INVESTIGATION INTO THE DETECTION AND CLASSIFICATION OF DEFECT COLONIES USING ACFM TECHNOLOGY

L. BLAIR CARROLL

0-612-42360-3

INVESTIGATION INTO THE DETECTION AND CLASSIFICATION OF DEFECT COLONIES USING ACFM TECHNOLOGY

BY

L. BLAIR CARROLL, B. ENG.

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING

FACULTY OF ENGINEERING AND APPLIED SCIENCE

MEMORIAL UNIVERSITY OF NEWFOUNDLAND

SEPTEMBER, 1998

ST. JOHN'S NEWFOUNDLAND CANADA

In memory of my Grandfathers:

William Carroll & Joseph Mercer

ACKNOWLEDGMENTS

The following people deserve recognition for their contributions:

My family for their support and encouragement.

Dr. Craig Monahan and Dr. Raymond Gosine of Memorial University of Newfoundland for their help and guidance throughout the course of this project.

The staff of Technical Services of Memorial University of Newfoundland for their assistance in manufacturing the defect colonies.

RTD Quality Services Ltd. for providing the SCC specimens.

Dr. Rafaat Khan of C-CORE for his input during the data analysis phase of the research.

The National Sciences and Engineering Council (NSERC) for their financial assistance.

Ms. Susan Miller, Manager of Interprovincial Pipeline's Pipeline Integrity Department, for providing computing and printing facilities during the final revisions of my thesis.

SUMMARY

A research project was undertaken to provide insight into the use of Alternating Current Field Measurement (ACFM) technology for the detection and classification of transgranular stress corrosion cracking (TGSCC) plaguing Canada's oil and natural gas transmission pipelines. Work was conducted using a series of machined defects and natural TGSCC samples. The results suggest that ACFM warrants further investigation in a larger scale project. Guidelines for future endeavors are provided.

Of the 63 machined defects, used in 21 colony configurations, visual inspection identified all of the defects and an automated computer algorithm missed only one. The data set was not large enough to develop a sizing algorithm but provided valuable insight into defect signal interactions.

Present ACFM technology is capable of detecting natural SCC colonies, but more work is required before individual crack classification can be achieved when the cracks appear in clusters.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Alternating current field measurement	Electromagnetic non-destructive testing technique using fluctuations in an induced alternating current field to detect and size surface breaking cracks.
Alternating current potential drop	Non-destructive testing technique using the electrical resistance of a metal to measure the path traveled by an impressed alternating current flowing under a surface breaking crack, thus indicating crack depth.
A/D Card	Analogue to digital convertor card used to convert analogue signals into digital data for computer processing purposes.
Canadian Energy Pipelines Association	An alliance of 15 transmission and distribution pipeline owners/operators in Canada established to address common concerns in pipeline integrity and product distribution issues.
Cathodic protection	Application of a current to a pipeline through soil media in an attempt to regulate the voltage potential range so that it is not in a range, which would promote corrosion of the pipeline.
Circumferential spacing	Perpendicular separation in the circumferential direction between parallel, axially aligned cracks on a pipe.
Cluster	A grouping of close proximity cracks
Coating	Material applied to the external surface of a buried pipeline to provide protection from a corrosive medium.
Colony	A grouping of close proximity cracks.
Disbondment	Region on a pipeline where the protective coating has lost adhesion to the surface.
Eddy current	Electromagnetic non-destructive testing technique most effective in non-ferromagnetic metals for detecting metal loss due to corrosion.
Elastic Wave Inspection Vehicle	Internal inspection tool developed by British Gas Ltd to detect cracking in transmission pipelines using ultrasonic transducers.

Intergranular stress corrosion cracking the metal.	Form of stress corrosion cracking associated with a high pH corrosive medium. Cracks tend to follow paths between grains in
Inter-defect spacing	Perpendicular separation between parallel cracks.
Isolated defect	A defect which is not alone on a specimen, but is sufficiently spaced on the specimen so that the ACFM field perturbations are not influenced by the other defects.
Machined colony	Colony of slots in a metal plate created with a slitting saw used to simulate cracking.
Microprobe	An ACFM probe using 2 mm diameter pick-up coils.
Magnetic particle inspection	Magnetic non-destructive testing technique used to detect surface or near surface cracking. Magnetic particles applied to the surface of a ferromagnetic metal are attracted to gaps in an applied magnetic field due to the presence of cracks.
National Energy Board of Canada	Federal government agency in Canada responsible for monitoring and regulating transmission pipeline operators.
Nondestructive testing	Inspection of materials, components, structures, etc., without affecting the intended purpose or operation of the component.
Pencil probe	An ACFM probe using 5 mm diameter pick-up coils.
Perturbation	Disturbance in a uniform induced (ACFM) or injected (ACPD) AC field in a test specimen.
Pig	Internal inspection tool used to clean and/or inspect a pipeline, often propelled by the medium transported through the pipeline.
pH	Measure of acidity.
Stress corrosion cracking	Cracking in metals resulting from a synergy between a fluctuating stress and a corrosive medium.
Single Defect	A defect which is alone on a test specimen (i.e. no other defects present).

Transgranular stress corrosion cracking	Form of stress corrosion cracking associated with a near-neutral pH corrosive medium. Crack path tends to propagate through grains in the metal.
Time of flight diffraction	An advance ultrasonics based non-destructive testing technique using the diffraction of sounds waves at crack tips and the speed of sound in a metal to identify and size cracks.
Ultrasonic testing	Non-destructive testing technology utilizing the reflection of ultrasonic signals to locate and size cracking and metal loss in a metal.

ABBREVIATIONS

AC	Alternating Current
ACFM	Alternating Current Field Measurement
ACPD	Alternating Current Potential Drop
A/D	Analogue to Digital Convertor
BG	British Gas Ltd.
CEPA	Canadian Energy Pipelines Association
CP	Cathodic Protection
EC	Eddy Current
EWIV	Elastic Wave Inspection Vehicle
FBE	Fusion Bonded Epoxy
IGSCC	Intergranular Stress Corrosion Cracking
ILI	In-Line Inspection
MPI	Magnetic Particle Inspection
NEB	National Energy Board of Canada
NDT	Nondestructive Testing
PE	Polyethylene
PR	Pattern Recognition
SMYS	Specified Minimum Yield Strength
TCPL	Trans Canada Pipelines Ltd.
TGSCC	Transgranular Stress Corrosion Cracking

TOFD	Time-of-Flight Diffraction
UT	Ultrasonic Testing

CHAPTER 1: INTRODUCTION

Transgranular stress corrosion cracking (TGSCC) has become a serious concern for operators of oil and natural gas transmission pipelines in western and central Canada. To date, a commercially available technology capable of reliably characterizing TGSCC damage does not exist. The research project described herein is a preliminary investigation into the application of an existing nondestructive testing technology, the Alternating Current Field Measurement (ACFM) technique, to the task of detecting and sizing TGSCC during pipeline excavation programs. ACFM is presently utilized in a variety of applications to characterize isolated cracks in electrically conducting metals, particularly weld toe fatigue cracks in carbon steel.

This research is seen as the first stage in a series of projects which could lead to a new application of ACFM technology to assist pipeline operators in preventing the failure of their systems, thus, preventing possible loss of product, downtime, expensive repair costs, environmental damage, and may be even loss of human life. Development of a commercially viable technology was beyond the research scope, as it would have required a much larger investment of time and financial resources. The focus was to determine the present ability of the ACFM technique to deal with multiple crack clusters and provide guidelines for further research.

Through an extensive literature review and consultation with the manufacturers of the ACFM technology, Technical Software Consultants Ltd. of the United Kingdom, it appeared that the application of the technology to TGSCC classification had not been addressed previously. This was not surprising as ACFM is still relatively new to North America and TGSCC has not been a problem on UK transmission pipelines. Several objectives were chosen, not only to determine the present capabilities of ACFM in the proposed task, but also to set guidelines for future work. The research was subdivided into two phases. The first phase dealt with crack detection and the second concentrated on sizing related issues.

With respect to crack colony detection, the limitations of existing probe technology first had to be identified. Two commercial probe configurations were applied to machined defect colonies and real crack clusters, with both probes yielded encouraging results. Signal and image processing techniques were applied to ACFM data to identify possible models for automating TGSCC detection. Finally, the capability of ACFM to detect colonies through polyethylene tape coatings, typically found on pipelines with TGSCC damage, was successfully demonstrated.

The sizing of cracks within clusters was anticipated to be the most difficult task. It was noted early on that the development of a successful sizing algorithm was likely beyond the capacity of a preliminary investigation. To address the task on a more manageable scale, three facets were examined in detail. First, the concept of linear superposition of signals from neighboring machined defects was examined. Unfortunately, it demonstrated poor results. Subsequently, the nature of defect signal interaction was examined to establish guidelines for future work in developing a non-linear model. An examination of existing ACFM theory applied to TGSCC sizing was possible with the acquisition of pipeline sections. Two isolated defects were sized to within 0.3 mm while a third offered difficulty in that it propagated through the pipe wall at and angle, leading to an overestimate in depth.

The concepts presented here are intended to be a stepping stone to the development of a reliable NDT method for characterizing TGSCC sites in transmission pipelines.

CHAPTER 2: REVIEW OF LITERATURE

2.1 REVIEW OF TRANSGRANULAR STRESS CORROSION CRACKING AND CURRENT INSPECTION PROCEDURES

The graduate work described herein was inspired by reported failures of Canadian hydrocarbon transmission pipelines as a result of stress-corrosion cracking. In Canada, soil and climatic conditions favor the development of transgranular rather than intergranular SCC. The phenomenon of intergranular stress-corrosion cracking (IGSCC) in buried pipelines appears to have been well established in studies conducted in other countries, particularly the United States, where this corrosion process has been evident for decades.

Transgranular stress corrosion cracking (TGSCC) appears to be unique to colder climates and has only become a problem in Canada over the last thirteen years. As the name suggests, cracks associated with TGSCC propagate through grains in the metal, while IGSCC crack tends to take a more irregular path following grain boundaries. TGSCC tends to occur when the pipe steel is exposed to an electrolyte containing high levels of dissolved carbon dioxide while IGSCC tends to form in the presence of bicarbonate solutions.

It is, in fact, more accurate to refer to the transgranular SCC experienced in Canada as near neutral pH SCC (sometimes referred to as low pH SCC). Intergranular SCC normally occurs under conditions of high pH levels in ground water near the pipeline (pH above 9) and is classified as high pH SCC. Transgranular cracking normally occurs when pH levels are between 6 and 8. Under certain conditions, high pH SCC can lead to transgranular cracking, but this is rare [NEB, 1996].

It should be noted that from this point forward that the abbreviation SCC, used alone, will refer to the transgranular rather than the intergranular form of the SCC process.

2.1.1 Transgranular Stress-Corrosion Cracking in Canadian Natural Gas Pipelines

To date, 22 incidents of pipeline failure (leakage or rupture) attributed to SCC have been reported in Canada [CEPA, 1996; Kumar, 1996]. As the pipelines continue to age, the potential for accidents will increase and the need for a method to accurately estimate the extent of SCC damage is becoming increasingly important.

Transgranular SCC is the result of the deterioration of pipe steel through grains due to a combination of cyclic stress loading and a corrosive medium. It relies upon the presence of the following five conditions [Transportation Safety Board, 1994]:

1) A combination of stresses due to internal pressure and residual tensile stress, exceeding the threshold stress level (below which SCC cannot occur), which may be present as a result of manufacturing and installation procedures.

2) Contact between the metal and a corrosive medium.

3) Electrochemical potential within the TGSCC range.

4) Elevated temperature of the product flowing in the pipeline.

5) Fluctuations in the internal pressure.

SCC colonies can consist of hundreds of shallow cracks on the outer pipe wall, generally axially oriented [CEPA, 1996; Transportation Safety Board, 1994], with a tendency to overlap [CEPA, 1996]. Over time, the small cracks may coalesce, forming long shallow cracks that may grow until the pipeline integrity is compromised [Parkins et. al., 1994]. According to the industry accepted definition, a "significant" SCC colony must concurrently satisfy a length and depth criteria [NEB, 1996].

Depth Criteria:	Greater than 10% of the wall thickness.
Length Criteria:	Longer than 75% of a 50% through wall crack that would cause failure at 100% of the specified minimum yield strength (SMYS) of the pipe steel.

SCC will initiate on the outer surface of a pipeline in an area of coating disbondment, where groundwater has been able to accumulate to act as the corrosive medium and cathodic protection (CP) is inadequate due to poor transmission properties of the local soil and/or the coating. Pipelines coated with polyethylene (PE) tape appear particularly susceptible because the tape is especially prone to disbondment. Sixteen of the reported failures have occurred on pipelines coated with polyethylene tape [CEPA, 1996; Kumar, 1996]. Of the remaining failures, 4 have occurred on asphalt/coal tar coated sections and one on a pipeline section coated with glass insulation [CEPA, 1996].

PE tape does not bond well to dented surfaces or over welded seams, providing groundwater collection sites next to the external pipe wall. In addition, the tape is an insulator and acts as a barrier to CP currents [Beavers and Thompson, 1995]. This is not a problem when well bonded to the pipe steel, but permits the formation of micro-environments under disbondments that can lead to SCC attack. Newer coatings, such as fusion bonded epoxy (FBE) and extruded polyethylene, are commonly used today because of their superior bonding properties [NEB, 1996].

Upon finding a significant colony pipeline operators are required to repair either by cutting out the section containing the defect or installing a welded repair sleeve. Significant colonies must be reported to the National Energy Board of Canada [NEB]. Insignificant colonies can be removed by grinding and the pipeline can be returned to full service.

It is estimated that 4% of the over 500,000 km of transmission pipelines in Canada are susceptible to SCC damage [NEB, 1996]. As recent events have shown, the validity of this estimate is yet to be proven. In 1995, Trans Canada Pipelines Ltd. (TCPL) experienced two pipeline failures attributed to SCC in areas previously considered SCC free.

There are several features common to both transgranular and intergranular SCC [Leis, 1995]. The size, shape and inter-crack spacing can vary greatly. Length to depth ratios can range from as low as 1.5 to as high as 20. Inter-crack spacing (Fig. 2.1) appears to be one of the significant features in determining the significance of a colony [Leis, 1995]. Generally, deep cracks will only form when the circumferential spacing within a colony is a large fraction or multiple of the pipe wall thickness. Otherwise, stress shielding occurs (see Fig. 2.1), resulting in the arrest of crack propagation. Thus, clusters of closely spaced cracks tend to contain only shallow defects and become dormant before any significant breach of structural integrity. Significant cracks tend to occur around the edges of clusters where the stress shielding effects due to neighboring cracks are not as significant [NEB, 1996].

2.1.2 Current Inspection and Detection Procedures

There are major initiatives underway, backed by pipeline operators and government agencies, to develop new, effective SCC detection procedures. At present, the only widely accepted means of identifying SCC are through hydrostatic retesting (hydrotesting) and magnetic particle inspection (MPI) [Transportation Safety Board, 1994]. In Line Inspection tools are showing more and more promise, but they are not 100% reliable and still require field verification.

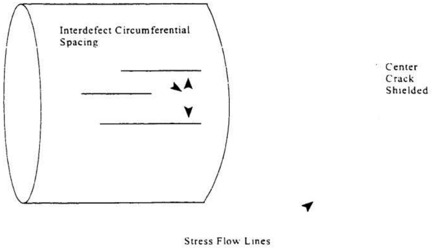


FIGURE 2.1: Schematics of SCC crack orientation and stress shielding (defects shown enlarged)

Hydrotesting is a destructive testing process, whereby, an out of service pipeline section is subjected to a short-term pressurization up to about 100% of SMYS. Critical crack locations are identified by a leak or rupture in the section. Based upon the test pressure and a fracture mechanics assessment of the pipe steel, a reinspection frequency is determined to prevent failure before the largest remaining defect in the line can grow to a critical size under normal operating conditions. While proving the short term integrity of a section, hydrotesting can lead to increased growth of sub-critical SCC cracks, which in effect, may reduce the lifetime of the structure [Zheng et. al., 1998].

Many companies and government agencies are working towards the elimination of hydrostatic retesting, however, there are presently very few options available. MPI has only proven effective in confirming SCC sites identified by in-line inspection (ILI) or predicted by SCC location models compiled from experience. It provides no information regarding the depths of the cracking. In addition, the technique requires coating removal and surface cleaning.

ILI tools using advanced ultrasonic testing (UT) technology (intelligent pigs) have had some success in identifying SCC, but are still prone to false calls from non-metallic inclusions and tend to have difficulty identifying small clusters. Sizing estimates from ILI tools are still very unreliable and require manual verification.

Since 1986, British Gas Limited (BG) has been developing an in-line inspection device based on ultrasonic NDT techniques [TCPL Proceedings, 1995] to identify cracking on in-service pipeline systems. The Elastic Wave Inspection Vehicle (EWIV) is a pigging device equipped with an ultrasonic probe configuration, which does not require a liquid coupling medium as the probes are contained within specially designed wheels. Designers anticipate that it will be capable of detecting and sizing crack colonies, however, they acknowledge that much work remains to be done before the equipment obtains an acceptable level of reliability. The latest EWIV prototype has been tested, with promising results, on some sections of the TCPL pipelines [Borrowman, 1996]. The Mark II model was designed to operate in pipes having 30", 32", 34" and 36" diameters and can negotiate bends of about three times the pipe diameter (3D), with the exception of 30" diameter pipe, which requires a 5D bend. Its data storage capacity is limited to 48 km and it has an optimal travel speed of 2 m/s, which means that testing can only be performed during downtime of natural gas lines. It is estimated that by the end of 1996, 3450 km of the TCPL lines will be equipped with proper launching and receiving equipment to allow for pigging operations. The total cost of refitting all of the TCPL lines with launchers and receivers, as well as replacing plug valves on older lines with the required full opening ball valves, is $221,130,000 [CEPA, 1996]. The EWIV has exhibited difficulties in distinguishing between ultrasonic signals from crack colonies and nonmetallic inclusions in pipe steels, which tend to be numerous, especially in the older vintage lines of the 1960's and 1970's [CEPA, 1996]. Refinements are being implemented in attempts to eliminate such false calls. The tremendous amount of data associated with each test run necessitates a time consuming analytical requirement taking months to complete.

The next generation EWIV, the Mark III, is expected to overcome some of the shortfalls of the Mark II [CEPA, 1996]. It should be capable of 60 km test runs in pipe diameters from 20" through to 42" and negotiate 1.5D bends. The Mark III will also be designed with gas-by-pass capabilities so that testing in natural gas lines may be carried out on in-service pipe.

In addition to the EWIV, other possible in-line tools, including the Pipetronix Ultrascan CD crack detection tool and C.W. Pope's Electro-Magnetic Acoustic Transducer are in use.

The use of the manual UT, time of flight diffraction (TOFD) and alternating current potential drop (ACPD) have also been investigated. UT and TOFD have only been successful when defects were quite isolated [Fingerhut, 1996, personal reference]. For instance, using TOFD on a 15 mm thick plate, there must be approximately 60 mm either side of the defect free of other defects. ACPD has been shown to significantly under predict defect depth when defects are closely spaced [Hodgkinson, 1997, personal reference].

Predictive soils models have been developed using a set of parameters (coating type, soil drainage, tomography, soil type, etc.) to identify areas having a high probability of SCC damage, utilizing information from past experiences [CEPA, 1996]. These models have shown limited success rates.

A technique which may prove promising in Western Canada is the Direct Current Voltage Gradient (DCVG) technique [CEPA, 1996]. Current can be applied to the pipeline using an approach similar to a cathodic protection system to provide information regarding possible locations of coating disbondment. Experimentation with this method has been carried out in central Canada, but the electrical shielding effects of the rock formations have provided only limited success. This technique cannot provide information with respect to SCC damage, but can assist in narrowing the search procedures for SCC, through the location of coating disbondments.

2.2 ALTERNATING CURRENT FIELD MEASUREMENT

The Alternating Current Field Measurement technique, is a thin-skin, non-contacting [Lugg et. al., 1988], electromagnetic technique [Lewis, 1991] used to detect and size surface cracks in electrically conducting, metallic components. It was first commercially introduced by Technical Software Consultants Ltd of the UK in 1991. ACFM has typically been used to characterize weld fatigue cracking in carbon steel structures, but may be adapted to other applications [Raine and Monahan, 1996] and to materials such as aluminum and stainless steel.

To gain an appreciation of the underlying principles of ACFM it may be useful to first consider the principle behind its predecessor, ACPD. ACPD is a direct resistence measurement technique used to size cracks in a known location and orientation. It has found use mainly for crack growth monitoring, particularly for laboratory experimentation. Electrodes are attached to the specimen and an AC current is passed between them perpendicular to the length of the crack. A probe is then placed to one side of the crack, oriented in the same direction as the electric current flow, and a reference voltage measurement is taken between the two pins on the probe (i.e. current passing between the pins multiplied by the resistance of the material). A second voltage measurement is taken with the pins of the probe straddling the crack and a decrease in the voltage is observed. Since the current flowing between the electrodes remains fairly constant and the resistivity of the material should not dramatically change over such a small range, the drop in voltage is a result of a change in resistance due to the increased path length between the probe pins as the current is forced to flow underneath the crack. Comparison of the two voltage readings provides a comparison of the path lengths traveled during both readings, thus indicating the depth of the crack.

Instead of making direct contact with the surface of the metal, ACFM theory takes advantage of the magnetic fields generated by AC currents passing through coils, to induce a current in the surface of the test specimen. ACFM probes are available in a variety of configurations, but they all contain the same primary components illustrated in Fig. 2.2. When the induction coil is excited with and AC current, the resulting intermittent magnetic field surrounding the coil will impress an AC current field in the surface of the test specimen. This effect is indicated in Fig. 2.3 by the arrows perpendicular to the crack length at the edges of the specimen. In the absence of a crack, the induced field is uniform over the specimen's surface producing a consistent secondary magnetic field effect. In the presence of a crack, the AC field is forced to flow under and around the defect, therefore, at the location of the defect the residual secondary magnetic field strength is affected. In a standard pencil probe (Fig. 2.2), two orthogonally wrapped pick-up coils are used to measure the strength of the secondary field components in two directions, designated B_X and B_Z, which are illustrated in Fig. 2.3. As the secondary magnetic field cuts the pick coils, currents are induced and measured by the ACFM hardware. The corresponding B_X and B_Z field fluctuations for a semi-elliptical surface breaking crack are illustrated in Fig. 2.4. These field fluctuations are predictable and reproducible based upon crack length and depth so that the crack dimensions can be determined

The drop in the B_x signal level in Fig. 2.3 is attributed to the reduction in field strength as the current is forced to flow around the crack. The trough and peak in the B_Z signal is the result of the current flow around the ends of the defect and provides information with regards to the crack length. For an isolated semi-elliptical crack, sizing is based upon the following four parameters: the background B_X level, the minimum B_X level, the length of the defect as indicated by the B_Z signal, and the location of the pick-up coils with respect to the crack [Lewis, 1991]. For sizing purposes, ACFM software typically compares the parameters to data for known defects, stored in look-up tables.

The depth of penetration (skin thickness) of the AC field depends upon the frequency of the induction current and the magnetic permeability of the material. For carbon steel (as used in transmission pipelines) the skin depth is approximately 0.2 mm at a frequency of 5 kHz, whereas in austenitic stainless steels it can approach 30 mm at 200 Hz. Therefore, in carbon steel ACFM will only detect and size surface breaking cracks, but in austenitic stainless it can detect subsurface and back wall defects in plate up to 30 mm thick.

Pencil probes are able to detect field disturbances within 5 to 10 mm of either side of the probe. It is possible to construct a linear array probe, which is comprised of an induction coil and a number of B_x and B_z pick-up coils to increase the coverage in a single scan pass. Two dimensional array probes can be constructed to cover areas 75 mm by 150 mm and contain 96 sets of sensor coils. Flexible array probes allow operators to easily scan curved, or non-uniform surfaces.

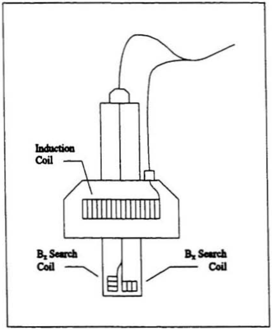


FIGURE 2.2: Schematic of pencil probe

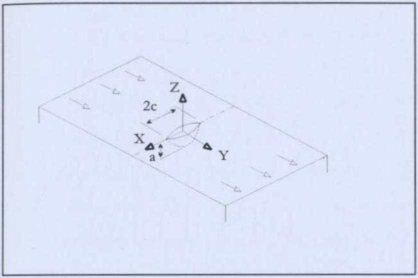


FIGURE 2.3: ACFM field directions

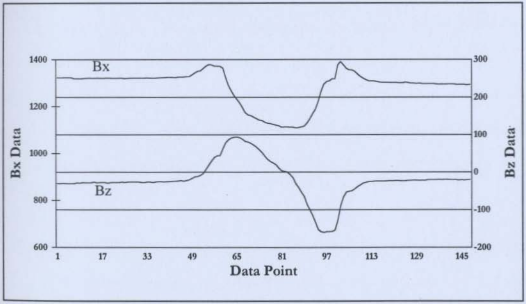


FIGURE 2.4: Sample B_x and B_z signals resulting from a semi-elliptical surface crack

ACFM could not be used to replace ILI technology, as unlike UT it cannot detect cracking from within the pipeline. Instead, it is seen a potential complimentary technology, which could be used by field personnel to verify ILI results. It is perceived to have key advantages over UT and MPI techniques, currently used in this capacity. ACFM is a non-contact technique, which is fairly insensitive to probe lift-off [Rain and Monahan, 1996] and does not require extensive surface preparation, as it is unaffected by most non-metallic coatings and oxide layers [Raine et. al., 1992]. This is in thanks to the use of an induced AC current field rather than a directly impressed current. Eliminating the need for coating removal and surface cleaning of excavated pipeline sections offers a significant time and cost advantage over current practices. ACFM provides less ambiguous data than UT, with regards to crack colony detection and characterization as UT reflections from multiple cracks can be difficult to identify. ACFM provides information with respect to crack depth that MPI cannot. Furthermore, calibration is not required for ACFM, unlike ultrasonic inspection equipment. UT equipment must be calibrated for slight variations in sound transmission properties, however, since ACFM simply compares magnetic field strength between crack and uncracked areas on the specimen, the equipment does not need to be calibrated prior to each use.

For isolated weld toe fatigue cracking, ACFM is generally considered accurate for sizing when the defect is greater than 1 mm deep and longer than 20 mm, although it can detect and size smaller defects remote from a weld. These defects may be large compared to many of the cracks resulting from TGSCC, however, ACFM was considered for this application since the cumulative affect of the signals from close proximity cracks was anticipated to improve the ability to detect the defects.

2.3 Literature Review of NDT Techniques and Algorithms Used in the Automated Characterization of Defects

ACFM is a relatively new NDT method for which little effort has been expended on developing techniques to automate the detection and sizing processes. However, the general trend in the inspection services industry is shifting toward maximizing the efficiency and reliability of NDT techniques by removing the human judgement element [Burch and Bealing, 1986]. Accurate means of estimating flaw parameters is essential in fracture mechanics predictions for life expectancy [Kline and Egle, 1986] and the use of the computers and pattern recognition algorithms to achieve better results appears to be gaining popularity in industry [Chen, 1989].

The field of pattern recognition (PR) is becoming increasingly important to scientists and engineers. The use of automated procedures for repetitive tasks related to data processing and decision making is an attractive alternative to having individuals fumbling through large quantities of information. The speed with which computers can perform such assignments has prompted interest in developing automated algorithms. A great deal of work has gone into maximizing the efficiency of PR techniques for a variety of applications in both engineering and the sciences [Alder et. al., 1992; Chen, 1985;

Ramasubramanian, and Paliwal, 1992; Trahanias and Skordalakis, 1990].

PR techniques attempt to break data into the common features or 'patterns' which humans would instinctively use to compare objects (i.e. colour, texture, size, shape, etc.) to determine whether or not they belong to the same group or classification [Fukunaga, 1990]. Humans perform such tasks with little thought, but programming machines to understand simple comparisons can often be difficult. There are three primary classifications of PR algorithms - statistical pattern recognition, syntactical pattern recognition, and artificial neural networks [Shalkoff, 1992]. Statistical PR uses vector and matrix statistical techniques to determine the probability that a particular object belongs to a particular group. The probability of error of misclassifying an object and the resulting cost of the misclassification must be determined to evaluate the suitability and reliability of the technique. Syntactical or structural PR methods, as the name suggests, divide objects into structural components for comparison with objects of known structures. Artificial neural networks try to emulate the function of the human brain through the development of network structures based on components or 'neurons', which are essentially discriminant functions. Complex algorithms may be developed using various combinations of neurons.

Most of the available literature relating to the use of pattern recognition in the NDE field concerns the use of either ultrasonic or eddy current data. Both processes provide easily digitized, numerical data that is ideal for statistical [Chen, 1989] or neural network [Chen, 1989; Dodd and Allen, Jr., 1992] techniques. The EWIV system employs pattern recognition techniques in the analysis of the UT data from pipeline inspection [Taffe, 1986].

Ultrasonic and eddy current data can be broken down into recognizable elements in both the time and frequency domains [Brown et. al., 1981], allowing for the use of both in signal recognition procedures [Clark et. al., 1986]. Digital signal processing techniques are often important in feature selection from digitized UT [Kline and Egle, 1986; Chen, 1987; Doctor et. al., 1986; Poulter, 1986; Rose and Meyer, 1974; Singh and Udpa, 1986] and eddy current data [Singh and Udpa, 1986]. Techniques such as Fourier transformations [Rose, 1984; Chaloner and Bond, 1986; Doctor et. al., 1981], deconvolution [Chen, 1989; Chen, 1987; Chaloner and Bond, 1986], autocorrelation [Chen, 1989; Singh and Udpa, 1986; Doctor et al., 1981], cross-correlation of signals [Chen, 1989; Chen, 1987; Singh and Udpa, 1986; Doctor et. al., 1981], spectral analysis [Chen, 1989; Chen, 1987; Smith, 1987] and imaging [Chen, 1989] have been used to establish effective descriptors of defect signals.

Deconvolution, spectral analysis and imaging techniques can be used to break down signals into a set of base components (i.e. frequency responses). Knowledge of the make-up of defect signals can provide clues to evaluate more complicated signals. Cross and auto-correlation functions provide clues to signal decoding through the determination of correlation patterns within the signal.

More traditional statistical descriptors of data, such as kurtosis [Burch and Bealing, 1986], have also been used as characterization features. The simplest signal descriptors are time domain features, such as signal amplitude and signal duration.

Researchers have attempted to automate the detection of a variety of defects using EC and UT data. The defects include weld porosity, corrosion pitting, SCC, and fatigue cracking. In many cases, when there was an inadequate supply of natural defects for test series, machined defects where used to simulate flaws. Electro discharge machining (EDM) is a common technique used to create cracks and pits [Doctor et. al., 1981]. In others cases, slitting saws [Olley, 1985] have been found to be useful in machining simulated cracks.

REFERENCES:

National Energy Board (NEB) (1996), "Report of the Inquiry: Stress Corrosion Cracking on Canadian Oil and Gas Pipelines", Public Inquiry Concerning Stress Corrosion Cracking on Oil and Gas Pipelines, MH-2-95, Cat. No. NE23-58/1996E, ISBN 0-662-25246-2, November.

Canadian Energy and Pipeline Association (CEPA) (1996), Submission to the National Energy Board Public Inquiry Concerning Stress Corrosion Cracking (SCC) on Canadian Oil and Gas Pipelines, Proceedings MH-2-95, Volume I, February.

Kumar, Ajay (1996), Memorandum to the Stress Corrosion Cracking Subject File, http://www.sigmarsisk.com/cracking.htm, May 2.

Transportation Safety Board of Canada (1994), "Trans Canada Pipelines Limited Natural Gas Pipeline Ruptures", Commodity Pipeline Occurrence Report No. P92 T0005, P91 H0117.

Beavers, J. A. and N. G. Thompson (1995), "Effects of Coatings on SCC of Pipelines: New Developments", *OMAE 1995*, Vol. 5, Pipeline Technology, ASME, pp. 249-263.

Zheng. W., Tyson. W.R., Revie, R.W., Shen, G., and J.E.M. Braid, "Effects of Hydrostatic Testing on the Growth of Stress-Corrosion Cracks", Proceedings of the International pipeline Conference 1998, Volume I. ASME, pp. 459-472, 1998.

Leis. B. N. (1995). "Characteristic Features of SCC Colonies in Gas Pipelines - Implications for NDI Systems and Related Integrity Analysis", *Pipeline Technology: Proceedings of the 2nd International Pipeline Technology Conference*. Ostend, Belgium. Elsevier. Amsterdam/New York.

Fingerhut. Martin (1996), Manager of Engineering Services. RTD Quality Services Ltd., Edmonton, Alberta, Canada.

Hodgkinson. Dave (1997). NDT Specialist. Facilities Provision. NOVA Gas Transportation Ltd., Calgary. Alberta, Canada.

