'rtol',0.01, ...
    'masssingular','maybe', ...
    'consistent','bweuler', ...
    'estrat',1, ...
    'tout','tlist', ...
    'tsteps','free', ...
    'complex','on', ...
    'atol',{'0.0010'}, ...
    'maxorder',5, ...
    'minorder',1, ...
    'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:                Concentration,                c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739199,'camprojection'
,'orthographic','transparency',1.0}, ...
    'linsolver','spooles', ...
    'thresh',0.1, ...
    'preorder','nd', ...
    'uscale','none', ...
    'mcase',0);

femt0=femt;
```

```
%=========================================================================
% Parametric Solver - (emqvw)
% Note: Second Parametric Solve with Right Channel
%=========================================================================
% Solve problem
femt.sol=femstatic(femt, ...
    'init',femt0.sol, ...
    'u',femt0.sol, ...
    'method','eliminate', ...
    'nullfun','auto', ...
    'complexfun','off', ...
    'matherr','on', ...
    'solfile','off', ...
    'conjugate','off', ...
    'symmetric','auto', ...
    'solcomp',{'Vportconstr1_g1_emqvw','V'}, ...
    'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
    'rowscale','on', ...
    'blocksize','auto', ...
    'reacf','on', ...
    'pname','nu_emqvw', ...
    'plist',[colon(10000,50000,500000)], ...
    'porder',1, ...
    'oldcomp',{}, ...
    'ntol',1.0E-6, ...
    'maxiter',25, ...
    'nonlin','auto', ...
    'damping','on', ...
    'hnlin','on', ...
    'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:               Concentration,               c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739201,'camprojection'
,'orthographic','transparency',1.0}, ...
    'linsolver','pardiso', ...
    'pardreorder','nd', ...
    'pardrreorder','on', ...
    'pivotstrategy','off', ...
    'pivotperturb','1.0E-8', ...
    'itol',1.0E-6, ...
    'rhob',20, ...
    'errorchk','on', ...
    'uscale','none', ...
    'mcase',0);

%Impedance Data Record
FFImpedanceMag2(FF,1) = postglobaleval(femt,{'abs(Z11_emqvw)'});
FFImpedancePhi2(FF,1) = postglobaleval(femt,{'180*arg(Z11_emqvw)/pi'});

 %=========================================================================
 %LEFT CHANNEL SIMULATION
 %=========================================================================
% Switch IS to scan LEFT Channel
% Note: this occurs for both the fault-free and fault model
%       boundary 1 is loaded as ground, boundary 11 needs to be ground
%=========================================================================
%Only a single subdomain
femt.appl{1}.equ.sigma = {'conductivityL'};

%No rubber-band function required here as the geometry remains
%fixed from when it was designed in the COMSOL GUI
femt.appl{1}.bnd.ind = [2 1 2 2 2 2 3 2 2 2 2];

%Compile Model
femt = geomanalyze(femt);
%Compile Model
femt = multiphysics(femt);

%=========================================================================
% Time-Dependant Solver
%=========================================================================
% Extend mesh
femt.xmesh=meshextend(femt, ...
    'geoms',[1], ...
    'eqvars','on', ...
```

```
            'cplbndeq','on', ...
            'cplbndsh','off', ...
            'linshape',[1], ...
            'linshapetol',0.1);

        % Solve problem
        femt.sol=femtime(femt, ...
            'u',0, ...
            'method','eliminate', ...
            'nullfun','auto', ...
            'complexfun','off', ...
            'matherr','on', ...
            'solfile','off', ...
            'conjugate','off', ...
            'symmetric','auto', ...
            'solcomp',{'u2','p2','c','v2','w2'}, ...
            'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
            'rowscale','on', ...
            'blocksize','auto', ...
            'reacf','on', ...
            'odesolver','bdf_ida', ...
            'tlist',[colon(0,1,50)], ...
            'rtol',0.01, ...
            'masssingular','maybe', ...
            'consistent','bweuler', ...
            'estrat',1, ...
            'tout','tlist', ...
            'tsteps','free', ...
            'complex','on', ...
            'atol',{'0.0010'}, ...
            'maxorder',5, ...
            'minorder',1, ...
            'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:              Concentration,              c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739199,'camprojection'
,'orthographic','transparency',1.0}, ...
            'linsolver','spooles', ...
            'thresh',0.1, ...
            'preorder','nd', ...
            'uscale','none', ...
            'mcase',0);

        femt0=femt;

        %=====================================================================
        % Parametric Solver - (emqvw)
        % Note: Second Parametric Solve with Right Channel
        %=====================================================================
        % Solve problem
        femt.sol=femstatic(femt, ...
            'init',femt0.sol, ...
            'u',femt0.sol, ...
            'method','eliminate', ...
            'nullfun','auto', ...
            'complexfun','off', ...
            'matherr','on', ...
            'solfile','off', ...
            'conjugate','off', ...
            'symmetric','auto', ...
            'solcomp',{'Vportconstr1_g1_emqvw','V'}, ...
            'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
            'rowscale','on', ...
            'blocksize','auto', ...
            'reacf','on', ...
            'pname','nu_emqvw', ...
            'plist',[colon(10000,50000,500000)], ...
            'porder',1, ...
            'oldcomp',{}, ...
            'ntol',1.0E-6, ...
            'maxiter',25, ...
            'nonlin','auto', ...
            'damping','on', ...
            'hnlin','on', ...
            'callback','postcallback', ...
```

```
'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:              Concentration,              c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739201,'camprojection'
,'orthographic','transparency',1.0}, ...
            'linsolver','pardiso', ...
            'pardreorder','nd', ...
            'pardrreorder','on', ...
            'pivotstrategy','off', ...
            'pivotperturb','1.0E-8', ...
            'itol',1.0E-6, ...
            'rhob',20, ...
            'errorchk','on', ...
            'uscale','none', ...
            'mcase',0);



        %Impedance Data Record
        FFImpedanceMag3(FF,1) = postglobaleval(femt,{'abs(Z11_emqvw)'});
        FFImpedancePhi3(FF,1) = postglobaleval(femt,{'180*arg(Z11_emqvw)/pi'});


    end;

else
    %=========================================================================
    % Fault Simulations
    %=========================================================================
    for u = 1:FaultStates

    %=========================================================================
    %Inner Simulation Loop
    %NOTE:
    %       One loop of the inner loop per nominal parameter simulation
    %=========================================================================
        for i = 1:n

        disp('Iteration Number')
        disp(i)                 %Iteration loop number for sanity check
        disp('Fault Number')
        disp(u)                 %Iteration loop number for sanity check
        disp(clock)             %[year month day hour minute seconds]

        %Record Start of Cycle Time
        FaultCycleTime(u,i,:) = clock;

