Flow Measurement of Pneumatically Conveyed Solids Using Intrusive Electrostatic Sensors

---

A Thesis Submitted to the
University of Kent
For the Degree of PhD
In Electronic Engineering

By
Jan Krabicka
Jan. 2010

Abstract

Particulate solids are commonly conveyed in industry by means of pneumatic pipelines. The particle flows often need to be controlled and maintained within certain bounds, but the development of instrumentation to monitor them remains a challenging area. A variety of techniques have been researched to measure various flow parameters. An overview of the existing technology is presented, along with advantages and limitations of each method.

A detailed investigation is conducted into the use of electrostatic sensors with intrusive electrodes to measure the velocity of pneumatic particle flows. Previous work has been reported on the use of non-intrusive ring electrodes, but few studies of intrusive electrodes have been undertaken to date. Modelling, based on the finite element method, is used to determine the characteristics of the charge induced by solid particle flows onto intrusive electrodes. These are then compared with the properties of non-intrusive circular ring electrode elements.

The effects of electrode intrusion depth are studied, and it is shown that whilst stability of the velocity measurements improves with intrusion depth, some types of flow are best measured using a particular intrusion that results in the most accurate average velocity reading. Electrode spacing, which must be close enough to allow a measurement to be taken but far enough to avoid unwanted interactive effects, is investigated, along with the effect of electrode cross sectional shape on sensor signals and the effect of common mode noise on cross correlation velocity measurement. This information is used in the development of a practical sensor system that uses embedded signal processing, which is then tested on laboratory and industrial flow rigs.

The results are used to characterise the features of intrusive electrostatic sensors and their response to different flow conditions. Most significantly, intrusive electrodes are shown to be sensitive to localised flow regimes. Finally, suggestions on aspects of electrostatic sensors that would benefit from further development are discussed.

Acknowledgements

The author wishes to express his gratitude to the people who have, in various ways, contributed to this project.

Prof. Yong Yan	my supervisor whose experience, insight and advice have been invaluable in givng direction and form to the work, as well as helping to overcome the difficulties encountered along the way.
Clive Birch	for his much appreciated technical advice and guidance in mechanical design.
Terry Rockhill	whose machining skills and patience with design modifications helped produce much of the sensor hardware involved in the project.
Dr. Robert Carter	whose professionalism, stimulating conversation, and cooperation during industrial testing made the collaborative aspects of the work a pleasure.
Dr. Jason Shao	for his collaboration in a joint paper resulting from this work, and for providing the data for some of the trials.
David Collins	whose comments, suggestions and proofreading of the final document greatly improved its readability.
DTI	the Department of Trade and Industry for its support through Project Number 10064, *An Integrated Sensor System for Combustion Plant Optimisation*.
RWE nPower	and its technical staff for making their combustion test facility available, and for their assistance and advice during testing.
University of Kent	for its generous support that made the project possible, and for providing a pleasant and enjoyable environment in which to work.

I would also like to thank my friends and family for their support and encouragement throughout the course of the project.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction to particle flow measurement

Industrial applications often require the conveyance of solid particulates under controlled flow conditions, including the transport of food products (rice, grain, etc.), pharmaceuticals, industrial chemicals and solid fuel for burners in the power and steel industries. Conveyance can be accomplished through several means, including:

(i) conveyor belts, where the belt mechanically transports the solids along its length,

(ii) dense phase plug flow, where the solids move in concentrated pulses separated by air gaps,

(iii) fluidised beds, where air is forced into the solids causing them to behave as single phase fluid and

(iv) dilute phase pneumatic flow, where the particles are suspended and carried along in a forced air stream.

The conveying parameters must be kept within certain bounds, and studies have been undertaken to define them [Mallick and Wypych, 2009, Thorn et al., 1982, Weber, 1981]. For example, if the velocity of the conveying air is too low in dilute phase flows then the particles may drop out of suspension, but if they are transported too quickly the conveying efficiency is reduced, the particles may be unintentionally broken up and there is an increased risk of damage to the pipeline through abrasive wear.

This research focuses on dilute phase pneumatic flows, i.e., where the volumetric solids concentration is below about 1%, or less than about 20:1 solids to air mass ratio [Rhodes, 2008].

1.1.1 Importance of particle flow measurement

There is a continuing need for the development of instrumentation for pneumatically conveyed particle flows, in order to keep the flow within permitted bounds discussed above. For example, in the coal fired power generation industry there is a drive for more sophisticated combustion systems that rely on more and more closely controlled combustion characteristics, which in turn rely on better developed and more sophisticated instrumentation [Kersch et al., 2001]. At a time when emissions control is becoming increasingly important (especially NOx emissions) and the demand for more power generation is rising, improvements in instrumentation technology are becoming more critical. Inappropriate conveying conditions should obviously be avoided, but at present many plant operators in the coal-fired power plant industry rely subjectively on the 'sound' of the material flowing in the pipeline to indicate a problem with conveying. A more scientific and quantitative approach is clearly desirable. Finally, multiple instruments measuring the same parameter may be needed in an industrial installation, as the particle flow may originate at a single source but then be distributed to a network of pipes. A coal-fired power plant, for example, may have a matrix

of 6 x 5 pipes ('tuyeres') to feed a single furnace. This demands that the instruments are as easy and convenient as possible to install, preferably at low cost.

1.1.2 Challenges in particle flow measurement

The challenges in pneumatic particle flow measurement stem largely from the fact that the flow is a two phase system, i.e., solid particles suspended in and carried along by a fluid. Instrumentation for single phase gas or liquid flow is well established, but when two phases are present the flow characteristics of the solid phase are generally not the same as those of the fluid, and they tend to be less predictable than that of the carrier. The following sections summarise the main challenges involved in effective particle flow measurement.

1.1.2.1 Solid phase velocity profile

Velocity profiles for laminar fluids are well established, but, in general, suspended solid particles conveyed by the flow do not follow the same profile. There is some evidence to suggest that in certain conditions particles follow a power law velocity profile, but in other cases the particles may have a random velocity field (chapter 3). This makes it difficult to measure the true average velocity of the flow if the instrumentation is not uniformly sensitive across the pipeline.

In general, particles travel with a lower velocity than the conveying fluid in a phenomenon known as 'slip'. The amount of slip depends on many factors, such as the number and severity of bends in the pipeline, particle size and particle shape. Sometimes the conveying fluid velocity is taken as the velocity of the particles, but the unknown amount of slip causes inaccuracy in the reading. It has been suggested [Mills, 2004] that the particle velocity in dilute phase flows is in the region of 80% of that of the conveying air.

1.1.2.2 Solids distribution

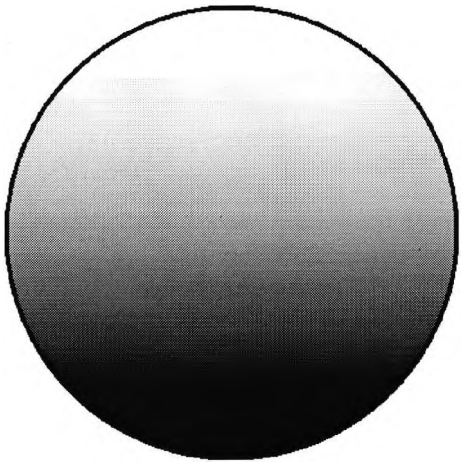
The distribution of the solid phase within the pipeline may not be uniform [Barry et al., 1997]. The particles in a horizontal pipeline may be more concentrated in the lower part of the pipeline, especially if the conveying velocity is low, causing some particles to drop out of suspension. In other cases the flow may be annular, i.e concentrated near the pipeline walls. Finally, there is a well known phenomenon called 'roping', where the flow is concentrated in narrow part of the pipeline cross section [Yilmaz and Levy, 1998, Yan, 1996]. Fig. 1.1 illustrates the main types of flow regime.

1.1.2.3 Solid phase physical parameters

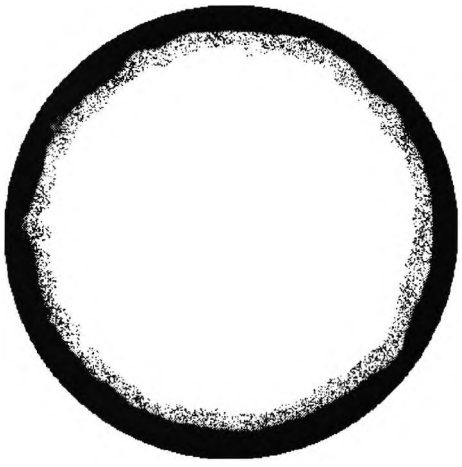
Difficulties are associated with the physical properties of the solid phase itself. Moisture and chemical composition may vary even within a particular material. For example, the chemical composition of coal can vary significantly in ash content between different grades, and moisture can vary from 1% to 30% [Mahajan, 1984]. The different sensing systems for measuring particle flows are sensitive to these parameters to varying degrees. Ideally, the instrumentation should depend only on the parameter of interest, but in practice the measurement is usually also affected by others, and this should be taken into consideration in the design of particle flow instruments.

1.1.2.4 Measurement reference

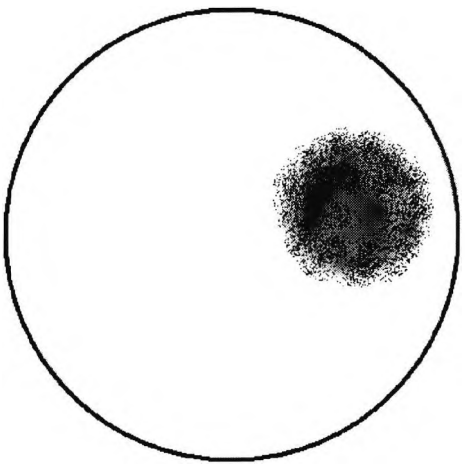
One of the most challenging difficulties in the design of particle flow instruments is the lack of a reliable reference against which the instrument under development can be assessed [Yan, 2005, DTI and Industry, 2004, Yan and Stewart, 2001]. In the measurement of particle size, a sample can be independently collected and measured offline, and instruments for average



(a) Stratified—solids distribution increases across pipeline



(b) Annular—flow/deposits near pipeline wall



(c) Roping—localised solids concentration

Figure 1.1: Nonuniform solids distributions

mass flow measurement can be compared with results from a load cell. For the measurement of particle velocity or volume concentration, however, an independent reference measurement is more difficult to obtain. Some optical methods may be appropriate, but they have difficulties resolving very small particle sizes and they often cover only a small proportion of the measurement volume. Mathematical modelling is a useful resource, as the model can be validated with experimental results in simplified and controlled conditions. Once confidence is gained in the model it can be extended to more complex scenarios, where experimental results are not possible.

1.2 Objectives of the research programme

A long term goal of the instrumentation group at the University of Kent is the development of online sensors to monitor a wide range of processes involved in the coal-fired power industry—from identification of the physical properties of the fuels, to the measurement of conveying parameters as the fuels are distributed to the burners and, finally, analysis of the flame characteristics and emissions produced during combustion. All stages influence the efficiency and emission levels of the combustion system. The broad strategy is to develop sensors for each of these processes separately and then to combine them in an integrated system for a comprehensive overview.

In this particular research, the objectives focus on the investigation of a measurement system based on the principle of electrostatic induction, i.e., a redistribution of surface charge on a conductor due to the presence of an external charge, to measure particle flow velocity. Two main types exist: sensors that use non-intrusive electrodes in the form of a circular ring, which forms an insulated part of the pipeline itself, and intrusive sensors, which use electrodes extending into the pipeline, into the particle flow.

Electrostatic sensors for velocity measurement, both intrusive and non-intrusive, have been

commercially available for many years, but there is much that is still unknown about their sensing mechanisms and optimal design. Both intrusive and non-intrusive designs have inherent strengths:

(i) there is no need for external signal source generation,

(ii) they can be made robust (subject to abrasive wear resistance) and inexpensive,

(iii) they are relatively unaffected by bulk solids characteristics such as moisture content for velocity measurement, and

(iv) they can be made to detect localised flow regimes.

Most research to date on electrostatic velocity instruments has concentrated on non-intrusive circular ring electrodes. Study of intrusive electrodes poses additional challenges, which are explored in this research programme. Table 1.1 summarises the main differences between intrusive and non-intrusive electrodes. Intrusive electrodes have several advantages in terms of measurement, as they can detect localised flow characteristics to help identify inhomogeneous flows, and they are relatively easy and inexpensive to install. The electrode diameter is usually very small relative to the pipeline so obstruction to the flow is minimal. Abrasive wear must be considered, but available options, including ceramic coating and the use of recently developed conductive ceramic materials, suggest that long lasting electrodes are possible.

The effective sensing zones of intrusive electrodes and the effect of their intrusion depth and cross sectional shape have not been reported, nor have models been presented to describe the signals acquired by sensor electrodes. The electric field of a non-intrusive circular ring electrode has an analytical solution (though complex), but no such solution exists for intrusive electrodes; finite element modelling is used in this research to approximate the induced signal. Also, the associated signal processing has, in general, been performed by external

equipment adding to system complexity and cost and making it less compact. In light of this, the main research objectives of the research are:

(i) to reveal the physical principles governing intrusive electrostatic sensors in order to develop a detailed characterisation of their operation

(ii) the design and implementation of a low cost embedded sensor system for the measurement of pulverised coal velocity inside pneumatic pipelines, suitable for installation on a commercial coal-fired power plant pipeline.

(iii) to evaluate system performance both in the laboratory and in industrial environments

Table 1.1: Comparison of intrusive and non-intrusive electrodes

	Circular Electrodes	Intrusive rod electrodes
Obstruction to flow	No	Yes
Susceptible to abrasive wear	No	Yes
Type of velocity information	Averaging effect	Localised information
Installation	Difficult	Easy
Cost	Expensive (spool piece required)	Low

The research programme comprises several principal areas of investigation:

A review of the main technical approaches for pneumatic particle flow measurement proposed to date is undertaken for comparison with electrostatic methods. Recent work on electrostatic sensors is summarised in order to highlight progress that has been made in recent years, as well as to identify areas which are still underdeveloped.

A theoretical analysis of the underlying principles involved regarding electrostatic velocity measurement instruments is undertaken, and a modelling method based on finite element modelling (FEM) is used in to investigate the sensing mechanism and the various parameters that affect the velocity measurement.

Practical considerations involved in the embedded design of an electrostatic velocity sensor are studied, including the design of hardware, electronics and signal processing algorithms. The results of practical tests and FEM modelling are then used to characterise sensor performance.

1.3 Thesis outline

The research is presented in six chapters:

Chapter 1 is an introduction to the topic of pneumatic particle flow measurement and aspects of it which would benefit from further development. The research objectives and contributions to the field are highlighted, along with the programme of research which was undertaken.

Chapter 2 introduces the main parameters of interest in pneumatic particle conveying and presents a literature review summarising the progress that has been made in the techniques to measure them. Instruments based on a variety of physical principles are introduced along with their relative strengths/weaknesses. A separate section is devoted to sensors using electrostatic methods, and the main principles underlying electrostatic measurement of various particle flow parameters are introduced.

Chapter 3 discusses the modelling of electrostatic sensors. A summary of the modelling methods proposed to date, and their limitations, is given. A more accurate model based on

finite element modelling (FEM) is presented for use with non-intrusive and intrusive electrostatic electrodes, and the results of the model are used to guide sensor design parameters including electrode separation distance, cross sectional shape, and spatial sensitivity. Other aspects of electrostatic measurement are also discussed, from a theoretical standpoint, including the effect of particle velocity profile on the measurement result and the effects of turbulent conveying air. The effect of external periodic noise on measurement is investigated, along with a proposed method of compensating for it. The results of practical tests using a belt rig conveyor to validate modelling are presented.

Chapter 4 is a detailed discussion of sensor design including the electronics, embedded hardware/software, user interface and mechanical design considerations. Different possible analogue circuits for signal acquisition are compared, and the embedded processing, based on a dsPIC® digital signal processing microcontroller, is presented, along with the signal processing algorithms. The mechanical designs for the laboratory and industrial test rigs is described.

Chapter 5 presents the practical results obtained from the laboratory and industrial test setups. Test results using various materials under various conveying conditions are compared and discussed. Suggested explanations for some interesting but unexpected results are discussed.

Chapter 6 draws together and summarises the research findings and their impact in the context of particle flow measurement. Further applications of modelling are suggested, in order to form a more complete characterisation of electrostatic velocity measurement instruments. Finally, areas and directions for future research are proposed, based on new questions arising from the research results.

Chapter 2

Review of measurement techniques for pneumatic particle flows

2.1 Introduction

The complex nature of the two phase gas/solids flow in pneumatic particle conveying is far from being completely understood. Advances in the understanding and measurement of particle flows, however, can improve efficiency and performance. For example, particle flows must be conveyed at some minimal velocity to avoid solids dropping out of suspension, but flow velocity which is too high is inefficient, leads to premature pipeline wear and may cause damage to the material being conveyed. Even incremental improvements in flow characteristics are important in industries such as power generation, where the flow parameters affect combustion quality, due to their large scale and environmental impact.

2.2 Gas/solids two phase flow parameters

A variety of instrumentation based on different physical principles has been investigated to measure the most important pneumatic particle flow parameters—velocity, solids volume concentration, mass flow rate and particle size distribution—and there are several publications available summarising the main strategies of measurement [Yan and Stewart, 2001, Williams et al., 1991, Boeck, 1989, Beck, 1981]. The strategies all attempt to relate some physical property of the particle flow to a particular flow parameter. The main methods that have been reported to date are outlined below. Most are concerned with measuring particle velocity, particle volume concentration, or mass flow rate.

In practice, only two of these need to be measured, since if any two are known then the third, in principle, can be derived using equation 2.1:

$$\dot{m} = v_{ave} C_V \rho A \tag{2.1}$$

where, v_{ave} is the average volumetric flow velocity, ρ is the density of the material, C_V is the volume concentration fraction, $\dot{m}$ is the mass flow rate and A is the cross sectional area of the pipeline. It should be noted that equation 2.1 is only valid when v_{ave}, C_V, ρ and A are all constant, which is often not the case, but it is a useful starting point from which to begin. The material density and cross sectional area are often taken to be constant, leaving only three variables. A fourth parameter—particle size distribution—must be measured independently, but it is an important parameter that can influence pipeline wear and flame quality [Hancke and Malan, 1998].

Most of the methods described in sections 2.3.1.1 to 2.3.6 require an active signal source, i.e., the sensors have two components: a transmitter to produce an externally generated signal

which is modulated in some way by the particle flow, and a receiver to pick up the modulated signal. By analysing the fluctuations in the receiver signal, information about the flow is inferred.

An externally powered signal source can be a disadvantage for applications in which potentially explosive materials are handled, such as the monitoring of pulverised coal flows. Strict safety regulations make plant operators apprehensive about introducing externally powered signals into the flow and can, rightly or wrongly, make them less willing to adopt instruments that use them. Electrostatic sensors, which are introduced in section 2.4 and which are the focus of this research, do not require an externally generated signal. Rather, they use the signal generated by the particle flow itself. A power source is still necessary to process the very weak signals produced, but this is done outside the pipeline, avoiding possible explosion hazards.

2.3 Sensing techniques

2.3.1 Capacitance

A well researched measurement method for solids/gas flows is to use the fluctuations in pipeline capacitance caused by the solid phase component of the flow. Most solids flows are of dielectric materials which cause fluctuations in the relative permeability across the pipeline. When two electrodes are positioned on the perimeter of the pipeline, with the solid phase flowing between, them a capacitor is formed, and information about the flow can be revealed by analysing fluctuations in capacitance. An advantage of this type of sensor is that the sensing zone is confined to the volume between the electrodes, with only a small amount of 'fringing' beyond the containing volume. Thus, a moving particle produces a rectangular pulse in the capacitance as it crosses the sensing volume [Hammer and Green, 1983]. This

makes modelling simpler than is the case for electrostatic sensors, which are sensitive to fields beyond the volume enclosed by the sensors. Capacitance sensors, however, do require an externally generated signal source for their operation.

2.3.1.1 Capacitance tomography

A two dimensional reconstruction of the solid material in a pipeline can be made using the technique of capacitance tomography. An arrangement of electrodes is set up on the perimeter of the pipeline forming a series of 'capacitors', with the air/particle mixture forming the dielectric between the capacitor plates. By applying tomographic reconstruction techniques a two dimensional cross sectional representation of the pipeline flow can be generated [Xie et al., 1989, Arko et al., 1999, Beck et al., 1987, Liu et al., 2005]. If several cross sectional capacitance 'images' are taken progressively downstream and techniques such as Linear Back Projection (LBP) together with numerical algorithms (such as Landweber's method [Landweber, 1951] are applied, it is possible to produce a three dimensional reconstruction of the material in the pipeline. However, the limited number of electrodes which can be practically installed, typically around twelve Liu et al. [2005], means that only coarse features can be resolved. This has made the application of capacitance tomography most successful in dense flows of material, such as fluidised beds. Also, since the relative permittivity of water ($\varepsilon_r \approx 81$) is much higher than that of commonly conveyed materials, solids concentration measurements are highly sensitive to moisture content Yan and Reed [1999].

2.3.1.2 Velocity measurement by capacitance

Velocity can be calculated from particle transit time between two axially spaced capacitance sensors using the technique of cross correlation (section 2.5). High solids concentrations,

however, are required to produce measurable changes in pipeline capacitance. Also, capacitance sensors have a non-uniform spatial sensitivity, and material near the center of the pipeline is largely invisible to them [Liu et al., 2005], making this type of sensor useful in a limited range of operating conditions.

2.3.2 Ultrasonic

Ultrasonic instruments have been used to measure particle velocity. The measurement principle is based on active acoustic signals which are transmitted upstream and downstream. The Doppler effect, i.e., the difference between upstream and downstream propagation velocities, is used to determine the pipeline flow velocity [Tallon and Davies, 2000]. For single phase flows of gas or liquid only, this method was reported to be in good agreement with computational fluid dynamics models [O'Sullivan and Wright, 2002], and commercial devices are available. Several difficulties arise when using ultrasonic measurement with two phase flows [Thorn et al., 1982]:

(i) the upstream/downstream propagation of signals is influenced by both particle and gas velocities, and the two cannot be easily distinguished

(ii) the ultrasonic velocity is significantly affected by temperature, which must be taken into consideration

(iii) it is difficult to obtain an efficient energy coupling to the gas flow from the ultrasonic transmitter

A different application of ultrasonic measurement is described by Hancke and Malan [1998], in which the vibrations of particles colliding with a resonant structure introduced into the pipeline are analysed to determine particle size. The principle is based on the observation that different particle size distributions cause related resonant vibration peaks in the structure

that can be detected by an accelerometer. Particle size information can then be determined using the power spectral densities of the vibrations and a neural network trained with particle flows of known size distribution.

2.3.3 Radiometric

Measurement systems using directed beams of radiation have been described and evaluated for velocity and mass flow rate measurement. They offer an advantage over 'soft field' measurement techniques, such as capacitance fluctuation, in that they provide better spatial resolution and are not as affected by inhomogeneous flow regimes [Yan, 1992]. Also they they can penetrate though material accumulated on the pipeline walls which would block the light used in optical systems [Mennell et al., 2000]. Radiation beams directed across the pipeline are attenuated by particle flow, and detectors on the opposite side of the pipeline detect the degree of attenuation. An absolute mass flow rate can be derived using separate measurements of mass concentration and a velocity measurement based on cross correlation.

Experimental results using a parallel beam X-ray source and an array of photodiode detectors have been reported by Barratt et al. [2000]. Tomographic reconstruction of the flow regime within the pipeline was demonstrated, and their mass flow measurements using a radiometric sensor show good agreement with load cell reference measurements. Velocity measurements, however, were less stable, and possible only in regions of dense flow (in common with capacitive sensors), suggesting that these types of sensors may be more applicable to mass flow and solids distribution monitoring. Precise alignment of signal source and detector across the pipeline is important, and this, along the safety concerns associated with radioactive sources, means that the systems can be difficult and expensive to install.

2.3.4 Laser Doppler velocimetry

Laser Doppler velocimetry (LDV) is a well established and studied technique [Albrecht et al., 2002, Thorn et al., 1982]. In this method, two intersecting laser beams are used causing an interference pattern, dependent on the beam intersection angle and the frequency of the laser source, to be set up at their intersection. Particles scatter the light as they pass through the alternating light and dark regions of the interference pattern. The scattered light is detected by a photodetector, and its frequency is proportional to the particle velocity [Woodhead et al., 1995].

This method can provide highly accurate measurements over a wide velocity range—measurements of particle velocities from 0.1 m/s to 100 m/s have been made with an accuracy of approximately 0.5% [Green and Thorn, 1998]— but it is regarded as too fragile and expensive to be of use in an industrial environment. The accuracy of this method depends on precise, accurate positioning of the optical beams, which can be difficult in industrial situations where access is restricted. Also, it measures velocity only at a particular point in the pipeline cross section, and a scanning mechanism would have to be built in to determine velocities of streamlines over the whole pipeline cross section.

2.3.5 Microwave

Microwave sensors are similar in principle to ultrasonic and laser techniques. To measure mass flow rate, they detect the degree to which an ultrasonic/ microwave beam is attenuated by the flow in the pipeline. Velocity can be calculated using a Doppler technique similar to the one for the laser method. The sensor can be configured as a system using a separate transmitter and receiver (bistatic) or by using a single antenna that functions as both transmitter and receiver (monostatic). Microwave methods have a larger measuring volume than

laser methods, but an acknowledged difficulty is the possibility of preferential or spurious reflections introducing errors into the measurement result [Thorn et al., 1982].

2.3.6 Optical Imaging

2.3.6.1 Particle image velocimetry (PIV)

Advances in low cost computing power and CCD imaging have facilitated developments in flow measurement through optical means. Most prominent among these is particle image velocimetry (PIV) [Kajitani and Dabiri, 2005], which has the potential to track particle motion and give two dimensional velocity vector flow fields, and is especially useful when tracking turbulent flows. Images at two cross sectional locations—one slightly downstream of the other—are compared to find the particles' transit time, using cross correlation, between the upstream and downstream locations. The most important advantage of this system is that a two dimensional image is resolved, and the particle velocity field over the entire cross section can be calculated.

This method been used to track tracer particles suspended in gas or liquid phase flows [Jonassen et al., 2006, Campbell et al., 2000]. In this case, it is the motion of the carrier fluid that is of interest, and the particles are chosen to have a density similar to that of the fluid in order to avoid gravitational forces and allow them to follow turbulent eddies. The resulting flow, although technically two phase, behaves as a single phase flow, and the particles streamlines are the same as those of the carrier. At the other extreme, very large scale two phase flows, such as flows of ice on a river, have been monitored using PIV in order to track the motion of the individual ice pieces. [Ettema et al., 1997].

A three dimensional version of PIV has also been tried to give a full 3-D vectorial representation of the velocity field Kurada et al. [1997]. Simultaneous stereo or orthogonal images

are required as inputs to the three dimensional cross correlation algorithms. The technique relies on performing cross correlation many times over small cubic volumes in the test region in order to build up an average measurement over many iterations. The test particles used are limited to a relatively large size (200-250μm), and although this method may be very useful for laboratory use, it is unlikely to be practical for industrial measurements due to the complexity of the setup and high computational demands of the system.

2.3.6.2 Optical tomography

Optical tomography has been proposed to measure solids concentration distribution, using multiple light sources and tomographic reconstruction algorithms. Whilst large objects can be resolved, with varying degrees of accuracy, using this method, reconstructing images of small particles is difficult. Zheng et al. [Zheng et al., 2006] have reported good results from optical tomography, using 4 to 15 light sources, for large objects, but smaller particles (in the range of 500-1000 μm) could not be resolved. As with PIV systems, the setup complexity and limitation to measurement of large particles means that it may have applications for laboratory use, but it is unlikely to be practical in industrial settings.

2.3.6.3 Direct imaging

Some particle flows, such as pneumatically conveyed pulverised coal, are composed of small particles (mean size of around 50μm, but down to particles of sub-micron size) at very low volume concentration in the pipeline. In this case, the particle motion can be tracked optically by direct imaging using a miniature camera with a high magnification connected to a PC [Carter, 2005, Carter and Yan, 2003]. The dilute composition of the flow means that individual particles can be resolved, with only an insignificant number of 'overlapping' particles

causing measurement distortion [Carter, 2005]. This method also allows online particle size to be measured [Carter et al., 2005, Hong and Tomita, 1995].

The pipeline cross section is illuminated by a sheet of laser light, and an image is recorded using CCD camera. The resulting image is then processed to find the particle size distribution and the proportion of the measurement volume composed of particles. A limitation of this technique is that the high optical magnification necessary to resolve very small particles often means that only a small proportion of the pipeline cross section can be seen, and a homogeneous particle distribution must usually be assumed when extrapolating the measurement to the entire pipeline cross section. Optical systems are also difficult to install and maintain in harsh industrial conditions, and in order to obtain a clear image of small, fast moving particles, illumination must be intense and of short duration. Finally, under high magnification particles travel through the field of vision very quickly, requiring even more illumination to allow exposure times fast enough to freeze the motion, or to at least capture a well defined 'blur'.

In general, whilst direct imaging optical systems have the potential for accurate and detailed flow analysis, there are several issues which must be resolved. Intense illumination is usually required, but the power of the laser light source must often be limited due to safety concerns, and the lenses—easily contaminated with particles—are notoriously difficult to keep clean. Mechanical cleaning of the lenses raises doubts about long term reliability, and schemes involving the use of purging air streams have tended to result in low pressure areas near the lenses that attract contaminants rather than repel them. These problems must be addressed before optical systems are suitable for widespread use in particle flow measurement instruments for industrial settings.

2.3.7 Thermal

Thermal flow meters have been proposed to measure mass flow directly, using the principle of heat transfer. A section of pipeline is heated and maintained at a known, constant temperature. As the solids flow passes through this section, it absorbs heat according to the laws of thermodynamics. Downstream of the heated section, the rate of cooling, dependent on the solids' specific heat capacity, is measured by infra red sensors, and the mass flow rate is inferred. The heat transfer coefficient is critical in this type of measurement and has been studied for circulation fluidised beds [Wang et al., 1997].

Zheng et al. [2008, 2007] have performed detailed modelling of the heat transfer mechanism for use in non-invasive mass flow measurement, and validated the results with experiments using a single particle carried downstream on a string at a known speed. The literature, however, suggests that this form of measurement is most suitable for dense phase 'plug' flows or circulation fluidised beds, where sufficient mass is present to provide enough heat capacity to make a measurement. To date there have been no reports of flow meter tests for dilute phase particle flows.

2.4 Electrostatic instruments

2.4.1 Principle of operation

2.4.1.1 Electrostatic signal acquisition

Electrostatic sensors have the advantage of sensing a signal that is generated as result of the conveying process itself. The solid particles develop a charge through interaction with the air, pipeline walls and collisions with other particles, and this in itself has been an area

of research [Matsusaka and Masuda, 2006, Zhu et al., 2004, Armour-Chelu and Woodhead, 2002, Matsusaka et al., 2002, Gajewski, 1997]. The charges residing on the particles flowing downstream can be regarded as an electrical current, with a magnitude dependent on the amount of charge on the particles themselves, and the velocity with which they are moving downstream. This current is small—the charge densities in pneumatically conveyed solids flows have been found to range from about 10^{-7} C/kg to 10^{-3} C/kg Yan et al. [1995a]—and requires a large amount of analogue amplification before the signals can be detected, recorded and analysed.

One way to do this is to connect the sensor element to a high impedance voltage amplifier, which is often opamp based. The charge induced on the sensor changes its electric potential, which is then magnified by the voltage amplifier to a level that can be conveniently recorded. [Williams et al., 1991, Xie et al., 1989, Gajewski, 2000]. Another method is to connect the sensor directly to the virtual earth of an inverting opamp circuit. Charge induced on the sensor electrodes allows a displacement current to flow to the circuit's virtual earth [Armour-Chelu et al., 1998]. In other words, the first approach is voltage amplification, and the second is a current to voltage conversion. Detailed information on the particular circuits used and their relative merits are not often found in the literature, but an in-depth analysis will be presented in chapter 4.

2.4.1.2 Velocity measurement

A) Single electrode

Electrostatic velocity measurement has been attempted using two approaches. The first is to use a single electrode exposed to the particle flow [Xu et al., 2007]. The electrode acts as a spatial low pass filter, with the frequency spectrum of the induced signal directly related to the flow velocity.

However, the sensor response is determined by several variables [Yan, 1992], and many of these must be evaluated experimentally for a particular sensor configuration and applied as a scaling factor. These factors include:

(i) a geometric shape factor that depends on the physical dimensions of the electrode

(ii) a particle velocity distribution factor that takes into account the velocity profile of the flow

(iii) a particle distribution factor that takes into account non-uniform particle distributions within the pipeline

(iv) other factors such as particle size, shape and permittivity

All but the geometric shape factor must be determined experimentally for particular types of flow. Despite this, Xu et al. [2007] have found that repeatability within +/-5% can be achieved for gravity driven flows of between 2 m/s and 6 m/s for particle concentrations between 0.5% and 6%. However, it is not clear how much the scaling factors are affected by the higher velocities typical of pneumatically conveyed flows.

B) Electrode pair

A more common and established method of electrostatic velocity measurement is performed using two axially spaced electrodes and applying cross correlation analysis (section 2.5). This has an advantage over single electrode frequency spectrum analysis in that the result is less influenced by the spatial filtering effect of the electrodes—only the time delay between upstream and downstream signals is significant. This technique has been investigated for several decades [Beck et al., 1969, Gajewski, 1983, Yan et al., 1995b, Gajewski, 1996b, Ma and Yan, 2000, Zhang and Coulthard, 2005, Peng et al., 2008], and the focus previous research has been to define sensor response in terms of bandwidth, spatial sensitivity and optimum electrode configuration. However, electrostatic fields due to charges in the presence

of conductors and dielectric materials are complex. Most previous studies have presented mathematical models of varying complexity, but it will be shown (chapter 3) that a more accurate model can be developed using the finite element method .

Previous work has concentrated almost exclusively on non-intrusive design (section 3.4.1). Early work [Beck et al., 1969] describes the use of dedicated analogue or digital correlators to calculate the velocity result, with later work done using general purpose computers. There is little or no reported research that addresses intrusive electrode design, or describes the online, embedded system implementation of the techniques.

2.4.1.3 Particle size distribution

Electrostatic sensors have been used to estimate particle size in pipelines. One methodology involves positioning the sensor after a bend. The principle makes use of the fact that heavier particles are carried further over to the far side of the pipeline after a bend, due to their increased inertia, where they induce a stronger signal than on the near side [TR-Tech, 2009]. However, detailed information on repeatability and accuracy of this commercial product have not been reported in the public domain.

Another reported method makes use of an electrode grid (Fig. 2.1) that measures the magnitude of passing charge. In principle, larger particle sizes are assumed to carry more charge than smaller ones, and provide a means to measure the median particle size. An experimental trial has been conducted using a square mesh electrode made of 0.5 mm copper wire, with 8 mm spacing [Zhang and Yan, 2003]. The average power of the resulting signal is compared to a reference power using particles with a known sizing to calibrate the output for a particular material and velocity. Results have been reported of mass median particle size measurements with a relative error of +/- 15%, using materials typical of industrial pneumatic flows.

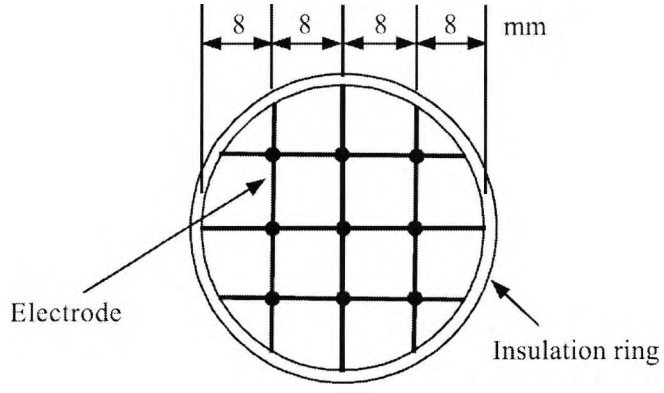


Figure 2.1: Electrode mesh for electrostatic particle sizing (reproduced from Zhang and Yan [2003])

Acknowledged difficulties with this method include the fragility of the electrode mesh, which must be fine enough not to significantly disturb the flow and yet robust enough to withstand abrasive particle flows. Also, the measurement result is dependent on factors such as mass flow rate which must be measured separately, although the passive nature and relative simplicity of the principle is an advantage for industrial applications.

2.4.1.4 Mass flow rate

Attempts have been made to measure mass rate using electrostatic signals. Gajewski [1996b, 1999], for example, has reported results using the rectified mean value of an electrostatic signal to infer mass flow rate. The principle is that with increased mass flow, the total charge present in the pipeline is increased along with the magnitude of the signals induced on an electrostatic sensor. There are, however, many unresolved difficulties in using the sensors in this way including the large number of factors that influence the total charge carried by any particular material. These factors are generally unknown, and an absolute mass flow measurement instrument must be calibrated for a particular set of flow parameters including material type, flow velocity, pipeline solids loading, humidity and temperature. Keeping all

parameters other than solids loading constant, Gajewski [1996b] was able to demonstrate that a calibration curve could be constructed over a limited range of mass flow rates (53 g/s to 62 g/s), using a load cell as a reference.

Another option is to use the magnitude measurement purely as a relative measure, in order to balance the flow between different sections of pipeline. This is generally regarded as more feasible, but the unpredictability of particle charge still creates difficulties. There is evidence that, in certain operating conditions, a 'saturation level' of charge is reached [Armour-Chelu and Woodhead, 2002], due to the increased possibilities of charge transfer to the grounded pipeline, and that beyond this level even relative measurements cannot be made. This lack of a reliable relationship between mass flow and charge carried has made the viability of even relative mass flow rate measurements uncertain.

A more successful method of measuring mass flow rate has been to combine the measurement of solids concentration by other means (e.g., optical, Fig. 2.2) with electrostatic velocity measurement. Results with a repeatability of +/- 6% have been reported using this method, at concentrations and velocities typical of pneumatic conveying systems [Carter et al., 2005].

2.4.2 Intrusive vs. non-intrusive electrostatic design

Most research has been concerned with the study of non-intrusive electrodes that are essentially an isolated section of the pipeline wall itself. The design parameters for the electrode are essentially to decide on the axial width of the electrodes, the electrode spacing and to determine the signal characteristics as a function of this width. The symmetry inherent in a non-intrusive circular design simplifies sensor modelling, and several mathematical models have been presented (chapter 3).

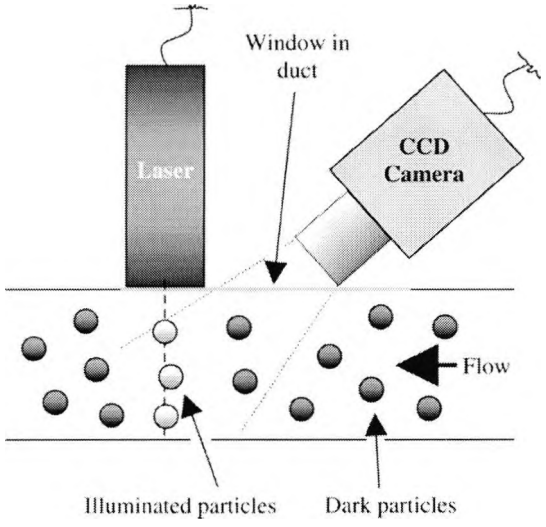


Figure 2.2: Optical measurement of solids concentration (reproduced from Carter et al. [2005])

Little research has been reported on the design and response of intrusive electrodes. There have, however, been commercial attempts to develop electrostatic sensors using intrusive electrodes. For example, ABB, PCME, TR-Tech and ESKOM (DTI, 2004) have developed intrusive rod sensors to monitor exhaust stack emissions in terms of both velocity and mass flow (Fig. 2.3). The dilute nature of the flow and relatively low abrasion make the use of intrusive probes feasible in this case. TR-Tech [TR-Tech, 2009] has also marketed intrusive instruments, using tungsten carbide intrusive probes to monitor PF flow. Very little scientific work, however, has been published on the characteristics and optimisation of these types of probes, and one of the primary aims of the present research is to extend the available information on this subject.

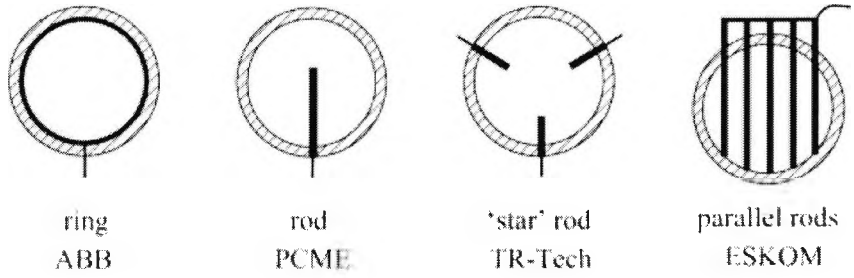


Figure 2.3: Intrusive and non-intrusive electrode configurations (reproduced from DTI [2004])

2.4.3 Influence of pipeline shape

The passive nature of electrostatic sensors means that the signal is generated in the pipeline by the particle flow itself. Consequently, the physical parameters of the pipeline must be taken into account. By far the most common pipeline cross sectional shape is circular. Some industrial processes, however, such as the conveying of fluidised beds [Liu et al., 2005], make use of pipelines with a square cross section, and the differences in sensor response compared to a circular cross section should be known.

An analysis has recently been reported on the response of non-intrusive electrodes in square pipelines by Peng et al. [2008]. In a method analogous to that which will be described in the present research (chapter 3), modelling results were validated by using a test rig with a single charged bead. The principle of superposition can then be to extend the results to flows composed of an arbitrary number of particles. Their main conclusions state that the sensor characteristics are broadly similar to those of sensors installed on circular pipelines in terms of bandwidth and sensitivity, although sensitivity in the areas close to the pipeline corners varies significantly from the circular case, as would be expected. They also found that the sharp corners of a square pipeline make modelling more difficult.

2.5 Cross correlation velocity measurement

Cross correlation is a well established statistical technique that can be applied to find the delay between similar, but time displaced, signals. It can be used to determine particle velocity in instruments that have 'upstream' and 'downstream' sensors, which are separated by a known distance. These instruments are generally referred to as 'cross correlation flow meters'. If the sensors are spaced closely enough so that the configuration of particles in the pipeline is not changed significantly, then downstream signal is similar to the upstream signal. The downstream signal can then be thought of as a 'corrupted' version of the upstream signal; the corruption arises from the fact that the particle configuration does change a certain amount due to random particle motion, and also from the fact the electrodes and their respective electronic conditioning circuits cannot be made perfectly identical. The downstream signal will also be delayed by the amount of time the flow takes to travel between the two sensors. In the past, some researchers have used storage oscilloscopes to measure this delay manually [Kacprzyk and Gajewski, 2001], but the advance of digital electronics means that today the cross correlation can be performed digitally. The peak value of the cross correlation function can be used to find the time delay between the two signals [Ifeachor and Jervis, 2002]. The velocity, s, can then be calculated by:

$$v = \frac{L_e}{\lambda_d} \tag{2.2}$$

where L_e is the sensor spacing, and λ_d is the peak cross correlation time delay. A further discussion of the use of this technique in various types of instruments can be found in Beck and Plaskowski [1987] and Beck [1981]. Also, although it is a well established technique for offline analysis, the advance of fast microprocessors and data acquisition systems has made online measurements feasible. A further discussion of cross correlation techniques, including dedicated correlation hardware realisations, is given by Boeck [1989].

2.6 Summary

A variety of techniques have been investigated in order to measure the various particle flow parameters of interest as discussed in section 2.2. Although each method has a sound theoretical basis for the measurement of particle flows that have been idealised in some respect, the challenges of real world operating conditions have prevented a definitive solution for particle flow measurement in industrial environments from being found. The methods described in the previous sections suffer from excessive sensitivity to flow characteristics that cannot be accurately monitored or controlled, such as moisture content or solids distribution, or are unsuitable for harsh operating environments.

The review has indicated that there has been very limited research into intrusive electrostatic sensors. The aim of this research is to characterise the response of these sensors and to develop a robust, cost effective and easily installed sensor that gives accurate and reliable measurements for pneumatic particle flows that may follow variable velocity profiles and solids distributions.

Table 2.1 is a summary of the main parameters of interest for pneumatically conveyed particulate flows and the instrumentation principles that have been researched in order to measure them. Note that these all refer to methods of online measurement, i.e., measurements that are performed in close to real time (dependent only on the signal processing speed available), as opposed to redirecting the flow and taking samples for later analysis.

Table 2.1: Flow measurement techniques

Particle Parameter	**Sensor Technique**	**Signal/Image Processing Technique**
Volumetric concentration	Capacitance Optical	Signal magnitude Signal attenuation
Mass concentration	Radiometric	Signal attenuation
Velocity	Capacitance Electrostatic Ultrasonic Laser/Optical Microwave	Cross correlation Cross correlation Doppler shift PIV, LDV Doppler shift
Solids distribution	Capacitance Electrostatic Optical	Tomographic reconstruction Signal magnitude Direct imaging
Particle size distribution	Electrostatic Optical	Signal magnitude Direct imaging
Mass flow rate	Electrostatic Thermal	Signal magnitude Heat transfer

Chapter 3

Modelling of electrostatic sensors

3.1 Introduction

Evaluating the performance pneumatically conveyed particle velocity sensors is a difficult task. Perhaps the most challenging aspect is the fact that the trajectory of every particle would have to be known in order to provide an accurate reference with which the sensor could be compared. This is not generally possible in either industrial or laboratory settings, and the role of simulation and modelling becomes more important in estimating the signal induced onto the sensor electrodes. In order to gain confidence in the results of modelling and simulation, however, some connection with real world performance is essential.

This chapter summarises electrostatic sensor models that have been proposed to date, along with their inherent strengths/weaknesses, and introduces a new method of studying sensor response based on finite element modelling. Also, the possible effects of the conveying air on the particle flow are discussed. The results are compared with those of laboratory test setups in order to provide confidence that the models accurately reflect sensor response.

3.2 Modelling the electrical field inside a pipeline

Several mathematical models have been presented in order to study the signals induced on electrostatic sensors by pneumatically conveyed particle flows. Usually grounded, metal, cylindrical pipelines are used for pneumatic conveying (although some process plants in China use pipelines with a square cross section, e.g., fluidised beds), so the pipeline is often modelled as a grounded, infinitely conducting cylinder. The models, however, are highly sensitive to the physical geometry of the pipeline/sensor electrode configuration.

The electrode itself is most commonly modelled in the non-intrusive electrode configuration, where the electrode is an insulated, ring shaped section of the pipeline itself. In this case, the only relevant electrode dimensions are its diameter, which is the same as that of the pipeline, and its axial length. In general, particles are modelled as point charges, and the aim is to determine the charge induced on the sensor by the presence of the point charge. This information is then used to estimate sensor response in terms of parameters such as its sensitivity and the frequency response of the induced signals.

One approximation considers the electric field, E, that is produced by a point charge in a dielectric or free space medium [Yan, 1992], given by

$$E = \frac{q}{4\pi\varepsilon_0 r^2} \tag{3.1}$$

where r is the distance from the charge, q is the magnitude of the charge in Coulombs, and ε_0 is the permittivity of free space. Gajewski uses this to investigate sensor bandwidth [2006, 1996a], but makes the further simplifying assumption that the field only affects the sensor when the charge is within the region of the pipeline formed by the ring electrode, i.e., it is

assumed that the sensing zone is only within the 'slice' of pipeline bounded by the sensor geometry. The −3dB cutoff frequency, F_{3dB}, is given by:

$$F_{3dB} = \frac{v_{ave}}{W} \tag{3.2}$$

where v_{ave} is the average particle velocity, and W is the ring probe axial width.

Yan et al. also derive the approximate the signal induced on a ring probe [Yan et al., 1995b], but the fact that the sensing zone of the sensor extends beyond the physical pipeline volume bounded by the sensing ring is taken into consideration. The resulting bandwidth, F_c, defined as the first zero crossing of the frequency domain spectrum magnitude response, is given as:

$$F_c = K_b \frac{v_{ave}}{W} \tag{3.3}$$

where K_b is a scaling factor. Equation 3.3 is then used to investigate optimum sensor dimensions in terms of the ratio of sensor ring axial width to pipeline diameter. Equations 3.2 and 3.3 differ only by the constant K_b, which is a meter constant included to account for factors such as spatial filtering effects, non-uniform distribution of solids, velocity profile, the particular definition of cutoff frequency and the fact that the sensing zone extends beyond the slice of pipeline enclosed surrounded by the sensor.

Other approaches to electrostatic modelling consider the effect of the grounded pipeline itself. The presence of a grounded conductor affects the charge induced on an electrostatic sensor, and is described by one of Maxwell's equations [Duffin, 1990]:

$$\nabla \cdot D = \rho_q \tag{3.4}$$

or equivalently,

$$\oint_S D \cdot dS = \int_V \rho_q dV \tag{3.5}$$

where ∇ is the divergence operator, D is the electric displacement field, ρ_p is the charge density, and V is the enclosed volume. This leads to the well known Poisson's equation (3.6), whose solution gives the electric potential, Φ, due to a known charge distribution and from which the induced charge can be found by equation 3.7.

$$\nabla^2 \Phi = -\rho_q / \varepsilon_0 \tag{3.6}$$

$$D = -\nabla \Phi \varepsilon_0 \tag{3.7}$$

An analytical solution, however, is difficult or impossible to determine, except for cases where a high degree of symmetry is present. A number of simplified approaches have been tried. One method involves modelling the point charge/pipeline system as a point charge adjacent to an infinite conducting plane. This special case has a relatively straightforward electrostatic field solution, made possible by a technique known as the 'method of images'. Armour-Chelu et al. [1998] use this method to model the induced field in the investigation of the charging trends of pneumatically conveyed particles. An extension of this approach involves using four such planes to model a pipeline with a square cross section [Murnane et al., 1996].

Interestingly, an analytical solution has in fact been derived and published for the non-intrusive circular ring electrode, for use in the seemingly unrelated field of atmospheric research [Weinheimer, 1988]. The charge induced on a circular ring electrode due to a point charge inside the pipeline is given by:

$$\left|\frac{q'}{q}\right| = \sum_{n=1}^{\infty} \frac{2}{x_n} f\left(\frac{c_r}{a_r}, \frac{z_0}{c_r}, x_n\right) \frac{J_0\left(x_n \left[\frac{r_0}{a_r}\right]\right)}{J_1(x_n)} \tag{3.8}$$

where

$$f = 1 - exp\left(-x_n \frac{c_r}{a_r}\right) cosh\left(x_n \frac{c_r}{a_r} \frac{|z_0|}{c_r}\right) \text{ for } \frac{|z_0|}{c_r} \leq 1$$

$$f = exp\left(-x_n \frac{c_r}{a_r} \frac{|z_0|}{c_r}\right) sinh\left(x_n \frac{c_r}{a_r}\right) \text{ for } \frac{|z_0|}{c_r} > 1$$

q is the value of the point charge,

q' is the charge induced on the circular electrode

a_r is the radius of the cylinder

c_r is one half of the length of the cylindrical electrode

r_0 and z_0 are the radial and axial coordinates of the point charge, in a cylindrical coordinate system with its origin at the centre of the cylindrical electrode segment

J_s is the Bessel function of order s

x_n is the n^{th} zero of J_0

The derivation of equation 3.8 makes use of the high degree of symmetry of the circular electrode configuration. For sensor configurations with less symmetry, such as the intrusive rod sensor, there is no known analytical solution or even rough approximation available, and another way of determining charge induction must be found.

The availability of various software packages and fast processors has made it possible to study the electrostatic fields of more complex sensor arrangements using the technique of finite element modelling (FEM). Zhang [2005] has used this method to investigate the electrostatic properties of non-intrusive ring electrodes, where it is used to find the theoretical spatial sensitivity and frequency response. FEM will be used in the following sections in order to study the characteristics of intrusive electrodes.

3.3 Finite element modelling

Finite element modelling has been established as a good method for situations where accurate results are required only for localised parts of the domain [Krabicka and Yan, 2009, Zhang and Coulthard, 2005]. For application to electrostatic sensors, the particular domain of interest is the surface of the electrode, on which charge is induced by passing charged particles. The charged particles, pipeline boundary and the air inside the pipeline must be included as part of the model, along with appropriate boundary conditions. The FEM solves for the electric field inside the pipeline, and the charge induced onto the electrode can subsequently be found.

The derivation and solution of the FEM equations can be developed from first principles [Silvester, 1996], i.e., an approximate solution of Poisson's equation for discrete locations on the system geometry can be formulated as a system of linear differential equations and solved. There are, however, a variety of software packages available (ANSYS, ALGOR, COMSOL etc.) with user interfaces that allow graphical representation of the system, definition of material properties and definition of boundary conditions, that can be solved without the burden of low-level formulation of the relevant system of equations.

3.4 Non-intrusive electrostatic ring electrode

3.4.1 FEM model

A finite element model of the case of a non-intrusive ring electrode was made and compared with the free space approximation and the analytical solution described in section 3.2, in order to validate the use of FEM in cases where no analytical solution is available. A commercial software package, COMSOL® 3.4, was used in the present study as a tool with

which to investigate the electric fields present in a particular electrode configuration. COMSOL was chosen for its graphical user interface and programmability in Matlab®, which enables loop structures to be used to perform multiple simulation iterations automatically.

A pipeline section, 40 mm in diameter—similar to the pipeline used in the practical trials described in section 5.2—and 100 mm in length, fitted with a flush mounted, 1.6 mm diameter circular electrode, was modelled. A free mesh of approximately 15600 tetrahedral quadratic Lagrange elements was automatically generated in COMSOL (Fig. 3.1). Boundary conditions were set to ground for the pipeline and the electrode and to zero charge for the free ends of the pipeline. The air in the pipeline was assumed to have a dielectric constant of 1. A single charged particle, modelled as a point charge, was then added upstream of the sensing electrode, and COMSOL was used to determine the total charge induced by the particle onto the electrode. The point charge was then moved downstream incrementally, in a streamline parallel to the pipeline wall. The charge induced on the electrode was determined at each step along the way, in a manner similar to an animation procedure. The presence of multiple particles in the pipeline can be modelled, in principle, by the superposition of charge streamlines from the FEM results. This important property follows from the fact that force between two charged particles is not modified by the presence of other charged particles [Shey, 1973]. Another way of looking at this approach (i.e., modelling a single particle moving along the streamline) is from a system modelling point of view—each passing particle can be regarded as an impulse input to the sensor system, and the output due to a single particle can be regarded as the impulse response of the system [Yan et al., 1995b, Zhang and Coulthard, 2005, Peng et al., 2008], which can then be used to determine the output resulting from an arbitrary configuration of particles.

Fig. 3.2 shows the induced charge curves for a modelled point charge of 1 μC moving downstream at distances of 0 mm, 10 mm and 16 mm from the pipeline centre. As expected, the induced charge has a higher magnitude and sharper peak when the point charge is closer

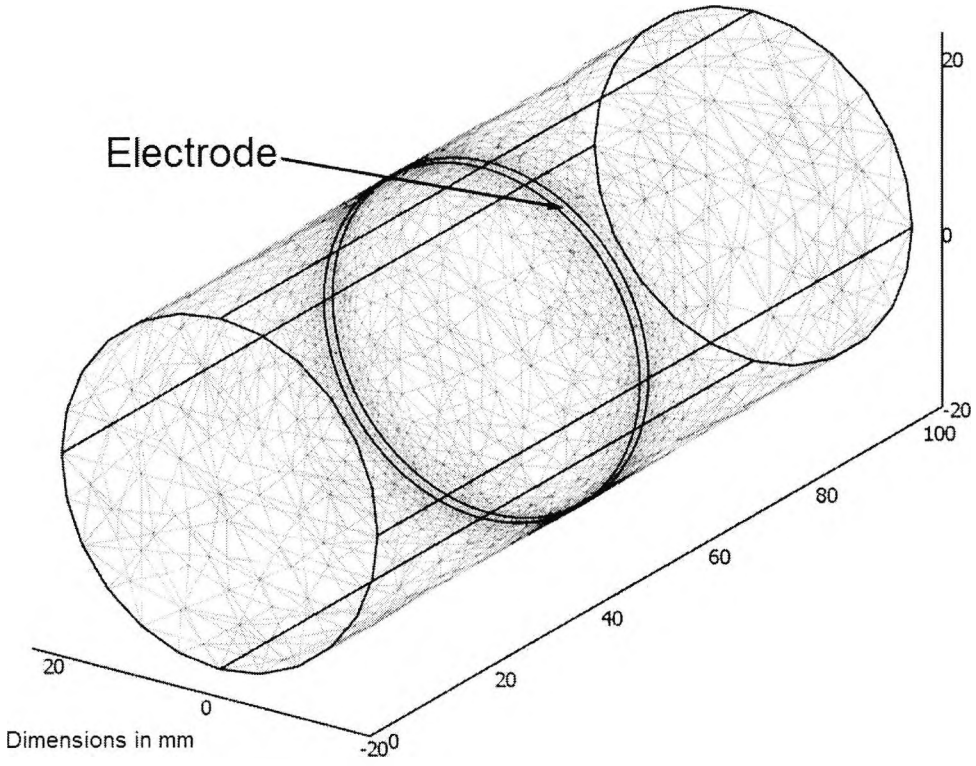


Figure 3.1: FEM model of a circular electrode

to the sensor electrode, i.e., near the pipeline wall. The FEM model results are found to be very close to the theoretical results calculated from the analytical solution (equation 3.8). Fig. 3.3 shows the relative error of the FEM results, with reference to the theoretical solution. The close agreement between the two sets of results (relative error < 2% in most cases) offers confidence that the FEM results can be used in the case of an intrusive rod electrode, in the absence of an analytical solution. It can be seen that the FEM results are more accurate in the centre of the pipe section than near to the pipe wall, i.e., the numerical errors increase towards the boundary of the model.

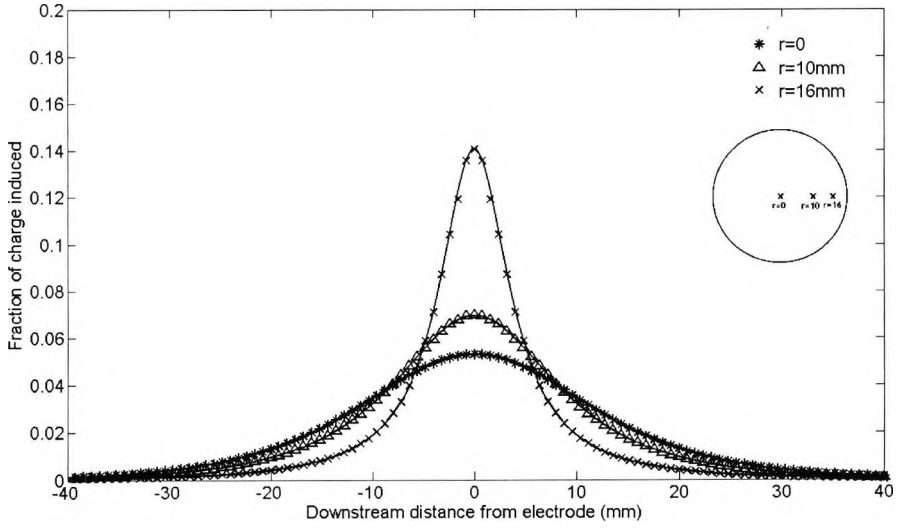


Figure 3.2: Induced charge on a circular electrode—3 sample streamlines

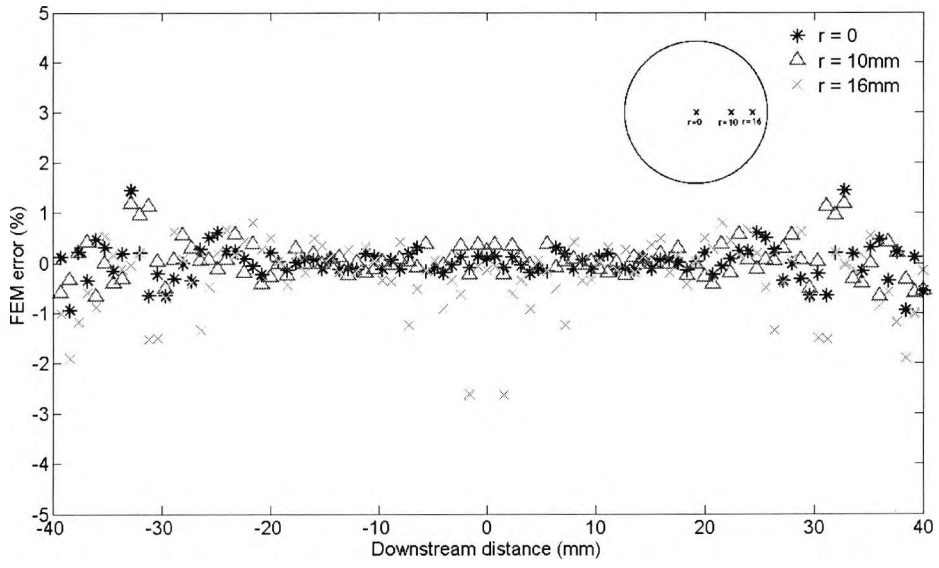


Figure 3.3: Relative error of FEM results for a circular electrode

3.4.2 Comparison of FEM with analytical solution and free space approximation

A comparison can be made between the results given by finite element modelling and the corresponding analytical and free space approximation solutions. As stated in section 3.2, an analytical solution is available for the special case of the non-intrusive ring electrode, and the model presented by [Yan et al., 1995b] gives an approximate solution which is easier to use but disregards the effect of the grounded pipeline. A comparison of the spatial sensitivity of the electrode using the analytical, approximate and FEM models is shown in Fig. (3.4). The spatial sensitivity is taken to be the charge fraction induced onto the electrode by a point charge when it is in the pipeline cross section through the center of the electrode, where maximum charge is induced. Four electrode widths are considered, expressed as width to pipeline diameter ratios (W/D) of 1/25, 1/2, 1 and 2. The radial location is normalised to

the pipeline diameter and ranges from 0, which corresponds to the radial center, to 0.5, corresponding to the pipeline wall. The errors of the free space approximation, taking the analytical solution as the reference, are -27.0%, -28.4%, -21.2%, and -10.1% for W/D ratios in increasing order. The respective errors for the FEM solution are 4.3%, 0.37%, 0.25% and 0.22%. Clearly, the FEM solution gives a significantly more accurate solution to the induced charge than the free space approximation.