Parkins, R. N., Blanchard. W. K. Jr., and B. S. Delanty (1994). "Transgranular Stress Corrosion Cracking of High Pressure Pipelines in Contact with Solutions of Near Neutral pH". Corrosion, Vol. 50. No. 5. pp. 354-408. May.

Taffe. P. (1986). "Intelligent Pigs Crack Pipe Corrosion Problems". Processing. pp. 19-20. September.

Borrowman. Kelly (1996). "A Matter of Integrity". Oilweek Magazine. Pipeline Issue. October.

Lugg. M. C., Lewis. A. M., Michael. D. H., and W. D. Dover (1988). " The Non-contacting ACFM technique", Electromagnetic Inspection. *IOP Meeting No. 12*. Institute of Physics. pp. 41-81.

Lewis. A. M. (1991), "The Modelling of Electromagnetic Methods for Nondestructive Testing of Fatigue Cracks", Ph. D. Thesis. Department of Mechanical Engineering. University of London. U.K.

Raine, Alan G. and Craig C. Monahan (1996), "Alternating Current Filed Measurement (ACFM): A New Technique for the NDT of Process Plant and Piping Components". *NDE Engineering Codes and Standards and Materials Characterization*, ASME PVP-Vol. 322, NDE-Vol. 15, pp. 87-93.

Raine. G. A., Dover, W. D., and J. R. Rudlin (1992), "Trials on Coated Nodes", *IOCE 92 Conference Proceedings*, Aberdeen. U.K., October.

Burch, S. F. and N. K. Bealing (1986), "A physical approach to the automated ultrasonic characterization of buried weld defects in ferritic steel", NDT International, Vol.

19, pp. 145-153, June.

Kline, R. A. and D. M. Egle (1986), "Applications of digital methods to ultrasonic materials characterization", NDT International, Vol. 19, No. 3, pp. 341-347, June.

Chen, C. H. (1989), "Tutorial on Signal Processing and Pattern Recognition in Nondestructive Evaluation of Materials", *Non-Destructive Testing (Proceedings of the 12th World Conference)*, Elsevier Science Publishers B. V., Amsterdam.

Alder, M., Lim, S. G., Hadingham, P. and Y. Attikouzel (1992), "Improving neural net convergence", Pattern Recognition Letters 13, No. 5, pp. 331-338, Elsevier Science Publishers B. V., May.

Chen, C. H. (1984/85), "On a Segmentation Algorithm for Seismic Signal Analysis", Geoexploration, Vol. 23, pp. 35-40, Elsevier Science Publishers B. V., Amsterdam.

Ramasubramanian, V. and K. K. Paliwal (1992), "An efficient approximation-elimination algorithm for fast nearest-neighbour search based on a spherical distance coordinate formulation", Pattern Recognition Letters 13, pp. 471-480, Elsevier Science Publishers B. V., July.

Trahanias, Panagiotis and Emmanuel Skordalakis (1990), "Syntactic Pattern Recognition of the ECG", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 7, July.

Fukunaga, Keinosuke (1990), "Introduction to Statistical Pattern Recognition", Academic Press, Inc., Harcourt Brace Jovanovich, Publishers.

Schalkoff, Robert J. (1992), "Pattern Recognition: Statistical, Structural and Neural Approaches", John Wiley & Sons, Inc.

Dodd, C. V. and J. D. Allen, Jr. (1992), "Automated Analysis of Eddy-Current Steam Generator Data", *United States Department of Energy CONF-920463-1*, December.

Brown, C. L., Defibaugh, D. C., Morgan, E. B. and A. N. Mucciardi (1981), "Automatic Detection and Sizing of Steam Generator Tubing Defects by Digital Signal Processing", Eddy-Current Characterization of Materials and Structures, ASTM STP 722, George Birnbaum and George Free, Eds., American Society for Testing and Materials, pp. 484-493.

Clark, G. A., Tilly, D. M. and W. D. Cook (1986), "Ultrasonic signal/image restoration for quantitative NDE", NDT International, Vol. 19, No. 3, pp. 169-176, June.

Chen, C. H. (1987), "Pattern Analysis for the Ultrasonic Nondestructive Evaluation of Materials", International Journal of Pattern Recognition, Vol. 1, No. 2, pp. 251-260.

Doctor, S. R., Hall, T. E. and L. D. Reid (1986), "SAFT - the evolution of signal processing technology for ultrasonic testing", NDT International, Vol. 19, No. 3, pp. 163-167, June.

Poulter, L. N. J. (1986), "Signal processing methods in the ultrasonic inspection of PWR inlet nozzles", NDT International, Vol. 19, No. 3, pp. 141-144, June.

Rose, Joseph L. and Paul A. Meyer (1974), "Signal Processing Concepts for Flaw Characterisation", British Journal of Nondestructive Testing, Vol. 16, pp. 97-106, July.

Singh G. P. and S. Udpa (1986), "The role of digital signal processing in NDT", NDT International, Vol. 19, No. 3, pp. 125-132, June.

Doctor, P. G., Harrington, T. P., Davis, T. J., Morris, C. J. and D. W. Fraley (1981), "Pattern-Recognition Methods for Classifying and Sizing Flaws Using Eddy-Current Data", Eddy-Current Characterization of Materials and Structures, ASTM STP 722, George Birnbaum and George Free, Eds., American Society for Testing and Materials, pp. 484-493.

Rose, J. L. (1984), "Elements of a Feature-based Ultrasonic Inspection System", Materials Evaluation, Vol. 42, February.

Chaloner, C. A. and L. J. Bond (1986), "Ultrasonic signal processing using Born inversion", NDT International, Vol. 19, No. 3, pp. 133-140, June.

Smith, R. L. (1987), "Ultrasonic materials characterization", NDT International, Vol. 20, No. 1, pp. 43-48, February.

Olley, D. A. (1985), "The Use of EDM Slots in Eddy Current Sensitivity Blocks", Non-Destructive Testing - Australia, Vol. 22, No. 6, pp. 133-134, Nov/Dec.

CHAPTER 3: EXPERIMENTAL PROCEDURE

The experimental apparatus used in this test series was assembled to provide data emulating an ACFM array probe as one was not available and the cost of constructing an array probe was beyond the budget of the project. A standard pencil probe was used with an X-Y positioning frame to scan regions of the test samples. Figure 3.1 is a schematic of the experimental apparatus. The positioning frame was originally designed for applications on an ice basin carriage and, therefore, had features not used in this series.

A Model U10 ACFM Crack Microgauge, equipped with an analogue BNC output port, was used in conjunction with a 5 mm pencil probe. The probe was mounted in a nylon plexiglass holder which, in turn, was mounted to the positioning frame. The frame's carriage was able to translate in two directions (designated X and Y, as shown in Fig. 3.1) using two stepper motors. The carriage was also equipped with two stepper motors which could be used to adjust the vertical height of any attached components. For this test series, however, vertical adjustments of the probe were conducted manually. The QFM 1.0 software, typically used for data acquisition from the U10 microgauge, could not be synchronized with the stepper motor controller program, thus, a Microsoft QuickBASIC program was written (Appendix C) to coordinate these tasks. The X and Y motors were controlled using DAEDAL PC21 Indexers, and data was collected using the analogue output on the U10 via a CYDAS 8 A/D card. Knowledge of the probe position during each scan pass was essential in determining defect locations. Probe position in the data stream was determined using the equation:

$$l = \frac{V}{SR} \cdot NP \tag{3.1}$$

Where:

	l	: the location of the probe (mm)
	V	: the speed of the probe (mm/s)
	SR	: the sampling rate of the A/D card (Hz)
	NP	: the number of data points collected

The QuickBASIC software was executed using a 16 MHz, 386 computer.

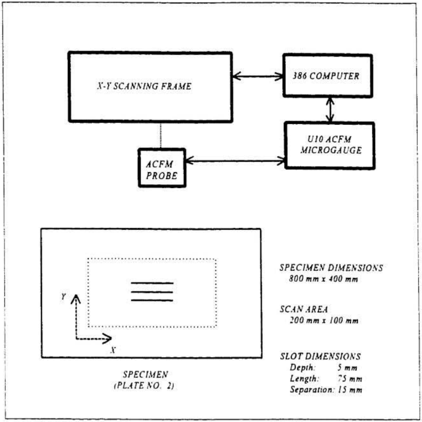


FIGURE 3.1: Schematic of the test apparatus

3.1 Machined Defect Colony Test Series

A series of defect colonies were machined in cold-rolled, mild steel plates to comprise the test sample set. In the preliminary stages of the research, it was decided to keep the defect colonies simple so each colony was limited to three defects. The configurations are illustrated in Fig. 3.2 and Table 3.1. There were 21 colonies used in total, designated colonies 1 through 21. Each of the three defects within the colonies were designated defect A, B and C. Appendix A contains the shop drawings for the 21 colonies.

To investigate the effect of defect depth on the cluster signals, two different depths (2 mm and 5 mm) were used. The majority of the defects were 50 mm long, with two colonies having defects 75 mm long. They were machined using a 40 thou (1.02 mm), 4 inch diameter (101.6 mm) slitting saw, giving the defects a flat bottom profile. The standard sizing tables developed for ACFM technology assumes defects have a semi-elliptical shape common to fatigue cracks. To properly profile semi-elliptical slits, electro-discharge machining would have been required. This option was not locally available, so the flat-bottomed profile was used. A series of slitting saws, ranging from 10 to 40 thou in thickness were tested, but the 40 thou saw produced the most repeatable results. A 1 mm crack opening is large compared to the typically tighter stress corrosion cracks, but this parameter was kept constant throughout testing.

At the outset of the project, only a 5 mm pencil probe was available for the tests. For this reason, it was decided that the colonies should be scaled in accordance with the probe size. Future work will likely require the development of a more sensitive probe for distinguishing closely spaced defects. A scale factor of five was used to increase the defect length and the inter-defect parallel spacing (corresponding to the circumferential spacing discussed in Chapter 2). This scale factor was not applied to the defect depths, as it would have lead to the use of unreasonably thick steel plates. New information was discovered subsequent to the machining of the first plates to suggest that the chosen inter-defect spacing may correspond to that of some natural SCC colonies containing significant defects. The inter-defect spacing was maintained at 15 mm throughout the test series.

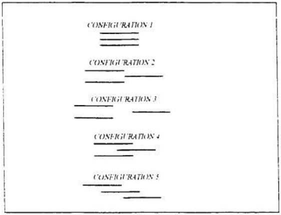


FIGURE 3.2: General machined defect configurations

In addition to the colonies, four isolated defects were machined, corresponding to each of the four defects used in the make up of the colonies; 2 mm deep by 50 mm long, 2 mm deep by 75 mm long, 5 mm deep by 50 mm long and 5 mm deep by 75 mm long. These were used in later investigations into defect characterization, discussed in Chapter 5.

TABLE 3.1: Machined defect depths

PLATE	CONFIG.	Slot A (mm)	Slot B (mm)	Slot C (mm)
1	1	2	2	2
2	1	5	5	5
3	1	2	5	2
4	2	2	2	2
5	2	5	5	5
6	2	2	5	2
7	2	5	2	5
8	3	2	2	2
9	3	5	5	5
10	3	2	5	2
11	3	5	2	5
12	4	2	2	2
13	4	5	5	5
14	4	2	5	2
15	4	5	2	5
16	5	2	2	2
17	5	5	5	5
18	5	2	5	2
19	5	5	2	5
20	5	5	2	2
21	5	2	5	5

Notes:

a) With reference to Fig. 3.2
b) Slot A is the bottom slot
c) Slot B is the middle slot
d) Slot C is the top slot
e) All slots 50 mm long except in plates 1 and 2, which were 75 mm long

All of the configurations shown in Table 3.1 were not machined simultaneously. The manufacturing process took place over several months. Plates 1, 4 , 8, 12 and 16 were the first to be machined. This provided an opportunity to analyze preliminary data before the scope of the project was finalized.

During scanning, the probe was translated parallel to the defect colonies, in a region 200 mm long in the X-direction and 100 mm wide in the Y-direction. Data was collect only while the probe moved in the positive X-direction. At the end of each pass, the data collection ceased and the probe was repositioned to begin the next scan pass. With a 5 mm probe, it was decided that a 2 mm spacing between each pass (in the Y-direction) was sufficient to adequately characterize the field perturbations. A 1 mm spacing was tested, but appeared to offer no improvement on the clarity of the signal. It merely doubled the required time to scan each plate and provided data files of twice the size. The minimum sampling rate of the A/D card was limited to 100 Hz and the scan speed used was 20 mm/s, to eliminate vibrations in the X-Y frame. Thus, 5 samples were taken each second, corresponding to a spacing of 0.2 mm in the X-direction. This was more data than required, but due to the limitations of the A/D card and the accuracy of the positioning frame, it was considered acceptable. The maximum acceleration possible (200 mm/s^2) was chosen for the stepper motor starts and stops. Thus, the lag time between stepper motor motion and data acquisition was minimized. The 20 mm/s scan speed and 200 mm/s^2 acceleration resulted in negligible vibrations in the test assembly, so that the probe positioning accuracy was not affected.

3.2 REAL SCC COLONY TESTS

An opportunity arose, during the machined defect test series, to test plate sections from natural gas pipelines containing natural SCC damage allowing for a preliminary evaluation of commercially available ACFM probe technology in detecting real SCC damage. In addition to the pencil probe used on the machined colonies, two, 2 mm ACFM microprobes were available at different times during the testing phase. Their performance was compared to that of the 5 mm probe.

Budget and time restrictions did not provide the opportunity to design and construct a new probe positioning frame modified for the curved pipe walls so the apparatus used for the machined defect colonies was adapted to the task. The X-Y frame did not permit rotation of the probe, thus, the probe could not be positioned normal to the surface of the plate at all times during a scan. It was discovered during the machined defect series that probe lift-off of a couple of millimeters had little affect on the ACFM B_z signal. The lift-off associated with the plate curvature was not anticipated to be a concern, as the curvature of the pipe wall sections was fairly gradual (approximately a 1 m pipe diameter). The primary interest was in detection (focusing on the B_Z signal) and not in sizing, as sizing verification would require a great deal of destructive sectioning of the plates. Sizing of selected defects was carried out, but in those cases, the sizing was performed manually using the QFM software.

Modifications were required to the QuickBASIC program. Because of the curvature of the pipe sections, vertical adjustment of the probe was also required. The frame was not equipped with proximity sensors, so these adjustments had to be made manually. Because of the large areas that had to be covered, the scan speed of the probe was increased to 50 mm/s. These factors did not appear to significantly affect the stability of the frame or the accuracy of probe placement.

MPI was used to determine the locations of the defects in the pipe walls. The regions affected by SCC were marked on the plates by the supplier. A layer of contrast paint was applied over these regions, and upon drying, a permanent marker was used to overlay a grid. Originally, 10 mm square blocks were used, but it was later decided that 20 mm square blocks were sufficient. Individual cracks were located and sketched for later reference.

The sections were first scanned using the 5 mm pencil probe. The probe mount that had been used with the machined plates was designed to accommodate this probe. A loose fit within the mount provided the probe with the ability to make slight adjustments over surface irregularities.

The methods for mounting the microprobes were more complicated, but yielded acceptable results. The first microprobe was attached to the underside of the 5 mm probe mount using a sponge backing between the probe and the plexiglass, allowing the probe freedom of vertical motion to ride up over slight surface irregularities and then spring back to maintain contact with the surface. The second 2 mm probe had a different construction and was mounted to the front of the plexiglass mount using rubber bands to provide some flexibility when dealing with surface irregularities. Unlike the 5 mm probe, which had a composite material covering the probe coils, the microprobes had a stainless steel wear plate. To reduce surface friction and wear between the microprobe and the plate, water soluble, UT gorp was used as a lubricant. All probes were covered with electric tape to reduce wear of the tips. This procedure is recommended by the manufacturer and does not affect the ACFM signals.

Three of the more isolated cracks where manually scanned and sized using the ACFM QFM software. The results of these sizing tests are discussed in Chapter 5.

3.3 PRELIMINARY DATA HANDLING

The QuickBASIC programs used during the test series provided data in a single column text file containing introductory information identifying the file. An example of the first 13 entries in the output files follows:

```
"ACFM Data File"
"pl_1_bx"
"20-09-96"
"9:11am"
"1"

-100
```

3384
3383
3383
3384
3384
3384

The first line identifies the file as a raw data file, with the second line giving the file name. This is followed by the date and time the file was created in lines 3 and 4 respectivly. The fifth line indicates the colony number.

To manipulate the data into a 51 column by 1000 row matrix representing the scan area, a Matlab M-file routine was written for the B_x and B_z data files. The files are provided in Appendix C. The introductory text was removed using a UNIX text editor. The BASIC code provided the flag number of -100 to indicate the beginning of each scan pass in the data stream. This value was chosen, as it is outside the possible range of the +/- 10 volt output of the A/D card. The M-files saved the data in matrix form in text files for future use and output postscript files containing 3- dimensional images of the ACFM signals. The plots will be discussed further in Chapter 4. The plotting of the files used every 5th row of data, since use of all rows was too dense for the Matlab plotting routines and was not necessary to identify the signal features.

CHAPTER 4: DETECTION OF DEFECTS

4.1 MACHINED DEFECTS

The resulting B_x and B_z signals from the 21 machined defect colonies are shown in Appendix A. Four views of the B_x and B_z data were plotted, giving a total of eight plots per colony. The first of the four views is a three-dimensional isometric, the second is a two-dimensional colour contour plot, the third is a view of the three-dimensional image along the length of the scan area (X-direction) and the fourth is a view along the width of the scan area (Y-direction). Examples of the plots follow.

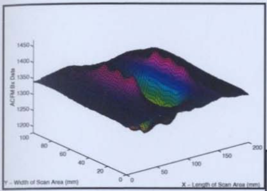


FIGURE 4.1:
B_x isometric for machined colony 2

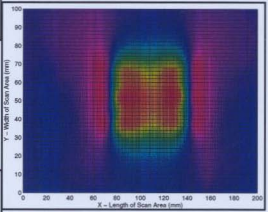


FIGURE 4.2:
B_x contour plot for machined colony 2

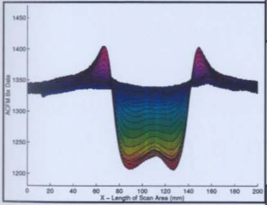


FIGURE 4.3:
B_x view along length of scan area for machined colony 2

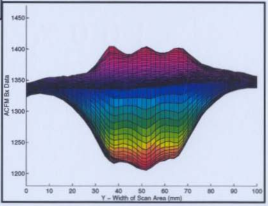


FIGURE 4.4:
B_x view along width of scan area for machined colony 2

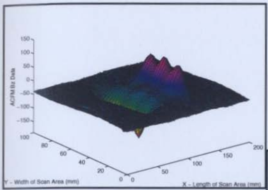


FIGURE 4.5:
B_z isometric for machined colony 2

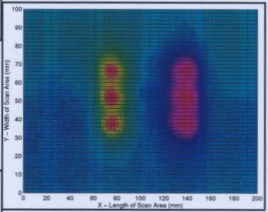


FIGURE 4.6:
B_z contour plot for machined colony 2

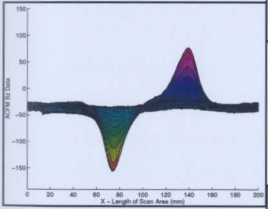


FIGURE 4.7:
B_z view along length of scan area for machined colony 2

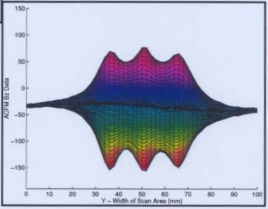


FIGURE 4.8:
B_z view along width of scan area for machined colony 2

Visual inspection of the plots, revealed two means of identifying the individual defects within the colonies. For most configurations, the peak points in the B_X signal where the magnetic field experiences a characteristic rise just before, and just after, the ends of a crack could be used. However, in some cases when defects overlapped, the 2 mm defects could not be clearly distinguished due to the more dominant signal created by a 5 mm deep defect. Using the peaks and troughs in the B_Z signal caused by the curvature of the AC field around the ends of the defect is traditionally used with ACFM technology. This proved to be a more useful means of identifying the defects, as future ACFM SCC sizing algorithms will likely require knowledge of the distance between the B_Z peak and trough as an estimate of defect length. The locations of the peaks and troughs for each defect were visible in all B_Z signals..

A Matlab routine was developed to automatically identify the peaks and troughs in the B_Z data, using the image analysis toolbox functions. This routine provided the location and length estimate of the defects. The code for the detection algorithm is included in the M-files lin_sup.m, def_size.m, and scc_det.m, given in Appendix C.

The data, in the form of a 51 column by 1000 row matrix, was normalized between 0 and 1, to apply the Matlab image processing toolbox functions. To determine the appropriate conversion formula, the maximum and minimum B_Z values in each of the 21 scans were identified. The range of values, in data acquisition card units, was between 1900 and 2040 and the resulting formula was:

$$Normalized\ B_z = \frac{(B_z - 1880)}{260} \tag{4.1}$$

The 0 to 1 range corresponds to data between 1880 and 2060, which is slightly below the minimum trough value and slightly above the maximum peak value.

Upon conversion, the 'imresize' function in Matlab was used to convert the matrix to 50 rows by 100 columns (an even number of rows and columns was required), giving it an aspect ratio similar to the actual scan area, with a matrix size dense enough to be accurate, while small enough to allow for fast computing time. The new matrix was plotted to the screen in 256 grayscale, providing a visual representation so the user could verify the results of the normalization routine.

B_Z peak identification was the next phase in the algorithm and required a three step process. Prior to writing this routine, the difference in intensity between a peak pixel and the eight surrounding pixels was determined from a 256 shade grayscale image of a scan using Corel PhotoPaint. The eight positions were designated north, south, east, west, northeast, northwest, southeast and southwest. This intensity difference was the threshold value used to identify local 'high' areas in the data. Eight mask matrices, given below, were convolved with the B_Z data from each colony to determine if the threshold criteria was valid for each of the eight directions surrounding a pixel.

```
east =          0  0  0  0  0 -1 -1
                0  0  0  6  0 -1 -1
                0  0  0  0  0 -1 -1

west =         -1 -1  0  0  0  0  0
               -1 -1  0  6  0  0  0
               -1 -1  0  0  0  0  0
```

north = east'
south = west'

```
nwest =        -1 -1 -1  0  0  0  0
               -1 -1  0  0  0  0  0
               -1  0  0  0  0  0  0
                0  0  0  6  0  0  0
                0  0  0  0  0  0  0
                0  0  0  0  0  0  0
                0  0  0  0  0  0  0

swest =         0  0  0  0  0  0  0
                0  0  0  0  0  0  0
                0  0  0  0  0  0  0
                0  0  0  6  0  0  0
               -1  0  0  0  0  0  0
               -1 -1  0  0  0  0  0
               -1 -1 -1  0  0  0  0
```

neast = swest'
seast = nwest'

The second step in peak identification scheme was a check to determine the number of directions for which the threshold intensity criteria was valid. The criteria had to be satisfied in all eight directions for a point to be considered a possible peak. The final check required that a peak point be greater than 20 units above the background level preventing any small noise spikes from being missclassified as peaks.

The process of identifying troughs in the signal was identical, except the data image was inverted prior to checking threshold levels. Figure 4.9 is an example of the defect location plot using data from colony 13.

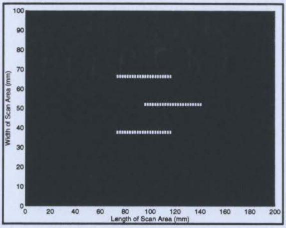


FIGURE 4.9: Defect detection algorithm results for machined colony 13

The results of the peak identification algorithm were quite impressive. From the 21 colonies, 62 of the 63 defects were clearly identified using a threshold level of 0.075. The unidentified defect was slot A in colony 3 (see Fig. 3.2 and Table 3.1) . A B_Z plot for this colony is shown in Fig. 4.10. The 5 mm deep central defect, slot B, masked part of the peak in the 2 mm deep slot, A, so that it did not satisfy the threshold criteria for all eight directions. It should be noted that the trough at the other end of slot A, and the endpoints of slot C, also 2 mm deep, were identified. Slot A was correctly identified when the number of locations for which the threshold level had to be satisfied was reduced to 6 and the minimum possible peak height was increased to 25 units from 20. These same values were not adopted for all of the colonies, because they lead to an erroneous length measurement in slot B of colony 17.

Future refinements to the peak detection algorithm should include a decision making process to enhance the peak detection capabilities under such circumstances.

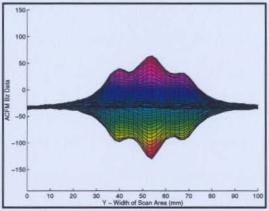


FIGURE 4.10: View of the B_z signal of machined colony 3 along the scan area width

4.2 DETECTION OF DEFECTS IN SCC COLONIES

The natural SCC colonies scanned in this test series provided more of a challenge than the machined defect colonies, primarily because the defects were smaller in length and often more closely spaced. The resolution of the 5 mm and 2 mm probes was adequate for detecting the defect clusters, but neither probe was capable of distinguishing every defect. The MPI results did not provide any information on the depth of the defect indications, so the number and location of significant cracks was not known. Some of the cracks may have been fractions of a millimeter deep and, therefore, insignificant. Detailed sectioning of the pipe material would have been necessary to determine which cracks were, in fact, significant and the available funding did not permit this.

Scans of selected regions within the SCC affected areas of the pipe wall sections are shown in Appendix B. The Matlab routines used to plot the machined defect colony data were modified to account for differences in data file size and the increased noise levels in the SCC colony data (presented in Appendix C). To reduce noise levels in the SCC plots, a time series averaging technique (also a part of the QFM software's signal processing) was utilized. Each data point was averaged with its neighbors in all surrounding directions to smooth signal noise. This is a fairly standard technique used in signal noise reduction and was deemed necessary as the noise in the microprobe scans was fairly significant. The 5 mm probe appears to have provided a superior signal to noise ratio when compared to the 2 mm probe. However, the data from the microprobe was enhanced fairly reliably using the data averaging technique. Figure 4.11 was sketched from the results of MPI examination of one of the pipe walls and clearly shows evidence of SCC damage. It must be stressed that no information were available concerning the depths of these defects. The results of a pencil probe scan is provided in Fig. 4.12. The brightest yellow and hot red regions are dips in the B_z signal at one end of the defect and the pink areas are the rises in the signal at the opposite end. The microprobe scan did not cover the entire area shown in Fig. 4.11, but was able to identify cracking, as indicated in Fig. 4.13.

The work described here did not utilize the B_x data. Effort was concentrated on the B_z data, to determine the ability of the ACFM probes to detect, rather than size, the SCC cracks. Sizing results for a few of the more isolated defects are described in Chapter 5.

The weld seams did not influence the B_z signals levels and, therefore, did not affect defect detection. In the B_x signals, the weld seams did affect the background levels immediately adjacent to the weld. However, provided the background levels are properly identified for defects located near weld seams, this interference should not affect sizing results, as ACFM's primary application to date has been the detection and sizing of weld toe defects.

The peak and trough detection algorithm described in Section 4.1 was modified (see Appendix C) and tested to evaluate its performance when used on actual crack colonies. The threshold parameter was changed from 0.075 to 0.15. In addition, the formula used to normalize the data was modified according to the probe used. Variations in the probe construction and gain settings had to be considered when normalizing the data. The normalization formula used for the microprobe was:

$$Normalized\ B_Z = \frac{(B_Z - 2580)}{140} \tag{4.2}$$

The normalization formula used for the pencil probe was:

$$Normalized\ B_Z = \frac{(B_Z - 1930)}{120} \tag{4.3}$$

Again the results were promising. The peak detection routine was applied to the scans depicted in Figures 4.12 and 4.13 and the results are illustrated in Figures 4.14 and 4.15. All of the cracks indicated by MPI could not be identified individually, but the majority of the clusters were detected by both probes. In practice, this technique could provide inspectors with the knowledge of where they should focus their attention and could also provide one of the key features needed to identify significant SCC colonies, namely the overall length of a cluster.

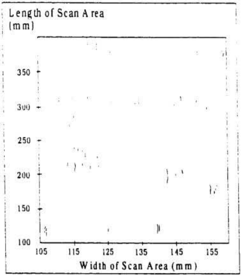


FIGURE 4.11: MPI results of a section of SCC affected pipe wall

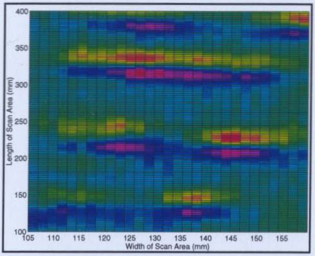


FIGURE 4.12: Pencil probe scan of pipe wall section from Fig. 4.11 (B_z Signal)

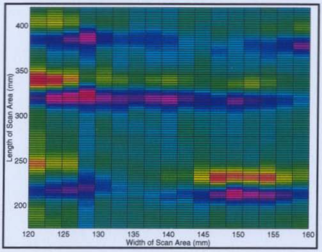


FIGURE 4.13: Microprobe scan of pipe wall section extracted from the region shown in Fig. 4.11 (B_z Signal)

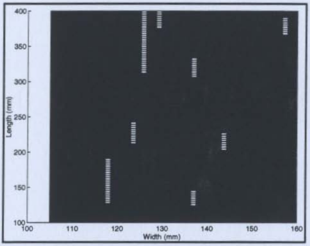


FIGURE 4.14: Peak detection algorithm results from the pencil probe scan of the region shown in Fig. 4.12

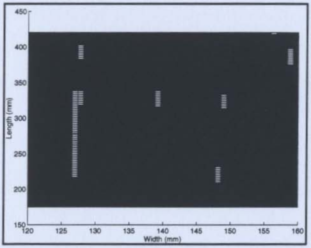


FIGURE 4.15: Peak detection algorithm results from the microprobe scan of the region shown in Fig. 4.13

Figure 4.16 is an illustration of the results of the peak detection algorithm on 5 mm pencil probe B_z data from the same pipe used to create Figures 4.11 through 4.15, encompassing a larger region, measuring 500 mm long and 180 mm in width. It is provided to assess the sensitivity of the peak detection algorithm on a larger scale. The threshold parameters were identical to those used to create Fig. 4.14. According to MPI inspection performed on the pipe section, there were 16 areas which contained one or more cracks. Figure 4.16 identifies 11 of the 16 regions. In all, 12 sets of peaks and troughs were actually matched and plotted according to the text output from the detection algorithm, but two sets were in such close proximity they were indistinguishable. The smaller area investigated in Fig. 4.14 provides greater detail of defect layout than does the area captured in Fig. 4.16, suggesting that the detection process may be developed as a multistage identification routine. The first step would identify defect regions within a large scan area. The second step would involve focusing in on affected areas, to obtain more detail. The success of the focusing stage would obviously depend upon the probe resolution and the sampling rate used to collect the data.

From an integrity assessment viewpoint, the probes appear to have achieved an acceptable level of detection. While the B_z signal alone cannot be used to size defects, the peak and trough magnitudes do increase with defect depth. Thus, the defect clusters not detected by the algorithm were likely shallower than the defect clusters that were detected. Assuming that the undetected defects would be too shallow to affect pipeline integrity, a fracture mechanics assessment would be based upon the deepest defects present.

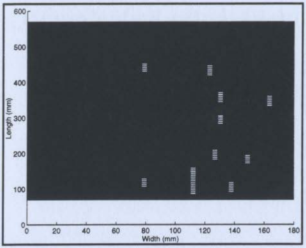


FIGURE 4.16: Detection algorithm results on a 180 mm by 500 mm section of SCC affected pipe wall.

4.3 SIGNAL PROCESSING TECHNIQUES

To this point, it has not been necessary to use signal processing techniques to identify areas of crack damage. Many of the techniques outlined in Chapter 2, that have been used with eddy current or ultrasonics, can only be applied when more than one excitation frequency is used during testing. This is not the case when using ACFM. For the work documented in this project, a single excitation frequency of 5 kHz was used to induce the AC field.

The use of cross-correlation and autocorrelation of the signals was investigated.

Cross-correlations of the ACFM data did not appear to provide useful information. Autocorrelation, on the other hand, could potentially be used to separate defect signals from background noise. Results of an autocorrelation of the B_z signal for machined colony 13 is shown in Fig. 4.17 and the autocorrelation of the signal depicted in Fig. 4.12 is illustrated in Fig. 4.18.

One drawback associated with the use of the autocorrelation function is the excessive processing time. The autocorrelation function in Matlab averaged 25 minutes to run on a Unix system, whereas, the peak detection algorithm described in Section 4.1 took only 2 minutes to process the same data file. It should also be pointed out that the autocorrelation results do not provide any information on the location of individual defects. It merely indicates that defects are present.

The autocorrelation function may find use when scan signals are noisy, to distinguish between spurious indications and actual crack damage.