        %allows monitoring of simulation efficieny

        %=====================================================================
        %Save the FEM Structure
        %=====================================================================
        % Geometry
        clear draw
        g2=flbinary('g2','draw',flbinaryfile);
        g5=flbinary('g5','draw',flbinaryfile);
        g4=flbinary('g4','draw',flbinaryfile);
        draw.p.objs = {g2,g5,g4,g3};
        draw.p.name = {'PT1','PT4','PT3'};
        draw.p.tags = {'g2','g5','g4'};
        g1=flbinary('g1','draw',flbinaryfile);
        draw.s.objs = {g1};
        draw.s.name = {'EXT1'};
        draw.s.tags = {'g1'};
        fem.draw = draw;
        fem.mesh = flbinary('m1','mesh',flbinaryfile);

        femt = fem; %work on a copy of the structure - re-load purposes

        %=====================================================================
        % User Defined specification
        %=====================================================================
        %The 'normrnd' function is used to obtain a random value from within
        %the normal distribution. This is created from the user defined
        %sepcification, in the presence of no inductive fault statistics.
        %=====================================================================
```

```
        %Need to parse constants to find values, or variable names!
        parameter1 = normrnd(1e10, (1e10*0.25));      %Electrode 1 Degradation ON
        parameter2 = normrnd(1e10, (1e10*0.25));      %Electrode 2 Degradation ON
        parameter3 = normrnd(1e10, (1e10*0.25));      %Electrode 1 Degradation ON
        parameter4 = normrnd(0, (0*0.05));          %Drive OFF Voltage
        parameter5 = normrnd(200, (200*0.05));      %Drive ON Voltage

        %=========================================================================
        %Determine whether fixed or statistical nominal values are used
        %Note:
        %=========================================================================
        if(Nominal == 0)

                s = sprintf('%10.2d',parameter1);
                femt.const{2} = s;
                FFParaLog(FF,1) = parameter1;
                FFCond(FF,1) = {'Electrode 1 Resistance'};

                s = sprintf('%10.2d',parameter2);
                femt.const{4} = s;
                FFParaLog(FF,2) = parameter2;
                FFCond(FF,2) = {'Electrode 2 Resistance'};

                s = sprintf('%10.2d',parameter3);
                femt.const{10} = s;
                FFParaLog(FF,3) = parameter3;
                FFCond(FF,3) = {'Electrode 3 Resistance};

                s = sprintf('%10.2d[V]',parameter4);
                femt.const{14} = s;
                FFParaLog(FF,4) = parameter4;
                FFCond(FF,4) = {'Drive OFF Voltage'};

                s = sprintf('%10.2d[V]',parameter5);
                femt.const{16} = s;
                FFParaLog(FF,5) = parameter5;
                FFCond(FF,5) = {'Drive ON Voltage'};

        else
            %fixed values
            disp('fixed values')
        end;

        %=========================================================================
        %Determine Fault Simulation Type
        %Note:
        %=========================================================================
        switch FaultNumber


                % Cell Blockage 3
            case 7
                FaultCond(u,i,6)={'Cell Blockage 3’};

                disp('Cell Blockage 3’)

                % Geometry
                %figure(1)
                g6=block3('98e-6','5e-3','98e-6','base','corner','pos',{'4e-3','-4e-
3','0'},'axis',{'0','0','1'},'rot','45')
                g7=geomcomp({g1,g6},'ns',{'EXT1','BLK1'},'sf','EXT1+BLK1','face','none','edge','all')
                %geomplot(g7)

                % Analyzed geometry
                clear p s
                femt.draw.p.objs={g2,g5,g4,g3};
                femt.draw.p.name={'PT1','PT4','PT3'};
                femt.draw.p.tags={'g2','g5','g4''};

                femt.draw.s.objs={g7};
                femt.draw.s.name={'CO1'};
                femt.draw.s.tags={'g7'};

                femt.draw=struct('p',femt.draw.p,'s',femt.draw.s);

                femt = geomanalyze(femt);

                % Initialize mesh
                femt.mesh=meshinit(femt, ...
                    'hauto',6, ...
```

```
                    'hpnt',20, ...
                    'xscale',1.0, ...
                    'yscale',1.0, ...
                    'zscale',1.0, ...
                    'jiggle','on', ...
                    'methodfac','tri');

                %Compile Model
                femt = multiphysics(femt);

                %Cell Blockage 2
            case 6

                FaultCond(u,i,6)={'Cell Blockage 2'};

                disp('Cell Blockage 2')

                % Geometry
                %figure(1)
                g6=block3('98e-6','5e-3','98e-6','base','corner','pos',{'8e-3','-8e-
3','0'},'axis',{'0','0','1'},'rot','45')
                g7=geomcomp({g1,g6},'ns',{'EXT1','BLK1'},'sf','EXT1+BLK1','face','none','edge','all')
                %geomplot(g7)

                % Analyzed geometry
                clear p s
                femt.draw.p.objs={g2,g5,g4,g3};
                femt.draw.p.name={'PT1','PT4','PT3'};
                femt.draw.p.tags={'g2','g5','g4''};

                femt.draw.s.objs={g7};
                femt.draw.s.name={'CO1'};
                femt.draw.s.tags={'g7'};

                femt.draw=struct('p',femt.draw.p,'s',femt.draw.s);

                femt = geomanalyze(femt);

                % Initialize mesh
                femt.mesh=meshinit(femt, ...
                    'hauto',6, ...
                    'hpnt',20, ...
                    'xscale',1.0, ...
                    'yscale',1.0, ...
                    'zscale',1.0, ...
                    'jiggle','on', ...
                    'methodfac','tri');

                %Compile Model
                femt = multiphysics(femt);

                %Cell Blockage 1
            case 5

                FaultCond(u,i,6)={'Cell Blockage 1'};

                disp('Cell Blockage 1')

                % Geometry
                %figure(1)
                g6=block3('98e-6','5e-3','98e-6','base','corner','pos',{'1.78e-3','-1.78e-
3','0'},'axis',{'0','0','1'},'rot','45')
                g7=geomcomp({g1,g6},'ns',{'EXT1','BLK1'},'sf','EXT1+BLK1','face','none','edge','all')
                %geomplot(g7)

                % Analyzed geometry
                clear p s
                femt.draw.p.objs={g2,g5,g4,g3};
                femt.draw.p.name={'PT1','PT4','PT3'};
                femt.draw.p.tags={'g2','g5','g4''};

                femt.draw.s.objs={g7};
                femt.draw.s.name={'CO1'};
                femt.draw.s.tags={'g7'};

                femt.draw=struct('p',femt.draw.p,'s',femt.draw.s);

                femt = geomanalyze(femt);

                % Initialize mesh
                femt.mesh=meshinit(femt, ...
```

```
            'hauto',6, ...
            'hpnt',20, ...
            'xscale',1.0, ...
            'yscale',1.0, ...
            'zscale',1.0, ...
            'jiggle','on', ...
            'methodfac','tri');

        %Compile Model
        femt = multiphysics(femt);


        %Drive OFF Voltage
    case 4

        s = sprintf('%10.2d',10);
        femt.const{2} = s;
        FaultParaLog(u,i,2) = parameter1;
        FaultCond(u,i,2) = {'Drive OFF Voltage'};

        FaultCond(u,i,6)={ Drive OFF Voltage'};

        disp(Drive OFF Voltage')

        %Drive ON Voltage
    case 3

        s = sprintf('%10.2d',1);
        femt.const{2} = s;
        FaultParaLog(u,i,2) = parameter1;
        FaultCond(u,i,2) = {'Drive ON Voltage'};