3.5 Intrusive electrostatic electrode

3.5.1 FEM model

For intrusive electrodes, with no analytical solution available, an experimental method was used to further validate the use of FEM. A FEM model of the intrusive rod electrode, constructed using COMSOL is shown in Fig. 3.5. The model consists of a rod electrode of 1.6 mm diameter embedded in a circular insulator of 10 mm diameter and protruding 5 mm into the same pipeline as described in section 3.4.1. As with the circular electrode (section 3.4.1), boundary conditions were set to ground for the sensor and pipeline and to zero charge for the insulator and pipeline ends. A total of eight point charge streamline locations were defined, as illustrated in Fig. 3.6. The FEM data points in Fig. 3.7 show the charge fraction induced onto the electrode as the point charge moves away from the electrode plane in the four streamlines, A, B, C and D, located 2 mm, 6 mm, 10 mm and 16 mm laterally away from the electrode. The result shows that charged particles located beyond streamline D will have an insignificant effect on the electrode.

From a sensor modelling point of view, it is desirable to fit a curve to the FEM data points so that an equivalent analytical expression can be found to represent the charge induced on an electrode by a passing particle. There are a limited number of mathematical functions

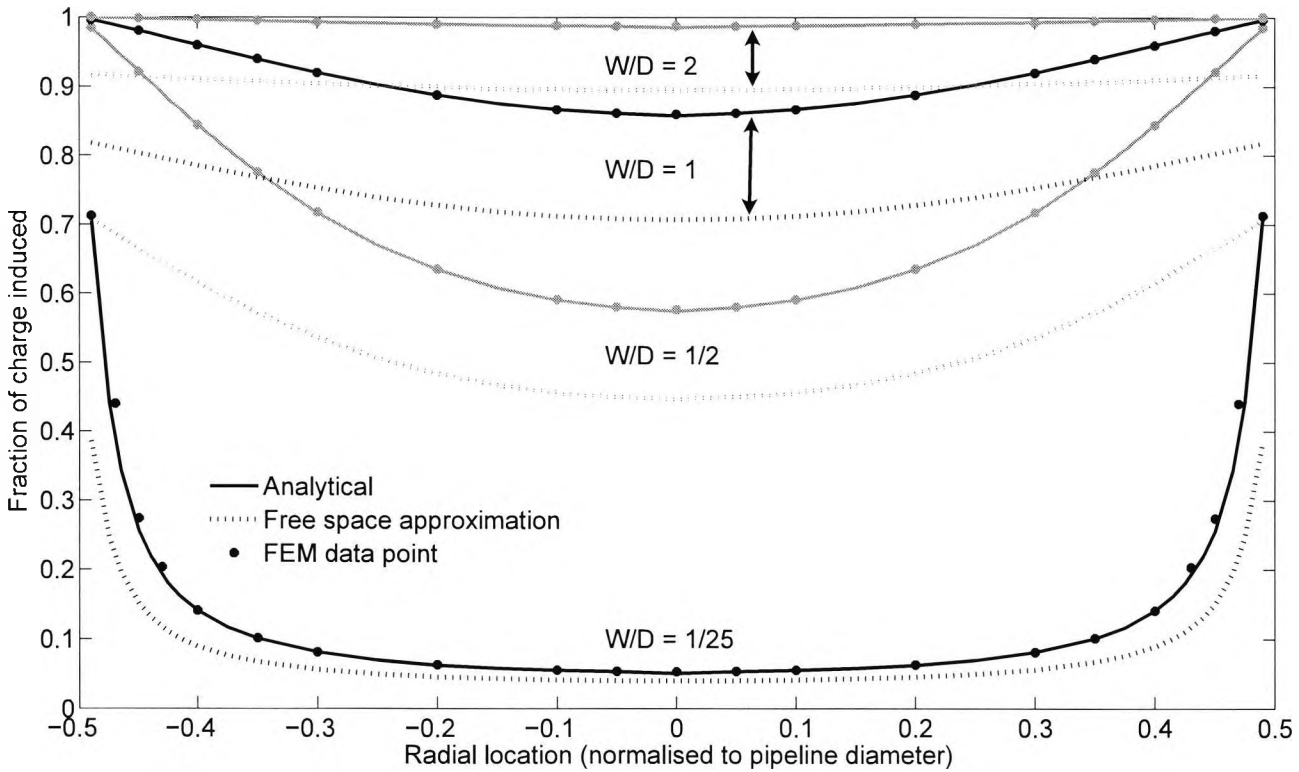


Figure 3.4: Spatial sensitivity of a ring electrode

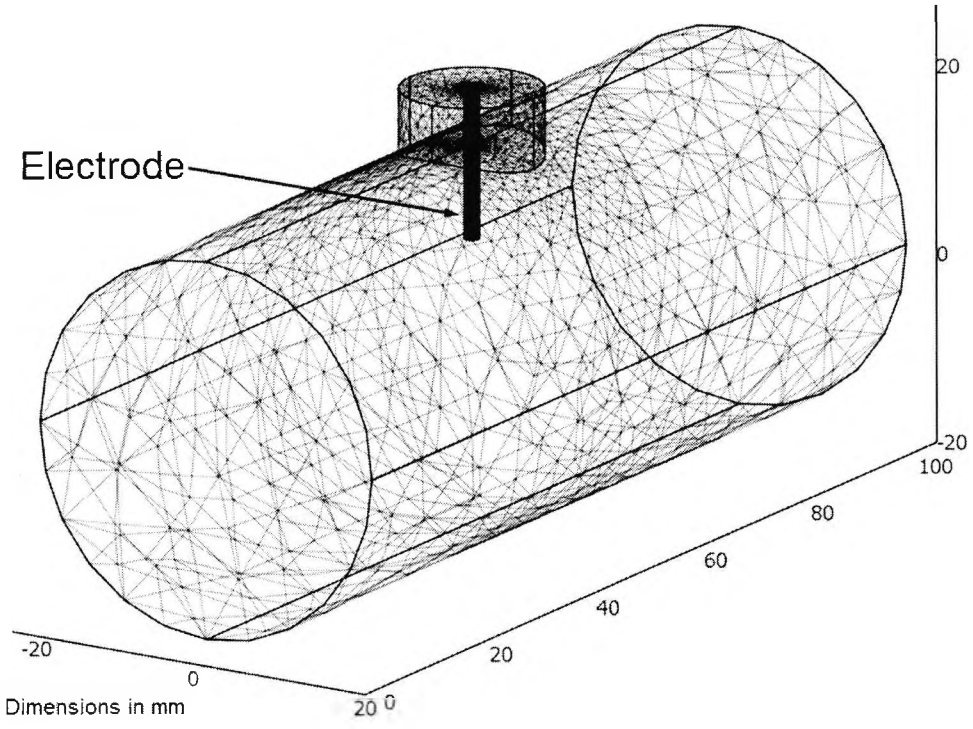
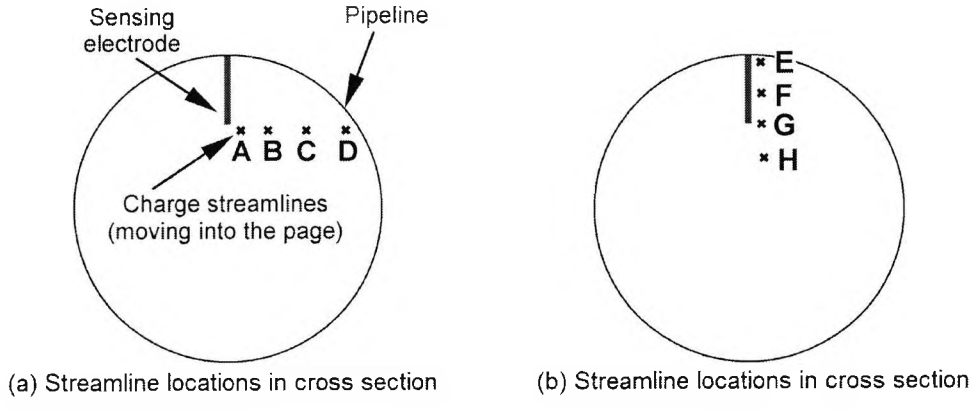


Figure 3.5: FEM model of the single intrusive rod electrode



(a) Streamline locations in cross section

(b) Streamline locations in cross section

Figure 3.6: Charge streamline locations

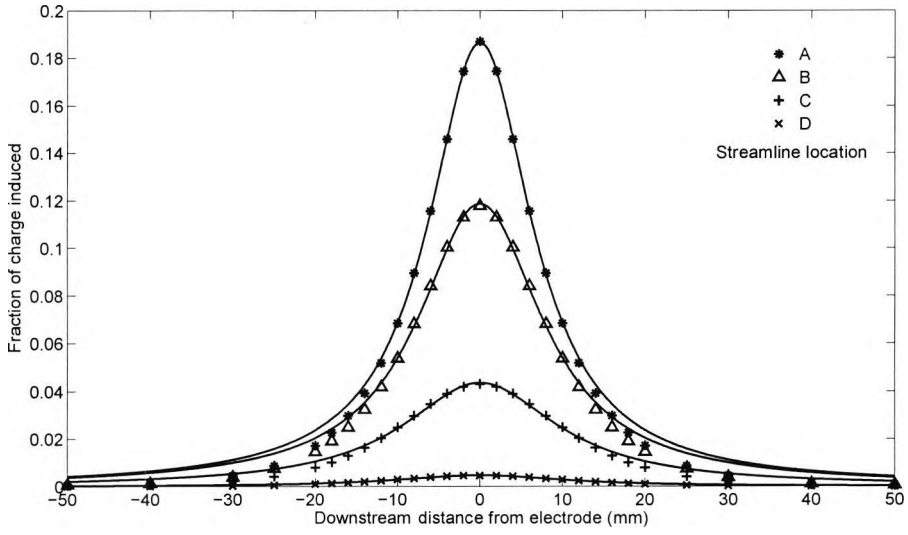


Figure 3.7: FEM data points of induced charge and fitted curves

that can be used to represent the characteristic bell shape of the induced charge (Fig. 3.7). A Gaussian curve [Weisstein, 2009a] of the following form can be considered:

$$q' = ae^{-b(vt)^2} \tag{3.9}$$

where q' is the charge induced on the electrode, a and b are positive real constants, t is time and v is the particle velocity.

Alternatively, a Lorentzian function [Weisstein, 2009b] of the form:

$$q' = \frac{a}{b + (vt)^2} \tag{3.10}$$

may also be a good candidate. Whilst both functions have been used for fitting with the least squares criterion, it is found that the Lorentzian curve gives the best result around the peak,

which is the region of most interest for the electrostatic sensor. The solid lines in Fig. (3.7) are the best fit Lorentzian curves to the FEM data points. The R-square values for the curve fits are greater than 0.987 in all cases, indicating good agreement between the data points and the fitted curves. The model for any particular streamline is therefore a pair of (a,b) coefficients for use with equation 3.10.

3.5.2 Experimental Evaluation

To validate the modelling results, a dedicated laboratory set-up with the same physical dimensions as the FEM model was constructed. Fig. 3.8 shows a schematic diagram of the laboratory set-up. A thin glass pipette was used to carry a single charged particle passing the rod electrode in a streamline halfway up the 5 mm electrode and 1.5 mm laterally away, in a grounded pipeline. The presence of the glass tube slightly affects the electric field, but has been included in the FEM modelling in order to enable a direct comparison. The electrode used in the experiment was electrically shielded from the rest of the environment and connected to a current-to-voltage amplifier, producing a voltage proportional to the induced charge derivative. The input of the current-to-voltage amplifier was kept at virtual ground so that stray capacitances could be ignored, and the 5 kHz bandwidth was designed to be greater than that of the induced signal, making the voltage captured a faithful representation of the induced charge. To derive the induced charge, the signal was integrated so that it could be compared to the FEM results. The particle velocity was measured as 14.3 m/s through cross-correlation of the signal with that of another electrode positioned downstream. Fig. 3.9 shows a direct comparison between the modelling results, fitted Lorentzian curve and experimental data when a particle passes the electrode through the streamline as described above. The corresponding frequency domain response, obtained by performing a Fourier transform on the differentiated signals from Fig. 3.9, is plotted in Fig. 3.10.

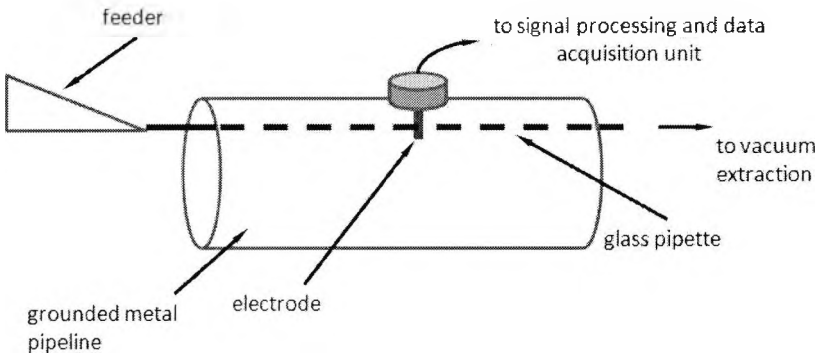


Figure 3.8: Experimental setup for modelling validation

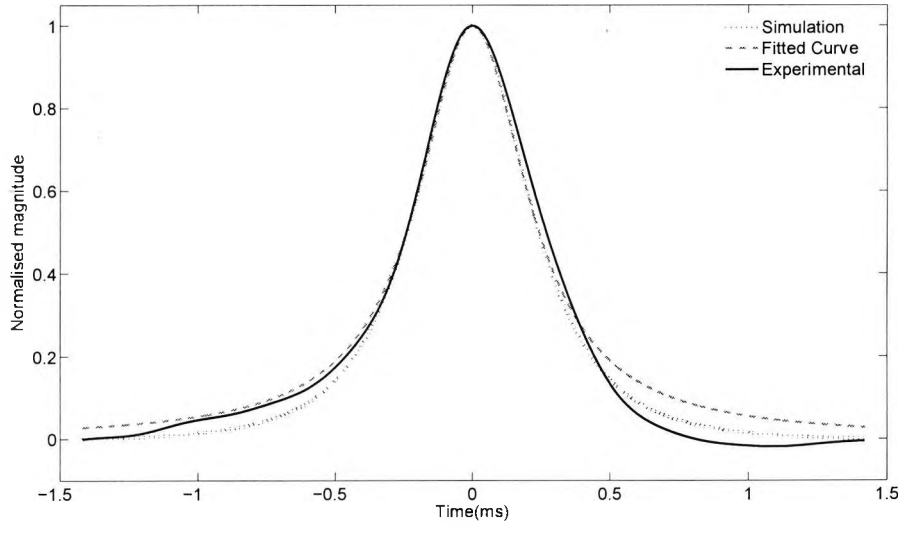


Figure 3.9: Charge induced by a single particle

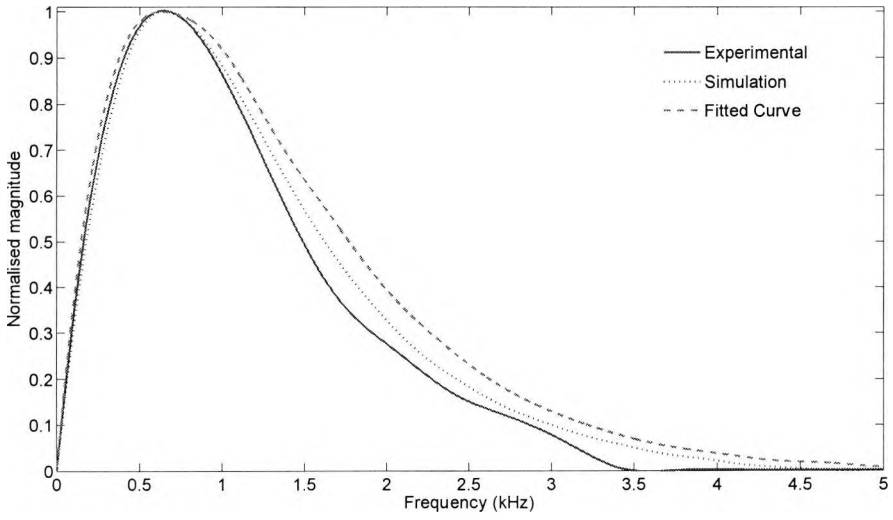


Figure 3.10: Spectral densities for experimental, simulated, and fitted curves

Fig. 3.9 demonstrates that the experimental and modelling results agree closely around peak region, and that the maximum induced charge on the electrode occurs when the charged particle is closest to it. Fig. 3.10 indicates that the curve fitted to the FEM results differs more significantly from the experimental results, but the peak frequency response is similar between all three plots. These similarities, along with the simplicity of the Lorentzian equation, suggest that it is a good model with which to investigate sensor response and improve sensor design. A plot of the maximum induced charge, depending upon the cross sectional location of the particle, has been generated from the FEM simulation and is shown in Fig. 3.11.

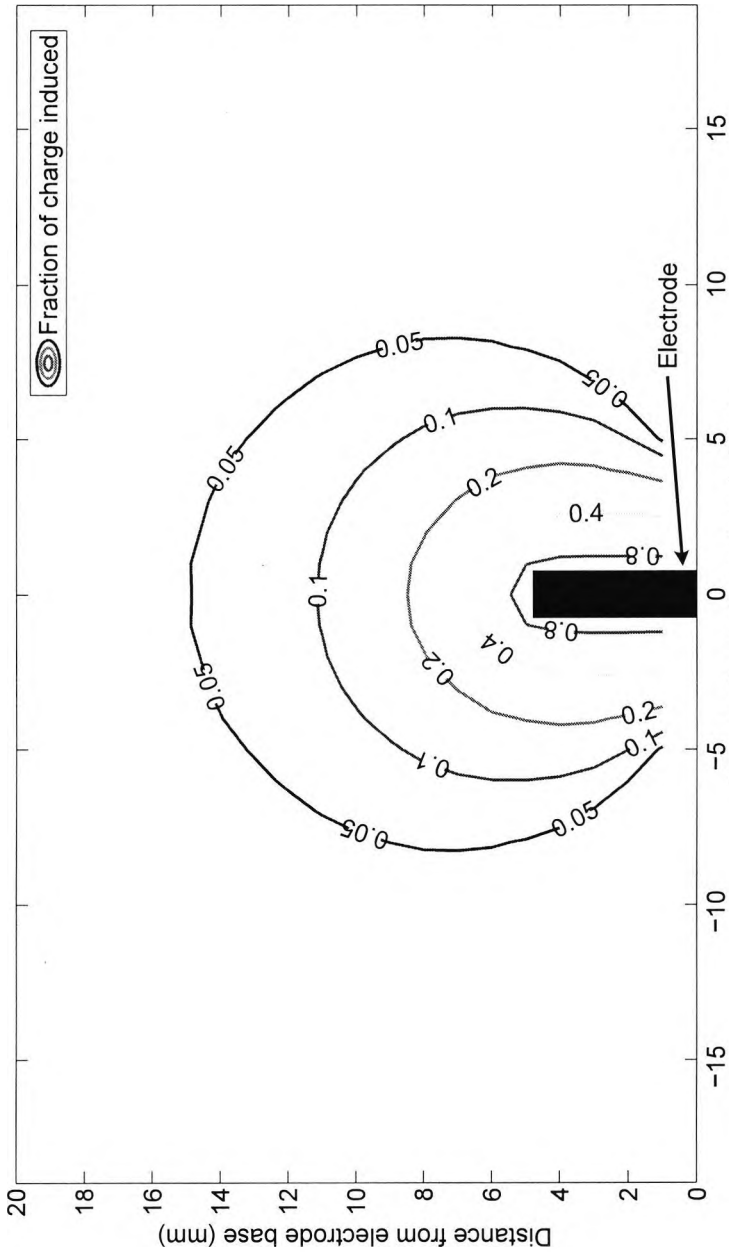


Figure 3.11: Sensing zone of the rod electrode

3.5.3 Electrode sensitivity/bandwidth using FEM

As an example illustrating how the FEM results can be applied, a more detailed examination of a particular electrode configuration was conducted. The configuration is the same as the one used to validate the modelling results, i.e., a 1.6 mm diameter rod electrode intruding 5 mm into a 40 mm bore grounded pipeline, and using a 1 μC point charge. As shown in Fig. 3.7, the sensitivity of the electrode is considerably reduced with lateral distance from the electrode. Fig. 3.12 shows that the actual current on the sensor is even more dramatically reduced when the charge streamline is below the electrode tip. Streamline locations E to H (Fig. 3.6(b)) represent distances of 1 mm, 2.5 mm, 5 mm and 10 mm from the pipe wall respectively, measuring downward from the base of the electrode. The curves in each letter group in Fig. 3.12 represent lateral distances 1 mm apart, the strongest signal being 1 mm from the sensor, the next strongest 2 mm away, and so on. Streamline locations E to G (Fig. 3.6(b)) are within lateral 'sight' of the sensor, whilst location H is below the tip, and contributes much less to the signal. This result, along with similar simulations for electrodes of differing intrusion depths, suggests that, for intrusive electrodes, particles within the lateral sight of the electrode make the most significant contribution in the sensing process. In other words, the sensing field is quite localized around the electrode, and the signal bandwidth is dominated by signals from particle streamlines near the electrode. In practice, however, the streamline distribution is unlikely to be homogeneous. In this case, consideration of particle velocity and bandwidth together can give an indication of how far away from the electrode most of the signal is coming from. The frequency spectrum of the induced signal changes as the charge streamline moves further from the electrode—streamlines further away induce signals that have lower frequency spectrum peaks. This effect, termed 'spatial filtering' [Yan, 1996], implies that particle velocity along with spectral density information can give an idea of flow distribution.

The scale of the X-axis of the induced charge plots (Fig. 3.9) is inversely proportional to the

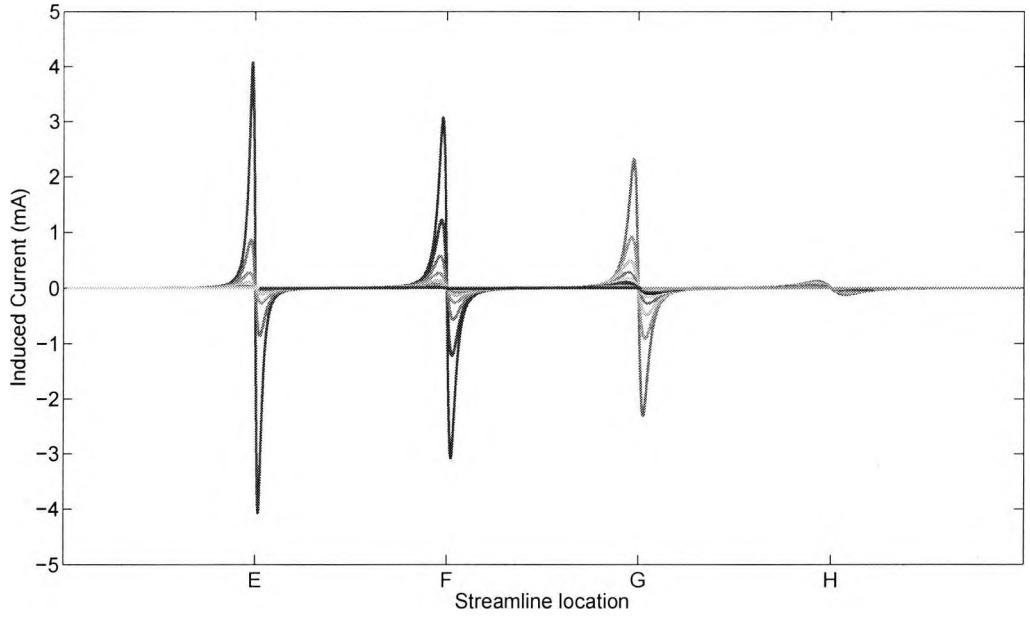


Figure 3.12: Induced current on rod electrode

particle velocity, which implies that the bandwidth of the signal induced on the electrode is proportional to the particle velocity. This makes the bandwidth of the signal a function of both particle velocity and spatial distribution. To demonstrate this effect, a signal sample was taken from tests of a 10 kg/h pulverized coal flow in an industrial test rig [Krabicka et al., 2006]. The same sensor and pipeline geometry described above, with the same electronic conditioning circuit, was used to record the signal. The velocity, as determined by cross correlation, was 11.9 m/s. The frequency spectrum of the signal revealed the peak spectral component to be 309 Hz. Fig. 3.13 shows the peak spectral component of the signal induced by streamlines in the FEM model in terms of their cross sectional location. The value of peak spectral component corresponds to a reference velocity of 11.9 m/s, so that it can be related to the real sensor signal on the test rig. The streamlines corresponding to a peak spectral component of 309 Hz are shown as a dotted line in Fig. 3.13. The peak spectral density of the real sensor signal is closest to that of modelled charge curves around 5 mm

away from the sensor, and this can be thought of as the 'effective distance' from the flow to the electrode. The lack of higher frequency components in the real sensor signal is unlikely to have been caused by the effects of signal conditioning electronics, stray capacitances or other such causes, since the signal frequencies concerned are quite low (only a few kHz) and the sensor was kept at virtual ground. Rather, it is an indication that a large proportion of the signal is due to charge streamlines further away from the sensor electrode, suggesting an inhomogeneous solids distribution in the pipeline. In view of the fact that the electrode was positioned at the top of the pipeline and the velocity of the particles was relatively low, it is not surprising that the bulk of the flow should be concentrated lower down in the pipeline [Krabicka et al., 2006].

3.6 Effect of particle velocity profile on correlation velocity

3.6.1 Power law velocity profile

A key sensor parameter is electrode intrusion depth. This is especially significant when considering flows with a prior knowledge of the velocity profile across the cross section. For example, previous studies using a laser diffraction method [Yan et al., 1995b] have indicated that, in some cases, the particle velocity profile follows a power law distribution (illustrated in Fig. 3.14) given by:

$$V_r = V_{max}\left(1 - \frac{r}{R}\right)^{\frac{1}{n}} \tag{3.11}$$

where V_r is the velocity of a streamline at distance r from the center of the pipeline, V_{max} is the velocity at the centre of the pipeline, r is the distance from the centre of the pipeline to the point of interest, R is the radius of the pipeline, and n is the power law index, ranging

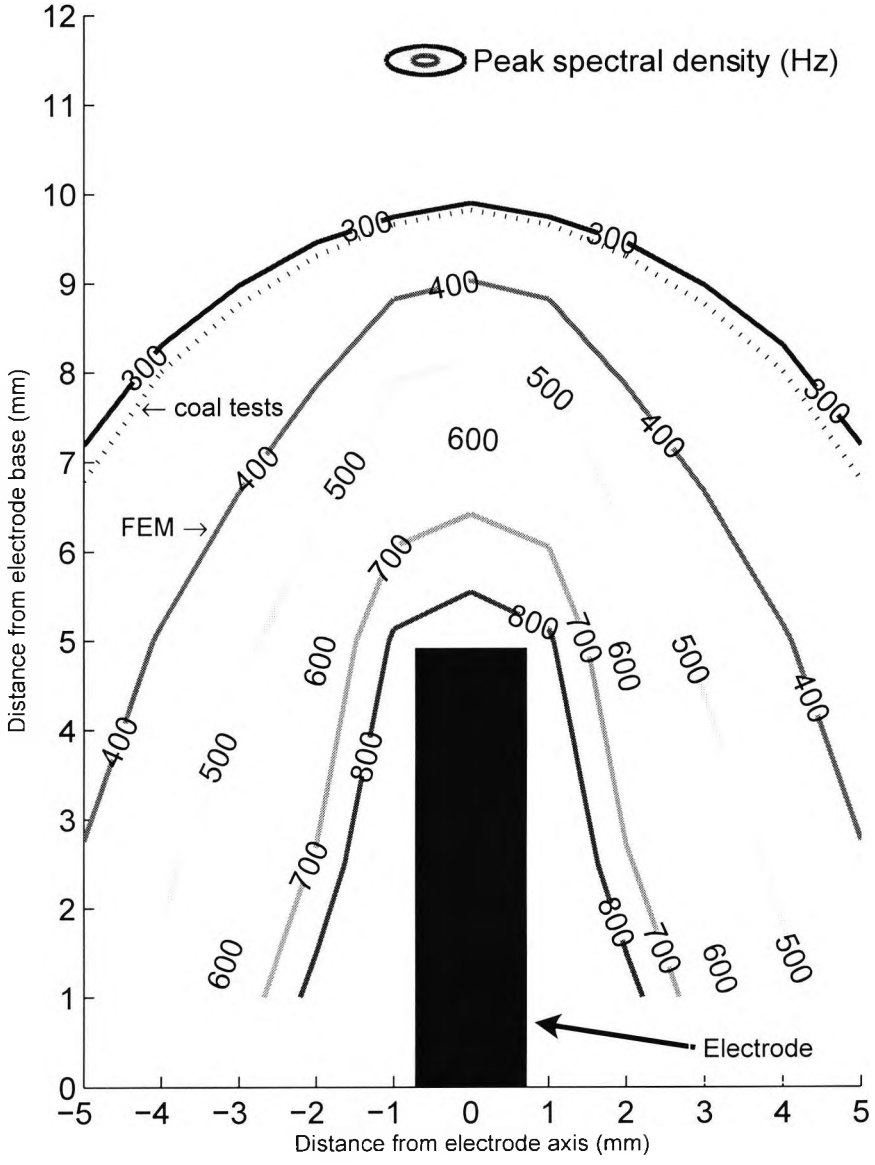


Figure 3.13: Modelled and measured peak spectral components

approximately from 6 to 12. This velocity profile has also been adopted by other researchers in the field [Xu et al., 2007].

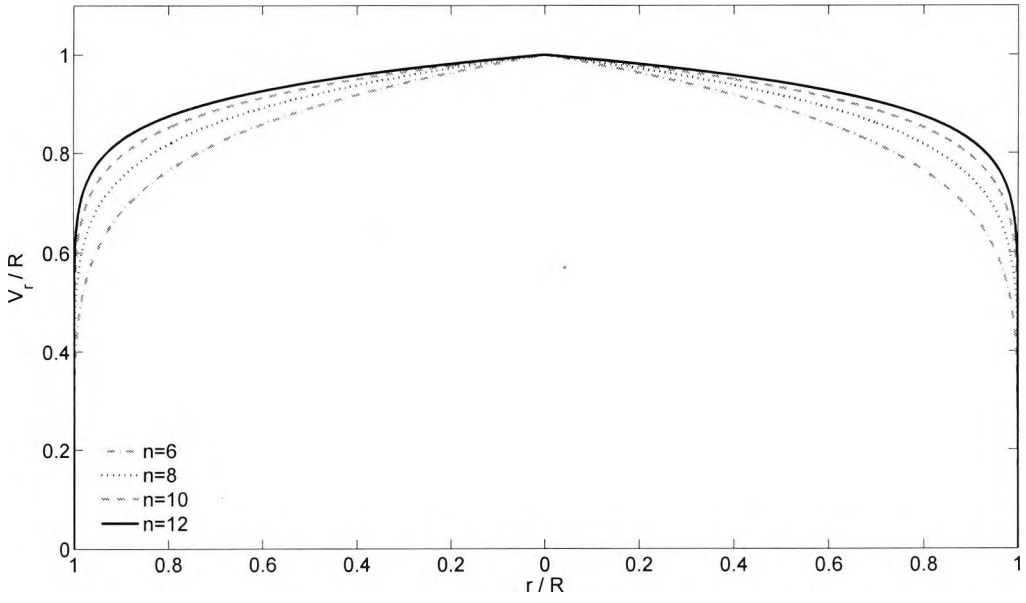


Figure 3.14: Power law velocity profiles

An intrusive electrode is exposed to all or part of this profile, depending on the intrusion depth. The cross correlation of the signals from a pair of intrusive electrodes, however, does not result in the average value of the velocity profile to which they are exposed. The correlation peak will occur at a time lag corresponding to a velocity somewhere between the minimum and maximum values, but it will not, in general, occur at a time lag corresponding to the mean of those values, as will be shown in the following analysis. A closer inspection of cross correlation applied to this situation must be made to reveal the true location of the correlation peak.

3.6.2 Cross correlation analysis for a system of particles travelling with different velocities

In a system of particles moving at different velocities, let $f(t)$ be the upstream signal induced by a single particle and $F(\omega)$ be its Fourier transform. For a system of N randomly distributed particles, the upstream signal can be represented in the time and Fourier domains as:

$$\sum_{n=1}^{N} f(t-\tau_n) \Longleftrightarrow \sum_{n=1}^{N} F(\omega)e^{-j\omega\tau_n} \tag{3.12}$$

where τ_n represents each particle's axial distance from a given reference (say, the upstream electrode), and depends on the particle distribution in the pipeline at any particular time, t. If the particles in the system are moving downstream with different velocities then the downstream signal can be represented as:

$$\sum_{n=1}^{N} f(t-\tau_n-\lambda_n) \Longleftrightarrow \sum_{n=1}^{N} F(\omega)e^{-j\omega(\tau_n+\lambda_n)} \tag{3.13}$$

where the additional parameter λ_n represents the time lag from the upstream to downstream positions and depends on each particle's velocity.

Working in the Fourier domain, and using the Correlation Theorem, the cross correlation, $P(\omega)$ of upstream and downstream signals is:

$$\begin{aligned} P(\omega) &= \left(\sum_{n=1}^{N} F(\omega)e^{-j\omega(\tau_n+\lambda_n)}\right)\left(\sum_{n=1}^{N}\overline{F(\omega)e^{-j\omega(\tau_n+\lambda_n)}}\right) \\ &= F(\omega)\left(e^{-j\omega\tau_1}+e^{-j\omega\tau_2}+\ldots+e^{-j\omega\tau_N}\right)\overline{F(\omega)\left(e^{-j\omega(\tau_1+\lambda_1)}+e^{-j\omega(\tau_2+\lambda_2)}+\ldots+e^{-j\omega(\tau_N+\lambda_N)}\right)} \\ &= F(\omega)\overline{F(\omega)}\left(e^{-j\omega\tau_1}+e^{-j\omega\tau_2}+\ldots+e^{-j\omega\tau_N}\right)\left(e^{j\omega(\tau_1+\lambda_1)}+e^{j\omega(\tau_2+\lambda_2)}+\ldots+e^{j\omega(\tau_N+\lambda_N)}\right) \\ &= C(\omega)\left(\sum_{i=1}^{N}\sum_{k=1}^{N}e^{j\omega\lambda_1+\tau_k-\tau_i}\right) \end{aligned} \tag{3.14}$$

where $c(t) \Leftrightarrow C(\omega)$ is the autocorrelation of $f(t) \Leftrightarrow F(\omega)$. When $i = k$ in (3.14), then the contribution to the cross correlation is the single particle autocorrelation signal, time shifted by the upstream to downstream transit time delay of that particle. When $i \neq k$, then the contribution is a randomly shifted autocorrelation signal and represents a noise element. This noise is evenly spread out over the cross correlation and, as such, it does not significantly affect the location of the correlation peak. The correlation can therefore be reduced to:

$$P(\omega) = C(\omega)\left(e^{-j\omega\lambda_1}+e^{-j\omega\lambda_2}+\ldots+e^{-j\omega\lambda_N}\right) \tag{3.15}$$

Converting back into the time domain, the cross correlation $\rho(t)$ is:

$$\rho(t) = c(t-\lambda_1)+c(t-\lambda_2)+\ldots+c(t-\lambda_N) \tag{3.16}$$

i.e., it is the scaled sum of time shifted autocorrelations of the single particle signal, with the time shifts corresponding to the individual delays of the particles.

Let $\lambda(r)$represent the time delay at radius r caused by the velocity profile. Assuming the velocity profile follows the power law distribution given in equation (3.11), then

$$\lambda(r) = \frac{\mathrm{l}}{V_r} = \frac{s}{V_{max}(1-\frac{r}{R})^{\frac{1}{n}}} \tag{3.17}$$

where s is the spacing between upstream and downstream electrodes.

Assuming a uniform particle distribution, the cross correlation of upstream and downstream signals can be expressed, using equation (3.16), as:

$$\rho(t) = \int_{R-L}^{R} c(t - \lambda(r))\, dr \tag{3.18}$$

Where L is the length of the electrode, and R is the radius of the pipeline.

Substitution for $\lambda(r)$ in equation 3.18, and using k to stand for a scaling factor depending on signal magnitude, gives:

$$\rho(t) = k \int_{R-L}^{R} c\left(t - \frac{s}{V_{max}(1 - \frac{r}{R})^{\frac{1}{n}}}\right) dr \tag{3.19}$$

for electrode lengths less than or equal to the pipeline radius, or

$$\rho(t) = k \left[\int_{0}^{R} c\left(t - \frac{s}{V_{max}(1 - \frac{r}{R})^{\frac{1}{n}}}\right) dr + \int_{0}^{L-R} c\left(t - \frac{s}{V_{max}(1 - \frac{r}{R})^{\frac{1}{n}}}\right) dr \right] \tag{3.20}$$

for electrode lengths greater than the pipeline radius.

The correlation velocity can then be calculated by dividing the electrode spacing, s, by the delay time corresponding to the peak cross correlation value. The autocorrelation, $c(t)$, depends on the single particle signal, which can be considered an impulse response and can be approximated by the Lorentzian curve fit from FEM results in order to model the sensor response, using equations 3.19 and 3.20. The non-uniform spatial sensitivity of the sensing field means that $c(t)$ will decrease in magnitude as the distance from the electrode increases. However, the decrease in magnitude with distance will be approximately the same along

its length, and the decreasing signal magnitudes will simply contribute to an overall scaling factor, which is absorbed into k in (3.20). In practice, k is of little significance, since the correlation is usually normalised expressed as the correlation coefficient.

The velocity profile V_r can be used to calculate the true mean particle velocity, v_{ave}, and true mean particle time delay, λ_{ave}, by:

$$\begin{aligned} v_{ave} &= \frac{1}{\pi R^2} \int_0^{2\pi} \int_0^R V_{max} \left(1 - \frac{r}{R}\right)^{\frac{1}{n}} r dr d\theta \\ &= \frac{2n^2}{(n+1)(2n+1)} V_{max} \end{aligned} \tag{3.21}$$

$$\lambda_{ave} = \frac{s}{v_{ave}} \tag{3.22}$$

The correlation velocity of the sensor can be compared to the true mean particle velocity to find the relation between them. (3.20) does not yield a closed form solution, and values of $\rho(t)$ for various values of electrode length must be calculated numerically. The results for a electrode intrusions at intervals of 0.1 of the diameter of the pipeline are shown in Fig. 3.15.

Fig. 3.15 indicates that, when the electrode intrusion depth is slightly less than the radius of the pipeline, the relative error between the correlation velocity and the true mean velocity converges to close agreement for the various values of the power law index, n. In fact, the agreement is closest for an intrusion depth of 0.40 of the pipeline diameter, where the correlation velocity averages 1.71% above the true mean (ranging from 1.62% to 1.84% depending on n). In this case, the correlation velocity can be scaled down by 1.71% to obtain the true mean to within -0.09% to +0.13%, for power law indices from 6 to 12. Another option would be to take the mean intrusion depth at which the correlation velocity and true

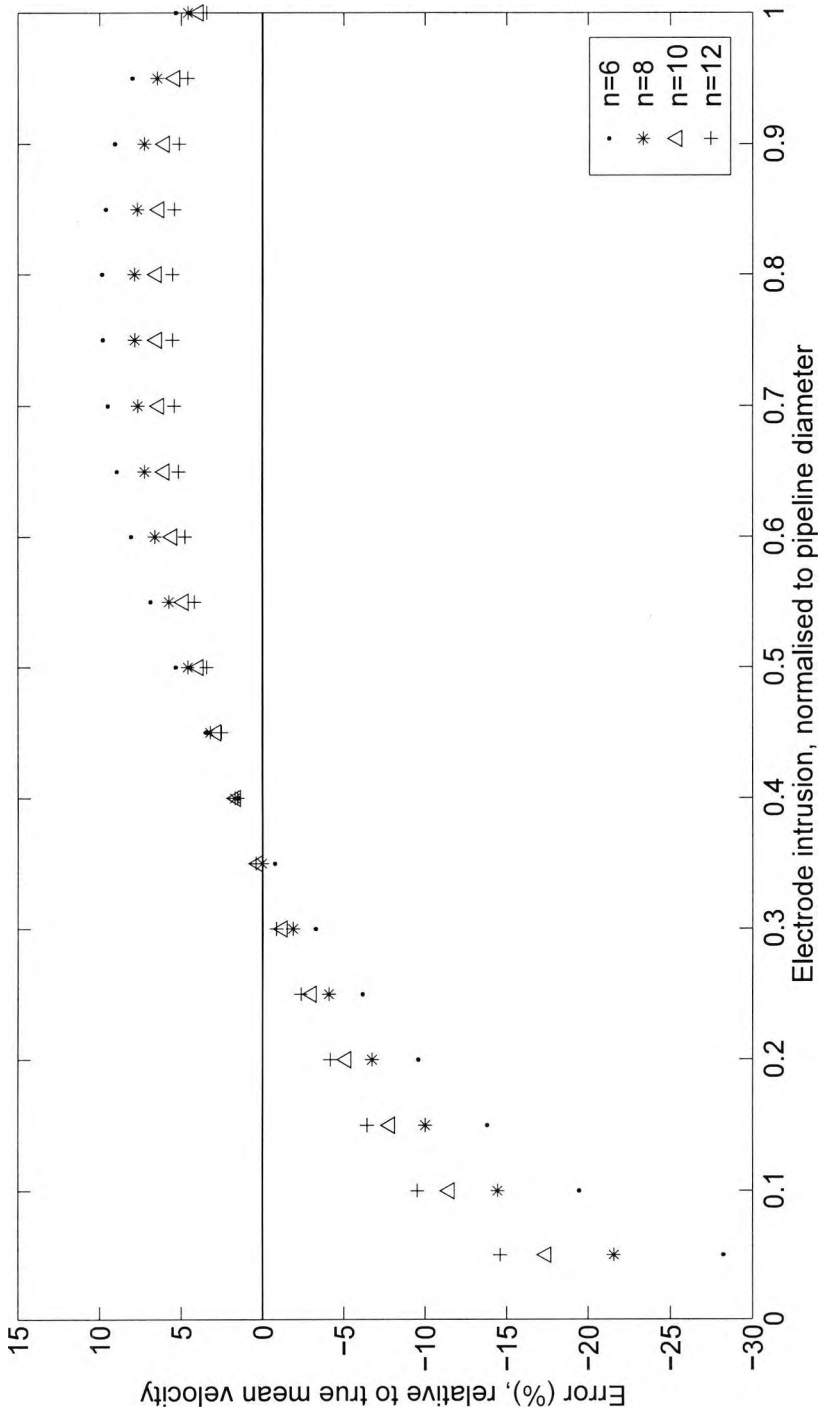


Figure 3.15: Correlation velocity vs. true mean velocity

mean velocity are equal. This is calculated as 0.35 of the pipeline diameter. No scaling is required in this case, but the error range is slightly greater, ranging between -0.90% and 0.34%.

3.7 Effects of turbulent conveying air

When the conveying air becomes more turbulent, the velocity profile across the pipeline becomes 'flattened', and the particles begin to behave as if they were suspended in a random velocity field. In these situations, the properties of the correlation function can offer insight into the nature of the velocity field.

It can be shown [Olla, 2008] that inertial particles in a turbulent medium behave as if they were in a smooth, incompressible random velocity field, i.e., they behave as if suspended in a time-uncorrelated Gaussian flow. This is valid for dilute suspensions—as is the case in pulverised coal flow—where the effect of the particles on the turbulence of the conveying air can be neglected [Lun and Liu, 1997, Bec et al., 2007, Hajji et al., 1996]. The presence of intrusive electrodes, however, causes an additional disturbance to the flow that is now considered.

The physical restriction due to the cross sectional area taken up by the electrodes is a consideration, but this area is relatively small. In the pipeline used for tests on the 500 kW combustion test facility in Didcot, UK, the cross sectional area of the pipeline (40 mm internal diameter) was 1257 mm^2. The 5 mm, 10 mm, and 20 mm intrusive electrodes used in these tests, each with a diameter of 1.6 mm, take up only 0.6%, 1.3% and 2.5% of the cross sectional area respectively.

The turbulent eddies caused by the electrodes should also be considered. In single phase fluid flow measurement, the eddies themselves are used to determine flow velocity, as the

rate of their formation, under certain conditions, depends only on pipeline geometry and fluid velocity [Liptak, 1993]. The effect of the extra turbulence on particles in a two phase flow is not as straightforward, and depends not only on electrode/pipeline geometry, but also on the properties of the particles and air stream. If the momentum of an individual particle is small, it will closely follow the turbulence of the airflow, but where the momentum of a particle is large enough, its inertia allows it to resist the influence of small scale turbulent eddies and moves in a more linear manner. With the availability of more powerful computing power and CFD algorithms, this has been the subject of recent research. [Apte et al., 2003, Hajji et al., 1996, Graham and James, 1996, Graham, 1996, Lun and Liu, 1997, Azzopardi, 1999, Huang et al., 2006, Bec et al., 2007, Reynolds, 2004]

The Stokes number (dimensionless) for gas-solids flow is the main parameter of interest, and is defined [Fokeer et al., 2004, Huang et al., 2006] as:

$$St = \frac{\rho_p d_p^2 U}{18 \mu_g L_c} \tag{3.23}$$

Where:

ρ_p is the kinematic viscosity of the fluid,

d_p is the particle diameter

μ_g is the dynamic viscosity of the gas

U is the free stream velocity

L_c is the characteristic length of the obstruction

For Stokes numbers $\ll 1$, the particles closely follow the turbulent streamlines of the conveying medium; for Stokes numbers $\gg 1$, the particles are effectively 'disentangled' from the turbulence [Reynolds, 2004, Fokeer et al., 2004].

Consider a typical coal flow around a 1.6 mm diameter electrode, taking the mean particle diameter to be $40\mu m$, $\rho_p = 1100 kg/m^3$, $\mu_g = 1.82 \times 10^{-5} Ns/m^2$, $U = 20 m/s$, and $L =$

$1.6 \times 10^{-3}m$, the Stokes number for this 67. For the smallest particles of around $5\mu m$ it is about 1.05, and for large particles of $150\mu m$ it is 944. It can be seen, then, that for most of the particles, the Stokes number is much greater than 1 except for the smallest range of particles, where is it approximately 1. This indicates that most of the particles effectively 'ignore' the turbulence around the electrode, and are subject only to the random velocity field of the turbulent conveying air. The smallest particles, however, are influenced by the turbulent eddies caused by the presence of the electrode, and cause an increase in their fluctuation.

The Stokes number also has an effect on particle-electrode collisions. Particles with small inertia (Stokes number around 1 or less) can follow the airstream around the stagnation point of the electrode and do not collide with it. Particles with higher Stokes numbers ($\gg 1$) have too much momentum to avoid the electrode and collide with it. As a result of this collision, the electrostatic charge on the particle is discharged on to the sensing electrode (which is at ground potential). This is evident in the results of industrial coal flow tests [Shao et al., 2009], where the RMS value of the signal on the upstream electrode is higher than that of the downstream electrode.

These collisions do not have an adverse affect correlation velocity measurement. A colliding particle is electrostatically discharged by the upstream electrode and the short distance between electrodes ensures that significant charge is not built up by the time the particle reaches the downstream electrode. Again, this is evident in the lower RMS value of the downstream signal. This means that the signal component due to the colliding particle will have no downstream counterpart with which to cross correlate. The transient signal on the upstream electrode, however, will contribute to the normalising factor in the denominator of the cross correlation coefficient calculation. Hence, the only effect will be extra uncorrelated noise on the upstream electrode resulting in a slight decrease in the overall correlation coefficient peak.

3.8 Electrode spacing

Spacing for optical and capacitance sensors based on cross correlation has been discussed by Mesch and Kipphan [1972] who calculate the optimal spacing to be

$$L_e = \frac{0.88V_m^2}{\sigma\omega_0} \tag{3.24}$$

where L_e is the sensor spacing, V_m is the mean velocity of the particles in the pipeline, σ is the standard deviation of the velocity of the particles in the pipe and $\omega_0$is the bandwidth of the flow noise in rad/s. For example, an average flow velocity of 10 m/s to 20 m/s, a flow noise bandwidth of 3kHz and a standard deviation of 1 m/s would give an optimal spacing of between 4.7 mm and 18.7 mm. The standard deviation is not generally known, but initial results from electrostatic sensors in laboratory tests on a 40 mm pipeline using materials with a wide particle size distribution, showed a well defined correlation peak, suitable for making velocity measurements, using an electrode spacing of 10 mm.

Although this spacing generates a strong correlation peak, subsequent analysis has revealed that it can create undesired effects. In section 5.2, for example, it will be seen that the velocity profile indicated by the different electrode intrusions are contrary to the normal tendency for greater velocities in the center, decreasing toward the pipeline wall. This effect can be at least partly explained by considering the electrode spacing. It has previously been noted [Yan, 1992] that with close electrode spacing there is the possibility that the electrodes will interfere with each other's electric fields, causing an error in correlation velocity. In industrial sensors, the spacing tends to be around 50 mm. Although the effect has been suggested the practical spacing limit necessary to avoid significant interaction of electric fields has not been reported. Ideally, there should be no interaction between upstream and downstream electrodes, resulting in an induced charge curve that is symmetrical about the center of the electrode.

An electrode spacing of 10 mm was used for the tests in section 5.2 without prior knowledge of the effect of electrode interaction due to this close spacing. Subsequent analysis, using finite element modelling, has revealed that there is, if fact, considerable interaction of electric fields for close electrode spacing, helping to account for the unexpected velocity profile of some of the industrial trials. Modelling of the effect was conducted as follows:

Charged particle streamlines were modelled for several electrode intrusion depths at 1 mm, 3 mm, and 5 mm laterally away from the electrode. The resulting induced charge curves (Fig. 3.16) were differentiated to give and induced current (Fig. 3.17), and the upstream and downstream current signals were cross correlated (Fig. 3.18) to find the correlation distance—ideally equal to the electrode spacing. The correlation distance (i.e., the lag at which the correlation peak is found) for each lateral distance was averaged at several points along its length. From Fig. 3.16, the asymmetry of the induced charge can be clearly seen—as the charge moves from the upstream to downstream electrodes (0 mm to 20 mm), the induced charge on the upstream electrode falls off more quickly than when the charge initially approaches the upstream electrode (-20 mm to 0 mm) and there is a visibly sharper 'knee' in the curve. Also, the peaks of the curves are seen to be slightly closer together than the 10 mm electrode separation distance.

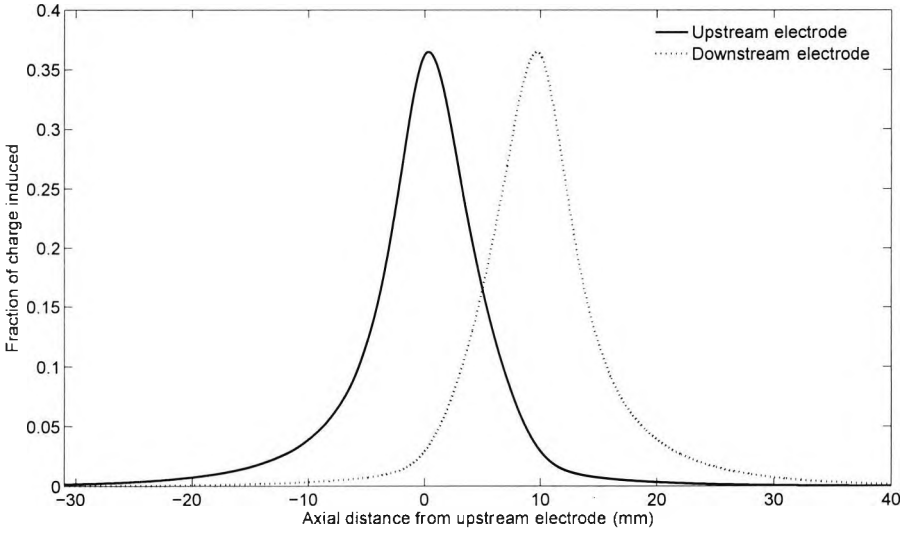


Figure 3.16: Induced charge—10 mm spacing

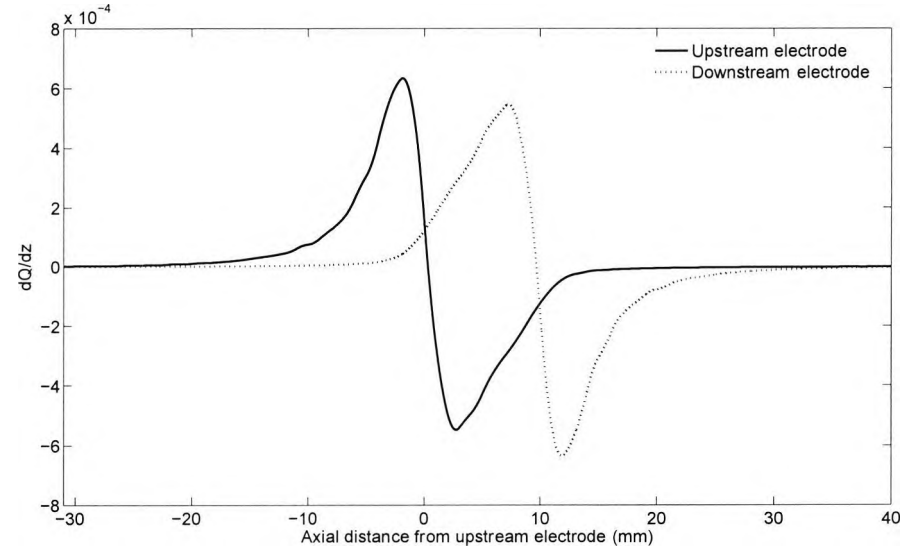


Figure 3.17: Differentiated induced charge—10 mm spacing

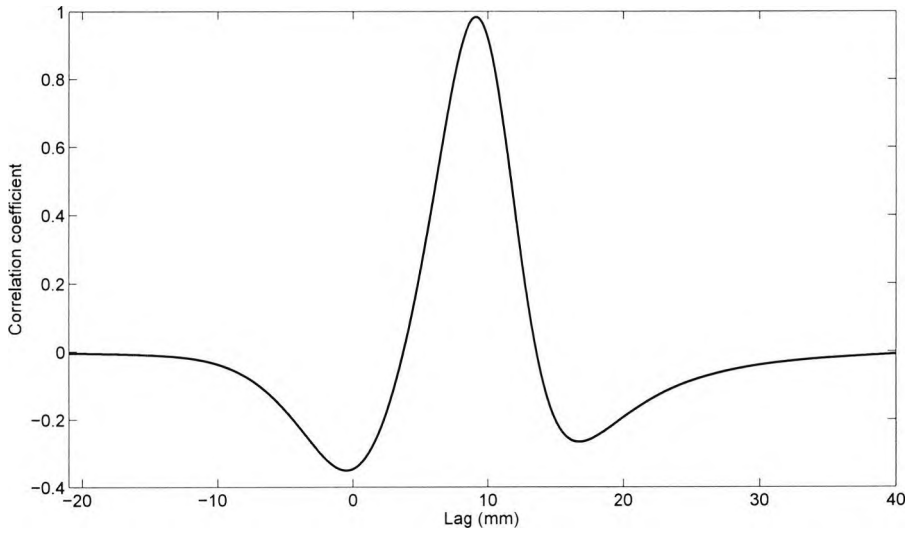


Figure 3.18: Cross correlation—10 mm spacing

The results can be seen in Fig. 3.19. It is immediately apparent that for the close 10 mm spacing, that the correlation velocity is overestimated by a significant amount. Also, the error is greater for shallower intrusions. This helps account for the 'inverse' velocity profile discussed in the Didcot results, as the short intrusion electrodes measuring the flow near the pipeline wall show a higher velocity than the true value.

Increasing the electrode separation distance reduces the effect of electric field interaction, creating charge induction that is symmetrical about the electrode (Fig 3.20). The error in correlation velocity, Fig.3.21, shows that by increasing the electrode separation distance to 20 mm or 30 mm, the correlation velocity error is reduced to a level that can be regarded as insignificant. Slightly lower errors are seen for a separation distance of 30 mm, but the stronger correlation peaks that would result from the closer 20 mm spacing suggest that this is a better option—reducing the electric field interaction to acceptable levels but with electrode spacing close enough to achieve a strong correlation result between upstream and downstream signals.