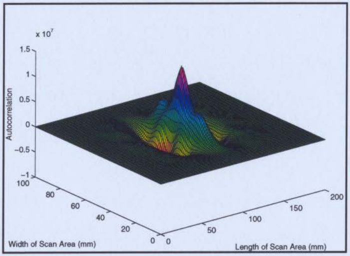


FIGURE 4.17: Autocorrelation of the B_z signal of machined colony 13

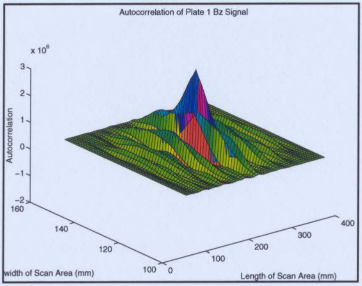


FIGURE 4.18: Autocorrelation of the B_z signal of pencil probe scan of the region shown in Fig. 4.12

4.4 DETECTION THROUGH POLYETHYLENE TAPE COATINGS

Verification of the ability of ACFM to detect SCC through pipeline coatings was obtained using a minor SCC cluster on an 8" diameter section of pipe. Figure 4.19 shows a plot of the B_X data collected with a pencil probe through the center of the cluster without tape coating on the surface. Figure 4.20 shows the results obtained for the same location covered in just under 2 mm of PE tape. No loss in signal strength is noticeable, as the background and minimum B_X levels in both plots are virtually identical.

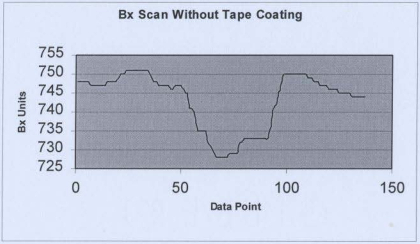


FIGURE 4.19: B_X Scan through an SCC cluster on 8" pipe section without coating

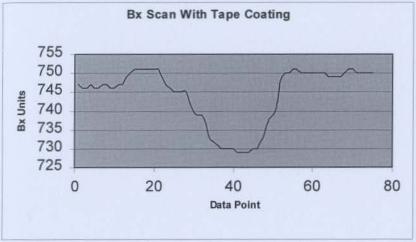


FIGURE 4.20: B_x scan through same region as in Fig, 4.19, with a PE tape coating covering the surface

CHAPTER 5: CLASSIFICATION OF COLONIES

At the outset of this project, one major issue was identified concerning the sizing of cracks occurring in clusters. The effects of close proximity defects on ACFM field perturbations had to be addressed to develop a sizing algorithm. With time limitations and budget constraints a crack cluster sizing algorithm could not be developed during the course of this research, however, information presented in this chapter could aid future investigations into this topic.

5.1 CLASSIFICATION OF MACHINED DEFECTS

Through consultation with ACFM equipment developers and an extensive literature search, no information was obtained to suggest that previous work had been carried out to model the affects of closely spaced defects on the AC field induced in a specimen. The machined defect configurations were designed in an attempt to evaluate the effects of crack proximity and position on signal behavior.

For comparison purposes, single defects were machined with geometries similar to those of the four different defects used in the machined colonies. These defects were sized using QFM software and the results are provided in Table 5.1. A point in terminology must be made at clear at this stage. *Single defects*, in the context of this chapter, were entirely alone on a test specimen. *Isolated defects* were not alone on a specimen, but were sufficiently separated from other defects so that their B_x and B_z signals were not affected.

There were some obvious discrepancies in the ACFM predictions for the single defects, which were attributed to the geometry of the machined defects, i.e. their flat-bottomed, rather than semi-elliptical profile. In all cases, the defect length was under-predicted and the depths of defects 2, 3 and 4 were also under-predicted. Curiously, the depth for defect 1 was over predicted. Since all of the defects used the same flat-bottomed profile, the investigations of the signal interactions within the colonies should not be adversely influenced by geometry effects. However, further investigation, using semi-elliptical defects, is recommended to confirm the findings presented here.

TABLE 5.1: Sizing results for single machined defects

Defect	Machined Length (mm)	Machined Depth (mm)	ACFM Predictions Length (mm)	ACFM Predictions Depth (mm)
1	75	2	67.2	2.4
2	75	5	71.4	3.1
3	50	2	43.0	0.9
4	50	5	43.0	3.1

The first colonies machined were numbers 4, 8, 12 and 16. They contained 2 mm deep defects only and, therefore, were the easiest to manufacture. Data from these colonies was imported into a spreadsheet and the scans made directly over each defect were extracted and plotted to evaluate whether or not the defect signals for single defects might linearly superimpose to form the patterns obtained for the colonies. The single 2 mm deep by 50 mm long defect was then manipulated within Matlab to form four colonies, artificially mimicking colonies 4, 8 12, and 16, illustrated in Figures 5.1 through 5.6 for machined colony 12. The signals shown in Figures 5.2, 5.4 and 5.6 were referenced to a background level of 0 to facilitate the signal addition of the single defect matrices. Adding or subtracting the background level associated with colony 12 does not affect the difference measure between the background and minimum B_X levels.

Looking first at Figures 5.1 and 5.2, the B_X signal drop for defect A in colony 12 was approximately 44 units, while the signal drop for defect A in the artificially created colony was 48 units. Similarly, for defect B, the drop in the signal for colony 12 was 56 units as opposed to 52 units for the artificially created signal. Finally, for defect C, the B_X signal drops were 40 units and 48 units, respectively. While the artificially created signals appeared to match the machined colonies reasonably well, the B_X signals drops did not vary significantly from that of the single 2 mm deep defect, for any of the cases examined. Two hypotheses were thus envisaged. Either the concept of linear superposition was valid, or the inter-crack spacing was too large for the 2 mm deep defects to have any significant influence upon each other. These findings warranted further investigation of linear superposition, so a decision was made to manufacture the remainder of the colonies.

Upon completion of the machining and scanning of all of the colonies, linear superposition was investigated in more detail. A Matlab M-file, lin_sup.m (see Appendix C), was written utilizing the peak detection algorithm described in Chapter 4 to first locate the defects within the machined colony scans and then manipulate the single defect scans into the appropriate geometries to form artificial B_X signals. Initially, the algorithm did not make any assumptions as to the depth of the defects. It used the lengths indicated in the B_Z signals to determine which defects might comprise the colonies and then constructed artificial signals based upon all possible combinations of 2 and 5 mm deep defects. For example, when analyzing colony 1 (see Fig. 3.1 and Table 3.1 or Appendix A), the algorithm recognized that based upon the lengths of the defects in colony 1 (75 mm), only single defects having a length of 75 mm could possibly form the colony. The 50 mm long defects were ignored because of their length. The algorithm then proceeded to construct the eight possible artificial colonies, based upon all of the possible combinations of single defects 1 and 2 (refer to Table 5.1). Each of the eight artificial colonies was then compared to the data file for machined colony 1. For each defect, a distance measure between the B_X signal drop for the machined colony was compared the B_X signal drop for the artificial colonies. Table 5.2 gives the results produced by this algorithm. A negative error indicated that the linearly superimposed signal had a larger drop in the B_X signal level than did the actual colony, indicating that linear superposition over predicted the depth of the defect in question. The artificial configuration best matching the machined defect signal is also given in the table. It was obvious from the results that linear superposition was not a satisfactory technique for characterizing these colonies, as it tended to under-predict defect depth.

Although it was recognized that the field perturbations associated with neighboring defects did influence one another, the TSC sizing tables for isolated defects, supplied with the QFM software, were adapted to a Matlab M-file, def_size.m (Appendix C). The defects in the machined colonies were then sized using these tables and compared to the results shown in Table 5.1. The results clearly indicate significant interactions between the 5 mm deep defects at the 15 mm inter-crack spacing used in all of the machined configurations. Table 5.3 provides the results of def_size.m. Less interaction was observed between neighboring 2 mm deep defects, however, 2 mm deep slots with 5 mm deep neighbors were oversized by as much as 140% (colony 14 in Table 5.3).

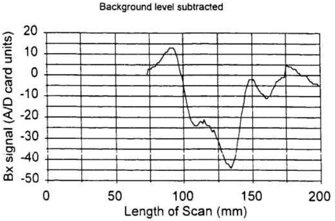


FIGURE 5.1: Defect 1 B_x signal from machined colony 12

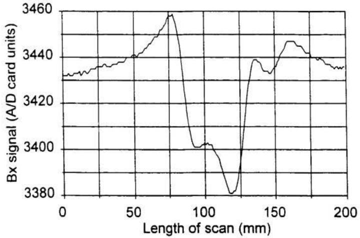


FIGURE 5.2: Defect 1 B_x signal from linear superposition in the colony 12 configuration

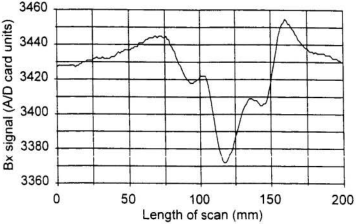


FIGURE 5.3: Defect 2 B_x signal from machined colony 12

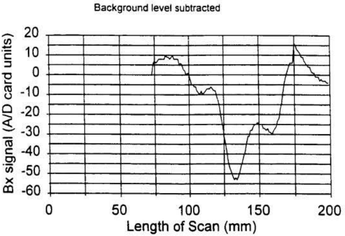


FIGURE 5.4: Defect 2 B_x signal from linear superposition in the colony 12 configuration

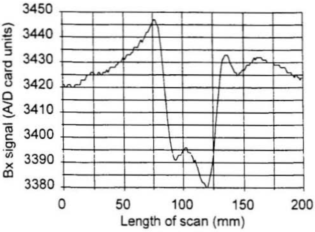


FIGURE 5.5: Defect 3 B_x signal from machined colony 12

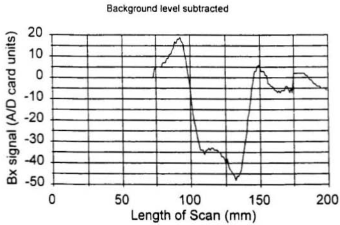


FIGURE 5.6: Defect 3 B_x signal from linear superposition in the colony 12 configuration

TABLE 5.2: Results of the linear superposition algorithm

Colony	Defect Depths Present (mm)	Superposition of Single Defects in Colony Configurations		Single Defect Combination with Minimum Error if not the Actual Combination	
		Defect	% Error	Defect	% Error
1	2	2	-0.0		
	2	2	-0.0		
	2	2	-0.1		
2	5	5	0.2	5	1.7
	5	5	-4.8	2	0.7
	5	5	-4.4	5	-1.5
3	2	2	-0.1	5	-2.5
	5	5	-4.9	2	1.3
	2	2	-1.3	2	0.5
4	2	2	0.4		
	2	2	-0.2		
	2	2	-0.3		
5	5	5	-1.4	5	-1.0
	5	5	-5.1	2	2.7
	5	5	-5.0	2	2.5
6	2	2	1.4	2	1.0
	5	5	-3.7	2	3.9
	2	2	1.2	2	0.9
7	5	5	-0.7		
	2	2	0.5		
	5	5	-3.5		
8	2	2	0.4		
	2	2	-0.1		
	2	2	-0.1		
9	5	5	-0.6	5	0.0
	5	5	-4.5	2	3.3
	5	5	-4.8	2	3.2
10	2	2	1.3	2	1.2
	5	5	-4.9	2	2.7
	2	2	0.0	2	0.1
11	5	5	-1.4	5	-0.7
	2	2	-4.6	2	-0.5
	5	5	2.2	2	3.1
12	2	2	0.3		
	2	2	0.3		
	2	2	0.1		

TABLE 5.2: Results of the linear superposition algorithm (cont'd)

Colony	Defect Depths Present (mm)	Superposition of Single Defects in Colony Configurations		Single Defect Combination with Minimum Error if not the Actual Combination	
		Defect	% Error	Defect	% Error
13	5	5	-1.5	5	-0.3
	5	5	-3.4	2	3.7
	5	5	-3.5	5	-2.7
14	2	2	1.0	2	1.7
	5	5	-8.2	2	-1.3
	2	2	-0.3	2	0.2
15	5	5	0.2		
	2	2	0.8		
	5	5	-2.6		
16	2	2	0.4		
	2	2	0.8		
	2	2	0.7		
17	5	5	-1.8	5	-1.2
	5	5	-3.9	2	3.5
	5	5	-3.5	5	-2.8
18	2	2	0.5		
	5	5	-3.5		
	2	2	0.2		
19	5	5	-1.3		
	2	2	0.8		
	5	5	-3.2		
20	2	2	0.8	2	0.7
	2	2	-0.0	2	0.8
	5	5	-4.8	2	3.0
21	5	5	-4.3	2	0.0
	5	5	-3.5	5	-3.0
	2	2	3.5	2	3.6

Investigation into linear superposition and the TSC sizing tables did not provide satisfactory solutions to the problem of developing a classification routine for defects within clusters. The development of an empirical influence factor formula (which would use parameters such as defect length, inter-crack spacing, a relative position measure to account for crack alignment, etc.) to obtain an approximate measure of the influence of a defect on its surroundings was also considered. Such a formula might then be used in an iterative routine for classification. However, the procedure could not be pursued in any great depth for a number of reasons. First of all, the data set available was too sparse to adequately determine the influence of most parameters and sufficient funding was not available to expand the size of the test set. Information based upon two length values (50 and 75 mm) would provide little insight into the influence of defect length, and a similar argument could be made for the depth parameter. Additionally, in order to properly examine these factors, the data set should have first been comprised of colonies containing only two defects. Then, upon analysis of that data, three defect groupings could be considered, and so on. Finally, the data, in its present form, relates to defects having flat-bottomed profiles. It may be a futile effort to develop a characterization routine for a defect geometry which is rarely observed in practice.

Another possible approach would have been an investigation using the theoretical equations developed at University College London for sizing isolated defects, in an attempt to relate them to multiple crack groupings. This option was not pursued as these equations are proprietary information.

Rather than attempting to develop the defect classification algorithm further with this limited data set, it was decided to study the information provided in Tables 5.2 and 5.3 in an effort to gain an improved understanding of the manner in which multiple defect signals interact.

Close examination of the linear superposition and sizing data for colonies containing only 2 mm deep defects revealed important observations. It appeared that at the 15 mm inter-defect spacing, the parallel 2 mm deep defects were sufficiently remote as not to significantly affect the B_X signal drop of neighboring defects. This was apparent in colonies 1, 4, and 8, where the linear superposition algorithm regenerated the ACFM signals with a maximum error of only 0.4%. This translated to a maximum error in sizing on the order of 0.3 mm. Colonies 12 and 16 also contained only 2 mm deep defects but the configurations contained overlapping defects. In those colonies the error in regenerating the signals increased to 0.8%, which while appearing quite small, translated to an over prediction in defect depth up to 0.9 mm. This suggested that overlapping of crack ends may affect sizing results.

To further investigate the relationship between defect depth and field perturbations, B_X scans of single 2, 3, 4 and 5 mm deep slits were examined (all 50 mm long). The results were quite interesting. The 2 mm deep defect appeared to cause perturbations in the field up to 12 mm on either side of the slit. This was consistent with the observation in colonies 1, 4 and 8. The 15 mm inter-defect spacings used in the colonies appeared to have negligible affect on sizing as the centerline of the slots were 3 mm beyond the range of the perturbations. For the 3 mm deep slot, the field perturbations extended to about 14 mm either side of the slot. For the 4 mm slot, field perturbations increased to 16 mm and for the 5 mm slot, the perturbations extended to about 18 mm. Thus, for every one millimeter increase in depth, field perturbations extended approximately two additional millimeters farther from the defect. An empirical relationship was developed to predict this distance as a function of crack depth for the 5 mm probe:

$$d = 2a + 8 \tag{5.1}$$

Where: d : distance either side of crack (mm)
a : crack depth (mm)

The same exercise was repeated using the microprobe. The extent of the field perturbations was found to be two millimeters less on either side of the defect than for the pencil probe. This suggested the following revision to Equation 5.1 for the microprobe:

$$d = 2a + 6 \tag{5.2}$$

At this stage, Equations 5.1 and 5.2 can only be considered valid for parallel defects. As previously discussed in relation to colonies 12 and 16, the overlapping of the ends of defects appeared to add an additional influence into the relationship, which increased the perturbation effects.

It appeared that defect length had little influence on the crack depth field perturbation relationships from Equations 5.1 and 5.2. Examining the scans of the single 2 mm deep by 75 mm long defect and the 5 mm deep by 75 mm long defect indicated the same extent of field perturbations either side of the defects as noticed for the shorter slots. These observations may only be valid for certain length and depth ranges and should be investigated further on semi-elliptical defect profiles of a variety of lengths.

The largest errors in lin_sup.m and def_size.m occurred when the 5 mm deep defects were present. This was seen to lead to gross over predictions for the depth of neighboring defects and usually lead to improper classifications using the linear superposition technique. Again, it was apparent that the deeper the defect, the greater its influence on the surrounding AC field.

Another point of interest arose when comparing colonies 5, 9 and 13, which all contain similar 5 mm defects at varying degrees of overlap (Fig. 5.7). In colony 9, all of the defects were fairly remote from one another and the sizing algorithm reflected this as defect depth predictions were fairly close to the expected prediction of 3.1 mm for a single 5 mm deep flat-bottomed defect. Looking to colony 5, there appeared to be cancellation effects as the rise in the B_X level at the ends of the defects lead to a significant under prediction in sizing results. In the colony 13, which had overlapping defects, the sizing error switched from an under prediction of defect depth to an over prediction. These findings suggested that the superposition of B_X signals was not necessarily additive, but may also have involved cancellation effects related to defect positioning.

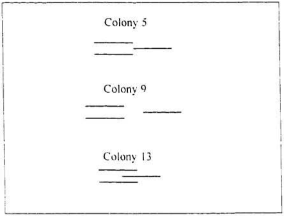


FIGURE 5.7: Configurations of Colonies 5, 9 and 13

TABLE 5.3: Results of the sizing Algorithm on the machined defect colonies

Colony	Defect	Machined Features		Colony Sizing Results		Isolated Defect Sizing		Percentage Difference	
		Length (mm)	Depth (mm)	Length (mm)	Depth (mm)	Length (mm)	Depth (mm)	Length (%)	Depth (%)
1	A	75	2	69.3	2.4	67.2	2.4	3.0	0.0
	B	75	2	69.3	2.5	67.2	2.4	3.0	4.0
	C	75	2	69.3	2.3	67.2	2.4	3.0	-4.3
2	A	75	5	67.2	4.4	71.4	3.1	-6.3	29.5
	B	75	5	67.2	5.3	71.4	3.1	-6.3	41.5
	C	75	5	67.2	4.6	71.4	3.1	-6.3	32.6
3	A	50	2	43.0	2.4	43.0	0.9	0.0	62.5
	B	50	5	43.0	3.1	43.0	3.1	0.0	0.0
	C	50	2	43.0	1.8	43.0	0.9	0.0	50.0
4	A	50	2	43.0	1.2	43.0	0.9	0.0	25.0
	B	50	2	43.0	0.9	43.0	0.9	0.0	0.0
	C	50	2	43.0	0.9	43.0	0.9	0.0	0.0
5	A	50	5	43.0	2.6	43.0	3.1	0.0	-19.2
	B	50	5	43.0	2.0	43.0	3.1	0.0	-55.0
	C	50	5	43.0	2.5	43.0	3.1	0.0	-24.0
6	A	50	2	45.1	1.6	43.0	0.9	4.7	43.8
	B	50	5	45.1	2.7	43.0	3.1	4.7	-14.8
	C	50	2	45.1	1.5	43.0	0.9	4.7	40.0
7	A	50	5	43.0	3.1	43.0	3.1	0.0	0.0
	B	50	2	43.0	1.8	43.0	0.9	0.0	50.0
	C	50	5	43.0	3.2	43.0	3.1	0.0	3.1

TABLE 5.3: Results of the sizing Algorithm on the machined.defect colonies (cont'd)

Colony	Defect	Machined Features		Colony Sizing Results		Isolated Defect Sizing		Percentage Difference	
		Length (mm)	Depth (mm)	Length (mm)	Depth (mm)	Length (mm)	Depth (mm)	Length (%)	Depth (%)
8	A	50	2	45.1	1.2	43.0	0.9	4.7	25.0
	B	50	2	45.1	1.0	43.0	0.9	4.7	10.0
	C	50	2	45.1	1.1	43.0	0.9	4.7	18.2
9	A	50	5	43.0	3.1	43.0	3.1	0.0	0.0
	B	50	5	43.0	2.5	43.0	3.1	0.0	-24.0
	C	50	5	43.0	2.6	43.0	3.1	0.0	-19.2
10	A	50	2	45.1	1.5	43.0	0.9	4.7	40.0
	B	50	5	45.1	2.4	43.0	3.1	4.7	-29.2
	C	50	2	45.1	1.2	43.0	0.9	4.7	25.0
11	A	50	5	43.0	2.9	43.0	3.1	0.0	-6.9
	B	50	2	43.0	1.7	43.0	0.9	0.0	47.1
	C	50	5	43.0	2.6	43.0	3.1	0.0	-19.2
12	A	50	2	45.1	1.2	43.0	0.9	4.7	25.0
	B	50	2	45.1	1.8	43.0	0.9	4.7	50.0
	C	50	2	45.1	1.6	43.0	0.9	4.7	43.8
13	A	50	5	45.1	3.6	43.0	3.1	4.7	13.9
	B	50	5	45.1	4.9	43.0	3.1	4.7	36.7
	C	50	5	45.1	3.6	43.0	3.1	4.7	13.9
14	A	50	2	45.1	2.4	43.0	0.9	4.7	62.5
	B	50	5	45.1	3.0	43.0	3.1	4.7	-3.3
	C	50	2	45.1	1.8	43.0	0.9	4.7	50.0

TABLE 5.3: Results of the sizing Algorithm on the machined defect colonies (cont'd)

Colony	Defect	Machined Features		Colony Sizing Results		Isolated Defect Sizing		Percentage Difference	
		Length (mm)	Depth (mm)	Length (mm)	Depth (mm)	Length (mm)	Depth (mm)	Length (%)	Depth (%)
15	A	50	5	45.1	3.6	43.0	3.1	4.7	13.9
	B	50	2	45.1	3.3	43.0	0.9	4.7	72.7
	C	50	5	45.1	3.7	43.0	3.1	4.7	16.2
16	A	50	2	45.1	1.6	43.0	0.9	4.7	43.8
	B	50	2	45.1	1.5	43.0	0.9	4.7	40.0
	C	50	2	45.1	1.5	43.0	0.9	4.7	40.0
17	A	50	5	47.6	3.3	43.0	3.1	9.7	6.1
	B	50	5	47.6	3.3	43.0	3.1	9.7	6.1
	C	50	5	47.6	3.2	43.0	3.1	9.7	3.1
18	A	50	2	45.1	2.2	43.0	0.9	4.7	59.1
	B	50	5	45.1	2.9	43.0	3.1	4.7	-6.9
	C	50	2	45.1	1.6	43.0	0.9	4.7	43.8
19	A	50	5	45.1	2.9	43.0	3.1	4.7	-6.9
	B	50	2	45.1	2.1	43.0	0.9	4.7	57.1
	C	50	5	45.1	3.0	43.0	3.1	4.7	-3.3
20	A	50	2	43.0	1.6	43.0	0.9	0.0	43.8
	B	50	2	43.0	1.8	43.0	0.9	0.0	50.0
	C	50	5	43.0	2.4	43.0	3.1	0.0	-29.2
21	A	50	2	45.1	2.1	43.0	0.9	4.7	57.1
	B	50	5	45.1	3.2	43.0	3.1	4.7	3.1
	C	50	5	45.1	3.1	43.0	3.1	4.7	0.0

5.2 SCC SIZING

No attempts were made to size individual stress corrosion cracks located within dense colony clusters. However, three of the more isolated defects were sized using three different probes and the pipe specimens were sectioned to verify the ACFM sizing results. The results are shown in Table 5.4.

TABLE 5.4: Sizing predictions for isolated stress corrosion cracks

Defect	ACFM Sizing results (mm)						Actual Defect Dimensions (mm)	
	Microprobe		Pencil Probe		Weld Probe			
	Depth	Length	Depth	Length	Depth	Length	Depth	Length
1	3.3	9.3	2.9	9.3	3.1	8.3	3.09	13.0
2	6.9	17.4	7.2	17.4	6.5	17.2	3.62	19.0
3	2.0	12.3	1.7	12.3	1.8	11.8	1.98	14.0

In the case of cracks 1 and 3, the depth estimates were quite reasonable within a 0.3 mm range. The depth estimate for the second crack was almost double that of the actual defect size. Upon further investigation, it was discovered that defect 2 actually grew at an angle that was not normal to the surface of the pipe wall. The value recorded as the actual depth of the defect, was the actual through thickness depth, which would be used in a fracture mechanics assessment. The value measured using ACFM was the slanted depth of the defect a common phenomenon with electromagnetic NDT techniques. While the measurement provided by ACFM was consistent with ACFM theory (measurement of AC field flow around the crack), it had not been anticipated that SCC defects grew at an angle other than normal to the surface of the material. This issue would have to be addressed by future sizing algorithms to allow for accurate fracture mechanics assessments.

CHAPTER 6: CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

While it appears that the goal of adapting ACFM to the detection of SCC damage in natural gas transmission pipelines can be achieved, a number of obstacles remain to be overcome before individual defects within colony clusters may be characterized. Such factors can only be addressed in a larger scale research program involving larger data sets and increased funding.

The use of the ACFM probes with both 5 and 2 mm pick up coils (pencil probe and microprobe) demonstrated that current technology has the capability of detecting SCC clusters, however, it will not provide as accurate an image as MPI since signals from closely spaced cracks can superimpose to appear as one single defect. Both NDT techniques working in conjunction could speed up inspection time, using ACFM to identify the exact crack locations without the need for extensive surface cleaning, then following up with localized cleaning and MPI.

The pencil probe, having larger pick-up and induction coils offered a better signal to noise ratio, while the smaller coils in the microprobe had provided better resolution. Future development of array probe technology using a large induction coil and 2 mm (or smaller) pick-up coils may provide an SCC probe with both a good signal to noise ratio and acceptable resolution.

Methodologies for automating the detection of defect signals were successful. When sizing weld fatigue cracks, the ends of a crack are identified from a peak and trough in the B_Z field component (see Fig. 2.4) resulting from the curvature of the induced AC field around the crack ends. The peak detection algorithm was adequate in identifying both the machined and real defects cluster used in the research. It will demonstrate its true potential when assessing meters of data from large diameter pipe in the field. Applying the autocorrelation function to defect signals appeared effective in identifying cracking, but was very demanding on computer processing power. The peak detection routine described in Chapter 4 was much more efficient, reducing processing time from 25 minutes, for the autocorrelation function, to 2 minutes on the same data.

For the 21 machined colonies containing a total of 63 defects, only one defect was missed using the initial threshold criteria in the peak detection algorithm. The defect was later identified with an adjustment to the criteria. The same detection algorithm also functioned well for natural SCC present in the pipe wall sections. While probe resolution was not sufficient to distinguish between all defects within clusters, the majority of the SCC sites were readily identified.

The ability of ACFM to detect cracking through non-conductive coatings up to 5 mm in thickness is an attractive feature as pipelines most susceptible to SCC damage are those coated with PE tape. Typically as the lift-off of the probe is increased, the signal strength is decreased, however, as documented in Chapter 4, a 5 mm pencil probe is able to detect SCC through 1.7 mm of PE tape without a loss in signal strength.

The test set used in this research provided insufficient information to develop an algorithm to accurately size defects within clusters. In addition, the flat-bottom profile of the machined defects was not representative of the semi-elliptical shapes usually observed in practice. Significant information was acquired which will be useful in designing a characterization routine in the future. First of all, the superposition of defect signals is not linear. The linear superposition algorithm will likely under predict the extent of damage in a region. Secondly, the influence of a defect on AC field perturbations is proportional to the depth of the defect (refer to Eqn's 5.1 and 5.2). Thirdly, the relative position of defects within a colony has a significant influence on the measured AC field perturbations as parallel defects of equal length seemed to affect the ACFM signals differently than defects with overlapping ends.

Sizing of two larger cracks in the SCC clusters resulted in depth estimates within 0.3 mm of the actual depth while slightly under predicting crack length. The use of MPI was able to verify the actual crack length. The depth of a third crack was over predicted because it propagated through the pipe wall at an angle of about 45°. When defects grow at some angle other than normal to the pipe surface, the tendency for ACFM is to measure the slanted depth of a defect, as opposed to the through thickness depth, leading to conservative depth estimates. This phenomenon will have to be addressed in future work to improve the reliability of sizing predictions. It may be possible to overcome this effect with more detailed modeling of AC field perturbations or the use of complimentary NDT techniques (i.e. shear wave UT or electromagnetic acoustic transmission probes).

In order to accurately evaluate the effects of parameters such as defect length, defect depth, defect location and inter-crack spacing, sample sets must be designed so that all variables, except the variable being observed, remain constant. The ACFM signal interactions of two neighboring defects should be thoroughly evaluated before moving on to larger groupings. A larger sample of real SCC colonies in pipe wall sections is required and allowances have to be made for detailed sectioning of these specimens to study colony configurations and their influences on AC field perturbations.

Access to the theoretical model developed at University College London for predicting the magnetic field perturbations associated with an isolated semi-elliptical surface crack could be useful when developing a characterization algorithm. This model may provide insight into defect signal interactions.

The use of ACFM probe configurations other than those commercially available, should also be investigated. One possible configuration, which would likely yield useful results, would be an array of 2 mm pick-up coils with a large induction coil. This arrangement may result in a more sensitive probe with a higher signal to noise ratio. The current 2 mm probe design incorporates a small induction coil to reduce "free edge" effects, but edge effects are never encountered when scanning continuous pipe walls.

Upon completion of future research into the development of refined detection and sizing algorithms, ACFM's role in the pipeline integrity industry's SCC management programs could be vital. After identification of a potential SCC site using an ILI tool and subsequent excavation of the site, ACFM could be used to inspect the suspect location prior to any significant surface preparation work and confirm the exact location of the anomaly. Current procedures require extensive and costly cleaning of an entire joint as the locations identified by ILI tools are not precise. Once identified, the defect location could be cleaned and MPI performed so that a photograph of the cluster could be taken for permanent record keeping. An ACFM SCC sizing algorithm could then identify the depth of cluster and the appropriate repair could be made.

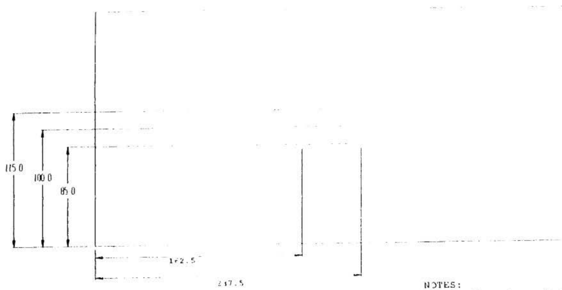
APPENDIX A

MACHINED DEFECT COLONY CONFIGURATIONS AND ACFM SCANS

NOTE:

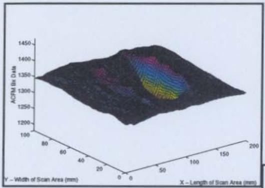
The plates used for the machining of the defect colonies were 200 mm by 400 mm and made form mild carbon steel.

