Methods for Computer Aided Inspection of Geometric Tolerances

By

Luiz C. R. Carpinetti

Submitted for the degree of Doctor of Philosophy
to the Higher Degrees Committee
University of Warwick

Department of Engineering
University of Warwick, Coventry, U. K.

October 1993

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor, Dr. D. G. Chetwynd, for his support and encouragement during my period of research and his many suggestions and comments on this piece of work, and also for his help in correcting the text.

I would also like to express my gratitude to "Conselho Nacional de Desenvolvimento Cientifico e Tecnologico" (CNPq), for making this work possible.

There are many technicians of the Microengineering and Metrology laboratory to whom I would also like to express my thanks: Dave Robinson for his help with data acquisition and transfer, and Steve Wallace, Rhod Mortimore and Frank Courtney for their general help throughout my period of study at the University of Warwick, and for the cheerful moments in the "tea club".

Finally, my thanks also go to some fellow PhD students (and now Drs): Dr. Ralph Dale-Jones, Dr. M. Li, Dr. P. Gosling, S. Farrall and L. Davis.

This thesis is dedicated to my parents, my wife, Solange, and my daughter Julia.

Declaration

This thesis is presented in accordance with the regulations for the degree of Doctor of Philosophy by the Higher Degree Committee at the University of Warwick. The thesis has been composed and written by myself based on the research undertaken by myself. The research materials have not been submitted in any previous application for a higher degree. All sources of information are specifically acknowledged in the content.

L. C. R. Carpinetti

Summary

This thesis investigates computational methods for assessing tolerance specifications of geometric features in a context of computer aided inspection. It is concerned with checking the sampled features for containment within tolerance zones specified at the design stage, not with explicit shape measurement. The significance of this difference is highlighted when two or more features are to be inspected in combination. The approach adopted is to express the tolerance information as a set of inequality constraints and then to seek efficient methods for determining the feasibility of the set, that is whether all the constraints can be simultaneously satisfied.

Roundness inspection is used to introduce all the concepts of the new formulations. By linearisation of the constraints, a standard approximation in roundness measurement, a new algorithm is implemented which provides a "GO-NOGO" result of inspection by checking for feasibility in a highly efficient way. This algorithmic approach is then extended to other inspection situations where naturally linear constraints or valid linearisation occur.

Since there are many inspection cases where linearisation is not appropriate, non-linear optimisation techniques are then investigated for their effectiveness in feasibility testing. The inspection of arrays of circular features is used here as a typical test case. Genetic search methods are explored as a possible alternative to formal non-linear programming and guidelines for their efficient use for this problem are proposed. These methods are then compared and contrasted with formal methods, particularly generalised reduced gradient (GRG) and sequential quadratic programming (SQP).

The linear algorithm is shown to be the most efficient when it can be used, although all techniques were fast enough for on-line use with modest sized data sets. Currently all the non-linear methods are too expensive for routine use on large data sets. GRG is recommended as having the most favourable combination of good and bad features, but there is some evidence that genetic search might be relatively more efficient for more complex inspection problems.

Chapter 1

Introduction

1.1 Introduction

Tolerancing questions in mechanical design are of great importance in the move to improve quality in manufacturing products. At the design stage, a part is specified in terms of sizes and shapes which are in some sense ideal for its functional requirements. Due to variability inherent in the manufacturing process, the designer will also have to specify how far the manufactured product can depart from its ideal specification. This is done by specifying various tolerances associated with the design, following standardised engineering practice. The width of the tolerance limits (or allowed error of any feature) defines the required precision in manufacturing. Since manufacturing to high precision is generally a costly exercise, it is good practice to design for as large a tolerance as is possible without seriously affecting the quality and functionality of the part.

At the manufacturing stage, parts will need to be checked against their design specifications. On routine inspection, the information obtained at this stage will be used to reject parts which do not meet specifications, on a "pass-fail" basis. In a computer aided inspection setting, a part should ideally be inspected by comparing a collection of digitised data points representing a part surface with its design and tolerancing specification. Thus, the inspection process gives rise to the question of how data points representing a part can be compared with its

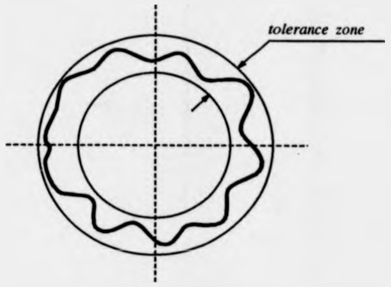


Figure 1.1: Geometric tolerance for circularity.