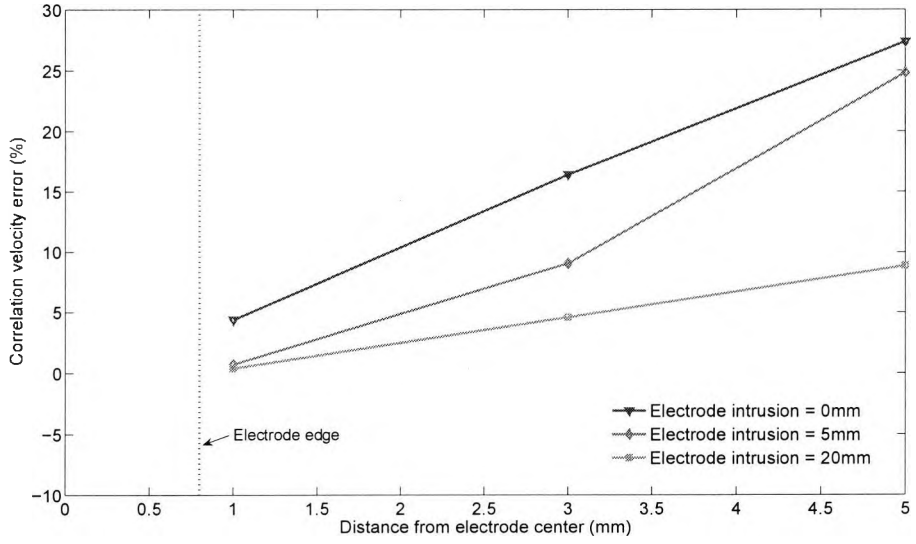


Figure 3.19: Correlation velocity error—10 mm spacing

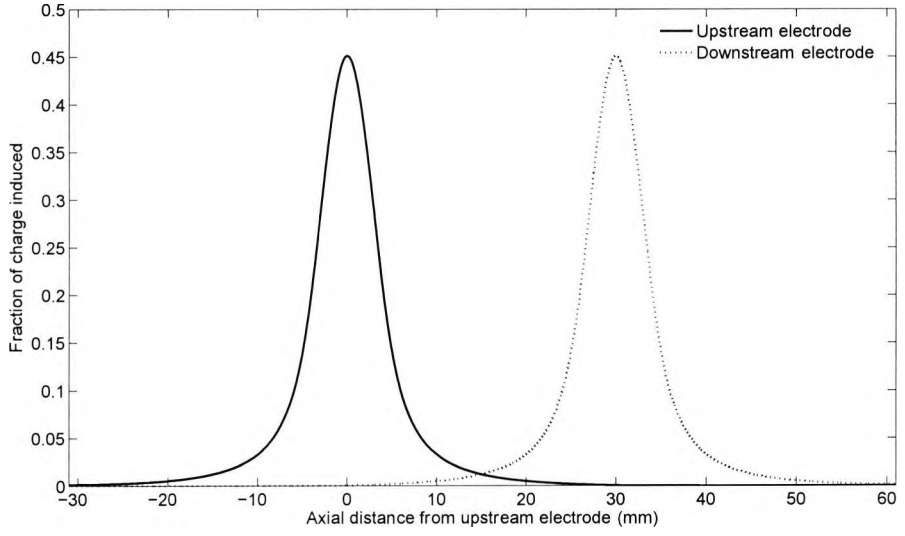


Figure 3.20: Induced charge—30 mm electrode separation

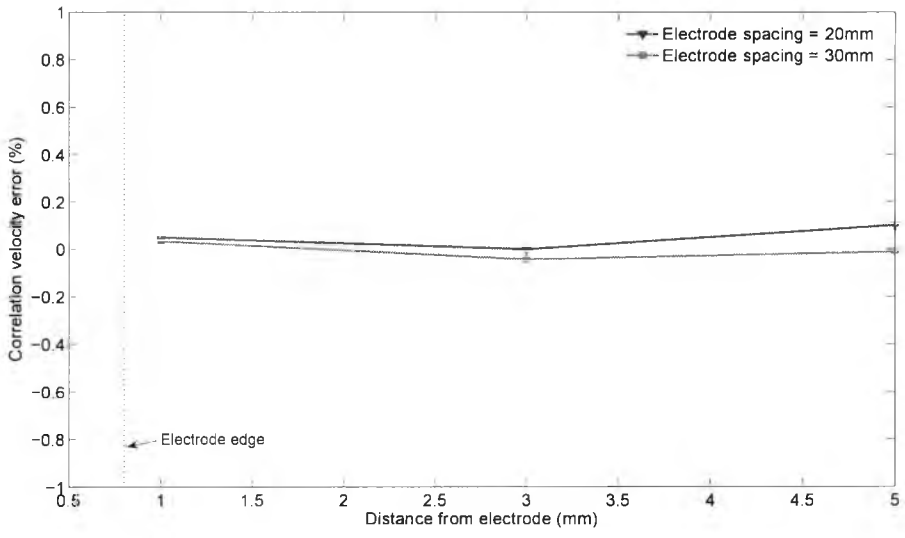


Figure 3.21: Correlation velocity error—16 mm electrode intrusion depth

3.9 Electrode cross sectional shape

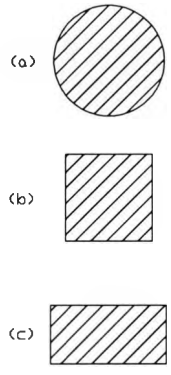


Figure 3.22: Electrode cross sectional shapes (a) circular, (b) square and (c) blade

Another consideration is the cross sectional shape of the intrusive electrode. Among the choices are round, square and 'blade' shaped (defined here as rectangular, with the long sides twice as long as the short ones), shown in Fig. 3.22. To determine the charge induced

onto electrodes of these shapes, a finite element model was constructed using commercial software, COMSOL. In each case, the electrode intrusion depth was 5 mm, modelled in a 40 mm internal diameter pipeline. The dimensions of each shape were set to ensure that the surface area was equal to that of an electrode with a 1.5 mm round cross section. Induced charge curves for particle streamlines 1 mm, 3 mm and 5 mm laterally from the electrode axis are shown in Fig. 3.23. It is evident that the induced charge on electrodes with different cross sectional shapes is broadly similar, suggesting that electrode shape is not critical. In light of this, practical results were obtained using only the circular cross section.

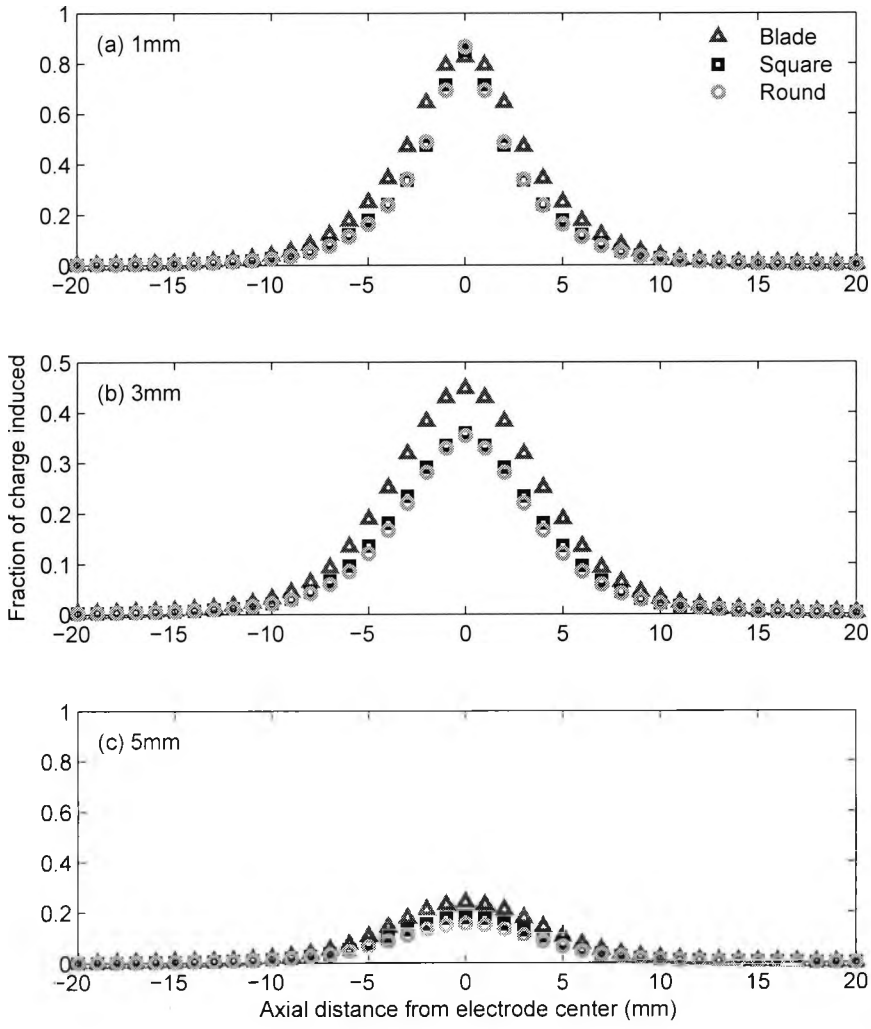


Figure 3.23: Induced charge on electrodes with blade, square, and round cross sections by streamlines 1 mm, 3 mm and 5 mm away from the electrode axis

3.10 FEM validation using belt rig

Induced charge curves on sensor electrodes determined by finite element modelling should be accurate, as FEM implementation is based on well established physical principles and Poisson's equation (equation 3.6). With any modelling, however, it is never certain that all relevant parameters have been considered. Comparison of modelling with experimental results provides increased confidence in the validity of simulation results. To this end, a belt rig setup was constructed (Fig. 3.24) on which a charged bead (a small droplet from a hot melt glue gun) was carried, at known speed, past a sensor electrode surrounded by a grounded metal casing. This setup is an approximation of a point charge as it passes a sensor electrode in a grounded metal pipeline. The signal induced on the electrode was captured by the same electronics as described in section 4.2.2.

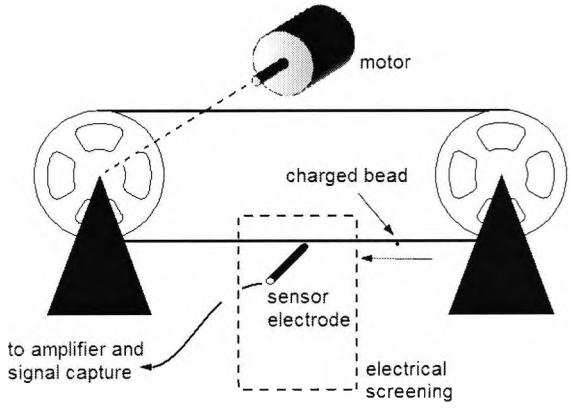


Figure 3.24: Belt rig for finite element electrode model validation

In order to compare the belt rig with finite element modelling, a single particle signal was simulated using FEM. To determine the single particle signal, a series of 'static' simulations

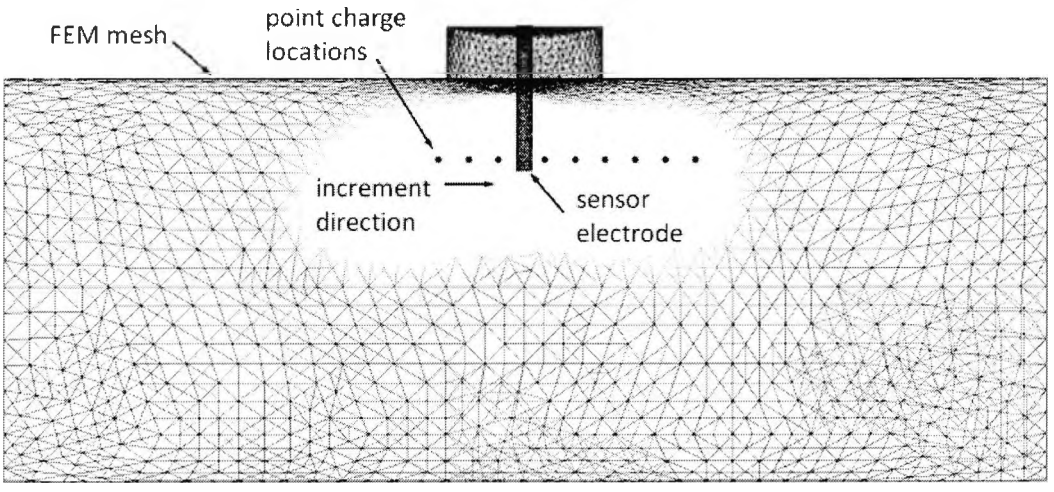


Figure 3.25: FEM simulation for comparison with belt rig

was conducted for different locations of a single point charge in the pipeline (Fig. 3.25), recording the charge induced onto the electrode after each simulation. The point charge was moved incrementally downstream, repeating the simulation for each step along the way in an animation-style procedure. In this manner, the charge induced onto the electrode was determined for a series of upstream to downstream locations of the point charge. To express the abscissa in terms of time rather than location, the substitution $s = vt$ was made, where s is the distance from the electrode, v is the velocity of the particle and t is time. The electronics used to capture the signal produce an output that is proportional to the time derivative of the induced charge signal, i.e., a signal proportional to the induced current . Therefore, the time derivative of the FEM signal was used in order to compare it with the result obtained from the belt rig.

The results from the belt rig and the finite element modelling are compared in Fig. 3.26, where the velocity of the FEM simulation has been scaled to match that of the velocity of the charged bead on the belt rig, measured as 4.3 m/s using a tachometer applied to the belt. The signals from both the FEM and belt rig results have been normalised to their respective

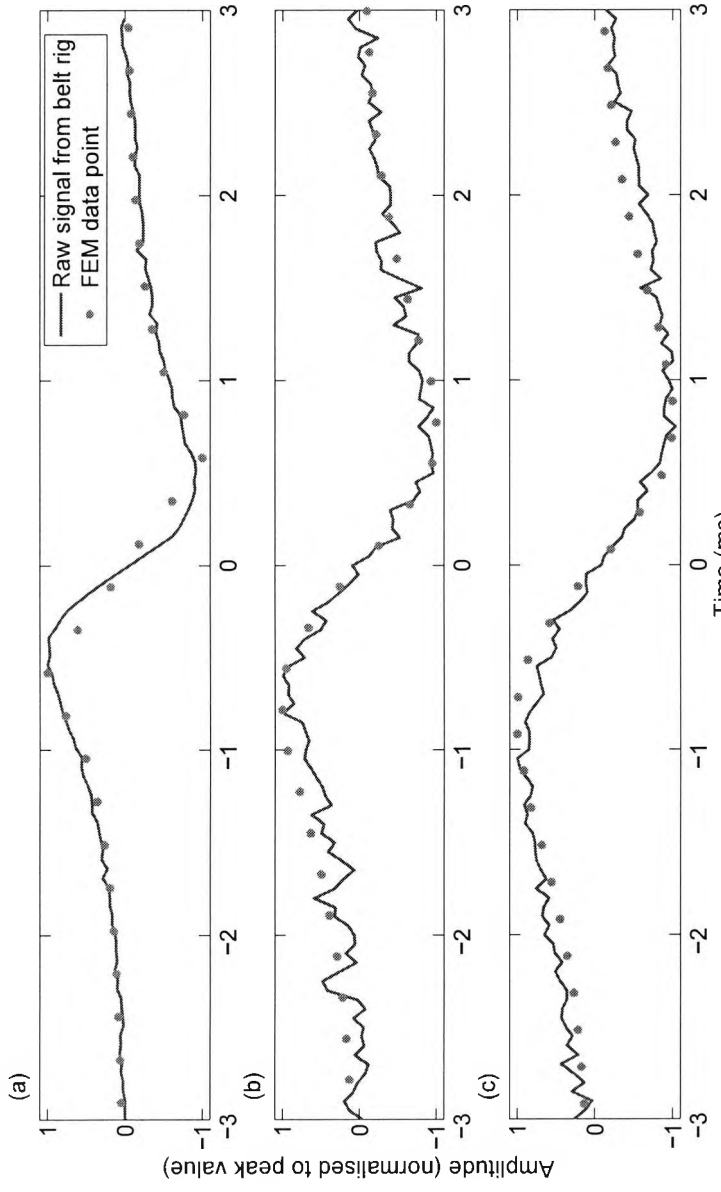


Figure 3.26: Belt rig and FEM streamlines compared at distances (a) 1 mm, (b) 3 mm and (c) 5 mm from electrode

maximum values in order to enable a direct a comparison. This is necessary because it is not practical to try to determine the exact charge induced on the charged bead. Even if a charge sensor was used to determine the charge at a particular time, friction with the air during belt movement and charge leakage to the belt itself would invalidate reading by the time a signal was induced onto the sensor electrodes. In any case, the principle of superposition ensures that the amount of charge induced is simply a scaling factor, and does not influence the shape of the induced charge curve on a particular streamline. The three induced charge curves in Fig. 3.26 correspond to particle streamlines which are (a) 1 mm, (b) 3 mm and (c) 5 mm laterally away from the electrode, midway along its length.

It can be seen from Fig. 3.26 that the belt rig and FEM simulation streamlines agree reasonably well, offering confidence that the finite element model is a valid way to study charge induced on an electrode. The slight differences in shape can be accounted for by the fact that it is difficult to avoid a slight 'wobble' in the belt as it is moving. This means that there is a degree of uncertainty regarding the exact distance of the charged bead from the electrode as it is moving past. This uncertainty is relatively larger for small distances from the electrode, as can be seen from the larger differences in shape for the streamline 1 mm away from the electrode compared with streamlines 3 mm and 5 mm away. In addition, more amplification is needed to capture the signal as the distance between the streamline and the electrode increases, accounting for the noisier signals as the bead is moved further away.

3.11 Removing the effects of common mode noise in cross correlation

3.11.1 Common mode noise on electrodes

In some situations, it may happen that the sensor electrodes are exposed to common mode noise. This can result, for example, as a consequence of strong physical vibration. In this case, correlation, $r[j]$, of signals X_1 and X_2 gives

$$r[j] = X_1 * X_2 \tag{3.25}$$

$$= \frac{1}{N}\sum_{n=0}^{N-1} X_1[n]X_2[n+j] \tag{3.26}$$

If, however, $X_1$and X_2 are contaminated with periodic noise $k[n]$, the actual correlation, $r_{noisy}[j]$ becomes:

$$\begin{aligned} r_{noisy}[j] &= \frac{1}{N}\sum_{n=0}^{N-1}\left[(X_1[n]+k[n])\,(X_2[n+j]+k[n+j])\right] \\ &= \frac{1}{N}\sum_{n=0}^{N-1}\left[X_1[n]X_2[n+j]+X_1[n]k[n+j]+k[n]X_2[n+j]+k[n]k[n+j]\right] \\ &= \frac{1}{N}\left[\sum_{n=0}^{N-1} X_1[n]X_2[n+j]+\sum_{n=0}^{N-1} X_1[n]k[n+j]+\sum_{n=0}^{N-1} k[n]X_2[n+j]+\sum_{n=0}^{N-1} k[n]k[n+j]\right] \end{aligned} \tag{3.27}$$

Looking at this term by term reveals:

$\frac{1}{N}\sum_{n=0}^{N-1} X_1[n]X_2[n+j]$ — the desired correlation

$\frac{1}{N}\sum_{n=0}^{N-1} X_1[n]k[n+j]$ — undesired, but harmless as it tends to 0

$\frac{1}{N}\sum_{n=0}^{N-1} k[n]\,X_2[n+j]$ — undesired, but harmless as it tends to 0

$\frac{1}{N}\sum_{n=0}^{N-1} k[n]\,k[n+j]$ —*problematic, because it is the autocorrelation of a periodic signal, which is itself periodic*

To solve this problem, consider the correlation of the first signal with the difference between the second and first signals, i.e.,

$$r_{diff} = X_1 * (X_2 - X_1) \tag{3.28}$$

Adding periodic noise, $k[n]$,

$X_1[n]$ becomes

$$X_1[n] + k[n]$$

and $X_2[n] - X_1[n]$becomes:

$$\begin{aligned}&(X_2[n] + k[n]) - (X_1[n] + k[n])\\ &= X_2[n] - X_1[n]\end{aligned}$$

The correlation, $r_{diff}[j]$ with periodic noise, becomes:

$$\begin{aligned} r_{diff}[j] &= \frac{1}{N}\sum_{n=0}^{N-1}\left[(X_1[n]+k[n])\,(X_2[n+j]-X_1[n+j])\right] \\ &= \frac{1}{N}\left[\sum_{n=0}^{N-1} X_1[n]\,X_2[n+j] - \sum_{n=0}^{N-1} X_1[n]\,X_1[n+j] + \sum_{n=0}^{N-1} k[n]\,X_2[n+j] - \sum_{n=0}^{N-1} k[n]\,X_1[n+j]\right] \end{aligned} \tag{3.29}$$

Expanding gives:

$\frac{1}{N}\sum_{n=0}^{N-1} X_1[n]X_2[n+j]$ — the desired correlation

$\frac{-1}{N}\sum_{n=0}^{N-1} X_1[n]X_1[n+j]$ — undesired, but tends to 0 at lags far from 0, since X_1 is a random signal

$\frac{1}{N}\sum_{n=0}^{N-1} k[n]X_2[n+j]$ — undesired, but tends to 0

$\frac{-1}{N}\sum_{n=0}^{N-1} k[n]X_1[n+j]$ — undesired, but tends to 0

Note that all undesired terms tend to 0. The correlation function begins with a strong negative autocorrelation of signal $X_1[n]$, but as long as the desired correlation peak occurs at a lag far enough from 0, it will not interfere with the desired correlation. *Therefore, the effect of common mode noise can be removed by correlating $X_1[n]$ with $X_2[n]-X_1[n]$.*

3.11.2 Common mode noise reduction example

The removal of the effect of common mode noise on cross correlation is illustrated in Figs. 3.27 to 3.30. A portion of the upstream and downstream signals captured from a 500 kW combustion testing facility (section 5.2) is shown in Fig. 3.27. The signals have then been contaminated with 500Hz noise, shown in Fig. 3.28. The effect of adding noise has changed the appearance of the raw signals slightly, but the effect when the signals are cross correlated (Fig. 3.29) is much more apparent. The correlation peak from the signals uncontaminated by noise has been completely obscured by the periodic correlation effects caused by the periodic noise in the signals. Correlating the signals using equation 3.28 removes the effects of the periodic noise on correlation (Fig. 3.30). The noise still has a detrimental effect, in that the correlation peak is reduced compared to the noise-free correlation, but the peak

is still clearly visible, making a velocity calculation possible. Clearly, it is better to work with signals not contaminated with common mode noise, but in cases where enough noise is present to disrupt a conventional correlation measurement, equation 3.28, can be used.

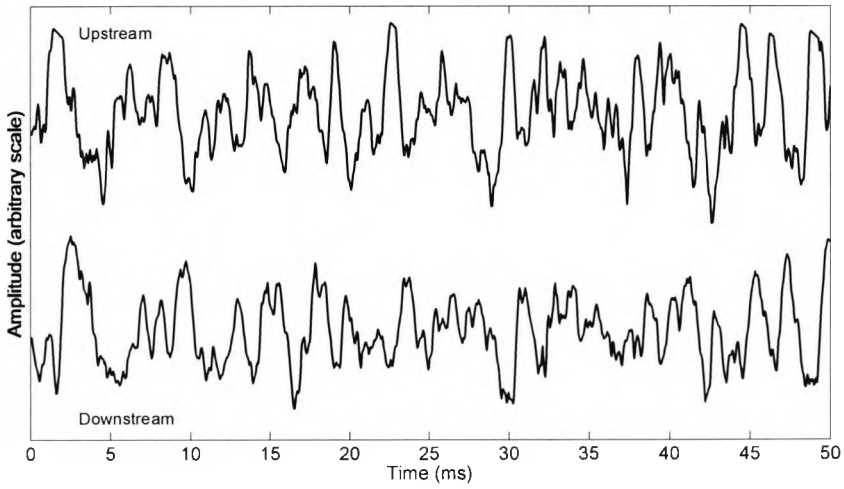


Figure 3.27: Common mode noise rejection example—original signals

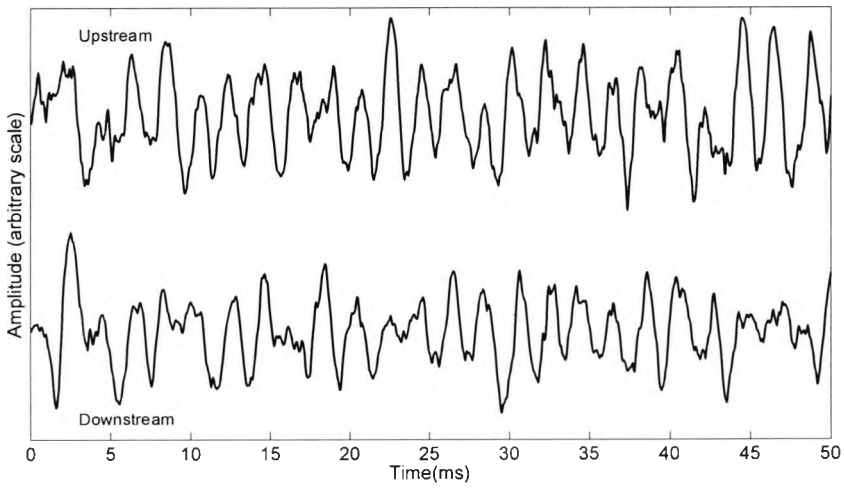


Figure 3.28: Common mode noise rejection example—noisy signals

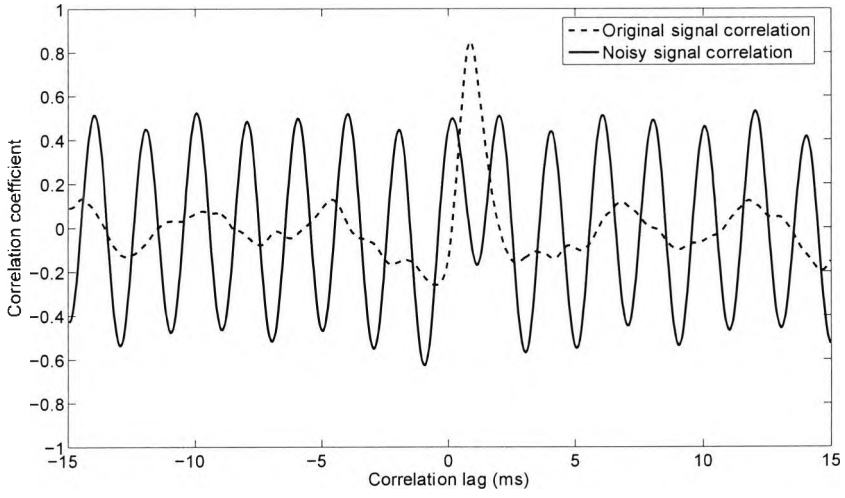


Figure 3.29: Noisy signal cross correlation

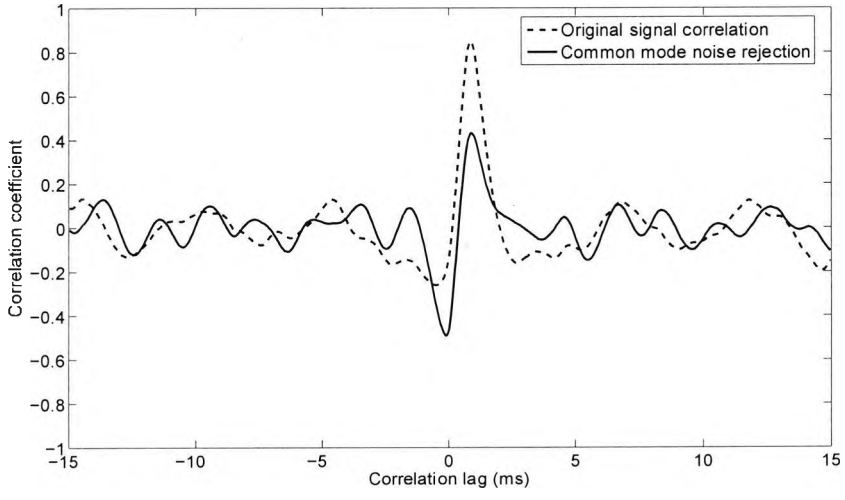


Figure 3.30: Cross correlation with common mode noise rejection

3.12 Summary

Modelling of the charge induced on an electrode in a metal pipeline by a charged particle flow is not a straightforward exercise. In anything other than a few classic configurations involving a high degree of symmetry, the electric field inside the pipeline, using Poisson's equation, has no known analytical solution, and an alternate method must be used. Finite element modelling provides a close approximation, and has been used to study the properties of intrusive rod electrodes for the measurement of velocity in pneumatic particle flows.

The modelling method was validated by comparing the FEM results of a non-intrusive circular ring electrode with those of an established analytical model. FEM has been used to develop an analytical expression, in the form of a relatively simple Lorentzian function, with which sensor characteristics can be studied. Both modelling and experimental results have demonstrated that the intrusive sensor is sensitive mainly to charge streamlines within the lateral 'line of sight' of the electrode. The modelling results have shown good agreement with the laboratory data. The models have been used to study the effects of non-uniform velocity profiles on the correlation velocity of electrostatic signals, and to provide a correction factor, depending on electrode length, for flows following a power law velocity profile. They have also been used to provide insight into the design of sensor electrodes with regard to cross sectional shape, and spacing.

Other aspects of the particle flow which can affect velocity measurement have been considered, such as the effect of turbulent conveying air, which was found to have a 'flattening' effect on the velocity profile, and periodic common mode noise, which was shown to capable of disrupting correlation velocity measurements. As a final note, a possible concern is that the presence a grounded conductor could induce forces on the charged particles themselves and influence their motion. According to Barlow [Barlow and Tinkle, 1999], however, this should not really play a part in industrial conveying, where the particle motion will be driven

mainly by the blower, and the electrostatic forces acting on the particles will be relatively small. In any case, the aim of the instrumentation discussed is to measure the velocity of the flow, and a slight deflection in direction would not influence this speed significantly.

Chapter 4

Electrostatic sensor design

4.1 Electrostatic sensor system overview

The following sections present a description of the electrostatic sensor system that was used to obtain the results described in chapter 5. It operates by capturing the signal produced by the fluctuating electric field due to particle motion in the pipeline and using analogue/digital processing techniques to yield a particle velocity measurement by cross correlation. The result can be recorded either in a data logging file or sent to a PC user interface.

The sensor can be divided into subsystems to manage the mechanical, signal acquisition and digital processing requirements. The mechanical details of the sensor depend on the pipeline geometry on which the sensor is to be installed, and are described in section 4.4. The analogue and digital systems, however, can be reused regardless of the particular pipeline dimensions.

The signal flow diagram for the complete sensor system is shown in Fig. 4.1. The signal induced onto the electrodes is in the form of a current (section 4.2.1), so it is first converted to

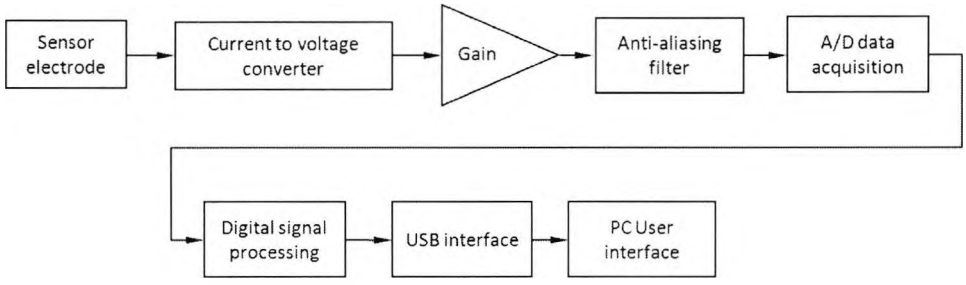


Figure 4.1: Electrostatic sensor flow diagram

a voltage in order that it can be sampled by an analogue to digital (A/D) converter. After digital sampling, cross correlation of upstream and downstream sensor electrodes is performed digitally on board a signal processing microcontroller, and the correlation function is used to calculate velocity. This is then communicated to a PC user interface via USB connection. The following sections describe these processes in detail.

4.2 Analogue signal conditioning

4.2.1 Electrostatic signal acquisition methods

The electrostatic signal induced on electrodes by passing charged particles is passive in nature in that it requires no external stimulus to create—charge is accumulated by the particles as a by-product of the pneumatic conveying process through air friction, collision with the pipeline wall and collision with other particles. However, the signal is weak and must be amplified by analogue electronics before it can be converted to digital form, ideally to a voltage range large enough to make full use of the available A/D converter resolution. There are several ways how this can be accomplished, which are discussed in the subsequent sections.

4.2.1.1 Voltage amplifiers

Perhaps the most obvious way to amplify the signal is to use a straightforward voltage amplifier. In this case, a high input impedance amplifier is used to detect the voltage induced on the electrode by a nearby charge. The solution of Poisson's equation:

$$\nabla^2\Phi = -\rho_q/\varepsilon_0 \tag{4.1}$$

with appropriate boundary conditions governs the voltage, Φ, induced on an electrode due to the presence of free charge, ρ_q, in a dielectric medium and where ε_0 is the permeability of free space. If the sensor electrode is connected to the input of an amplifier with a very high impedance, then this voltage can be amplified directly to obtain a voltage level appropriate to the A/D converter. This is the approach adopted by some researchers [Gajewski, 1984, 2006]. There are, however, several factors that must be taken into consideration when using this method.

Firstly, the voltage induced on the electrode is small, and the effective output impedance of the signal source is very high, as any current leaking from the electrode rapidly reduces the voltage induced by a charged particle (the particle has a certain amount of charge at its surface, and if current was allowed to flow from the electrode to ground, a state of equilibrium would be established where there would no longer be a potential difference between the electrode and ground). This means that the finite input impedance of a practical amplifier will degrade the signal by leaking away the voltage through the internal impedance of the amplifier. This is particularly true of relatively slow voltage fluctuations where the charge has more time to leak away, causing the electrode to act as a high pass filter. Also, the various input capacitances—including the capacitance from the sensor to the rest of the pipeline, cable capacitance and the input capacitance of the amplifier—must be considered in the electrode output. The effect of these capacitances has been analysed in detail [Gajewski, 1994],

but the calculations require that input capacitances are known in advance. In practice, this is difficult to do accurately and estimates must usually be made, with their associated errors.

4.2.1.2 Charge amplifiers

Charge amplifiers can be used in the measurement of electrostatic phenomena. They are used when a the physical output of a sensing device is in the form of a quantity of electric charge. An example of this is a piezoelectric sensor, where charge output is a function of the physical deformation of the sensor. Although the charge input is most commonly converted to a voltage, making it more correct to describe them as charge to voltage amplifiers, the term 'charge amplifier' is commonly used. The basic of a charge amplifier design is as shown in Fig. 4.2. C_{in} is not a circuit component, but represents the combined capacitance of the charge transducer and connecting wires. Assuming large opamp gain, the output voltage is independent of C_{in} and given [Weber, 2008] by:

$$V_{out} = \frac{q_{in}}{C_f} \tag{4.2}$$

In practice, the fact that an opamp requires a nonzero input current for its operation means that a resistor is usually placed in parallel with C_f. Otherwise, this current would cause C_f to accumulate charge, eventually causing the output to saturate. This type of amplifier is useful in situations where the charge input is relatively steady. However, when the input charge is a time dependent signal, $\frac{dq_{in}}{dt}$, the higher frequency components are subject to lower gain due to the reduced capacitor impedance at higher frequencies, resulting in a low pass filtering effect. This can be seen as follows:

Referring to Fig. 4.2, and differentiating equation 4.2 with respect to time,

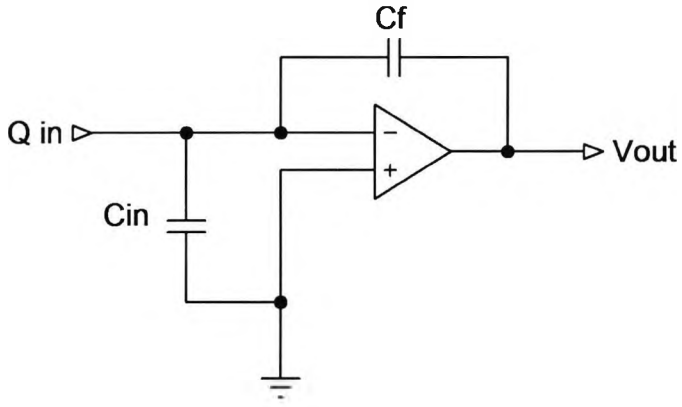


Figure 4.2: Charge amplifier

$$\frac{dV_{out}}{dt} = \frac{\frac{dq_{in}}{dt}}{C_f} \tag{4.3}$$

Using the fact that $\frac{dq_{in}}{dt}$ is a current, I_{in} Converting to the Laplace domain,

$$sV = \frac{I_{in}}{C_f} \tag{4.4}$$

$$V = \frac{I_{in}}{sC_f} \tag{4.5}$$

This can be recognised as an integrator, or low pass filter, and causes both amplitude and phase distortion of continuous time inputs. Since the charge induced by a varying electric field is a continuous function of time, using a charge amplifier as described above would distort the signal. A solution is to use a current to voltage amplifier, discussed in the following section.

4.2.1.3 Current to voltage amplifier

In an electrostatic sensor, charge is not directly transferred to the electrode, but induced by the electrostatic field created by the passing charges. The use of the term *electrostatic* should be clarified at this point. Since the charges on the passing particles are in motion, the consequent electric field is continually changing, and it would seem more appropriate to refer to a electro*dynamic* rather than an electro*static* system. However, the *static* part of the term reflects the principle that if the system was 'frozen' at any given instant, the distribution of charges in the system would be accurately reflected in the charge induced on the electrode. In other words, the assumption is made that the electric field generated by the charges in the system propagates instantaneously to the rest of the system.

If the distance between the particles and the sensor was very large, this would not be a reasonable assumption, and the propagation time of the changing electric field (at the speed of light), would have to be considered. However, the distances involved in the sensor are small relative to the speed of light, so the time delay between a change in charge configuration in the pipeline and the corresponding rearrangement of induced charges on the electrode is negligible. It is in this sense that the system is *static*, and there is no need to introduce a relaxation time constant in measuring the response of the induced charge distribution due to a change in the surrounding charge configuration.

The signal induced onto the electrode, however, is time dependent, as it responds to the continually changing charge configuration in the pipeline. This change of charge with time, $\frac{dq}{dt}$, can be considered as an input current. To produce an output in the form of a voltage, a current to voltage (transresistance) amplifier must be used, with units of volts/amperes.

A particular advantage of using a transresistance amplifier is that the capacitance from the sensor to the pipeline, cable capacitance and amplifier input capacitance can be ignored. This is shown in the following analysis, with reference to Figs. (4.3) and 4.4:

Let Z_t represent the parallel impedances of R_s (the impedance of the Thevenin equivalent current source), C_s(the sum of the stray capacitances from the electrode to the input amplifier), R_{in} and C_{in} (the input resistance and capacitance of the opamp).

Then, using Kirchhoff's current law at the V_{in} node,

$$I_{in} = \frac{V_{in}}{Z_t} + \frac{V_{in} - V_{out}}{R_f} \tag{4.6}$$

Substituting , $V_{in} = \frac{-V_{out}}{A_0}$ into equation (4.6) , where A_0 is the open loop gain of the opamp, gives:

$$I_{in} = \frac{\left(\frac{-V_{out}}{A_0}\right)}{Z_t} + \frac{\left(\frac{-V_{out}}{A_0}\right) - V_{out}}{R_f} \tag{4.7}$$

Solving for V_{out} gives:

$$V_{out} = \frac{A_0 R_f Z_t I_{in}}{A_0 Z_t + R_f + Z_t} \tag{4.8}$$

By inspection of (4.8), it is clear that for high values of A,

$$V_{out} = R_f I_{in} \tag{4.9}$$

which is independent of the input impedance, Z_t. Typical opamp open loop gains are of the order $10^5$and higher, so equation (4.9) is reasonable to assume for practical circuits. A less formal, but more intuitive, analysis would be to realise that since the V_{in} node is kept at virtual ground, no current is lost through the input resistances or capacitances, which can therefore be ignored.

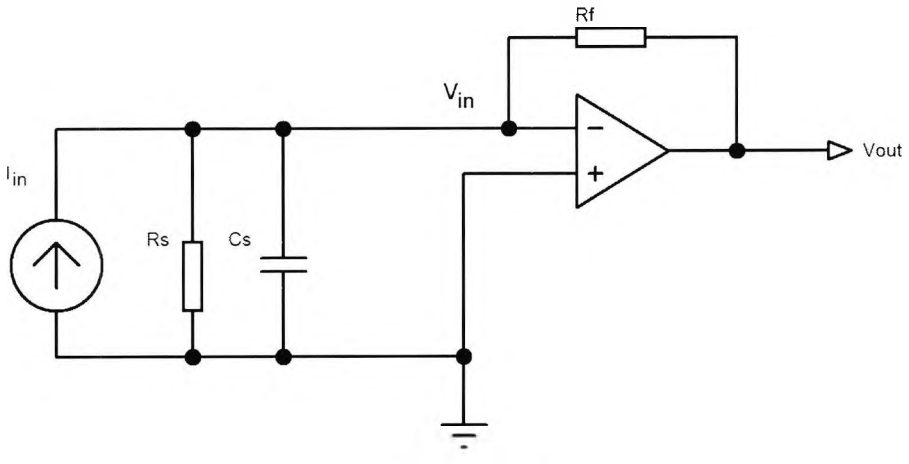


Figure 4.3: Current to voltage amplifier

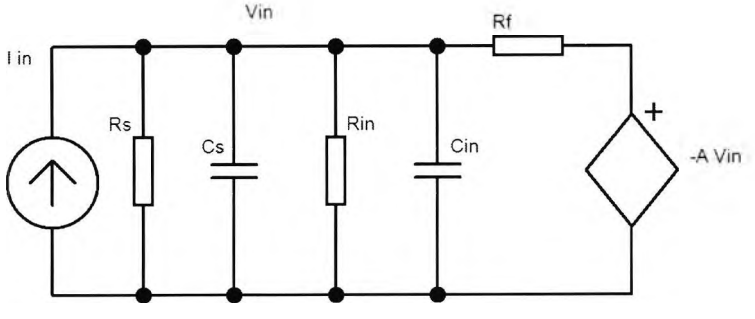


Figure 4.4: Current to voltage amplifier—equivalent schematic

4.2.2 Sensor amplifier design

For signal acquisition in the system constructed for this research, the sensor electrode is kept at a steady reference voltage. As the electric field fluctuates around it due to charged particle flow, the surface charge on the electrode fluctuates, maintaining the reference voltage.

This fluctuating charge—i.e., current—is converted to a voltage using a current to voltage amplifier as described in section 4.2.1.3.

Although this method of signal acquisition does not require consideration of stray capacitances in the system, it is still sensitive to the motion of the connecting cable. Even tiny displacements of charge in connecting cables due to physical handling can produce currents large enough to cause significant noise in the system. When a sensor such as this is installed, the connecting cables may no longer be subject to motion, but they would still be susceptible to external vibrations. In an embedded system, however, such as the one constructed for the present research, the signal processing is performed on board, and the connecting cable consists only of a short internal connecting wire from the electrodes to the input of the analogue electronics, making it immune from this kind of disturbance.

4.2.2.1 Induced signal amplifier requirements

The mass flow pneumatically conveyed coal pipelines is typically around 70 kg/h, with charge densities of 10^{-3} C/kg to 10^{-7} C/kg. This means that, expressed as a current, there is on the order of nanoamps to a few microamps flowing in the pipeline. Only a fraction of this current is induced on the electrode, as seen from the modelling results (section 3). The A/D converter (section 4.3.2) used to digitise the results has an input range of 0 V to 3.3 V. This means that a large transresistance amplifier is required in order to make full use of the available A/D resolution.

In addition, the A/D converter works for positive voltages only, so the amplifier output voltage is required to swing about the midpoint of the converter range, i.e., 1.65 V. Finally, the power to drive the circuit is taken from a USB connection from the sensor to the PC, which provides a single supply voltage source of 5 V. Therefore, a single supply opamp circuit is necessary.

4.2.2.2 Current to voltage stage

The first stage of sensor amplifier consists of the current to voltage conversion. For a large transresistance, however, an undesirably large value of feedback resistance (R_f) and/or several subsequent voltage gain stages would normally be necessary to achieve the required mount of gain. Early circuit experiments using two or more subsequent gain stages proved to be unstable, producing spurious oscillation at times. The solution was to replace the feedback resistor, R_f, with a 'T' network of resistors in the basic circuit of Fig. 4.3. The transresistance (V_{out}/I_{in}) of the 'T' network circuit can be derived using nodal analysis, with reference to Fig. 4.5.

From Kirchhoff's current law at node V_2,

$$\frac{V_1 - V_2}{R1} - \frac{V_2 - V_{ref}}{R_2} + \frac{V_o - V_2}{R3} = 0 \tag{4.10}$$

Also,

$$V_{out} = A_0(V_{ref} - V_1) \tag{4.11}$$

where A_0 is the open loop gain of the opamp.

Finally, assuming high opamp input impedance (i.e., negligible input current),

$$V_1 - V_2 = I_{in}R_1 \tag{4.12}$$

Equations (4.10), (4.11) and (4.12), can be combined into the matrix equation:

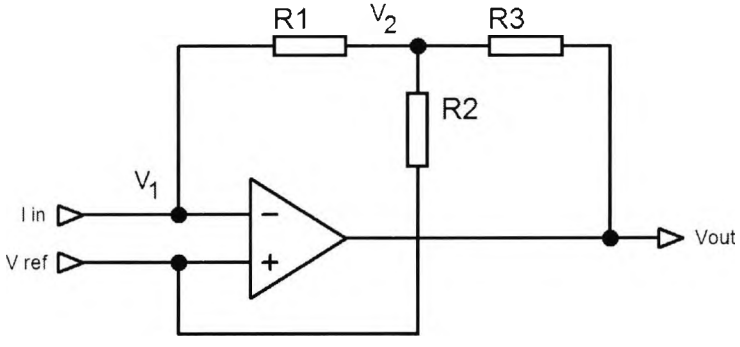


Figure 4.5: 'T' network feedback

$$\begin{bmatrix} \frac{1}{R_1} & -\left(\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}\right) & \frac{1}{R_3} \\ A & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_{out} \end{bmatrix} = \begin{bmatrix} \frac{-V_{ref}}{R2} \\ AV_{ref} \\ I_{in}R_1 \end{bmatrix} \tag{4.13}$$

Solving (4.13) for V_{out} , and assuming a very high opamp gain ($A \rightarrow \infty$) results in:

$$V_{out} = -\frac{I_{in}(R_1R_2 + R_1R_3 + R_2R_3)}{R2} + V_{ref} \tag{4.14}$$

Choosing $R_1 = 47k\Omega$, $R_2 = 470\Omega$, and $R3 = 47k\Omega$ in equation (4.14) gives a transresistance of 4.8 mV/nA, which was found to be sufficiently sensitive for the coal flows used in the practical trials (chapter (5)). Without the 'T' network, either a feedback resistor of 4.8MΩ would be necessary to achieve equivalent gain, or more gain stages would have to be used. Either way would result in risk of instability and increased amplifier noise [Wu, 2004], which should be avoided for such low input current. Also, the assumption of opamp input impedance high enough to be neglected is valid only if the feedback resistance is much lower than the input impedance.

In the actual amplifier constructed for the sensor, a small capacitor, $C_1 = 10\,pF$, was added

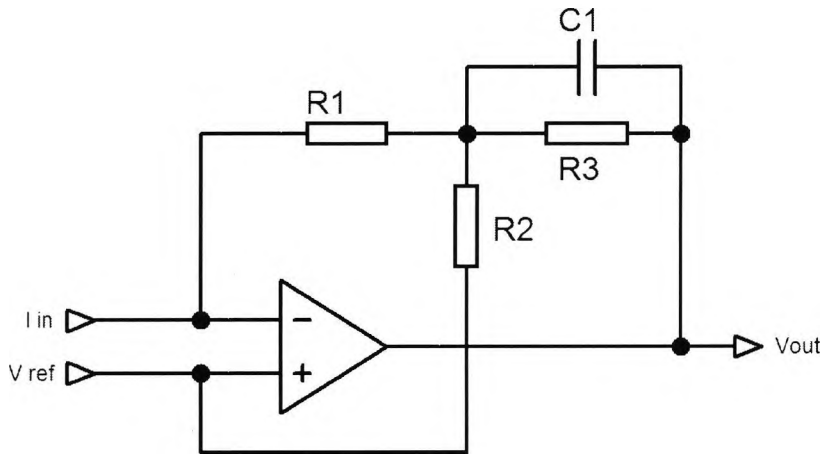


Figure 4.6: 'T' network feedback with high frequency rolloff

in parallel with R_3, as shown in Fig. 4.6. This creates a pole at $\omega = 1/C_1R_3$, in order to roll off the gain at high frequencies, increasing the stability of the amplifier.

4.2.2.3 Voltage amplification stage

After current to voltage conversion, the signal is still relatively weak and needs further voltage amplification before it can be presented to the input of the A/D converter. However, the total charge in the pipeline is dependent on many variables [Armour-Chelu et al., 1998, Armour-Chelu, 1998, Armour-Chelu and Woodhead, 2002, Woodhead et al., 2005], and different amounts of gain are required, depending on flow conditions, to make full use of the available A/D converter resolution. Also, in industrial applications, the sensor may have to be mounted in places with limited access, making manual gain control on the sensor itself inconvenient. A remote gain control was incorporated into the sensor design to alleviate this problem. A 'digital resistor' (Dallas Semiconductor, DS1844) was used, and accepts digital inputs to control the analogue resistance between its input/output terminals. The PC user interface is used to send the desired resistance value to the microprocessor via USB, which

then sends the appropriate bit sequence to the digital resistor. This way, the voltage gain can be controlled from the user interface either directly by the user or automatically through software, in order to keep the signal power within an acceptable range.

The voltage amplification stage must amplify the varying current signal from the particle flow, but the DC offset must remain midway between the supply voltage and ground in order to be usable by the D/A converter. In other words, the AC component of the signal needs further amplification, but the DC gain must remain at unity. This can be achieved using the circuit of Fig. 4.7. Here, capacitor C_1 ensures unity gain at DC voltages. Its relatively high value (22 μF) means that although DC gain is constant at unity, the AC gain is within 1dB of the maximum gain at frequencies above 15Hz—low enough to avoid distortion of the lower frequency components of the signal.

A more commonly used method of AC coupling is to place the capacitor between the output of the previous stage and the input of the stage under consideration. In this case, however, the high capacitance value needed to allow low frequencies would cause unwanted interaction with the previous stage. Also, an additional voltage offset would have to be applied in order that the zero signal reference voltage is maintained at the midpoint of the supply voltage. Placing the capacitor in the feedback loop as shown avoids both of these problems.

4.2.2.4 Anti-aliasing filter

A low pass, anti-aliasing filter is required in order not to exceed the Nyquist frequency of the sampling device. The frequency spectrum of the signal induced on the electrodes was found to be approximately 3 kHz or less. Therefore, a filter with a cutoff frequency of 9.9 kHz was used, in order to allow the full range of the signal through without attenuation, but suppressing high frequency noise. In order to avoid aliasing, the sampling frequency should exceed twice the cutoff frequency of the filter. In practice, considering that the actual

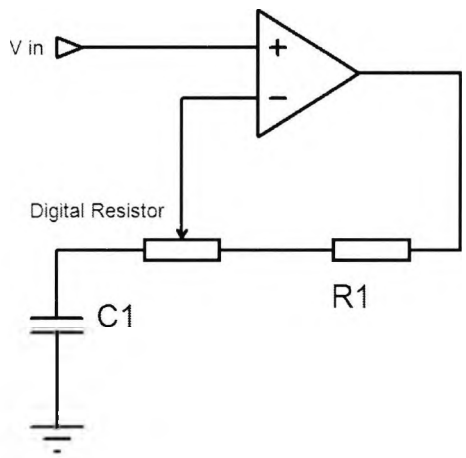


Figure 4.7: Variable AC gain stage

signal frequency is less than about 3 kHz, slightly lower sampling rates can be acceptable, since any aliasing will be due to the small amount of high frequency noise generated in the amplifier circuit. This noise is random in nature, and the cross correlation technique used in subsequent digital processing (section 4.3) removes the effects of such noise. The filter is implemented in a standard Sallen and Key topology seen in Fig. 4.8. The sampling rate used for industrial trials was 25 kHz, but in the laboratory 15 kHz often produced more stable results.

4.3 Digital signal processing

4.3.1 Sensor DSP overview

After analogue signal conditioning has produced a signal appropriate for sampling, it is digitised and processed using a dsPIC® microcontroller, which has features enabling it to perform A/D signal conversions and fast DSP operations. From this point onward, all processing

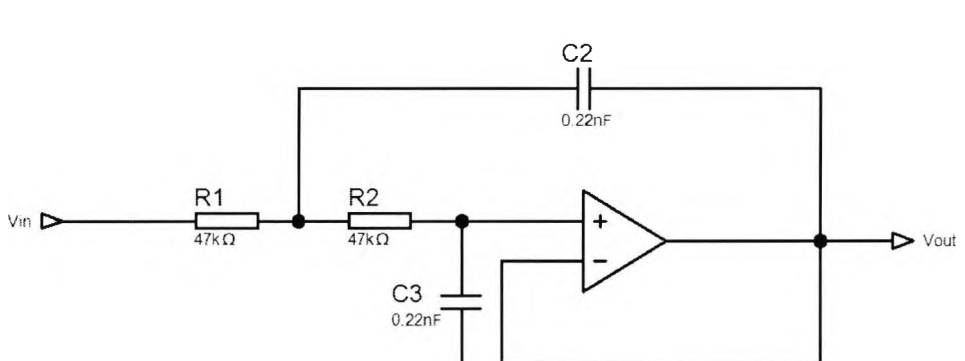


Figure 4.8: Anti-aliasing filter

is performed digitally on board the microcontroller. The following sections detail the specific steps used in processing the analogue signal to derive the velocity measurement, which is output to the user interface.

4.3.2 Microcontroller description

Digital sampling and processing of electrode signals, in real time and at low cost, has only recently become available. For minimal cost (less than £10), it is possible to digitise and process analogue signals within a single integrated chip. Previous to this, these operations had to be performed using discrete and relatively expensive instruments. Some signal processing such as cross correlation, now done conveniently using digital techniques, often had to be done in the analogue domain [Mavor et al., 1977].

The particular device used in the sensor design is the dsPIC (33FJ128GP306), from Microchip®. Sampling rates of up to 1.1 MSPS are possible, in 10 bit resolution, with simultaneous sampling of up to four channels. This, together with its maximum processing speed of 40MIPS and a built in UART interface, makes it suitable for real time electrostatic sensor signal pro-

cessing. Programmability via a 'C' language compiler makes it relatively straightforward to implement both high level programming constructs as well as low level control of data movement to make optimal use of on-chip resources.

4.3.3 Microcontroller program structure

A 'C' program developed for the dsPIC is used to perform the signal processing. It uses the A/D peripheral module to convert the analogue signal to digital form. The internal processor is then used to perform the cross correlation of the upstream/downstream signals and, subsequently, to find the time lag corresponding to the location of the correlation peak. From this, the correlation velocity can be calculated ($\frac{electrode\ spacing}{time\ lag}$). Finally, the UART module is used to communicate with the PC user interface. Flowcharts showing a simplified structure for the program are shown in Figs. 4.9 to 4.11

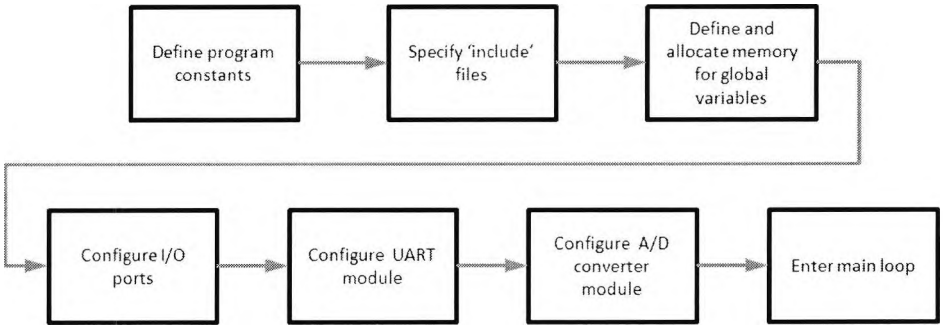


Figure 4.9: Program initialisation flowchart

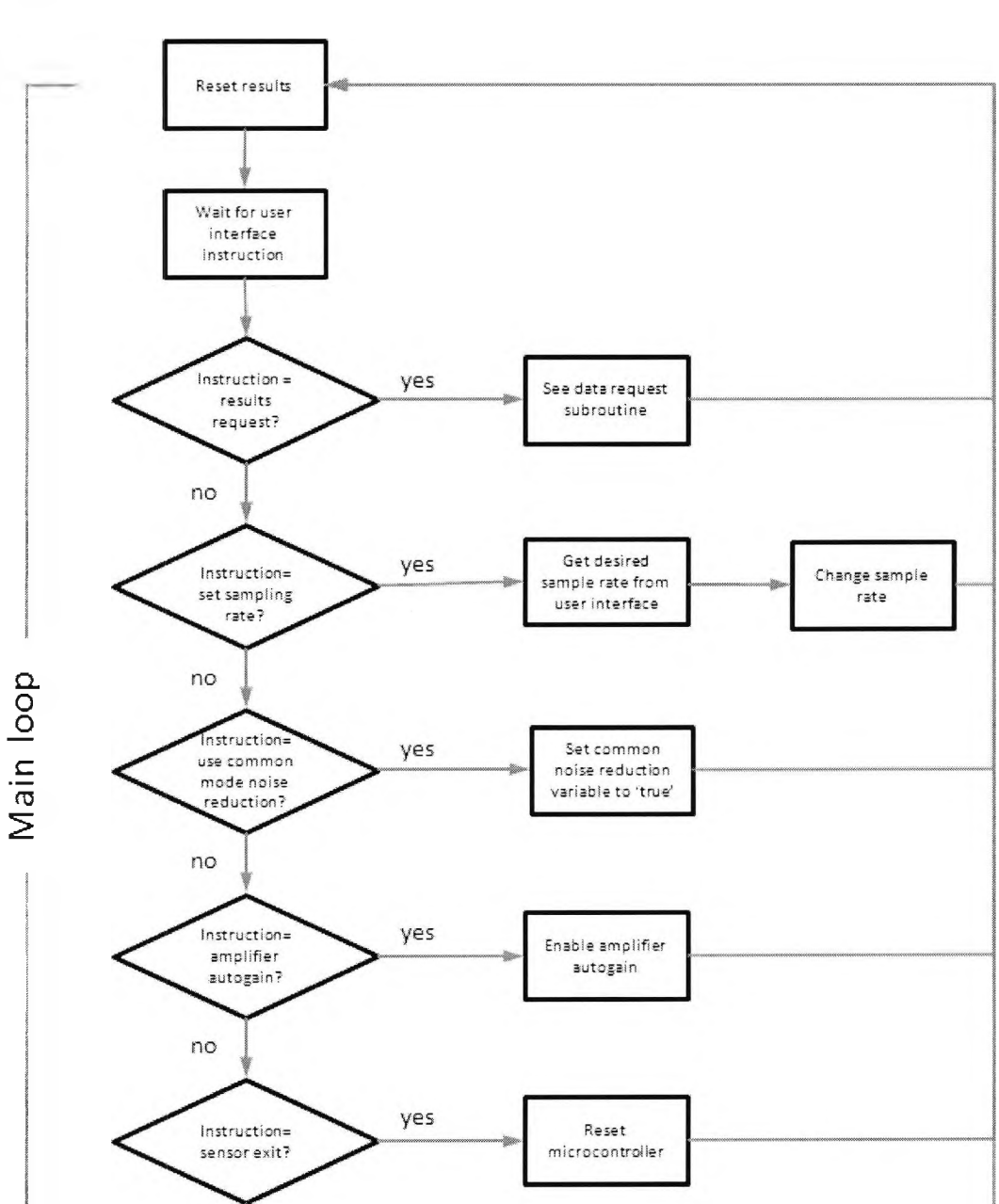


Figure 4.10: Signal processing flowchart—main loop

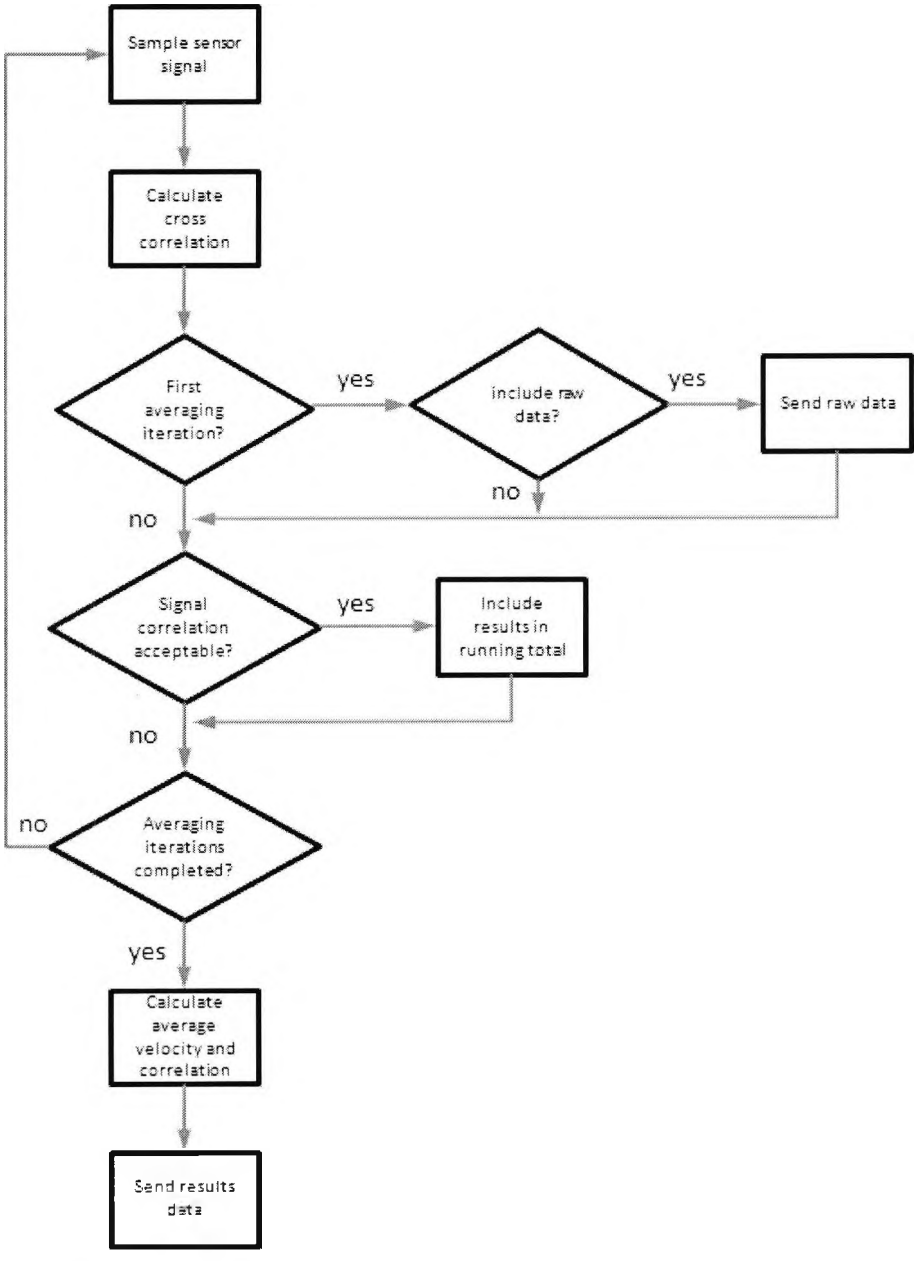


Figure 4.11: Signal processing flowchart—data request subroutine

4.4 Mechanical design

4.4.1 Introduction

The mechanical design is more dependent on the particular pipeline setup on which the sensor is to be installed than the electronic and digital processing subsystems. Whilst the electrical/digital design can be reused in a wide range of electrode/pipeline configurations, the mechanical design must be adapted to the particular test setup being used, with the pipeline diameter being the most important. The face of the sensor must be made to roughly match that of the inside of the pipeline in order to avoid unwanted exposure to the particle flow, and the length of the sensing electrodes must be scaled to the appropriate length in order to acquire signals from the appropriate cross sectional location.