Slotted Plate I

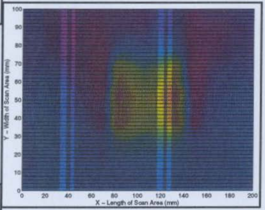


NOTES:
1) all dimensions in mm
2) all defects 2 mm deep

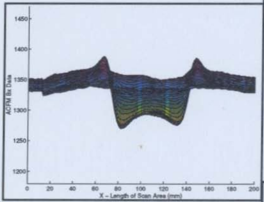
MACHINED COLONY 1



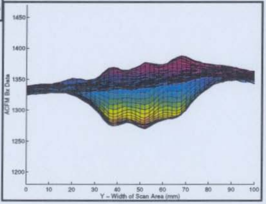
B_x Isometric



B_x Contour Plot

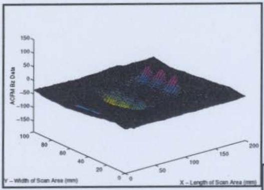


B_x View Along Length of Scan Area

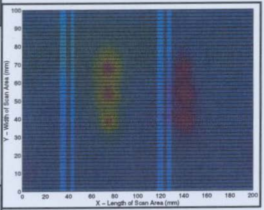


B_x View Along Width of Scan Area

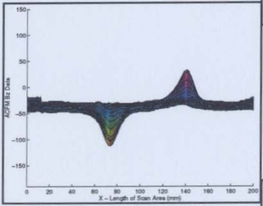
MACHINED COLONY 1



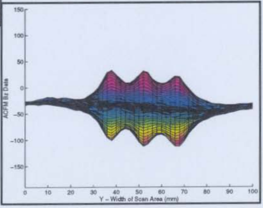
B_z Isometric



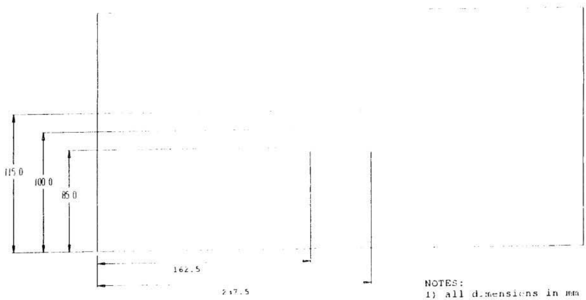
B_z Contour Plot



B_z View Along Length of Scan Area

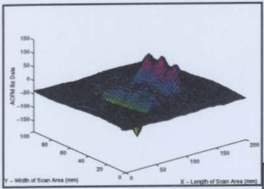


B_z View Along Width of Scan Area



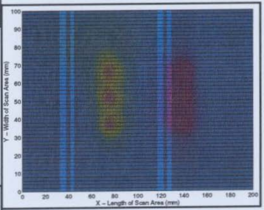
NOTES:
1) all dimensions in mm
2) all slots 5 mm deep

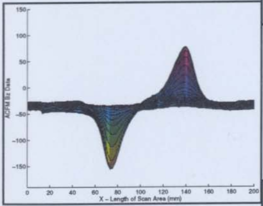
MACHINED COLONY 2



B_z Isometric

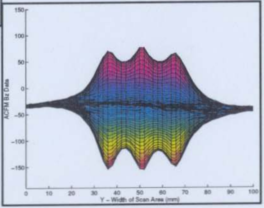
B_z Contour Plot



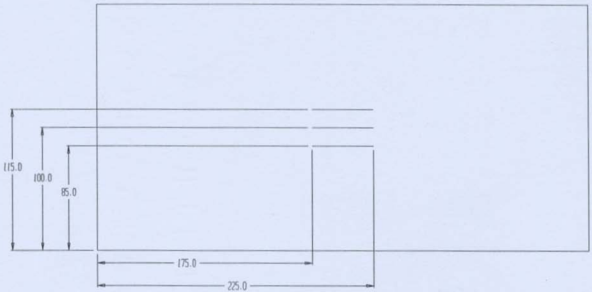


B_z View Along Length of Scan Area

B_z View Along Width of Scan Area

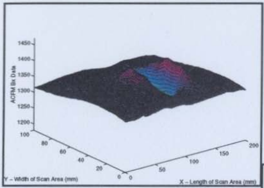


Slitted Plate 3

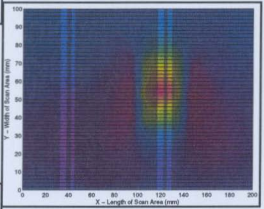


NOTES:
1) all dimensions in mm
2) Outside slits 2 mm deep
5) Middle slit 5 mm deep

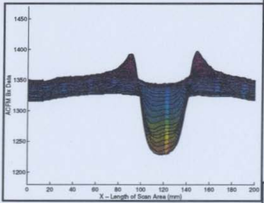
MACHINED COLONY 3



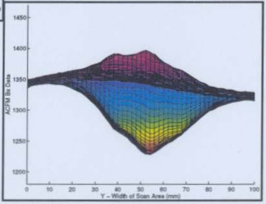
B_x Isometric



B_x Contour Plot

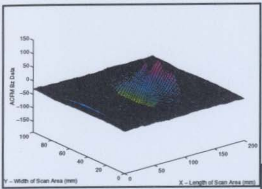


B_x View Along Length of Scan Area

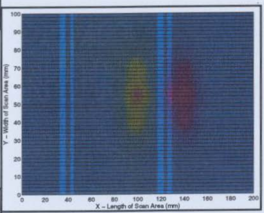


B_x View Along Width of Scan Area

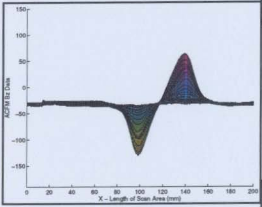
MACHINED COLONY 3



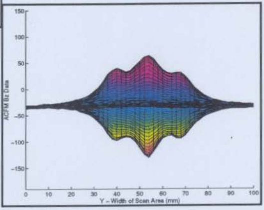
B_z Isometric



B_z Contour Plot

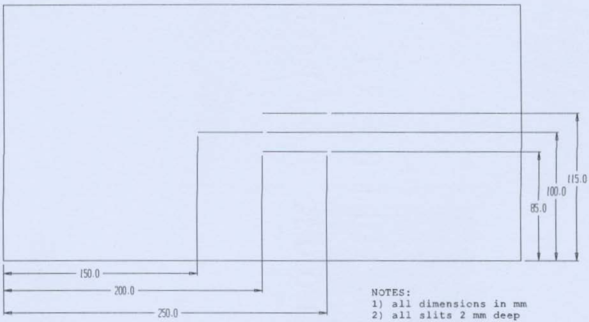


B_z View Along Length of Scan Area



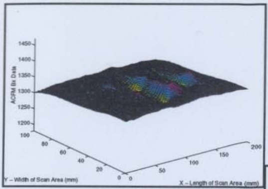
B_z View Along Width of Scan Area

Slitted Plate 4



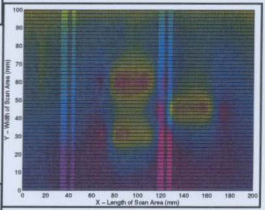
NOTES:
1) all dimensions in mm
2) all slits 2 mm deep

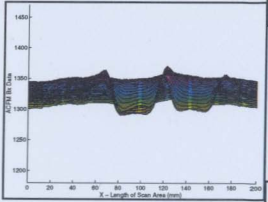
MACHINED COLONY 4



B_x Isometric

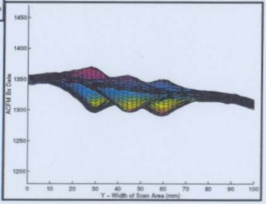
B_x Contour Plot



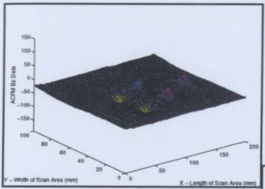


B_x View Along Length of Scan Area

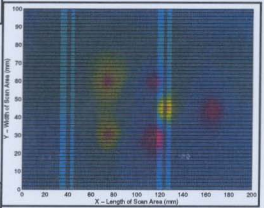
B_x View Along Width of Scan Area



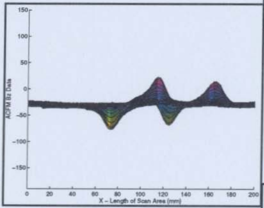
MACHINED COLONY 4



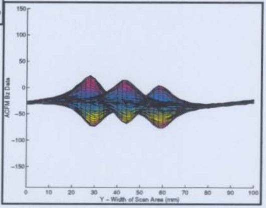
B_z Isometric



B_z Contour Plot

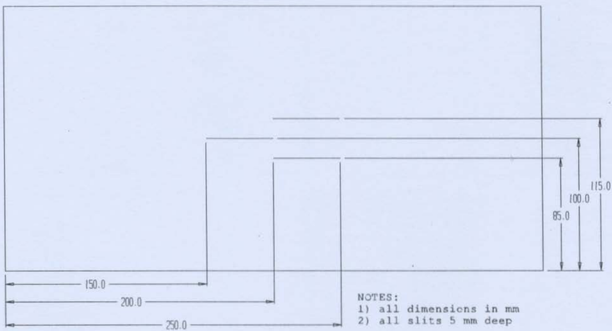


B_z View Along Length of Scan Area

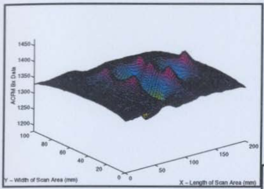


B_z View Along Width of Scan Area

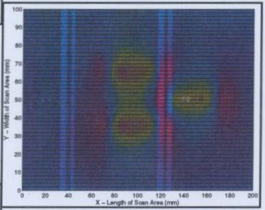
Slitted Plate 5



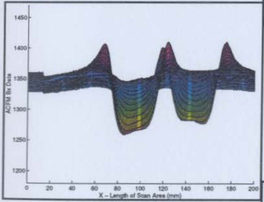
MACHINED COLONY 5



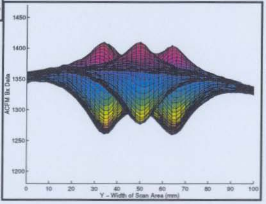
B_x Isometric



B_x Contour Plot

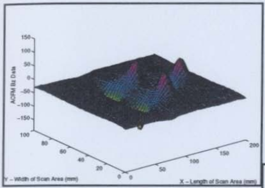


B_x View Along Length of Scan Area



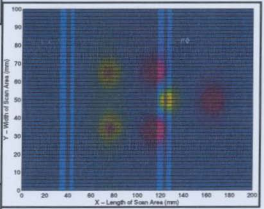
B_x View Along Width of Scan Area

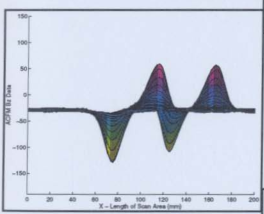
MACHINED COLONY 5



B_z Isometric

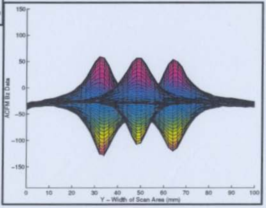
B_z Contour Plot



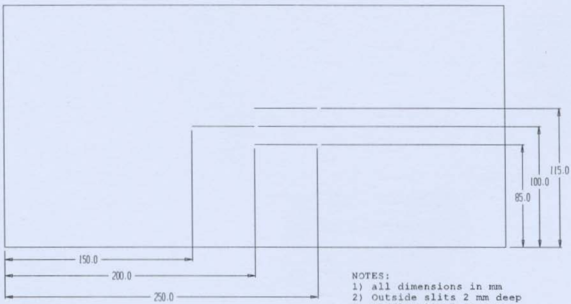


B_z View Along Length of Scan Area

B_z View Along Width of Scan Area

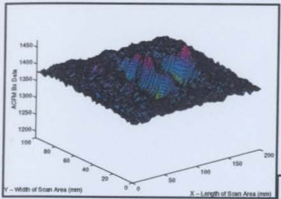


Slitted Plate 6



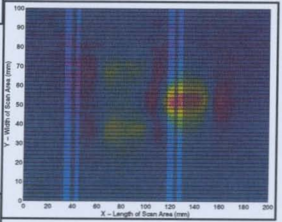
NOTES:
1) all dimensions in mm
2) Outside slits 2 mm deep
3) Middle slit 5 mm deep

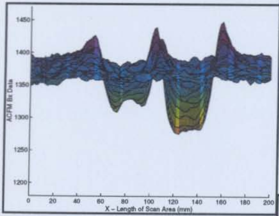
MACHINED COLONY 6



B_x Isometric

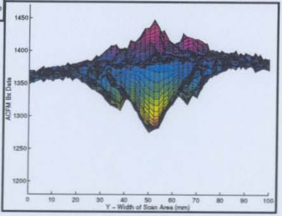
B_x Contour Plot



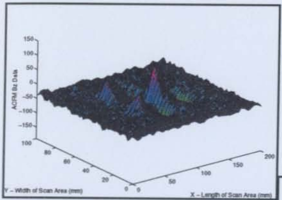


B_x View Along Length of Scan Area

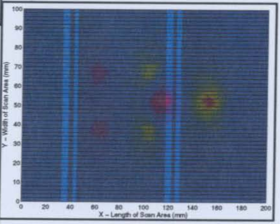
B_x View Along Width of Scan Area



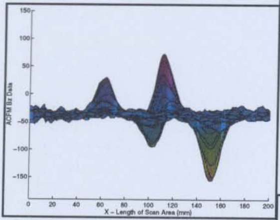
MACHINED COLONY 6



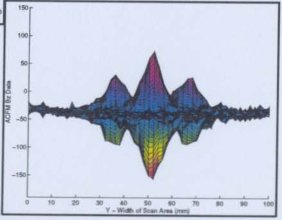
B_z Isometric



B_z Contour Plot

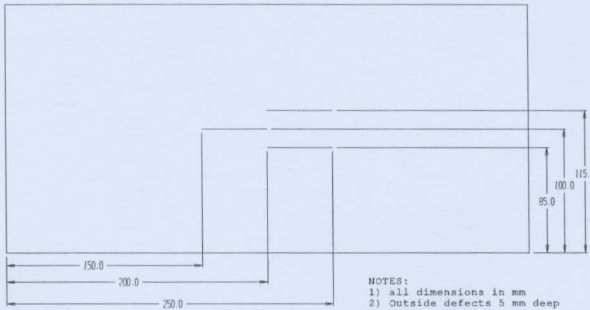


B_z View Along Length of Scan Area



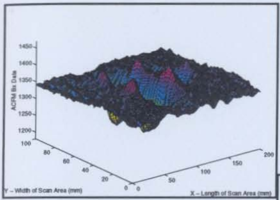
B_z View Along Width of Scan Area

Slitted Plate 7



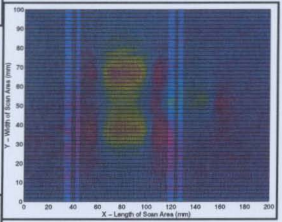
NOTES:
1) all dimensions in mm
2) Outside defects 5 mm deep
3) Middle defect 2 mm deep

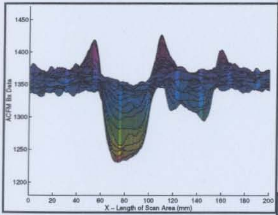
MACHINED COLONY 7



B_x Isometric

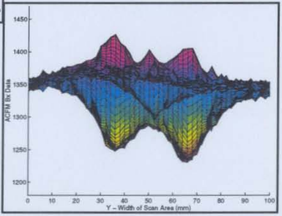
B_x Contour Plot



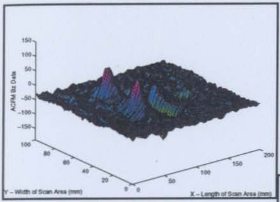


B_x View Along Length of Scan Area

B_x View Along Width of Scan Area

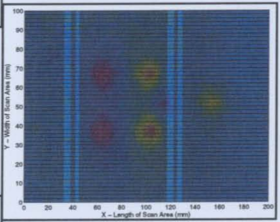


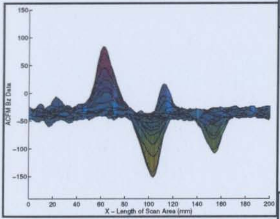
MACHINED COLONY 7



B_z Isometric

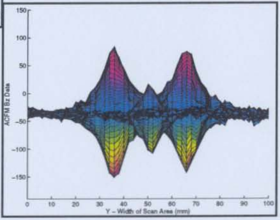
B_z Contour Plot



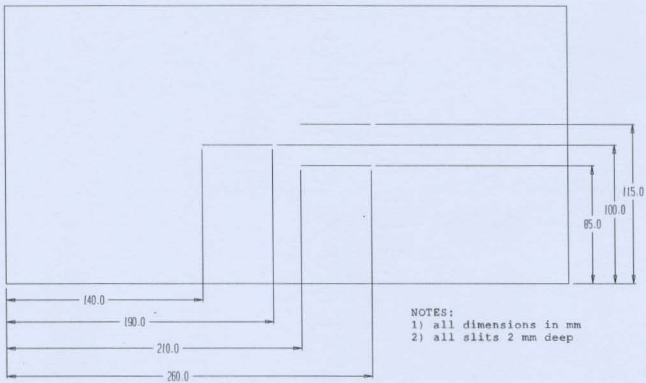


B_z View Along Length of Scan Area

B_z View Along Width of Scan Area

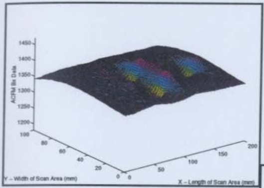


Slitted Plate 8



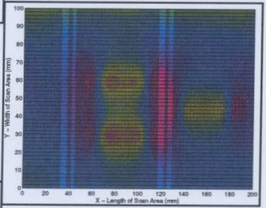
NOTES:
1) all dimensions in mm
2) all slits 2 mm deep

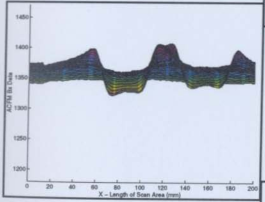
MACHINED COLONY 8



B_x Isometric

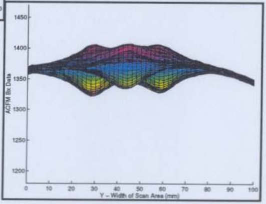
B_x Contour Plot



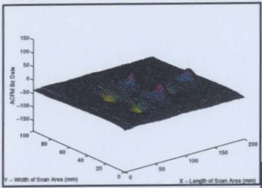


B_x View Along Length of Scan Area

B_x View Along Width of Scan Area

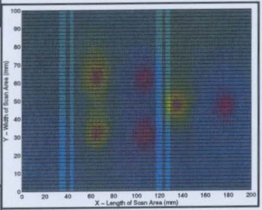


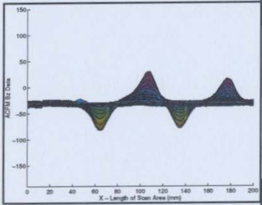
MACHINED COLONY 8



B_z Isometric

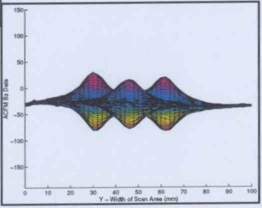
B_z Contour Plot



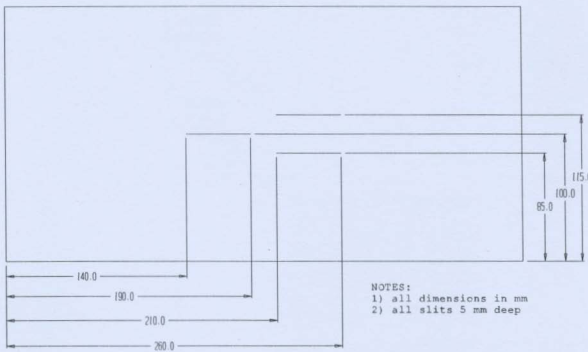


B_z View Along Length of Scan Area

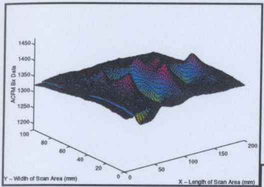
B_z View Along Width of Scan Area



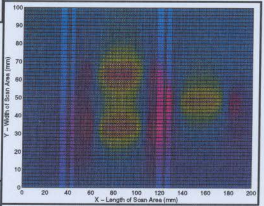
Slitted Plate 9



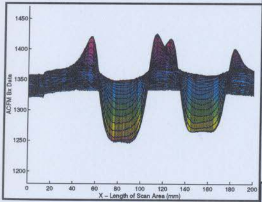
MACHINED COLONY 9



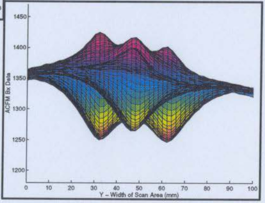
B_x Isometric



B_x Contour Plot

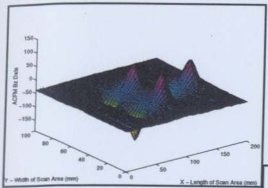


B_x View Along Length of Scan Area



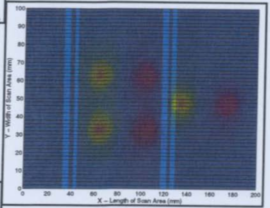
B_x View Along Width of Scan Area

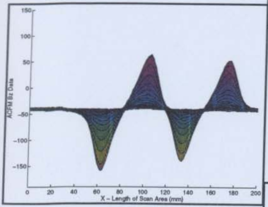
MACHINED COLONY 9



B_z Isometric

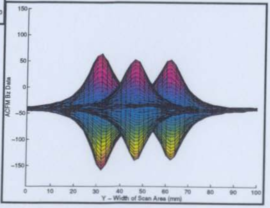
B_z Contour Plot



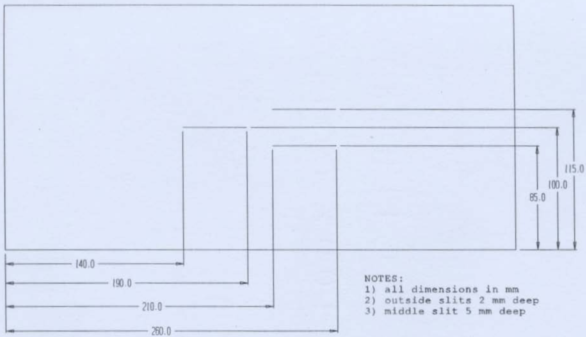


B_z View Along Length of Scan Area

B_z View Along Width of Scan Area

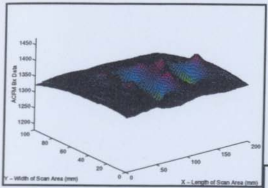


Slitted Plate 10

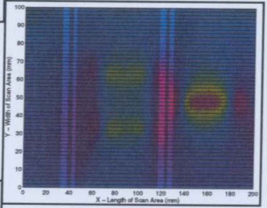


NOTES:
1) all dimensions in mm
2) outside slits 2 mm deep
3) middle slit 5 mm deep

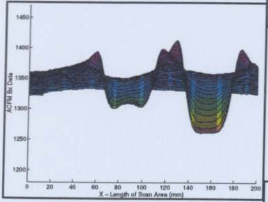
MACHINED COLONY 10



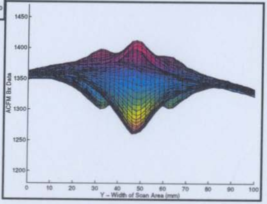
B_x Isometric



B_x Contour Plot

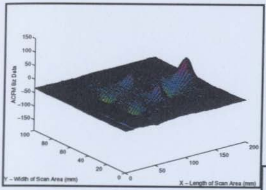


B_x View Along Length of Scan Area

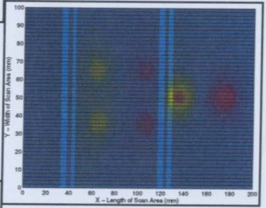


B_x View Along Width of Scan Area

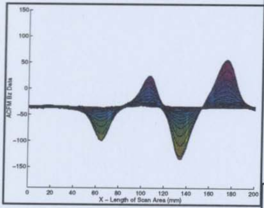
MACHINED COLONY 10



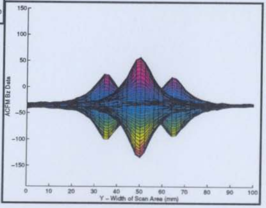
B_z Isometric



B_z Contour Plot

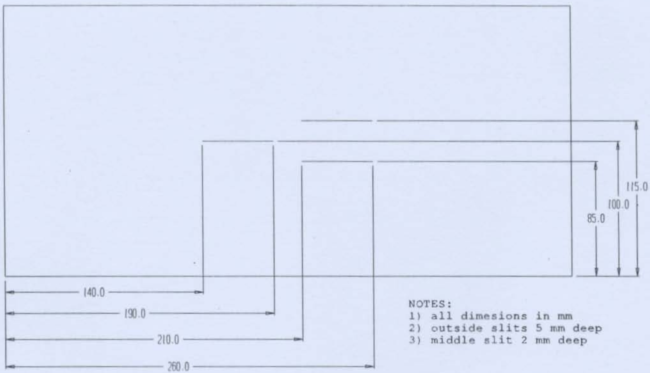


B_z View Along Length of Scan Area

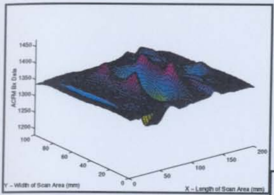


B_z View Along Width of Scan Area

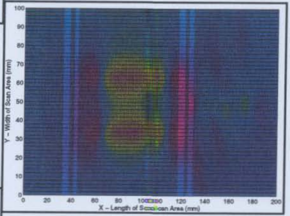
Slitted Plate II



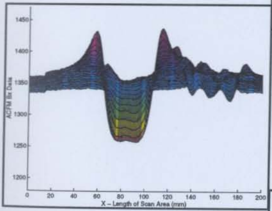
MACHINED COLONY 11



B_x Isometric

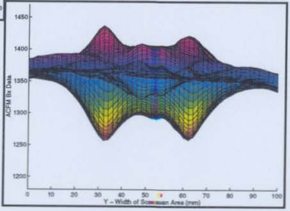


B_x Contour Plot

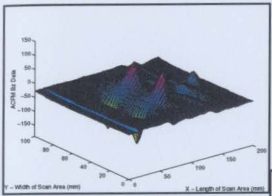


B_x View Along Length of Scan Area

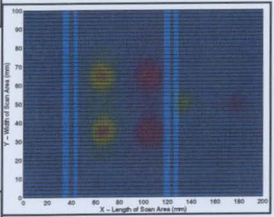
B_x View Along Width of Scan Area



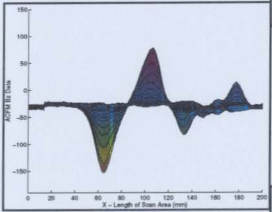
MACHINED COLONY 11



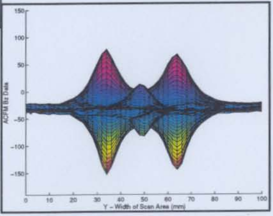
B_z Isometric



B_z Contour Plot

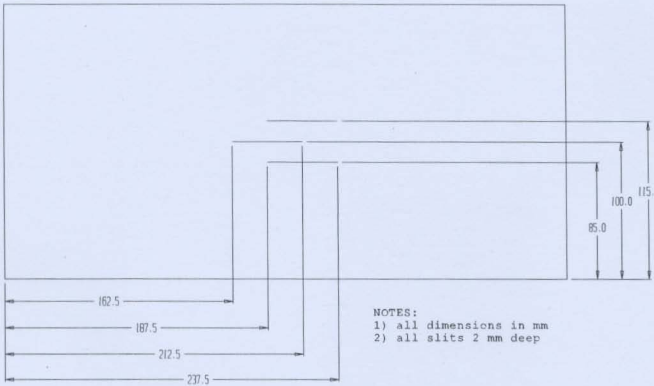


B_z View Along Length of Scan Area

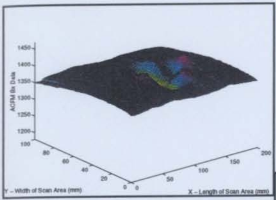


B_z View Along Width of Scan Area

Slitted Plate 12

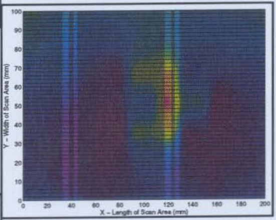


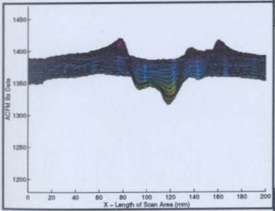
MACHINED COLONY 12



B_x Isometric

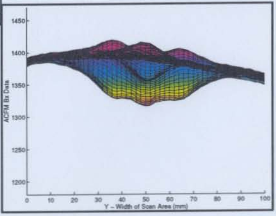
B_x Contour Plot



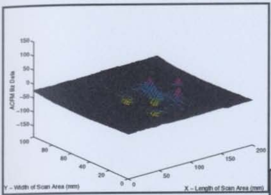


B_x View Along Length of Scan Area

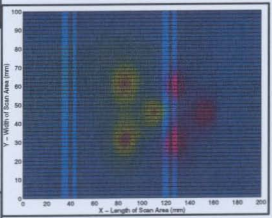
B_x View Along Width of Scan Area



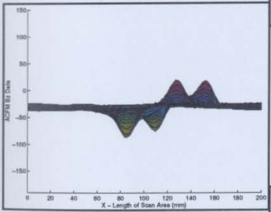
MACHINED COLONY 12



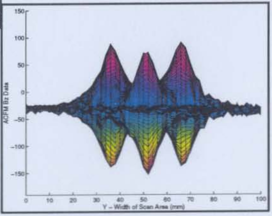
B_z Isometric



B_z Contour Plot

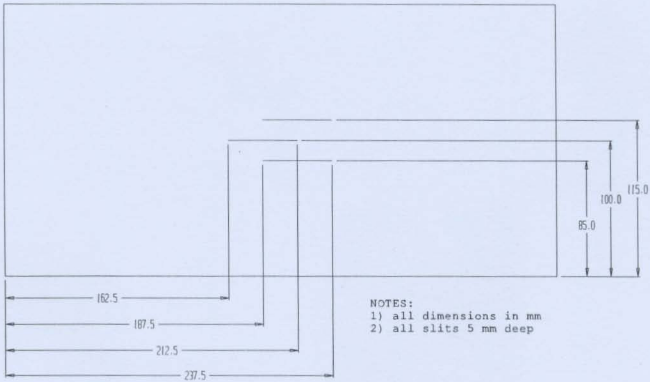


B_z View Along Length of Scan Area



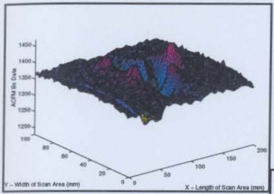
B_z View Along Width of Scan Area

Slitted Plate 13

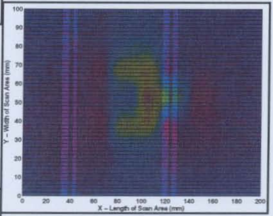


NOTES:
1) all dimensions in mm
2) all slits 5 mm deep

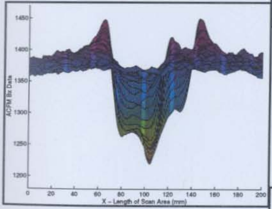
MACHINED COLONY 13



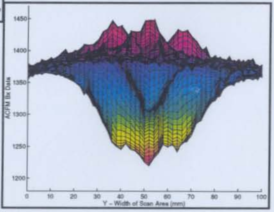
B_x Isometric



B_x Contour Plot

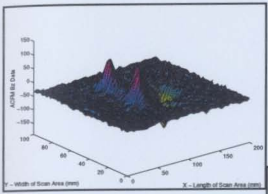


B_x View Along Length of Scan Area

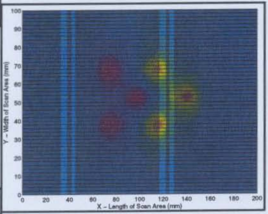


B_x View Along Width of Scan Area

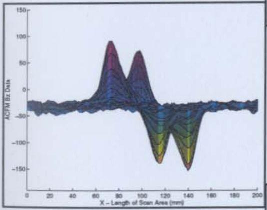
MACHINED COLONY 13



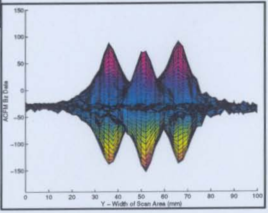
B_z Isometric



B_z Contour Plot

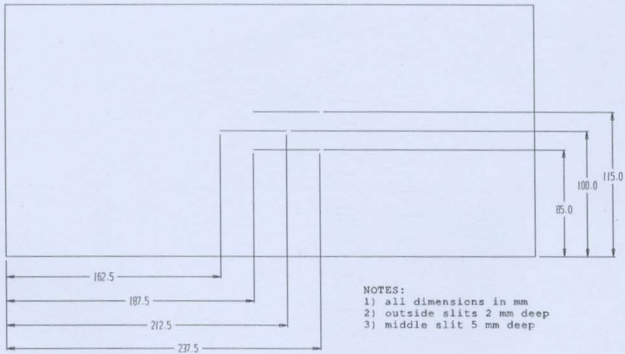


B_z View Along Length of Scan Area



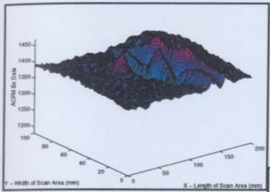
B_z View Along Width of Scan Area

Slitted Plate 14



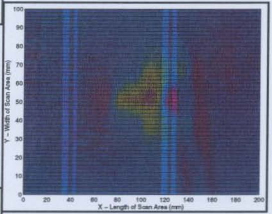
NOTES:
1) all dimensions in mm
2) outside slits 2 mm deep
3) middle slit 5 mm deep