        FaultCond(u,i,6)={ Drive ON Voltage'};

        disp(Drive ON Voltage')

        %Electrode 3 Degradation
    case 2
        s = sprintf('%10.2d',1e10);
        femt.const{2} = s;
        FaultParaLog(u,i,2) = parameter1;
        FaultCond(u,i,2) = {'Electrode 3 Degradation'};

        FaultCond(u,i,6)={'Electrode 3 Degradation' };

        disp('Electrode 3 Degradation')


        %Electrode 2 Degradation
    case 1

        s = sprintf('%10.2d',1e10);
        femt.const{2} = s;
        FaultParaLog(u,i,2) = parameter1;
        FaultCond(u,i,2) = {'Electrode 2 Degradation'};

        FaultCond(u,i,6)={'Electrode 2 Degradation' };

        disp('Electrode 2 Degradation')

        %Electrode 1 Degradation
    case 0

        s = sprintf('%10.2d',1e10);
        femt.const{2} = s;
        FaultParaLog(u,i,2) = parameter1;
        FaultCond(u,i,2) = {'Electrode 1 Degradation'};

        FaultCond(u,i,6)={'Electrode 1 Degradation' };

        disp('Electrode 1 Degradation')

end;

%Compile Model
femt = multiphysics(femt);

 %======================================================================
 %CENTRE CHANNEL SIMULATION
%======================================================================
% Time-Dependant Solver
```

```
%======================================================================
% Extend mesh
femt.xmesh=meshextend(femt, ...
    'geoms',[1], ...
    'eqvars','on', ...
    'cplbndeq','on', ...
    'cplbndsh','off', ...
    'linshape',[1], ...
    'linshapetol',0.1);

% Solve problem
femt.sol=femtime(femt, ...
    'u',0, ...
    'method','eliminate', ...
    'nullfun','auto', ...
    'complexfun','off', ...
    'matherr','on', ...
    'solfile','off', ...
    'conjugate','off', ...
    'symmetric','auto', ...
    'solcomp',{'u2','p2','c','v2','w2'}, ...
    'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
    'rowscale','on', ...
    'blocksize','auto', ...
    'reacf','on', ...
    'odesolver','bdf_ida', ...
    'tlist',[colon(0,1,50)], ...
    'rtol',0.01, ...
    'masssingular','maybe', ...
    'consistent','bweuler', ...
    'estrat',1, ...
    'tout','tlist', ...
    'tsteps','free', ...
    'complex','on', ...
    'atol',{'0.0010'}, ...
    'maxorder',5, ...
    'minorder',1, ...
    'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:               Concentration,               c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739199,'camprojection'
,'orthographic','transparency',1.0}, ...
    'linsolver','spooles', ...
    'thresh',0.1, ...
    'preorder','nd', ...
    'uscale','none', ...
    'mcase',0);

femt0=femt;
%======================================================================
%Parametric Solver - (emqvw)
%Note: Default when model loaded is Left
%======================================================================
% Multiphysics
femt=multiphysics(femt);

% Extend mesh
femt.xmesh=meshextend(femt, ...
                      'geoms',[1], ...
                      'eqvars','on', ...
                      'cplbndeq','on', ...
                      'cplbndsh','off', ...
                      'linshape',[1], ...
                      'linshapetol',0.1);

% Solve problem
femt.sol=femstatic(femt, ...
    'init',femt0.sol, ...
    'u',femt0.sol, ...
    'method','eliminate', ...
    'nullfun','auto', ...
    'complexfun','off', ...
    'matherr','on', ...
    'solfile','off', ...
    'conjugate','off', ...
```

```
            'symmetric','auto', ...
            'solcomp',{'Vportconstr1_g1_emqvw','V'}, ...
            'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
            'rowscale','on', ...
            'blocksize','auto', ...
            'reacf','on', ...
            'pname','nu_emqvw', ...
            'plist',[colon(10000,50000,500000)], ...
            'porder',1, ...
            'oldcomp',{}, ...
            'ntol',1.0E-6, ...
            'maxiter',25, ...
            'nonlin','auto', ...
            'damping','on', ...
            'hnlin','on', ...
            'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:                Concentration,                c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739201,'camprojection'
,'orthographic','transparency',1.0}, ...
            'linsolver','pardiso', ...
            'pardreorder','nd', ...
            'pardrreorder','on', ...
            'pivotstrategy','off', ...
            'pivotperturb','1.0E-8', ...
            'itol',1.0E-6, ...
            'rhob',20, ...
            'errorchk','on', ...
            'uscale','none', ...
            'mcase',0);

        %Impedance Data Record
        FFImpedanceMag1(FF,1) = postglobaleval(femt,{'abs(Z11_emqvw)'});
        FFImpedancePhi1(FF,1) = postglobaleval(femt,{'180*arg(Z11_emqvw)/pi'});


         %=========================================================================
         %RIGHT CHANNEL SIMULATION
         %=========================================================================
        % Switch IS to scan Right Channel
        % Note: this occurs for both the fault-free and fault model
        %       boundary 1 is loaded as ground, boundary 11 needs to be ground
        %=================================================================
        %Only a single subdomain
        femt.appl{1}.equ.sigma = {'conductivityR'};

        %No rubber-band function required here as the geometry remains
        %fixed from when it was designed in the COMSOL GUI
        femt.appl{1}.bnd.ind = [2 2 2 2 2 2 3 2 2 2 1];

        %Compile Model
        femt = geomanalyze(femt);
        %Compile Model
        femt = multiphysics(femt);

        %=========================================================================
        % Time-Dependant Solver
        %=========================================================================
        % Extend mesh
        femt.xmesh=meshextend(femt, ...
            'geoms',[1], ...
            'eqvars','on', ...
            'cplbndeq','on', ...
            'cplbndsh','off', ...
            'linshape',[1], ...
            'linshapetol',0.1);

        % Solve problem
        femt.sol=femtime(femt, ...
            'u',0, ...
            'method','eliminate', ...
            'nullfun','auto', ...
            'complexfun','off', ...
            'matherr','on', ...
            'solfile','off', ...
```

```
            'conjugate','off', ...
            'symmetric','auto', ...
            'solcomp',{'u2','p2','c','v2','w2'}, ...
            'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
            'rowscale','on', ...
            'blocksize','auto', ...
            'reacf','on', ...
            'odesolver','bdf_ida', ...
            'tlist',[colon(0,1,50)], ...
            'rtol',0.01, ...
            'masssingular','maybe', ...
            'consistent','bweuler', ...
            'estrat',1, ...
            'tout','tlist', ...
            'tsteps','free', ...
            'complex','on', ...
            'atol',{'0.0010'}, ...
            'maxorder',5, ...
            'minorder',1, ...
            'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:              Concentration,              c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739199,'camprojection'
,'orthographic','transparency',1.0}, ...
            'linsolver','spooles', ...
            'thresh',0.1, ...
            'preorder','nd', ...
            'uscale','none', ...
            'mcase',0);

        femt0=femt;