Practical results are fully described in section 5, but the tests were carried out on pipelines with internal diameters of 40 mm 48 mm, 300 mm and 500 mm. Therefore, hardware setups to suit these pipeline dimensions were constructed. The same hardware was used for the 300 mm and 500 mm sections due to the relatively small difference in internal curvature. The 40 mm and 48 mm pipelines are also similar, but a different hardware setup was used on the 48 mm section to make use of insights drawn from the results from the 40 mm diameter pipeline tests.

4.4.2 Sensor for 40 mm diameter pipelines

A 40 mm internal diameter pipeline was used for tests at the 500 kW combustion test facility in Didcot, UK (section 5.2). The pipeline was easily accessible, and it was possible to insert a

spool piece in line with the flow. The possibility of using a spool piece enabled a sensor with 4 sensing heads to be constructed, orientated symmetrically around the pipeline, as shown in Fig. 4.12. This allowed velocity monitoring not only for varying electrode intrusion depth, but also around the circumference of the pipeline itself, to detect asymmetries in the particle flow.

The electrodes were constructed of 1.6 mm diameter stainless steel rod embedded into PVC insulators for electrical isolation. The other components were made of aluminium for ease of machining. The function of the aluminium housing components is not only to provide mechanical positioning and stability, but also to screen the signals induced on the probes from external electrical noise, which is always present from room lighting, mains power, etc.

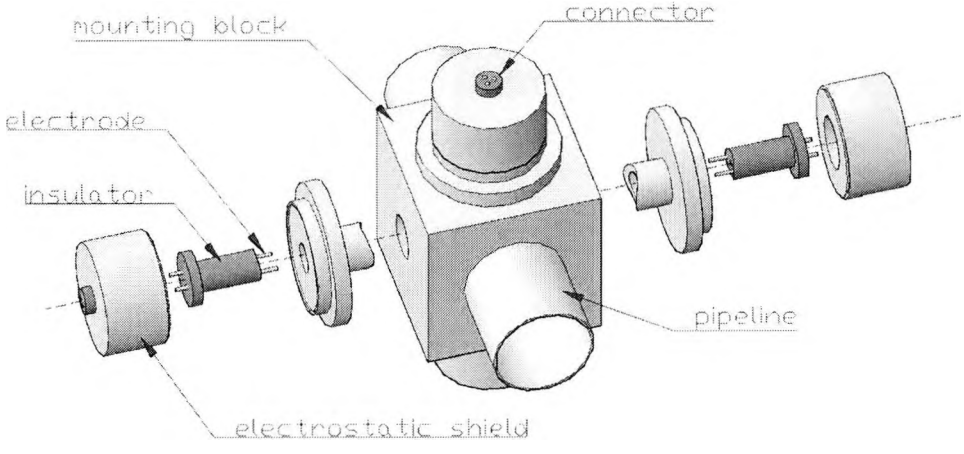


Figure 4.12: Sensor for 40 mm diameter pipeline

4.4.3 Sensor for 300 mm/500 mm diameter pipelines

The design for 300 mm/500 mm pipelines was tested at the PCME test center. The large diameter of the pipeline means that it was not critical for the curve of the face of the sensor to

exactly match that of the pipeline interior. In fact, the sensor face consists of two angled planar sections that follow the internal curvature of the pipeline as closely as possible. Ideally, the face of the sensor would be flush with the pipeline wall, but the relatively small size of the sensor means that small differences in curvature will represent an insignificant distortion of the wall. For this reason, only one sensor was made for both the 300 mm and 500 mm pipelines. The design is shown in Fig. 4.13.

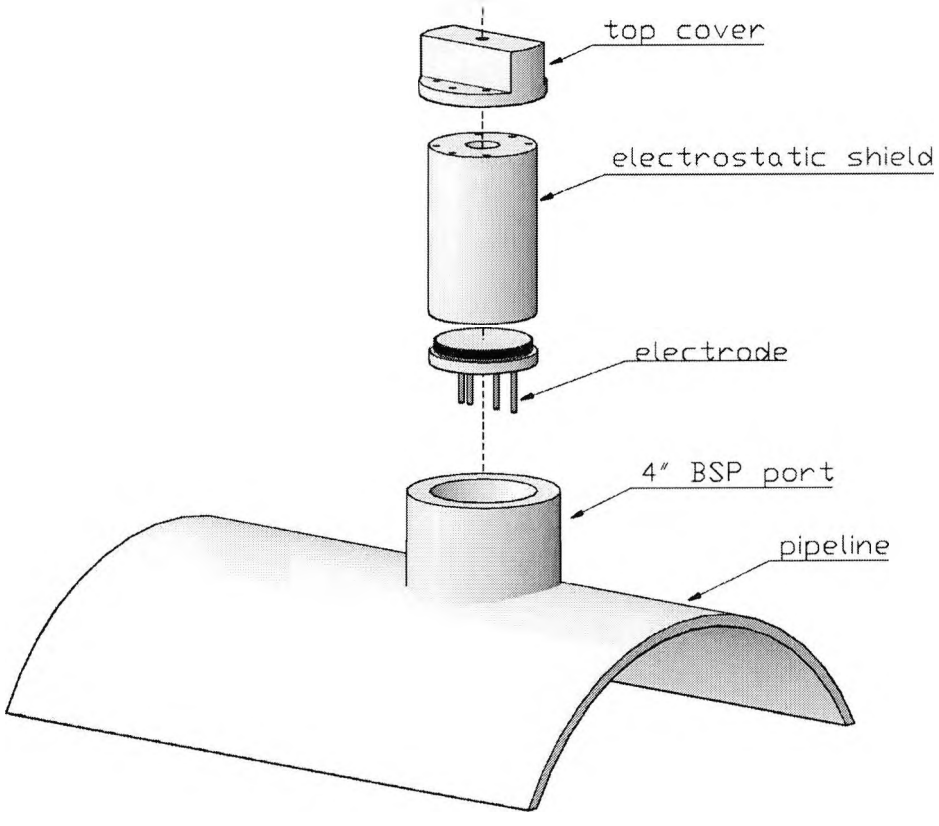


Figure 4.13: Sensor for 300 mm/500 mm diameter pipelines

4.4.4 Sensor for 48 mm diameter pipelines

The test pipeline at the University of Kent has an internal diameter of 48 mm. It is designed for sections to be easily removed and rotated, so only one pair of electrodes is necessary. The sensor can be rotated around the pipeline if necessary, to monitor flow conditions around the pipeline. The design can be seen in Fig. 4.14.

Another feature of the lab sensor is an additional port on the pipeline, opposite the sensor. This can be used to insert a hot wire anemometer, in order to measure a reference air velocity. The delicate nature of the anemometer means that only empty pipeline air velocity can be measured (section 5.4.2). Air velocity is slightly lower for a given suction pump setting when the flow is loaded with solids, but it is a useful reference for comparison with particle flows.

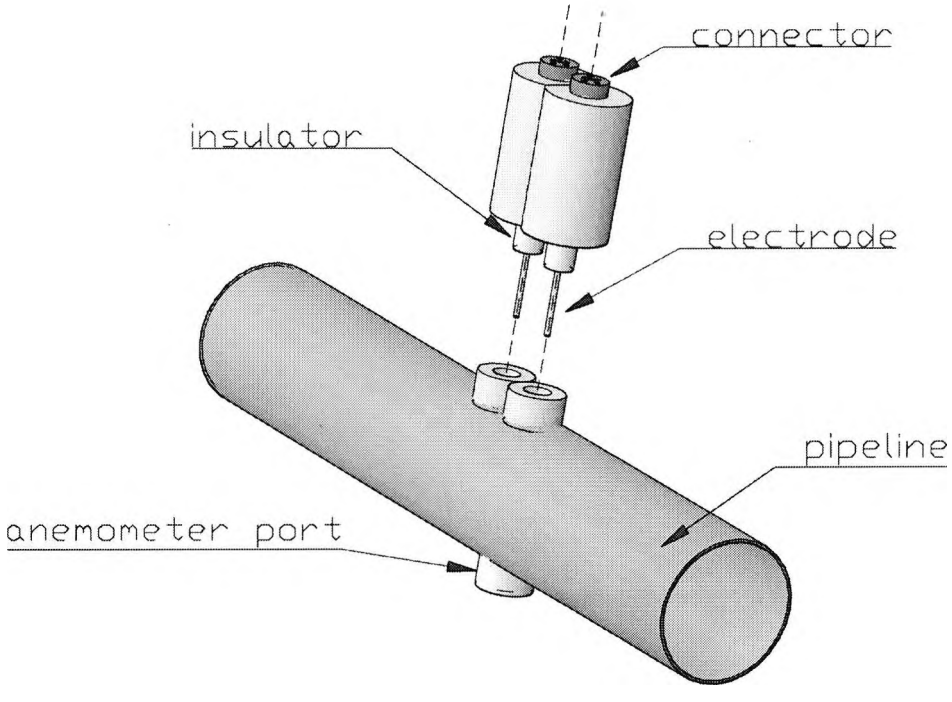


Figure 4.14: Sensor for 48 mm diameter pipeline

4.5 Summary

The hardware and signal processing software was designed to be appropriate for use with the available test rigs in order to obtain experimental results. The weak signals induced onto sensor electrodes are processed with analogue electronics that perform a current to voltage conversion, before being sampled and processed by an embedded dsPIC® microcontroller. The correlation velocity is derived using digital techniques, and is subsequently communicated to a PC. Whilst the electronic design is reusable regardless of the physical sensor configuration, the hardware is specific to a particular installation setup, and three different hardware configurations were constructed.

Chapter 5

Experimental results

5.1 Introduction

An electrostatic sensor fitted with intrusive electrodes was implemented in several electrode configurations, and installed on several different test rigs. Identical signal conditioning electronics (section 4.2.2) were used, but gain in each case was set to an appropriate level depending on the magnitude of the charge induced onto the electrodes by the particle flow. Tests were conducted on a small scale industrial coal combustion test facility (section 5.2), on an instrument manufacturer's pneumatic flow test rig (section 5.3) and on the pneumatic particle flow setup at the University of Kent (section 5.4). Additional tests were conducted on a larger scale 4MW combustion test facility by a colleague, in order to obtain practical results with which to compare intrusive and non-intrusive designs.

The purpose of the tests was to evaluate the response of intrusive electrode configurations in real world conditions involving flows with different particle concentrations, in pipelines of different sizes, with different materials and using different electrode intrusion depths. This way, flow parameters that make a significant difference to sensor output could be identified,

along with those that make little or negligible difference to performance. The relative characteristics of intrusive rod electrodes and a non-intrusive circular ring electrodes were also investigated.

5.2 500 kW combustion test facility tests

5.2.1 Background

This section presents details of tests carried out at the RWE npower 0.5 MW combustion test facility (CTF) in Didcot, UK. These were part of a Department of Trade and Industry Project in collaboration with RWE npower (Project Number 10064, *An Integrated Sensor System for Combustion Plant Optimisation*).

The purpose of the tests was to evaluate the performance of the intrusive electrostatic design in near-industrial conditions. Although simulation and preliminary laboratory tests suggested that intrusive electrodes, together with the signal conditioning electronics and digital processing, should provide a reliable velocity measurement, there were several unknowns, including sensor response to materials such as pulverised coal which cannot be used in the laboratory setup.

As previously discussed (chapter 2), the overall charge in the pipeline is in difficult to predict. As such, the signal strength induced onto the electrodes is not known in advance, and the appropriate gain setting for the amplifying electronics becomes a matter of trial and error. The correct gain for maximum signal to noise ratio appropriate for the lab setting is no guarantee that it will be appropriate for flows in other test setups, or a full scale industrial burner. A digitally controlled. variable gain stage was designed into the electronics (section 4.2.2.3) to allow some room for adjustment but, even so, the gain range can cover only a

limited range of operating conditions. Also, for the trials, bare stainless steel electrodes were used. Given the abrasive qualities of a pneumatically conveyed pulverised coal flow, it was not known if the flow would cause measurable wear on the electrodes over the test period of several days.

5.2.2 Sensor setup for 500 kW test rig

A 4-head electrostatic sensor was set up on the CTF in order to assess its performance using realistic PF materials, under realistic flow conditions (Fig. 5.1). The sensor was designed to house electrodes of varying intrusion depth, so that the effect of intrusion could be studied. In addition, the 4 sensing heads allowed flow conditions at equally spaced points around the pipeline perimeter to be monitored (Fig. 5.2). The distance from the last pipeline bend to the sensor was not precisely measured, but is at least 10 pipeline diameters in length.

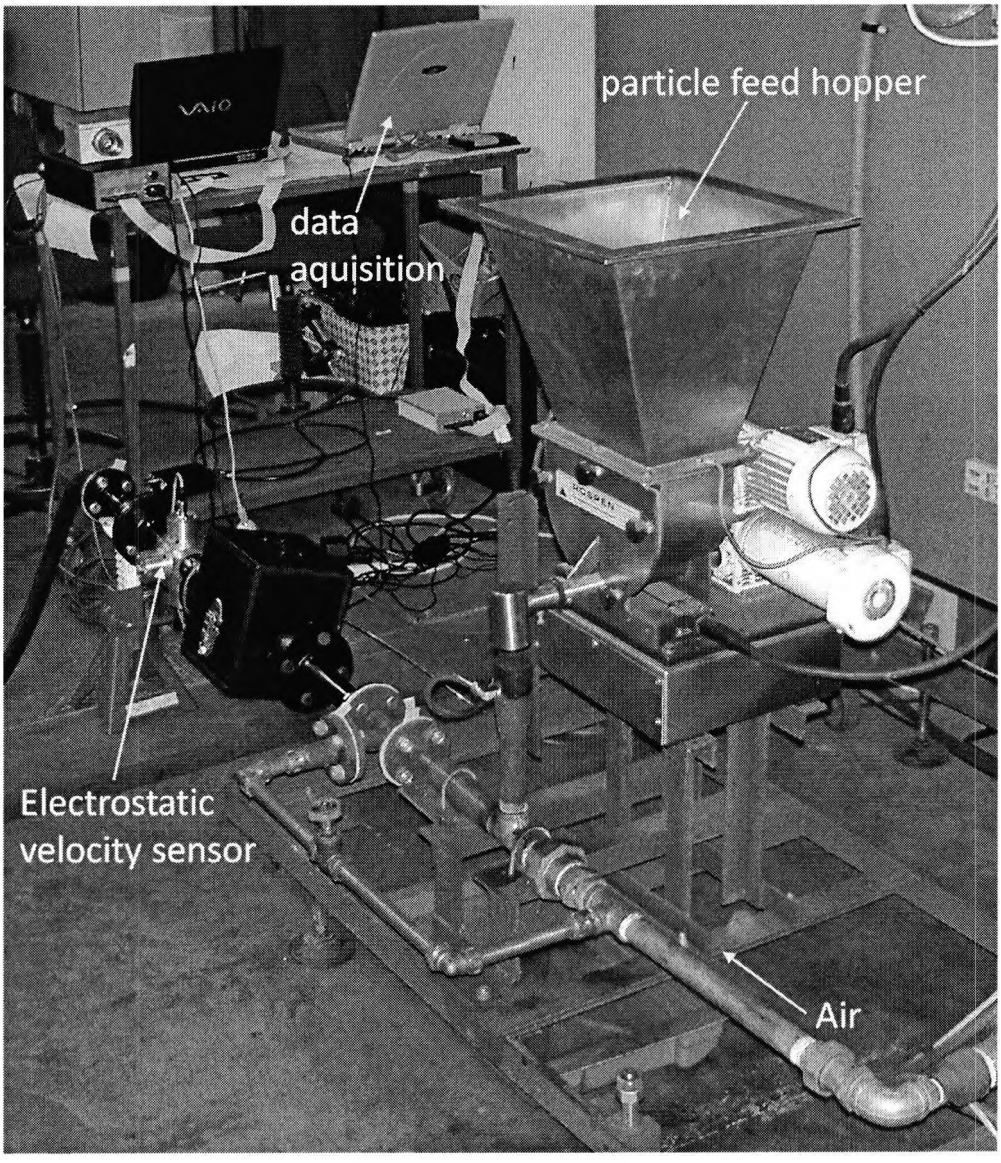


Figure 5.1: 500 kW flow rig feeder

The sensing heads were made of aluminium, and spaced at 90^o intervals around the pipeline. The sensor electrodes (1.6 mm diameter stainless steel rod) were embedded into PVC inserts to provide electrical isolation from the main body of the sensor. The assembly was then mounted onto a 300 mm long, flanged steel pipeline with an internal diameter of 40 mm,

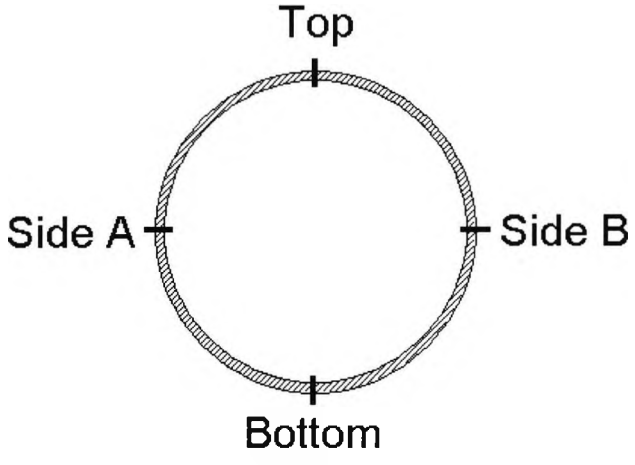


Figure 5.2: Sensing head locations (looking upstream)

making it compatible for use with the CTF test rig. Fig. 5.3 shows the entire assembly. The cross sectional area taken up by the electrode was at most 2.5% (at the maximum electrode intrusion depth of 20 mm), and was considered to have a negligible on the flow.

The primary materials tested were coal (Optimum Middleburgh), rice flour and biomass (wood chips). Each material was tested at high/low mass flow rates of 5 kg/h and 10 kg/h, and at high/low conveying air speeds of 15 m/s and 25 m/s in the air feed pipeline. Note that the airspeed indications, while providing a convenient and repeatable reference, do not give the actual airspeed in the electrostatic measurement section. This is due to the fact that the air feed pipeline, where air velocity was measured, was not the same size as that of the pipeline on which the electrostatic sensor was installed and, in addition, the airflow meter was not near the electrostatic test section.

5.2.3 500 kW test rig results

Results indicate that there is a significant difference in sensor performance, in terms of velocity output fluctuation and peak correlation, when using different materials and sensor

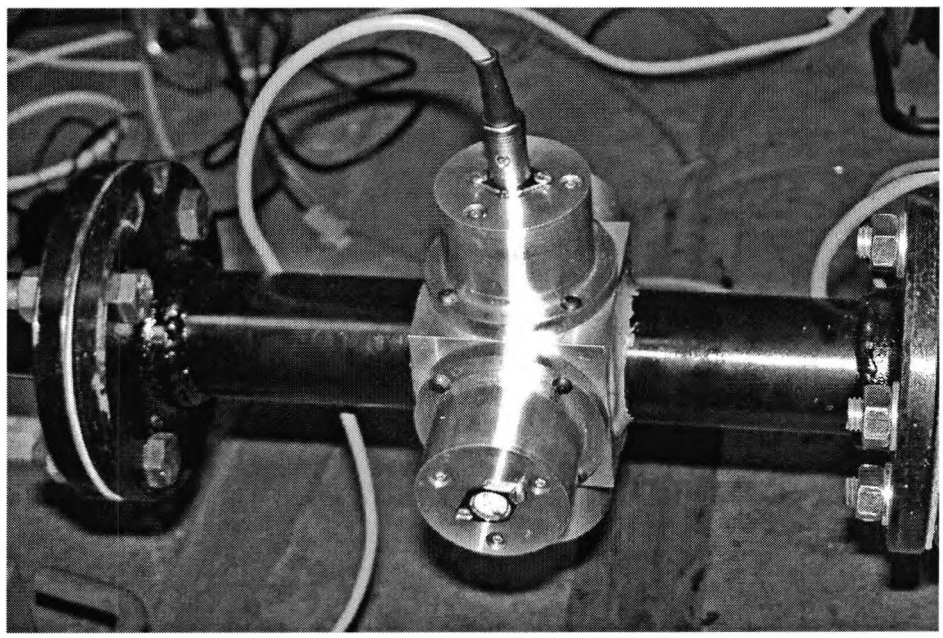


Figure 5.3: Photograph of sensor head assembly

intrusion depths, and less performance variation for different mass flow rates and air speeds. As will be seen, coal and flour produced similar results, both giving much stronger signal strengths, peak correlation and velocity fluctuation than for signals from wood chips. Increasing sensor intrusion improved signal power and peak correlation for all materials. Better performance was also achieved with higher mass flow rates and conveying air speeds. However, the differences observed with different flow rates and air speeds were not as great as those between materials and electrode intrusion depth.

5.2.3.1 Biomass (wood chips)

Figs. 5.4 and 5.5 show typical results for the velocity measurement of wood chips. The electrode intrusion was set to 20 mm, and the mass flow rate and air velocity in the air feed pipeline were 10 kg/h and 15 m/s respectively. The results from this setup are generally poor. Visually, the downstream signal should be similar to a slightly delayed version of the

upstream signal (i.e., similar to the upstream signal but shifted over to the right by several milliseconds). Any similarity, however, is barely evident. This is reflected in the low and irregularly shaped cross correlation coefficient peak of around 0.36. The correlation velocity history for 100 measurements, shown in Fig. 5.6, varies significantly, although for practical use a running average of the results could be used to smooth the results.

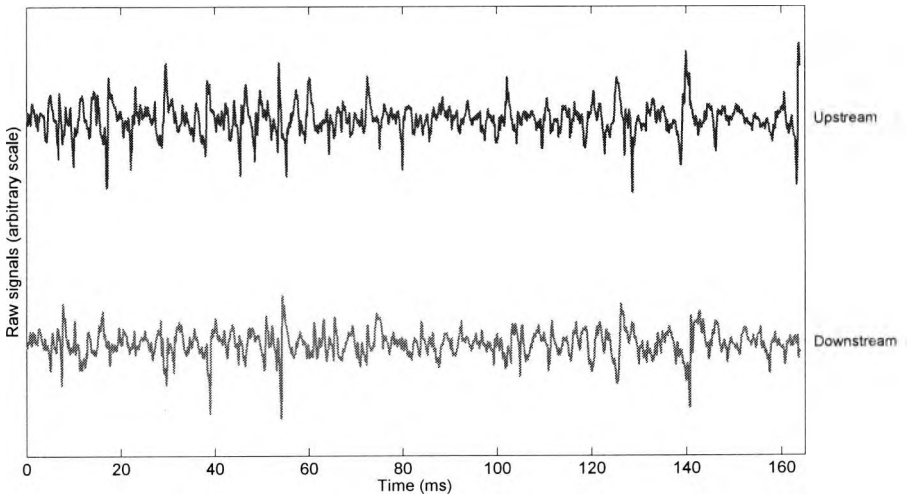


Figure 5.4: Induced charge signals—wood chips

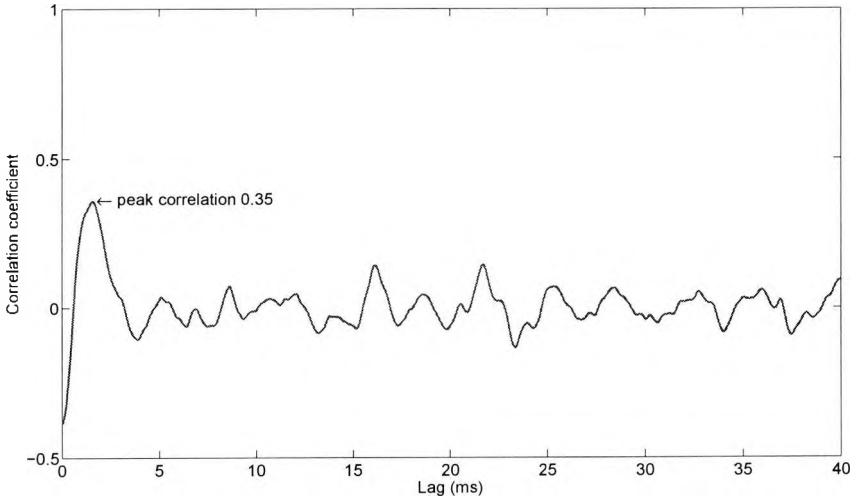


Figure 5.5: Upstream/downstream signal cross correlation—wood chips

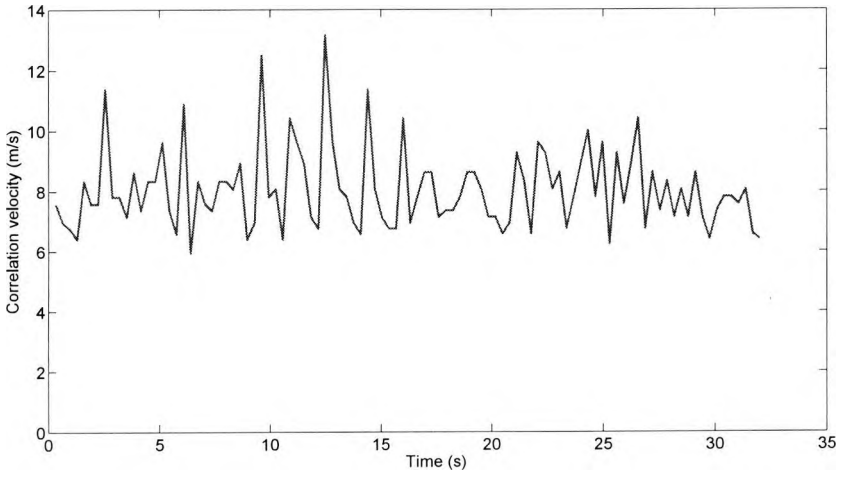


Figure 5.6: Correlation velocity history—wood chips

5.2.3.2 Rice flour

The rice flour results are considerably better (Figs. 5.7 to 5.9). The flow parameters and sensor setup are the same as for the previous case, but there is much more similarity between upstream and downstream signals as well as a strong, well defined cross correlation peak. Also, the velocity results are much more stable than those for wood chips. Unfortunately, rice flour had a tendency to 'clump' together in the particular screw feeder setup used, often resulting in sporadic results, and no further tests were conducted with it in this set of trials.

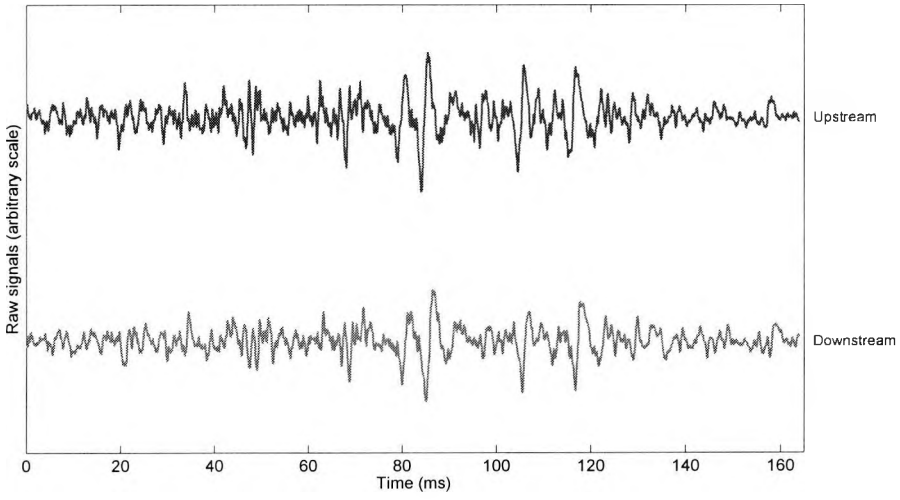


Figure 5.7: Induced charge signals—rice flour

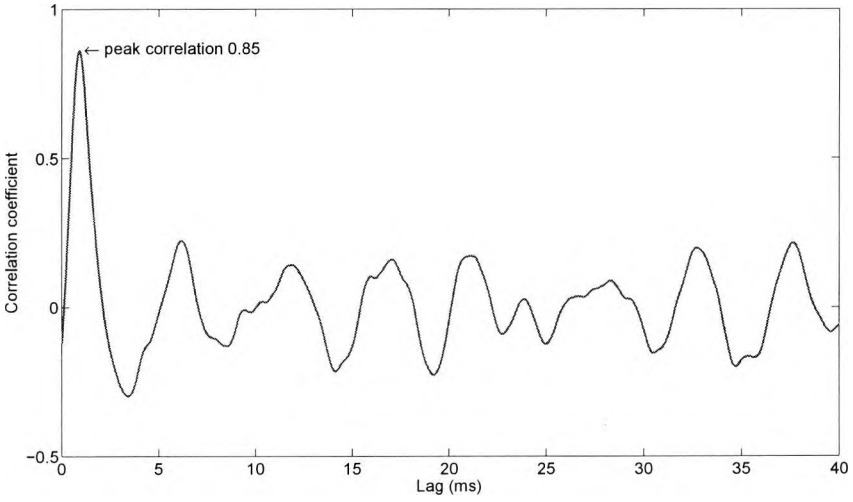


Figure 5.8: Upstream/downstream signal cross correlation—rice flour

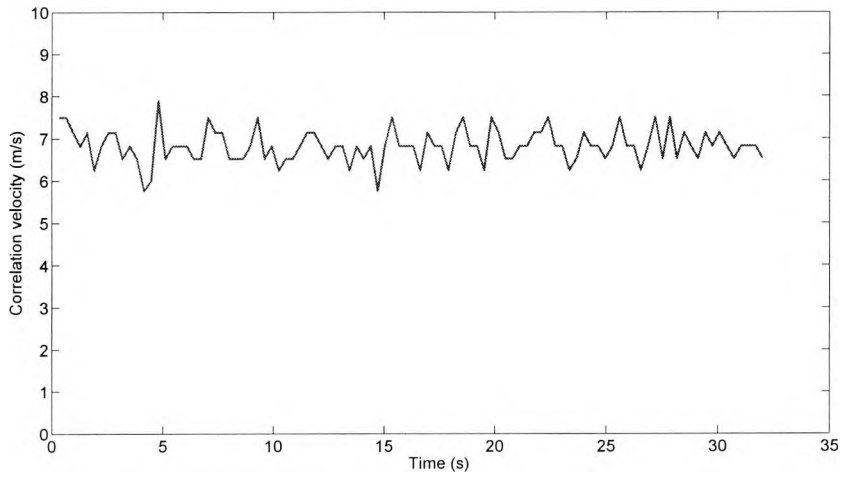


Figure 5.9: Correlation velocity history—rice flour

5.2.3.3 Pulverised coal

Typical results from the coal, with flow parameters as above are shown in Figs. 5.10 to 5.12.

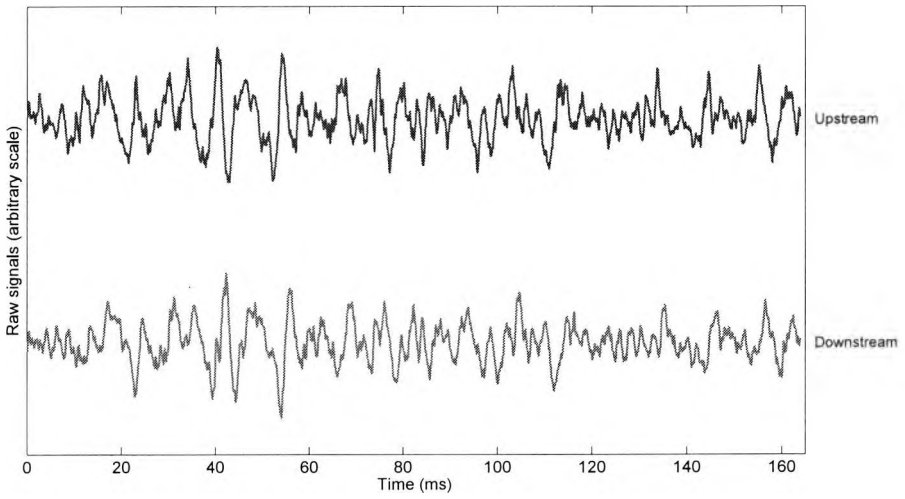


Figure 5.10: Induced charge signals—coal

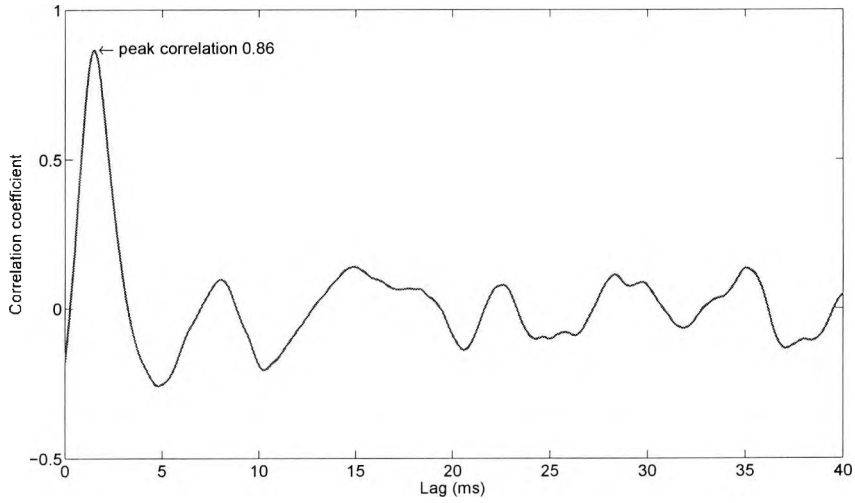


Figure 5.11: Upstream/downstream signal cross correlation—coal

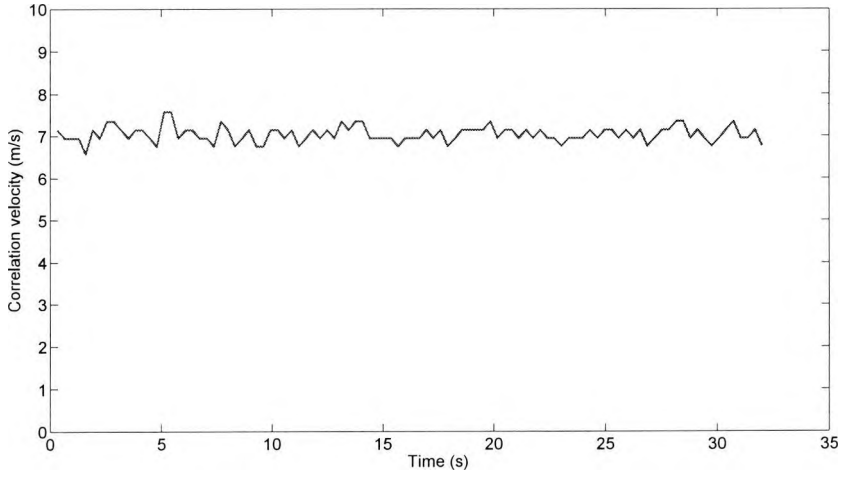


Figure 5.12: Correlation velocity history—coal

The coal produced the best results in terms of the correlation peak value, which is likely due to the small particle size (mean size approx. 50 μm). Additional trials for both coal and wood chip tests were conducted for electrode intrusion depths of 0 mm, 5 mm, 10 mm and 20 mm at the 4 sensor locations around the pipeline perimeter (Fig. 5.2), with 100 readings taken in each case.

5.2.3.4 Velocity measurement fluctuation

The difference in wood chip and coal velocity measurement is evident from Figs. 5.13 and 5.14. Here, the standard deviation of velocity results for the two materials is compared for different sensor electrode intrusion depths. The wood chip velocity results vary by approximately 10% to 23% of the mean value, depending on sensor electrode intrusion depth, compared to a variation of about 3% to 5% for coal. Also, sensor electrode intrusion is more significant for wood chip velocity measurement, with deeper intrusion resulting in more stable readings. Intrusion depth made less difference in the coal velocity measurement—there

is a clear increase in velocity measurement stability as electrode intrusion is increased from 0 mm to 5 mm, but increasing intrusion further makes little difference in the variation of the results.

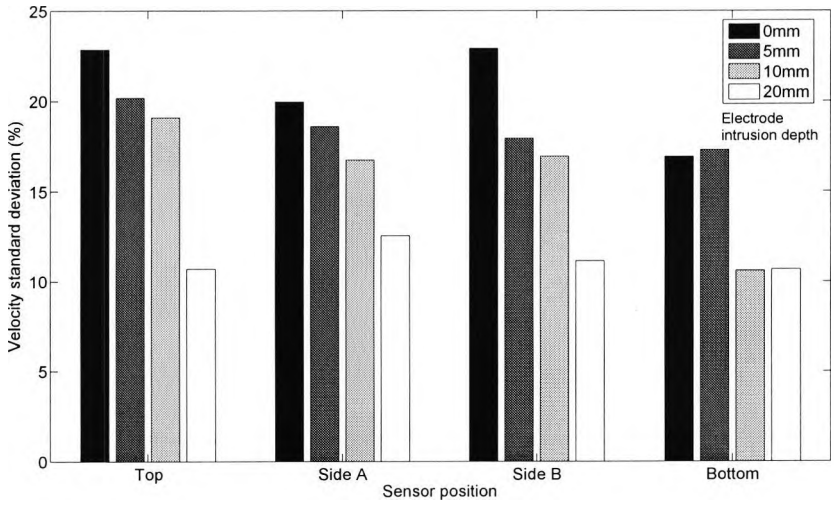


Figure 5.13: Correlation velocity standard deviation—wood chips

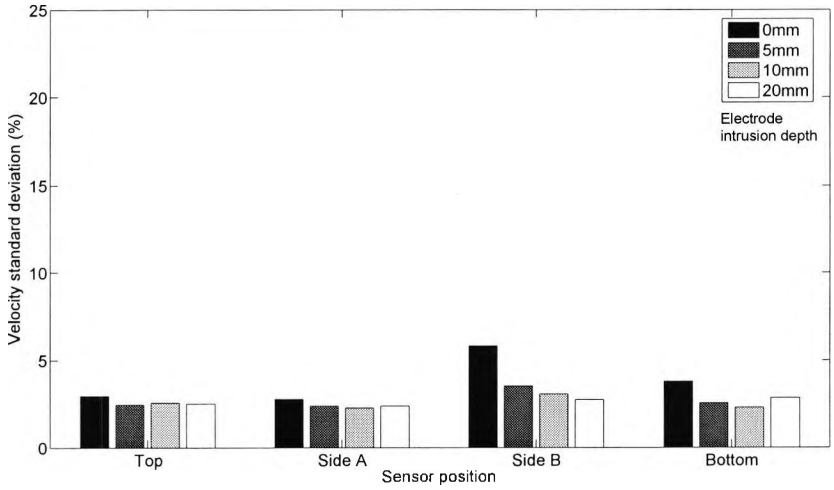


Figure 5.14: Correlation velocity standard deviation—coal

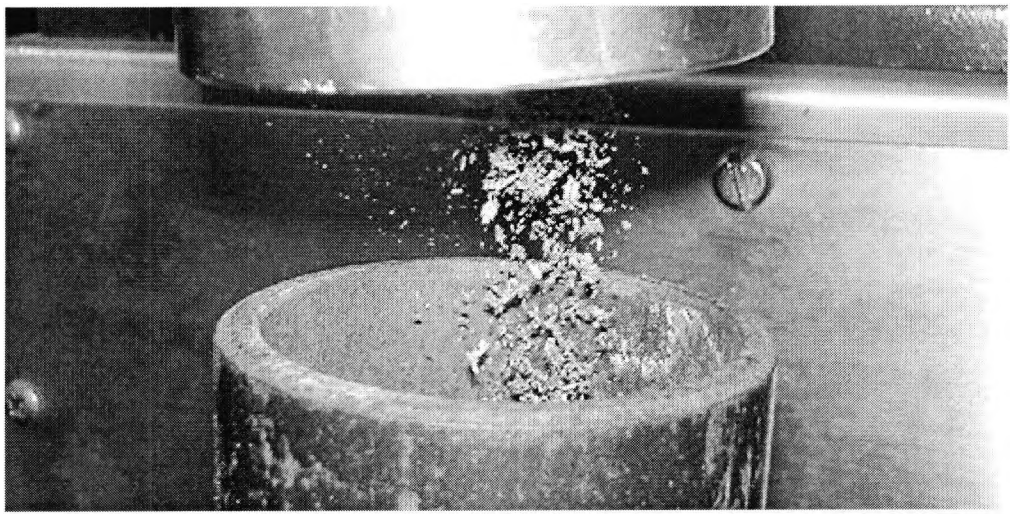


Figure 5.15: Wood chips leaving the feeder

It is suspected that the shape of the coal and wood chip particles accounts for much of the difference in the sensor's velocity measurement performance. The coal particles are much smaller and closer to spherical in shape, and any spin on the particles as they travel down

the pipeline is unlikely to significantly affect the signal induced on the sensor electrodes. However, wood chip particles are more irregularly shaped (Fig. 5.15), some being relatively long and thin. This asymmetry means that any spin on these particles is more likely to affect the induced signals, since the orientation of a particle as it passes the first probe will, in general, be different to its orientation as it passes the second probe, and the resulting upstream/downstream signals can be expected to vary significantly.

Due to the high level of variation and low correlation peaks in wood chip velocity measurement and the clumping properties of rice flour, only the coal results were considered in further analysis. Figs. 5.16 to 5.19 compare correlation velocity measurements for different electrode intrusions depths. Most noticeable is the fact that the non-intrusive electrode (0 mm plot) gives a consistently higher velocity reading. The velocity results for the other electrode depths are less clearly separated, although the deepest intrusion (20 mm) seems to give the slowest measurements. The fact that the non-intrusive electrode is most sensitive to particles travelling near the pipeline wall makes this an unexpected result, as it is contrary to the general velocity profile of a fluid, where the velocity tends to decrease towards the pipeline wall. The spacing of the electrodes was verified in order to ensure that this observation was not due to errors in spacing during sensor construction. Some variation (max. 5.8%) from the design specification of 10 mm was found, and the velocity data has been scaled proportionately, but this does not completely explain the phenomenon, since the trend of slower particles towards the center of the pipeline remains even after scaling.

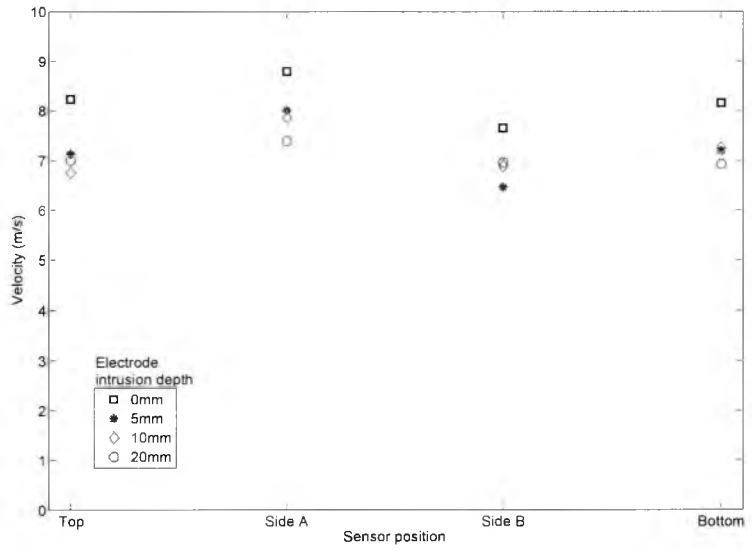


Figure 5.16: Coal correlation velocity vs Sensor position, mass flow rate 5 kg/h, air velocity of feed pipe 15 m/s

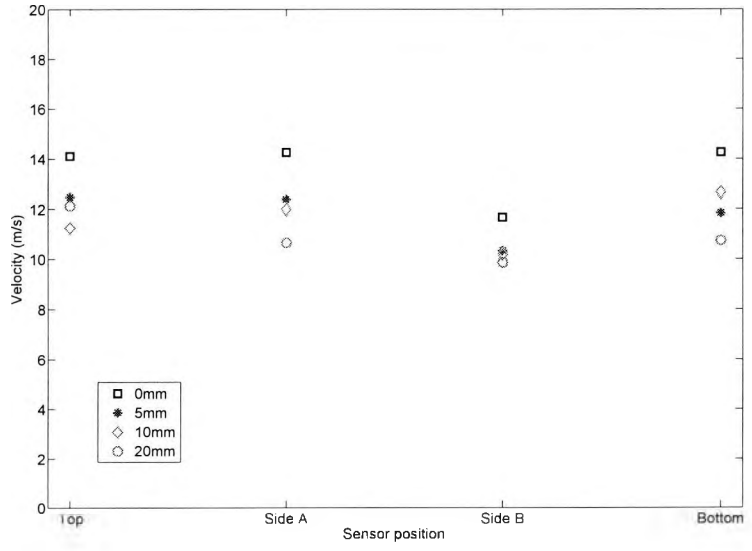


Figure 5.17: Coal correlation velocity vs Sensor position, mass flow rate 5 kg/h, air velocity of feed pipe 25 m/s

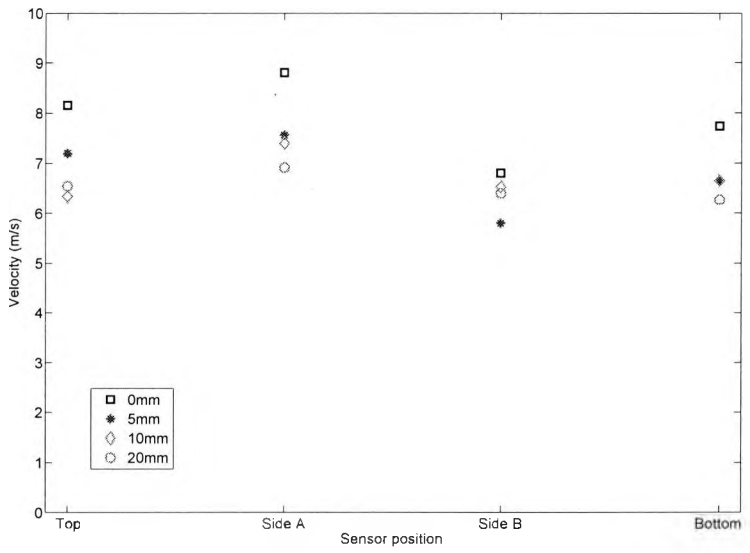


Figure 5.18: Coal correlation velocity vs Sensor position, mass flow rate 10 kg/h, air velocity of feed pipe 15 m/s

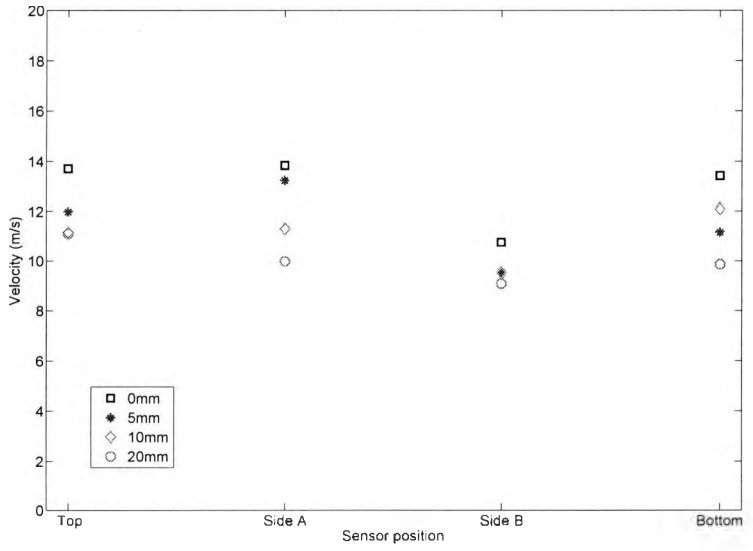


Figure 5.19: Coal correlation velocity vs Sensor position, mass flow rate 10 kg/h, air velocity of feed pipe 25 m/s

It is now thought that the close spacing of the sensor electrodes may have been distorting the sensing field, making them appear to be more closely spaced than in reality, and accounting for the higher correlation velocity near the pipeline wall. Modelling has revealed that this is indeed a contributing factor, as discussed in section 3.8. There it was seen that this effect is most serious for shallow intrusion depths, and that increasing sensor spacing reduces this phenomenon.

Another insight can be found in the work of Li and Tomita [2000], where particle motion is studied optically for swirling flows. The particles used were relatively large (averages of 3.13 mm to 4.26 mm), and were conveyed by air with varying degrees of swirling flow. Overall, the localised particle velocities (calculated from high speed camera images) were roughly uniform regardless of the amount of swirl, but in some conditions the particle velocities were, in fact, greater near the pipeline wall, demonstrating that the velocity field for particle flows does not necessarily follow that of the conveying air.

There is also a noticeable difference in correlation velocity depending on sensor location. The velocities at side 'B' (Fig. 5.2) are generally lower, than at side 'A'. This result is consistent regardless of mass flow rate and conveying air velocity. Each point is the average of 100 trials, so it is highly unlikely that this velocity difference is due to random sample variation. In addition, there is no bend in the pipeline near the measurement section to cause disruption to the flow, i.e., the particle speed variation was present along a straight run. Upon disassembly, however, the flanged section connecting the measurement section to the rest of the pipeline was found to be slightly misaligned. This means that on one side of the pipeline particles close to the wall will have collided with the joint between the pipeline sections, reducing their velocity. This is a clear example of the localised sensing field of the electrodes suggested by earlier by modelling (chapter 3), and demonstrates that particle speed variation around the circumference of the pipeline can be detected.

Figs. 5.20 to 5.23 compare the coal correlation velocity measurements for the different

electrode intrusion depths used at the two mass flow rates (5 kg/h and 10 kg,h) and air supply velocities (15 m/s and 25 m/s). The correlation velocity trends at the sensor locations are broadly similar for the different intrusion depths. Higher mass flow rate has the expected effect of slightly reducing the particle velocity (except with 5 mm electrode intrusion at side 'A', where the velocity is higher for the higher mass flow).

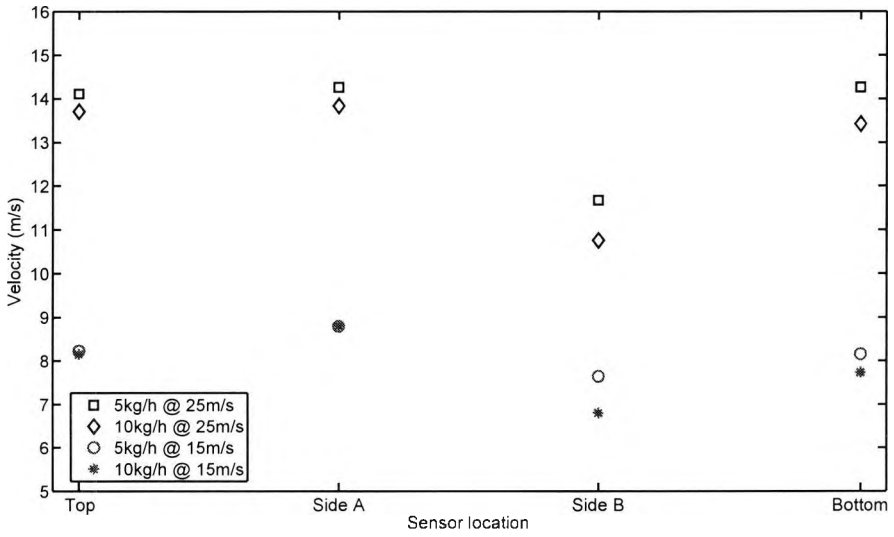


Figure 5.20: Coal correlation velocity—0 mm electrode intrusion

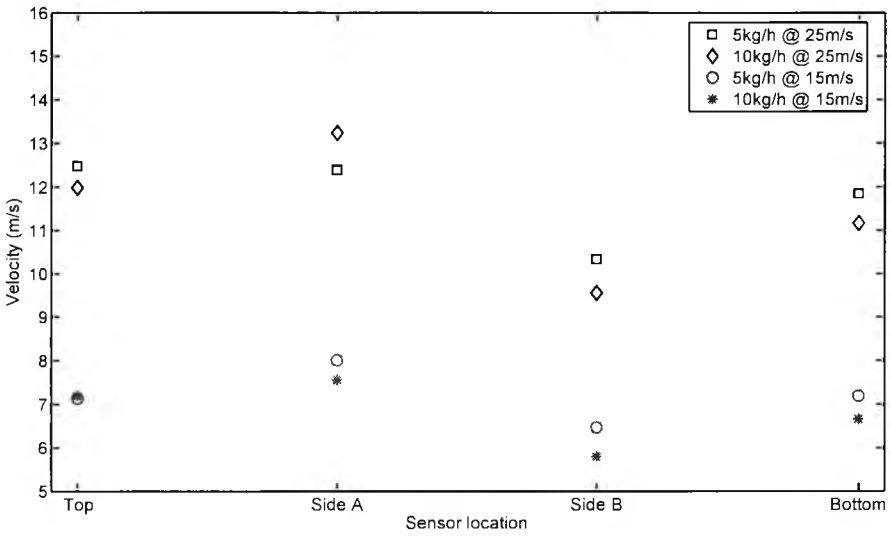


Figure 5.21: Coal correlation velocity—5 mm electrode intrusion

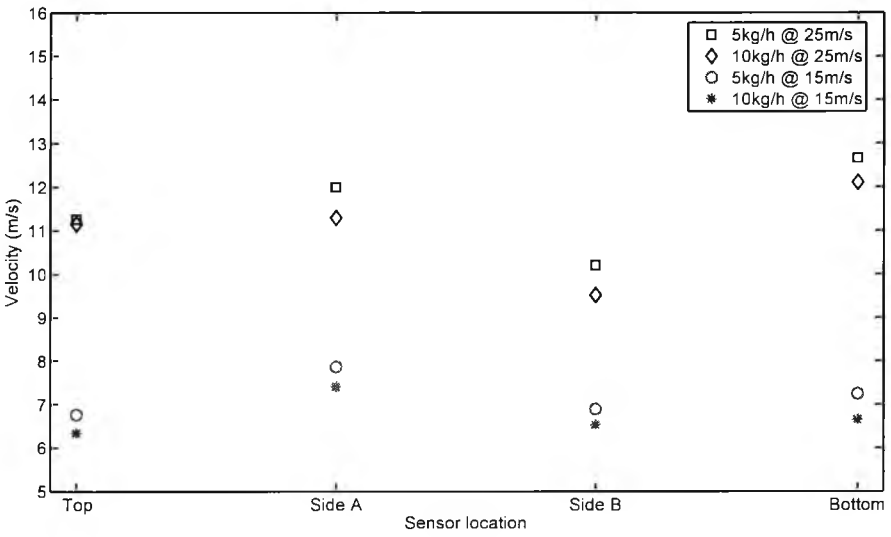


Figure 5.22: Coal correlation velocity—10 mm electrode intrusion

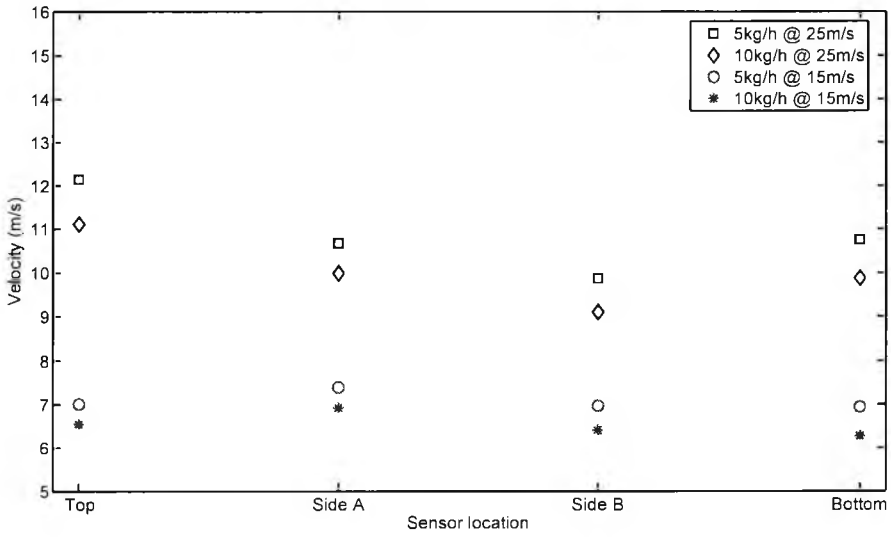


Figure 5.23: Coal correlation velocity—20 mm electrode intrusion

5.2.3.5 Signal power

Figs. 5.24 to 5.27 show the relative signal powers induced onto the sensor electrodes. It is immediately seen that the signals induced by coal are much stronger that those induced by wood chips. This is not thought to be of major importance, as only signal similarity between upstream and downstream electrode locations is considered in velocity measurement. Also, it is clear that a stronger signal is induced for greater electrode intrusion. This is as expected since a greater intrusion causes the electrode to be exposed to the electrostatic charge of a greater number of particles.

There is a noticeable difference between upstream and downstream signal power—the downstream power is reduced in all cases. This is to be expected since some particles are expected to collide with the first electrode, releasing their charge. This effect is less pronounced on the second electrode because the particles colliding with the first electrode do not have time to

charge up to the same level and, in addition, the second electrode is partially shielded from collisions by the first electrode. For coal, however, this effect is not enough to degrade the signals significantly enough to interfere with the cross correlation technique used in velocity measurement.