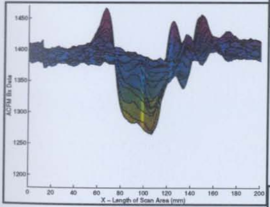
MACHINED COLONY 14



B_x Isometric

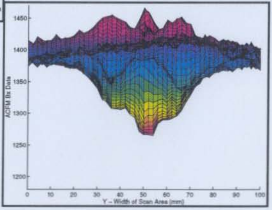
B_x Contour Plot



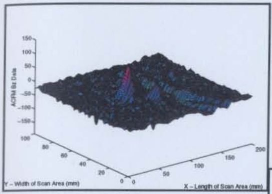


B_x View Along Length of Scan Area

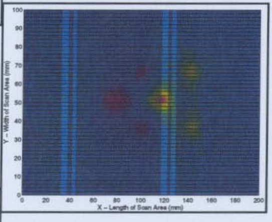
B_x View Along Width of Scan Area



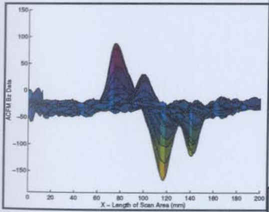
MACHINED COLONY 14



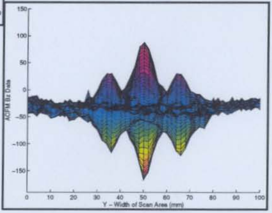
B_z Isometric



B_z Contour Plot

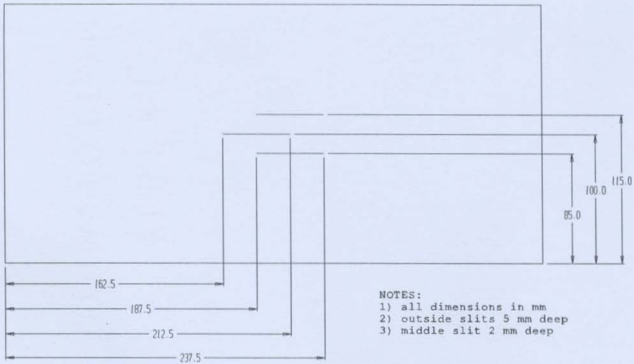


B_z View Along Length of Scan Area



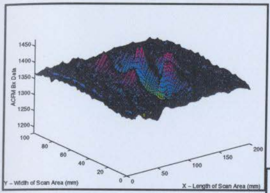
B_z View Along Width of Scan Area

Slitted Plate 15

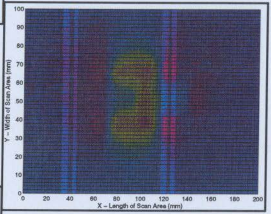


NOTES:
1) all dimensions in mm
2) outside slits 5 mm deep
3) middle slit 2 mm deep

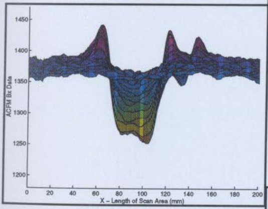
MACHINED COLONY 15



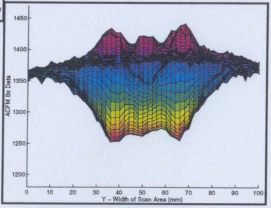
B_x Isometric



B_x Contour Plot

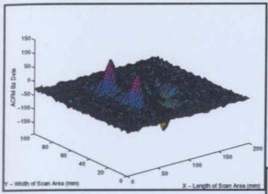


B_x View Along Length of Scan Area

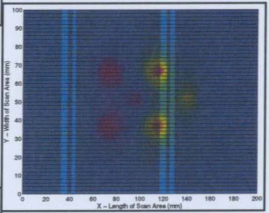


B_x View Along Width of Scan Area

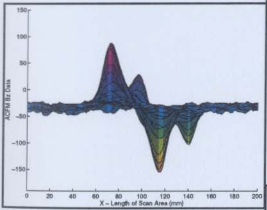
MACHINED COLONY 15



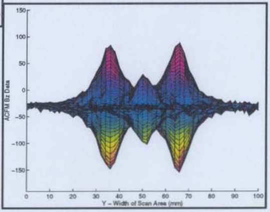
B_z Isometric



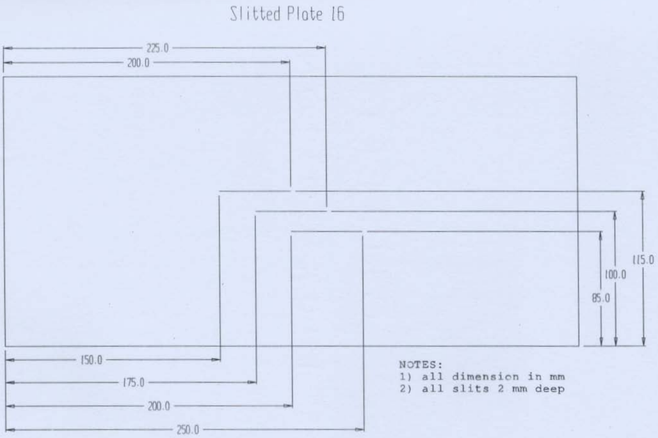
B_z Contour Plot

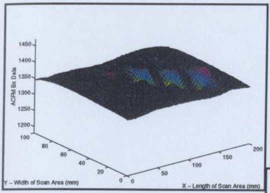


B_z View Along Length of Scan Area



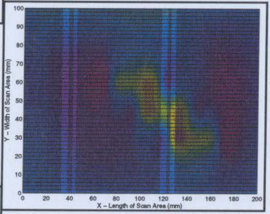
B_z View Along Width of Scan Area

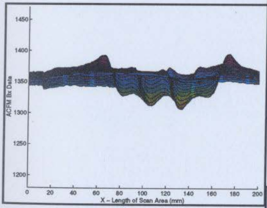

Slitted Plate 16
225.0
200.0
115.0
100.0
85.0
150.0
175.0
200.0
250.0
NOTES:
1) all dimension in mm
2) all slits 2 mm deep



B_x Isometric

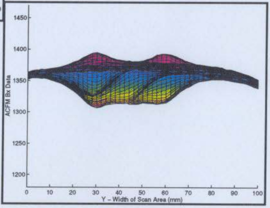
B_x Contour Plot



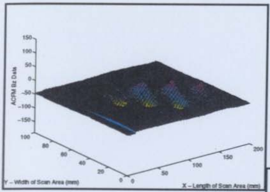


B_x View Along Length of Scan Area

B_x View Along Width of Scan Area

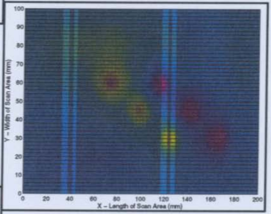


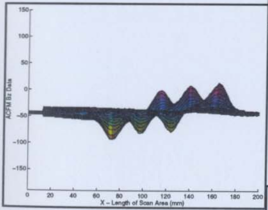
MACHINED COLONY 16



B_z Isometric

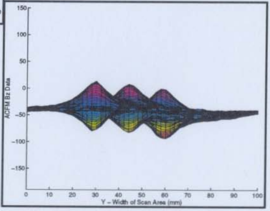
B_z Contour Plot



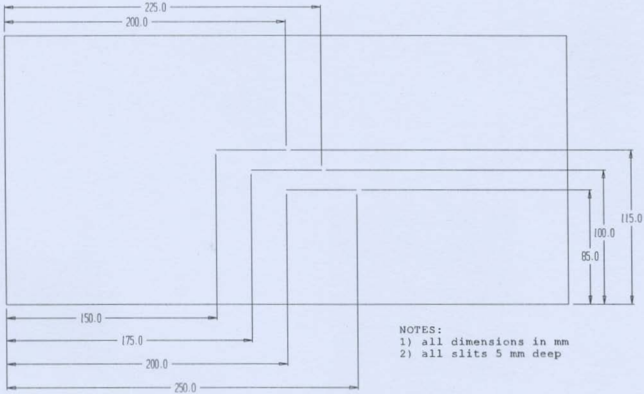


B_z View Along Length of Scan Area

B_z View Along Width of Scan Area

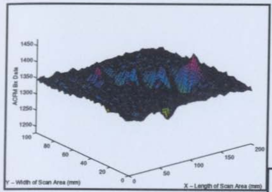


Slitted Plate 17

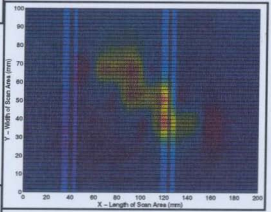


NOTES:
1) all dimensions in mm
2) all slits 5 mm deep

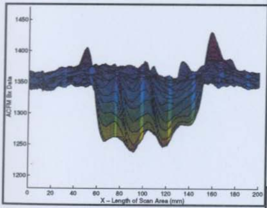
MACHINED COLONY 17



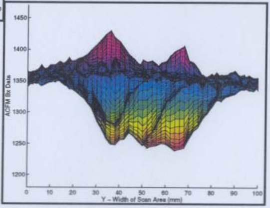
B_x Isometric



B_x Contour Plot

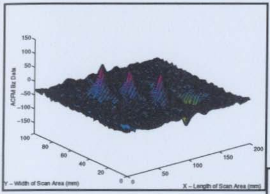


B_x View Along Length of Scan Area

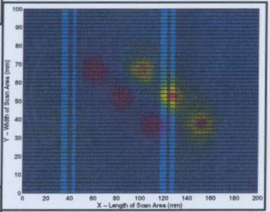


B_x View Along Width of Scan Area

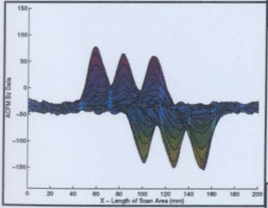
MACHINED COLONY 17



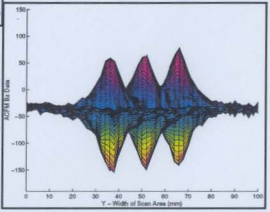
B_z Isometric



B_z Contour Plot

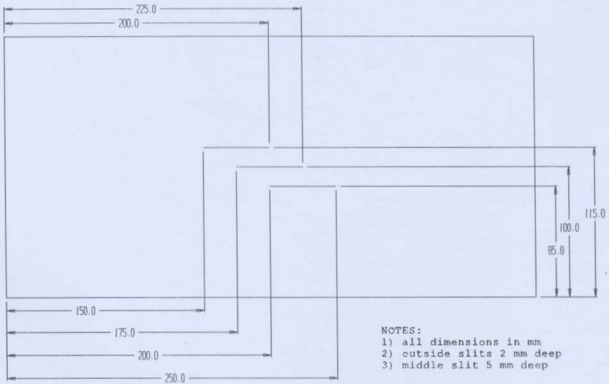


B_z View Along Length of Scan Area



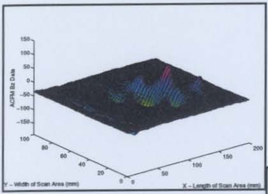
B_z View Along Width of Scan Area

Slitted Plate 18

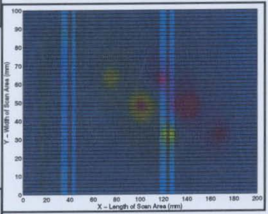


NOTES:
1) all dimensions in mm
2) outside slits 2 mm deep
3) middle slit 5 mm deep

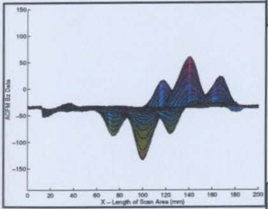
MACHINED COLONY 18



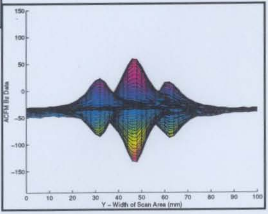
B_z Isometric



B_z Contour Plot

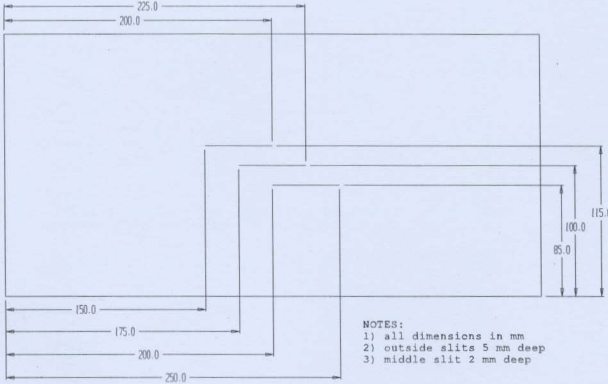


B_z View Along Length of Scan Area



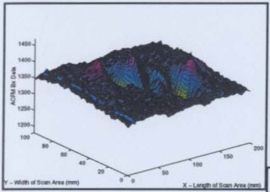
B_z View Along Width of Scan Area

Slitted Plate 19

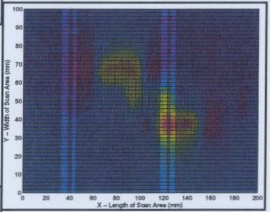


NOTES:
1) all dimensions in mm
2) outside slits 5 mm deep
3) middle slit 2 mm deep

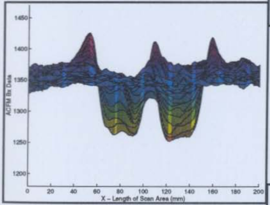
MACHINED COLONY 19



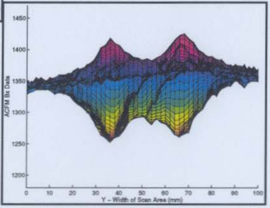
B_x Isometric



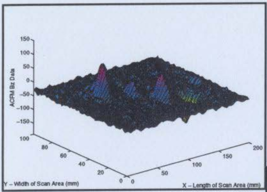
B_x Contour Plot



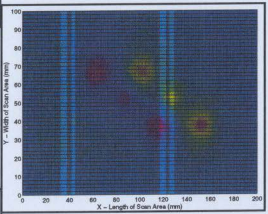
B_x View Along Length of Scan Area



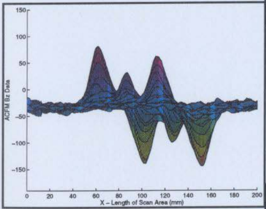
B_x View Along Width of Scan Area



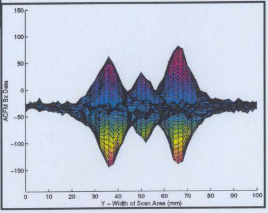
B_z Isometric



B_z Contour Plot

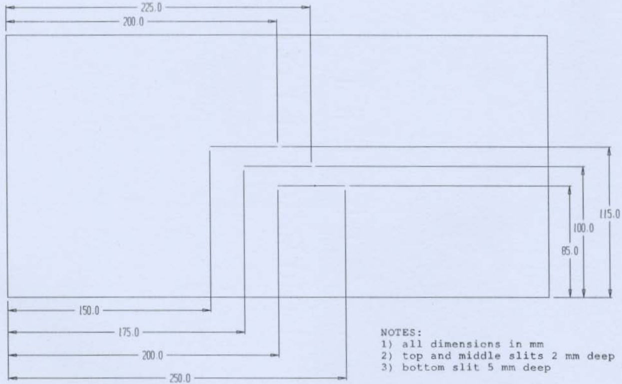


B_z View Along Length of Scan Area



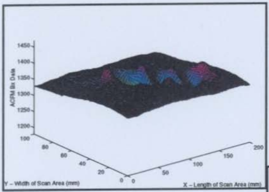
B_z View Along Width of Scan Area

Slitted Plate 20

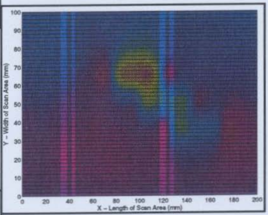


NOTES:
1) all dimensions in mm
2) top and middle slits 2 mm deep
3) bottom slit 5 mm deep

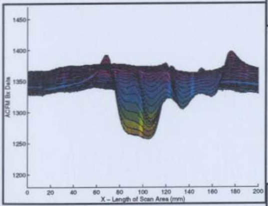
MACHINED COLONY 20



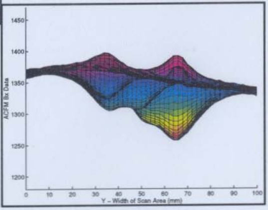
B_x Isometric



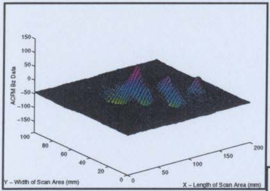
B_x Contour Plot



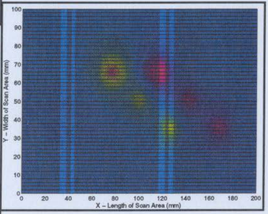
B_x View Along Length of Scan Area



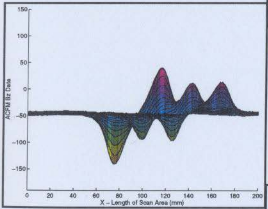
B_x View Along Width of Scan Area



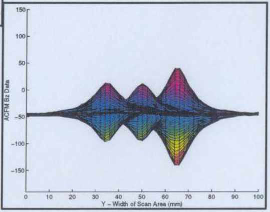
B_z Isometric



B_z Contour Plot

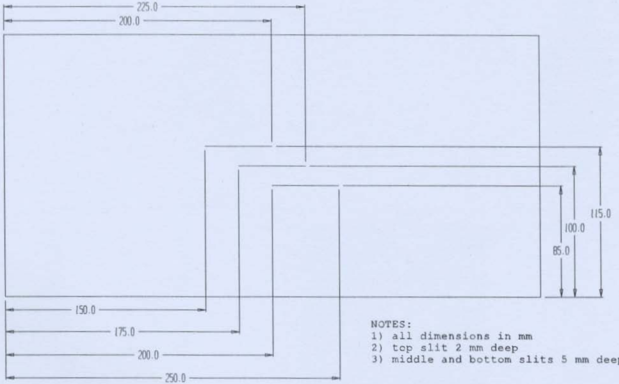


B_z View Along Length of Scan Area

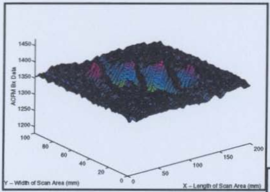


B_z View Along Width of Scan Area

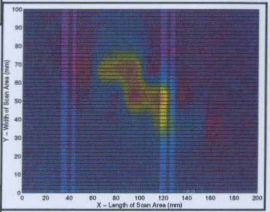
Slitted Plate 21



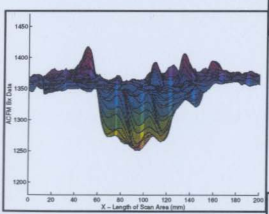
NOTES:
1) all dimensions in mm
2) top slit 2 mm deep
3) middle and bottom slits 5 mm deep



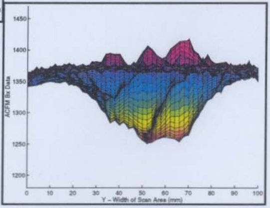
B_x Isometric



B_x Contour Plot

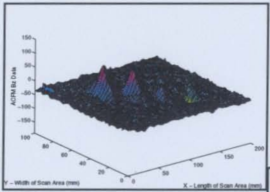


B_x View Along Length of Scan Area

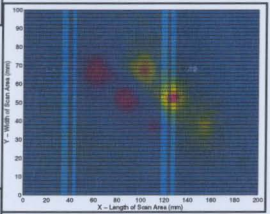


B_x View Along Width of Scan Area

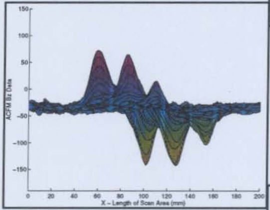
MACHINED COLONY 21



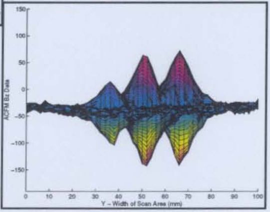
B_z Isometric



B_z Contour Plot



B_z View Along Length of Scan Area



B_z View Along Width of Scan Area

APPENDIX B

Sample of Scans from Natural SCC Colonies

This appendix contains samples of scans taken of natural SCC colonies that correspond to the scan area shown in Figure 4.11 (presented again below). The data files shown in Appendix A provide valuable insight into the influence of defect depth and position on AC field perturbations. The research project did not advance to the stage where similar conclusions could be drawn from natural colonies and presenting the scans from all of the colonies was not deemed necessary. Instead, representative samples of the B_Z scans are shown here to compare the signal to noise ratio and resolution of the pencil probe and microprobe.

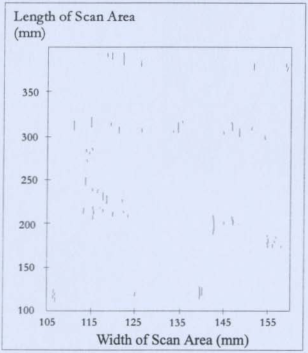
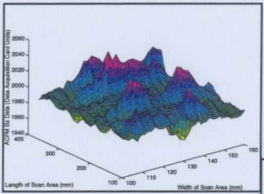
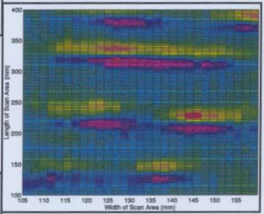


FIGURE 4.11: MPI results of a section of SCC affected pipe wall

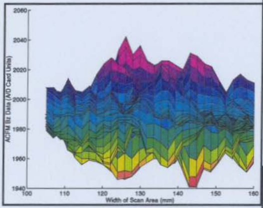
Pencil Probe Data



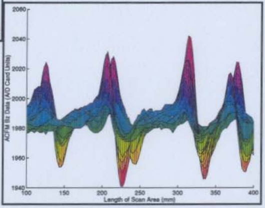
B_z Isometric



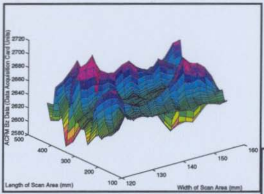
B_z Contour Plot



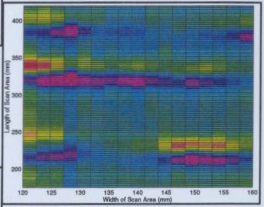
B_Z View Along Length of Scan Area



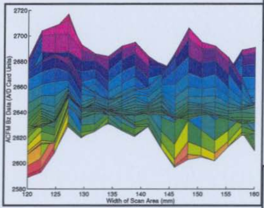
B_Z View Along Width of Scan Area



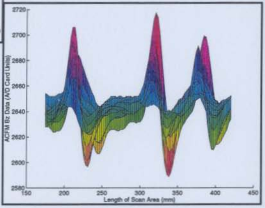
B_z Isometric



B_z Contour Plot



B_z View Along Length of Scan Area



B_z View Along Width of Scan Area

APPENDIX C

QUICKBASIC PROGRAMS AND MATLAB M-FILES

PROGRAM: SCAN.BAS

Purpose: Used to control the X-Y frame and data acquisition for the machined defect colonies