        %=====================================================================
        % Parametric Solver - (emqvw)
        % Note: Second Parametric Solve with Right Channel
        %=====================================================================
        % Solve problem
        femt.sol=femstatic(femt, ...
            'init',femt0.sol, ...
            'u',femt0.sol, ...
            'method','eliminate', ...
            'nullfun','auto', ...
            'complexfun','off', ...
            'matherr','on', ...
            'solfile','off', ...
            'conjugate','off', ...
            'symmetric','auto', ...
            'solcomp',{'Vportconstr1_g1_emqvw','V'}, ...
            'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
            'rowscale','on', ...
            'blocksize','auto', ...
            'reacf','on', ...
            'pname','nu_emqvw', ...
            'plist',[colon(10000,50000,500000)], ...
            'porder',1, ...
            'oldcomp',{}, ...
            'ntol',1.0E-6, ...
            'maxiter',25, ...
            'nonlin','auto', ...
            'damping','on', ...
            'hnlin','on', ...
            'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:              Concentration,              c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739201,'camprojection'
,'orthographic','transparency',1.0}, ...
            'linsolver','pardiso', ...
            'pardreorder','nd', ...
            'pardrreorder','on', ...
```

```
        'pivotstrategy','off', ...
        'pivotperturb','1.0E-8', ...
        'itol',1.0E-6, ...
        'rhob',20, ...
        'errorchk','on', ...
        'uscale','none', ...
        'mcase',0);

    %Impedance Data Record
    FFImpedanceMag2(FF,1) = postglobaleval(femt,{'abs(Z11_emqvw)'});
    FFImpedancePhi2(FF,1) = postglobaleval(femt,{'180*arg(Z11_emqvw)/pi'});

     %=========================================================================
     %LEFT CHANNEL SIMULATION
     %=========================================================================
    % Switch IS to scan LEFT Channel
    % Note: this occurs for both the fault-free and fault model
    %       boundary 1 is loaded as ground, boundary 11 needs to be ground
    %=================================================================
    %Only a single subdomain
    femt.appl{1}.equ.sigma = {'conductivityL'};

    %No rubber-band function required here as the geometry remains
    %fixed from when it was designed in the COMSOL GUI
    femt.appl{1}.bnd.ind = [2 1 2 2 2 2 3 2 2 2 2];

    %Compile Model
    femt = geomanalyze(femt);
    %Compile Model
    femt = multiphysics(femt);

    %=========================================================================
    % Time-Dependant Solver
    %=========================================================================
    % Extend mesh
    femt.xmesh=meshextend(femt, ...
        'geoms',[1], ...
        'eqvars','on', ...
        'cplbndeq','on', ...
        'cplbndsh','off', ...
        'linshape',[1], ...
        'linshapetol',0.1);

    % Solve problem
    femt.sol=femtime(femt, ...
        'u',0, ...
        'method','eliminate', ...
        'nullfun','auto', ...
        'complexfun','off', ...
        'matherr','on', ...
        'solfile','off', ...
        'conjugate','off', ...
        'symmetric','auto', ...
        'solcomp',{'u2','p2','c','v2','w2'}, ...
        'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
        'rowscale','on', ...
        'blocksize','auto', ...
        'reacf','on', ...
        'odesolver','bdf_ida', ...
        'tlist',[colon(0,1,50)], ...
        'rtol',0.01, ...
        'masssingular','maybe', ...
        'consistent','bweuler', ...
        'estrat',1, ...
        'tout','tlist', ...
        'tsteps','free', ...
        'complex','on', ...
        'atol',{'0.0010'}, ...
        'maxorder',5, ...
        'minorder',1, ...
        'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:               Concentration,              c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739199,'camprojection'
,'orthographic','transparency',1.0}, ...
```

```
            'linsolver','spooles', ...
            'thresh',0.1, ...
            'preorder','nd', ...
            'uscale','none', ...
            'mcase',0);

        femt0=femt;

        %=====================================================================
        % Parametric Solver - (emqvw)
        % Note: Second Parametric Solve with Right Channel
        %=====================================================================
        % Solve problem
        femt.sol=femstatic(femt, ...
            'init',femt0.sol, ...
            'u',femt0.sol, ...
            'method','eliminate', ...
            'nullfun','auto', ...
            'complexfun','off', ...
            'matherr','on', ...
            'solfile','off', ...
            'conjugate','off', ...
            'symmetric','auto', ...
            'solcomp',{'Vportconstr1_g1_emqvw','V'}, ...
            'outcomp',{'Vportconstr1_g1_emqvw','V','u2','p2','c','v2','w2'}, ...
            'rowscale','on', ...
            'blocksize','auto', ...
            'reacf','on', ...
            'pname','nu_emqvw', ...
            'plist',[colon(10000,50000,500000)], ...
            'porder',1, ...
            'oldcomp',{}, ...
            'ntol',1.0E-6, ...
            'maxiter',25, ...
            'nonlin','auto', ...
            'damping','on', ...
            'hnlin','on', ...
            'callback','postcallback', ...

'callbparam',{'slicedata',{'c','cont','internal','recover','off','unit','mol/m^3'},'slicexspacing',0,'sliceyspa
cing',0,'slicezspacing',1,'sliceedgestyle','none','slicefacestyle','interp','slicebar','on','slicemap','Rainbow
','slicemapstyle','auto','solnum','end','phase',0,'title','Slice:               Concentration,               c
[mol/m^3]','refine','auto','geom','on','geomnum',1,'sdl',{[1]},'complexfun','on','matherr','off','axisvisible',
'on','axisequal','on','grid','on','camlight','off','scenelight','off','campos',[-0.001167642643415917,-
0.049373822117261344,0.024408344314566763],'camtarget',[5.000003147870302E-5,7.323776371777058E-
4,1.9999999494757503E-
5],'camup',[0.3120376174052381,0.4096520367531424,0.8572151037563819],'camva',7.239368362739201,'camprojection'
,'orthographic','transparency',1.0}, ...
            'linsolver','pardiso', ...
            'pardreorder','nd', ...
            'pardrreorder','on', ...
            'pivotstrategy','off', ...
            'pivotperturb','1.0E-8', ...
            'itol',1.0E-6, ...
            'rhob',20, ...
            'errorchk','on', ...
            'uscale','none', ...
            'mcase',0);



        %Impedance Data Record
        FFImpedanceMag3(FF,1) = postglobaleval(femt,{'abs(Z11_emqvw)'});
        FFImpedancePhi3(FF,1) = postglobaleval(femt,{'180*arg(Z11_emqvw)/pi'});

        %Record End of Cycle Time
        InnerCycleTimeEnd(u,i,:) = clock;
        end;