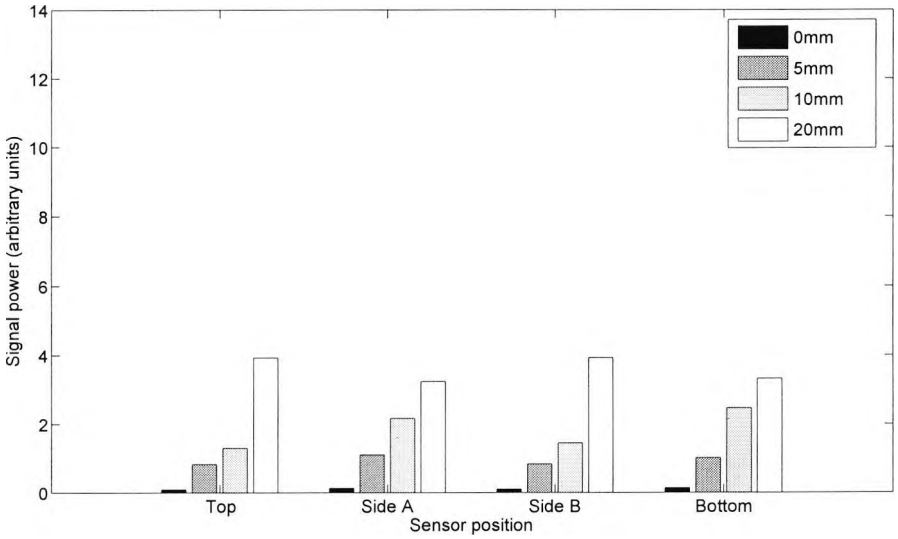


Figure 5.24: Upstream signal power—wood chips

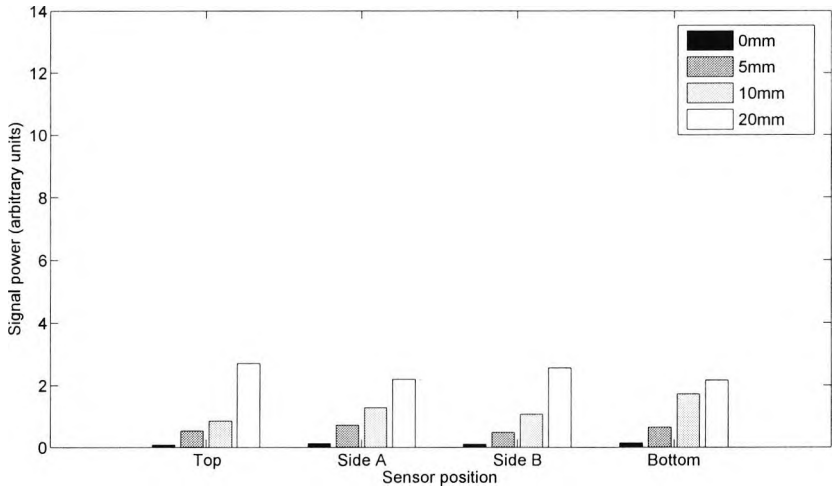


Figure 5.25: Downstream signal power—wood chips

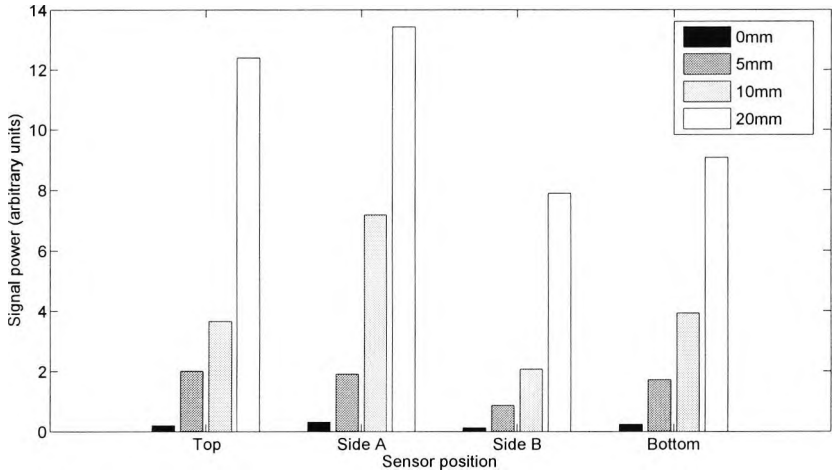


Figure 5.26: Upstream signal power—coal

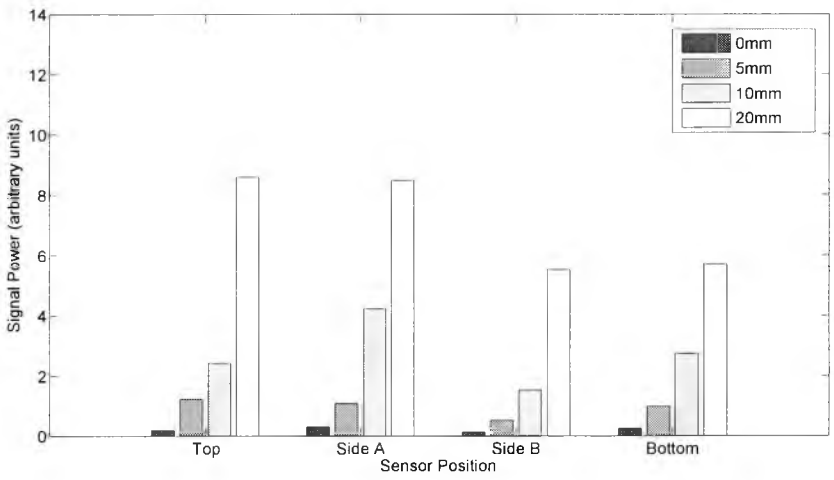


Figure 5.27: Downstream signal power—coal

The practical results can be related to the modelling work in chapter 3. Fig. 3.11 showed that the sensing zone is nearly uniform along the electrode length, suggesting that induced power should be proportional to the square of the intrusion depth. In practice, signals due to the individual passing particles will not all be in phase, and a doubling of electrode length will not necessarily produce a signals with double the amplitude (or four times the power). From Fig. 5.28, however, where a quadratic curve is fitted to the mean of the upstream signal powers, it can be seen that the power is in fact roughly proportional to the square of the electrode intrusion depth.

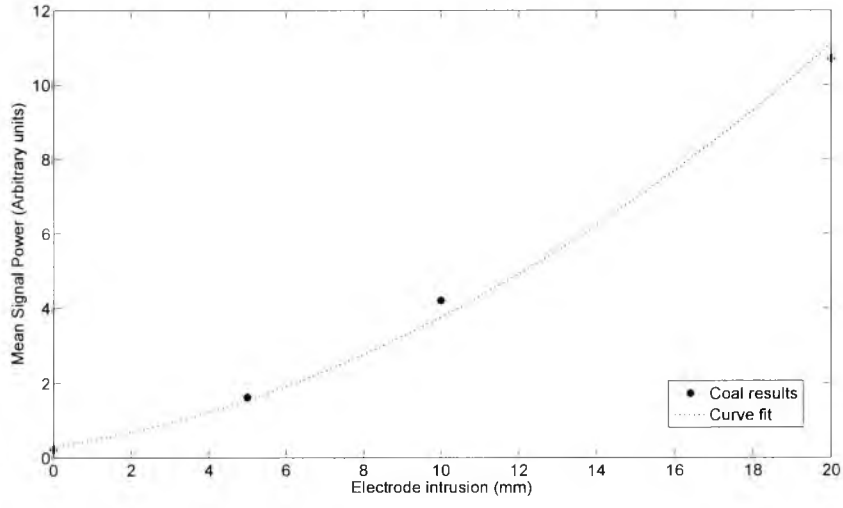


Figure 5.28: Average upstream signal power and fitted quadratic curve

5.3 PCME flow rig trials

5.3.1 Background

Following the tests at the 500 kW (section 5.2) combustion test facility, an integrated optical/-electrostatic sensor was constructed. The optical component was designed and constructed by a partner in the DTI project, Dr. R. Carter. The sensor is shown (unscrewed from its aluminium casing) in Fig. 5.30. The round windows for the optical system can be seen on the sensor face, which forms part of the pipeline wall when installed. The four intrusive electrodes, unevenly spaced so that the effects of electrode spacing can be studied, are also seen. A ceramic coating was later applied to the sensor face and electrodes in order to increase its resistance to abrasive wear.

The motivation for constructing an integrated sensor was to use data from optical images, providing concentration and particle size information, together with velocity data from the

electrostatic sensors in order to determine the mass flow rate in the pipeline. A fan-driven pneumatic flow test rig (Fig. 5.29) was made available at PCME, a UK based instrument design and manufacturing company with an interest in coal power plant instrumentation (www.pcme.co.uk). Two pipeline diameters were available: 300 mm and 500 mm. The fan was capable of driving the air at a maximum air speed of about 18 m/s in the smaller diameter sections. A close up photograph of the mounting flange (a standard 4" BSP port) is seen in Fig. 5.31.

5.3.2 PCME test setup

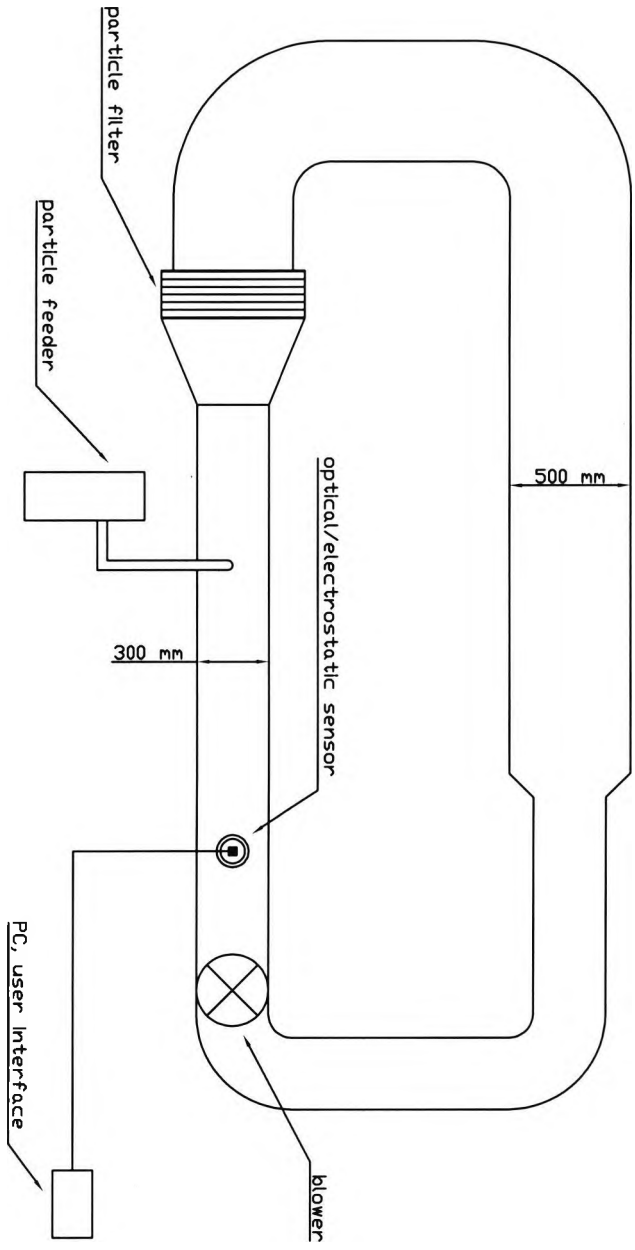


Figure 5.29: PCME flow rig schematic

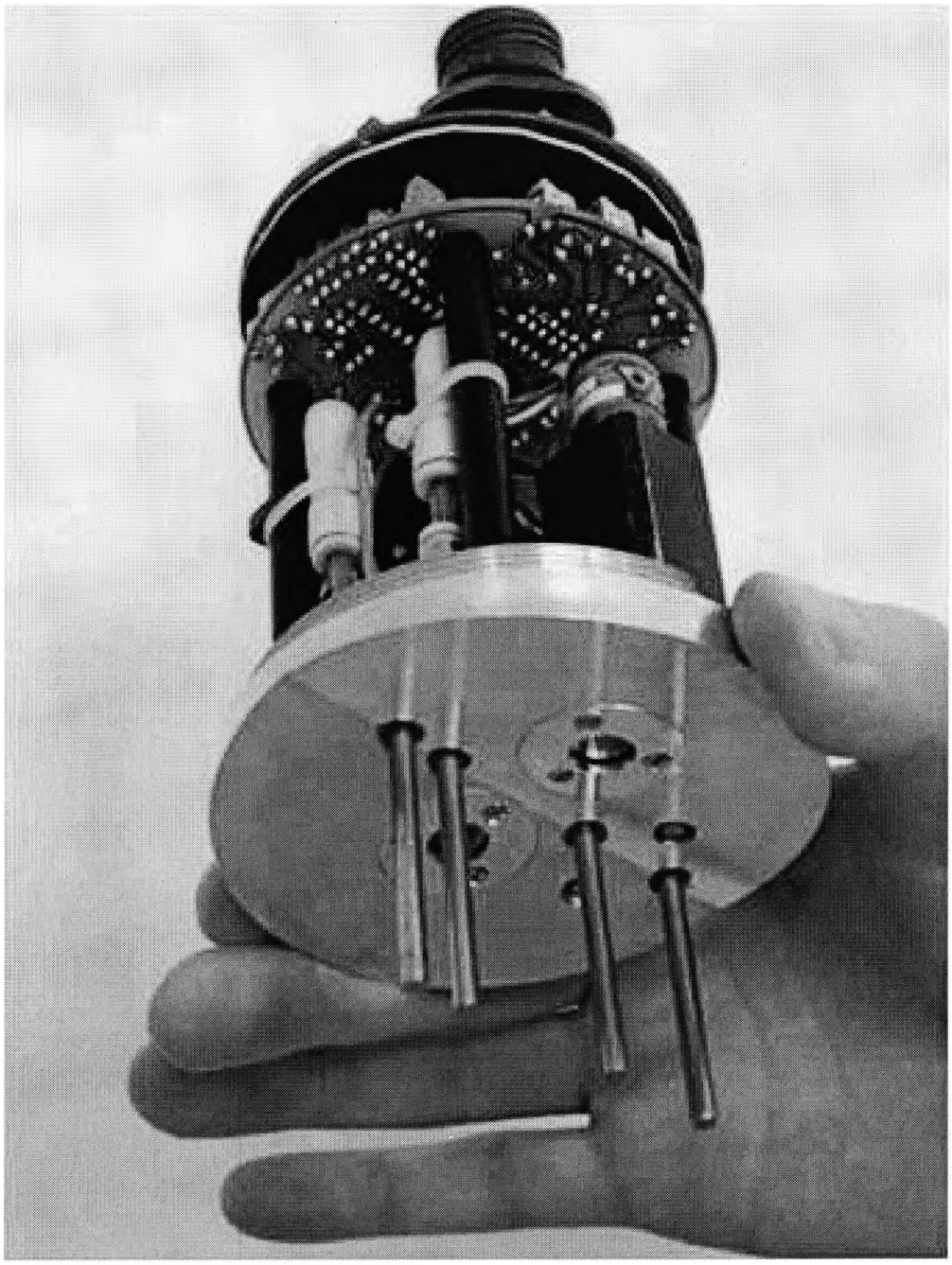


Figure 5.30: Optical/electrostatic integrated sensor photograph

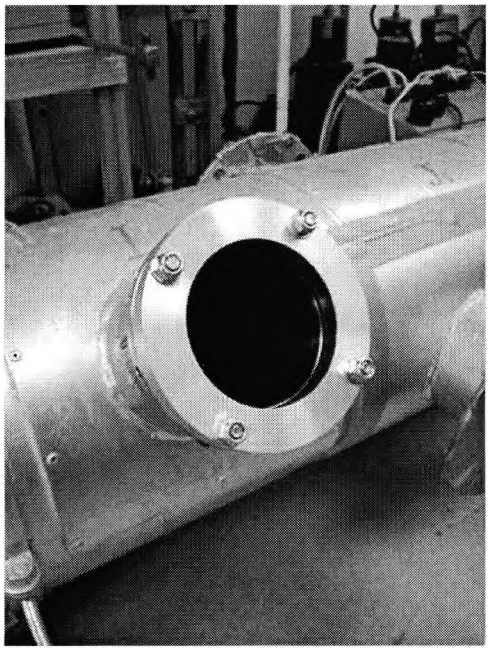


Figure 5.31: PCME flow rig port photograph

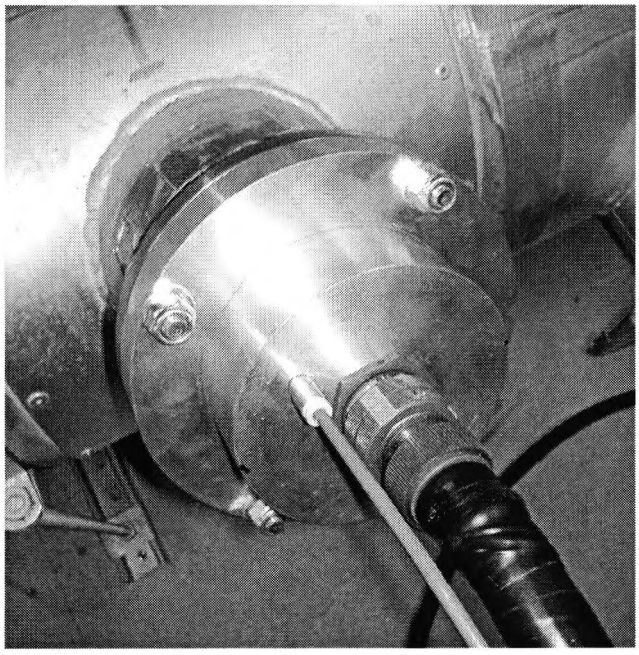


Figure 5.32: Optical/electrostatic integrated sensor installed on pipeline

Earlier results from the 500 kW trials indicated that the non-intrusive stud electrodes, i.e., an intrusion depth of 0 mm, gave a weak signal, low correlation results and an unstable velocity measurement (section 5.2). Using intrusive electrodes, even at a shallow intrusion depth of around 0.1 pipeline diameters, improved measurements considerably, with only modest improvement with further intrusion. This was used as a guide to determining the intrusion depth of the integrated sensor, which was set to 40 mm, giving an intrusion depth of 0.08 and 0.13 diameters into the 300 mm and 500 mm diameter pipelines.

Also, in order to help determine the best electrode spacing, 4 electrodes were designed into the instrument. These were spaced in such a way that, when considered in pairs, spacings of 10 mm to 60 mm in increments of 10 mm could be compared (Fig. 5.33).

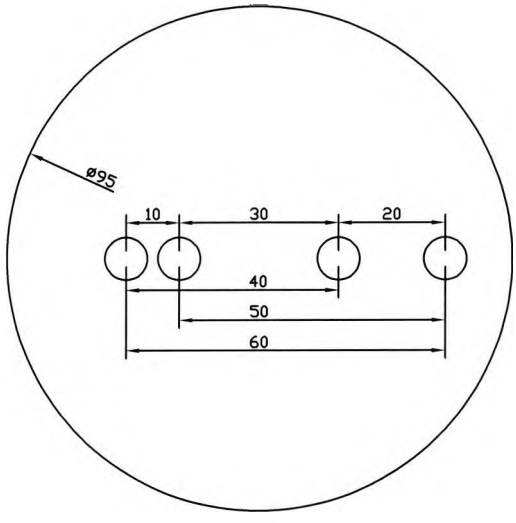


Figure 5.33: Electrode spacings for 300 mm/500 mm pipeline

The optical and electrostatic parts of the sensor were integrated into the same housing, shown installed on the pipeline in Fig. 5.32. However, the data from each sensor were collected and processed independently, so that the sensors could be tested individually. Ultimately, the aim was to produce integrated software to combine the sensors into a single user interface.

The material used was finely ground salt, using a particle feeder (Land, model RBG 1000) capable of delivering a maximum mass flow rate of approximately 0.2 kg/h.

5.3.3 PCME test results

The particle flow concentrations available in this test setup were considerably more dilute than in the 500 kW trials, as the test rig is designed for very dilute particle flows, characteristic of stack emissions. With a maximum of 0.2 kg/h in a 300 mm diameter pipeline, the volume fraction taken up by the particles moving at 18 m/s was approximately 9×10^{-9}. With such a dilute flow, the induced electrostatic signal was very weak, and only occasional particles were visible using the optical sensor.

Figs. 5.34 to 5.36 summarise the test results. It can be seen that the electrode spacing did not have a significant effect on the velocity measurement. A wider spacing results in a proportionately longer delay interval between upstream and downstream signals, but when the spacing is divided by the delay time in order to calculate the correlation velocity, the results are similar. This is consistent with the modelling results (section 3.8), which indicate that velocity measurement distortion due electrode interaction, even when using close spacing, is only significant for shallow or non-intrusive electrode depths. Also, the correlation peak value is reduced as electrode spacing is made wider. This is to be expected, since the particle configuration in the pipleline at the upstream electrode gradually changes as the flow progresses, due to variation in particle velocity and trajectory.

Velocity measurement using the optical sensor was also consistent with the electrostatic results. Optically, a velocity measurement was performed (by Dr. R. Carter) by using the particle image displacement between optical exposures, in a manner similar to particle image velocimetry (section 2.3.6.1). In the example shown (Fig. 5.36), the particle moved an

average of 18.33 pixels per $30\mu s$ exposure, giving a velocity result of 5.55 m/s, similar to the electrostatic correlation velocity.

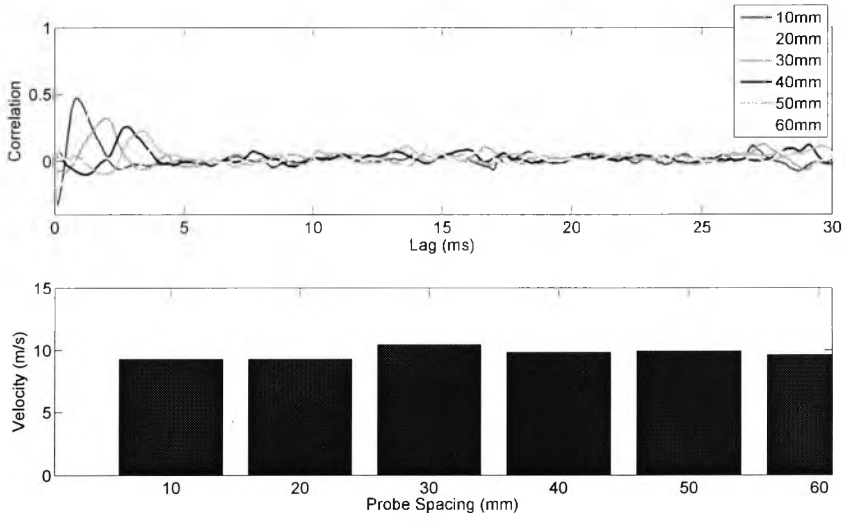


Figure 5.34: Correlation velocities in 300 mm pipeline

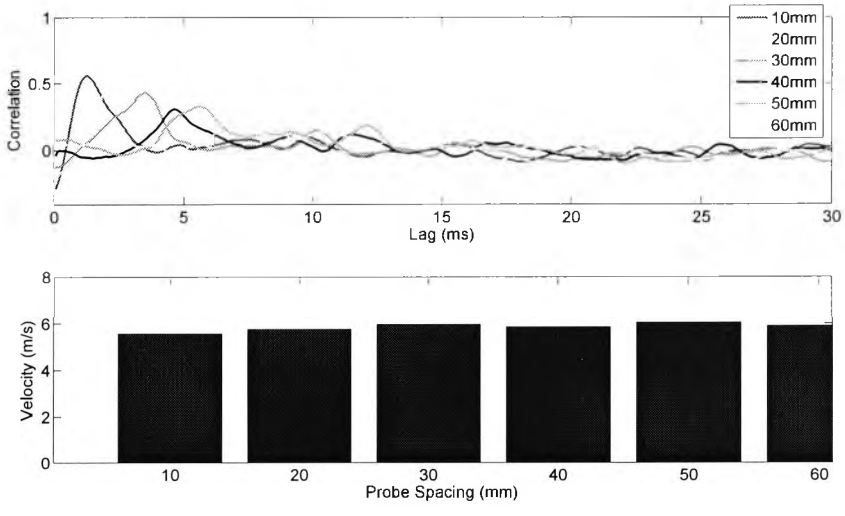


Figure 5.35: Correlation velocities in 500 mm pipeline

Figure 5.36: Particle image from optical system

5.4 Flow rig at the University of Kent

5.4.1 Introduction

The test rig at the University of Kent consists of a U-shaped stainless steel pipeline of 48 mm internal diameter (Fig. 5.37). The pneumatic flow is driven using negative pressure from a suction fan at the pipeline outlet. Particles are introduced into the system by means of a storage hopper positioned above a revolving turntable. The rotation of the turntable causes the particles to flow from the hopper onto the turntable platter on which they are carried towards the mouth of the pipeline, positioned just above the opposite side of the turntable, where the negative pressure in the pipeline draws the particles into the system. The test rig has several removable flanged sections so that instruments can be installed in-line. This setup enables positioning of sensors either along a straight, horizontal section or in a vertical drop.

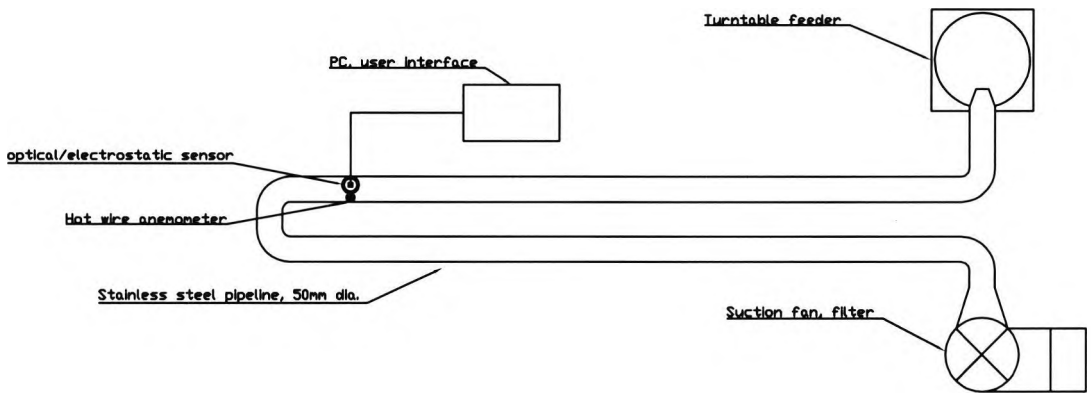


Figure 5.37: Flow rig setup at the University of Kent

From the results at 500 kW and PCME it was seen that a correlation peak, sufficiently well defined to generate a velocity reading, was readily obtained under a range of operating conditions. However, the lack of a definitive reference with which to compare the results means that the accuracy of the velocity measurement could not be compared against known standards. The laboratory rig was fitted with a hot wire anemometer that measures air velocity, so that particle correlation velocity could be compared with the velocity of the conveying air.

5.4.2 Laboratory test rig results

Following a theoretical analysis of electrode intrusion depth (section 3.6), another set of tests was conducted using the laboratory pipeline setup. The electrostatic sensor was mounted in a location on top of the stainless steel pipeline. The material used in this case was rice flour, which has a particle size similar to that of pulverised coal, but has the advantage of being a clean and safer alternative. Rice flour can be explosive in the right mixture with air. However the risk is minimised by the relatively short overall length of the pipeline, together with the fact that the pipeline is grounded to the large metal frame on which it is supported. The

turntable feeder system (5.4.1) minimises the clumping tendency of the material that caused difficulty in the 500 kW test facility (section 5.2.3.2).

A commercial hot wire anemometer (TPI 565) was installed opposite the sensor electrodes in order to measure air velocity in the pipeline. Whilst this does not measure the velocity of the particle flow itself, it acts as a useful reference. However, the delicate nature of hot wire anemometers necessitates that air velocity is measured in an empty pipeline to avoid the risk of damage to the instrument by the particle flow.

Particles were fed into the system using the turntable feeder. The mass flow rate of particles introduced into the system was 0.934 kg/h, found by averaging the results of three timed trials (Table 5.1). The suction pump power was controlled by setting the supply voltage using a variac. Continuous settings from 0% to 100% were available, but it was found that particles do not remain in suspension below a voltage setting of about 40%. Therefore, vacuum settings from 40% to 100% were used in the trials.

Table 5.1: Mass flow rate calculation for laboratory particle feeder

Starting mass (g)	Ending mass (g)	Time (s)	Mass flow rate (kg/h)
106	52.9	205	0.932
112.5	60.6	200	0.934
107.2	55.2	200	0.936

Four intrusion depths were tested: 5 mm, 10 mm 19 mm, and 24 mm. The 5 mm and 10 mm depths were chosen to determine the response to particles near to the pipeline wall, 24 mm represents an intrusion of halfway across the pipeline, and 19 mm is 0.4 of the pipeline diameter—the intrusion depth that was found to give the most consistent reading regardless of the velocity profile in flow regimes following a power law profile (section 3.6). A representative plot of raw data (19 mm intrusion, 100% vacuum setting) is shown in Fig. 5.38.

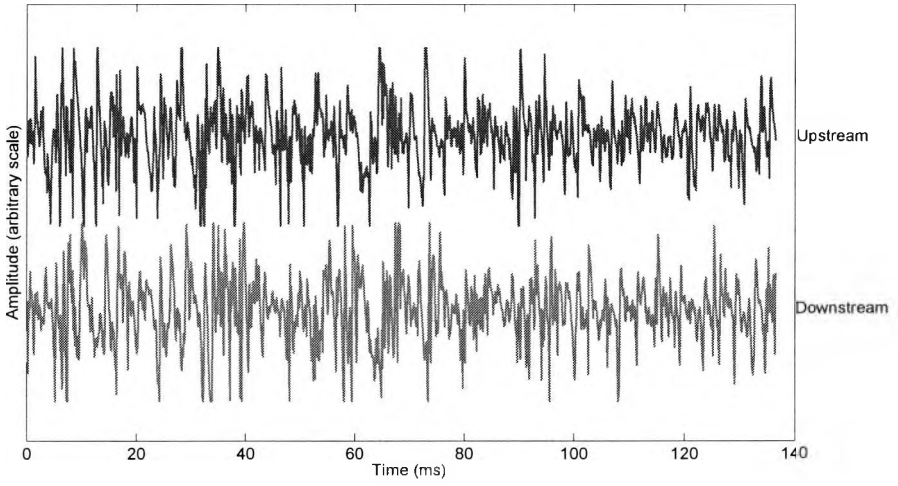


Figure 5.38: Raw electrostatic signal—19 mm intrusion depth, 100% vacuum setting

Trials were conducted at each intrusion depth at vacuum settings from 40% to 100%, in 10% increments, and the correlation velocities are summarised in Fig. 5.39. 50 sets of results were used to produce each data point. The empty pipeline air velocity at each vacuum setting is also included for comparison.

The mean peak correlation coefficients for the different electrode intrusions and vacuum settings are shown in Fig. 5.40.

A Windows® user interface was created in Visual C++ for online monitoring of the results. Figs. 5.41 and 5.42 show examples of the software in operation. Fig. 5.41 shows the instrument running with the suction pump initially switched off. Each data point represents a time of interval of 1s. After 38s, the power to the vacuum pump was set to 40% of full voltage, and this step change can be clearly seen in the graph. In the second example, Fig. 5.42, a step change of 40% to 50% of full voltage was applied and, again, a step change is observed in the output window. This fast response to step changes in flow conditions is made possible by the fast signal processing speed of the embedded dsPIC microcontroller, which

was able to process four cross correlation operations per second.

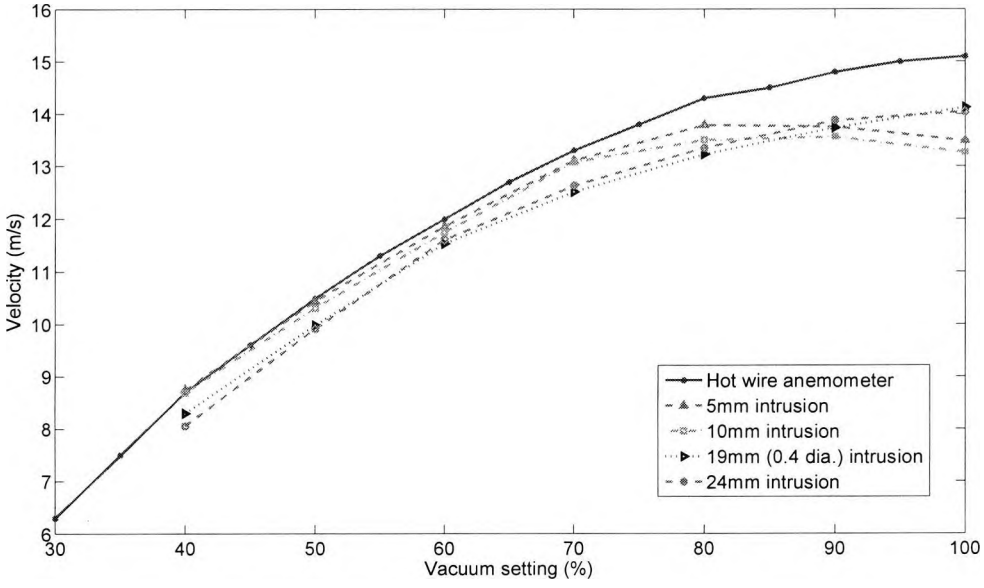


Figure 5.39: Correlation velocity vs vacuum power

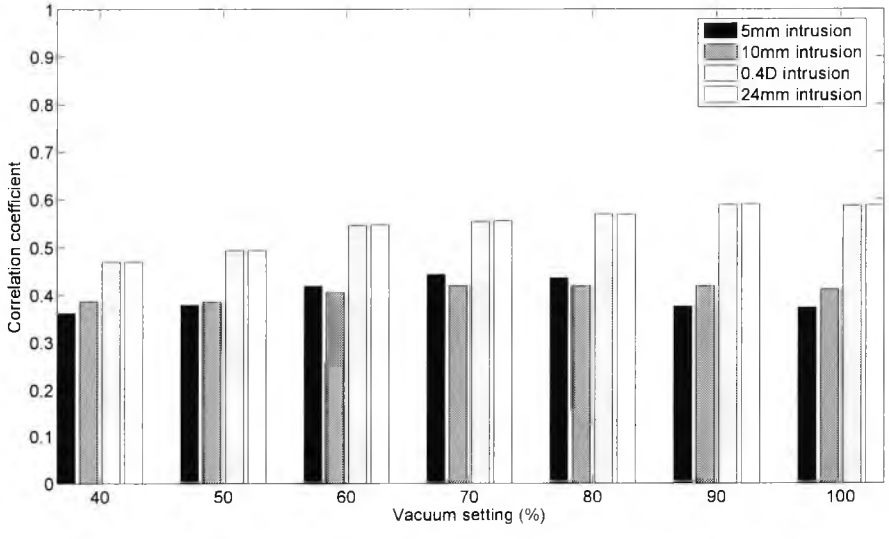


Figure 5.40: Peak correlation coefficient vs vacuum power

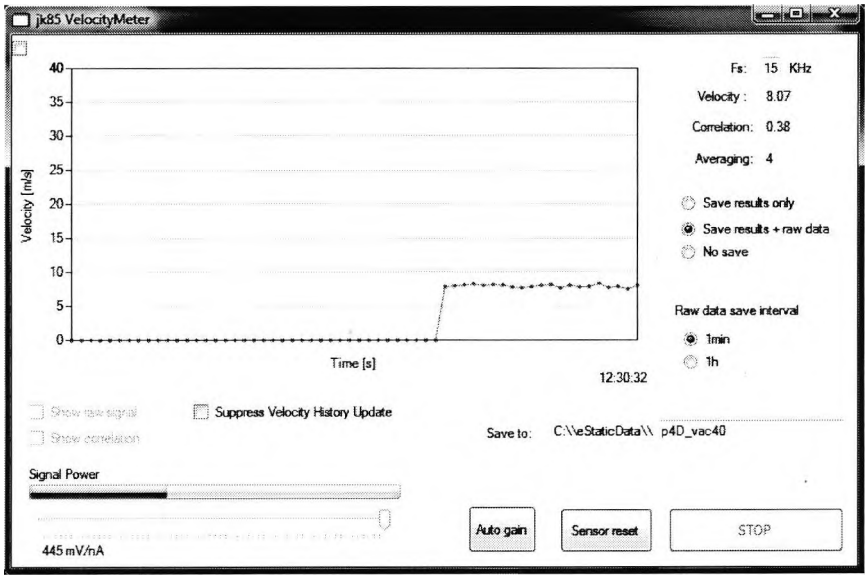


Figure 5.41: Visual C++ software example—vacuum 0% to 40% step

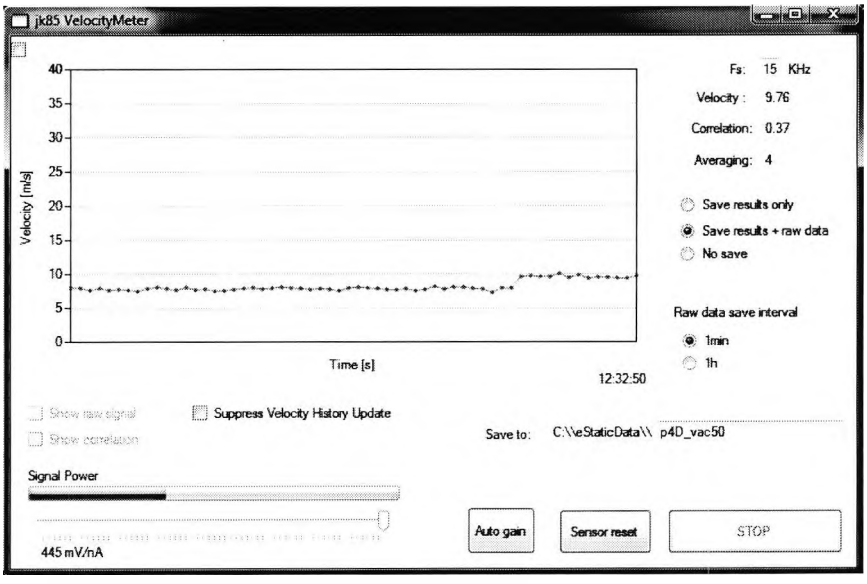


Figure 5.42: Visual C++ software example—vacuum 40% to 50% step

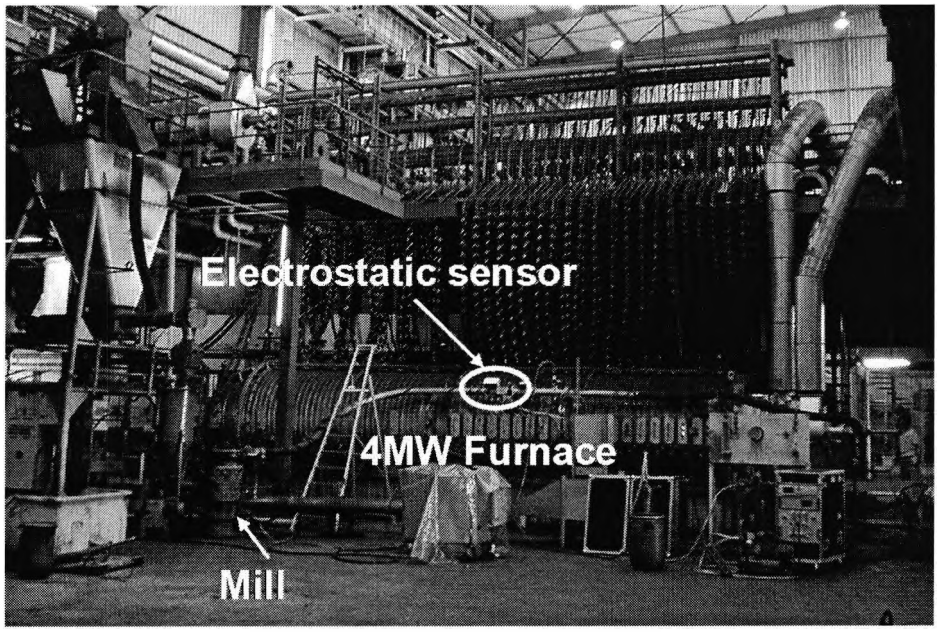
5.5 4 MW combustion facility tests

5.5.1 Introduction

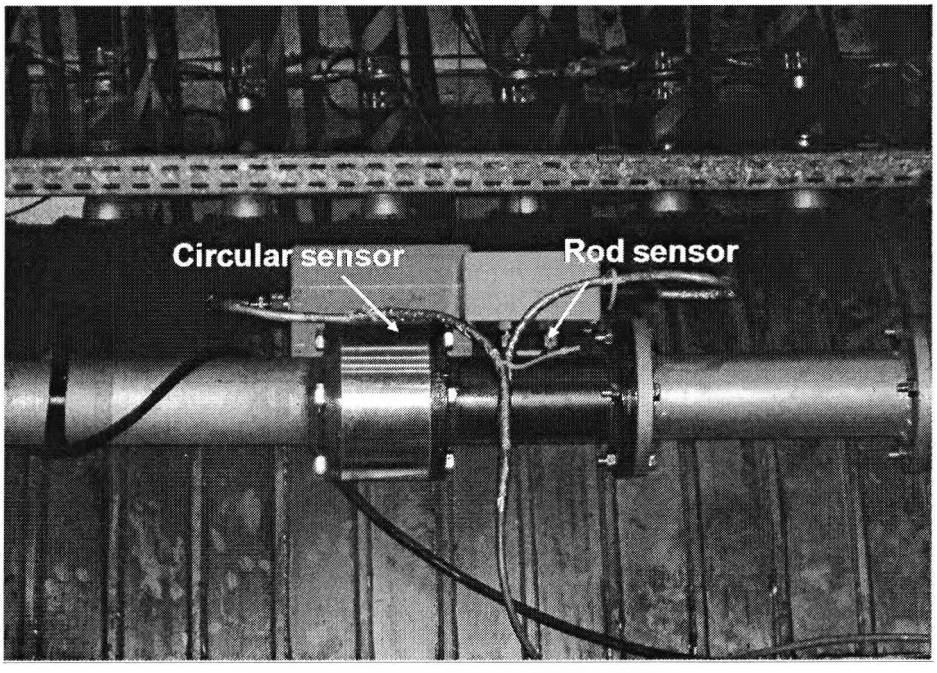
Tests on a large scale, 4 MW combustion test facility were conducted in order to compare the performance of intrusive and non-intrusive sensor designs. The localised sensing zones of electrostatic sensors suggest that velocity measurement in identical flow conditions will vary between the two sensors. The non-intrusive ring electrodes were installed upstream of the intrusive ones to avoid interference between the two sensors. The hardware was constructed and tests performed by Dr. J. Shao, co-author of the publication resulting from the tests conducted in this section Shao et al. [2009, In print].

5.5.2 4 MW test rig setup

The two sensors were installed inline with each other on a pipeline with an internal diameter of 94 mm (Fig. 5.43). Both the non-intrusive and intrusive electrodes were made of stainless steel, spaced 50 mm apart. The non-intrusive electrodes were constructed with an width of 4 mm, whilst the intrusive electrodes were made of 4 mm diameter stainless steel rod extending 47 mm into the pipeline (half the pipeline diameter). Tests were conducted with the intrusive electrodes installed on the top and side of the pipeline (Fig. 5.44). The sensors were installed on a horizontal section of the pipeline, approximately 7 m from the outlet of the mill. Two types of coal—Colombian coal (CC) and South African coal (SA)—were fired for the tests. The South African coal was tested with additions of 0% to 20% biomass (dry sawdust pellets) premixed before milling.



(a) Overview of the combustion test rig



(b) Close up view of the installed sensors

Figure 5.43: Sensor installation on a 4MW combustion test rig

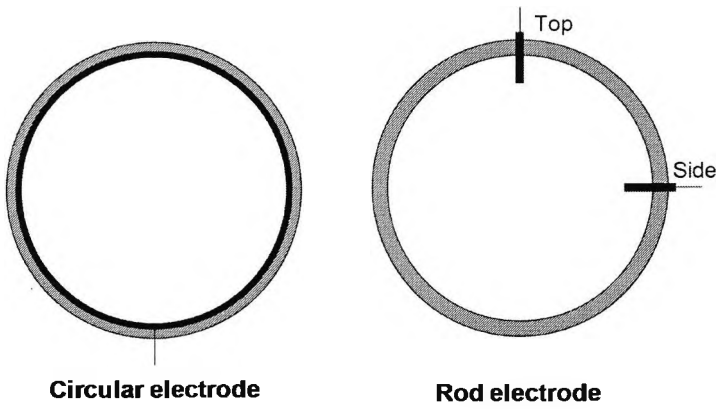


Figure 5.44: Electrode installation locations

5.5.3 4 MW test rig results

Both intrusive and non-intrusive sets of electrodes gave consistent correlation velocity readings, with clearly defined correlation peaks and correlation coefficients of around 0.35-0.50 for the non-intrusive electrodes and 0.55-0.75 for the intrusive rod electrodes (in both the top and side locations). Fig. 5.45 shows typical results from the sensor. The lower correlation coefficients for the non-intrusive sensor are consistent with the results from the 500 kW combustion test facility (section 5.2), and it is thought that this is at least partially due to the lower signal strength of non-intrusive electrodes (Fig. 5.47 (a)), although it could also be due to a more irregular flow near the pipeline wall.

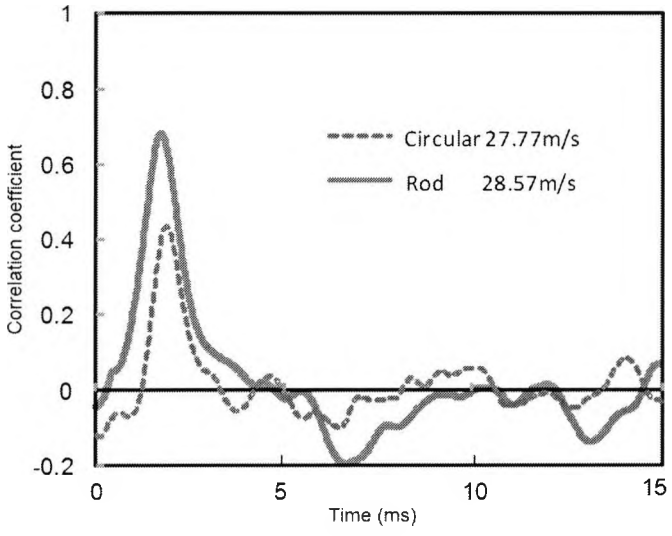


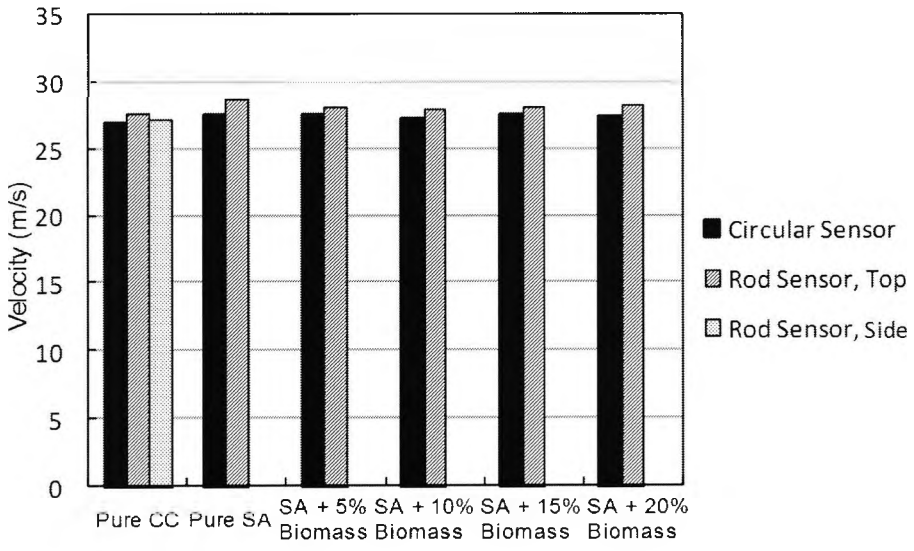
Figure 5.45: Cross correlation function example for intrusive and non-intrusive electrodes

Fig. 5.46 shows the correlation velocity and its standard deviation for the two sets of electrodes under various firing conditions. Fig. 5.47 shows the RMS value and standard deviation of the induced signals, averaged over 500 readings, taken at 1s intervals under steady flow conditions. Note that for tests using pure Colombian coal the rod electrode was positioned in both the top and side positions but for the South African coal tests, with varying amounts of added biomass, the electrode was placed in the top position only.

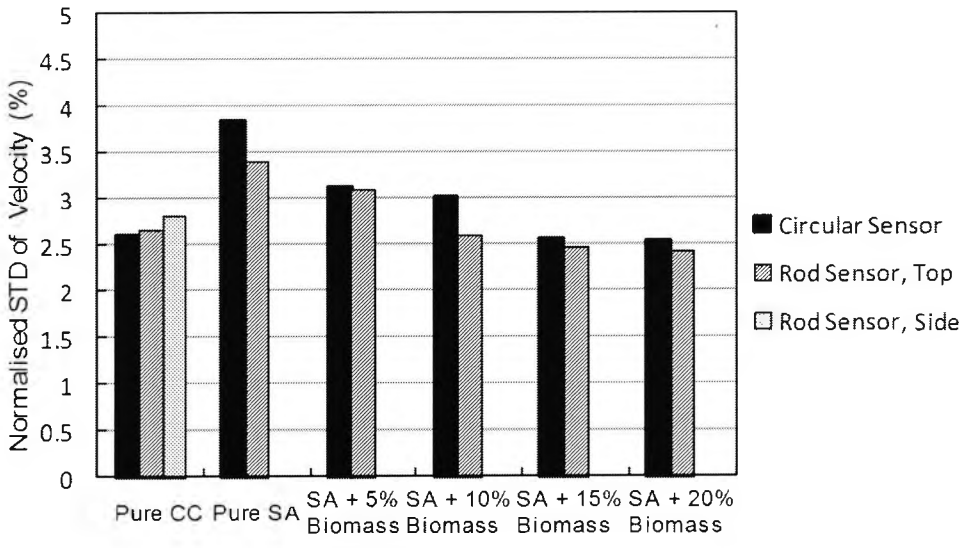
From Fig. 5.46 it is seen that the intrusive electrode in the top position gives a consistently higher correlation velocity than the non-intrusive circular ring for both the Colombian and South African coals. In addition, the electrode mounted in the side position (Colombian coal trials only) gives a velocity measurement between that obtained from the top position and the circular electrode. This is most likely due to stratification of the flow within the pipeline, with larger, slower particles concentrated in the lower part of the pipleline. As the

circular non-intrusive electrode obtains its signal from around the entire circumference of the pipeline, it is not surprising that its correlation velocity is lower than from both intrusive electrodes.

It can also be seen that the normalised standard deviation of the RMS value of the signal decreases with the addition of biomass. This is thought to be due to the fact that the biomass added was in the form of dry sawdust pellets, creating a drier fuel with better mixing of the two phases, and resulting in a steadier flow. The signal on the upstream electrode is consistently higher than on the downstream electrode. This is consistent with the results from section 5.2, and due to the charge released onto the upstream electrode from particle collisions.

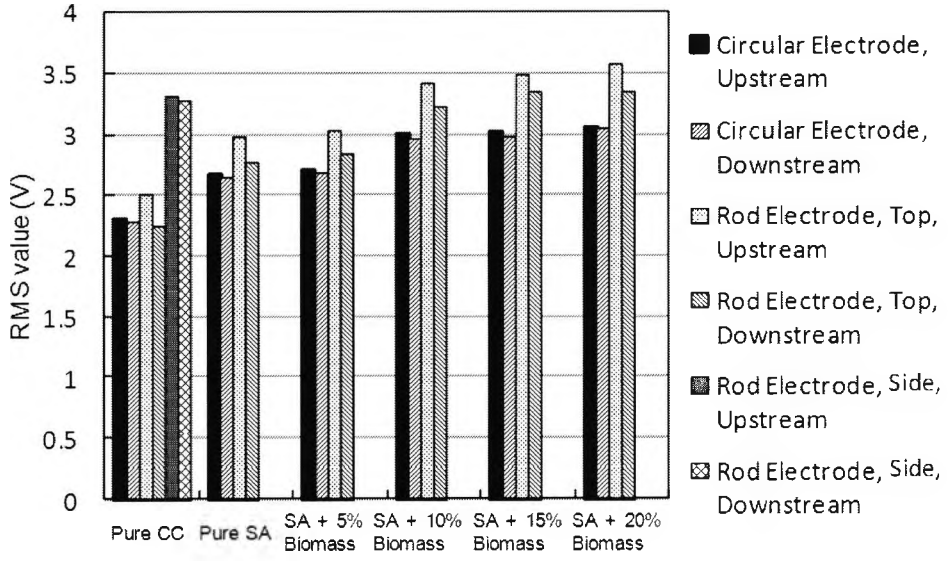


(a) Correlation velocity

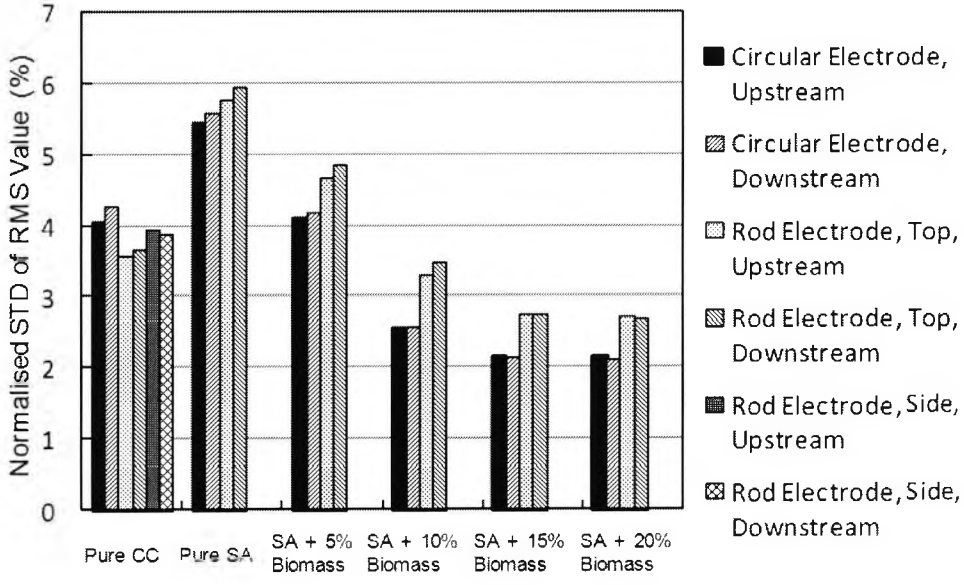


(b) Standard deviation of correlation velocity

Figure 5.46: Correlation velocity and standard deviation for Colombian (CC) and South African (SA) coals with varying amounts of added biomass



(a) RMS value of the signals



(b) Standard deviation of the RMS values

Figure 5.47: RMS values of electrostatic signals and their standard deviations

5.6 Summary of practical results

The four test setups produced results of varying quality. In terms of signal strength and peak correlation the 500 kW combustion test facility, when running pulverised coal, produced the strongest results with consistent, well defined correlation peaks of around 0.80 to 0.85. This high value means that the correlation velocities should be the most reliable, and that the particles in the sensing zone of the sensor were moving at similar speeds. Wide variation in particle speed would have had a detrimental effect on correlation peak value—this follows from the fact that, for particles moving at widely varying velocities, the configuration of particles at the upstream and downstream electrode locations is different, resulting in dissimilar induced signals.

However, the 500 kW results suggest a velocity profile contrary to that of a laminar fluid, with particles near the pipeline wall showing a higher speed than in the center of the pipeline. As discussed in section 5.2, this can be partially explained by the close electrode spacing in these trials, but it also suggests that the particles may not always follow a regular laminar velocity profile, especially in narrow pipelines with turbulent flow. This effect is also seen, to a smaller degree, in the University of Kent laboratory results, where the deeper intrusions gave somewhat lower velocity readings. It should be noted that the Kent pipeline internal diameter was similar to that used in the 500 kW trials (48 mm and 40 mm respectively) and so a turbulent flow regime could be expected.

The comparison of different electrode spacings in the PCME rig suggest that, at least for flows in a much larger diameter pipeline, and using a relatively deep intrusion depth of 40 mm, the electrode spacing does not have a significant effect on velocity results. This is corroborated by the modelling results which suggest that electrode interaction is significant only when spacing is very close.

Intrusive electrode depth was kept constant during the 4 MW combustion facility tests, but

consistently gives a slightly higher correlation velocity than the non-intrusive circular electrode when placed at the top location. When placed at the side location, the correlation velocity is between the value at the top location and the non-intrusive electrode. The broad similarity between all the results suggests that the flow is a turbulent one, without the well defined velocity profile characteristic of laminar flows. The slight difference between velocity readings when the electrode is installed in the top and side locations may also be due to stratification of the flow (Fig. 1.1) where larger, slower moving particles are concentrated near the bottom of the pipeline.

When the material used was coal, the best signal strength and correlation peak was achieved. Rice flour produced results almost as good, and the relatively coarse wood chips gave comparatively poor results in the 500 kW trials, although the addition of sawdust improved the 4 MW results in terms of induced signal strength and correlation velocity repeatability. The use of pulverised coal was only possible at the 500 kW and 4 MW test rigs. The reason for this is that the particles were fed directly into the 500 kW and 4 MW test furnaces, with the exhaust safely vented outdoors. Both the PCME and the University of Kent setups were enclosed rigs with the material collected by a filtration system. Coal is too hazardous to be used in such enclosed conditions.

The comparatively better results using added biomass on the 4 MW installation can be at least partly explained by particle size. The small (40 mm) diameter of the 500 kW test rig and correspondingly small proportions of sensor electrodes, along with and the relatively large and irregular shapes of the wood chips, means that as the individual woodchip particles spin they induce significantly different signals on upstream and downstream electrodes. However, the sawdust for the 4 MW trials was added with the coal before the milling process, and was therefore ground to a particle size closer to that of the coal. Therefore, there is less disruption to the signal due to spinning particles. The larger scale of the pipeline and electrodes relative to the particles further reduces the disruption.

At the end of all the tests, the electrodes were examined and showed no abrasive wear at all. The industrial partners at RWE, however, advise that under full scale industrial conditions abrasive wear would certainly take place. In response, the sensor electrodes and the face plate forming part of the pipeline wall were coated in silicon carbide to control abrasion, in anticipation of future trials on a full scale industrial power plant.

Chapter 6

Conclusions and recommendations for future research

6.1 Contributions from this research

The work conducted in the course of this research has advanced progress in many areas of pneumatic particle flow measurement that have unresolved challenges associated with them. In particular, intrusive electrostatic velocity sensors have been studied using finite element modelling, implemented in a low cost embedded system design and shown to perform well under industrial operating conditions, obviating many of the difficulties characteristic of other measurement systems. FEM has been shown to improve on previous, simplified electrostatic models which can have significant inaccuracies. It has also shown that intrusive electrodes can monitor localised flow phenomena inside the pipeline, and this has been corroborated in industrial trials. Guidelines regarding electrode cross sectional shape and intrusion depth have been established, and areas that would benefit from further research have been highlighted.

6.2 Conclusions from modelling

6.2.1 Electrode spatial sensitivity

It was seen in chapters 2 and 3 that, compared to other measurement techniques, the sensing field of electrostatic sensors is relatively difficult to define. This was shown to be due to the fact that the electric field resulting from charged particles in the presence of conducting materials has a complex distribution, with no analytical solution available for all but a few simple configurations involving a high degree of symmetry. As such, numerical methods based on finite element modelling were used to solve for the electric field, from which the induced charge distribution on conducting sensor electrodes was determined. This is a time consuming and computationally intensive procedure, and it is necessary to model each sensor system configuration of interest separately. Some generalisations, however, may be made regarding the electrical fields and induced charge.