---

```
'****************************************************************
'*              XYCUTT routine for ACFM Probe                    *
'*                Main Program Body                              *
'****************************************************************

'$INCLUDE: 'CB.BI'          ' Mandatory INCLUDE file to access default
                            ' parameter values

DECLARE SUB pc21reset (address%)
DECLARE SUB PC21output (CMD$, address%)
DECLARE SUB PC21out (char$, address%)
DECLARE SUB PC21input (address%, Status$)
DECLARE SUB menu (ADD(), ax$())
DECLARE SUB xaxis (address%)
DECLARE SUB yaxis (address%)
DECLARE SUB scan (ADD())
DECLARE SUB convert (number)
DECLARE SUB gohome (address%)
DECLARE FUNCTION movedone (address%)


CLS
PRINT "    **************************************************"
PRINT "    *            ACFM SCAN PROGRAM            *"
PRINT "    *               JULY 1996             *"
PRINT "    *                  by             *"
PRINT "    *            L. Blair Carroll          *"
PRINT "    **************************************************"
PRINT
PRINT
PRINT "NOTE: This program was written for a 386 processor. Delays may have"
PRINT "    to be modified for faster machines"
PRINT
PRINT "              press <SPACE> to continue"
PRINT
PRINT
PRINT

DO UNTIL INKEY$ = " ": LOOP

' Dimension appropriate arrays
'     ADD() ..... address array for controller cards
DIM ADD(2)
DIM ax$(2)

ax$(1) = "x"
ax$(2) = "y"
ADD(1) = 768          ' x-axis
ADD(2) = 772          ' y-axis

' The following list of variables are use to keep track of addressing and
' the Control Byte
```

```
CONTROL = 96   ' the normal state of the control byte (only bites 5 and 6
        '   high)
STOPPED = 2   ' a mask for testing status bit 1 (is the motor moving)
CRASH = 4     ' a mask for handling control bit 2 (to signal the "Board
        '   Monitor" to time out or not)
LOADRDY = 4   ' a mask for testing status bit 2 (are move parameters
        '   loaded (load and go mode))
INTACK = 8    ' a mask for handling control bit 3 (to signal "Interrupt
        '   Acknowledged" or not)
MESSAGE = 8   ' a mask for testing status bit 3 (is a response waiting in
        '   output data buffer)
IDBREADY = 16  ' a mask for handling control bit 4 (to signal "Command Byte
        '   Ready in IDB or not") also a mask for status bit 4
        '   (is the Input Data Buffer ready for command byte)
FAULT = 32    ' a mask for handling control bit 5 (to signal "Restart
        '   Board Monitor" or not) also a mask for status bit
        '   5 (has the PC21 suffered a processing failure)
INTERRUPT = 64  ' a mask for handling control bit 6 (to signal "Clear
        '   Interrupts" or not) also a mask for testing status
        '   bit 6 (has the PC21 generetaed an interrupt)
RECEIVED = 128  ' a mask for handling control bit 7 (to signal "Message
        '   Received form ODB" or not)


'********************************************************************
' Set up emergency stop key = <exc>
PRINT "Defining emergency stop as <esc> ....";

KEY 15, CHR$(0) + CHR$(1)
ON KEY(15) GOSUB EmerStop
KEY(15) ON

PRINT "done."
FOR j = 1 TO 30000: NEXT j

'********************************************************************

' Resetting motors
FOR axis = 1 TO 2
    address% = ADD(axis)
    PRINT "Resetting "; ax$(axis); "-axis stepper motor indexer....";
    CALL pc21reset(address%) ' reset indexers
    CMD$ = " MN PZ X0 ST1 ": 'normal mode, zero position, energize
    CALL PC21output(CMD$, address%)
                            ' deenergize motor windings
    PRINT "done."
    FOR j = 1 TO 30000: NEXT j
NEXT axis

' Set initial position of motors to home
PRINT " Setting x-axis home position .... ";
address% = ADD(1)     ' x-axis
CMD$ = " PZ X0 "
CALL PC21output(CMD$, address%)
' send step counter reset
FOR i = 1 TO 100000: NEXT i
PRINT "done"
```

```
PRINT " Setting y-axis home position .... ";
address% = ADD(2)      ' y-axis
CMD$ = " PZ X0 "
CALL PC21output(CMD$, address%)
' send step counter reset
FOR i = 1 TO 100000: NEXT i
PRINT "done"

FOR j = 1 TO 40000: NEXT j

'*****************************************************************
' Provide the user with the option of repositioning the motors or
' running the preset scanning pattern of the probe with user specified
' velocity and displacement parameters

CALL menu(ADD(), ax$())
'*****************************************************************

END

'*****************************************************************
EmerStop:
    FOR axis = 1 TO 2
        address% = ADD(axis)
        CMD$ = " K "
        CALL PC21output(CMD$, address%)
    NEXT axis
    PRINT "!!!!!!!!!!  EMERGENCY STOP INITIATED  !!!!!!!!!"
RETURN
'*****************************************************************

SUB gohome (address%)
' determines position relative to home position and then sends axis home

SHARED Status$

DO: LOOP UNTIL movedone(address%)      ' wait until move is finished

FOR i = 1 TO 100000: NEXT i            ' pause

DO
    CMD$ = " X1 "                      ' ask for position
    CALL PC21output(CMD$, address%)
    CALL PC21input(address%, Status$)  ' retrieve position

    IF INSTR(Status$, "+") <> 0 THEN   ' check if positive
        sign$ = "-"                    ' set move negative
        ready = -1
    ELSEIF INSTR(Status$, "-") <> 0 THEN  ' check if negative
        sign$ = "+"                    ' set move positive
        ready = -1
    ELSE
        ready = 0
    END IF

LOOP UNTIL ready
```

```
disp$ = MID$(Status$, 3, 8)    ' trim displacement string

IF disp$ <> "00000000" THEN
     CMD$ = " MN D" + sign$ + disp$ + " V2 A10 G "
     CALL PC21output(CMD$, address%)        ' send instruction
     DO: LOOP UNTIL movedone(address%)       ' wait until move is finished
ELSE
     PRINT
     PRINT ".....the motor is already home."
     FOR i = 1 TO 10000: NEXT i
END IF

END SUB

SUB menu (ADD(), ax$())

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

CLS

PRINT "MENU OPTIONS"
PRINT "1) Reposition x-axis using the arrow keys"
PRINT "2) Reposition y-axis using the arrow keys"
PRINT "3) Select velocity and displacement parameters and begin scan"
PRINT "4) Reset Indexers"
PRINT "5) Set current x-axis position as home"
PRINT "6) Set current y-axis position as home"
PRINT "7) Send x-axis home"
PRINT "8) Send y-axis home"
PRINT "9) Quit"
INPUT "Select the number of the desired option..... ", opt


SELECT CASE opt

     CASE IS = 1
          CLS
          address% = ADD(1)
          PRINT "REPOSITION X-AXIS"
          PRINT "For rapid movement use the left and right arrow keys."
          PRINT "For small increments use <INS> and <HOME>"
          PRINT "To KILL motor use <SPACE>"
          PRINT "When Finished hit <RETURN>"
          CALL xaxis(address%)

     CASE IS = 2
          CLS
          address% = ADD(2)
          PRINT "REPOSITION Y-AXIS"
          PRINT "For rapid movement use the up and down arrow keys."
          PRINT "For small increments use <PGUP> and <PGDN>"
          PRINT "To KILL motor use <SPACE>"
          PRINT "When Finished hit <RETURN>"
          CALL yaxis(address%)

     CASE IS = 3
          CLS
```

```
    PRINT "PROBE SCAN ROUTINE INITIATED"
    PRINT
    PRINT
    INPUT "IS THE PROBE POSITIONED PROPERLY (y or n).... ", check$
    IF check$ = "y" THEN
        CALL scan(ADD())
    ELSEIF check$ = "n" THEN
        PRINT
        PRINT
        PRINT "Reposition probe before proceeding"
        PRINT "hit <space> to continue"
        DO UNTIL INKEY$ = " ": LOOP
        CALL menu(ADD(), ax$())
    END IF

CASE IS = 4
    CLS
    ' Resetting motors
    FOR axis = 1 TO 2
        address% = ADD(axis)
        PRINT "Resetting "; ax$(axis); "-axis stepper motor indexer....";
        CALL pc21reset(address%) ' reset indexers
        CMD$ = " MN PZ X0 ST1 " 'normal mode, zero position, energize
        CALL PC21output(CMD$, address%)
                    ' energize motor windings
        PRINT "done."
    NEXT axis

CASE IS = 5
    CLS
    ' Setting present x-axis position as home
    PRINT " Setting x-axis home position .... ";
    address% = ADD(1)       ' x-axis
    CMD$ = " PZ X0 "
    CALL PC21output(CMD$, address%)
                    ' send step counter reset
    FOR i = 1 TO 100000: NEXT i
    PRINT "done"

CASE IS = 6
    CLS
    ' Setting present y-axis position as home
    PRINT " Setting y-axis home position .... ";
    address% = ADD(2)       ' y-axis
    CMD$ = " PZ X0 "
    CALL PC21output(CMD$, address%)
                    ' send step counter reset
    FOR i = 1 TO 100000: NEXT i
    PRINT "done"

CASE IS = 7
    CLS
    ' Sending x-axis home
    address% = ADD(1)           ' x-axis
    PRINT "Sending x-axis to home position."
    CALL gohome(address%)
    FOR i = 1 TO 100000: NEXT i
```

```
    CASE IS = 8
        CLS
        ' Sending y-axis home
        address% = ADD(2)            ' y-axis
        PRINT "Sending y-axis to home position."
        CALL gohome(address%)
        FOR i = 1 TO 100000: NEXT i


    CASE IS = 9
        DO
        PRINT
        PRINT
        PRINT
        PRINT "MAKE SURE PROBE HAS BEEN RETURNED TO HOME POSITION!!!!!!"
        INPUT "DO YOU REALLY WISH TO QUIT (y OR n)? ", quit$
        IF quit$ = "y" THEN
            END
        ELSE quit$ = "n"
            CLS
            EXIT DO
        END IF
        LOOP

    CASE ELSE
        CLS

END SELECT

CALL menu(ADD(), ax$())

END SUB

FUNCTION movedone (address%)
' check to see if indexer has stopped sending pulses to the motor
' if the motor has stopped a logical true (-1) is returned
' if not a logical false (0) is returned

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

CMD$ = " R "        ' request move status
CALL PC21output(CMD$, address%)
FOR i = 1 TO 1000: NEXT i

CALL PC21input(address%, Status$)
Status$ = UCASE$(Status$)

IF (INSTR(Status$, "R")) = 0 AND (INSTR(Status$, "S")) = 0 THEN
    movedone = 0
ELSE
    movedone = -1
END IF

END FUNCTION

SUB PC21input (address%, Status$)
' retrieves response of the indexer of the speciifiied axis and forms a
```

```
' string

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

Status$ = " "  ' clear string

DO
    BYTE = INP(address% + 1)

    IF (MESSAGE AND NOT BYTE) THEN
        EXIT DO
    ELSE
        answer% = INP(address%)
        OUT address% + 1, (CONTROL OR RECEIVED)
        DO
            BYTE = INP(address% + 1)
        LOOP WHILE (MESSAGE AND BYTE)
        OUT address% + 1, (CONTOL AND NOT RECEIVED)
        char$ = CHR$(answer%)
        Status$ = Status$ + char$

        FOR i = 1 TO 10000: NEXT i
    END IF
LOOP UNTIL char$ = CHR$(13)

END SUB

SUB PC21out (char$, address%)
'*****************************************************************
'*              PC21 Output: Stage 2                 *
'*****************************************************************

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

DO
BYTE = INP(address% + 1)
LOOP WHILE IDBREADY AND NOT BYTE

OUT address%, ASC(char$)
OUT address% + 1, (CONTROL OR IDBREADY)

DO
BYTE = INP(address% + 1)
LOOP WHILE IDBREADY AND BYTE

OUT address% + 1, (CONTROL AND NOT IDBREADY)

END SUB

SUB PC21output (CMD$, address%)
'*****************************************************************
'*              PC21 Output: Stage 1                 *
'*****************************************************************

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED
```

```
FOR i = 1 TO LEN(CMD$)
    char$ = MID$(CMD$, i, 1)      ' isolate each character in command
                        ' string to send to PC21
    CALL PC21out(char$, address%)
                        ' send character to PC21
NEXT i                  ' repeat for next character

char$ = CHR$(13)                ' send carriage return to signal
                        ' end of command
CALL PC21out(char$, address%)

END SUB

SUB pc21reset (address%)
' the following subprogram allows the "Board Monitor" to timeout, reset
' the PC21, and then the restart the Board Monitor Timer

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

OUT address% + 1, (CONTROL OR CRASH)          ' control bit 2 high
OUT address% + 1, (CONTROL AND NOT CRASH)     ' control bit 2 low
FOR Y = 1 TO 10000: NEXT                      ' wait for BMA
OUT address% + 1, (CONTROL AND NOT FAULT)     ' control bit 5 low
OUT address% + 1, (CONTROL OR FAULT)          ' control bit 5 high
FOR i = 1 TO 10000: NEXT                      ' pause

END SUB

SUB scan (ADD())

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED, CurCount&, CurIndex&

CONST BoardNum = 0          ' Board number
CONST NumPoints = 1050      ' Number of data points to collect

DIM ADData%(NumPoints)  'dimension an array to hold the input values

' Set up the appropriate motion parameters for the x and y axis
' of the scan pattern

INPUT "Desired x-axis travel (mm).... ", xtrav
INPUT "Total desired y-axis travel (mm).... ", ytrav
INPUT "Separation of each scan pass [enter 0 for single pass] (mm).... ", yspace
PRINT
PRINT "Allowable velocity range: 0.05 to 50 mm/s"
INPUT "Desired scan velocity (mm/s).... ", vel

IF vel < .05 OR vel > 50 THEN            ' check velocity
    PRINT "VELOCITY VALUE OUT OF RANGE"
    FOR i = 1 TO 50000: NEXT i
    CALL scan(ADD())
ELSE
END IF

PRINT
PRINT "Allowable acceleration range: 0.01 to 200 mm/s^2"
```

```
INPUT "Desired acceleration (mm/s^2).... ", acc

IF acc < .01 OR acc > 200 THEN              ' check acceleration
     PRINT "ACCELERATION OUT OF RANGE"
     FOR i = 1 TO 50000: NEXT i
     CALL scan(ADD())
ELSE
END IF

PRINT
INPUT "Do you wish to save the data to a file? ", fle$
IF fle$ = "y" THEN
     INPUT "Provide the data file name ('.DAT' automatically added). ", Basename$
     INPUT "Provide the date. ", dte$
     INPUT "Provide the time. ", tme$
     INPUT "Provide the Plate Number. ", pltno$
ELSE
END IF
PRINT


INPUT "Have you input the parameters properly (y or n).... ", check$

IF check$ = "y" THEN
     IF fle$ = "y" THEN
          OPEN Basename$ + ".DAT" FOR APPEND AS #1
          Nm$ = "ACFM Data File"
          WRITE #1, Nm$
          WRITE #1, Basename$
          WRITE #1, dte$
          WRITE #1, tme$
          WRITE #1, pltno$
          WRITE #1,
     ELSE
     END IF

     PRINT
     PRINT "Then hit <SPACE> to begin scan"
     DO UNTIL INKEY$ = " ": LOOP
     CLS
     PRINT "SCANNING"
ELSE
     CLS
     CALL pc21reset(address%)
     CALL scan(ADD())
END IF

' Convert mm to number of steps
xmove = INT(xtrav * 1968.50394#)           ' change to microsteps
ymove = INT(yspace * 1968.50395#)          ' change to microsteps

IF yspace = 0 THEN                         ' no of scan passes
     passes = 1
ELSE
     passes = INT(ytrav / yspace) + 1
END IF

velocity = (.196850394# * vel) / 2.392     ' change to rev/s
```

```
accel = (.196850394# * acc)          ' change to rev/s^2

dispx$ = LTRIM$(STR$(xmove))  ' change numbers to strings for output to
dispy$ = LTRIM$(STR$(ymove))  ' indexers

IF velocity < 1 THEN          ' trim velocity string to indexer format
     vel$ = LEFT$(LTRIM$(STR$(velocity)), 4)
ELSE
     vel$ = LEFT$(LTRIM$(STR$(velocity)), 5)
END IF

accel$ = LEFT$(LTRIM$(STR$(accel)), 4)
                 ' trim acceleration string to indexer format

' Send velocity and acceleration to indexers
FOR i = 1 TO 2
     address% = ADD(i)
     CMD$ = " MN V" + vel$ + " A" + accel$ + " "
     CALL PC21output(CMD$, address%)
NEXT i


' Move the probe and scan
FOR j = 1 TO passes

'$STATIC

' Declare UL Revision Level

   ULStat% = cbDeclareRevision(CURRENTREVNUM)

   ' Initiate error handling
   '  activating error handling will trap errors like
   '  bad channel numbers and non-configured conditions.
   '  Parameters:
   '    PRINTALL    all warnings and errors encountered will be printed
   '    DONTSTOP    :if an error is encountered, the program will not stop,
   '                errors must be handled locally


   ULStat% = cbErrHandling%(PRINTALL, DONTSTOP)
   IF ULStat% <> 0 THEN STOP

   ' If cbErrHandling% is set for STOPALL or STOPFATAL during the program
   ' design stage, Quick Basic will be unloaded when an error is encountered.
   ' We suggest trapping errors locally until the program is ready for compiling
   ' to avoid losing unsaved data during program design.  This can be done by
   ' setting cbErrHandling options as above and checking the value of ULStat%
   ' after a call to the library. If it is not equal to 0, an error has occurred.

  'set up the display screen
   CLS
   PRINT "Samples are displayed for user to visually verify data is being collected."
   PRINT
   Xs% = POS(0)
   Ys% = CSRLIN
```

```
'Collect the values with cbAInScan%() in BACKGROUND mode, CONTINUOUSLY
' Parameters:
'   BoardNum    :the number used by CB.CFG to describe this board
'   LowChan%    :the first channel of the scan
'   HighChan%   :the last channel of the scan
'   Count&      :the total number of A/D samples to collect
'   Rate&       :sample rate in samples per second
'   Gain%       :the gain for the board
'   ADData%     :the array for the collected data values
'   Options%    :data collection options

  LowChan% = 0
  HighChan% = 0
  Count& = NumPoints          ' total number of data points to collect
  Rate& = 100                 ' sampling rate (samples per second)
  Options% = BACKGROUND + CONTINUOUS + CONVERTDATA + SINGLEIO
                              ' collect data continuously in background
                              ' return data as 12-bit values
  Gain% = BIP10VOLTS          ' set the gain


  ULStat% = cbAInScan%(BoardNum, LowChan%, HighChan%, Count&, Rate&, Gain%, ADData%(0), Options%)
  IF ULStat% = 84 THEN
    PRINT "The CONVERT option cannot be used with 16 bit convertors. Set Options% to NOCONVERTDATA."
    STOP 'Change Options% above to NOCONVERTDATA (Options% = 0)
  END IF
  IF ULStat% <> 0 THEN STOP

  X% = CSRLIN
  Y% = POS(0)

 address% = ADD(1)
 CMD$ = " MN D-" + dispx$ + " G "
 CALL PC21output(CMD$, address%)

 'during the BACKGROUND operation, check the status, print the values

 DO                     ' wait until move is finished

     '  CurCount&   :current number of samples collected
     '  CurIndex&   :index to the data buffer pointing to the last value transferred

     ULStat% = cbGetStatus%(BoardNum, Status%, CurCount&, CurIndex&)
     IF ULStat% <> 0 THEN STOP

     IF Status% = RUNNING THEN
          LOCATE X%, Y%
          PRINT USING "Data Point: ###### "; CurIndex&;
          IF CurIndex& >= 0 THEN
               PRINT USING "   Value: ######"; ADData%(CurIndex&);
          END IF
     END IF

 LOOP WHILE Status% = RUNNING AND NOT movedone(address%)
 PRINT

 'the BACKGROUND operation must be explicitly stopped
 ' Parameters:
```

```
'   BoardNum :the number used by CB.CFG to describe this board

  ULStat% = cbStopBackground%(BoardNum)
  IF ULStat% <> 0 THEN STOP

PRINT
PRINT "Scan pass "; j; "of "; passes; "complete."

FOR k = 1 TO 10000: NEXT k           ' pause


CMD$ = " MN D+" + dispx$ + " G "
CALL PC21output(CMD$, address%)
DO: LOOP UNTIL movedone(address%)     ' wait until move is finished
FOR k = 1 TO 10000: NEXT k           ' pause

address% = ADD(2)
CMD$ = " MN D-" + dispy$ + " G "
CALL PC21output(CMD$, address%)
DO: LOOP UNTIL movedone(address%)     ' wait until move is finished
FOR k = 1 TO 10000: NEXT k           ' pause

' Write data to data file
IF fle$ = "y" THEN
      WRITE #1, -100   'flag marker for scan separation

      FOR m = 1 TO NumPoints
           WRITE #1, ADData%(m)
      NEXT m
ELSE
END IF

NEXT j

PRINT
PRINT
PRINT "Press <SPACE> to continue"
DO UNTIL INKEY$ = " ": LOOP

' close data file
CLOSE #1

CALL gohome(772)

END SUB

SUB xaxis (address%)

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

' Set up keys
lft$ = CHR$(0) + CHR$(75)
rght$ = CHR$(0) + CHR$(77)
inst$ = CHR$(0) + CHR$(82)
hm$ = CHR$(0) + CHR$(71)
spacebar$ = " "
rtn$ = CHR$(13)
```

```
done = 0

DO UNTIL done = 1
    kbd$ = INKEY$
    SELECT CASE kbd$
        CASE IS = lft$
            ' move left
            CMD$ = " MC H- V3 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i


        CASE IS = rght$
            ' move right
            CMD$ = " MC H+ V3 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i

        CASE IS = inst$
            ' move slowly left
            CMD$ = " MC H- V1 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i

        CASE IS = hm$
            ' move slowly right
            CMD$ = " MC H+ V1 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i

        CASE IS = spacebar$
            CMD$ = " K "
            CALL PC21output(CMD$, address%)
            done = 1

        CASE IS = rtrn$
            CMD$ = " S "
            CALL PC21output(CMD$, address%)
            done = 1

        CASE ELSE
            CMD$ = " S "
            CALL PC21output(CMD$, address%)
            done = 0
    END SELECT

LOOP

END SUB

SUB yaxis (address%)

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED
```

```
address% = 772

' Set up keys
up$ = CHR$(0) + CHR$(72)
dwn$ = CHR$(0) + CHR$(80)
pgup$ = CHR$(0) + CHR$(73)
pgdwn$ = CHR$(0) + CHR$(81)
spacebar$ = " "
rtrn$ = CHR$(13)

done = 0

DO UNTIL done = 1
    kbd$ = INKEY$
    SELECT CASE kbd$
        CASE IS = up$
            ' move in
            CMD$ = " MC H+ V3 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i

        CASE IS = dwn$
            ' move out
            CMD$ = " MC H- V3 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i

        CASE IS = pgup$
            ' move slowly in
            CMD$ = " MC H+ V1 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i

        CASE IS = pgdwn$
            ' move slowly out
            CMD$ = " MC H- V1 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i

        CASE IS = spacebar$
            CMD$ = " K "
            CALL PC21output(CMD$, address%)
            done = 1

        CASE IS = rtrn$
            CMD$ = " S "
            CALL PC21output(CMD$, address%)
            done = 1

        CASE ELSE
            CMD$ = " S "
            CALL PC21output(CMD$, address%)
            done = 0
    END SELECT
```

```
LOOP

END SUB
```

PROGRAM: PIPESCAN.BAS

Purpose: Used to control the X-Y frame and data acquisition for the pipe wall section scans

---

```
'****************************************************************
'*            XYCUTT routine for ACFM Probe                    *
'*               Main Program Body                        *
'****************************************************************

'$INCLUDE: 'CB.BI'        ' Mandatory INCLUDE file to access default
                          ' parameter values

DECLARE SUB pc21reset (address%)
DECLARE SUB PC21output (CMD$, address%)
DECLARE SUB PC21out (char$, address%)
DECLARE SUB PC21input (address%, Status$)
DECLARE SUB menu (ADD(), ax$())
DECLARE SUB xaxis (address%)
DECLARE SUB yaxis (address%)
DECLARE SUB scan (ADD())
DECLARE SUB convert (number)
DECLARE SUB gohome (address%)
DECLARE FUNCTION movedone (address%)


CLS
PRINT "     **************************************************"
PRINT "     *          ACFM SCAN PROGRAM              *"
PRINT "     *             JULY 1996             *"
PRINT "     *                by              *"
PRINT "     *           L. Blair Carroll           *"
PRINT "     **************************************************"
PRINT
PRINT
PRINT "NOTE: This program was written for a 386 processor. Delays may have"
PRINT "     to be modified for faster machines"
PRINT
PRINT "              press <SPACE> to continue"
PRINT
PRINT
PRINT

DO UNTIL INKEY$ = " ": LOOP

' Dimension appropriate arrays
'     ADD() ..... address array for controller cards
DIM ADD(2)
DIM ax$(2)

ax$(1) = "x"
ax$(2) = "y"
ADD(1) = 768         ' x-axis
```

```
ADD(2) = 772          ' y-axis

' The following list of variables are use to keep track of addressing and
' the Control Byte

CONTROL = 96   ' the normal state of the control byte (only bites 5 and 6
          '      high)
STOPPED = 2    ' a mask for testing status bit 1 (is the motor moving)
CRASH = 4      ' a mask for handling control bit 2 (to signal the "Board
          '      Monitor" to time out or not)
LOADRDY = 4    ' a mask for testing status bit 2 (are move parameters
          '      loaded (load and go mode))
INTACK = 8     ' a mask for handling control bit 3 (to signal "Interrupt
          '      Acknowledged" or not)
MESSAGE = 8    ' a mask for testing status bit 3 (is a response waiting in
          '      output data buffer)
IDBREADY = 16  ' a mask for handling control bit 4 (to signal "Command Byte
          '      Ready in IDB or not") also a mask for status bit 4
          '      (is the Input Data Buffer ready for command byte)
FAULT = 32     ' a mask for handling control bit 5 (to signal "Restart
          '      Board Monitor" or not) also a mask for status bit
          '      5 (has the PC21 suffered a processing failure)
INTERRUPT = 64 ' a mask for handling control bit 6 (to signal "Clear
          '      Interrupts" or not) also a mask for testing status
          '      bit 6 (has the PC21 generetaed an interrupt)
RECEIVED = 128 ' a mask for handling control bit 7 (to signal "Message
          '      Received form ODB" or not)


'*******************************************************************
' Set up emergency stop key = <exc>
PRINT "Defining emergency stop as <esc> ....";

KEY 15, CHR$(0) + CHR$(1)
ON KEY(15) GOSUB EmerStop
KEY(15) ON

PRINT "done."
FOR j = 1 TO 30000: NEXT j

'*******************************************************************

' Resetting motors
FOR axis = 1 TO 2
    address% = ADD(axis)
    PRINT "Resetting "; ax$(axis); "-axis stepper motor indexer....";
    CALL pc21reset(address%) ' reset indexers
    CMD$ = " MN PZ X0 ST1 ": 'normal mode, zero position, energize
    CALL PC21output(CMD$, address%)
                              ' deenergize motor windings
    PRINT "done."
    FOR j = 1 TO 30000: NEXT j
NEXT axis

' Set initial position of motors to home
PRINT " Setting x-axis home position .... ";
address% = ADD(1)     ' x-axis
CMD$ = " PZ X0 "
```

```
CALL PC21output(CMD$, address%)
' send step counter reset
FOR i = 1 TO 100000: NEXT i
PRINT "done"

PRINT " Setting y-axis home position .... ";
address% = ADD(2)      ' y-axis
CMD$ = " PZ X0 "
CALL PC21output(CMD$, address%)
' send step counter reset
FOR i = 1 TO 100000: NEXT i
PRINT "done"

FOR j = 1 TO 40000: NEXT j

'*****************************************************************
' Provide the user with the option of repositioning the motors or
' running the preset scanning pattern of the probe with user specified
' velocity and displacement parameters

CALL menu(ADD(), ax$())
'*****************************************************************

END


'*****************************************************************
EmerStop:
    FOR axis = 1 TO 2
        address% = ADD(axis)
        CMD$ = " K "
        CALL PC21output(CMD$, address%)
    NEXT axis
    PRINT "!!!!!!!!!  EMERGENCY STOP INITIATED  !!!!!!!!"
RETURN
'*****************************************************************

SUB gohome (address%)
' determines position relative to home position and then sends axis home

SHARED Status$

DO: LOOP UNTIL movedone(address%)     ' wait until move is finished

FOR i = 1 TO 100000: NEXT i           ' pause

DO
    CMD$ = " X1 "                       ' ask for position
    CALL PC21output(CMD$, address%)
    CALL PC21input(address%, Status$)   ' retrieve position

    IF INSTR(Status$, "+") <> 0 THEN     ' check if positive
        sign$ = "-"                     ' set move negative
        ready = -1
    ELSEIF INSTR(Status$, "-") <> 0 THEN ' check if negative
        sign$ = "+"                     ' set move positive
        ready = -1
    ELSE
```

```
        ready = 0
    END IF

LOOP UNTIL ready

disp$ = MID$(Status$, 3, 8)    ' trim displacement string

IF disp$ <> "00000000" THEN
    CMD$ = " MN D" + sign$ + disp$ + " V2 A10 G "
    CALL PC21output(CMD$, address%)        ' send instruction
    DO: LOOP UNTIL movedone(address%)      ' wait until move is finished
ELSE
    PRINT
    PRINT ".....the motor is already home."
    FOR i = 1 TO 10000: NEXT i
END IF

END SUB

SUB menu (ADD(), ax$())

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

CLS

PRINT "MENU OPTIONS"
PRINT "1) Reposition x-axis using the arrow keys"
PRINT "2) Reposition y-axis using the arrow keys"
PRINT "3) Select velocity and displacement parameters and begin scan"
PRINT "4) Reset Indexers"
PRINT "5) Set current x-axis position as home"
PRINT "6) Set current y-axis position as home"
PRINT "7) Send x-axis home"
PRINT "8) Send y-axis home"
PRINT "9) Quit"
INPUT "Select the number of the desired option..... ", opt


SELECT CASE opt

    CASE IS = 1
        CLS
        address% = ADD(1)
        PRINT "REPOSITION X-AXIS"
        PRINT "For rapid movement use the left and right arrow keys."
        PRINT "For small increments use <INS> and <HOME>"
        PRINT "To KILL motor use <SPACE>"
        PRINT "When Finished hit <RETURN>"
        CALL xaxis(address%)

    CASE IS = 2
        CLS
        address% = ADD(2)
        PRINT "REPOSITION Y-AXIS"
        PRINT "For rapid movement use the up and down arrow keys."
        PRINT "For small increments use <PGUP> and <PGDN>"
        PRINT "To KILL motor use <SPACE>"
```

```
    PRINT "When Finished hit <RETURN>"
    CALL yaxis(address%)

CASE IS = 3
    CLS
    PRINT "PROBE SCAN ROUTINE INITIATED"
    PRINT
    PRINT
    INPUT "IS THE PROBE POSITIONED PROPERLY (y or n).... ", check$
    IF check$ = "y" THEN

        CALL scan(ADD())
    ELSEIF check$ = "n" THEN
        PRINT
        PRINT
        PRINT "Reposition probe before proceeding"
        PRINT "hit <space> to continue"
        DO UNTIL INKEY$ = " ": LOOP
        CALL menu(ADD(), ax$())
    END IF

CASE IS = 4
    CLS
    ' Resetting motors
    FOR axis = 1 TO 2
        address% = ADD(axis)
        PRINT "Resetting "; ax$(axis); "-axis stepper motor indexer....";
        CALL pc21reset(address%) ' reset indexers
        CMD$ = " MN PZ X0 ST1 ": 'normal mode, zero position, energize
        CALL PC21output(CMD$, address%)
                        ' energize motor windings
        PRINT "done."
    NEXT axis

CASE IS = 5
    CLS
    ' Setting present x-axis position as home
    PRINT " Setting x-axis home position .... ";
    address% = ADD(1)      ' x-axis
    CMD$ = " PZ X0 "
    CALL PC21output(CMD$, address%)
                    ' send step counter reset
    FOR i = 1 TO 100000: NEXT i
    PRINT "done"

CASE IS = 6
    CLS
    ' Setting present y-axis position as home
    PRINT " Setting y-axis home position .... ";
    address% = ADD(2)      ' y-axis
    CMD$ = " PZ X0 "
    CALL PC21output(CMD$, address%)
                    ' send step counter reset
    FOR i = 1 TO 100000: NEXT i
    PRINT "done"

CASE IS = 7
    CLS
```

```
        ' Sending x-axis home
        address% = ADD(1)          ' x-axis
        PRINT "Sending x-axis to home position."
        CALL gohome(address%)
        FOR i = 1 TO 100000: NEXT i

    CASE IS = 8
        CLS
        ' Sending y-axis home
        address% = ADD(2)          ' y-axis
        PRINT "Sending y-axis to home position."
        CALL gohome(address%)
        FOR i = 1 TO 100000: NEXT i


    CASE IS = 9
        DO
        PRINT
        PRINT
        PRINT
        PRINT "MAKE SURE PROBE HAS BEEN RETURNED TO HOME POSITION!!!!!!"
        INPUT "DO YOU REALLY WISH TO QUIT (y OR n)? ", quit$
        IF quit$ = "y" THEN
            END
        ELSE quit$ = "n"
            CLS
            EXIT DO
        END IF
        LOOP

    CASE ELSE
        CLS

END SELECT

CALL menu(ADD(), ax$())

END SUB

FUNCTION movedone (address%)
' check to see if indexer has stopped sending pulses to the motor
' if the motor has stopped a logical true (-1) is returned
' if not a logical false (0) is returned

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

CMD$ = " R "          ' request move status
CALL PC21output(CMD$, address%)
FOR i = 1 TO 1000: NEXT i

CALL PC21input(address%, Status$)
Status$ = UCASE$(Status$)

IF (INSTR(Status$, "R")) = 0 AND (INSTR(Status$, "S")) = 0 THEN
    movedone = 0
ELSE
    movedone = -1
```

```
END IF

END FUNCTION

SUB PC21input (address%, Status$)
' retrieves response of the indexer of the speciified axis and forms a
' string

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

Status$ = " "  ' clear string

DO
    BYTE = INP(address% + 1)

    IF (MESSAGE AND NOT BYTE) THEN
        EXIT DO
    ELSE
        answer% = INP(address%)
        OUT address% + 1, (CONTROL OR RECEIVED)
        DO
            BYTE = INP(address% + 1)
        LOOP WHILE (MESSAGE AND BYTE)
        OUT address% + 1, (CONTOL AND NOT RECEIVED)
        char$ = CHR$(answer%)
        Status$ = Status$ + char$

        FOR i = 1 TO 10000: NEXT i
    END IF
LOOP UNTIL char$ = CHR$(13)

END SUB

SUB PC21out (char$, address%)
'***************************************************************
'*              PC21 Output: Stage 2              *
'***************************************************************

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

DO
BYTE = INP(address% + 1)
LOOP WHILE IDBREADY AND NOT BYTE

OUT address%, ASC(char$)
OUT address% + 1, (CONTROL OR IDBREADY)

DO
BYTE = INP(address% + 1)
LOOP WHILE IDBREADY AND BYTE

OUT address% + 1, (CONTROL AND NOT IDBREADY)

END SUB

SUB PC21output (CMD$, address%)
```

```
'***************************************************************
'*          PC21 Output: Stage I          *
'***************************************************************

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

FOR i = 1 TO LEN(CMD$)
    char$ = MID$(CMD$, i, 1)      ' isolate each character in command
                        ' string to send to PC21
    CALL PC21out(char$, address%)
                        ' send character to PC21
NEXT i                  ' repeat for next character

char$ = CHR$(13)              ' send carriage return to signal
                        ' end of command
CALL PC21out(char$, address%)

END SUB

SUB pc21reset (address%)
' the following subprogram allows the "Board Monitor" to timeout, reset
' the PC21, and then the restart the Board Monitor Timer

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

OUT address% + 1, (CONTROL OR CRASH)          ' control bit 2 high
OUT address% + 1, (CONTROL AND NOT CRASH)     ' control bit 2 low
FOR Y = 1 TO 10000: NEXT                      ' wait for BMA
OUT address% + 1, (CONTROL AND NOT FAULT)     ' control bit 5 low
OUT address% + 1, (CONTROL OR FAULT)          ' control bit 5 high
FOR i = 1 TO 10000: NEXT                      ' pause

END SUB

SUB scan (ADD())

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED, CurCount&, CurIndex&

CONST BoardNum = 0        ' Board number
INPUT "Set the necessary scan length array size.... ", NumPoints
                        ' Number of data points to collect

DIM ADData%(NumPoints)   'dimension an array to hold the input values

' Set up the appropriate motion parameters for the x and y axis
' of the scan pattern

INPUT "Desired x-axis travel (mm).... ", xtrav
INPUT "Total desired y-axis travel (mm).... ", ytrav
INPUT "Separation of each scan pass [enter 0 for single pass] (mm).... ", yspace
PRINT
PRINT "Allowable velocity range: 0.05 to 50 mm/s"
INPUT "Desired scan velocity (mm/s).... ", vel

IF vel < .05 OR vel > 50 THEN            ' check velocity
```

```
     PRINT "VELOCITY VALUE OUT OF RANGE"
     FOR i = 1 TO 50000: NEXT i
     CALL scan(ADD())
ELSE
END IF

PRINT
PRINT "Allowable acceleration range: 0.01 to 200 mm/s^2"
INPUT "Desired acceleration (mm/s^2).... ", acc

IF acc < .01 OR acc > 200 THEN                ' check acceleration
     PRINT "ACCELERATION OUT OF RANGE"
     FOR i = 1 TO 50000: NEXT i
     CALL scan(ADD())
ELSE
END IF

PRINT
INPUT "Do you wish to save the data to a file? ", fle$
IF fle$ = "y" THEN
     INPUT "Provide the data file name ('.DAT' automatically added). ", Basename$
     INPUT "Provide the date. ", dte$
     INPUT "Provide the time. ", tme$
     INPUT "Provide the Plate Number. ", pltno$
ELSE
END IF
PRINT


INPUT "Have you input the parameters properly (y or n).... ", check$

IF check$ = "y" THEN
     IF fle$ = "y" THEN
          OPEN Basename$ + ".DAT" FOR APPEND AS #1
          Nm$ = "ACFM Data File"
          WRITE #1, Nm$
          WRITE #1, Basename$
          WRITE #1, dte$
          WRITE #1, tme$
          WRITE #1, pltno$
          WRITE #1,
     ELSE
     END IF

     PRINT
     PRINT "Then hit <SPACE> to begin scan"
     DO UNTIL INKEY$ = " ": LOOP
     CLS
     PRINT "SCANNING"
ELSE
     CLS
     CALL pc21reset(address%)
     CALL scan(ADD())
END IF

' Convert mm to number of steps
xmove = INT(xtrav * 1968.50394#)          ' change to microsteps
ymove = INT(yspace * 1968.50395#)         ' change to microsteps
```

```
IF yspace = 0 THEN                    ' no of scan passes
    passes = 1
ELSE
    passes = INT(ytrav / yspace) + 1
END IF

velocity = (.196850394# * vel) / 2.392   ' change to rev/s
accel = (.196850394# * acc)          ' change to rev/s^2

dispx$ = LTRIM$(STR$(xmove))   ' change numbers to strings for output to
dispy$ = LTRIM$(STR$(ymove))   ' indexers

IF velocity < 1 THEN        ' trim velocity string to indexer format
    vel$ = LEFT$(LTRIM$(STR$(velocity)), 4)
ELSE
    vel$ = LEFT$(LTRIM$(STR$(velocity)), 5)
END IF

accel$ = LEFT$(LTRIM$(STR$(accel)), 4)
               ' trim acceleration string to indexer format

' Send velocity and acceleration to indexers
FOR i = 1 TO 2
    address% = ADD(i)
    CMD$ = " MN V" + vel$ + " A" + accel$ + " "
    CALL PC21output(CMD$, address%)
NEXT i


' Move the probe and scan
FOR j = 1 TO passes

PRINT
PRINT "MAKE SURE PROBE IS IN CONTACT WITH PIPE SURFACE AND PRESS <SPACE> TO CONTINUE."
DO UNTIL INKEY$ = " ": LOOP

'$STATIC

' Declare UL Revision Level

  ULStat% = cbDeclareRevision(CURRENTREVNUM)

  ' Initiate error handling
  '  activating error handling will trap errors like
  '  bad channel numbers and non-configured conditions.
  '  Parameters:
  '    PRINTALL    :all warnings and errors encountered will be printed
  '    DONTSTOP    :if an error is encountered, the program will not stop,
  '             errors must be handled locally


  ULStat% = cbErrHandling%(PRINTALL, DONTSTOP)
  IF ULStat% <> 0 THEN STOP

  ' If cbErrHandling% is set for STOPALL or STOPFATAL during the program
  ' design stage, Quick Basic will be unloaded when an error is encountered.
  ' We suggest trapping errors locally until the program is ready for compiling
  ' to avoid losing unsaved data during program design.  This can be done by
```

```
 ' setting cbErrHandling options as above and checking the value of ULStat%
 ' after a call to the library. If it is not equal to 0, an error has occurred.

'set up the display screen
  CLS
  PRINT "Samples are displayed for user to visually verify data is being collected."
  PRINT
  Xs% = POS(0)
  Ys% = CSRLIN


'Collect the values with cbAInScan%() in BACKGROUND mode, CONTINUOUSLY
' Parameters:
'   BoardNum    :the number used by CB.CFG to describe this board
'   LowChan%    :the first channel of the scan
'   HighChan%   :the last channel of the scan
'   Count&      :the total number of A/D samples to collect
'   Rate&       :sample rate in samples per second
'   Gain%       :the gain for the board
'   ADData%     :the array for the collected data values
'   Options%    :data collection options

  LowChan% = 0
  HighChan% = 0
  Count& = NumPoints          ' total number of data points to collect
  Rate& = 100                 ' sampling rate (samples per second)
  Options% = BACKGROUND + CONTINUOUS + CONVERTDATA + SINGLEIO
                              ' collect data continuously in background
                              ' return data as 12-bit values
  Gain% = BIP10VOLTS          ' set the gain


  ULStat% = cbAInScan%(BoardNum, LowChan%, HighChan%, Count&, Rate&, Gain%, ADData%(0), Options%)
  IF ULStat% = 84 THEN
    PRINT "The CONVERT option cannot be used with 16 bit convertors. Set Options% to NOCONVERTDATA."
    STOP 'Change Options% above to NOCONVERTDATA (Options% = 0)
  END IF
  IF ULStat% <> 0 THEN STOP

  X% = CSRLIN
  Y% = POS(0)

address% = ADD(1)
CMD$ = " MN D-" + dispx$ + " G "
CALL PC21output(CMD$, address%)

'during the BACKGROUND operation, check the status, print the values

DO                      ' wait until move is finished

    '   CurCount&   :current number of samples collected
    '   CurIndex&   :index to the data buffer pointing to the last value transferred

    ULStat% = cbGetStatus%(BoardNum, Status%, CurCount&, CurIndex&)
    IF ULStat% <> 0 THEN STOP

    IF Status% = RUNNING THEN
        LOCATE X%, Y%
```

```
        PRINT USING "Data Point: ###### "; CurIndex&;
        IF CurIndex& >= 0 THEN
            PRINT USING "  Value: ######"; ADData%(CurIndex&);
        END IF
    END IF

LOOP WHILE Status% = RUNNING AND NOT movedone(address%)
PRINT

'the BACKGROUND operation must be explicitly stopped
' Parameters:
'   BoardNum :the number used by CB.CFG to describe this board

  ULStat% = cbStopBackground%(BoardNum)
  IF ULStat% <> 0 THEN STOP

PRINT
PRINT "Scan pass "; j; "of "; passes; "complete."

FOR k = 1 TO 10000: NEXT k          ' pause

PRINT
PRINT "LIFT PROBE AWAY FROM SURFACE AND PRESS <SPACE> TO CONTINUE."
DO UNTIL INKEY$ = " ": LOOP
PRINT

CMD$ = " MN D+" + dispx$ + " G "
CALL PC21output(CMD$, address%)
DO: LOOP UNTIL movedone(address%)     ' wait until move is finished
FOR k = 1 TO 10000: NEXT k          ' pause

address% = ADD(2)
CMD$ = " MN D" + dispy$ + " G "
CALL PC21output(CMD$, address%)
DO: LOOP UNTIL movedone(address%)     ' wait until move is finished
FOR k = 1 TO 10000: NEXT k          ' pause

' Write data to data file
IF fle$ = "y" THEN
     WRITE #1, -100   'flag marker for scan separation

     FOR m = 1 TO NumPoints
          WRITE #1, ADData%(m)
     NEXT m
ELSE
END IF

NEXT j

PRINT
PRINT
PRINT "Press <SPACE> to continue"
DO UNTIL INKEY$ = " ": LOOP

' close data file
CLOSE #1

CALL gohome(772)
```

```
END SUB

SUB xaxis (address%)

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

' Set up keys
lft$ = CHR$(0) + CHR$(75)
rght$ = CHR$(0) + CHR$(77)
inst$ = CHR$(0) + CHR$(82)
hm$ = CHR$(0) + CHR$(71)
spacebar$ = " "
rtrn$ = CHR$(13)

done = 0

DO UNTIL done = 1
    kbd$ = INKEY$
    SELECT CASE kbd$
        CASE IS = lft$
            ' move left
            CMD$ = " MC H- V3 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i


        CASE IS = rght$
            ' move right
            CMD$ = " MC H+ V3 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i

        CASE IS = inst$
            ' move slowly left
            CMD$ = " MC H- V1 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i

        CASE IS = hm$
            ' move slowly right
            CMD$ = " MC H+ V1 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i

        CASE IS = spacebar$
            CMD$ = " K "
            CALL PC21output(CMD$, address%)
            done = 1

        CASE IS = rtrn$
            CMD$ = " S "
            CALL PC21output(CMD$, address%)
            done = 1
```

```
        CASE ELSE
            CMD$ = " S "
            CALL PC21output(CMD$, address%)
            done = 0
    END SELECT

LOOP

END SUB

SUB yaxis (address%)

SHARED CONTROL, STOPPED, CRASH, LOADRDY, INTACK, MESSAGE, IDBREADY
SHARED FAULT, INTERRUPT, RECEIVED

address% = 772

' Set up keys
up$ = CHR$(0) + CHR$(72)
dwn$ = CHR$(0) + CHR$(80)
pgup$ = CHR$(0) + CHR$(73)
pgdwn$ = CHR$(0) + CHR$(81)
spacebar$ = " "
rtrn$ = CHR$(13)

done = 0

DO UNTIL done = 1
    kbd$ = INKEY$
    SELECT CASE kbd$
        CASE IS = up$
            ' move in
            CMD$ = " MC H+ V3 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i

        CASE IS = dwn$
            ' move out
            CMD$ = " MC H- V3 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i

        CASE IS = pgup$
            ' move slowly in
            CMD$ = " MC H+ V1 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i

        CASE IS = pgdwn$
            ' move slowly out
            CMD$ = " MC H- V1 A20 G "
            CALL PC21output(CMD$, address%)
            done = 0
            FOR i = 1 TO 10000: NEXT i
```

```
        CASE IS = spacebar$
            CMD$ = " K "
            CALL PC21output(CMD$, address%)
            done = 1

        CASE IS = rtrn$
            CMD$ = " S "
            CALL PC21output(CMD$, address%)
            done = 1

        CASE ELSE
            CMD$ = " S "
            CALL PC21output(CMD$, address%)
            done = 0
    END SELECT

LOOP

END SUB
```