        %Move to the Next Fault
        FaultNumber = FaultNumber+1;
        %Record End of Cycle Time
        OuterCycleTimeEnd(u,:) = clock;

    end;

end;

disp('Finished - Enjoy Analysing Faulty Microchemical Reactors!');
```

References

1. Whitesides, G.M., *The origins and future of microfluidics.* Nature, 2006. **442**(27): p. 368-373.
2. Yager, P., et al., *Microfluidic diagnostic technologies for global public health.* Nature, 2006. **442**(27): p. 412-418.
3. Perkel, J.M., *Microfluidics: bringing new things to life science*, in *Life Science Technologies*. 2008, www.sciencemag.org. p. 975-977.
4. Becker, H., *Hype, Hope and Hubris: the quest for the killer application in microfluidics.* Lab on a Chip, 2009. **9**: p. 2119-2122.
5. Marle, L. and G.M. Greenway, *Microfluidic devices for environmental monitoring.* TrAC Trends in Analytical Chemistry, 2005. **24**(9): p. 795-802.
6. Koushanfar, F., M. Potkonjak, and L.S.-V. Alberto. *On-line Fault Detection of Sensor Measurements*. in *Proc. of IEEE Sensors 2003*. 2003: IEEE.
7. Grace, R.H. (2009) *MEMS based systems solutions: Thinking outside the chip*. Micronanosystems
8. Manz, A., N. Graber, and H.M. Widmer, *Miniaturized total chemical analysis systems: A novel concept for chemical sensing.* Sensors and Actuators B: Chemical, 1990. **1**(1-6): p. 244-248.
9. *International Technology Roadmap for Semiconductors*. 2007 [cited 2008 November]; Test and Test Equipment:[Available from: http://www.itrs.net/.
10. Gaßmann, S., I. Ibendorf, and L. Pagel, *Realization of a flow injection analysis in PCB technology.* Sensors and Actuators A: Physical, 2007. **133**(1): p. 231-235.
11. Ebrahim, G., Zadeh, and S. Mohamad, *Towards fully integrated Lab-on-Chip: design, assembly and experimental results.* Int. J. Adv. Media Commun., 2009. **3**(1/2): p. 154-166.
12. Stokes, P.A. and R.E. Mallard, *Towards heterogeneous microsystems design-for-test in a graduate student environment*, in *International Conference on Microelectronic Systems Education*, IEEE, Editor. 2009. p. 81-84.
13. COMSOL, *COMSOL Multi-Physics*. 2008.
14. Gilbert, J.M. and I.M. Bell, *The Effectiveness of Test in Controlling Quality Costs: A Conformability Analysis Based Approach.* Journal of Electronic Testing Theory and Applications, 2007. **23**(4): p. 293-307.
15. Milor, L.S., *A tutorial introduction to research on analog and mixed-signal circuit testing.* Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, 1998. **45**(10): p. 1389-1407.
16. Burns, M. and G.W. Roberts, *An Introduction to Mixed-Signal IC Test and Measurements*. 2001: Oxford University Press.
17. Wilkins, B.R., *Testing Digital Circuits: An Introduction*. Aspects of Information Technology. 1986, Wokingham: Van Nostrand Reinhold (UK).
18. Boy, D.A., F. Gibou, and S. Pennathur, *Simulation tools for lab on a chip research: advantages, challenges, and thoughts for the future.* Lab on a Chip, 2008. **8**(9): p. 1424 - 1431.
19. Coventor, *MEMulator*. 2008.
20. Erickson, D., *Towards numerical prototyping of labs-on-chip: modeling for integrated microfluidic devices.* Microfluidics and Nanofluidics, 2005. **1**(4): p. 301-318.
21. Kerkhoff, H.G. and G. Hendriks, *Analogue Fault Modelling and Simulation Techniques in Electronic/Fluidic Microsystems.*
22. Kerkhoff, H.G. and H.P.A. Hendriks, *Fault Modeling and Fault Simulation in Mixed Micro-Fluidic Microelectronic Systems.* Journal of Electronic Testing, 2001. **17**(5): p. 427-437.
23. Berli, C., *Equivalent circuit modeling of electrokinetically driven analytical microsystems.* Microfluidics and Nanofluidics, 2008. **4**(5): p. 391-399.

24. Chatterjee, A.N. and N.R. Aluru, *Combined Circuit / Device Modelling and Simulation of Integrated Microfluidic Systems.* Journal of Microelectromechanical Systems, 2005. **14**(1): p. 81 - 95.
25. Wilson, P.R., et al. *Multiple Domain Behavioral Modelling Using VHDL-AMS*. in *Proc. of the IEEE Symposium on Circuits and Systems*. 2004: IEEE.
26. Voigt, P., G. Schrag, and G. Wachutka, *Microfluidic system modeling using VHDL-AMS and circuit simulation.* Microelectronics Journal, 1998. **29**(11): p. 791-797.
27. Kerkhoff, H.G., et al. *VHDL-AMS Fault Simulation for Testing DNA Bio-Sensing Arrays*. in *IEEE Sensors 2005*. 2005: IEEE.
28. Kerkhoff, H.G. and A. Mustafa, *Testable Design and Testing of Micro-Electro-Fluidic Arrays*, in *Proceedings of the 21st IEEE VLSI Test Symposium*. 2003, IEEE Computer Society.
29. Kerkhoff, H.G., et al., *Fault Modelling and Co-Simulation in FlowFET-Based Biological Array Systems*, in *Third IEEE International Workshop on Electronic Design, Test & Applications,DELTA*, IEEE, Editor. 2006: Kuala Lumpur, Malaysia.
30. Kerkhoff, H.G. and X. Zhang, *Fault co-simulation for test evaluation of heterogeneous integrated biological systems.* Microelectronics Journal, 2009. **40**(7): p. 1048-1053.
31. Schlegel, M., et al., *Analyzing and simulation of MEMS in VHDL-AMS based on reduced-order FE models.* IEEE Sensors Journal, 2005. **5**(5): p. 1019 - 1026.
32. Chatterjee, A., *Generalized numerical formulations for multi-physics microfluidics-type applications* in *Journal of Micromechanics and Microengineering*. 2003. p. 758.
33. Zhang, T., K. Chakrabarty, and R.B. Fair, *Integrated hierarchical design of microelectrofluidic systems using SystemC.* Microelectronics Journal, 2002. **33**(5-6): p. 459-470.
34. Zhang, T., K. Chakrabarty, and R.B. Fair. *Behavioral Modelling and Performance Evaluation of Microelectrofluidics - Based PCR Systems Using SystemC*. in *IEEE Transactions on Computer-Aided Design of Integrated Circuits And Systems*. 2004.
35. Wang, Y., et al. *System-oriented modeling and simulation of a Biofluidic Lap-on-a-Chip*. in *The Thirteenth International Conference on Solid-State Sensors, Actuators and Microsystems*. 2005. Seoul, Korea: IEEE.
36. Wang, Y., Q. Lin, and T. Mukherjee, *COMPOSABLE BEHAVIORAL MODELS AND SCHEMATIC-BASED SIMULATION OF ELECTROKINETIC LAB-ON-A-CHIP SYSTEMS*, in *Design Automation Methods and Tools for Microfluidics-Based Biochips*. 2006. p. 109-142.
37. Wang, Y., et al., *System-level modeling and simulation of biochemical assays in lab-on-a-chip devices.* Microfluidics and Nanofluidics, 2007. **3**(3): p. 307-322.