Most importantly, the sensing field was found to be localised to the region very close to the electrodes. Away from the electrodes, the grounded pipeline evidently attracts most of the field lines from the charged particles, and relatively little charge is induced onto the sensor electrodes. In the case of a non-intrusive electrode, considerably less charge can be induced when the effect of the conducting pipeline is considered, compared to using a free space approximation which disregards the effect of pipeline, even when the charge is located in the axial center of the electrode where maximum charge is induced (Fig. 3.4). When the axial position of the particle is further upstream or downstream, the effect of the pipeline is likely to be even more significant, due to the closer relative distance of the particle to the pipeline wall than to the sensor electrode.

Localised sensitivity can be either an advantage or a disadvantage. If only one set of electrodes is available, then the solids distribution and velocity profile in the pipeline will have

a more pronounced effect on the result. Inaccuracies will occur if the flow in the vicinity of the electrode is not representative of the flow in the rest of the pipeline. However, if several electrodes are positioned around the perimeter and electrode intrusion depth is sufficient to sample a representative portion of the pipeline cross section, then information regarding the localised flow regime can be gained without the complication of adopting tomographic methods. This could be clearly seen in the results from the 500 kW test rig (section 5.2), where the reduction in flow velocity on one side of the pipeline, due to a slight pipe section misalignment, was consistently observed, even when operating with different conveying air velocities and mass flow rates (Figs. 5.20 to 5.23).

6.2.2 Electrode spacing

Theoretical studies into electrode spacing have been previously conducted, and the possibility of unwanted electrode interaction has been suggested (section 3.8). However, there are competing design considerations—theoretical studies favour a reasonably close spacing (equation 3.24), depending on the flow velocity, bandwidth and velocity fluctuation, but separation must be great enough to avoid electrode interaction. Evidence interaction from close spacing was seen on the 500 kW test rig (section 5.2), where the correlation velocity was found to be greater towards the pipeline wall. Also, if spacing is too great, then the effects of the velocity profile within the pipeline and the changing particle configuration due to random particle motion will result in a low cross correlation coefficient, producing less reliable velocity readings, or even making measurement impossible due to an insufficiently high correlation peak.

Modelling results have confirmed that, for 10 mm electrode spacing in a 40 mm diameter pipeline with 1.6 mm diameter electrodes, the effect of the interaction is to make the electrodes appear to be closer than their physical separation. This error was found to increase for particle streamlines farther from the electrodes, but to decrease with electrode intrusion

depth. By increasing the separation distance to 20 mm, the effect of electrode interaction is reduced to a negligible amount.

6.2.3 Electrode cross sectional shape

Electrode modelling has suggested that the cross sectional shape of the electrode (keeping a constant surface area), has little influence on the induced signal (section 3.9). This can also be understood on an intuitive level, since unless the charged particle is very near the electrode, the differences in distance from the particle to any particular point on the electrode surface are small regardless of electrode cross sectional shape, and when the particle is near the electrode, the electric field lines tend to terminate somewhere on the electrode surface. When the induced charge is integrated over the entire surface, the results are similar for electrodes with equal surface areas. From Fig. 3.23, it can be seen that the largest difference in induced charge is from a blade shape, due to its more extreme aspect ratio (i.e., the ratio of the largest cross sectional width to the smallest) than square or circular electrodes.

In light of this, the most practical electrode cross section is circular, since circular rod electrodes are easiest to manufacture. Cross sections with corners are likely to suffer more abrasive wear and would, in any case, eventually be eroded to a rounded shape. A blade shape offers less resistance to the flow, as the narrow end could be orientated to face the flow, but the lower mechanical strength in the lateral direction means that care must be taken to avoid vibration due to the flow of turbulent air around it.

6.2.4 Electrode intrusion depth

Modelling streamlines of particles with different velocities has shown that correlation velocity, derived from the peak value of the cross correlation function, depends on how much

of the profile is detected by the sensing electrode. For flows following a power law velocity profile (equation (3.11)) across the cross section, it has been found that an electrode intrusion depth of 0.40 of the diameter of the pipeline results in a consistent velocity measurement regardless of the power law index. However, the result must be scaled to derive the true mean velocity across the pipeline. For flows not following a regular velocity profile, more work needs to be undertaken to determine the effect this has on correlation velocity.

6.3 Sensor design evaluation and conclusions from experimental results

6.3.1 Effects of electrode intrusion on practical results

Intrusive electrodes have been shown to be sensitive to localised flow conditions as predicted by the localised spatial sensitivity results from modelling. This offers the possibility of installing several electrodes around the pipeline perimeter in order to detect inhomogeneous flow regimes, and is an advantage over circular ring electrodes, which are sensitive mainly to the region close to the pipeline walls. Also, the results indicate that a shallow intrusion depth (about 1/8 of the pipeline diameter) gives a stronger signal and results in a higher correlation peak compared with non-intrusive stud electrodes, but that further intrusion makes little difference in this regard. This allows the depth of intrusion to be decided primarily on other criteria, such as the cross sectional zone of interest, or the optimal intrusion depth based on the type of flow regime.

Results comparing sensor performance using coal, rice flour and biomass (woodchips) indicate that using coal and rice flour results in high correlation coefficient peaks (typically better that 0.8 for coal/rice flour and less than 0.4 for woodchips) and low variation in velocity readings (3%-5% variation from the mean value for coal compared to 10%-23% for

woodchips). This is most likely due to the fact that coal and rice particles are small (on the order of micrometers or tens of micrometers), whereas wood chips are thin, but can be several millimeters wide. This higher aspect ratio for woodchips means that the degradation in similarity of upstream and downstream signals, due to particle spinning, is increased. Further research would be useful to determine if sensor design could be altered to improve performance when using biomass or other materials with a relatively large particle size and aspect ratio.

6.3.2 Electronics and signal processing

The hardware, sensor electronics, and embedded processing software developed for the flow rig tests operate as intended. The use of a current to voltage amplifier rather than a high impedance voltage amplifier makes the effects of stray capacitance are negligible, since no fluctuating voltages appear between the sensor and the pipeline. Also, on-board processing within the sensor itself means that long cable runs, with their associated risks of induced noise, are avoided.

The dsPIC® microcontroller is not among the fastest processors for digital signal processing. However, several signal acquisition and signal processing operations can be performed per second—fast enough to allow the averaging of several results for each measurement. In addition, the many additional peripheral functions included in a microcontroller (e.g., A/D sampling and UART serial data interface) enable a compact and inexpensive system to be developed.

6.3.3 Correlation velocity results

Correlation velocity results were obtained in all test setups, but different amounts fluctuation in the velocity measurement were observed, depending strongly on the material being con-

veyed and the peak correlation coefficient achieved. Correlation coefficients peaks of less than about 0.3 risk give widely varying results, since the correlation 'noise' from the off peak portions of the correlation function begins to compete with the true peak corresponding to the upstream/downstream signal delay time.

This was observed in the practical tests, where a stronger the correlation coefficient resulted in a more stable correlation velocity. This can also be interpreted as an indicator of irregular flow, since a high correlation coefficient is only possible if the upstream/downstream particle configurations have a high degree of similarity. If the particle streamlines in the flow have widely varying velocities, then the downstream particle configuration will be significantly different from the upstream configuration, resulting in dissimilar signals, a low peak correlation and more variability in the correlation velocity measurements.

The particle size and shape also affect the correlation peak value and hence correlation velocity stability. Smaller particles tend to produce more consistent correlation velocity results than larger, irregularly shaped ones with a higher aspect ratio. This is thought to be due to a combination of more variation in particle velocity streamlines for particles of widely differing shapes and sizes, and the effect of particle spin inducing varying signals on the upstream/downstream electrodes, depending on the precise orientation of the particle as it passes the electrode.

6.3.4 Effect of electrode spacing

The practical effects of electrode spacing were first observed in the unusual trend of the velocity profile from the results performed on the 500 kW test facility. Subsequent modelling work has indicated that the effect is most severe for shallow intrusion depths, and the results from the tests at PCME, with relatively longer electrodes than those of used in the 500 kW tests, show a consistent correlation velocity regardless of electrode spacing. In practice,

longer electrodes result in a stronger induced signal, and less significant electrode interaction, but the velocity profile of the flow should also be considered when choosing electrode intrusion depth.

6.4 Recommendations for further research

6.4.1 Directions for further modelling applications

The modelled spatial sensitivity and signal bandwidths are consistent with practical results. For flow regimes with variable velocity profiles, however, not all pneumatic particle flows follow the power law velocity profile that was modelled in section 3.6, and further work should be undertaken to determine sensor response to other velocity profiles. For example, in section 3.7, the 'flattening' effect of turbulent conveying air on the velocity profile was discussed. This effect causes the particles to move with a random fluctuating velocity component superimposed on the average particle velocity. The random component of the particle velocity has a detrimental effect on the cross correlation of upstream/downstream signals, since it means that the particle distribution changes as the flow progresses, causing dissimilarity in the signals. This could be used to yield useful information, since it follows that, keeping other flow parameters constant, a lower peak correlation value for a given separation distance corresponds to a larger random velocity component in the particle streamlines. Therefore, with suitable modelling, a relationship can be found between the peak correlation coefficient value and the magnitude of random velocity fluctuation in the particle streamlines. In addition, since smaller particles are likely to have higher velocity fluctuations due to their reduced inertia, this work could conceivably be extended to infer the range of particle size distribution in the pipeline.

In chapter 2, research on instruments using a single electrostatic electrode was reported.

This is a good candidate for further modelling work because in order for these instruments to give accurate measurements, the spatial filtering characteristics of the sensor must be known. This follows from the fact that the signal bandwidth depends not only on flow velocity, but also on the proximity of the particle to the sensing electrode. Modelling would reveal the sensitivity of the measurement result to the effects of inhomogeneous solids distribution, since the distances from the particles to the electrode would no longer be uniform.

6.4.2 Abrasion resistance testing

The intrusive electrodes of the test system designed for 300 mm/500 mm pipelines (section 4.4.3) were coated with silicon carbide to help withstand the abrasion of particle flow, but tests have not been conducted under full scale operating conditions for extended periods, due to the unavailability of the power plant. Clearly, an evaluation of the expected service life of electrodes exposed to particle flows would be necessary before they could be installed for long term use. Informal estimates by coal fired power plant operators suggests that uncoated electrodes would last only a matter of days or less, but the improvement in durability offered by ceramic coatings is not yet clear. Also, abrasion resistant, conductive ceramic materials are becoming available (e.g., titanium diboride) which may prove to be more durable than ceramic coated stainless steel.

6.4.3 Common mode noise testing

A technique for removing the effect of common mode noise on cross correlation velocity measurement was demonstrated in section 3.11. Although the likelihood of external periodic noise has been suggested by power plant operators, the extent of the noise on real sensors has not been established. The simulation in section 3.11, using artificially added periodic noise, provides an illustration of the method, but the magnitude and frequency spectrums of

real world noise are not clear. Additional tests in electrically noisy industrial environments would establish the efficacy of the technique.

6.4.4 Independent validation of electrostatic velocity measurements

Ongoing research by the instrumentation team at the University of Kent is also developing other particle flow measurement techniques, such as optical imaging. When successfully set up on the same pipeline section, this will provide independent validation of electrostatic velocity measurements (which have heretofore proven difficult) using techniques such as particle image velocimetry (section 2.3.6). This will aid in the calibration of electrostatic sensors, as well as providing experimental results regarding the effects of inhomogeneous solids distributions and velocity profiles.

References

H. E. Albrecht, N. Damaschke, M. Borys, and C. Tropea. *Laser Doppler and Phase Doppler Measurement Techniques*. Sprigner-Verlag Berlin, Berlin, 2002. ISBN 978-3-540-67838-0.

S. V. Apte, K. Mahesh, P. Moin, and J.C. Oefelein. Large-eddy simulation of swirling particle-laden flows in a coaxial-jet combustor. *International Journal of Multiphase Flow*, 29:1311–1331, 2003.

A. Arko, R. C. Waterfall, M. S. Beck, T. Dyakowski, P. Sutcliffe, and M Byars. Development of electrical capacitance tomography for solids mass flow measurement and control of pneumatic conveying systems. In *1st World Congress on Industrial Process Tomography*, Buxton, Greater Manchester, 1999.

D. I. Armour-Chelu. *The Measurement of the Charging Properties of Fine Particulate Materials in Pneumatic Suspension.* PhD thesis, University of Greenwich, 1998.

D. I. Armour-Chelu and S. R. Woodhead. Comparison of the electric charging properties of particulate materials in gas-solids flows in pipelines. *Journal of Electrostatics*, 56(1):87–102, 2002.

D. I. Armour-Chelu, S. R. Woodhead, and R. N. Barnes. The electrostatic charging trends and signal frequency analysis of a particulate material during pneumatic conveying. *Powder Technology*, 96(03):181–190, 1998.

B. J. Azzopardi. Turbulence modification in annular gas/liquid flow. *International Journal of Multiphase Flow*, 25(6-7):945–955, 1999.

S. E. Barlow and M. D. Tinkle. The image charge pseudopotential. Technical report, Environmental Moleclular Sciences Laboratory (EMSL), 1999.

I. R. Barratt, Y. Yan, B. Byrne, and M. S. A. Bradley. Mass flow measurement of pneumatically conveyed solids using radiometric sensors. *Flow Measurement and Instrumentation*, 11(3):223–235, 2000.

B. J. Barry, S. Tallon, and C. E. Davies. Concentration variations within pipe cross-sections in a dilute phase pneumatic conveying system. In *IPENZ*, volume 24, No. 1, 1997.

J. Bec, M. Cencini, and R. Hillerbrand. Heavy particles in incompressible flows: The large stokes number asymptotics. *Physica D*, 226:11–22, 2007.

M. S. Beck. Correlation in instruments: cross correlation flowmeters. *J. Phys. E: Sci. Instrum.*, 14(1):7–19, 1981.

M. S. Beck and A. Plaskowski. *Cross Correlation Flowmeters–Their Design and Application*. Adam Hilger, Bristol, 1987. ISBN 9780852745328.

M. S. Beck, J. Drane, A. Plaskowski, and N. Wainwright. Particle velocity and mass flow measurement in pneumatic conveyors. *Powder Technology*, 2(5):269–277, 1969.

M. S. Beck, R. G. Green, and R. Thorn. Non-intrusive measurement of solids mass flow in pneumatic conveying. *J. Phys. E: Sci. Instrum.*, 20(7):835–840, 1987.

T. Boeck. Measurement of velocity and mass flow rate of pnuematically conveyed solids by the use of a correlation measuring technique. In *Proc. of the international conference on the flow of particulate solids*, Bergen, 1989.

M. Campbell, J. A. Cosgrove, C. A. Greated, S Jack, and D. Rockliff. Review of lda

and piv applied to the measurement of sound and acoustic streaming. *Optics & Laser Technology*, 32:629–639, 2000.

R. M. Carter. *On - line measurement of size distribution and volumetric concentration of pneumatically conveyed solids using digital imaging techniques*. PhD thesis, University of Kent, 2005.

R. M. Carter and Y. Yan. Online particle sizing of pulverized and granular fuels using digital imaging techniques. *Meas. Sci. Technol.*, 14:1099–1109, 2003.

R. M. Carter, Y. Yan, and S. D. Cameron. Online measurement of particle size distribution and mass flow rate of particles in a pneumatic suspension using combined imaging and electrostatic sensors. *Flow Measurement and Instrumentation*, 16(5): 309–314, 2005.

Department of Trade DTI and Industry. Multiphase flow technologies in coal-fired power plants, TRS022. Technical report, 2004.

W. J. Duffin. *Electricity and Magnetism*. McGraw-Hill, London, 4th edition, 1990. ISBN 007707209X.

R. Ettema, I. Fujita, M. Muste, and A. Kruger. Particle-image velocimetry for whole-field measurement of ice velocities. *Cold Regions Science and Technology*, 26(2): 97–112, 1997.

S. Fokeer, S. Kingman, I. Lowndes, and A. Reynolds. Characterisation of the cross sectional particle concentration distribution in horizontal dilute flow conveying–a review. *Chemical Engineering and Processing*, 43(6):677–691, 2004.

J. B. Gajewski. Inductive non-contact method for measuring a velocity. *J. Phys. E: Sci. Instrum.*, 16(7):622–624, 1983.

J. B. Gajewski. Mathematical model of non-contact measurements of charges while moving. *Journal of Electrostatics*, 15(1):81–92, 1984.

J. B. Gajewski. Measuring probes, head, and system for the non-contact, electrostatic measurements of the two-phase flow parameters in pneumatic transport of solids. *Journal of Electrostatics*, 32(3):297–303, 1994.

J. B. Gajewski. Electrostatic, inductive ring probe bandwidth. *Meas. Sci. Technol.*, 7: 1766–1775, 1996a.

J. B. Gajewski. Monitoring electrostatic flow noise for mass flow and mean velocity measurement in pneumatic transport. *Journal of Electrostatics*, 37(4):261–276, 1996b.

J. B. Gajewski. Electric charge measurement in pneumatic installations. *Journal of Electrostatics*, 40-41:231–236, 1997.

J. B. Gajewski. Electrostatic flow probe and measuring system calibration for solids mass flow rate measurement. *Journal of Electrostatics*, 45(4):255–264, 1999.

J. B. Gajewski. Frequency response and bandwidth of an electrostatic flow probe. *Journal of Electrostatics*, 48(3-4):279–294, 2000.

J. B. Gajewski. Non-contact electrostatic flow probes for measuring the flow rate and charge in the two-phase gas-solids flows. *Chemical Engineering Science*, 61(7): 2262–2270, 2006.

D. I. Graham. On the inertia effect in eddy interaction models. *International Journal of Multiphase Flow*, 22(1):177–184, 1996.

D. I. Graham and P. W. James. Turbulent dispersion of particles using eddy interaction models. *International Journal of Multiphase Flow*, 22(1):157–175, 1996.

R. G. Green and R. Thorn. Sensor systems for lightly loaded pneumatic conveyors. *Powder Technology*, 95:79–92, 1998.

L. Hajji, Ph. Pascal, and B. Oesterle. A simple description of some inertia effects in the behaviour of heavy particles in a turbulent gas flow. *International Journal of*

Non-Linear Mechanics, 31(3):387–403, 1996.

E. A. Hammer and R. G. Green. The spatial filtering effect of capacitance transducer electrodes. *J. Phys. E: Sci. Instrum.*, 16(5):438–443, 1983.

G. P. Hancke and R. Malan. A modal analysis technique for the on-line particle size measurement of pneumatically conveyed pulverized coal. *IEEE Transactions on Instrumentation and Measurement*, 47(1), 1998.

J. Hong and Y. Tomita. Measurement of distribution of solids concentration on high density gas-solids flow using an optical-fiber probe system. *Powder Technology*, 83(1):85–91, 1995.

Y. Huang, W. Wu, and H. Zhang. Numerical study of particle dispersion in the wake of gas-particle flows past a circular cylinder using discrete vortex method. *Powder Technology*, 162:73–81, 2006.

E.C. Ifeachor and B.W. Jervis. *Digital Signal Processing: A Practical Approach*. Prentice Hall, Harlow, 2nd edition, 2002. ISBN 978-0201596199.

D. R. Jonassen, G. S. Settles, and M. D. Tronosky. Schlieren "PIV" for turbulent flows. *Optics and Lasers in Engineering*, 44(3-4):190–207, 2006.

R. Kacprzyk and Juliusz B. Gajewski. Charging of metal probes by a flux of particles. *Journal of Electrostatics*, 51-52:124–130, 2001.

L. Kajitani and D. Dabiri. A full three-dimensional characterization of defocusing digital particle image velocimetry. *Meas. Sci. Technol.*, 16(3):790, 2005. Publisher: Institute of Physics Publishing.

J. D. Kersch, S. Laux, and T. Rosin. Ect on-line coal flow measurement technology - benefits and experience from three years of operation. In *EPRI-DOE-EPA Combined Utility Air Pollutant Control Symposium*, Chicago, 2001.

J. Krabicka and Y. Yan. Finite-element modeling of electrostatic sensors for the flow

measurement of particles in pneumatic pipelines. *IEEE Transactions on Instrumentation and Measurement*, 58(8):2730–2736, 2009.

J. Krabicka, Y. Yan, and R. M. Carter. Modeling and experimental evaluation of electrostatic sensors for pulverised coal and biomass flow metering. In *Proceedings of the 5th International Symposium on Measurement Techniques for Multiphase Flows*, Macau, 2006.

S. Kurada, G. W. Rankin, and K. Sridhar. A new particle image velocimetry technique for three-dimensional flows. *Optics and Lasers in Engineering*, 28(5):343–376, 1997.

L. Landweber. An iterative formula for fredholm integral equations of the first kind. *Amer. J. Math*, 73:615–624, 1951.

H. Li and Y. Tomita. Particle velocity and concentration characteristics in a horizontal dilute swirling flow pneumatic conveying. *Powder Technology*, 107(1-2):144–152, 2000.

B. G. Liptak. *Flow Measurement*. Chilton Book Company, Radnor, 1993. ISBN 0-8019-8386-X.

S. Liu, Q. Chen, H. G. Wang, F. Jiang, I. Ismail, and W. Q. Yang. Electrical capacitance tomography for gas-solids flow measurement for circulating fluidized beds. *Flow Measurement and Instrumentation*, 16:135–144, 2005.

C. K. K. Lun and H. S. Liu. Numerical simulation of dilute turbulent gas-solid flows in horizontal channels. *International Journal of Multiphase Flow*, 23(3):575–605, 1997.

J. Ma and Y. Yan. Design and evaluation of electrostatic sensors for the measurement of velocity of pneumatically conveyed solids. *Flow Measurement and Instrumentation*, 11(3):195–204, 2000.

O. P. Mahajan. Physical characterisation of coal. *Powder Technology*, 40:1–15, 1984.

S. S. Mallick and P. W. Wypych. Minimum transport boundaries for pneumatic conveying of powders. *Powder Technology*, 194:181–186, 2009.

S. Matsusaka and H. Masuda. Simultaneous measurement of mass flow rate and charge-to-mass ratio of particles in gas-solids pipe flow. *Chemical Engineering Science*, 61:2254–2261, 2006.

S. Matsusaka, H. Umemoto, M. Nishitani, and H. Masuda. Electrostatic charge distribution of particles in gas-solids pipe flow. *Journal of Electrostatics*, 55(1):81–97, 2002.

J. Mavor, J. W. Arthur, and P. B. Denyer. Analogue c.c.d. correlator using monolithic m.o.s.t. multipliers. *Electronics Letters*, 13(12):373–374, 1977.

J. Mennell, B. Byrne, and Y. Yan. Appraisal of radiometric techniques to determine absolute solids fraction in pneumatic suspensions of particulate solids. *Flow Measurement and Instrumentation*, 11(3):213–221, 2000.

F. Mesch and H. Kipphan. Solids flow measurement by correlation methods. *Optical and Quantum Electronics*, 4(4):451–462, 1972.

D. Mills. *Pneumatic conveying design guide*. Elsevier Butterworth-Heinemann, Burlington, 2 edition, 2004. ISBN 978-0750654715.

S. N. Murnane, R. N. Barnes, S. R. Woodhead, and J. E. Amadi-Echendu. Electrostatic modelling and measurement of airborne particle concentration. *IEEE Transactions on Instrumentation and Measurement*, 45(2):488–492, 1996.

P. Olla. Clustering and collision of inertial particles in random velocity fields. *Physical Review*, E77, 2008.

I. J. O'Sullivan and W. M. D. Wright. Ultrasonic measurement of gas flow using electrostatic transducers. *Ultrasonics*, 40(1):407–412, 2002.

L. H. Peng, Y. Zhang, and Y. Yan. Characterization of electrostatic sensors for flow measurement of particulate solids in square-shaped pneumatic conveying pipelines. *Sensors and Actuators A: Physical*, 141(1):59–67, 2008.

A. M. Reynolds. Stokes number effects in lagrangian stochastic models of dispersed two-phase flows. *Journal of Colloid and Interface Science*, 275:328–335, 2004.

M. J. Rhodes. *Introduction to Particle Technology*. John Wiley and Sons, Hoboken, N.J., 2nd edition, 2008. ISBN 978-0470014288.

J. Shao, J. Krabicka, and Y. Yan. Velocity measurement of pneumatically conveyed particles using intrusive electrostatic sensors. *IEEE Transactions on Instrumentation and Measurement*, in print, 2009.

H. M. Shey. *Div, grad, curl and all that: An informal text on vector calculus*. W. W. Norton & Company, Inc., New York, 3rd edition, 1973. ISBN 978-0393969979.

P. P. Silvester. *Finite Elements for Electrical Engineers*. Cambridge University Press, Cambridge; New York, 3 edition, 1996. ISBN 0521445051.

S. J. Tallon and C. E. Davies. The effect of pipeline location on acoustic measurement of gas-solid pipeline flow. *Flow Measurement and Instrumentation*, 11(3):165–169, 2000.

R. Thorn, M. S. Beck, and R. G. Green. Non-intrusive methods of velocity measurement in pneumatic conveying. *J. Phys. E: Sci. Instrum.*, 15(11):1131–1139, 1982.

TR-Tech. www.trtech.com. 2009.

X. S. Wang, M. J. Rhodes, B. M. Gibbs, and D. Geldart. Heat transfer in dilute gas-particle suspensions. *Chemical Engineering Science*, 52(20):3617–3621, 1997.

M. Weber. Principles of hydraulic and pneumatic conveying in pipes. *Bulk Solids Handling*, 1:57–63, 1981.

M. Weber. Application note AN6E: Piezoelectric accelerometers, 2008.

A. J. Weinheimer. The charge induced on a conducting cylinder by a point charge and its application to the measurement of charge on precipitation. *Journal of Atmospheric and Oceanic Technology*, 5:298–304, 1988.

E. W. Weisstein. Gaussian function, 2009a.

E. W. Weisstein. Lorentzian function, 2009b.

R. A. Williams, C. G. Xie, F. J. Dickin, S. J. R. Simons, and M. S. Beck. Multiphase flow measurements in powder processing. *Powder Technology*, 66(3):203–224, 1991.

S. R. Woodhead, A. N. Pittman, and S. J. Ashenden. Laser doppler velocimetry measurements of particle velocity profiles in gas-solid two-phase flows. In *IEEE Conf. Instrumentation and Measurement Technology*, pages 770–773, Piscataway, NJ, 1995. IEEE.

S. R. Woodhead, J. C. Denham, and D. I. Armour-Chelu. Electrostatic sensors applied to the measurement of electric charge transfer in gas-solids pipelines. *Journal of Physics: Sensors & their Applications*, 15:108–112, 2005.

R. Wu. Frequency performance compensation of operational amplifiers with the 't' feedback network. *Analog Integrated Circuits and Signal Processing*, 41:79–83, 2004.

C. G. Xie, A. L. Stott, S. M. Huang, A. Plaskowski, and M. S. Beck. Mass-flow measurement of solids using electrodynamic and capacitance transducers. *J. Phys. E: Sci. Instrum.*, 22:712–719, 1989.

C. Xu, B. Zhou, D. Yang, G. Tang, and S Wang. Velocity measurement of pneumatically conveyed solid particles using an electrostatic sensor. *Meas. Sci. Technol.*, 19, 2007.

Y. Yan. *Mass flow measurement of pneumatically conveyed solids*. PhD thesis, University of Teeside, 1992.

Y. Yan. Mass flow measurement of bulk solids in pneumatic pipelines. *Meas. Sci. Technol.*, 7(12):1687–1706, 1996.

Y. Yan. Continuous measurement of particulate emissions. *IEEE Instrumentation & Measurement Magazine*, October 2005 2005.

Y. Yan and A. R. Reed. Experimental evaluation of capacitance transducers for volumetric concentration measurement of particulate solids. *Flow Measurement and Instrumentation*, 10(1):45–49, 1999.

Y. Yan and D. Stewart. Guide to the flow measurement of particulate solids in pipelines. Technical report, Institute of Measurement and Control, 2001.

Y. Yan, B. Byrne, and J. Coulthard. Sensing field homogeneity in mass flow rate measurement of pneumatically conveyed solids. *Flow Measurement and Instrumentation*, 6(2):115–119, 1995a.

Y. Yan, B. Byrne, S. R. Woodhead, and J. Coulthard. Velocity measurement of pneumatically conveyed solids using electrodynamic sensors. *Meas. Sci. Technol.*, 6: 515–537, 1995b.

A. Yilmaz and E. K. Levy. Roping phenomena in pulverized coal conveying lines. *Powder Technology*, 95:43–48, 1998.

J Zhang and J. Coulthard. Theoretical and experimental studies of the spatial sensitivity of an electrostatic pulverised fuel meter. *Journal of Electrostatics*, (63): 1133–1149, 2005.

J. Q. Zhang and Y. Yan. On-line continuous measurement of particle size using electrostatic sensors. *Powder Technology*, 135-136:164–168, 2003.

Y. Zheng, Q. Liu, Y. Li, and N. Gindy. Investigation on concentration distribution and mass flow rate measurement for gravity chute conveyor by optical tomography system. *Measurement*, 39:643–654, 2006.

Y. Zheng, J. R. Pugh, D. McGlinchey, and E. Knight. Investigation of the heat transfer coefficient for particle mass flow measurement. In *The 9th International Conference on Bulk Materials Storage, Handling and Transportation*, Newcastle, Australia, 2007.

Y. Zheng, J. R. Pugh, D. McGlinchey, and R. O. Ansell. Simulation and experimental study of gas-to-particle heat transfer for non-invasive mass flow measurement. *Measurement*, 41:446–454, 2008.

K. Zhu, S. Madhusudana Rao, Q. H. Huang, C. H. Wang, S. Matsusaka, and H. Masuda. On the electrostatics of pneumatic conveying of granular materials using electrical capacitance tomography. *Chemical Engineering Science*, 59(15):3201–3213, 2004.

Nomenclature

$\dot{m}$	Mass flow rate
ε_0	Permittivity of free space
λ_{ave}	Average delay time
λ_d	Peak cross correlation time delay
λ_n	Upstream to downstream time lag
μ_g	Dynamic viscosity
Φ	Electric potential
ρ	Density
$\rho(t)$	Cross correlation (time domain)
ρ_p	Kinematic viscosity
ρ_q	Charge density
τ_n	Distance from electrode
A	Cross sectional area

a	Scaling constant
A_0	Open loop gain
a_r	Cylinder radius
b	Scaling constant
$C(\omega)$	Autocorrelation
$C(t)$	Autocorrelation
C_f	Feedback capacitance
c_r	1/2 axial length of cylindrical electrode
C_V	Volume concentration fraction
D	Electric displacement field
d_p	Particle diameter
E	Electric field
$F(\omega)$	Fourier transform
F_{3dB}	-3dB Cutoff frequency
F_c	Cutoff frequency, first zero crossing of the spectral magnitude
I_{in}	Input current
J_s	Bessell function of order s
k	Scaling constant
$k[n]$	Periodic noise, discrete time

K_b	Meter constant
L	Electrode length
L_c	Characteristic length of obstruction
N	Number of samples
n	Index
$P(\omega)$	Cross correlation
q	Electric charge
q'	Induced charge
q_{in}	Input charge
R	Pipeline radius
r	Distance from charge
$r[j]$	Cross correlation at lag j, discrete time
R_1, R_2	Resistor values
r_0	Radial component in a cylindrical coordinate system
r_{diff}	Cross correlation of the difference between upstream and downstream signals
R_f	Feedback resistance
r_{noisy}	Noisy signal
s	Electrode spacing
t	Time

U Free stream velocity

V Volume

v Velocity

V_1, V_2 Circuit node voltages

v_{ave} Average velocity

V_{in} Input voltage

V_{max} Maximum streamline velocity

V_{out} Output voltage

V_{ref} Reference voltage

V_r Streamline velocity

W Ring electrode axial length

X_1, X_2 Upstream, downstream signals, discrete time

x_n n^{th} zero of J_0

z_0 Axial component in a cylindrical coordinate system

Z_t Total input impedance

Publications from this work

1. J. Krabicka and Y. Yan. Finite-element modelling of electrostatic sensors for the flow measurement of particles in pneumatic pipelines. *IEEE Transactions on Instrumentation and Measurement*, 58(8): 2730–2736, 2009

2. J. Krabicka and Y. Yan. Optimised design of intrusive electrostatic sensors for the velocity measurement of pneumatically conveyed particles. In I2MTC, Singapore, 2009

3. J. Shao, J. Krabicka, and Y. Yan. Velocity measurement of pneumatically conveyed particles using intrusive electrostatic sensors. *IEEE Transactions on Instrumentation and Measurement*, In Print, 2009.

4. R. M. Carter, Y. Yan, J. Krabicka, M Whitehouse, S Cornwell, and G Riley. An integrated sensor system for combustion plant optimisation. In International Conference on Coal Combustion Science and Technology, Nottingham, 28-31 August, 2007.

5. J. Q. Shao, J. Krabicka, and Y. Yan. Comparative studies of electrostatic sensors with circular and probe electrodes for the velocity measurement of pulverised coal. *Chinese Journal of Scientific Instruments*, 28, No. 11 1921–1926, 2007

6. J. Krabicka and Y. Yan. Finite element modelling of intrusive electrostatic sensors for the measurement of pulverised fuel flows. In Proceedings of IEEE Instrumentation and Measurement Technology Conference, Warsaw, 2007.

7. J. Krabicka, Y. Yan, and R. M. Carter. Modelling and experimental evaluation of electrostatic sensors for pulverised coal and biomass flow metering. Proceedings of the 5th International Symposium on Measurement Techniques for Multiphase Flows, Macau, 2006.

Appendix A

Electronics and hardware schematics

The following are schematics for the electronics and hardware used in the tests on the 500 kW test rig in Didcot, the flow test rig at PCME, and the flow test rig at the University of Kent. The hardware used for the 4 MW power plant tests was not constructed by the author.

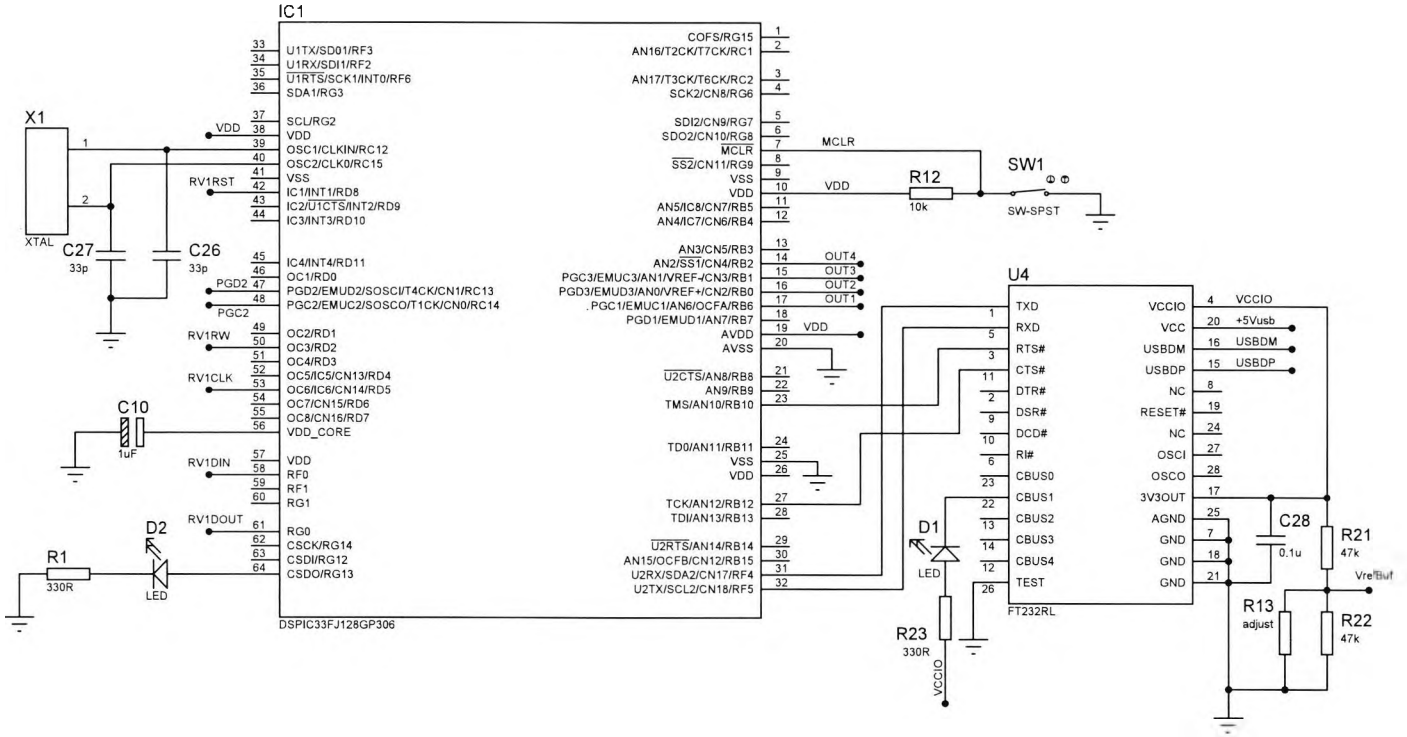


Figure A.1: Sensor electronics schematic (1 of 4)

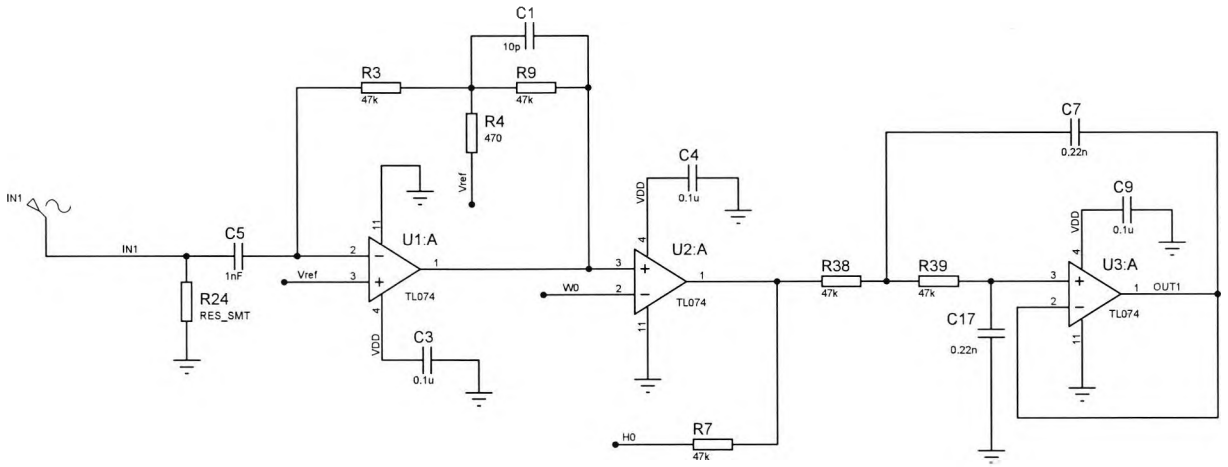


Figure A.2: Sensor electronics schematic (2 of 4)

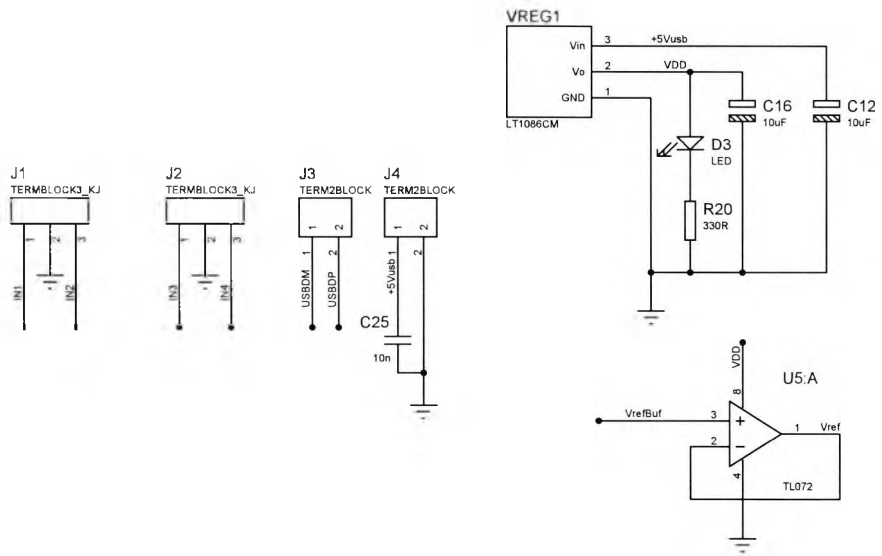


Figure A.3: Sensor electronics schematic (3 of 4)

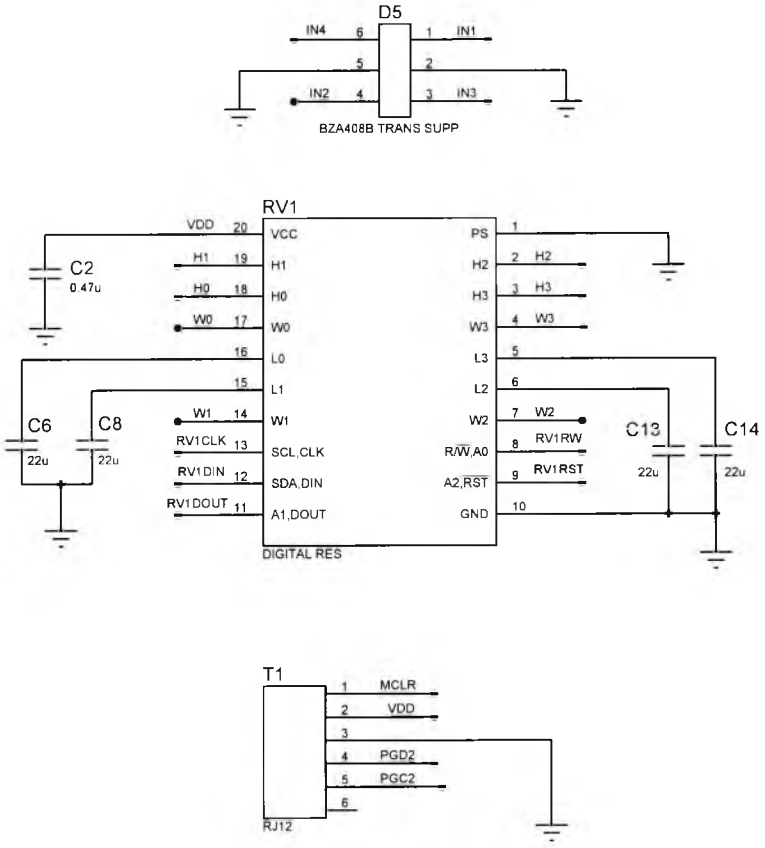


Figure A.4: Sensor electronics schematic (4 of 4)