M-file: acfmscanx.m

Purpose: Used to plot the B_x data acquired from SCAN.BAS for visual verification and save it in a proper matrix format for further processing. acfmscanz.m was used for the B_z data.

```
% ********************************************
% Matlab File for Parsing ACFM/QFM Data Files
% ********************************************

clear
% load data file. It must be modified so that text is removed and only
% one channel of data is present.

load pl_1_bx.txt;

% Save to temporary matrix for manipulation
A = pl_1_bx;

% find size of matrix
m = size(A,1);

% Begin separating scans
point = 0;
vect = 1;

for l = 1:m;
        if A(l,1) == -100;
                if l == m;
                else;
                        point = l+1;
                        point2 = point + 1046;
                        eval(['x' int2str(vect) ' = A(' int2str(point) ':' int2str(point2) ',1);']);
                        vect = vect + 1;
                end;
        end;
end;

vect = vect-1;

% recombine scan into 3D matrix
for q = 1:vect;
        eval(['Z = x' int2str(q) ';' ]);
        W = Z';
        V = [V; W];
end;

clear q Z W;


% Reduce the number of data points to a manageable number for plotting
clear l c;
for l = 1:vect;
        count = 1;
        for c = 1:200;
                eval(['y' int2str(l) '(c,1) = x' int2str(l) '(count,1);']);
                count = count + 5;
        end;
```

```
end;

% recombine scan into 3D matrix
for q = 1:vect;
        eval(['Z = y' int2str(q) ';' ]);
        W = Z';
        U = [U; W];
end;


clear p q k s c d B N W Z;

for p = 1:vect;
        eval(['clear x' int2str(p)]);
end;

clear p vect;

% Begin Surface Plotting routine

[m,n] = size(U);

for k = 1:m;
        y(1,k) = (k-1)*100/(m-1);
end;

for l = 1:n;
        x(1,l) = (l-1)*200/(n-1);
end;

figure(1);
surf(x,y,U);

xlabel('X - Length of Scan Area (mm)');
ylabel('Y - Width of Scan Area (mm)');
zlabel('ACFM Bx Data (Data Acquisition Card Units)');
title('Isometric of Plate 1 Surface Bx Scan');

% Second plot of top down view
figure(2);
pcolor(x,y,U);
xlabel('X - Length of Scan Area (mm)');
ylabel('Y - Width of Scan Area (mm)');
title('2-D Top Down Representation of Plate 1 Bx Scan');

% Third plot side view along x
figure(3);
surf(x,y,U);
view(0,0);
xlabel('Length of Scan Area (mm)');
zlabel('ACFM Bx Data (A/D Card Units)');
title('View of Plate 1 Bx Scan Along the Scan Area Length')

% Fourth plot side view along y
figure(4);
surf(x,y,U);
view(90,0);
ylabel('Width of Scan Area (mm)');
```

```
zlabel('ACFM Bx Data (A/D Card Units)');
title('View of Plate 1 Bx Scan Along the Scan Area Width');

% save data in matrix format
D = V';
save bx1.txt D -ascii;
```

M-file: sccxm.m

Purpose: Used to plot the B_x data acquired from PIPESCAN.BAS for visual verification and save it in a proper matrix format for further processing. scczm.m was used for the B_z data.

```
% ******************************************************
% Matlab File for Plotting ACFM SCC Pipe Scans Data Files
% ******************************************************

clear
% load data file. It must be modified so that text is removed and only
% one channel of data is present.

load pipe2abx.txt;

% Save to temporary matrix for manipulation
A = pipe2abx;

% find size of matrix
m = size(A,1);

% Begin separating scans
point = 0;
vect = 1;

for l = 1:m;
        if A(l,1) == -100;
                if l == m;
                else;
                        point = i+1;
                        point2 = point + 1359;
                        eval(['x' int2str(vect) ' = A(' int2str(point) ':' int2str(point2) ',1);']);
                        vect = vect + 1;
                end;
        end;
end;

vect = vect-1;

clear l c;

% reduce number of data points to manageable number if required
for l = 1:vect;
        count = 1;
        for c = 1:1359;
                eval(['y' int2str(l) '(c,1) = x' int2str(l) '(count,1);']);
                count = count + 1;
        end;
end;

% recombine scan into 3D matrix
for q = 1:vect;
        eval(['Z = y' int2str(q) ';' ]);
        W = Z';
        U = [U; W];
end;
```

```
V = U';
%save mp2x.txt V -ascii

clear p q k s c d m B N W Z;

for p = 1:vect;
        eval(['clear x' int2str(p)]);
end;

clear p vect;

% Time averaging of data to reduce noise.

[m,n] = size(V);

for p = 1:n;
        if p == 1;
                for q = 1:m;
                        if q == 1;
                                M = V(1:2,1:2);
                                o = mean(M);
                                r = o';
                                Z(q,p) = mean(r);
                                clear M o r;
                        elseif q == m;
                                a = q-1;
                                M = V(a:m,1:2);
                                o = mean(M);
                                r = o';
                                Z(q,p) = mean(r);
                                clear M o r a;
                        else;
                                a = q-1;
                                b = q+1;
                                M = V(a:b,1:2);
                                o = mean(M);
                                r = o';
                                Z(q,p) = mean(r);
                                clear M o r a b;
                        end;
                end;
        elseif p == n;
                for q = 1:m;
                        if q == 1;
                                c = p-1;
                                M = V(1:2,c:p);
                                o = mean(M);
                                r = o';
                                Z(q,p) = mean(r);
                                clear M o r c;
                        elseif q == m;
                                a = q-1;
                                c = p-1;
                                M = V(a:m,c:p);
                                o = mean(M);
                                r = o';
                                Z(q,p) = mean(r);
                                clear M o r a c;
```

```
                else;
                        a = q-1;
                        b = q+1;
                        c = p-1;
                        M = V(a:b,c:p);
                        o = mean(M);
                        r = o';
                        Z(q,p) = mean(r);
                        clear M o r a b c;
                end;
            end;
    else;
            for q = 1:m;
                if q == 1;
                        c = p-1;
                        d = p+1;
                        M = V(1:2,c:d);
                        o = mean(M);
                        r = o';
                        Z(q,p) = mean(r);
                        clear M o r c d;
                elseif q == m;
                        a = q-1;
                        c = p-1;
                        d = p+1;
                        M = V(a:q,c:d);
                        o = mean(M);
                        r = o';
                        Z(q,p) = mean(r);
                        clear M o r a c d;
                else;
                        a = q-1;
                        b = q+1;
                        c = p-1;
                        d = p+1;
                        M = V(a:b,c:d);
                        o = mean(M);
                        r = o';
                        Z(q,p) = mean(r);
                        clear M o r a b c d;
                end;
            end;
    end;
end;

% save data to ASCII file
%save mp2xavg.txt Z -ascii


% Begin Surface Plotting routine

% Extract every fifth point from matrix Z for plotting

u = (m/5);
v = (u-0.5);
round(v);

for p = 1:n;
```

```
        count = 1;
        for q = 1:v;
%       for q = 1:m;
                B(q,p) = Z(count,p);
                count = count + 5;
        end;
end;

clear m n;

[m,n] = size(B);

for k = 1:m;
        y(1,k) = (k-1)*680/(m-1) + 0;
end;

for l = 1:n;
        x(1,l) = (l-1)*180/(n-1) + 0;
end;

%figure(5)
surf(x,y,B);
xlabel('Width of Scan Area (mm)');
ylabel('Length of Scan Area (mm)');
zlabel('ACFM Bx Data (Data Acquisition Card Units)');
title('Microprobe Bx Scan Isometric from Section 2a: Time Averaged Data');

% Second plot of top down view
figure(6);
pcolor(x,y,B);
xlabel('Width of Scan Area (mm)');
ylabel('Length of Scan Area (mm)');
title('Contour Plot of Microprobe Bx Scan from Section 2a: Time Averaged Data');

% Third plot side view along x
figure(7);
surf(x,y,B);
view(0,0);
xlabel('Width of Scan Area (mm)');
zlabel('ACFM Bx Data (A/D Card Units)');
title('Microprobe Bx Scan Along Length from Section 2a: Time Averaged data');

% Fourth plot side view along y
figure(8);
surf(x,y,B);
view(90,0);
ylabel('Length of Scan Area (mm)');
zlabel('ACFM Bx Data (A/D Card Units)');
title('Microprobe Bx Scan Along Width from Section 2a: Time Averaged Data');


E = B'
save penp_2ax.txt E -ascii
```

M-file: def_size.m

Purpose: Used to identify the locations of the machined defects within the colonies and obtain a size estimate of the defects based upon the TSC sizing tables.

```
%*****************************************************************************
%*          Matlab Routine for Extracting Single Defect Signals          *
%*                    From ACFM Scans                    *
%*                And Obtaining Depth Estimates          *
%*****************************************************************************

clear;

% Load the Bx and Bz data file to be classified
load bz5x50.txt;
load bx5x50.txt;

% Invert the file so that x is the horizontal axis
A = bz5x50';
AA = bx5x50';


% For the machine defects due to a delay in finishing the data acquistion
% the matrices are 47 data points too long so they must be cut down.
[o,p] = size(A);

for q = 1:o;
        eval(['z' int2str(q) '= A(q,1:1000);']);
        eval(['x' int2str(q) '= AA(q,1:1000);']);
end;

for q = 1:o;
        eval(['Z = z' int2str(q) ';']);
        eval(['X = x' int2str(q) ';']);
        C = [C; Z];
        CC = [CC; X];
end;

clear q;

for q = 1:o;
        eval(['clear z' int2str(q)]);
        eval(['clear x' int2str(q)]);
end;

% Convert values in matrix to between 0 and 1
B = (C - 1880)/240;

% Adjust the image size
D = imresize(B, [50 100], 'nearest', 0);

% Invert the data in D to assist with trough location
E = (D * (-1)) + 1;

% Display the Bz Data converting it to a 200x100 image
%figure(1);
```

```
%imshow(D,256);
%title('Bz Contour Plot')

clear B C;

%*******************************************************
%*            Peak and Trough Identification          *
%*******************************************************

% This routine looks in eight locations around a pixel to determine if
% if the pixel in the point of interest is brighter than its neighbours


% Set the threshold value to detect the bright areas in the Bz image

p_thresh = 0.075;


% It is not essential that all eight locations satisfy this criteria
% The percentage of directions which should satisfy the criteria is:

per_locations = 0.7;


%*********************************************************************

% The following are the direction masks to determine if the threshold
% criteria is met. For ease, compass point locations are used


east = [   0 0 0 0 0 -1 -1;
           0 0 0 6 0 -1 -1;
           0 0 0 0 0 -1 -1];

west = [  -1 -1 0  0 0 0 0;
          -1 -1 0  6 0 0 0;
          -1 -1 0  0 0 0 0];

north = west';
south = east';

nwest = [ -1 -1 -1  0  0 0 0;
          -1 -1  0  0  0 0 0;
          -1  0  0  0  0 0 0;
           0  0  0  6  0 0 0;
           0  0  0  0  0 0 0;
           0  0  0  0  0 0 0;
           0  0  0  0  0 0 0];

swest = [  0  0  0  0  0 0 0;
           0  0  0  0  0 0 0;
           0  0  0  0  0 0 0;
           0  0  0  6  0 0 0;
          -1  0  0  0  0 0 0;
          -1 -1  0  0  0 0 0;
          -1 -1 -1  0  0 0 0];
```

```
neast = swest';
seast = nwest';

%*********************************************************************
% Staring with the peaks in the signal

% Use a median filter to remove excess noise from the Bz data

Bz_peaks = medfilt2(D);


% Convolve the image file with each direction mask and record the directions
% which satisfy p_thresh

% North
peak = conv2(Bz_peaks, north, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = p_sig;

% South
peak = conv2(Bz_peaks, south, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% East
peak = conv2(Bz_peaks, east, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% West
peak = conv2(Bz_peaks, west, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;


% North_East
peak = conv2(Bz_peaks, neast, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;


% North-West
peak = conv2(Bz_peaks, nwest, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% South-East
peak = conv2(Bz_peaks, seast, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% South-West
peak = conv2(Bz_peaks, swest, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% Plot the high areas
```

```
%figure(2);
%imshow(num_dir/8,256);
%title('High Areas');

% Filter out the peaks based upon the criteria of the number of
% directions surrounding the pixels

p_sig = im2bw(num_dir/8, per_locations);

% Plot the final results
%figure(3);
%imshow(p_sig,2);
%title('High Areas');

clear num_dir;

%*************************************************************
% Now identify the troughs

% Use a median filter to remove excess noise from the Bz data

Bz_troughs = medfilt2(E);


% Convolve the image file with each direction mask and record the directions
% which satisfy p_thresh

% North
peak = conv2(Bz_troughs, north, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = t_sig;

% South
peak = conv2(Bz_troughs, south, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% East
peak = conv2(Bz_troughs, east, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% West
peak = conv2(Bz_troughs, west, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;


% North_East
peak = conv2(Bz_troughs, neast, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;


% North-West
peak = conv2(Bz_troughs, nwest, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;
```

```
% South-East
peak = conv2(Bz_troughs, seast, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% South-West
peak = conv2(Bz_troughs, swest, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% Plot the low areas
%figure(4);
%imshow(num_dir/8,256)
%title('Low Areas')

% Filter out the peaks based upon the criteria of the number of
% directions surrounding the pixels which must be lower

t_sig = im2bw(num_dir/8, per_locations);

% Plot the final results
%figure(5);
%imshow(t_sig,2)
%title('Low Areas');

clear north south east west neast nwest seast swest;
clear per_locations p_thresh num_dir peak;

%****************************************************************
% Find the centers of the peak and trough locations

% Peaks
[p1,p2] = center(p_sig);

% Troughs
[t1,t2] = center(t_sig);


% Combine the peak and trough locations into indivdual matrices
P = [p1:p2]';
T = [t1:t2]';

% Order the matrices in ascending order according to the Y position
[Y1,Ip] = sort(P(:,1));
[Y2,It] = sort(T(:,1));

[m,n] = size(P);

for j = 1:m;
        r = Ip(j);
        Y1(j,2:n) = P(r,2:n);
end;

[m,n] = size(T);

for j = 1:m;
        o = It(j);
        Y2(j,2:n) = T(o,2:n);
```

```
end;

% Make sure that each indicated area is a peak or trough and not just
% signal noise.

[m,n] = size(Y1);
count = 1;

for k = 1:m;
        Bz = abs(D(Y1(k,1),Y1(k,2)) - D(Y1(k,1),1));
        BZ = (Bz*240);
        if BZ >= 25;
                PEAK(count,:) = Y1(k,:);
                count = count+1;
        end;
end;

[m,n] = size(Y2);
count = 1;

for k = 1:m;
        Bz = abs(D(Y2(k,1),Y2(k,2)) - D(Y2(k,1),1));
        BZ = (Bz*240);
        if BZ >= 20;
                TROUGH(count,:) = Y2(k,:);
                count = count+1;
        end;
end;


PEAKS = round(PEAK);
TROUGHS = round(TROUGH);


% Determine whether the peaks of troughs appear first for later use
% in constructing colonies from single defects.

if PEAKS(1,2) < TROUGHS(1,2);
        FIRST = PEAKS;
        SECOND = TROUGHS;
else,
        FIRST = TROUGHS;
        SECOND = PEAKS;
end;

clear Y1 Y2 t_sig p_sig P T p1 p2 t1 t2 Ip It Bz_peaks Bz_troughs;

% Plot the defect positions:

[m,n] = size(D);

for k = 1:m;
        for j = 1:n;
                pos(k,j) = 0;
        end;
end;

[m,n] = size(FIRST);
```

```
for l = 1:m;
        a = FIRST(l,1);
        for k = FIRST(l,2):SECOND(l,2);
                pos(a,k) = 1;
        end;
end;

%figure(6);
%imshow(pos,2);

%**********************************************************************
%*    Resize the Bx signal and Determine the background level        *
%**********************************************************************

% NOTE: Assume the background level is 40 mm on either side of the colony
% or at x=0 or x=200 whichever applies

CONVx = (CC - 3255)/250;

BB = imresize(CONVx, [50 100], 'nearest', 0);

% Convert to ACFM Units
Bx = (((250*BB) - 3255) * 1.161) - 2598;

[p,q] = size(Bx);

% Determine the background level
b1 = min(FIRST(:,2));
b2 = max(SECOND(:,2));

if (b1 - 20) >= 1;
        p1 = b1 - 20;
else;
        p1 = 1;
end;

if (b2 + 20) <= 100;
        p2 = b2 + 20;
else;
        p2 = 100;
end;

BACK(:,1) = Bx(:,p1);
BACK(:,2) = Bx(:,p2);

for l = 1:p;
        BACK_AVG(l,1) = mean(BACK(l,:));
end;

clear p1 p2 b1 b2;


%******************************************************
% Sizing routine routine
%******************************************************

[m,n] = size(FIRST);
```

```
% Determine (back-min)/back ratio for each defect

for l = 1:m;
        BACK = BACK_AVG(FIRST(l,1),1);
        MIN = min(Bx(FIRST(l,1),:));
        eval(['rat' int2str(l) ' = (BACK - MIN)/BACK;']);
end;

% Determine the length of each defect:

for j = 1:m;
        eval(['len' int2str(j) ' = (SECOND(l,2) - FIRST(l,2)) * 2;']);
end;

clear l j;

% Determine the lengths of the defects based upon ACFM theory
load onval3z.txt;

[o,p] = size(onval3z);

for l = 1:m;
        l_temp = onval3z(1,2);
        count = 1;

        eval(['len_tmp = len' int2str(1) ';']);
        while l_temp < len_tmp;
                count = count + 1;
                l_temp = onval3z(count,2);
        end;

% interpolate the length parameter as required
        a = onval3z(count,2);
        b = onval3z((count - 1),2);
        c = onval3z(count,1);
        d = onval3z((count - 1),1);
        eval(['e = len' int2str(l) ';']);
        eval(['LENGTH' int2str(l) ' = c - ((c-d)/(a-b)*(a-e));']);

end;

clear a b c d e o p l l_temp count offval3z;


% Determine the crack depth
load onval3x.txt


for l = 1:m;
        count = 1;
        eval(['l_temp2 = LENGTH' int2str(l) ';']);
        while onval3x(count,1) < l_temp2;
                count = count + 21;
        end;

        if onval3x(count,1) == l_temp2;

% If the predicted length is a factor of 5
```

```
                    count2 = count + 1,
                    eval(['rat_tmp = rat' int2str(l) ';']).
                    while onval3x(count2,1) < rat_tmp;
                                count2 = count2 + 1;
                    end;

% Get the correct multiplier for B/A
                    a = onval3x(count2,1);
                    b = onval3x((count2 - 1),1);
                    c = rat_tmp;

                    mult = count2 + (a-c)/(a-b);

% Calculate the depth
                    B = 0.05 * l_temp2 / 2;
                    eval(['Depth' int2str(l) ' = B * mult;']);


            else;
% Have to interate between length values
                    count2 = count - 20;
                    count3 = count + 1;
                    eval(['rat_tmp = rat' int2str(l) ';']);

                    while onval3x(count2,1) < rat_tmp;
                                count2 = count2 + 1;
                    end;

                    while onval3x(count3,1) < rat_tmp;
                                count3 = count3 + 1;
                    end;

% Get the correct multiplier for B/A
                    a = onval3x(count2,1);

                    if onval3x((count2 - 1),1) > 1;
                                b = 0;
                    else;
                                b = onval3x((count2 - 1),1);
                    end;

                    c = rat_tmp;

                    d = onval3x(count3,1);

                    if onval3x((count2 - 1),1) > 1;
                                e = 0;
                    else;
                                e = onval3x((count3 - 1),1);
                    end;

                    mult2 = (count2 + 20 - count) + 1 - (a-c)/(a-b);
                    mult3 = (count3 - count - 1) + 1 - (d-c)/(d-e);

% Calculate the depth
                    B2 = 0.05 * onval3x((count - 21),1) / 2;
                    B3 = 0.05 * onval3x(count,1) / 2;
                    Dep2 = B2 * mult2;
```

```
                Dep3 = B3 * mult3;

                f = onval3x(count,1);
                g = onval3x((count - 21),1);

                dep_tmp = Dep3 - (f - l_temp2)/(f-g) * (Dep3 - Dep2);
                eval(['Depth' int2str(l) ' = dep_tmp;']);
        end;
end;


% Display values
for l = 1:m;
        eval(['LENGTH' int2str(l) ''])
        eval(['Depth' int2str(l) ''])
end;
```

M-file: lin_sup.m

Purpose: Used to identify the locations of the machined defects within the colonies and linear superimpose single defects into the colony configurations.

```
%*************************************************************************
%*          Matlab Routine for Linearly Superimposing          *
%*                   Single Defects into                 *
%*                 Machined Colony Config.'s    *
%*************************************************************************

clear;

% Load the Bx and Bz data file to be classified
load bz13.txt;
load bx13.txt;

% Invert the file so that x is the horizontal axis
A = bz13';
AA = bx13';


% For the machine defects due to a delay in finishing the data acquistion
% the matrices are 47 data points too long so they must be cut down.
[o,p] = size(A);

for q = 1:o;
        eval(['z' int2str(q) '= A(q,1:1000);']);
        eval(['x' int2str(q) '= AA(q,1:1000);']);
end;

for q = 1:o;
        eval(['Z = z' int2str(q) ';']);
        eval(['X = x' int2str(q) ';']);
        C = [C; Z];
        CC = [CC; X];
end;

clear q;

for q = 1:o;
        eval(['clear z' int2str(q)]);
        eval(['clear x' int2str(q)]);
end;

% Convert values in matrix to between 0 and 1
B = (C - 1880)/240;

% Adjust the image size
D = imresize(B, [50 100], 'nearest', 0);

% Invert the data in D to assist with trough location
E = (D * (-1)) + 1;

% Display the Bz Data converting it to a 200x100 image
figure(1);
imshow(D,256);
```

```
title('Bz Contour Plot')

clear B C;

%*******************************************************
%*          Peak and Trough Identification           *
%*******************************************************

% This routine looks in eight locations around a pixel to determine if
% if the pixel in the point of interest is brighter than its neighbours


% Set the threshold value to detect the bright areas in the Bz image

p_thresh = 0.075;


% It is not essential that all eight locations satisfy this criteria
% The percentage of directions which should satisfy the criteria is:

per_locations = 0.9;


%*******************************************************************

% The following are the direction masks to determine if the threshold
% criteria is met. For ease, compass point locations are used


east = [   0 0 0 0 0 -1 -1;
           0 0 0 6 0 -1 -1;
           0 0 0 0 0 -1 -1];

west = [  -1 -1 0 0 0 0 0;
          -1 -1 0 6 0 0 0;
          -1 -1 0 0 0 0 0];

north = west';
south = east';

nwest = [ -1 -1 -1 0 0 0 0;
          -1 -1 0 0 0 0 0;
          -1 0 0 0 0 0 0;
          0 0 0 6 0 0 0;
          0 0 0 0 0 0 0;
          0 0 0 0 0 0 0;
          0 0 0 0 0 0 0];

swest = [ 0 0 0 0 0 0 0;
          0 0 0 0 0 0 0;
          0 0 0 0 0 0 0;
          0 0 0 6 0 0 0;
          -1 0 0 0 0 0 0;
          -1 -1 0 0 0 0 0;
          -1 -1 -1 0 0 0 0];
```

```
neast = swest';
seast = nwest';

%*********************************************************************
% Staring with the peaks in the signal

% Use a median filter to remove excess noise from the Bz data

Bz_peaks = medfilt2(D);


% Convolve the image file with each direction mask and record the directions
% which satisfy p_thresh

% North
peak = conv2(Bz_peaks, north, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = p_sig;

% South
peak = conv2(Bz_peaks, south, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% East
peak = conv2(Bz_peaks, east, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% West
peak = conv2(Bz_peaks, west, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;


% North_East
peak = conv2(Bz_peaks, neast, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;


% North-West
peak = conv2(Bz_peaks, nwest, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% South-East
peak = conv2(Bz_peaks, seast, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% South-West
peak = conv2(Bz_peaks, swest, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% Plot the high areas
figure(2);
```

```
imshow(num_dir/8,256);
title('High Areas');

% Filter out the peaks based upon the criteria of the number of
% directions surrounding the pixels

p_sig = im2bw(num_dir/8, per_locations);

% Plot the final results
figure(3);
imshow(p_sig,2);
title('High Areas');

clear num_dir;

%*****************************************************************
% Now identify the troughs

% Use a median filter to remove excess noise from the Bz data

Bz_troughs = medfilt2(E);


% Convolve the image file with each direction mask and record the directions
% which satisfy p_thresh

% North
peak = conv2(Bz_troughs, north, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = t_sig;

% South
peak = conv2(Bz_troughs, south, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% East
peak = conv2(Bz_troughs, east, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% West
peak = conv2(Bz_troughs, west, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;


% North_East
peak = conv2(Bz_troughs, neast, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;


% North-West
peak = conv2(Bz_troughs, nwest, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;
```

```
% South-East
peak = conv2(Bz_troughs, seast, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% South-West
peak = conv2(Bz_troughs, swest, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% Plot the low areas
figure(4);
imshow(num_dir/8,256)
title('Low Areas')

% Filter out the peaks based upon the criteria of the number of
% directions surrounding the pixels which must be lower

t_sig = im2bw(num_dir/8, per_locations);

% Plot the final results
figure(5).
imshow(t_sig,2)
title('Low Areas');

clear north south east west neast nwest seast swest;
clear per_locations p_thresh num_dir peak;

%*****************************************************************
% Find the centers of the peak and trough locations

% Peaks
[p1,p2] = center(p_sig);

% Troughs
[t1,t2] = center(t_sig);


% Combine the peak and trough locations into indivdual matrices
P = [p1;p2]';
T = [t1;t2]';

% Order the matrices in ascending order according to the Y position
[Y1,Ip] = sort(P(:,1));
[Y2,It] = sort(T(:,1));

[m,n] = size(P);

for j = 1:m;
        r = Ip(j);
        Y1(j,2:n) = P(r,2:n);
end;

[m,n] = size(T);

for j = 1:m;
        o = It(j);
        Y2(j,2:n) = T(o,2:n);
```

```
end;

% Make sure that each indicated area is a peak or trough and not just
% signal noise.

[m,n] = size(Y1);
count = 1;

for k = 1:m;
        Bz = abs(D(Y1(k,1),Y1(k,2)) - D(Y1(k,1),1));
        BZ = (Bz*240);
        if BZ >= 20;
                PEAK(count,:) = Y1(k,:);
                count = count+1;
        end;
end;

[m,n] = size(Y2);
count = 1;

for k = 1:m;
        Bz = abs(D(Y2(k,1),Y2(k,2)) - D(Y2(k,1),1));
        BZ = (Bz*240);
        if BZ >= 20;
                TROUGH(count,:) = Y2(k,:);
                count = count+1;
        end;
end;


PEAKS = round(PEAK);
TROUGHS = round(TROUGH);


% Determine whether the peaks of troughs appear first for later use
%  in constructing colonies from single defects.

if PEAKS(1,2) < TROUGHS(1,2);
        FIRST = PEAKS;
        SECOND = TROUGHS;
else;
        FIRST = TROUGHS;
        SECOND = PEAKS;
end;

clear Y1 Y2 t_sig p_sig P T p1 p2 t1 t2 lp lt Bz_peaks Bz_troughs;

% Plot the defect positions:

[m,n] = size(D);

for k = 1:m;
        for j = 1:n;
                pos(k,j) = 0;
        end;
end;

for h = 1:m;
```

```
        y(1,h) = (h-1)*100/(m-1);
end;

for h = 1:n;
        x(1,h) = (h-1)*200/(n-1);
end;

[m,n] = size(FIRST);

for l = 1:m;
        a = FIRST(l,1);
        for k = FIRST(l,2):SECOND(l,2);
                pos(a,k) = 1;
        end;
end;

figure(6);
colormap(gray(2));
surf(x,y,pos);
view(0,90);
xlabel('Length of Scan Area (mm)');
ylabel('Width of Scan Area (mm)');
print -deps fig49;

%*********************************************************************
%*      Resize the Bx signal and Determine the background level      *
%*********************************************************************

% NOTE: Assume the background level is 40 mm on either side of the colony
% or at x=0 or x=200 whichever applies

CONVx = (CC - 3255)/250;

BB = imresize(CONVx, [50 100], 'nearest', 0);

% Convert to ACFM Units
Bx = (((250*BB) + 3255) * 1.161) - 2598;

[p,q] = size(Bx);

% Determine the background level
b1 = min(FIRST(:,2));
b2 = max(SECOND(:,2));

if (b1 - 20) >= 1;
        p1 = b1 - 20;
else;
        p1 = 1;
end;

if (b2 + 20) <= 100;
        p2 = b2 + 20;
else;
        p2 = 100;
end;

BACK(:,1) = Bx(:,p1);
BACK(:,2) = Bx(:,p2);
```

```
for l = 1:p;
        BACK_AVG(l,1) = mean(BACK(l,:));
end;

clear p1 p2 b1 b2;


%****************************************
% Create a blank matrix with background
% level of BACK_AVG
%****************************************

[m,n] = size(Bx);

for l = 1:m;
        for k = 1:n;
                Blank(l,k) = BACK_AVG(l,1);
        end;
end;


%****************************************************
% Characterization routine
%****************************************************

% Load the length and ratio files for each single defect

load s2x75.txt;
load s2x75_len.txt;
load s2x75_back.txt;
tmp = s2x75;
len1 = s2x75_len;
back1 = s2x75_back;

[o,r] = size(tmp);
for j = 1:o;
for k = 1:r;
def1(j,k) = BACK_AVG(j,1) - (back1(j,1) - tmp(j,k))*BACK_AVG(j,1)/back1(j,1);
end;
end;
rat1 = (def1(16,1) - min(def1(16,:)))/def1(16,1);
for l = 1:o;
        def1(l,:) = def1(l,:) - BACK_AVG(l,1);
end;

load s5x75.txt;
load s5x75_len.txt;
load s5x75_back.txt;
tmp = s5x75;
len2 = s5x75_len;
back2 = s5x75_back;

[o,r] = size(tmp);
for j = 1:o;
for k = 1:r;
def2(j,k) = BACK_AVG(j,1) - (back2(j,1) - tmp(j,k))*BACK_AVG(j,1)/back2(j,1);
end;
end;
```

```
rat2 = (def2(16,1) - min(def2(16,:)))/def2(16,1);
for l = 1:o;
        def2(l,:) = def2(l,:) - BACK_AVG(l,1);
end;

load s2x50.txt;
load s2x50_len.txt;
load s2x50_back.txt;
tmp = s2x50;
len3 = s2x50_len;
back3 = s2x50_back;

[o,r] = size(tmp);
for j = 1:o;
for k = 1:r;
def3(j,k) = BACK_AVG(j,1) - (back3(j,1) - tmp(j,k))*BACK_AVG(j,1)/back3(j,1);
end;
end;
rat3 = (def3(16,1) - min(def3(16,:)))/def3(16,1);
for l = 1:o;
        def3(l,:) = def3(l,:) - BACK_AVG(l,1);
end;


load s5x50.txt;
load s5x50_len.txt;
load s5x50_back.txt;
tmp = s5x50;
len4 = s5x50_len;
back4 = s5x50_back;

[o,r] = size(tmp);
for j = 1:o;
for k = 1:r;
def4(j,k) = BACK_AVG(j,1) - (back4(j,1) - tmp(j,k))*BACK_AVG(j,1)/back4(j,1);
end;
end;
rat4 = (def4(16,1) - min(def4(16,:)))/def4(16,1);
for l = 1:o;
        def4(l,:) = def4(l,:) - BACK_AVG(l,1);
end;



% Check the possible defects which may match each defect in the colony
% for length and depth

[m,n] = size(FIRST);

for l = 1:m;
        len = abs(FIRST(l,2) - SECOND(l,2)) * 2;
        rat = (Bx(FIRST(l,1),1) - min(Bx(FIRST(l,1),:)))/Bx(FIRST(l,1),1);
        for k = 1:4;
                eval(['check_len = abs((len - len' int2str(k) ')/len);']);
                eval(['check_rat = abs((rat - rat' int2str(k) ')/rat);']);
                if check_len <= 0.15 & check_rat <= 0.50;
                        eval(['d' int2str(l) '(1,k) = 1;']);
                else;
```

```
                    eval(['d' int2str(l) '(1,k) = 0;']);
               end;
          end;
end;

%clear l k;

% Begin to construct the artificial colonies;
% Starting with the possible combinations for defect 1;


for l = 1:4;
     if d1(1,l) == 1;
               A = Blank;
               eval(['[p,q] = size(def' int2str(l) ');']);
               y1 = FIRST(1,1) - 15;
               y2 = y1 + p - 1;

               if y2 > 50;
                         y2 = 50;
                         a = y2 - y1 + 1;
               eval(['de' int2str(l) ' = def' int2str(l) '(1:a,:);']);
               else;
                         eval(['de' int2str(l) ' = def' int2str(l) ';']);
               end;

               x1 = FIRST(1,2) - 20;
               x2 = x1 + q - 1;

               if x2 > 100;
                         x2 = 100;
                         b = x2 - x1 + 1;
               eval(['de' int2str(l) ' = de' int2str(l) '(:,1:b);']);
               end;

               eval(['A(y1:y2,x1:x2) = A(y1:y2,x1:x2) + de' int2str(l) ';']);
               eval(['col' int2str(l) ' = A;']);
               c1(1,l) = 1;
     else;
               c1(1,l) = 0;
     end;
end;



for j = 1:4;
     if c1(1,j) == 1;
               eval(['B = col' int2str(j) ';']);
               for k = 1:4;
                         if d2(1,k) == 1;
                         eval(['[p,q] = size(def' int2str(k) ');']);
                         y3 = FIRST(2,1) - 15;
                         y4 = y3 + p - 1;

                         if y4 > 50;
                                   y4 = 50;
                                   a = y4 - y3 + 1;
               eval(['de' int2str(k) ' = def' int2str(k) '(1:a,:);']);
```

```
else.
        eval(['de' int2str(k) ' = def' int2str(k) ';']);
end;

x3 = FIRST(2,2) - 20;
x4 = x3 + q - 1;

if x4 > 100;
        x4 = 100;
        b = x4 - x3 + 1;
eval(['de' int2str(k) ' = de' int2str(k) '(:,1:b);']);
end;


eval(['B(y3:y4,x3:x4) = B(y3:y4,x3:x4) + de' int2str(k) ';']);
        eval(['col' int2str(j) int2str(k) ' = B;']);
        c2(j,k) = 1;
        else;
        c2(j,k) = 0;
        end;
    end;
else;
for t = 1:4;
        c2(j,t) = 0;
end;
end;
end;
```