38. Roman, C., S. Mir, and B. Charlot, *Building an analogue fault simulation tool and its application to MEMS.* Microelectronics Journal, 2003. **34**(10): p. 897-906.
39. Reppa, V. and A. Tzes, *Application of Set Membership Identification for Fault Detection of MEMS*, in *IEEE International Conference on Robotics and Automation*, IEEE, Editor. 2006, IEEE: Orlando, Florida, USA. p. 643 - 648.
40. Reppa, V., M. Vagia, and A. Tzes, *Fault Detection using Set Membership Identification*, in *16th IEEE International Conference on Control Applications*, IEEE, Editor. 2007, IEEE: Singapore. p. 789 - 794.
41. Alippi, C., et al., *An embedded automatic SBT diagnosis system for analog circuits*, in *Instrumentation and Measurement Technology Conference, 2004. IMTC 04. Proceedings of the 21st IEEE*. 2004. p. 2156- 2160.
42. Manikandan, V. and N. Devarajan, *SBT Approach towards Analog Electronic Circuit Fault Diagnosis.* Active and Passive Electronic Components, 2007. **2007**: p. 12.
43. Walraven, J.A., *Future Challenges for MEMS Failure Analysis*, in *ITC International Test Conference*. 2003, IEEE. p. 850 - 855.
44. Gray, B.L., et al., *Mechanical and fluidic characterization of microfluidic interconnects for lab-on-a-chip applications*, in *Proceedings of the 2008 IEEE 14th International Mixed-Signals, Sensors, and Systems Test Workshop*. 2008, IEEE Computer Society.

45. Korivi, N.S. and L. Jiang. *A Generic Chip-to-World Fluidic Interconnect System for Microfluidic Devices*. in *Thirty-Ninth Southeastern Symposium on System Theory, 2007*. 2007. Macon, GA: IEEE.
46. Kano, M., et al., *Data-based and model-based blockage diagnosis for stacked microchemical processes.* Chemical Engineering Science, 2007. **62**(4): p. 1073-1080.
47. Chapman, G., B.L. Gray, and V.K. Jain. *Defect tolerance in microfluidic chambers for capacitive biosensors*. in *Microfluidics, BioMEMS, and Medical Microsystems VIII*. San Francisco, California, USA: SPIE.
48. Amador, C., A. Gavriilidis, and P. Angeli, *Flow distribution in different microreactor scale-out geometries and the effect of manufacturing tolerances and channel blockage.* Chemical Engineering Journal, 2004. **101**(1-3): p. 379-390.
49. Chow, A.W., *Prevention of Precipitate Blockage in Microfluidic Channels*, I. Caliper Life Sciences, Editor. 2008: USA.
50. Urbanski, J.P., et al., *The effect of step height on the performance of three-dimensional ac electro-osmotic microfluidic pumps.* Journal of Colloid and Interface Science, 2007. **309**(2): p. 332-341.
51. Liu, H., et al., *Embedded Test & Health Monitoring Strategies for Bio-Fluidic Microsystems*, in *2nd Electronics System-Integration Technology Conference, ESTC 2008*, IEEE, Editor. 2008: Greenwich, London,UK. p. 427 - 433.
52. Leung, S.-A., et al., *Continuous real-time bubble monitoring in microchannels using refractive index detection* in *Measurement Science and Technology*. 2004. p. 290.
53. Chien, R.L., *Voltage/Current Testing Equipment for Microfluidic Devices*, I. Caliper Life Sciences, Editor. 2007: USA.
54. Zhang, X., et al., *Dependable MEF Systems based on Control and Direct Sensing Electrodes via Current and Impedance Self-Tests*, in *18th annual workshop on Circuits, Systems and Signal Processing*. 2007: Veldhoven.
55. Kerkhoff, H.G., et al., *A dependable microelectronic peptide synthesizer using electrode data*, in *VLSI Design*. 2008. p. 1 - 9.
56. Deb, N. and R.D. Blanton, *Analysis of Failure Sources in Surface-Micromachined MEMS*, in *Proceedings of the 2000 IEEE International Test Conference*. 2000, IEEE Computer Society.
57. Östergren, K.C.E. and C. Trägårdh, *Characterization of hydrodynamic dispersion in a chromatographic column under compression.* Chemical Engineering Journal, 2000. **79**(2): p. 103-111.
58. Sun, Y., et al., *Design, Simulation and Experiment of Electroosmotic Microfluidic Chip for Cell Sorting.* Sensors and actuators A, 2007. **133**: p. 340-348.
59. Tang, G.Y. and et al., *Joule heating and its effects on electroosmotic flow in microfluidic channels.* Journal of Physics: Conference Series, 2006. **34**(1): p. 925.
60. Luo, C., et al., *Application of microfluidic chip to realize controllable pH gradient.* Chinese Science Bulletin, 2005. **50**(4): p. 365-367.
61. Herr, A.E., et al., *Electroosmotic Capillary Flow with Nonuniform Zeta Potential.* Analytical Chemistry, 2000. **72**(5): p. 1053-1057.
62. Su, F. and K. Chakrabarty, *Defect tolerance for gracefully-degradable microfluidics-based biochips*, in *VLSI Test Symposium*, IEEE, Editor. 2005. p. 321-326.
63. Zhang, T., K. Chakrabarty, and R.B. Fair, *Design of Reconfigurable Composite Microsystems Based on Hardware/Software Codesign Principles.* IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, 2002. **21**(8): p. 987-995.
64. Rufer, L., et al. *On-Chip testing of MEMS using pseudo-random test sequences*. in *Design, Test, Integration and Packaging of MEMS/MOEMS*. 2003.
65. Abhijeet, K. and R.D. Blanton, *Development of a MEMS Testing Methodology*, in *Proceedings of the 1997 IEEE International Test Conference*. 1997, IEEE Computer Society.

66. Charlot, B., et al., *Generation of Electrically Induced Stimuli for MEMS Self-Test.* Journal of Electronic Testing, 2001. **17**(6): p. 459-470.
67. Mir, S., B. Charlot, and B. Courtois, *Extending Fault-Based Testing to Microelectromechanical Systems.* Journal of Electronic Testing, 2000. **16**(3): p. 279-288.
68. Fair, R., *Digital microfluidics: is a true lab-on-a-chip possible?* Microfluidics and Nanofluidics, 2007. **3**(3): p. 245-281.
69. Kerkhoff, H.G *Testing Microelectronic Biofluidic Systems.* IEEE Des. Test, 2007. **24**(1): p. 72-82.
70. Venuto, D.D. and D. Laterza. *Layout Based Fault List Generation and Fault Simulation for DNA Sensor Arrays Testing*. in *International Mixed Signal Test Workshop*. 2007: IEEE.
71. De Venuto, D. and B. Riccò, *Fault diagnosis and test of DNA sensor arrays by using IFA approach.* Microelectronics Journal. **In Press, Corrected Proof**.
72. Karim, A. and K. Bozena, *Parametric and Catastrophic Fault Coverage of Analog Circuits in Oscillation-Test Methodology*, in *Proceedings of the 15th IEEE VLSI Test Symposium (VTS'97)*. 1997, IEEE Computer Society.