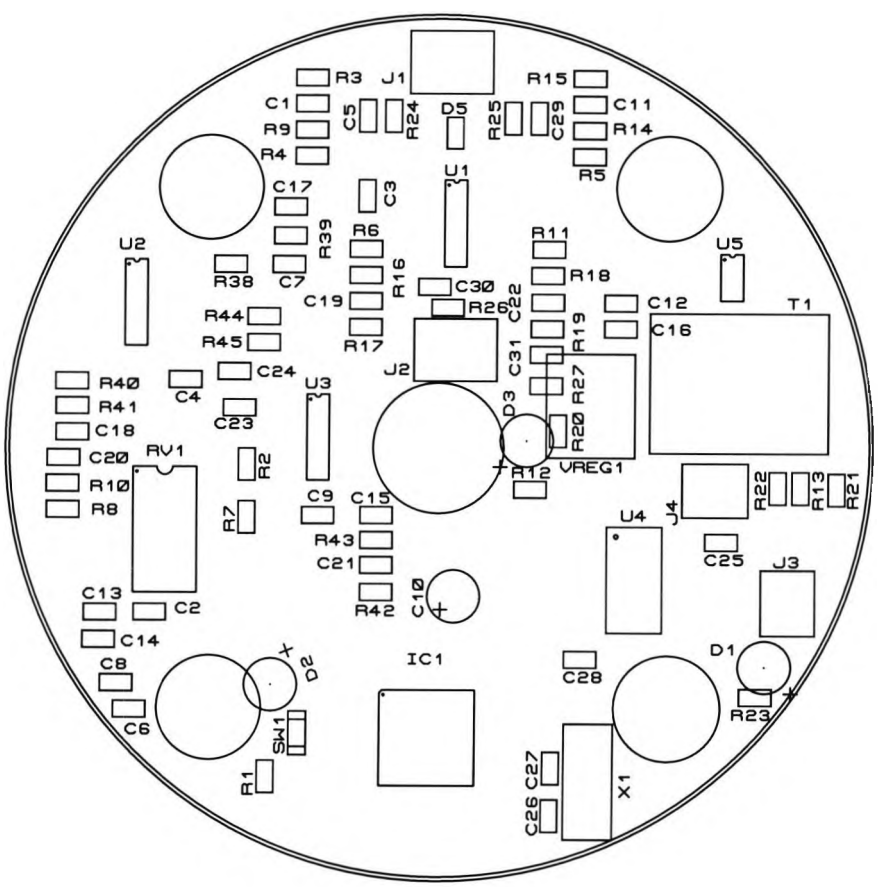


Figure A.5: Sensor electronics PCB layout—top silk

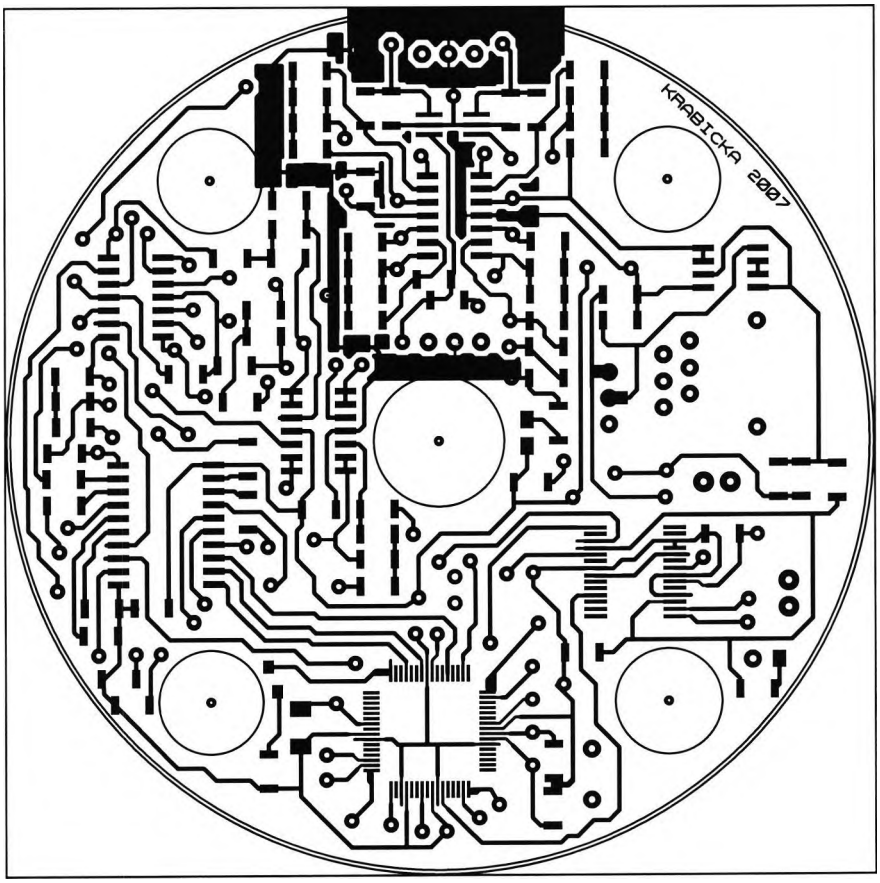


Figure A.6: Sensor electronics PCB layout—top copper

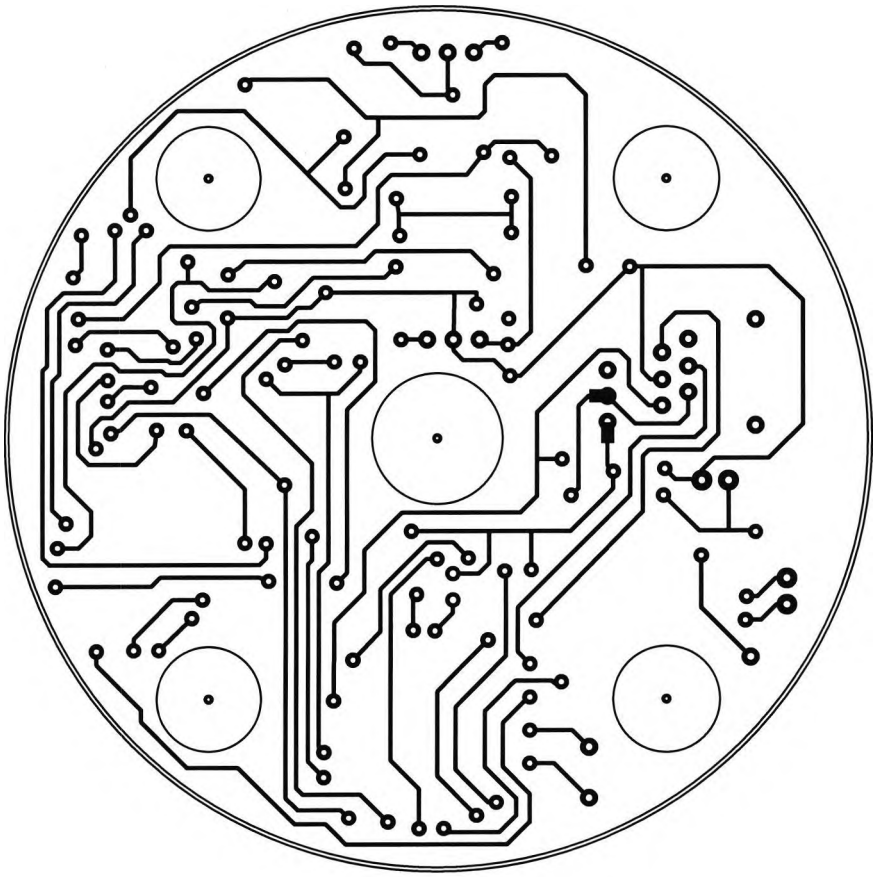


Figure A.7: Sensor electronics PCB layout—bottom copper

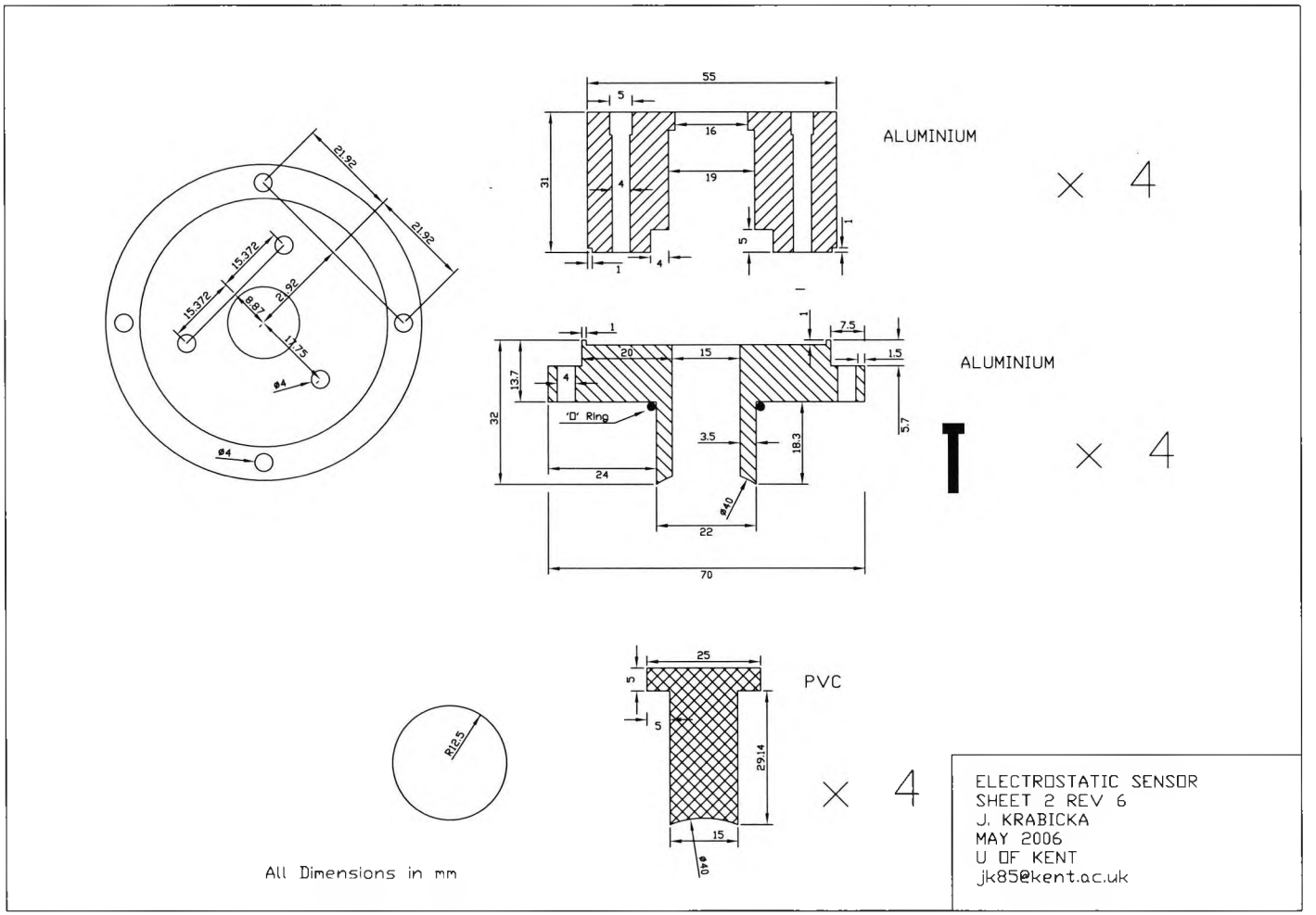


Figure A.8: Sensor hardware for 40 mm pipeline—components

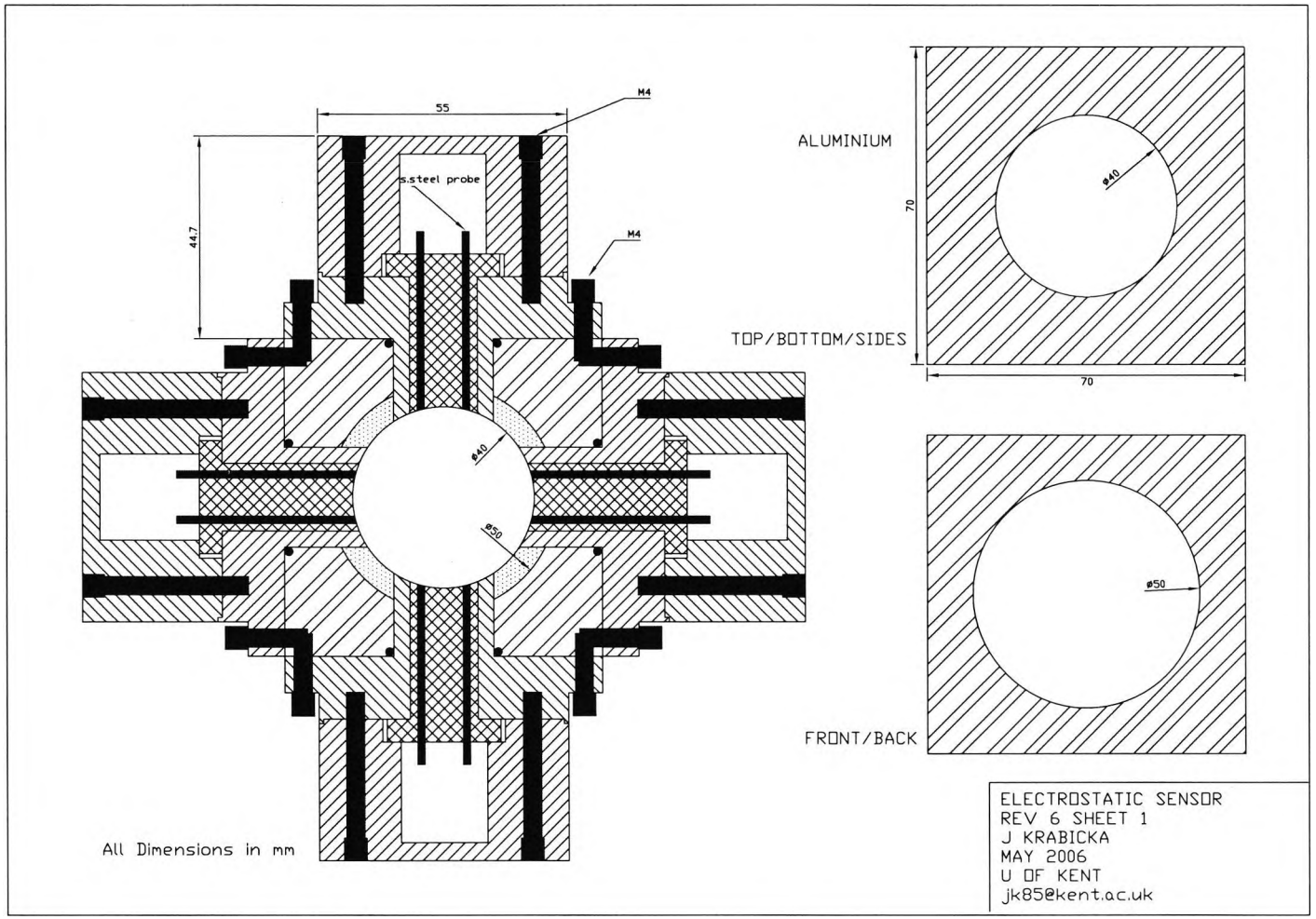


Figure A.9: Sensor hardware for 40 mm pipeline—assembled

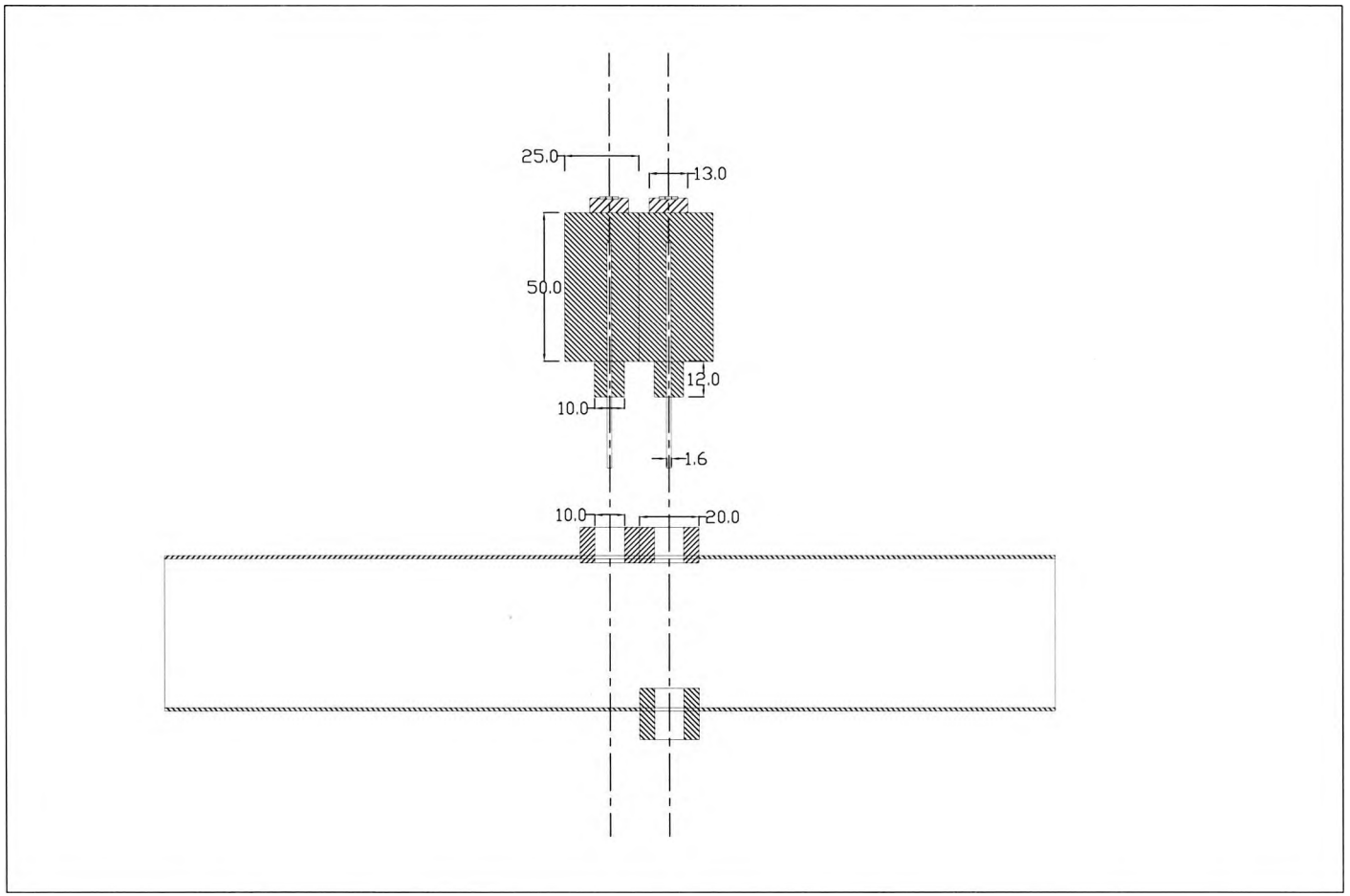


Figure A.10: Sensor hardware for 48 mm pipeline—components

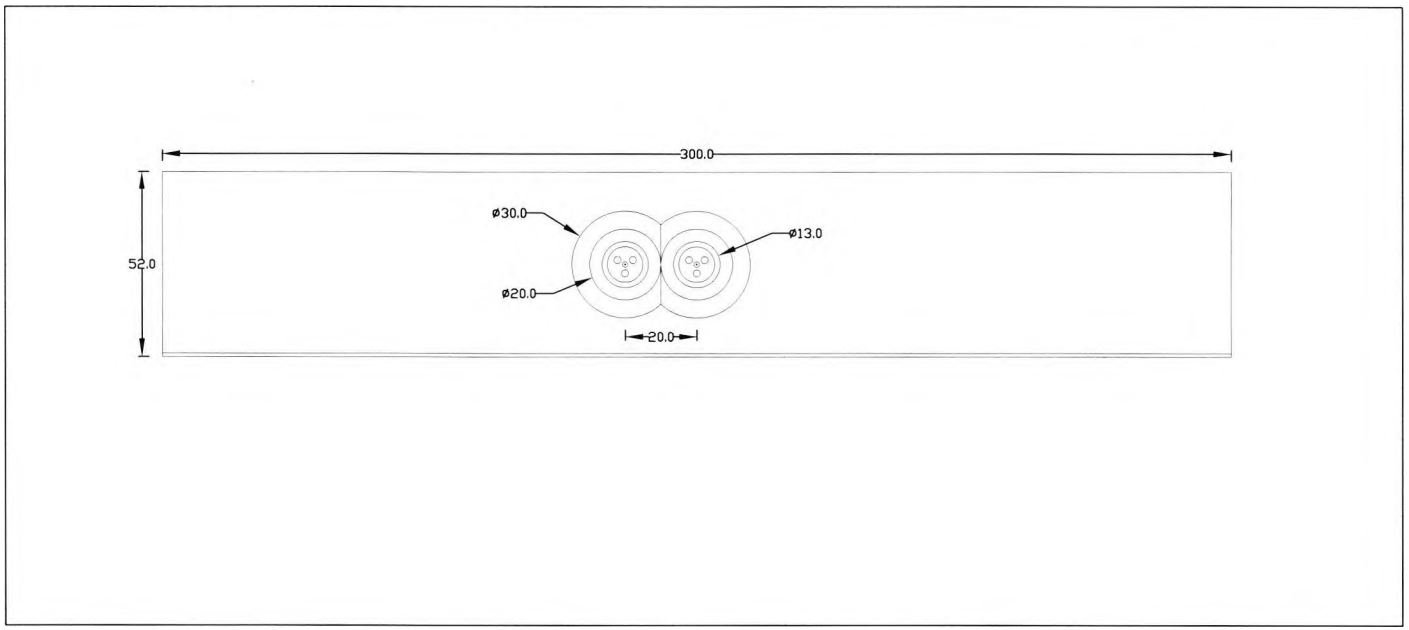


Figure A.11: Sensor hardware for 48 mm pipeline—assembled

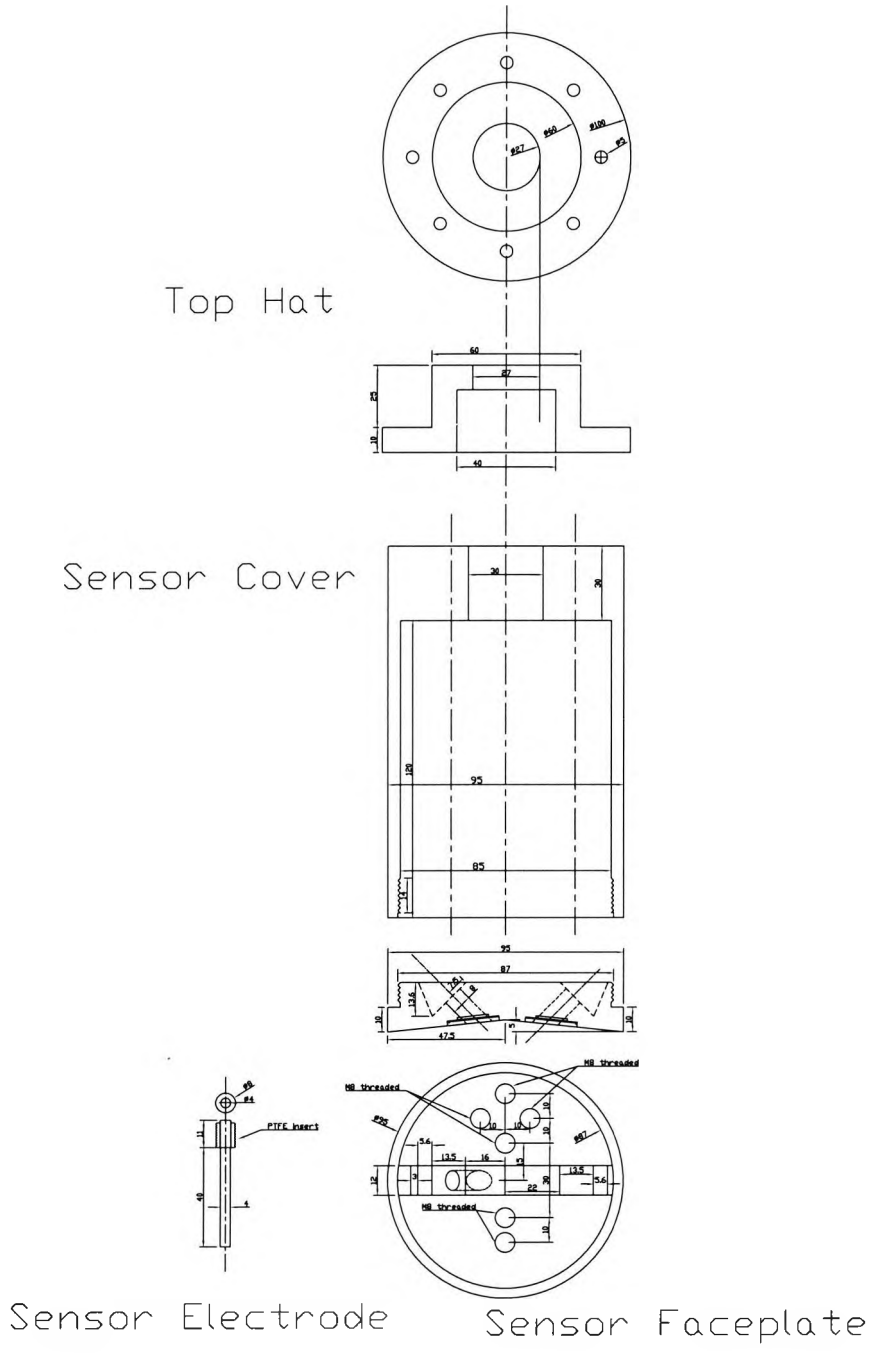


Figure A.12: Sensor hardware for 300 mm/500 mm pipelines—components

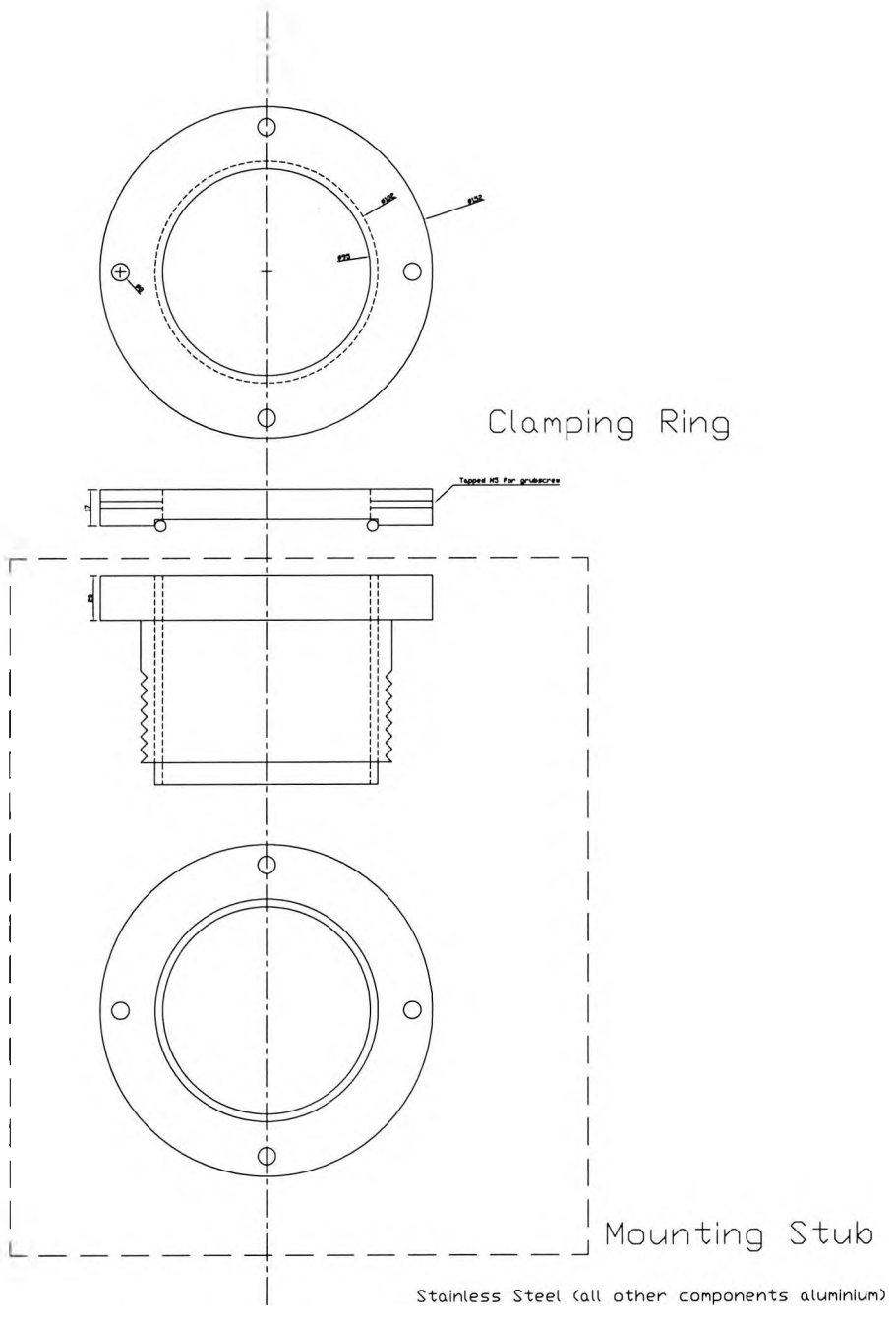


Figure A.13: Sensor hardware for 300 mm/500 mm pipelines—mounting stub

Appendix B

Comsol® code

The following is an example of FEM simulation code written for Comsol®3.4 software, and used to calculate the charge induced onto a sensor electrode by a point charge at a given location. The preliminary code is automatically generated by the software to encode the physical configuration designed using the graphical user interface.

```
% COMSOL Multiphysics Model M-file
% Calculates charge induced onto sensor electrode
% by point charge at location x,y,z
% Result is stored in matrix I1

flclear fem

% COMSOL version
clear vrsn
vrsn.name = 'COMSOL 3.4';
vrsn.ext = '';
vrsn.major = 0;
vrsn.build = 248;
vrsn.rcs = '$Name:  $';
vrsn.date = '$Date: 2007/10/10 16:07:51 $';
fem.version = vrsn;

flbinaryfile='justOneRoundProbeMinimal_x19.mphm';

% Geometry
clear draw
g7=flbinary('g7','draw',flbinaryfile);
g1=flbinary('g1','draw',flbinaryfile);
g3=flbinary('g3','draw',flbinaryfile);
draw.s.objs = {g7,g1,g3};
draw.s.name = {'CO5','CO1','CO3'};
draw.s.tags = {'g7','g1','g3'};
fem.draw = draw;
fem.geom = geomcsg(fem);
```

```
x=15;
yrange=0:19;
zrange=[10:39 40:0.5:50];
for ycoord=1:length(yrange)
    for zcoord=1:length(zrange)

    % Geometry
    parr={point3(x,yrange(ycoord),zrange(zcoord))};
    g2=geomcoerce('point',parr);


    % Analyzed geometry
    clear p s
    p.objs={g2};
    p.name={'PT1'};
    p.tags={'g2'};

    s.objs={g7,g1,g3};
    s.name={'C05','C01','C03'};
    s.tags={'g7','g1','g3'};

    fem.draw=struct('p',p,'s',s);
    fem.geom=geomcsg(fem);

    % Initialize mesh
    fem.mesh=meshinit(fem, ...
                      'hauto',5, ...
                      'hpnt',20, ...
                      'xscale',1, ...
                      'yscale',1, ...
                      'zscale',1, ...
                      'jiggle','on', ...
                      'methodfac',{1,'tri',2,'tri',3,'tri',4,'tri',5,'tri', ...
        6,'tri',7,'tri',8,'tri',9,'tri',10,'tri', ...
        11,'tri',12,'tri',13,'tri',14,'tri',15,'tri', ...
        16,'tri',17,'tri',18,'tri',19,'tri',20,'tri', ...
        21,'tri',22,'tri',23,'tri',24,'tri',25,'tri', ...
        26,'tri',27,'tri',28,'tri',29,'tri',30,'tri'});

    % Application mode 1
    clear appl
    appl.mode.class = 'EmElectrostatics';
    appl.mode.type = 'cartesian';
    appl.dim = {'V'};
    appl.sdim = {'x','y','z'};
    appl.name = 'emes';
    appl.module = 'ACDC';
    appl.shape = {'shlag(2,''lm1'')','shlag(2,''V'')'};
    appl.gporder = {10,4};
    appl.cporder = 2;
    appl.sshape = 2;
    appl.border = 'on';
    appl.assignsuffix = '_emes';
    clear prop
    prop.elemdefault='Lag2';
    prop.input='nD';
    prop.frame='ref';
    clear weakconstr
    weakconstr.value = 'on';
    weakconstr.dim = {'lm1'};
    prop.weakconstr = weakconstr;
    prop.constrtype='ideal';
```

```
appl.prop = prop;
clear pnt
pnt.Q0 = {0,1};
pnt.V0 = 0;
pnt.name = '';
pnt.type = 'Q';
pnt.ind = [1,1,1,1,1,1,1,1,1,1,1,1,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, ...
  1,1,1];
appl.pnt = pnt;
clear edg
edg.V0 = 0;
edg.type = 'Q';
edg.Ql = 0;
edg.ind = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, ...
  1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1];
appl.edg = edg;
clear bnd
bnd.inport = 0;
bnd.chsrcdst = 0;
bnd.index = 0;
bnd.wcshape = 1;
bnd.V0 = 0;
bnd.pertype = 'sym';
bnd.wcinit = 0;
bnd.type = {'V0','r','V0'};
bnd.portnr = 1;
bnd.weakconstr = 1;
bnd.Q0 = 0;
bnd.name = {'pipe_air_pvc_gnd','pvcInterior','probeA'};
bnd.rhos = 0;
bnd.Vref = 0;
bnd.wcgporder = 1;
bnd.D0 = {{0;0;0}};
bnd.epsilonrbnd = 1;
bnd.dbnd = 1;
bnd.ind = [1,1,2,2,1,1,3,3,3,3,3,2,1,1,2,1,1,1,3,3,1,3,3,1,3,3,3,3,3,3];
appl.bnd = bnd;
clear equ
equ.Sd = {{'Sdx_guess_emes';'Sdy_guess_emes';'Sdz_guess_emes'}};
equ.srcpnt = {{0;0;0}};
equ.S0 = {{'S0x_guess_emes';'S0y_guess_emes';'S0z_guess_emes'}};
equ.gporder = 2;
equ.coord0n = {{0;0;0}};
equ.P = {{0;0;0}};
equ.epsilonr = {'mat2_epsilonr',1,'mat3_epsilonr'};
equ.name = '';
equ.nTsrcaxis = {{{0;0;1}}};
equ.rho = 0;
equ.dr = 'dr_guess_emes';
equ.r0n = 0;
equ.cporder = 1;
equ.maxwell = {{}};
equ.init = 0;
equ.Stype = 'none';
equ.shape = 2;
equ.R0 = 'R0_guess_emes';
equ.elconstrel = 'epsr';
equ.Dr = {{0;0;0}};
equ.usage = 1;
equ.srcaxis = {{0;0;1}};
equ.user = {{0;0;0}};
equ.nTsrcpnt = {{{0;0;0}}};
```

```
equ.ind = [2,1,3,1,1];
appl.equ = equ;
appl.var = {'epsilon0','8.854187817e-12'};
fem.appl{1} = appl;
fem.sdim = {'x','y','z'};
fem.frame = {'ref'};

% Simplify expressions
fem.simplify = 'on';
fem.border = 1;
clear units;
units.basesystem = 'SI';
fem.units = units;

% Scalar expressions
fem.expr = {};

% Global expressions
fem.globalexpr = {};

% Functions
clear fcns
fem.functions = {};

% Solution form
fem.solform = 'weak';

% Library materials
clear lib
lib.mat{1}.name='Air, 1 atm';
lib.mat{1}.varname='mat1';
lib.mat{1}.variables.nu0='nu0(T[1/K])[m^2/s]';
lib.mat{1}.variables.eta='eta(T[1/K])[Pa*s]';
lib.mat{1}.variables.C='Cp(T[1/K])[J/(kg*K)]';
lib.mat{1}.variables.rho='rho(p[1/Pa],T[1/K])[kg/m^3]';
lib.mat{1}.variables.k='k(T[1/K])[W/(m*K)]';
clear fcns
fcns{1}.type='inline';
fcns{1}.name='nu0(T)';
fcns{1}.expr='(-7.887E-12*T^2+4.427E-08*T+5.204E-06)/(1.013e5*28.8e-3/8.314/T)';
fcns{1}.dexpr={'diff((-7.887E-12*T^2+4.427E-08*T+5.204E-06)/ ...
     (1.013e5*28.8e-3/8.314/T),T)'};
fcns{2}.type='inline';
fcns{2}.name='Cp(T)';
fcns{2}.expr='0.0769*T+1076.9';
fcns{2}.dexpr={'diff(0.0769*T+1076.9,T)'};
fcns{3}.type='inline';
fcns{3}.name='rho(p,T)';
fcns{3}.expr='p*28.8e-3/8.314/T';
fcns{3}.dexpr={'diff(p*28.8e-3/8.314/T,p)','diff(p*28.8e-3/8.314/T,T)'};
fcns{4}.type='inline';
fcns{4}.name='eta(T)';
fcns{4}.expr='-7.887E-12*T^2+4.427E-08*T+5.204E-06';
fcns{4}.dexpr={'diff(-7.887E-12*T^2+4.427E-08*T+5.204E-06,T)'};
fcns{5}.type='inline';
fcns{5}.name='k(T)';
fcns{5}.expr='10^(0.8616*log10(abs(T))-3.7142)';
fcns{5}.dexpr={'diff(10^(0.8616*log10(abs(T))-3.7142),T)'};
lib.mat{1}.functions = fcns;
lib.mat{2}.name='Steel AISI 4340';
lib.mat{2}.varname='mat2';
lib.mat{2}.variables.nu='0.28';
```

```
lib.mat{2}.variables.E='205e9[Pa]';
lib.mat{2}.variables.mur='1';
lib.mat{2}.variables.sigma='4.032e6[S/m]';
lib.mat{2}.variables.epsilonr='1';
lib.mat{2}.variables.alpha='12.3e-6[1/K]';
lib.mat{2}.variables.C='475[J/(kg*K)]';
lib.mat{2}.variables.rho='7850[kg/m^3]';
lib.mat{2}.variables.k='44.5[W/(m*K)]';
lib.mat{3}.name='Nylon';
lib.mat{3}.varname='mat3';
lib.mat{3}.variables.E='2e9[Pa]';
lib.mat{3}.variables.epsilonr='4';
lib.mat{3}.variables.alpha='280e-6[1/K]';
lib.mat{3}.variables.C='1700[J/(kg*K)]';
lib.mat{3}.variables.rho='1150[kg/m^3]';
lib.mat{3}.variables.k='0.26[W/(m*K)]';


fem.lib = lib;

% Multiphysics
fem=multiphysics(fem);

% Extend mesh
fem.xmesh=meshextend(fem, ...
                     'geoms',[1], ...
                     'eqvars','on', ...
                     'cplbndeq','on', ...
                     'cplbndsh','off', ...
                     'linshape',[1], ...
                     'linshapetol',0.1);

% Solve problem
fem.sol=femstatic(fem, ...
                  'method','eliminate', ...
                  'nullfun','auto', ...
                  'blocksize',5000, ...
                  'complexfun','off', ...
                  'matherr','on', ...
                  'solfile','off', ...
                  'conjugate','off', ...
                  'symmetric','auto', ...
                  'solcomp',{'V','lm1'}, ...
                  'outcomp',{'V','lm1'}, ...
                  'rowscale','on', ...
                  'ntol',1e-006, ...
                  'maxiter',25, ...
                  'nonlin','auto', ...
                  'damping','on', ...
                  'hnlin','off', ...
                  'linsolver','pardiso', ...
                  'pardreorder','nd', ...
                  'pardrreorder','on', ...
                  'pivotperturb','1e-008', ...
                  'itol',1e-006, ...
                  'rhob',400, ...
                  'errorchk','on', ...
                  'uscale','auto', ...
                  'mcase',0);

% Save current fem structure for restart purposes
fem0=fem;
```

```
    % Integrate
    I1(x,ycoord,zcoord)=-postint(fem,'nD_emes', ...
                'unit','C', ...
                'dl',[7,8,9,10,11,19,20,22,23,25,26,27,28,29,30], ...
                'edim',2, ...
                'intorder',4, ...
                'geomnum',1, ...
                'solnum','end', ...
                'phase',0);
            I1(x,ycoord,zcoord)
            [yrange(ycoord) zrange(zcoord) toc]
    end
end

% I1(x,y,z) contains the charge induced onto the electrode
% by a point charge at (x,y,z)
```

Appendix C

dsPIC® microcontroller code

The following code was written for the dsPIC® 33FJ128GP306 signal processing microcontroller, embedded in the electrostatic sensor head. The user interface, data logging, and data presentation functions are performed via communications with a separate Windows® application. None of this code is automatically generated.

Main program loop

```
//Main loop for sensor signal acquisition and processing
//Data is communicated to PC via USB link, and Window interface kj3.vproj

//define codes used in main case statement
#define RESULTS 254 // send results
#define RAW 247  // send raw data
#define CORR 246 // send cross correlation data
#define NO_OTHER_DATA 245
#define AUTOGAIN 252
#define SENSOREXIT 249
#define SET_FS 248
#define SAMPLING_FREQUENCY 2000 //  in Hz, can be changed in Windows user interface
#define CMNR 244 // common mode noise rejection
#define NO_CMNR 243

#include "globalConstants.h"
#include "dsp.h"
#include  "p33FJ128GP306.h"
#include <stdio.h>
#include "ioDefined.h"

//memory allocation:    cross check the memory locations assigned (see watch window)
            // with memory map (see datasheet)
            //make sure no variables cross over into DMA memory!
```

```
fractional corrPeak1[CORR_PEAK_LENGTH] //to hold correlation result
__attribute__ ((section (".ydata, data, ymemory"),far, aligned (2)));

fractional s1[Nsamples] __attribute__((section(".ydata, data, ymemory"),far,aligned(2)));
fractional s2[Nsamples] __attribute__((section(".xdata, data, xmemory"),far,aligned(2)));

//Globals to hold data to be transmitted to PC:
char s1Pwr,s2Pwr=0;
float v1=0;
float c1=0; //maximum correlation values
int cScalei1=32767; //scaler to for c1 values to get the correlation coefficient.
unsigned char controlByte=0;
unsigned char whatData=0;
char gainSetting=0;
long  Fs=SAMPLING_FREQUENCY;

int main(void)
{
  unsigned int i = 0; //general purpose counter

  float v1mean; //mean velocitie
  float c1mean; //mean correlation coefficient
  char validReadings1;
  float v1Prev;
  int eOrient=0; //electrode orientation, i.e. forward (0) or reversed (1)
  int cmnr=0;  //common mode noise compensation false


  //configuration bits to select
    // oscillator parameters done in IDE using configure ==> configuration bits
  //select primary oscillator (XT) with PLL  (XTPLL)
  //set oscillator/pre/postscalers etc. to give Fosc of 80MHz, therefore 40MIPS
  //see dspic33f p.151  example for details
  //N.B. this assumes that a 10MHz crystal is being used
  CLKDIVbits.PLLPRE=0;
  PLLFBDbits.PLLDIV=0x1E;
  CLKDIVbits.PLLPOST=0;

  //set up the i/o registers:
  IOsetup();

  //set up the UART:
  UART_setup();

  //set up the a/d converter:
  ADsetup_kj(Fs);


  //determine electrode orientation (in case sensor is installed wrong way around):
  //eOrient=findElectrodeOrientation(&s1[0],&s2[0],&corrPeak1[0],&cScalei1,Fs);

  //test UART transmission:
  //UARTtest();

  //test UART reception:
  //UARTrecieveTest(&s1[0],Nsamples);

  //set the digital resistor
  //gainSetting=autoGain(&s1[0], &s2[0], &s3[0], &s4[0]);

  //*** use flashLED  or adcFlasher (higher input voltage=higher frequency flash)
```

```
//    to confirm dsPIC operation
//
//flashLED();  //test to see if dsPIC working ok. note that this lasts indefinitely,
//         and subsequent code is not executed
//adcFlasher();

while(1)  //INFINITE LOOP
{
  LED=1; //TURN ON LED TO INDICATE WAITING FOR RECIEVE
  while(!U2STAbits.URXDA) {}; //WAIT WHILE RECIEVE BUFFER EMPTY
  LED=0; //TO INDICATE PROCESSING

  RCON=0;  //reset indicator flags
  controlByte=U2RXREG;
  v1mean=0; //reset
  c1mean=0;
  validReadings1=0;

  switch(controlByte)
  {
    case RESULTS:
    {
        while(!U2STAbits.URXDA) {;} //wait for byte to indicate what data to send
        whatData=U2RXREG;

        for(i=0;i<AVERAGING;i++)
        {
          /*get samples from adc
            the argument list:
            Nsamples determines how many samples on each sensor are to be taken
            &s1[0] is the starting address of s1 (sensor 1) data.
            &s2[0] is the starting address of s2
            Could be written as just s1, but less clear
            etc.. */
          getSamples_kj(Nsamples,Fs,&s1[0],&s2[0],ELECTRODEa,eOrient,cmnr);
          //velocity result from electrode set 1
          sigProcB(s1,s2,Fs,&corrPeak1[0],spacingA,&c1,&cScalei1,&v1,&s1Pwr,&s2Pwr);
          //send the first 'batch' of raw data (i.e. when i==0)
          if(whatData==RAW && i==0) {sendRaw(&s1[0],&s2[0]);}
          //only consider velocity and correlation if minimum correlation is met:
          if (c1>=MINCORR) {v1mean=v1mean+v1; c1mean=c1mean+c1; validReadings1++;}
        }//for AVERAGING

      v1mean=v1mean/(float)validReadings1;
      c1mean=c1mean/(float)validReadings1;

      //set results to 0 if no valid readings
      if(validReadings1==0) {v1mean=0;c1mean=0;}//add this code for orientation check:
      //eOrient=findElectrodeOrientation(&s1[0],&s2[0],&corrPeak1[0],&cScalei1,Fs);

      if(whatData==CORR) {sendCorr(&corrPeak1[0],&cScalei1);}

      //NOTE, ONLY THE RAW DATA FOR THE LAST AVERAGING ITERATION ARE SENT:
      sendResultsC(v1mean,c1mean,s1Pwr,s2Pwr,gainSetting,validReadings1);
      break;
      }//case RESULTS

      case AUTOGAIN:
      {
      gainSetting = autoGain(Fs,&s1[0],&s2[0],ELECTRODEa,eOrient);
      sendUARTdata(&gainSetting,1); //SEND THE SETTING RESULT
      break;
```

```
        }

        case SENSOREXIT:
        {
        asm("RESET");
        break;
        }

        case SET_FS:
        {
        while(!U2STAbits.URXDA) {}; //WAIT WHILE RECIEVE BUFFER EMPTY
        Fs=U2RXREG*1000; //convert from kHz to hz
        break;
        }

        case CMNR: {cmnr=1;break;}  //apply common mode noise compensation

        case NO_CMNR: {cmnr=0;break;} //do not apply common mode noise compensation


        case 255: {break;}

        default:  //MANUAL DIGITAL RESISTOR SETTING
        {
        //ensure resistor setting is not above maximum allowed
        if(controlByte>62) {controlByte=62;}
        gainSetting=controlByte;
        digitalResSet(controlByte);
        sendUARTdata(&controlByte,1);  //CONFIRM SETTING TO MAIN PROGRAM
        }//default

      }//switch
  } //INFINITE LOOP

}
```

Supporting files

acdFlasher.c

```
//flashes according to adc value. higher value=faster flash
#include <dsp.h>
#include "fft.h"
#include "ioDefined.h"
#include  "p33FJ128GP306.h"

void delay_us(long d);
void adcFlasher()

{
long delayValue;
int* adcPtr;

TMR2=0; //initialize timer 2
```

```
T2CONbits.TON=1; //start timer2
adcPtr=(int*)(&_DMA_BASE+0);    //***use +0 for ch1, +1 for ch2, +2 for ch3, or +3 for ch4

while(1)
{
  while(AD1CON1bits.DONE==0) //wait until adc conversion is done
  {
  asm("nop");
  }
  AD1CON1bits.DONE=0;

  delayValue=*adcPtr;
  delayValue=(0xFF-delayValue)*1961; //for lowest flash freq 1Hz

  LED=1;  //flash LED on and off
  delay_us(delayValue);
  LED=0;
  delay_us(delayValue);
  }
}
```

ADsetup_kj.c

```
//sets up a/d converter

#include "p33FJ128GP306.h"
#include "globalConstants.h"
void ADsetup_kj(int Fs)
{

  AD1CON1bits.ADON = 0x0; //setting to 1 turns on the ADC module.  Keep off while changing settings
  AD1CON1bits.ADSIDL = 0x0;  //Continue module operation in idle mode
  AD1CON1bits.ADDMABM = 0x1;  //DMA buffers are written in the order of conversion
  AD1CON1bits.AD12B = 0x0 ; //10-bit 4-channel mode
  AD1CON1bits.FORM = 0x0; //0x2 output from ADC is in fractional format (dddd dddd dd00 0000).
  //  3=signed fractional, 1=signed integer, 0=integer
  //NB:whatever you load into the adc buffer in simulation (using register injection)
  // gets shifted over to the left by 6 bits when the adc buffer is read
  //  AD1CON1bits.SSRC = 0x7; //internal counter ends sampling and starts conversion (auto-convert)
  AD1CON1bits.SSRC = 0x2; //GP timer (Timer 3 for ADC1) compare ends sampling and starts conversion
  AD1CON1bits.SIMSAM = 0x1;//samples ch0-ch3 simultaneously (when CHPS=1x)
                           // setting SIMSAM=0 samples in sequence
  //  AD1CON1bits.ASAM = 0x0; //sampling begins when SAMP bit is set
  AD1CON1bits.ASAM = 0x1; //sampling begins immediatedly after last conversion is done
  AD1CON1bits.SAMP = 0x0; //1=ADC sample/hold amps are sampling, 0=ADC sample/hold amps are holding
  AD1CON1bits.DONE = 0x0; //ADC conversion status bit


  AD1CON2bits.VCFG = 0x0; //voltage reference is Avdd and Avss
  AD1CON2bits.CSCNA = 0x0; //input scan bit. Do not scan inputs
  AD1CON2bits.CHPS = 0x2; //converts ch0-ch3. 0x1 converts ch0 and ch1, 0x0 converts only ch0
  AD1CON2bits.SMPI = 0x0; //0 increments DMA address after every sample/conversion operation
  AD1CON2bits.BUFM = 0x0;//always start filling the buffer from the start address
  AD1CON2bits.ALTS = 0x0; //always uses channel input selects for sample A

  AD1CON3bits.ADRC = 0x0; //clock derived from system clock
```

```
  AD1CON3bits.SAMC = 0xC; //auto sample time bits, 0xC=12 Tad.  After this, the conversion starts
  AD1CON3bits.ADCS = 0xB; //conversion clock select. see notes.txt


  AD1CON4 = 0x0000; //DMA setting --Allocates 1 word of buffer to each analog input
  AD1CHS123 = 0x0000; //ch1 to ch3 positive A input is AN0, AN1, and AN2
  AD1CHS0 = 0x0006; //ch0 positive A input is AN6
  AD1CSSH = 0x0000; //select channels for input scan.  None selected since don't want scan
  AD1CSSL = 0x0000; //select channels for input scan.
  AD1PCFGH = 0x0000;  //select all analog rather than digital ports
  AD1PCFGL = 0x4100;  //

  INTCON1bits.NSTDIS = 1;   // Disable interrupt nesting
  IEC0bits.AD1IE = 0;      // Disable ADC1 interrupts
  IFS0bits.AD1IF = 0;      // Clear the A/D interrupt flag bit

  AD1CON1bits.ADON = 1; //start adc

  // DMA0 configuration
  // Direction: Read from peripheral address 0x300 (ADC1BUF0) and write to DMA RAM
  // IRQ: ADC Interrupt

  DMA0CONbits.CHEN=0;    // disable dma channel 0
  DMA0CONbits.SIZE=0;    // word size data (1=byte size data)
  DMA0CONbits.DIR=0;     // read from peripheral address
  DMA0CONbits.HALF=0;    // initiate block transfer complete interrupt when all data moved
  DMA0CONbits.NULLW=0;   // don't null write
  DMA0CONbits.AMODE=0;   // register indirect with post increment mode
  DMA0CONbits.MODE=0;    // continuous, ping-pong disabled
  DMA0REQ=13;            // IRQ# for ADC1
  DMA0PAD=0x0300;          // ADC1BUF0 address
  DMA0CNT=3;             // this value + 1 dma requests per block transfer
  DMA0STA=0x0000;          // BUFFER A: DMA_BASE[0] to DMA_BASE[3]

  IFS0bits.DMA0IF = 0;       // clear the DMA interrupt flag bit
  IEC0bits.DMA0IE = 0;       // disable DMA0 interrupts

  DMA0CONbits.CHEN=1;        // now that everything is set up, enable dma channel 0

  //Set up timer 2
  T2CONbits.TON=0; // don't start timer yet
  T2CONbits.TSIDL=0; //continue module operation in idle mode
  T2CONbits.TGATE=0; //gated time accumulation disabled
  T2CONbits.TCKPS=0; //prescale 1:1
  T2CONbits.T32=1;  //Timer2 and 3 act as one 32-bit timer
  T2CONbits.TCS=0; //internal clock source (Fcy)

  //Set up timer2 period

  PR3=0; //most sig bits of the 32-bit timer 2/3
  PR2= (int)(FCLOCK/Fs)-1;
}
```

autogain.c

```
//setting resistor gain:
```

```
#define POWERSAMPLES 1024
#define gTarget 1024    //peak = approx 6 to 8 times std. use 8, so std=peak/8
            //(or for sine it is 2sqrt(2) = 2.83)
            //pwr is std^2,  so pwr=(peak/8)^2
            //peak to peak is fractional) 1023(=0.0156) for the signed 10 bit input,
            // so maximum pwr =(0.0156/8)^2 =  3.7998e-006
            //The VectorPower function does not scale by 1/N or 1/(N-1),
            //so unscaled value is   3.7998e-006 * Nsamples.
            //This works out to 0.0039, or fractional 127.5 for 1024 samples;
            //(or for sin, it is 1020)
            //(or for sin, and 2048 samples, it is 2042)
            // See Mean,Standard Deviation,Signal Power, Computation Forms.pdf tutorial
#include  "p33FJ128GP306.h"
#include "dsp.h"
#include "ioDefined.h"
#include "globalConstants.h"


char autoGain(long Fs,fractional* p1, fractional* p2,int eOrient)
{
char g=0;  //g is the gain setting (from 0 to 62). initialize to 0
char done=0; //done flag
fractional pwr1=0;
fractional pwr2=0;
long long sum1=0;
long long sum2=0;
int i; //counter
int cmnr=0; //no common mode noise reduction

digitalResSet(g); //initialise gain

while(!done)
{
  getSamples_kj(POWERSAMPLES,Fs,p1,p2,ELECTRODEa,eOrient,cmnr);

  //work out mean of each:
  for (i=0;i<POWERSAMPLES;i++)
  {
    sum1 = sum1 + *(p1+i);
    sum2 = sum2 +  *(p2+i);
  }
    sum1=sum1/POWERSAMPLES;
    sum2=sum2/POWERSAMPLES;

  //subtract the average for a 0 mean set of samples:
  for (i=0;i<Nsamples;i++)
  {
    *(p1+i) -= sum1;
    *(p2+i) -= sum2;
  }

  pwr1 = VectorPower(POWERSAMPLES,p1);
  pwr2 = VectorPower(POWERSAMPLES,p2);

  //if power on all probes < gTarget and gTarget remains within the limit of 63:
  if ((pwr1 < gTarget) && (pwr2 < gTarget) && g<63)
  {
  g++;   //increase gain on digital resistor
  digitalResSet(g);
  }
  else
  {
```

```
  done=1;    //****** SHOULD BE DONE=1. 0 IS JUST FOR DEBUGGING
  }

}//while

digitalResSet(g-1);
return g-1;

}//function
```

delay_us.c

```
//general purpose delay for 40M instruction clock

void delay_us(long d)
{

long count=0;

d--; //decrease by 1us to account for branching overheads

//each time through the for loop takes 20 instruction cycles = 0.5us:
for(count=1;count<=(d*2);count++)
                  //so 2 times through takes 1us
  {
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
  }

//add nop instructions to bring the total overhead time to 1us,
//compensating for the decrease of 1us (before the loop)
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
asm("nop");
}
```

digitalResSet.c

```
//send digital resistor value
//each of the 4 resistors on the chip are sent the same value
//resBit0 and resBit1 identify which resistor is being set
//the 6 bits of bitvalue set the resistance position (from 0 to 63)
```

```
#include "ioDefined.h"
#include "p33FJ128GP306.h"

void delay_us(long int d);
void digitalResSet(int R)  //R must be in the range 0 to 63 for the DS1844-100 digital resistor

{
char res; //counter to choose each resistor in turn
char bitvalue[6]={0,0,0,0,0,0}; //initialize to all 0
char resBit0,resBit1 = 0;// resistor number identifiers

// The way the resistor is set up in the circuit, lower values produce higher gain
// so the next line makes higher values produce a higher gain.
R=63-R;

//digital resistor initialization
RV1RW=0;  //0 = mode to write resistor value , 1 = mode toread resistor value
          //must be stable before any communication
RV1RST=0;  //communication starts with a 0 to 1 transition on this pin

if (R& 0b100000) //check each bit of R to see if it is a 1. If so, then that bitvalue is set to 1
   bitvalue[5]=1;

if (R& 0b10000)
  bitvalue[4]=1;

if (R& 0b1000)
  bitvalue[3]=1;

if (R& 0b100)
  bitvalue[2]=1;

if (R& 0b10)
  bitvalue[1]=1;

if (R& 0b1)
  bitvalue[0]=1;

RV1RST=1;  //communication starts with a 0 to 1 transition on this pin
RV1CLK=0;  //data bit is sent on a 0 to 1 transition of this bit

for (res=0;res<4;res++)
{
  //determine the 2 bits that identify which resistor is being set
  // (00=resistor 0, 01=resistor1, 10=resistor 2, 11=resistor 3)

  resBit1=0;
  resBit0=0;

  if (res& 0b10)
    {resBit1=1;}

  if (res& 0b1)
    {resBit0=1;}

  //identify the resistor to be set with resBit1 and resBit0:
  RV1DIN=resBit1;
  delay_us(1);
  RV1CLK=1;
  delay_us(1);
  RV1CLK=0;
```

```
  RV1DIN=resBit0;
  delay_us(1);
  RV1CLK=1;
  delay_us(1);
  RV1CLK=0;

  //send the resistance position with the 6 bits of bitvalue
  RV1DIN=bitvalue[5];
  delay_us(1);
  RV1CLK=1;
  delay_us(1);
  RV1CLK=0;

  RV1DIN=bitvalue[4];
  delay_us(1);
  RV1CLK=1;
  delay_us(1);
  RV1CLK=0;

  RV1DIN=bitvalue[3];
  delay_us(1);
  RV1CLK=1;
  delay_us(1);
  RV1CLK=0;

  RV1DIN=bitvalue[2];
  delay_us(1);
  RV1CLK=1;
  delay_us(1);
  RV1CLK=0;

  RV1DIN=bitvalue[1];
  delay_us(1);
  RV1CLK=1;
  delay_us(1);
  RV1CLK=0;

  RV1DIN=bitvalue[0];
  delay_us(1);
  RV1CLK=1;
  delay_us(1);
  RV1CLK=0;
  }//for loop
}//function
```

findElectrodeOrientation.c

```
//Determine which way the electrodes are orientated
//Method:  take several sets of samples.
//The orientation with the highest cumulative peak correlation coefficient 'wins'
#include  "p33FJ128GP306.h"
#include "dsp.h"
#include "ioDefined.h"
#include "globalConstants.h"
#define TRIALS 4 //use this many sets of samples
```

```
int findElectrodeOrientation(fractional* s1ptr, fractional* s2ptr, fractional* cVect,
                             fractional* cScalei, long Fs)
{
  //assume forwards orientation.
  // if sensor installed backwards, this will change to orient=1
  int orient=0;
  int i=0; //general purpose counter
  float c1,c2;
  float v1,v2;
  float c1sum=0; //initialise
  float c2sum=0;
  char dummypwr1,dummypwr2;

  for(i=0;i<TRIALS;i++)
  {
    getSamples_kj(Nsamples,Fs,s1ptr,s2ptr,ELECTRODEa,orient);
    //try forwards orientation
    sigProcB(s1ptr,s2ptr,Fs,cVect,spacingA,&c1,cScalei,&v1,dummypwr1,dummypwr2);
    c1sum=c1sum+c1;
    getSamples_kj(Nsamples,Fs,s1ptr,s2ptr,ELECTRODEa,orient);
    //try backwards orientation
    sigProcB(s2ptr,s1ptr,Fs,cVect,spacingA,&c2,cScalei,&v2,dummypwr1,dummypwr2);
    c2sum=c2sum+c2;
  }

  if(c2sum> c1sum) {orient=1;}  //i.e. the electrodes are reversed

return orient;

}//function
```

flashLED.c

```
// flashes LED connected to dsPIC
#include "p33FJ128GP306.h"
#include "ioDefined.h"


void flashLED()
{
while(1)
{
  LED=1;
  delay_us(250000);

  LED=0;
  delay_us(250000);

} //while

} //flashLED
```

floatSend.c

```
//send results to PC
#include  "p33FJ128GP306.h"

//SENDS A FLOAT OVER UART IN 2 BYTES. FIRST BYTE IS THE INTEGER PART (0 TO 255)
//SECOND BYTE CONTAINS THE FIRST 2 DIGITS OF THE FRACTIONAL PART:
void floatSend(float floatNum)

{
  int digits=0; //USED TO BREAK UP THE FLOAT INTO DIGITS

  digits=floatNum; //CAREFUL --DIGITS IS AN INT, FLOATNUM IS A FLOAT.
                   //THE FRACTIONAL PART OF FLOATNUM IS DISCARDED
                   //IN THE TYPE CONVERSION FROM FLOAT TO INT
  sendUARTdata(&digits,1);  //SEND THE INTEGER PART OF FLOATNUM.  FLOATNUM HAS TO BE <255
  digits = ((floatNum-digits+0.005) * 100);  //THIS SENDS THE FIRST 2 DIGITS
                // OF THE FRACTIONAL PART OF FLOATNUM.
                // FOLLOW THE TYPE CONVERSION:
                //v1-(float)digits RETURNS THE FRACTIONAL PART OF FLOAT
                //MULTIPLYING BY 100 'PROMOTES' THE FIRST 2 DIGITS OF THE FRACTIONAL
                // PART INTO THE INTEGER PART
                //FINALLY, THE TYPE CONVERSION INTO DIGITS DISCARDS THE REST
                // OF THE FRACTIONAL PART OF FLOATNUM
  sendUARTdata(&digits,1); //SEND THE FIRST 2 FRACTIONAL PLACES OF FLOATNUM
}//floatSend
```

getSamples_kj.c

```
#include <dsp.h>
#include "globalConstants.h"
#include  "p33FJ128GP306.h"


void getSamples_kj(int N,long int Fs, fractional* p1, fractional* p2,
                   int electrodeSet, int eOrient, int cmnr)
{
unsigned int i; //general purpose counter variable
unsigned int* adcPtr = (unsigned int*)(&_DMA_BASE); //starting address of DMA buffer.
                                                    // _DMA_BASE is defined in the linker
//offsets to the adsPtr to retrieve the correct data from the corresponding electrode
int os1,os2,os3,os4;

PR2= (int)(FCLOCK/Fs)-1;
TMR2=0; //initialize timer 2
T2CONbits.TON=1; //start timer2
AD1CON1bits.ADON = 1; //start adc

//electrodes oriented forwards or backwards:
if(eOrient==0) {os1=0; os2=1; os3=3; os4=2;}
else       {os1=1; os2=0; os3=2; os4=3;}

for(i=1;i<=N;i++)
  {
      IFS0=0;
      //  AD1CON1bits.SAMP=1; //starts the sampling process.
      // Tsamp (charge-up time of the sampling capacitor) set to 12 Tad (see above)
      //automatic conversion start causes SAMP bit to clear automatically
```

```
      while(!IFS0bits.DMA0IF)  //wait until DMA channel 0 data transfer is comlete
        {
        asm("nop");
        asm("nop");
      //  asm("nop");//watch IFS0 register. Bit 13 is the the adc1 conversion done interrupt flag
      //  asm("nop");//Bit 4 (DMA0IF) is the dma0 block transfer complete done interrupt flag
      //  asm("nop");
        }

        switch(electrodeSet)
        {
          case 1:
          {
          *p1 = (*(adcPtr+os1)); //samples are 10 bits long
          *p2 = (*(adcPtr+os2));
          if(cmnr==1) {*p1 = (*p1 - *p2)/2;}  //subtract signals to remove common mode noise
          p1++;
          p2++;
          break;
          }

          case 2:
          {
          *p1 = (*(adcPtr+os3)); //samples are 10 bits long
          p1++;
          *p2 = (*(adcPtr+os4));
          p2++;
          break;
          }
        } //switch
  } //for i

T2CONbits.TON=0; //stop timer
TMR2=0;
AD1CON1bits.ADON = 0; //stop adc
}
```

IOsetup.c

```
//sets up i/o ports
#include "p33FJ128GP306.h"

void IOsetup()
{
  AD1PCFGH = 0xFFFF;//all pins with possible input functions set to digital i/o
  AD1PCFGL = 0b1111111110111000;  //except the ones used for sensor inputs (AN0,AN1,AN2, and AN6)
          //this is necessary, otherwise the tris and port reads for the digital pins won't work

  TRISB=0xFFFF; //sets all pins in portB to inputs.  Will need some outputs later
  TRISD=0xFFFF;
  TRISF=0xFFFF;
  TRISG=0xFFFF;

  //TRIS SETTINGS FOR PINS TO SET DIGITAL RESISTOR:
  TRISDbits.TRISD8=0; //RV1RST
  TRISDbits.TRISD2=0; //RV1RW
```

```
  TRISDbits.TRISD5=0; //RV1CLK
  TRISFbits.TRISF0=0; //RV1DIN
  TRISGbits.TRISG0=0; //RV1DOUT

  //OTHER TRIS SETTINGS:
  TRISGbits.TRISG13=0; //LED
  TRISFbits.TRISF5=1;  //RXD
  TRISFbits.TRISF4=0;  //TXD
  TRISBbits.TRISB12=1;  //CTS
  TRISBbits.TRISB10=0;  //RTS
}
```

sendCorr.c

```
//send correlation results to PC
#include  "p33FJ128GP306.h"
#include "globalConstants.h"
#include "dsp.h"

void sendCorr(fractional* corrPeak1, int* cScalePtr1)
//!! note: the pointer to corrData will actually be a fractcomplex*.
//However it will be used to send one byte at a time
{
  int i=0; //general purpose counter
    sendUARTdata(cScalePtr1,2); //send scaling value as 2 bytes
    sendUARTdata(corrPeak1,CORR_PEAK_LENGTH*2);  // *2 because fractional is 2 bytes
}//sendCorr
```

sendRaw.c

```
//send raw data to PC

#include  "p33FJ128GP306.h"
#include "globalConstants.h"
#include "dsp.h"

void sendRaw(fractional* s1,fractional* s2)
{
  int i=0; //general purpose counter
  unsigned char message; //FOR HANDSHAKING ETC.

  while(!U2STAbits.URXDA) {}; //JUST WAIT FOR A MESSAGE TO TRIGGER NEXT BLOCK OF DATA
  message=U2RXREG;
  sendUARTdata(s1,Nsamples*2); //*2 because fractional is 2 bytes long
  while(!U2STAbits.URXDA) {}; //JUST WAIT FOR A MESSAGE TO TRIGGER NEXT BLOCK OF DATA
  message=U2RXREG;
  sendUARTdata(s2,Nsamples*2);
}//sendRaw
```

sendResultsC.c

```
//send results to PC

#include  "p33FJ128GP306.h"
#include "globalConstants.h"
#include "dsp.h"

void sendResultsC(float v1,float c1,
                  char s1Pwr,char s2Pwr,
                  char g,char validReadings1)
//!! note: the pointer to corrData will actually be a fractcomplex*.
// However it will be used to send one byte at a time
{
  int i=0; //general purpose counter
  unsigned char message; //FOR HANDSHAKING ETC.

  while(!U2STAbits.URXDA) {}; //JUST WAIT FOR A MESSAGE TO TRIGGER.
  message=U2RXREG;
  //SEND RESULTS ONLY (v1, c1, s1Pwr, s2Pwr)
  //SENDS v1 AS 2 BYTES: INTEGER PART (0 TO 255) AND FIRST 2 DIGITS OF FRACTIONAL PART
  floatSend(v1);
  floatSend(c1);
  sendUARTdata(&s1Pwr,1);
  sendUARTdata(&s2Pwr,1);
  sendUARTdata(&g,1);
  sendUARTdata(&validReadings1,1);
}//sendResults
```

sendUARTdata.c

```
//sends data over the UART interface

#include  "p33FJ128GP306.h"

//data is a pointer to the data to send, length is the number of bytes to send:
void sendUARTdata(char* data,int length)
{
int i=0;

for (i=0;i<length;i++)
{
U2TXREG = *(data+i); //load data and transmit (should be *(data+i) for final version)
while (U2STAbits.TRMT==0) //
  {} //do nothing, just wait for transmission to finish
}//for
}//sendUARTdata
```

signalProcB.c

```
//Signal cross correlation:
#include  "p33FJ128GP306.h"
#include "ioDefined.h"
```

```
#include "dsp.h"
#include "globalConstants.h"
#include "float.h"
#include "math.h"


float sigProcB (fractional* s1ptr,fractional* s2ptr,long Fs, fractional* corrPtr,float spacing,
                float* c,fractional* cScalePtr,float* velocity,char* pwr1, char* pwr2)
{

  int i=0; //general purpose counter
  int cMaxIndex = 0;
  fractional cMax = 0; //maximum correlation coefficient
  fractional cScaleFract[3]; //element1= s1 squared, el.2 = s2 squared, el3= product of squares
  int vint; //just for debug
  unsigned long long vSum1=0;
  unsigned long long vSum2=0; //USED TO CALCULATE THE AVERAGE OF THE INPUT VECTORS
  int av1=0;
  int av2=0; //TO STORE THE AVERAGES OF THE RAW DATA IN ORDER TO CREATE A ZERO MEAN SEQUENCE
  int cint=0; //just for debug


  //calculate the mean of the input:
  for (i=0;i<Nsamples;i++)
  {
    vSum1=vSum1+ *(s1ptr+i);
    vSum2=vSum2+ *(s2ptr+i);
  }//for
  av1=vSum1/Nsamples;
  av2=vSum2/Nsamples;

  //subtract the mean from the sequences to get zero mean data
  for (i=0;i<Nsamples;i++)
  {
    s1ptr[i] = (s1ptr[i])-av1;
    s2ptr[i] = (s2ptr[i])-av2;
  }//for

  //CALCULATE CORRELATION SCALING FACTOR:
  cScaleFract[0]=VectorDotProduct(Nsamples,s1ptr,s1ptr);
  cScaleFract[1]=VectorDotProduct(Nsamples,s2ptr,s2ptr);
    VectorMultiply(1,&cScaleFract[2],&cScaleFract[0],&cScaleFract[1]);
  *cScalePtr = Float2Fract(sqrt(Fract2Float(cScaleFract[2])));
  *pwr1=cScaleFract[0]>>6; // >>6 to change from a 15 significant bit fractional to a char.
  *pwr2=cScaleFract[0]>>6; // use the entire range of the power bar (0 to 255).
          \\ depends on maximum power of the signal. >>6 for max power of (int)2042

  if(*cScalePtr==0) {*cScalePtr=32767;} //something clearly not right if *cScalePtr==0

  //CALCULATE CORRELATION FOR CORR_PEAK_LENGTH LAGS (0 to CORR_PEAK_LENGTH-1 lags)
  for(i=0;i<CORR_PEAK_LENGTH;i++) {*(corrPtr+i)=VectorDotProduct(Nsamples-i,s1ptr,s2ptr+i);}

  cMax=VectorMax(100,corrPtr+2,&cMaxIndex);
  //+2 to ignore any high correlation at 0 lag due to common mode noise

  //division doesn't work properly with fractionals. they just seem to get treated as integers:
  *c = Fract2Float(cMax)/Fract2Float(*cScalePtr);
  cint=*c * 100; //for watch window in debug

  *velocity = ((float)spacing * (float)Fs) / (float)(cMaxIndex);
//  if(*velocity < 4) { *velocity=0;} //limit unreasonable results
  if(*velocity > 40) {*velocity==40;} //limit unreasonable results
```

```
} //function

//notes:
/*
remember:  you can't just multiply fractional format numbers.
They seem to be treated as plain integers,and when multiplying causes the registers
to overflow, you don't just lose precision --you get incorrect results.That's why
the trouble was taken to put the real and imaginary parts of the ffts into their own
respective arrays.That way the VectorMultiply function could be used,
and gives correct results.

When using cMaxIndex, keep in mind the the vector pointed at is a fractcomplex.
This means that there is a 16 fractionalholding the real part, and another to hold
the imaginary part.  Therefore, when the VectorMax function is looking at a single
fractcomplex entry, the (fractional) index moves by 2 places,not just 1.
Therefore we must divide by 2 to find the index of the fractcomplex location.


NOTE:
    Matlab-- ifft(conj(fft1)) .* fft2 equals
    mplab--  ifft(fft1 .* conj(fft2))
    Don't know why. It's not to do with the twiddle factors--they just affect the sign
    of the imaginary parts


MATLAB:  xcorr(b,a)  --b slids back in time under a
            corresponds to ifft(fft(b).*conj(fft(a)))
            with a and b padded with N-1 zeros
    therefore:  a is upstream, b is downstream
*/
```

UART_setup.c

```
//sets up UART interface

#define CHECKREADY 5
#define READY 7

#include  "p33FJ128GP306.h"
#include "ioDefined.h"

void UART_setup()  //UART 2 is used  -that's the way the pins were wired
{

char count=0;
char readyMessage=0;

U2MODEbits.UARTEN = 1; //enable UART2
U2MODEbits.RTSMD = 0; //U2RTS pin in flow control mode
U2MODEbits.UEN = 0b10; //0b10 =>  U2TX, U2RX U2CTS AND U2RTS are enabled and used
U2MODEbits.BRGH = 0; // High Baud Rate Enable bit disabled
//IEC1bits.U2TXIE = 1;  //enable UART transmit interrupts if desired


U2BRG = 10; //set baud rate --see datasheet for formula
```

```
//interrupt set when the last character is shifted out of the transmit shift register:
//U2STAbits.UTXISEL1 = 0;
//U2STAbits.UTXISEL0 = 1; //

U2STAbits.UTXEN = 1; //transmit enabled
          //when enabled, data is sent immediately after being loaded into the U2TXREG
          //this register can only buffer 4 bytes,
          // so wait until a byte is sent before sending the next one

//FLUSH OUT ANY DATA SITTING IN THE INPUT BUFFER:
while(U2STAbits.URXDA) //WHILE RECIEVE BUFFER NOT EMPTY
  {count=U2RXREG;} //LOAD ANY BUFFER DATA TO count TO GET RID OF IT

U2STAbits.OERR=0;

/*

    //READY HANDSHAKE WITH WINDOWS
    while(readyMessage != CHECKREADY)
    {
      while(!U2STAbits.URXDA) //WAIT FOR INPUT BUFFER BYTE READY FLAG
        {
        LED=1;
        delay_us(75000);
        LED=0;
        delay_us(75000);
        }
      readyMessage=U2RXREG;
    }

    readyMessage=READY;  //HANDSHAKE TO SEND BACK TO WINDOWS
    sendUARTdata(&readyMessage,1);
*/

}//UART_setup
```

UARTrecieveTest.c

```
//Tests the UART recieve. 'N' bytes are read by the serial port, and saved to the s1 array
//Note: the s1 array is normally used to store the input samples

#include  "p33FJ128GP306.h"

void UARTrecieveTest(char* p1, int N)
{
int i=0;

while(1)
{
  for (i=0;i<N;i++)
    {
    while(!U2STAbits.URXDA)
    {;} //WHILE RECIEVE BUFFER EMPTY DO NOTHING
      *(p1+i) = U2RXREG;

    }
  asm("NOP");
```

```
}//While, infiite loop

}//UARTrecieveTest
```

UARTtest.c

```
//test UART interface
#include  "p33FJ128GP306.h"
#include "ioDefined.h"

void UARTtest()  //sends first 10 alphabet letters in ascii
{
char count[10]={65,66,67,68,69,70,71,72,73,74};
char selection=0;

while(1)
{
  sendUARTdata(&count,10);

  /*
  if(U2STAbits.URXDA) //if data character is available in the receive buffer
    {
    selection=U2RXREG; //read the receive buffer
    }
  */

  LED=1;
  delay_us(250000);
  LED=0;
  delay_us(250000);
}

}//UARTtest
```

globalConstants.h

```
/* Constant Definitions */
#define CORR_PEAK_LENGTH 200
#define MINCORR 0.30
#define Nsamples 2048
#define FCLOCK 40000000 //timer speed in MHz
#define AVERAGING 4
#define spacingA 0.02 //probe separation distance in m
#define spacingB 0.02 //probe separation distance in m
#define ELECTRODEa 1
#define ELECTRODEb 2 //only used on sensor with two sets of electrodes
```

ioDefined.h

```
//defines registers to correspond to the
//pins on the PIC
#define RV1RST PORTDbits.RD8
#define RV1RW PORTDbits.RD2
#define RV1CLK PORTDbits.RD5
#define RV1DIN PORTFbits.RF0
#define RV1DOUT PORTGbits.RG0
#define LED PORTGbits.RG13
#define RXD PORTFbits.RF5
#define TXD PORTFbits.RF4
#define CTS PORTBbits.RB12
#define RTS PORTBbits.RB10
```