% Insert the remaining defect and perform the comparison to the colonies

```
for j = 1:4;
    for k = 1:4;
        if c2(j,k) == 1;
            eval(['C = col' int2str(j) int2str(k) ';']);
            for l = 1:4;
                if d3(1,l) == 1;
                eval(['[p,q] = size(def' int2str(l) ');']);
                y5 = FIRST(3,1) - 15;
                y6 = y5 + p - 1;

                if y6 > 50;
                        y6 = 50;
                        a = y6 - y5 + 1;
            eval(['de' int2str(l) ' = def' int2str(l) '(1:a,:);']);
                else;
                eval(['de' int2str(l) ' = def' int2str(l) ';']);
                end;

                x5 = FIRST(3,2) - 20;
                x6 = x5 + q - 1;

                if x6 > 100;
                        x6 = 100;
                        b = x6 - x5 + 1;
            eval(['de' int2str(l) ' = de' int2str(l) '(:,1:b);']);
                end;
```

```
                    eval(['C(y5:y6,x5:x6) = C(y5:y6,x5:x6) + de' int2str(l) ';']);
                    eval(['col' int2str(j) int2str(k) int2str(l) ' = C;']);

% Check colony against Bx and evaluate
Config = sprintf('col%1.0f', eval([ int2str(j) int2str(k) int2str(l) ]))

A1 = Bx(FIRST(1,1),FIRST(1,2):SECOND(1,2));
A2 = Bx(FIRST(2,1),FIRST(2,2):SECOND(2,2));
A3 = Bx(FIRST(3,1),FIRST(3,2):SECOND(3,2));

eval(['col = col' int2str(j) int2str(k) int2str(l) ';']);

B1 = col(FIRST(1,1),FIRST(1,2):SECOND(1,2));
B2 = col(FIRST(2,1),FIRST(2,2):SECOND(2,2));
B3 = col(FIRST(3,1),FIRST(3,2):SECOND(3,2));

err1 = mean((B1 - A1)/A3);
err2 = mean((B2 - A2)/A3);
err3 = mean((B3 - A3)/A3);

error = (abs(err1) + abs(err2) + abs(err3))/3

pause;
                                        end;
                              end;
                    end;
          end;
end;
```

M-file: scc_det.m

Purpose: Peak detection algorithm for detecting SCC sites on pipe wall sections

---

```
%**********************************************************************
%*          Matlab Routine for Detecting Stress Corrosion            *
%*                    Sites                   *
%**********************************************************************

clear;

% Load the Bx and Bz data file to be classified
load penp_2abz.txt;

% Invert the file so that x is the horizontal axis
A = penp_2abz(101:1100,:)';


% Convert values in matrix to between 0 and 1
B = (A - 1930)/120;

% Adjust the image size
D = imresize(B, [50 100], 'nearest', 0);

% Invert the data in D to assist with trough location
E = (D * (-1)) + 1;
```

```
% Display the Bz Data converting it to a 200x100 image
figure(1);
imshow(D,256);
title('Bz Contour Plot')

clear B C;

%*********************************************************
%*          Peak and Trough Identification          *
%*********************************************************

% This routine looks in eight locations around a pixel to determine if
% if the pixel in the point of interest is brighter than its neighbours


% Set the threshold value to detect the bright areas in the Bz image

p_thresh = 0.3;


% It is not essential that all eight locations satisfy this criteria
% The percentage of directions which should satisfy the criteria is:

per_locations = 0.9;


%**********************************************************************

% The following are the direction masks to determine if the threshold
% criteria is met. For ease, compass point locations are used


east = [  0 0 0 0 0 -1 -1;
          0 0 0 6 0 -1 -1;
          0 0 0 0 0 -1 -1];

west = [  -1 -1 0 0 0 0 0;
          -1 -1 0 6 0 0 0;
          -1 -1 0 0 0 0 0];

north = west';
south = east';

nwest = [ -1 -1 -1 0 0 0 0;
          -1 -1 0 0 0 0 0;
          -1 0 0 0 0 0 0;
          0 0 0 6 0 0 0;
          0 0 0 0 0 0 0;
          0 0 0 0 0 0 0;
          0 0 0 0 0 0 0];

swest = [ 0 0 0 0 0 0 0;
          0 0 0 0 0 0 0;
          0 0 0 0 0 0 0;
          0 0 0 6 0 0 0;
          -1 0 0 0 0 0 0;
          -1 -1 0 0 0 0 0;
          -1 -1 -1 0 0 0 0];
```

```
neast = swest';
seast = nwest';

%***************************************************************
% Staring with the peaks in the signal

% Use a median filter to remove excess noise from the Bz data

Bz_peaks = medfilt2(D);


% Convolve the image file with each direction mask and record the directions
% which satisfy p_thresh

% North
peak = conv2(Bz_peaks, north, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = p_sig;

% South
peak = conv2(Bz_peaks, south, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% East
peak = conv2(Bz_peaks, east, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% West
peak = conv2(Bz_peaks, west, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;


% North_East
peak = conv2(Bz_peaks, neast, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;


% North-West
peak = conv2(Bz_peaks, nwest, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% South-East
peak = conv2(Bz_peaks, seast, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% South-West
peak = conv2(Bz_peaks, swest, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;
```

```
% Plot the high areas
figure(2);
imshow(num_dir/8,256);
title('High Areas');

% Filter out the peaks based upon the criteria of the number of
% directions surrounding the pixels

p_sig = im2bw(num_dir/8, per_locations);

% Plot the final results
figure(3);
imshow(p_sig,2);
title('High Areas');

clear num_dir;

%*************************************************************
% Now identify the troughs

% Use a median filter to remove excess noise from the Bz data

Bz_troughs = medfilt2(E);


% Convolve the image file with each direction mask and record the directions
% which satisfy p_thresh

% North
peak = conv2(Bz_troughs, north, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = t_sig;

% South
peak = conv2(Bz_troughs, south, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% East
peak = conv2(Bz_troughs, east, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% West
peak = conv2(Bz_troughs, west, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;


% North_East
peak = conv2(Bz_troughs, neast, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;


% North-West
peak = conv2(Bz_troughs, nwest, 'same');
t_sig = im2bw(peak, p_thresh);
```

```
num_dir = num_dir + t_sig;

% South-East
peak = conv2(Bz_troughs, seast, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% South-West
peak = conv2(Bz_troughs, swest, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% Plot the low areas
figure(4);
imshow(num_dir/8,256)
title('Low Areas')

% Filter out the peaks based upon the criteria of the number of
% directions surrounding the pixels which must be lower

t_sig = im2bw(num_dir/8, per_locations);

% Plot the final results
figure(5);
imshow(t_sig,2)
title('Low Areas');

clear north south east west neast nwest seast swest;
clear per_locations p_thresh num_dir peak;

%***************************************************************
% Find the centers of the peak and trough locations

% Peaks
[p1,p2] = center(p_sig);

% Troughs
[t1,t2] = center(t_sig);


% Combine the peak and trough locations into indivdual matrices
P = [p1:p2]';
T = [t1:t2]';

% Order the matrices in ascending order according to the Y position
[Y1,Ip] = sort(P(:,1));
[Y2,It] = sort(T(:,1));

[m,n] = size(P);

for j = 1:m;
        r = Ip(j);
        Y1(j,2:n) = P(r,2:n);
end;

[m,n] = size(T);

for j = 1:m;
```

```
        o = It(j);
        Y2(j,2:n) = T(o,2:n);
end;

% Make sure that each indicated area is a peak or trough and not just
% signal noise.

[m,n] = size(Y1);
count = 1;

for k = 1:m;
        Bz = abs(D(Y1(k,1),Y1(k,2)) - D(Y1(k,1),1));
        BZ = (Bz*140);
        if BZ >= 15;
                PEAK(count,:) = Y1(k,:);
                count = count+1;
        end;
end;

[m,n] = size(Y2);
count = 1;

for k = 1:m;
        Bz = abs(D(Y2(k,1),Y2(k,2)) - D(Y2(k,1),1));
        BZ = (Bz*240);
        if BZ >= 15;
                TROUGH(count,:) = Y2(k,:);
                count = count+1;
        end;
end;


PEAKS = round(PEAK);
TROUGHS = round(TROUGH);


clear Y1 Y2 t_sig p_sig P T p1 p2 t1 t2 Ip It Bz_peaks Bz_troughs;



% Determine which of the 2 matrices is smaller and use it to match
% peaks with troughs

[mp, np] = size(PEAKS);
[mt, nt] = size(TROUGHS);


if mp <= mt;
        A1 = PEAKS;
        A2 = TROUGHS;

        for l = 1:mp;
                a = A1(l,:);
                for j = 1:mt;
                        b = A2(j,:);
                        diff1(j,1) = abs(a(1,1) - b(1,1));
                        if diff1(j,1) < 6;
                                eval(['c' int2str(j) ' = b;']);
```

```
                else;
                        eval(['c' int2str(j) ' = 0;']);
                end;
        end;

        for j = 1:mt;
                eval(['d = c' int2str(j) ';']);
                if d > 0;
                        diff2(j,1) = abs(a(1,2) - d(1,2));
                else;
                        diff2(j,1) = 1000;
                end;
        end;

        min_diff = min(diff2(:,1));

        for h = 1:mt;
                if diff2(h,1) == min_diff;
                        row = TROUGHS(h,:);
                end;
        end;
        TROUGHS_r = [TROUGHS_r;row];
    end;

    PEAKS_r = PEAKS;


elseif mp > mt;
        A1 = TROUGHS;
        A2 = PEAKS;

    for l = 1:mt;
        a = A1(l,:);
        for j = 1:mp;
                b = A2(j,:);
                diff1(j,1) = abs(a(1,1) - b(1,1));
                if diff1(j,1) < 6;
                        eval(['c' int2str(j) ' = b;']);
                else;
                        eval(['c' int2str(j) ' = 0;']);
                end;
        end;

        for j = 1:mp;
                eval(['d = c' int2str(j) ';']);
                if d > 0;
                        diff2(j,1) = abs(a(1,2) - d(1,2));
                else;
                        diff2(j,1) = 1000;
                end;
        end;


        min_diff = min(diff2(:,1));

        for h = 1:mp;
                if diff2(h,1) == min_diff;
                        row = PEAKS(h,:);
```

```
                    end;
                end;
                PEAKS_r = [PEAKS_r;row];
        end;

        TROUGHS_r = TROUGHS;
end;


% Determine whether the peaks of troughs appear first for later use
% in constructing colonies from single defects.

if PEAKS_r(1,2) < TROUGHS_r(1,2);
        FIRST = PEAKS_r;
        SECOND = TROUGHS_r;
else;
        FIRST = TROUGHS_r;
        SECOND = PEAKS_r;
end;



% Plot the defect positions

[m,n] = size(D);

for k = 1:m;
        for j = 1:n;
                pos(k,j) = 0;
        end;
end;

for h = 1:m;
        y(1,h) = (h-1)*180/(m-1) + 0;
end;

for h = 1:n;
        x(1,h) = (h-1)*500/(n-1) + 70;
end;

[m,n] = size(FIRST);

for l = 1:m;
        a = FIRST(l,1);
        for k = FIRST(l,2):SECOND(l,2);
                pos(a,k) = 1;
        end;
end;


figure(6);
colormap(gray(2));
surf(x,y,pos);
view(90,-90);
xlabel('Length (mm)');
ylabel('Width (mm)');
print -deps fig416
```

```
%FIRST(:,1) = FIRST(:,1) + 114;
%FIRST(:,2) = (FIRST(:,2) * 2.45) + 155;
%SECOND(:,1) = SECOND(:,1) + 114;
%SECOND(:,2) = (SECOND(:,2) * 2.45) + 155;


%***********************************************************************
%*          Matlab Routine for Detecting Stress Corrosion        *
%*                   Sites                *
%***********************************************************************

clear;

% Load the Bx and Bz data file to be classified
load penp_2abz.txt;

% Invert the file so that x is the horizontal axis
A = penp_2abz(101:1100,:)';


% Convert values in matrix to between 0 and 1
B = (A - 1930)/120;

% Adjust the image size
D = imresize(B, [50 100], 'nearest', 0);

% Invert the data in D to assist with trough location
E = (D * (-1)) + 1;

% Display the Bz Data converting it to a 200x100 image
figure(1);
imshow(D,256);
title('Bz Contour Plot')

clear B C;

%*****************************************************
%*          Peak and Trough Identification        *
%*****************************************************

% This routine looks in eight locations around a pixel to determine if
% if the pixel in the point of interest is brighter than its neighbours


% Set the threshold value to detect the bright areas in the Bz image

p_thresh = 0.3;


% It is not essential that all eight locations satisfy this criteria
% The percentage of directions which should satisfy the criteria is:

per_locations = 0.9;


%***********************************************************************

% The following are the direction masks to determine if the threshold
```

```
% criteria is met.  For ease, compass point locations are used

east = [   0 0 0 0 0 -1 -1;
           0 0 0 6 0 -1 -1;
           0 0 0 0 0 -1 -1];

west = [  -1 -1 0 0 0 0 0;
          -1 -1 0 6 0 0 0;
          -1 -1 0 0 0 0 0];

north = west';
south = east';

nwest = [ -1 -1 -1  0  0 0 0;
          -1 -1  0  0  0 0 0;
          -1  0  0  0  0 0 0;
           0  0  0  6  0 0 0;
           0  0  0  0  0 0 0;
           0  0  0  0  0 0 0;
           0  0  0  0  0 0 0];

swest = [  0  0  0  0  0 0 0;
           0  0  0  0  0 0 0;
           0  0  0  0  0 0 0;
           0  0  0  6  0 0 0;
          -1  0  0  0  0 0 0;
          -1 -1  0  0  0 0 0;
          -1 -1 -1  0  0 0 0];

neast = swest';
seast = nwest';

%*************************************************************************
% Staring with the peaks in the signal

% Use a median filter to remove excess noise from the Bz data

Bz_peaks = medfilt2(D);

% Convolve the image file with each direction mask and record the directions
% which satisfy p_thresh

% North
peak = conv2(Bz_peaks, north, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = p_sig;

% South
peak = conv2(Bz_peaks, south, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% East
```

```
peak = conv2(Bz_peaks, east, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% West
peak = conv2(Bz_peaks, west, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;


% North_East
peak = conv2(Bz_peaks, neast, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;


% North-West
peak = conv2(Bz_peaks, nwest, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% South-East
peak = conv2(Bz_peaks, seast, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% South-West
peak = conv2(Bz_peaks, swest, 'same');
p_sig = im2bw(peak, p_thresh);
num_dir = num_dir + p_sig;

% Plot the high areas
figure(2);
imshow(num_dir/8,256);
title('High Areas');

% Filter out the peaks based upon the criteria of the number of
% directions surrounding the pixels

p_sig = im2bw(num_dir/8, per_locations);

% Plot the final results
figure(3);
imshow(p_sig,2);
title('High Areas');

clear num_dir;

%**************************************************************
% Now identify the troughs

% Use a median filter to remove excess noise from the Bz data

Bz_troughs = medfilt2(E);


% Convolve the image file with each direction mask and record the directions
% which satisfy p_thresh
```

```
% North
peak = conv2(Bz_troughs, north, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = t_sig;

% South
peak = conv2(Bz_troughs, south, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% East
peak = conv2(Bz_troughs, east, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% West
peak = conv2(Bz_troughs, west, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;


% North_East
peak = conv2(Bz_troughs, neast, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;


% North-West
peak = conv2(Bz_troughs, nwest, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% South-East
peak = conv2(Bz_troughs, seast, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% South-West
peak = conv2(Bz_troughs, swest, 'same');
t_sig = im2bw(peak, p_thresh);
num_dir = num_dir + t_sig;

% Plot the low areas
figure(4);
imshow(num_dir/8,256)
title('Low Areas')

% Filter out the peaks based upon the criteria of the number of
% directions surrounding the pixels which must be lower

t_sig = im2bw(num_dir/8, per_locations);

% Plot the final results
figure(5);
imshow(t_sig,2)
title('Low Areas');

clear north south east west neast nwest seast swest;
```

```
clear per_locations p_thresh num_dir peak;

%****************************************************************
% Find the centers of the peak and trough locations

% Peaks
[p1,p2] = center(p_sig);

% Troughs
[t1,t2] = center(t_sig);


% Combine the peak and trough locations into indivdual matrices
P = [p1;p2]';
T = [t1;t2]';

% Order the matrices in ascending order according to the Y position
[Y1,Ip] = sort(P(:,1));
[Y2,It] = sort(T(:,1));

[m,n] = size(P);

for j = 1:m;
        r = Ip(j);
        Y1(j,2:n) = P(r,2:n);
end;

[m,n] = size(T);

for j = 1:m;
        o = It(j);
        Y2(j,2:n) = T(o,2:n);
end;

% Make sure that each indicated area is a peak or trough and not just
% signal noise.

[m,n] = size(Y1);
count = 1;

for k = 1:m;
        Bz = abs(D(Y1(k,1),Y1(k,2)) - D(Y1(k,1),1));
        BZ = (Bz*140);
        if BZ >= 15;
                PEAK(count,:) = Y1(k,:);
                count = count+1;
        end;
end;

[m,n] = size(Y2);
count = 1;

for k = 1:m;
        Bz = abs(D(Y2(k,1),Y2(k,2)) - D(Y2(k,1),1));
        BZ = (Bz*240);
        if BZ >= 15;
                TROUGH(count,:) = Y2(k,:);
                count = count+1;
```

```
        end;
end;


PEAKS = round(PEAK);
TROUGHS = round(TROUGH);


clear Y1 Y2 t_sig p_sig P T p1 p2 t1 t2 Ip It Bz_peaks Bz_troughs;



% Determine which of the 2 matrices is smaller and use it to match
% peaks with troughs

[mp, np] = size(PEAKS);
[mt, nt] = size(TROUGHS);


if mp <= mt;
        A1 = PEAKS;
        A2 = TROUGHS;

        for l = 1:mp;
                a = A1(l,:);
                for j = 1:mt;
                        b = A2(j,:);
                        diff1(j,1) = abs(a(1,1) - b(1,1));
                        if diff1(j,1) < 6;
                                eval(['c' int2str(j) ' = b;']);
                        else;
                                eval(['c' int2str(j) ' = 0;']);
                        end;
                end;

                for j = 1:mt;
                        eval(['d = c' int2str(j) ';']);
                        if d > 0;
                                diff2(j,1) = abs(a(1,2) - d(1,2));
                        else;
                                diff2(j,1) = 1000;
                        end;
                end;

                min_diff = min(diff2(:,1));

                for h = 1:mt;
                        if diff2(h,1) == min_diff;
                                row = TROUGHS(h,:);
                        end;
                end;
                TROUGHS_r = [TROUGHS_r;row];
        end;

        PEAKS_r = PEAKS;


 elseif mp > mt;
```

```
			A1 = TROUGHS;
			A2 = PEAKS;

	for l = 1:mt;
			a = A1(l,:);
			for j = 1:mp;
					b = A2(j,:);
					diff1(j,1) = abs(a(1,1) - b(1,1));
					if diff1(j,1) < 6;
							eval(['c' int2str(j) ' = b;']);
					else;
							eval(['c' int2str(j) ' = 0;']);
					end;
			end;

			for j = 1:mp;
					eval(['d = c' int2str(j) ';']);
					if d > 0;
							diff2(j,1) = abs(a(1,2) - d(1,2));
					else;
							diff2(j,1) = 1000;
					end;
			end;


			min_diff = min(diff2(:,1));

			for h = 1:mp;
					if diff2(h,1) == min_diff;
							row = PEAKS(h,:);
					end;
			end;
			PEAKS_r = [PEAKS_r;row];
	end;

	TROUGHS_r = TROUGHS;
end;


% Determine whether the peaks of troughs appear first for later use
% in constructing colonies from single defects.

if PEAKS_r(1,2) < TROUGHS_r(1,2);
	FIRST = PEAKS_r;
	SECOND = TROUGHS_r;
else;
	FIRST = TROUGHS_r;
	SECOND = PEAKS_r;
end;



% Plot the defect positions:

[m,n] = size(D);

for k = 1:m;
	for j = 1:n;
```

```
                    pos(k,j) = 0;
          end;
end;

for h = 1:m;
          y(1,h) = (h-1)*180/(m-1) + 0;
end;

for h = 1:n;
          x(1,h) = (h-1)*500/(n-1) + 70;
end;

[m,n] = size(FIRST);

for l = 1:m;
          a = FIRST(l,1);
          for k = FIRST(l,2):SECOND(l,2);
                    pos(a,k) = 1;
          end;
end;


figure(6);
colormap(gray(2));
surf(x,y,pos);
view(90,-90);
xlabel('Length (mm)');
ylabel('Width (mm)');
```

M-file subroutines: center.m cclabel.m repmat.m

Purpose: Required by M-files performing peak detection (these files were not written by author)

---

center.m

```
function [centx,centy]=center(BW)
% Label and locate blobs
labeled=cclabel(BW);
all_objects=max(max(labeled));
[rows cols]=size(BW);

sum(1:all_objects)=zeros(size(1:all_objects));
centx(1:all_objects)=zeros(size(1:all_objects));
centy(1:all_objects)=zeros(size(1:all_objects));

for i=1:rows
   for j=1:cols
      if labeled(i,j)~=0
         sum(labeled(i,j))=sum(labeled(i,j))+1;
         centx(labeled(i,j))=centx(labeled(i,j))+i;
         centy(labeled(i,j))=centy(labeled(i,j))+j;
      end
   end
end
```

```
for c=1:all_objects
   centx(c)=centx(c)/sum(c);
   centy(c)=centy(c)/sum(c);
end
```

cclabel.m

```
function L = cclabel(BW,mode)
%CCLABLE label connected componebts.
%   CCLABEL(BW), where BW is a matrix containing 0's and 1's,
%   returns a matrix the same size containing labels for the
%   8-connected components in BW.
%
%   CCLABEL(BW,4) returns a label matrix for the 4-connected
%   components in B.
%
%   Steven L. Eddins, August 1995
%   Copyright (c) 1993-1996 by the MathWorks, Inc.
%   $Revision: 1.2 $ $Date: 1995/09/06 15:06:13 $
%
%   - Copied from ftp://ftp.mathworks.com/pub/mathworks....
%   on March 6, 1996 (Tish)

if (nargin<2)
   mode = 8;
end

[M,N] = size(BW);

%
% Compute run-length encoding
%
rowZeros = zeros(1,size(BW,2));
BW = [rowZeros; BW ; rowZeros];
d = diff(BW);
[sr,sc] = find(d == 1);          % start row and column indices
[er,ec] = find(d == -1);         % end row and column indices
numRuns = length(sr);
if (numRuns == 0)
   runs = [];
else
   runs = [sc sr (er-1) ];
end

%
% First labeling pass
%
labels = zeros(numRuns,1);
equivTable = [];
nextEquivTableIdx = 1;
currentLabel = 1;
for k = 1:numRuns
   column = runs(k,1);
   rowStart = runs(k,2);
   rowEnd = runs(k,3);
   idx = find(runs(:,1) == column - 1); % find runs on previous column
   if (isempty(idx))
```

```
    labels(k) = currentLabel;
        currentLabel = currentLabel + 1;
  else
        if (mode == 8)
           % find 8-connected objects
           overlaps = find(((rowEnd >= (runs(idx,2)-1)) &...
                            (rowStart <= (runs(idx,3)+1))));
        else
           overlaps = find(((rowEnd >= runs(idx,2)) &...
                            (rowStart <= runs(idx,3))));
        end
        if (isempty(overlaps))
           labels(k) = currentLabel;
           currentLabel = currentLabel + 1;
        elseif (length(overlaps) == 1)
           labels(k) = labels(idx(overlaps));
        else
           labels(k) = labels(idx(overlaps(1)));
           for n = 2:length(overlaps)
                 if (labels(idx(overlaps(1))) ~= labels(idx(overlaps(n))))
                    if (nextEquivTableIdx > (size(equivTable,1)+1))
                          equivTable = [equivTable; zeros(size(equivTable))];
                    end
                    equivTable(nextEquivTableIdx:nextEquivTableIdx+1,:) = ...
                          [labels(idx(overlaps(1))) labels(idx(overlaps(n)));
                          labels(idx(overlaps(n))) labels(idx(overlaps(1))) ];
                    nextEquivTableIdx = nextEquivTableIdx + 2;
                 end
           end
        end
   end
end
numLabels = currentLabel - 1;
equivTable = [equivTable(1:nextEquivTableIdx-1,:);repmat((1:numLabels)',1,2)];

%
% Equivalence class resolution
%
A = sparse(equivTable(:,1), equivTable(:,2), ones(size(equivTable,1),1), ...
      numLabels, numLabels);
[p,p,r,r] = dmperm(A);

sizes = diff(r);                 % Sizes of components, in vertices.
numObjs = length(sizes);         % Number of components.

% Now compute an array "blocks" that maps vertices of A to components;
% First, it will map vertices of A(p,p) to components...

blocks = zeros(1,numLabels);
blocks(r(1:numObjs)) = ones(1,numObjs);
blocks = cumsum(blocks);

% Second, permute it so it maps vertices of A to components.
blocks(p) = blocks;

%
% Fill in each run with a label
%
```

```
L = zeros(M,N);
for k = 1:numRuns
  column = runs(k,1);
  rowStart = runs(k,2);
  rowEnd = runs(k,3);
  value = blocks(labels(k));
  L(rowStart:rowEnd,column) = value(ones(rowEnd-rowStart+1,1),1);
end
```

repmat.m

```
function B = repmat(A,M,N)
%REPMAT Replicate a matrix by factors of M and N.
%        B = REPMAT(A,M,N) replicates the matrix A to produce the
%        M-by-N block matrix B.
%
%        Example:
%           repmat(magic(2),2,3)

%        Copyright (c) 1995 by The MathWorks, Inc.
%        $Revision: 1.2 $  $Date: 1994/12/22 15:44:57 $

if nargin < 2
  error('Requires at least 2 inputs.')
elseif nargin == 2
  N = M;
end
[m,n] = size(A);
mind = (1:m)';
nind = (1:n)';
mind = mind(:,ones(1,M));
nind = nind(:,ones(1,N));
B = A(mind,nind);
```

M-file: acorrz.m

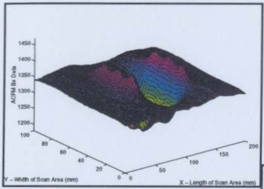
Purpose: Used to calculate the autocorrelation of B_z signals and plot the results

---

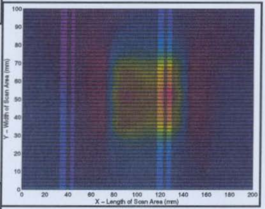
```
% Autocorrelation routine used on teh Bz data

clear

load bz13.txt;

A = bz13';

C = A - mean(mean(A));

D = xcorr2(C);


[m,n] = size(D);

% Begin Surface Plotting routine
```

```
u = (n/20);
v = (u - 0.5);
v = round(v);

for p = 1:m;
        count = 1;
        for q = 1:v;
                B(p,q) = D(p,count);
                count = count + 20;
        end;
end;

for k = 1:v;
        x(1,k) = (k-1)*200/(v-1);
end;

for l = 1:m;
        y(1,l) = (l-1)*100/(m-1);
end;

figure(1);
surf(x,y,B);
grid off;
xlabel('Length of Scan Area (mm)');
ylabel('Width of Scan Area (mm)');
zlabel('Autocorrelation');
title('Autocorrelation of Plate 1 Bz Signal');
```

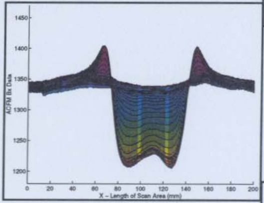
MACHINED COLONY 2



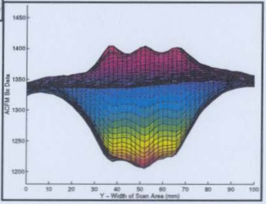
B_x Isometric



B_x Contour Plot



B_x View Along Length of Scan Area



B_x View Along Width of Scan Area