73. Mansfeld, F., *Use of electrochemical impedance spectroscopy for the study of corrosion protection by polymer coatings.* Journal of Applied Electrochemistry, 1995. **25**(3): p. 187-202.
74. Shao, Z. and S.M. Haile, *A high-performance cathode for the next generation of solid-oxide fuel cells.* Nature, 2004. **431**(7005): p. 170-173.
75. Liu, H. and A. Richardson. *Self-Testing of Micro-Electrode Array Implemented as a Bio-Sensor*. in *International Mixed Signal Test Workshop*. 2007: IEEE.
76. Ayliffe, H.E., A.B. Frazier, and R.D. Rabbit, *Electric Impedance Spectroscopy using Microchannels with Integrated Metal Electrodes.* IEEE Journal of Microelectromechanical Systems, 1999. **8**(1): p. 50 - 57.
77. Ayliffe, H.E. and R.D. Rabbit, *An Electric Impedance based Microelectromechanical system flow sensor for ionic solutions.* Journal of Measurement Science and Technology, 2003. **14**: p. 1321 - 1327.
78. Zou, Z., et al., *Functionalized nano interdigitated electrodes arrays on polymer with integrated microfluidics for direct bio-affinity sensing using impedimetric measurement.* Sensors and Actuators A: Physical, 2007. **136**(2): p. 518-526.
79. Ghafar-Zadeh, E. and M. Sawan, *A Hybrid Microfluidic/CMOS Capacitive Sensor Dedicated to Lab-on-Chip Applications.* IEEE Transactions on Biomedical Circuits and Systems, 2007. **1**(4): p. 270-277.
80. Ghafar-Zadeh, E., et al., *Bacteria growth monitoring through an on-chip capacitive sensor*, in *IEEE 14th International Mixed-Signals, Sensors, and Systems Test Workshop, 2008. IMS3TW 2008.*, IEEE, Editor. 2008, IEEE: Vancouver, BC. p. 1-4.
81. Su, F., S. Ozev, and K. Chakrabarty. *Testing of Droplet-Based Microelectrofluidic Systems.* in *Proc. ITC*. 2003.
82. Chakrabarty, K. *Design, Testing and Applications of Digital Microfluidics-Based Biochips.* in *Proc. Eighteenth International Conference on VLSI Design*. 2005: IEEE.
83. Fei, S. and C. Krishnendu, *Design of Fault-Tolerant and Dynamically-Reconfigurable Microfluidic Biochips*, in *Proceedings of the conference on Design, Automation and Test in Europe - Volume 2*. 2005, IEEE Computer Society.
84. Su, F., S. Ozev, and K. Chakrabarty, *Test Planning and Test Resource Optimization for Droplet-Based Microfluidic Systems.* Journal of Electronic Testing, 2006. **22**(2): p. 199-210.
85. Tao, X. and C. Krishnendu, *Parallel Scan-Like Testing and Fault Diagnosis Techniques for Digital Microfluidic Biochips*, in *Proceedings of the 12th IEEE European Test Symposium*. 2007, IEEE Computer Society.
86. Fei, S., et al., *Ensuring the operational health of droplet-based microelectrofluidic biosensor systems*. 2005.

87. Xu, T. and K. Chakrabarty, *Design-for-Testability for Digital Microfluidic Biochips*, in *VLSI Test Symposium*, IEEE, Editor. 2009. p. 309-314.
88. Xu, T. and K. Chakrabarty, *Towards design-for-testability for digital microfluidics*, in *Design, Test, Integration & Packaging of MEMS/MOEMS*. 2009. p. 329-333.
89. Zhao, Y. and K. Chakrabarty, *On-Line Testing of Lab-on-Chip Using Reconfigurable Digital Microfluidic Compactors.* International Journal of Parallel Programming, 2009. **37**(4): p. 370-388.
90. Zhao, Y. and K. Chakrabarty, *On-Line Testing of Lab-on-Chip Using Digital Microfluidic Compactors*, in *14th IEEE International On-Line Testing Symposium*. 2008. p. 213-218.
91. Xu, T. and K. Chakrabarty, *Fault Modeling and Functional Test Methods for Digital Microfluidic Biochips.* IEEE Transactions on Biomedical Circuits and Systems, 2009. **3**(4): p. 241-253.
92. Mitra, D., et al., *Accelerated Functional Testing of Digital Microfluidic Biochips*, in *Asian Test Symposium*. 2008. p. 295-300.
93. Datta, S., et al., *Efficient parallel testing and diagnosis of digital microfluidic biochips.* Journal Emerging Technology Computer Systems, 2009. **5**(2): p. 1-17.
94. Daniel, D., et al., *Multiple fault diagnosis in digital microfluidic biochips.* J. Emerg. Technol. Comput. Syst., 2006. **2**(4): p. 262-276.
95. Zhang, X., F. van Proosdij, and H.G. Kerkhoff, *A droplet routing technique for fault-tolerant digital microfluidic devices*, in *14th International Mixed-Signals, Sensors, and Systems Test Workshop*, IEEE, Editor. 2008: Vancouver, BC.
96. Venkatasubramanian, V., et al., *A review of process fault detection and diagnosis: Part I: Quantitative model-based methods.* Computers & Chemical Engineering, 2003. **27**(3): p. 293-311.
97. Venkatasubramanian, V., R. Rengaswamy, and S.N. Kavuri, *A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies.* Computers & Chemical Engineering, 2003. **27**(3): p. 313-326.
98. Venkatasubramanian, V., et al., *A review of process fault detection and diagnosis: Part III: Process history based methods.* Computers & Chemical Engineering, 2003. **27**(3): p. 327-346.
99. Vesely, W.E., et al., *Fault Tree Handbook*, in *NUREG-0492*. 1981, U.S Nuclear Regulatory Commision.
100. Abu-Hakima, S. *Automating Knowledge Acquisition in Diagnosis*. in *Proc. fourth International Workshop on Principles of Diagnosis*. 1993. Aberstwyth, Wales.
101. Kleer, J.d. and B.C. Williams, *Diagnosing Multiple Faults.* Journal of Artificial Intelligence, 1987. **32**: p. 97 - 130.
102. Madden, M.G. and P.J. Nolan. *Monitoring and Diagnosis of Multiple Incipient Faults Using Fault Tree Induction*. in *Proc. of IEEE Control Theory and Applications*. 1999.
103. Madden, M.G. and P.J. Nolan. *Generation of Fault Trees from Simulated Incipient Fault Case Data*. in *Proc. Ninth International Conference on Applications of AI in Engineering*. 1994. Pennsylvania, USA.
104. Papadopoulos, Y., *Safety-Directed System Monitoring Using Safety Cases*, in *Department of Computer Science*. 2000, University of York: York.
105. Quinlan, J.R., *Induction of decision trees.* Machine Learning, 1986. **1**(1): p. 81-106.
106. Lee, C., R.L. Alena, and P. Robinson, *Migrating Fault Trees to Decision Trees for Real Time Fault Detection on International Space Station*, in *IEEE Aerospace Conference*. 2005. p. 1 - 6.
107. Assaf, T. and J.B. Dugan. *Diagnostic Expert Systems from Dynamic Fault Tress*. in *IEEE Reliability and Maintainability*. 2004: IEEE.
108. Pattipati, K.R. and M.G. Alexandridis, *Application of Heuristic Search and Information Theory to Sequential Fault Diagnosis.* IEEE Transactions on Systems, Man, and Cybernetics, 1990. **20**(4): p. 872 - 887.