Appendix D

Visual® C++code

The following code was written for the Windows® Visual C++ application, which serves as the user interface and data logging software for the embedded electrostatic sensor. Some of the code has been automatically generated to encode the structure designed using the graphical user interface (mostly commands beginning with '*this...*' in the first half of the program listing).

```
#pragma once
#define N 2048
#define RESULTS 254
#define NORAW 251
#define AUTOGAIN 252
#define CHECKREADY 5
#define READY 7
#define SENSOREXIT 249
#define CORR_DATA_LENGTH 200 //correlation length 100
#define SET_FS 248
#define RAW 247
#define CORR 246
#define NO_OTHER_DATA 245
#define CMNR 244
#define NO_CMNR 243



namespace kj2 {

  using namespace System;
  using namespace System::ComponentModel;
  using namespace System::Collections;
  using namespace System::Windows::Forms;
  using namespace System::Data;
  using namespace System::Drawing;
  using namespace Teroid::DataGraph;
  using namespace System::IO;
```

```
  /// <summary>
  /// Summary for Form1
  ///
  /// WARNING: If you change the name of this class, you will need to change the
  ///          'Resource File Name' property for the managed resource compiler tool
  ///          associated with all .resx files this class depends on.  Otherwise,
  ///          the designers will not be able to interact properly with localized
  ///          resources associated with this form.
  /// </summary>
  public ref class Form1 : public System::Windows::Forms::Form
  {
  public:
    Form1(void)
    {
      InitializeComponent();
      //CONSTRUCTOR CODE KJ:
      //TODO: Add the constructor code here
      dataRequest=gcnew array<unsigned char>(1); //serial port write function needs an array
      backgroundWorker3->RunWorkerAsync(); //SPLASH SCREEN
      kjTable=gcnew DataTable;//KJ STORES RECENT VELOCITY AND CORRELATION VALUES FOR GRAPHS
      go=false;       //TO GET IN AND OUT OF THE MAIN LOOP (in backgroundWorker1)
      vHistory1 = gcnew array<float^>(100); //KEEP RECENT VELOCITY VALUES FROM SENSOR 1 FOR GRAPHS
      vHistory2 = gcnew array<float^>(100); //KEEP RECENT VELOCITY VALUES FROM SENSOR 2 FOR GRAPHS
//      R  = gcnew Random;     //RANDOM NUMBER GENERATOR FOR TEST PURPOSES
      stime = new SYSTEMTIME();//GETS THE CURRENT SYSTEM TIME FOR DATE STAMPS ETC.
      writeRes=false;
      path = "c:\\eStaticData\\";//MAIN SAVE FOLDER FOR THE DATA
      pathadd=textBox1->Text; //USER DECIDES THE ACTUAL FILENAME

//port initialisation code moved to the 'online' radio button (radioButton6)


      kjData=gcnew array<unsigned char>(N*2); // *2 because fractional type is 2 bytes long

      validReadings1=0;
      label19->Text="-";


      radioButton2->Checked=true;      //SAVE TO FILE ENABLED
      radioButton4->Checked=true;      //RAW DATA SAVE EVERY MINUTE
      graphShow=true;
      checkBox2->Checked=false;      //GRAPH UPDATE ON
      watchdogTime=60; //i.e. never for debug
      backgroundWorker2->RunWorkerAsync(); //watchdog
      label6->Text="-";
      label7->Text="-";



      //xdata=1;//KJ
      //ydata=1;//KJ
      //zdata=1;//KJ
      //array<Char>^kjChars = {'0','1','2'};
      // portNames[1]=jk;
      // textBox1->Text=portNames[1];

    }

  protected:
```

```
    /// Clean up any resources being used.
    /// </summary>
    ~Form1()
    {
      dataRequest[0]=SENSOREXIT;
      serialPort1->Write(dataRequest,0,1);
      serialPort1->Close();
      if (components)
      {
        delete components;
      }
    }

  //FORM VARIABLES (GLOBAL TO FORM)
  private:
      DataTable^ kjTable;        //FOR DATAGRAPH
      DataRow^ kjDataRow;        //FOR TABLE
//      System::Int32 xdata,ydata,zdata;//FOR TESTING
      bool go ;            //TO GET IN AND OUT OF MAIN LOOP
      bool writeRes;           // true=write results to file
      array<Object^>^ vHistory1;    //KEEP TRACK OF VELOCITY HISTORY
      array<Object^>^ vHistory2;
//      Random^ R;              //random number generator
      StreamWriter^ sw;        //FOR WRITING TO FILE
      String^ path;          //TO STORE THE SAVE PATH DIRECTORY
      String^ pathadd;         //TO STORE THE SAVE PATH NAME
      SYSTEMTIME* stime;         //RETRIEVES SYSTEM TIME
      array<unsigned char>^ kjData; //TO STORE DATA READ FROM SERIAL PORT
      WORD hour,minute,sec,day,month,year,nextSec;
      String^ secFill;
      String^ minFill;
      array<unsigned char>^  dataRequest;
      array<String^>^ portNames;
      int watchdogTime;
      bool graphShow;
      float v1,v2,c1,c2;
      int pwr1,pwr2;
      char g; //digital resistor gain setting
      int i; //general purpose counter
      char validReadings1;
      char validReadings2;

  private: System::Windows::Forms::Button^  button1;
  private: System::Windows::Forms::Label^  label1;



  private: System::Windows::Forms::RadioButton^  radioButton1;
  private: System::Windows::Forms::RadioButton^  radioButton2;
  private: System::Windows::Forms::RadioButton^  radioButton3;

  private: System::Windows::Forms::RadioButton^  radioButton5;

  private: System::Windows::Forms::Panel^  panel1;
  private: System::Windows::Forms::Panel^  panel2;

  private: Teroid::DataGraph::DataGraph^  dataGraph1;

  private: System::IO::Ports::SerialPort^  serialPort1;
  private: System::Windows::Forms::Label^  label2;
  private: System::Windows::Forms::ProgressBar^  progressBar1;
```

```
    private: System::Windows::Forms::TextBox^  textBox1;
    private: System::Windows::Forms::Label^  label3;
    private: System::Windows::Forms::Label^  label4;
    private: System::Windows::Forms::RadioButton^  radioButton4;

    private: System::Windows::Forms::TrackBar^  trackBar1;
    private: System::Windows::Forms::Label^  label5;
private: System::Windows::Forms::Button^  button3;
private: System::Windows::Forms::CheckBox^  checkBox1;
private: System::ComponentModel::BackgroundWorker^  backgroundWorker2;
private: System::ComponentModel::BackgroundWorker^  backgroundWorker3;



private: System::Windows::Forms::CheckBox^  checkBox2;
private: System::Windows::Forms::Label^  label6;
private: System::Windows::Forms::Label^  label7;




private: System::Windows::Forms::Button^  button2;
private: System::Windows::Forms::CheckBox^  checkBox3;
private: System::Windows::Forms::Label^  label10;

private: System::Windows::Forms::Label^  label12;



private: System::Windows::Forms::Label^  label16;
private: System::Windows::Forms::TextBox^  textBox2;
private: System::Windows::Forms::Label^  label17;
private: System::Windows::Forms::Label^  label18;
private: System::Windows::Forms::Timer^  timer1;
private: System::Windows::Forms::Label^  label19;

private: System::Windows::Forms::Label^  label21;
private: System::Windows::Forms::CheckBox^  checkBox4;
private: System::Windows::Forms::ContextMenuStrip^  contextMenuStrip1;



private: System::Windows::Forms::Label^  label9;
private: System::Windows::Forms::TextBox^  textBox3;
private: System::Windows::Forms::Button^  button4;



    private: System::ComponentModel::IContainer^  components;


    protected:

    private:
      /// <summary>
      /// Required designer variable.


#pragma region Windows Form Designer generated code
```

```
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
void InitializeComponent(void)
{
  this->components = (gcnew System::ComponentModel::Container());
  System::ComponentModel::ComponentResourceManager^  resources = (gcnew System::
            ComponentModel::ComponentResourceManager(Form1::typeid));
  this->button1 = (gcnew System::Windows::Forms::Button());
  this->label1 = (gcnew System::Windows::Forms::Label());
  this->radioButton1 = (gcnew System::Windows::Forms::RadioButton());
  this->radioButton2 = (gcnew System::Windows::Forms::RadioButton());
  this->radioButton3 = (gcnew System::Windows::Forms::RadioButton());
  this->radioButton5 = (gcnew System::Windows::Forms::RadioButton());
  this->panel1 = (gcnew System::Windows::Forms::Panel());
  this->panel2 = (gcnew System::Windows::Forms::Panel());
  this->radioButton4 = (gcnew System::Windows::Forms::RadioButton());
  this->label2 = (gcnew System::Windows::Forms::Label());
  this->dataGraph1 = (gcnew Teroid::DataGraph::DataGraph());
  this->serialPort1 = (gcnew System::IO::Ports::SerialPort(this->components));
  this->progressBar1 = (gcnew System::Windows::Forms::ProgressBar());
  this->textBox1 = (gcnew System::Windows::Forms::TextBox());
  this->label3 = (gcnew System::Windows::Forms::Label());
  this->label4 = (gcnew System::Windows::Forms::Label());
  this->trackBar1 = (gcnew System::Windows::Forms::TrackBar());
  this->label5 = (gcnew System::Windows::Forms::Label());
  this->button3 = (gcnew System::Windows::Forms::Button());
  this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());
  this->backgroundWorker2 = (gcnew System::ComponentModel::BackgroundWorker());
  this->backgroundWorker3 = (gcnew System::ComponentModel::BackgroundWorker());
  this->checkBox2 = (gcnew System::Windows::Forms::CheckBox());
  this->label6 = (gcnew System::Windows::Forms::Label());
  this->label7 = (gcnew System::Windows::Forms::Label());
  this->button2 = (gcnew System::Windows::Forms::Button());
  this->checkBox3 = (gcnew System::Windows::Forms::CheckBox());
  this->label10 = (gcnew System::Windows::Forms::Label());
  this->label12 = (gcnew System::Windows::Forms::Label());
  this->label16 = (gcnew System::Windows::Forms::Label());
  this->textBox2 = (gcnew System::Windows::Forms::TextBox());
  this->label17 = (gcnew System::Windows::Forms::Label());
  this->label18 = (gcnew System::Windows::Forms::Label());
  this->timer1 = (gcnew System::Windows::Forms::Timer(this->components));
  this->label19 = (gcnew System::Windows::Forms::Label());
  this->label21 = (gcnew System::Windows::Forms::Label());
  this->checkBox4 = (gcnew System::Windows::Forms::CheckBox());
  this->contextMenuStrip1 = (gcnew System::Windows::Forms::
              ContextMenuStrip(this->components));
  this->label9 = (gcnew System::Windows::Forms::Label());
  this->textBox3 = (gcnew System::Windows::Forms::TextBox());
  this->button4 = (gcnew System::Windows::Forms::Button());
  this->panel1->SuspendLayout();
  this->panel2->SuspendLayout();
  (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(
                  this->trackBar1))->BeginInit();
  this->SuspendLayout();
  //
  // button1
  //
  this->button1->BackColor = System::Drawing::SystemColors::Control;
  this->button1->ForeColor = System::Drawing::Color::Green;
  this->button1->Location = System::Drawing::Point(579, 418);
```

```
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(153, 39);
this->button1->TabIndex = 1;
this->button1->Text = L"GO";
this->button1->UseVisualStyleBackColor = false;
this->button1->Click += gcnew System::EventHandler(this, &Form1::button1_Click);
//
// label1
//
this->label1->AutoSize = true;
this->label1->Location = System::Drawing::Point(515, 296);
this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(35, 13);
this->label1->TabIndex = 5;
this->label1->Text = L"label1";
//
// radioButton1
//
this->radioButton1->AutoSize = true;
this->radioButton1->Location = System::Drawing::Point(12, 12);
this->radioButton1->Name = L"radioButton1";
this->radioButton1->Size = System::Drawing::Size(105, 17);
this->radioButton1->TabIndex = 5;
this->radioButton1->TabStop = true;
this->radioButton1->Text = L"Save results only";
this->radioButton1->UseVisualStyleBackColor = true;
this->radioButton1->CheckedChanged += gcnew System::EventHandler(
                    this, &Form1::radioButton1_CheckedChanged);
//
// radioButton2
//
this->radioButton2->AutoSize = true;
this->radioButton2->Location = System::Drawing::Point(12, 35);
this->radioButton2->Name = L"radioButton2";
this->radioButton2->Size = System::Drawing::Size(136, 17);
this->radioButton2->TabIndex = 6;
this->radioButton2->TabStop = true;
this->radioButton2->Text = L"Save results + raw data";
this->radioButton2->UseVisualStyleBackColor = true;
this->radioButton2->CheckedChanged += gcnew System::EventHandler(
                    this, &Form1::radioButton2_CheckedChanged);
//
// radioButton3
//
this->radioButton3->AutoSize = true;
this->radioButton3->Location = System::Drawing::Point(12, 55);
this->radioButton3->Name = L"radioButton3";
this->radioButton3->Size = System::Drawing::Size(65, 17);
this->radioButton3->TabIndex = 7;
this->radioButton3->TabStop = true;
this->radioButton3->Text = L"No save";
this->radioButton3->UseVisualStyleBackColor = true;
this->radioButton3->CheckedChanged += gcnew System::EventHandler(
                    this, &Form1::radioButton3_CheckedChanged);
//
// radioButton5
//
this->radioButton5->AutoSize = true;
this->radioButton5->Location = System::Drawing::Point(14, 55);
this->radioButton5->Name = L"radioButton5";
this->radioButton5->Size = System::Drawing::Size(37, 17);
this->radioButton5->TabIndex = 9;
```

```
this->radioButton5->TabStop = true;
this->radioButton5->Text = L"1h";
this->radioButton5->UseVisualStyleBackColor = true;
//
// panel1
//
this->panel1->Controls->Add(this->radioButton3);
this->panel1->Controls->Add(this->radioButton2);
this->panel1->Controls->Add(this->radioButton1);
this->panel1->Location = System::Drawing::Point(578, 128);
this->panel1->Name = L"panel1";
this->panel1->Size = System::Drawing::Size(147, 75);
this->panel1->TabIndex = 11;
//
// panel2
//
this->panel2->Controls->Add(this->radioButton5);
this->panel2->Controls->Add(this->radioButton4);
this->panel2->Controls->Add(this->label2);
this->panel2->Location = System::Drawing::Point(578, 225);
this->panel2->Name = L"panel2";
this->panel2->Size = System::Drawing::Size(117, 75);
this->panel2->TabIndex = 12;
//
// radioButton4
//
this->radioButton4->AutoSize = true;
this->radioButton4->Location = System::Drawing::Point(14, 35);
this->radioButton4->Name = L"radioButton4";
this->radioButton4->Size = System::Drawing::Size(47, 17);
this->radioButton4->TabIndex = 8;
this->radioButton4->TabStop = true;
this->radioButton4->Text = L"1min";
this->radioButton4->UseVisualStyleBackColor = true;
//
// label2
//
this->label2->AutoSize = true;
this->label2->Location = System::Drawing::Point(3, 12);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(116, 13);
this->label2->TabIndex = 11;
this->label2->Text = L"Raw data save interval";
//
// dataGraph1
//
this->dataGraph1->BackColor = System::Drawing::SystemColors::Control;
this->dataGraph1->BackgroundImageLayout = System::Windows::Forms::ImageLayout::None;
this->dataGraph1->BackShadingColor = System::Drawing::SystemColors::Control;
this->dataGraph1->BorderStyle = System::Windows::Forms::BorderStyle::None;
this->dataGraph1->CausesValidation = false;
this->dataGraph1->ConcatenateColumnOrBarText = false;
this->dataGraph1->Connection = nullptr;
this->dataGraph1->ConnectionString = L"";
this->dataGraph1->DataColumn = L"Sales";
this->dataGraph1->DataGroupingColumn = L"Month";
this->dataGraph1->DataMember = L"";
this->dataGraph1->DataNameColumn = L"Region";
this->dataGraph1->DataSet = nullptr;
this->dataGraph1->DataValueIndexInterval = 5;
this->dataGraph1->DataView = nullptr;
this->dataGraph1->FooterText = L"";
```

```
this->dataGraph1->GraphColors->Add(System::Drawing::Color::YellowGreen);
this->dataGraph1->GraphColors->Add(System::Drawing::Color::SteelBlue);
this->dataGraph1->GraphColors->Add(System::Drawing::Color::Goldenrod);
this->dataGraph1->GraphColors->Add(System::Drawing::Color::Purple);
this->dataGraph1->GraphType = Teroid::DataGraph::GraphTypes::Line;
this->dataGraph1->HeaderText = L"";
this->dataGraph1->LineFormatAbscissae->Color = System::Drawing::Color::Gray;
this->dataGraph1->LineFormatAbscissae->Thickness = 1;
this->dataGraph1->LineFormatAbscissae->Visible = false;
this->dataGraph1->LineFormatBottomBorder->Color = System::Drawing::Color::Black;
this->dataGraph1->LineFormatBottomBorder->Thickness = 1;
this->dataGraph1->LineFormatBottomBorder->Visible = true;
this->dataGraph1->LineFormatColumnOrBarBorders->Color = System::Drawing::Color::Black;
this->dataGraph1->LineFormatColumnOrBarBorders->Thickness = 1;
this->dataGraph1->LineFormatColumnOrBarBorders->Visible = true;
this->dataGraph1->LineFormatDataGroup->Color = System::Drawing::Color::Transparent;
this->dataGraph1->LineFormatDataGroup->Thickness = 1;
this->dataGraph1->LineFormatDataGroup->Visible = false;
this->dataGraph1->LineFormatDataGroupIndexes->Color = System::Drawing::Color::Black;
this->dataGraph1->LineFormatDataGroupIndexes->Thickness = 1;
this->dataGraph1->LineFormatDataGroupIndexes->Visible = false;
this->dataGraph1->LineFormatDataGroupIndexesOpposite->Color = System::Drawing::Color::Black;
this->dataGraph1->LineFormatDataGroupIndexesOpposite->Thickness = 1;
this->dataGraph1->LineFormatDataGroupIndexesOpposite->Visible = false;
this->dataGraph1->LineFormatDataValue->Color = System::Drawing::Color::LightGray;
this->dataGraph1->LineFormatDataValue->Thickness = 1;
this->dataGraph1->LineFormatDataValue->Visible = true;
this->dataGraph1->LineFormatDataValueIndexes->Color = System::Drawing::Color::Black;
this->dataGraph1->LineFormatDataValueIndexes->Thickness = 1;
this->dataGraph1->LineFormatDataValueIndexes->Visible = true;
this->dataGraph1->LineFormatDataValueIndexesOpposite->Color = System::Drawing::Color::Black;
this->dataGraph1->LineFormatDataValueIndexesOpposite->Thickness = 1;
this->dataGraph1->LineFormatDataValueIndexesOpposite->Visible = false;
this->dataGraph1->LineFormatLeftBorder->Color = System::Drawing::Color::Black;
this->dataGraph1->LineFormatLeftBorder->Thickness = 1;
this->dataGraph1->LineFormatLeftBorder->Visible = true;
this->dataGraph1->LineFormatOrdinates->Color = System::Drawing::Color::Gray;
this->dataGraph1->LineFormatOrdinates->Thickness = 1;
this->dataGraph1->LineFormatOrdinates->Visible = false;
this->dataGraph1->LineFormatRightBorder->Color = System::Drawing::Color::Black;
this->dataGraph1->LineFormatRightBorder->Thickness = 1;
this->dataGraph1->LineFormatRightBorder->Visible = true;
this->dataGraph1->LineFormatTopBorder->Color = System::Drawing::Color::Black;
this->dataGraph1->LineFormatTopBorder->Thickness = 1;
this->dataGraph1->LineFormatTopBorder->Visible = true;
this->dataGraph1->LineFormatZero->Color = System::Drawing::Color::Black;
this->dataGraph1->LineFormatZero->Thickness = 1;
this->dataGraph1->LineFormatZero->Visible = false;
this->dataGraph1->LineGraphXAxisType = Teroid::DataGraph::LineGraphXAxisTypes::Points;
this->dataGraph1->Location = System::Drawing::Point(2, -1);
this->dataGraph1->MaxValue = 40;
this->dataGraph1->Name = L"dataGraph1";
this->dataGraph1->Size = System::Drawing::Size(569, 320);
this->dataGraph1->Spacings->Bottom = 50;
this->dataGraph1->Spacings->ColumnsAndBars = 5;
this->dataGraph1->Spacings->Left = 50;
this->dataGraph1->Spacings->Right = 20;
this->dataGraph1->Spacings->Top = 30;
this->dataGraph1->TabIndex = 0;
this->dataGraph1->TextFormatColumnOrBarText->Alignment = Teroid::DataGraph::Alignments::Center;
this->dataGraph1->TextFormatColumnOrBarText->Color = System::Drawing::Color::Black;
this->dataGraph1->TextFormatColumnOrBarText->Font = (
```

```
                gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.25F));
this->dataGraph1->TextFormatColumnOrBarText->Visible = true;
this->dataGraph1->TextFormatDataGroupAxis->Alignment = Teroid::DataGraph::Alignments::Center;
this->dataGraph1->TextFormatDataGroupAxis->Color = System::Drawing::Color::Black;
this->dataGraph1->TextFormatDataGroupAxis->Font = (
                gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.25F));
this->dataGraph1->TextFormatDataGroupAxis->Visible = true;
this->dataGraph1->TextFormatDataGroupIndex->Alignment = Teroid::DataGraph::Alignments::Center;
this->dataGraph1->TextFormatDataGroupIndex->Color = System::Drawing::Color::Black;
this->dataGraph1->TextFormatDataGroupIndex->Font = (
                gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.25F));
this->dataGraph1->TextFormatDataGroupIndex->Visible = false;
this->dataGraph1->TextFormatDataValueAxis->Alignment = Teroid::DataGraph::Alignments::Center;
this->dataGraph1->TextFormatDataValueAxis->Color = System::Drawing::Color::Black;
this->dataGraph1->TextFormatDataValueAxis->Font = (
                gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.25F));
this->dataGraph1->TextFormatDataValueAxis->Visible = true;
this->dataGraph1->TextFormatDataValuesIndex->Alignment = Teroid::DataGraph::Alignments::Right;
this->dataGraph1->TextFormatDataValuesIndex->Color = System::Drawing::Color::Black;
this->dataGraph1->TextFormatDataValuesIndex->Font = (
                gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.25F));
this->dataGraph1->TextFormatDataValuesIndex->Visible = true;
this->dataGraph1->TextFormatFooter->Alignment = Teroid::DataGraph::Alignments::Center;
this->dataGraph1->TextFormatFooter->Color = System::Drawing::Color::Black;
this->dataGraph1->TextFormatFooter->Font = (
                gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.25F));
this->dataGraph1->TextFormatFooter->Visible = true;
this->dataGraph1->TextFormatHeader->Alignment = Teroid::DataGraph::Alignments::Center;
this->dataGraph1->TextFormatHeader->Color = System::Drawing::Color::Black;
this->dataGraph1->TextFormatHeader->Font = (
                gcnew System::Drawing::Font(L"Microsoft Sans Serif", 8.25F));
this->dataGraph1->TextFormatHeader->Visible = true;
this->dataGraph1->Load += gcnew System::EventHandler(this, &Form1::dataGraph1_Load);
//
// serialPort1
//
this->serialPort1->BaudRate = 230400;
this->serialPort1->Handshake = System::IO::Ports::Handshake::RequestToSend;
this->serialPort1->ReadBufferSize = 8192;
this->serialPort1->RtsEnable = true;
//
// progressBar1
//
this->progressBar1->Location = System::Drawing::Point(15, 397);
this->progressBar1->Maximum = 8;
this->progressBar1->Name = L"progressBar1";
this->progressBar1->Size = System::Drawing::Size(328, 13);
this->progressBar1->Step = 0;
this->progressBar1->TabIndex = 13;
//
// textBox1
//
this->textBox1->Location = System::Drawing::Point(570, 340);
this->textBox1->Name = L"textBox1";
this->textBox1->Size = System::Drawing::Size(163, 20);
this->textBox1->TabIndex = 15;
this->textBox1->Text = L"Tilbury";
this->textBox1->TextChanged += gcnew System::EventHandler(
                this, &Form1::textBox1_TextChanged);
//
// label3
//
```

```
this->label3->AutoSize = true;
this->label3->Location = System::Drawing::Point(416, 345);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(47, 13);
this->label3->TabIndex = 16;
this->label3->Text = L"Save to:";
//
// label4
//
this->label4->AutoSize = true;
this->label4->Location = System::Drawing::Point(476, 343);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(93, 13);
this->label4->TabIndex = 17;
this->label4->Text = L"C:\\\\eStaticData\\\\";
//
// trackBar1
//
this->trackBar1->Location = System::Drawing::Point(15, 416);
this->trackBar1->Maximum = 62;
this->trackBar1->Name = L"trackBar1";
this->trackBar1->Size = System::Drawing::Size(328, 45);
this->trackBar1->TabIndex = 19;
this->trackBar1->ValueChanged += gcnew System::EventHandler(
                this, &Form1::trackBar1_ValueChanged);
//
// label5
//
this->label5->AutoSize = true;
this->label5->Location = System::Drawing::Point(24, 448);
this->label5->Name = L"label5";
this->label5->Size = System::Drawing::Size(29, 13);
this->label5->TabIndex = 20;
this->label5->Text = L"Gain";
//
// button3
//
this->button3->Location = System::Drawing::Point(403, 416);
this->button3->Name = L"button3";
this->button3->Size = System::Drawing::Size(60, 41);
this->button3->TabIndex = 21;
this->button3->Text = L"Auto gain";
this->button3->UseVisualStyleBackColor = true;
this->button3->Click += gcnew System::EventHandler(this, &Form1::button3_Click);
//
// checkBox1
//
this->checkBox1->AutoSize = true;
this->checkBox1->Location = System::Drawing::Point(15, 325);
this->checkBox1->Name = L"checkBox1";
this->checkBox1->Size = System::Drawing::Size(103, 17);
this->checkBox1->TabIndex = 22;
this->checkBox1->Text = L"Show raw signal";
this->checkBox1->UseVisualStyleBackColor = true;
this->checkBox1->CheckStateChanged += gcnew System::EventHandler(
                this, &Form1::checkBox1_CheckStateChanged);
//
// backgroundWorker2
//
this->backgroundWorker2->DoWork += gcnew System::ComponentModel::DoWorkEventHandler(
                this, &Form1::backgroundWorker2_DoWork);
//
```

```
// backgroundWorker3
//
this->backgroundWorker3->DoWork += gcnew System::ComponentModel::DoWorkEventHandler(
                this, &Form1::backgroundWorker3_DoWork);
//
// checkBox2
//
this->checkBox2->AutoSize = true;
this->checkBox2->Location = System::Drawing::Point(160, 325);
this->checkBox2->Name = L"checkBox2";
this->checkBox2->Size = System::Drawing::Size(183, 17);
this->checkBox2->TabIndex = 27;
this->checkBox2->Text = L"Suppress Velocity History Update";
this->checkBox2->UseVisualStyleBackColor = true;
this->checkBox2->CheckedChanged += gcnew System::EventHandler(
                this, &Form1::checkBox2_CheckedChanged);
//
// label6
//
this->label6->AutoSize = true;
this->label6->Location = System::Drawing::Point(661, 48);
this->label6->Name = L"label6";
this->label6->Size = System::Drawing::Size(35, 13);
this->label6->TabIndex = 28;
this->label6->Text = L"label6";
//
// label7
//
this->label7->AutoSize = true;
this->label7->Location = System::Drawing::Point(661, 75);
this->label7->Name = L"label7";
this->label7->Size = System::Drawing::Size(35, 13);
this->label7->TabIndex = 29;
this->label7->Text = L"label7";
//
// button2
//
this->button2->Location = System::Drawing::Point(484, 418);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(81, 39);
this->button2->TabIndex = 33;
this->button2->Text = L"Sensor reset";
this->button2->UseVisualStyleBackColor = true;
this->button2->Click += gcnew System::EventHandler(this, &Form1::button2_Click);
//
// checkBox3
//
this->checkBox3->AutoSize = true;
this->checkBox3->Location = System::Drawing::Point(15, 348);
this->checkBox3->Name = L"checkBox3";
this->checkBox3->Size = System::Drawing::Size(105, 17);
this->checkBox3->TabIndex = 34;
this->checkBox3->Text = L"Show correlation";
this->checkBox3->UseVisualStyleBackColor = true;
this->checkBox3->CheckStateChanged += gcnew System::EventHandler(
                this, &Form1::checkBox3_CheckStateChanged);
//
// label10
//
this->label10->AutoSize = true;
this->label10->Location = System::Drawing::Point(601, 48);
this->label10->Name = L"label10";
```

```
this->label10->Size = System::Drawing::Size(50, 13);
this->label10->TabIndex = 35;
this->label10->Text = L"Velocity :";
//
// label12
//
this->label12->AutoSize = true;
this->label12->Location = System::Drawing::Point(597, 75);
this->label12->Name = L"label12";
this->label12->Size = System::Drawing::Size(60, 13);
this->label12->TabIndex = 37;
this->label12->Text = L"Correlation:";
//
// label16
//
this->label16->AutoSize = true;
this->label16->Location = System::Drawing::Point(12, 381);
this->label16->Name = L"label16";
this->label16->Size = System::Drawing::Size(69, 13);
this->label16->TabIndex = 41;
this->label16->Text = L"Signal Power";
//
// textBox2
//
this->textBox2->Location = System::Drawing::Point(664, 20);
this->textBox2->Name = L"textBox2";
this->textBox2->Size = System::Drawing::Size(19, 20);
this->textBox2->TabIndex = 42;
this->textBox2->Text = L"25";
this->textBox2->TextAlign = System::Windows::Forms::HorizontalAlignment::Right;
this->textBox2->TextChanged += gcnew System::EventHandler(
                this, &Form1::textBox2_TextChanged);
//
// label17
//
this->label17->AutoSize = true;
this->label17->Location = System::Drawing::Point(682, 23);
this->label17->Name = L"label17";
this->label17->Size = System::Drawing::Size(27, 13);
this->label17->TabIndex = 43;
this->label17->Text = L"KHz";
//
// label18
//
this->label18->AutoSize = true;
this->label18->Location = System::Drawing::Point(630, 23);
this->label18->Name = L"label18";
this->label18->Size = System::Drawing::Size(21, 13);
this->label18->TabIndex = 44;
this->label18->Text = L"Fs:";
//
// timer1
//
this->timer1->Interval = 1000;
this->timer1->Tick += gcnew System::EventHandler(this, &Form1::timer1_Tick);
//
// label19
//
this->label19->AutoSize = true;
this->label19->Location = System::Drawing::Point(661, 103);
this->label19->Name = L"label19";
this->label19->Size = System::Drawing::Size(41, 13);
```

```
this->label19->TabIndex = 45;
this->label19->Text = L"label19";
//
// label21
//
this->label21->AutoSize = true;
this->label21->Location = System::Drawing::Point(599, 103);
this->label21->Name = L"label21";
this->label21->Size = System::Drawing::Size(58, 13);
this->label21->TabIndex = 47;
this->label21->Text = L"Averaging:";
//
// checkBox4
//
this->checkBox4->AutoSize = true;
this->checkBox4->Location = System::Drawing::Point(160, 348);
this->checkBox4->Name = L"checkBox4";
this->checkBox4->Size = System::Drawing::Size(197, 17);
this->checkBox4->TabIndex = 48;
this->checkBox4->Text = L"Common Mode Noise Compensation";
this->checkBox4->UseVisualStyleBackColor = true;
this->checkBox4->CheckedChanged += gcnew System::EventHandler(
                        this, &Form1::checkBox4_CheckedChanged);
//
// contextMenuStrip1
//
this->contextMenuStrip1->Name = L"contextMenuStrip1";
this->contextMenuStrip1->Size = System::Drawing::Size(61, 4);
//
// label9
//
this->label9->AutoSize = true;
this->label9->Location = System::Drawing::Point(475, 381);
this->label9->Name = L"label9";
this->label9->Size = System::Drawing::Size(93, 13);
this->label9->TabIndex = 52;
this->label9->Text = L"C:\\\\eStaticData\\\\";
//
// textBox3
//
this->textBox3->Location = System::Drawing::Point(569, 378);
this->textBox3->Name = L"textBox3";
this->textBox3->Size = System::Drawing::Size(163, 20);
this->textBox3->TabIndex = 53;
this->textBox3->Text = L"8_19_11_9_30";
//
// button4
//
this->button4->Location = System::Drawing::Point(404, 376);
this->button4->Name = L"button4";
this->button4->Size = System::Drawing::Size(65, 22);
this->button4->TabIndex = 54;
this->button4->Text = L"Load from:";
this->button4->UseVisualStyleBackColor = true;
this->button4->Click += gcnew System::EventHandler(this, &Form1::button4_Click);
//
// Form1
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->BackColor = System::Drawing::SystemColors::Control;
this->ClientSize = System::Drawing::Size(744, 469);
```

```
        this->Controls->Add(this->label19);
        this->Controls->Add(this->checkBox4);
        this->Controls->Add(this->checkBox3);
        this->Controls->Add(this->label18);
        this->Controls->Add(this->label16);
        this->Controls->Add(this->label21);
        this->Controls->Add(this->button4);
        this->Controls->Add(this->textBox3);
        this->Controls->Add(this->label7);
        this->Controls->Add(this->label9);
        this->Controls->Add(this->checkBox2);
        this->Controls->Add(this->label12);
        this->Controls->Add(this->label10);
        this->Controls->Add(this->checkBox1);
        this->Controls->Add(this->label17);
        this->Controls->Add(this->textBox2);
        this->Controls->Add(this->label1);
        this->Controls->Add(this->label6);
        this->Controls->Add(this->button3);
        this->Controls->Add(this->button2);
        this->Controls->Add(this->panel2);
        this->Controls->Add(this->panel1);
        this->Controls->Add(this->label4);
        this->Controls->Add(this->label3);
        this->Controls->Add(this->label5);
        this->Controls->Add(this->progressBar1);
        this->Controls->Add(this->textBox1);
        this->Controls->Add(this->button1);
        this->Controls->Add(this->trackBar1);
        this->Controls->Add(this->dataGraph1);
        this->Icon = (cli::safe_cast<System::Drawing::Icon^  >(resources->GetObject(L"$this.Icon")));
        this->Name = L"Form1";
        this->Text = L"jk85 VelocityMeter";
        this->panel1->ResumeLayout(false);
        this->panel1->PerformLayout();
        this->panel2->ResumeLayout(false);
        this->panel2->PerformLayout();
        (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this->trackBar1))->EndInit();
        this->ResumeLayout(false);
        this->PerformLayout();

    }
#pragma endregion
private: System::Void dataGraph1_Load(System::Object^  sender, System::EventArgs^  e)
        {
         array<Object^>^ myArray={1,1,1}; //DUMMY DATA TO INITIALISE DATAGRAPH
         int i;

         Bitmap^ saveGraph; //TO STORE THE DATAGRAPH AS A BITMAP
         kjTable->Clear();

         //INITIALISE VHISTORY
         for(i=0;i<61;i++)
         {
           vHistory1[i]=(double)0;
         }

         //DEFINE AND INITIALISE DATA COLUMNS
          DataColumn^ kjColumn=gcnew DataColumn;
          kjColumn->DataType = System::Type::GetType("System.Double");
          kjColumn->ColumnName="Time [s]";
          kjTable->Columns->Add(kjColumn);
```

```
        //REDEFINE DATA COLUMN AS A NEW COLUMN:
        kjColumn=gcnew DataColumn("Velocity [m/s]",System::Type::GetType("System.Double"));
        kjTable->Columns->Add(kjColumn); //DON'T FORGET TO ADD IT TO THE DATA TABLE.
        kjColumn=gcnew DataColumn("kjColumn3",System::Type::GetType("System.String"));
        kjTable->Columns->Add(kjColumn);

       //ADD TABLE TO DATAGRAPH
       dataGraph1->DataTable=kjTable;
       //identifies different data sets. drawn with different line colours. Can be a string
       dataGraph1->DataNameColumn="kjColumn3";
         dataGraph1->DataColumn="Velocity [m/s]"; // 'y' values.
       dataGraph1->DataGroupingColumn="Time [s]"; // 'x' values

       kjDataRow=kjTable->NewRow();
       kjDataRow->ItemArray=myArray;
       kjTable->Rows->Add(kjDataRow);

       dataGraph1->DrawGraph();

       saveGraph=dataGraph1->GetGraphAsBitmap();
       //saveGraph->Save("C:\\kjdel\\saveGraph.bmp");
       //USE THIS BLANK GRAPH AS A BACKGROUND. THAT WAY THE GRAPH WONT FLICKER WHEN UPDATED

       label1->Text=" ";
      } //dataGraph1_Load
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e)
      {
        go=!go; //TOGGLE go
        if (go)
        {
         //initialisation code
         int jkUpperBound; //USED WHEN RETRIEVING AVAILABLE 'COM' PORTS FROM THE SYSTEM
         //INITIALIZE WITH AT LEAST AN EMPTY STRING, OTHERWISE ERROR WHEN ASSIGNING STRINGS
         portNames = gcnew array<String^>(50);
         portNames=serialPort1->GetPortNames(); //GETS AVAILABLE 'COM' PORTS FROM SYSTEM
         jkUpperBound=portNames->GetUpperBound(0);//DETERMINES HOW MANY 'COM' PORTS EXIST
         //SENSOR WILL BE THE LAST PORT TO HAVE BEEN RECOGNIZED BY THE SYSTEM
         serialPort1->PortName=portNames[jkUpperBound];
         serialPort1->Open();
         while(!serialPort1->IsOpen) {;} //wait until port is open

         //HANDSHAKE WITH dsPIC:
         serialPort1->DiscardInBuffer();
         serialPort1->DiscardOutBuffer();
         dataRequest[0]=CHECKREADY;
         kjData[0]=READY; //disable handshake in debug
         while(kjData[0] !=READY)
         {
           serialPort1->Write(dataRequest,0,1); //ASK DSPIC IF READY
           serialPort1->Read(kjData,0,1);
         }

         serialPort1->DiscardInBuffer();
         serialPort1->DiscardOutBuffer();

         //dataRequest[0]=SENSOREXIT; //reset PIC/
         serialPort1->Write(dataRequest,0,1);
         serialPort1->Write(dataRequest,0,1);
         dataRequest[0]=RESULTS;


         button1->Text="STOP";
```

```
            button1->ForeColor=System::Drawing::Color::Red;
            checkBox1->Enabled=false;
            checkBox3->Enabled=false;
            trackBar1->Enabled=false;
            button3->Enabled=false;
            textBox1->Enabled=false;

            dataGraph1->TextFormatDataValuesIndex->Visible=true;
            dataGraph1->TextFormatDataValueAxis->Visible=true;
            dataGraph1->IndexLength=5;
            dataGraph1->TextFormatDataGroupAxis->Visible=true;

            //Creates new file, or overwrites existing file. Need to use namespace System::IO:
            sw = File::AppendText(String::Concat(path,pathadd,".txt"));

            GetSystemTime(stime); //RETRIEVES THE CURRENT SYSTEM TIME
            sec=stime->wSecond;
             //Set watchdog timer to prevent hanging in the loop:
            watchdogTime=sec+5;
            if(watchdogTime>59) {watchdogTime=watchdogTime-60;}

            //serialPort1->RtsEnable=true;
            timer1->Enabled=true;
    //   trackBar1->Value=62; //INITIALISE DIGITAL RESISTOR:
          }
          else
          {
            button1->Text="GO";
            button1->ForeColor=System::Drawing::Color::Green;
            checkBox1->Enabled=true;
            checkBox3->Enabled=true;
            trackBar1->Enabled=true;
            button3->Enabled=true;
            watchdogTime=60; //i.e. never
            textBox1->Enabled=true;
            serialPort1->DiscardInBuffer();
            timer1->Enabled=false;
            sw->Close();
            serialPort1->Close();
          }
        }//button1_Click
private: System::Void textBox1_TextChanged(System::Object^  sender, System::EventArgs^
e)
     {
       pathadd=textBox1->Text; //FOR USER SELECTED SAVE FILE NAME
     }


private: System::Void radioButton1_CheckedChanged(System::Object^  sender, System::EventArgs^
e)
     {
       if(radioButton1->Checked)
       {
       //DISABLE THE PANEL THAT CHOOSES RAW SAVE INTERVAL, SINCE NO RAW SAVE HAS BEEN SELECTED:
         panel2->Enabled=false;
         radioButton4->Checked=false;
         radioButton5->Checked=false;
         writeRes=true; //ENABLE WRITE TO FILE
         dataRequest[0]=RESULTS;
       }
     }//radioButton1_CheckedChanged
private: System::Void radioButton2_CheckedChanged(System::Object^  sender, System::EventArgs^
```

```
e)
     {
       if(radioButton2->Checked)
       {
         panel2->Enabled=true;
         writeRes=true; //ENABLE WRITE TO FILE
         radioButton4->Checked=true; //DEFAULT
         dataRequest[0]=RESULTS;
       }
     }
private: System::Void radioButton3_CheckedChanged(System::Object^  sender, System::EventArgs^
e)
     {
       if(radioButton3->Checked)
       {
         //DISABLE THE PANEL THAT CHOOSES RAW SAVE INTERVAL, SINCE NO RAW SAVE HAS BEEN SELECTED:
         panel2->Enabled=false;
         radioButton4->Checked=false;
         radioButton5->Checked=false;
         dataRequest[0]=RESULTS;  //STILL SEND RESULTS FOR THE GRAPH
         writeRes=false;
       }
     }
private: System::Void trackBar1_ValueChanged(System::Object^  sender, System::EventArgs^
e)
     {
       float trans=0;
       int itrans=0;
       serialPort1->DiscardInBuffer();
       dataRequest[0]=(unsigned char)trackBar1->Value;
       //NEXT DATA REQUEST IS ACTUALLY A REQUEST TO CHANGE GAIN:
       serialPort1->Write(dataRequest,0,1); //REQUEST GAIN CHANGE
       while(serialPort1->BytesToRead<1) {}; //WAIT FOR CONFIRMATION
       serialPort1->Read(kjData,0,1); //CONFIRM GAIN SETTING
       itrans= 10*(4.8* (92.61/(63-(float)kjData[0])) + 0.5);
       //multiply by 10 and round to integer (hence 0.5).
       //use trackBar1 for testing, use kjData[0] for practical
       //4.8MV/A is the I to V analog conversion with t-network
       //92.61 from the second (gain) stage
       trans=(float)itrans/10;  //divide by 10 to end up with 1 decimal place
       label5->Text=String::Concat(trans.ToString()," mV/nA");
       dataRequest[0]=RESULTS; //RESTORE DATAREQUEST
     }//trackBar1_ValueChanged

private: System::Void button3_Click(System::Object^  sender, System::EventArgs^  e)
     {
      dataRequest[0]=AUTOGAIN;
      serialPort1->DiscardInBuffer();
      serialPort1->Write(dataRequest,0,1);
      while(serialPort1->BytesToRead<1) {};  //WAIT FOR GAIN SETTING TO BE RETURNED
      serialPort1->Read(kjData,0,1);
      if(kjData[0]>62) {kjData[0]=62;} //limit gain
      trackBar1->Value=kjData[0];

     }
private: System::Void checkBox1_CheckStateChanged(System::Object^  sender, System::EventArgs^
e)
     {
       //DISPLAY RAW DATA FROM SENSOR
       int i,j;//GENERAL PURPOSE COUNTERS
       int pwr1;
       array<Object^>^ myArray={nullptr,nullptr,nullptr}; //USED TO MAKE DATA ROWS
```

```
 button1->Enabled=false;
 checkBox3->Enabled=false;
 dataGraph1->TextFormatDataValuesIndex->Visible=false;
 dataGraph1->TextFormatDataValueAxis->Visible=false;
 dataGraph1->TextFormatDataGroupAxis->Visible=false;
 dataGraph1->LineFormatDataValue->Visible=false;
 dataGraph1->IndexLength=0; //tick lines on graph
 label1->Visible=false; //DON'T DISPLAY CURRENT TIME
 trackBar1->Enabled=false;
 radioButton1->Enabled=false;
 radioButton2->Enabled=false;
 radioButton3->Enabled=false;
 textBox1->Enabled=false;

while(checkBox1->Checked) //get and display raw data from sensor
{
 GetSystemTime(stime); //RETRIEVES THE CURRENT SYSTEM TIME
 sec=stime->wSecond;

 //Set watchdog timer to prevent hanging in the loop:
 watchdogTime=sec+5;
 if(watchdogTime>59) {watchdogTime=watchdogTime-60;}

 Application::DoEvents();
 kjTable->Clear();
 dataRequest[0]=RESULTS;
 serialPort1->DiscardInBuffer();
 serialPort1->Write(dataRequest,0,1);

 dataRequest[0]=RAW;
 serialPort1->Write(dataRequest,0,1);

 //READ RAW DATA FROM SENSOR (THE DATA WE ARE INTERESTED IN)
 for(i=0;i<2;i++)
 {
   serialPort1->Write(dataRequest,0,1); //READ THE NEXT N DATA BYTES
   while(serialPort1->BytesToRead<N*2); // *2 because the data is fractional type
   serialPort1->Read(kjData,0,N*2); //READ SENSOR RAW DATA

   //even j gives LSB, odd j gives MSB
   for(j=0;j<N*2;j=j+(N/50))
   {
   myArray[0]=(double)j; //'x' value

   //raw data dynamic range is signed 9 bits (512). Scale for -0.5:0.5.
   //(2*i+0.5) is an offset to display traces separately :
   myArray[1]=(float)(Int16)(kjData[j]+(kjData[j+1]*256))/1024 +(2*i+0.5);
   //myArray[1]=R->NextDouble()*10 + (i*10);
   myArray[2]=i; //series identifier
   kjDataRow=kjTable->NewRow();
   kjDataRow->ItemArray=myArray;
   kjTable->Rows->Add(kjDataRow);
   }
 }//for

 //READ RESULTS DATA FROM SENSOR
 //kjData[0:11] will hold: v1,c1,pwr1up,pwr1down,g,validReadings1
 serialPort1->Write(dataRequest,0,1); //just a trigger to send the results
 while(serialPort1->BytesToRead<(8)) {}; //WAIT UNTIL ALL DATA IS IN THE INPUT BUFFER
 serialPort1->Read(kjData,0,8);
 //INTERPRET RESULTS FROM SENSOR
```

```
        pwr1=(kjData[4]+kjData[5])/2;
        if(pwr1>255) {pwr1=255;}  //limit pwr1 to 128
        v1=kjData[0]+((float)kjData[1]/100);
        if (v1 > 40) {v1 = 40;}
        c1=kjData[2]+((float)kjData[3]/100);
        label19->Text=kjData[7].ToString();  //validReadings1
        progressBar1->Value=log((double)pwr1+1);
        label6->Text=v1.ToString();
        label7->Text=c1.ToString();

        if((checkBox1->Checked))
        {
          dataGraph1->MaxValue=4;  //each trace has a range of -1 to 1
          dataGraph1->MinValue=-1;
          dataGraph1->DrawGraph();
        }  //DrawGraph IS SUPPRESSED AS SOON AS THE USER UNCHECKS THE BOX
      }//while

        button1->Enabled=true;
        checkBox3->Enabled=true;
        watchdogTime=60; //i.e. never
        //Restore dataRequest
        dataRequest[0]=RESULTS;
        label1->Visible=true;  //re-enable time display on graph
        trackBar1->Enabled=true; //re-enable gain bar
        radioButton1->Enabled=true;
        radioButton2->Enabled=true;
        radioButton3->Enabled=true;
        textBox1->Enabled=true;
        dataGraph1->LineFormatDataValue->Visible=true;
    }  //END OF CHECKBOX1 LOOP

private: System::Void backgroundWorker2_DoWork(System::Object^  sender,
                       System::ComponentModel::DoWorkEventArgs^  e)
    {
      while(0)
      {
       GetSystemTime(stime);
       if(stime->wSecond == watchdogTime)
       {
         dataRequest[0]=SENSOREXIT;
         serialPort1->Write(dataRequest,0,1);
         if(serialPort1->IsOpen) {serialPort1->Close();}
         Application::Restart();
       } //if(stime.....
       } //while(1)
     }//function

private: System::Void backgroundWorker3_DoWork(System::Object^  sender,
                       System::ComponentModel::DoWorkEventArgs^  e)
    {
     // MessageBox::Show("Loading..."); //show splash screen here
    }

private: System::Void checkBox2_CheckedChanged(System::Object^  sender, System::EventArgs^
e)
    {
      if(checkBox2->Checked)
      {graphShow=false;}
      else
      {graphShow=true;}
    }
```

```
private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e)
     {
      dataRequest[0]=SENSOREXIT;
      serialPort1->Write(dataRequest,0,1);

      //HANDSHAKE WITH dsPIC:
      serialPort1->DiscardInBuffer();
      serialPort1->DiscardOutBuffer();
      dataRequest[0]=CHECKREADY;
kjData[0]=READY; //disable handshake in debug
      while(kjData[0] !=READY)
      {
        serialPort1->Write(dataRequest,0,1); //ASK DSPIC IF READY
        serialPort1->Read(kjData,0,1);
      }
      serialPort1->DiscardInBuffer();
      serialPort1->DiscardOutBuffer();
     }
private: System::Void checkBox3_CheckStateChanged(System::Object^  sender, System::EventArgs^
e)
     {
      //SHOW CORRELATION
      System::Int16 cUnscaled=0;
      System::UInt16 cScalei1=0;
      int i=0;//GENERAL PURPOSE COUNTER
      int pwr1=0;
      array<Object^>^ myArray={nullptr,nullptr,nullptr}; //USED TO MAKE DATA ROWS

        button1->Enabled=false; //disable go button
        checkBox1->Enabled=false; //disable raw data show
        dataGraph1->TextFormatDataValuesIndex->Visible=false;
        dataGraph1->TextFormatDataValueAxis->Visible=false;
        dataGraph1->TextFormatDataGroupAxis->Visible=false;
        dataGraph1->IndexLength=0; //tick lines on graph
        label1->Visible=false; //DON'T DISPLAY CURRENT TIME
        trackBar1->Enabled=false;
        radioButton1->Enabled=false;
        radioButton2->Enabled=false;
        radioButton3->Enabled=false;
        textBox1->Enabled=false;

       while(checkBox3->Checked) //get and display correlation data from sensor
       {
        GetSystemTime(stime); //RETRIEVES THE CURRENT SYSTEM TIME
        sec=stime->wSecond;

        //Set watchdog timer to prevent hanging in the loop:
          watchdogTime=sec+5;
          if(watchdogTime>59) {watchdogTime=watchdogTime-60;}

        kjTable->Clear();
        serialPort1->DiscardInBuffer();
        dataRequest[0]=RESULTS;
        serialPort1->Write(dataRequest,0,1);
        dataRequest[0]=CORR;
        serialPort1->Write(dataRequest,0,1);

        // READ SCALING FACTOR AND CORRELATION DATA *2 because fractional,
        while(serialPort1->BytesToRead < (CORR_DATA_LENGTH *2) + 2);
        serialPort1->Read(kjData,0,2); //READ SCALING FACTORS (1 FRACTIONALS VALUE)

        //READ CORRELATION SCALER1 VALUE:
```

```
        //pic sends least significant byte first. works.
        cScalei1= (System::UInt16)kjData[0] + ((System::UInt16)kjData[1]<<8);
        if (cScalei1==0) {cScalei1=32767;}

        //READ CORRELATION DATA FROM SENSOR (THE DATA WE ARE INTERESTED IN)
        serialPort1->Read(kjData,0,CORR_DATA_LENGTH*2);
        for(i=0;i<CORR_DATA_LENGTH;i++) //actual data is in fractional form. 2 bytes for each
        {
          myArray[0]=(double) i; //'x' axis value
          //i*2 and offsets to account for fractional data type being 2 bytes long
          cUnscaled=(System::Int16)
            ((System::UInt16)(kjData[i*2])+(System::UInt16)(kjData[i*2+1]<<8));
          myArray[1]=(float)((float)cUnscaled/(float)cScalei1);
          myArray[2]=1; //series identifier (only 1 series in this case)
          kjDataRow=kjTable->NewRow();
          kjDataRow->ItemArray=myArray;
          kjTable->Rows->Add(kjDataRow);
        }

        //READ RESULTS DATA FROM SENSOR
        serialPort1->Write(dataRequest,0,1); // trigger dsPIC
        //kjData will hold: v1,c1,pwr1up,pwr1down,gain setting,validReadings1
        while(serialPort1->BytesToRead<(8)) {}; //WAIT UNTIL ALL DATA IS IN THE INPUT BUFFER
        serialPort1->Read(kjData,0,8);
        //INTERPRET RESULTS FROM SENSOR
        pwr1=(kjData[4]+kjData[5])/2;
        if(pwr1>255) {pwr1=255;}  //limit pwr1 to 128
        v1=kjData[0]+((float)kjData[1]/100);
        if (v1 > 40) {v1 = 40;}
        c1=kjData[2]+((float)kjData[3]/100);
        label19->Text=kjData[7].ToString();
        progressBar1->Value=log((double)pwr1+1);
        label6->Text=v1.ToString();
        label7->Text=c1.ToString();

        Application::DoEvents();
        if(checkBox3->Checked)
        {
          dataGraph1->MaxValue=1;
          dataGraph1->MinValue=-1;
          dataGraph1->DrawGraph();
          //DrawGraph IS SUPPRESSED AS SOON AS THE USER UNCHECKS THE BOX
        }
      }//while

        button1->Enabled=true;
        checkBox1->Enabled=true; //re-enable raw data show
        watchdogTime=60; //i.e. never
        //Restore dataRequest
        dataRequest[0]=RESULTS;
        label1->Visible=true;  //re-enable time display on graph
        trackBar1->Enabled=true; //re-enable gain bar
        radioButton1->Enabled=true;
        radioButton2->Enabled=true;
        radioButton3->Enabled=true;
        textBox1->Enabled=true;
    } //checkBox3
private: System::Void textBox2_TextChanged(System::Object^  sender, System::EventArgs^
e)
    {
      unsigned char Fs;
      if(textBox2->Text->Length) //i.e. if length not 0
```

```
        {
          Fs=Convert::ToByte(textBox2->Text);
          if(Fs<=100 && Fs>=5) //limit sampling to 5kHz:100kHz
          {
            dataRequest[0]=SET_FS;
            serialPort1->Write(dataRequest,0,1);
            dataRequest[0]=Fs;
            serialPort1->Write(dataRequest,0,1);
          } //if Fs<100
        } //if length not 0
      } //textBox2_TextChanged

private: System::Void timer1_Tick(System::Object^  sender, System::EventArgs^  e)
      {
       array<Object^>^ myArray={nullptr,nullptr,nullptr}; //USED TO MAKE DATA ROWS

       GetSystemTime(stime); //RETRIEVES THE CURRENT SYSTEM TIME
       hour=stime->wHour;
       minute=stime->wMinute;
       sec=stime->wSecond;
       day=stime->wDay;
       month=stime->wMonth;
       year=stime->wYear;

       serialPort1->DiscardInBuffer(); //CLEAR THE BUFFER JUST IN CASE THERE WAS ANY ERROR
       kjData->Clear(kjData,0,N);

       //SEND DATA REQUEST MESSAGE TO SENSOR:
       dataRequest[0]=RESULTS;
       serialPort1->Write(dataRequest,0,1); //SEE RADIO BUTTON CODE FOR dataRequest
         //no data actually gets sent
         //--just lets the dsPIC know to go to the data sending routine,
         //not gain setting, fs setting etc.

       //CHECK IF RAW DATA IS REQUIRED:
       if (((radioButton4->Checked) && (sec==0)) ||
         ((radioButton5->Checked) && (sec==0) && (minute==0)))

       {
         dataRequest[0]=RAW;
         serialPort1->Write(dataRequest,0,1);
         for(i=0;i<2;i++)
         {
           serialPort1->Write(dataRequest,0,1); //TRIGGER FOR PIC TO SEND DATA
           //WAIT UNTIL ALL THE DATA COMES THROUGH.  *2 because fractional type 2 bytes long:
           while(serialPort1->BytesToRead<N*2) {};
           serialPort1->Read(kjData,0,N*2); //READ SENSOR RAW DATA
           File::WriteAllBytes(String::Concat(
             path,"RawData\\",pathadd,"Raw_",year,"_",month,"_",day,"_",hour,"_",minute,"_",
             sec,"_e",i,"_g",g.ToString(),".txt"),kjData);
         }//for
       }
       else
       {
         dataRequest[0]=NO_OTHER_DATA;
         serialPort1->Write(dataRequest,0,1);
       }

       //READ RESULTS DATA FROM SENSOR
       //trigger. Doesn't really matter what the value of dataRequest is:
       serialPort1->Write(dataRequest,0,1);
       //kjData will hold: v1,c1,pwr1up,pwr1down,g,validReadings1
```

```
while(serialPort1->BytesToRead<8) {};
//1 VELOCITY BYTES + 1 CORRELATION BYTES + 1 POWER BYTES  +
//1 validReadings BYTES FOR EACH OF 2 SENSORS, + 1 g BYTE,
serialPort1->Read(kjData,0,8);
//INTERPRET RESULTS FROM SENSOR
pwr1=(kjData[4]+kjData[5])/2;
if(pwr1>255) {pwr1=255;}  //limit pwr1 to 128
v1=kjData[0]+((float)kjData[1]/100);
if (v1 > 40) {v1 = 40;}
c1=kjData[2]+((float)kjData[3]/100);
g=kjData[6];
label19->Text=kjData[7].ToString();

//WRITE RESULTS TO FILE IF REQUIRED
if(writeRes)
{
  sw->WriteLine(String::Concat("v:",v1.ToString(),",c:",c1.ToString(),",","g",g,",",
    year,",",month,",",day,",",hour,",",minute,",",sec)); //v1,v2, and date stamp
}// if(writeRes)

  //UPDATE THE DATATABLE FOR THE DATAGRAPH
  kjTable->Clear();

  for(i=0;i<59;i++)
  {
    vHistory1[i]=vHistory1[i+1]; //shift vHistory back by one position

    myArray[0]=(double)i; //'x' values
    myArray[1]=vHistory1[i]; //'y' values
    myArray[2]=1; //SERIES1
    kjDataRow=kjTable->NewRow();
    kjDataRow->ItemArray=myArray;
    kjTable->Rows->Add(kjDataRow);
  }
    //FILL MOST RECENT PLACE WITH NEW DATA:
    vHistory1[59]=v1;
    myArray[0]=(double)i; //'x' value
    myArray[1]=vHistory1[59];
    myArray[2]=1;
    kjDataRow=kjTable->NewRow();
    kjDataRow->ItemArray=myArray;
    kjTable->Rows->Add(kjDataRow);

  if (go)
  {
    dataGraph1->MaxValue=40;
    dataGraph1->MinValue=0;
    if(graphShow) {dataGraph1->DrawGraph();}
    progressBar1->Value=log((double)pwr1+1);
    label6->Text=v1.ToString();
    label7->Text=c1.ToString();

    if (sec<10) {secFill="0";} else {secFill="";}
    if (minute<10) {minFill="0";} else {minFill="";}
    label1->Text=String::Concat(hour.ToString(),":",minFill,minute.ToString(),":",
                secFill,sec.ToString()); //UPDATE SYSTEM TIME ON GRAPH
  }
  Application::DoEvents();
}
```

```
private: System::Void checkBox4_CheckedChanged(System::Object^  sender,
         System::EventArgs^  e)
     {
       if(checkBox4->Checked) {dataRequest[0]=CMNR;}
       else {dataRequest[0]=NO_CMNR;}
       serialPort1->Write(dataRequest,0,1);
     }
};
}
```