109. Assaf, T. and J.B. Dugan, *Diagnosis based on reliability analysis using monitors and sensors.* Reliability Engineering & System Safety, 2008. **93**(4): p. 509-521.

110. Reay, K.A. and J.D. Andrews, *A fault tree analysis strategy using binary decision diagrams.* Reliability Engineering & System Safety, 2002. **78**(1): p. 45-56.
111. Bobbio, A., et al., *Improving the analysis of dependable systems by mapping fault trees into Bayesian networks.* Reliability Engineering & System Safety, 2001. **71**(3): p. 249-260.
112. Limbourg, P., et al., *Fault tree analysis in an early design stage using the Dempster-Shafer theory of evidence.* Journal of Risk, Reliability and Societal Safety, 2007: p. 713 - 722.
113. Batzias, F.A. and C.C. Siontorou, *Investigating the Causes of Biosensor SNR Decrease by Means of Fault Tree Analysis.* IEEE Transactions on Instrumentation and Measurements, 2005. **54**(4): p. 1395 - 1406.
114. Myers, T.O. and I.M. Bell, *Fault modelling and test development for continuous flow microchemical sensor systems*, in *Mixed-Signals, Sensors, and Systems Test Workshop, 2008. IMS3TW 2008. IEEE 14th International*, IEEE, Editor. 2008, IEEE: Vancouver, BC. p. 1 - 6.
115. *Reliability of Systems, Equipment and Components: Guide to failure modes, effects and criticality analysis (FMEA and FMECA)*, in *BS 5760 : 5*, B.S. Institute, Editor. 1991.
116. Rosing, R., A.M. Richardson, and A.P. Dorey, *A fault simulation methodology for MEMS*, in *Proceedings of the conference on Design, automation and test in Europe*. 2000, ACM: Paris, France.
117. Olbrich, T., *Design -for-Test and Built-In Self-Test for Integrated Systems*. 1996, University of Lancaster.
118. jiang, T. and R.D. Blanton, *Inductive fault analysis of surface-micromachined MEMS.* Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2006. **25**(6): p. 1104 - 1116.
119. Mohammadi, K. and R. Asgary, *Pattern Recognition and Fault Detection in MEMS*, in *Computer Recognition Systems*. 2005. p. 877-884.
120. Asgary, R., K. Mohammadi, and M. Zwolinski, *Using neural networks as a fault detection mechanism in MEMS devices.* Microelectronics and Reliability, 2007. **47**(1): p. 142-149.
121. Peyman, S.A., A. Iles, and N. Pamme, *Mobile magnetic particles as solid-supports for rapid surface-based bioanalysis in continuous flow.* Lab on a Chip, 2009. **9**(21): p. 3110-3117.
122. Sun, Y., et al., *Design, simulation and experiment of electroosmotic microfluidic chip for cell sorting.* Sensors and Actuators A: Physical, 2007. **133**(2): p. 340-348.
123. Li, D., *Electrokinetics in Microfluidics*. Interface Science and Technology. Vol. 2. 2004: Elsevier Academic Press.
124. Erickson, D. and D. Li, *Integrated microfluidic devices.* Analytica Chimica Acta, 2004. **507**(1): p. 11-26.
125. Barsoukov, E. and J.R. Macdonald, *Imedance Spectroscopy: Theory, Experiment, and Applications*, ed. Wiley. Vol. 2nd Edition. 2005.
126. Hywel, M. and et al., *Single cell dielectric spectroscopy.* Journal of Physics D: Applied Physics, 2007. **40**(1): p. 61.
127. Wu, J. and H.-C. Chang. *Micro-Electrical Impedance Spectroscopy for Particle Detection*. in *Second International Conference on Microchannels and Minichannels (ICMM 2004)*. 2004. Rochester, New York: ASME.
128. Gawad, S., L. Schild, and P. Renaud, *Micromachined impedance spectroscopy flow cytometer for cell analysis and particle sizing.* Lab on a Chip, 2001. **1**(1): p. 76-82.
129. Liu, Q., et al., *Impedance studies of bio-behavior and chemosensitivity of cancer cells by micro-electrode arrays.* Biosensors and Bioelectronics, 2009. **24**(5): p. 1305-1310.
130. Compton, R.G., et al., *Hydrodynamic voltammetry with microelectrodes: channel microband electrodes; theory and experiment.* The Journal of Physical Chemistry, 1993. **97**(40): p. 10410-10415.
131. Collins, J. and A.P. Lee, *Microfluidic flow transducer based on the measurement of electrical admittance*. Lab on a Chip, 2004. **4**(1): p. 7-10.

132. Levich, V.G., *Physicochemical Hydrodynamics*. The Physical and Chemical Engineering Sciences, ed. P.-H. Inc. 1962.
133. Godino, N., J. del Campo, and F.X. Munoz, *Microband Electrodes in Microchannels: Edge diffusion and Wall effects*, in *COMSOL Users Conference 2007*. 2007: Grenoble.
134. Rees, N.V., et al., *Voltammetry under High Mass Transport Conditions. A High Speed Channel Electrode for the Study of Ultrafast Kinetics.* The Journal of Physical Chemistry, 1995. **99**(18): p. 7096-7101.
135. Singh, H., et al., *A Bayesian Approach to Reliability Prediction and Assessment of Component Based Systems*, in *Proceedings of the 12th International Symposium on Software Reliability Engineering*. 2001, IEEE Computer Society.
136. Scott, C. *The Neyman-Pearson Criterion.* 2004 [cited; Available from: http://cnx.org/content/m11548/1.2/.
137. Zjajo, A., J.P. Gyvez, and G. Gronthoud, *Structural Fault Modeling and Fault Detection Through Neyman-Pearson Decision Criteria for Analog Integrated Circuits.* J. Electron. Test., 2006. **22**(4-6): p. 399-409.
138. Khouas, A. and A. Derieux, *Fault Simulation for Analog Circuits Under Parameter Variations.* Journal of Electronic Testing, 2000. **16**(3): p. 269-278.
139. Khouas, A. and A. Derieux, *FDP: Fault Detection Probability Function for Analog Circuits*, in *IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS*. 2001. p. 17 - 20.
140. Abderrahman, A., B. Kaminska, and E. Cerny, *Optimization-based multifrequency test generation for analog circuits.* Journal of Electronic Testing, 1996. **9**(1): p. 59-73.
141. Abderrahman, A., et al., *New Analog Test Metrics Based on Probabilistic and Deterministic Combination Approaches*, in *Electronics, Circuits and Systems, 2007. ICECS 2007. 14th IEEE International Conference on*. 2007. p. 82-85.
142. Creveling, C.M., *Tolerance Design: a handbook for developing optimal specifications*. 1997.
143. Spinks, S.J., *Fault Simulation for Structural Testing of Analogue Integrated Circuits*, in *Engineering*. 1998, University of Hull.
144. *Do splitting rules really matter?*, in *Salford Systems*. 2003.
145. Becker, H., *Chips, Money, Industry, Education and the "Killer Application".* Lab on a Chip, 2009. **9**: p. 1659-1660.
146. Becker, H., *Mind the Gap!* Lab on a Chip, 2010. **10**: p. 271-273.