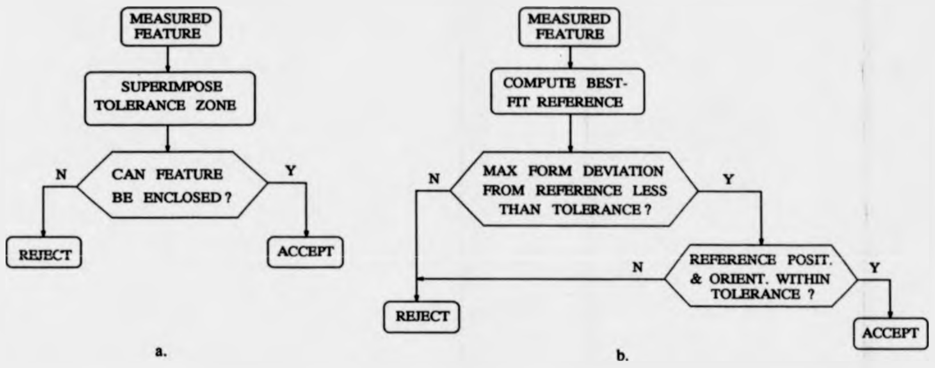
tolerance specification in a efficient manner.

1.2 The Inspection Problem

Geometric tolerances are defined and classified in design standards [e.g. BSI, 1990], including form, orientation and position tolerances. Common features are roundness, flatness, squareness, concentricity, amongst others. For example, the tolerance for out of roundness is defined as the annular space between two concentric circles lying in the same plane, figure 1.1. A component is within tolerance if its profile, or the profile of the cross section whose error is to be controlled, is enclosed by these two circles. Thus, the essential objective of inspection is to check for containment of a feature within its tolerance zone.

On the other hand, as, over the past forty years, form error measurement has become recognised as an important discipline in its own right, there has been a strong concentration in the literature on algorithms for form measurements, notably roundness and flatness errors. The measurement of out of roundness, for instance, is by measuring the peak to valley deviation of the actual profile from a reference circle fitted to that profile. Metrology standards for out of roundness measurement [e.g. BSI, 1987] recommend that an optimal reference figure be used, in the sense some measure of the interaction between the reference and the profile is maximised or minimised.

Algorithms for estimating reference figure fits for shape measurements use

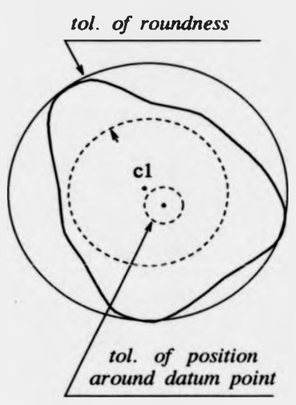


MEASURED FEATURE
SUPERIMPOSE TOLERANCE ZONE
N
CAN FEATURE BE ENCLOSED ?
Y
REJECT
ACCEPT
a.
MEASURED FEATURE
COMPUTE BEST-FIT REFERENCE
N
MAX FORM DEVIATION FROM REFERENCE LESS THAN TOLERANCE ?
Y
N
REFERENCE POSIT. & ORIENT. WITHIN TOLERANCE ?
Y
REJECT
ACCEPT
b.

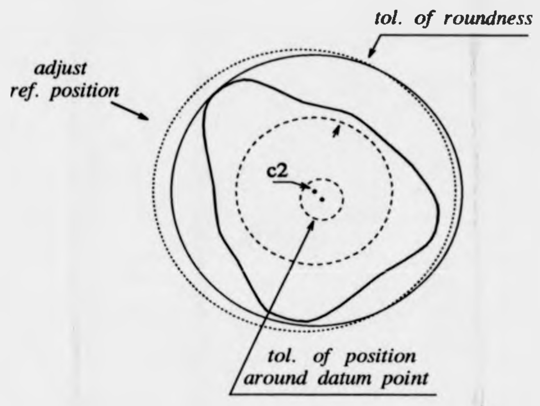
Figure 1.2: Inspection procedures: according to design standards (a)and normal practice (b).

either a least-squares approach or, based on metrological requirements, an optimisation of a geometric parameter of the figure. Thus in practice it is usual, since standard algorithms exist, first to find an optimal reference for form measurement and test the deviation of the feature from it against the design specifications. The position and orientation of the feature, if they also are toleranced, are then tested by checking whether the parameters of the best-fit reference are adequately close to design specifications. This approach is illustrated in figure 1.2b, while figure 1.2a illustrates the way suggested by standards that a toleranced feature should be assessed

This difference of approach is not significant when geometric tolerances of form only are being inspected, but it may lead to different results when geometric tolerances of orientation or position have also to be met. In order to illustrate this point, consider that a circular profile, illustrated in figure 1.3, has to be inspected for circularity and centre position. This is the case of a crank shaft for example, where in addition to the requirement of roundness of each individual bearing, they must be aligned axially. A concentricity tolerance is therefore specified, so as to limit the deviation of the position of the centre or axis of the toleranced feature from its true position, i. e. the centre or axis of the datum feature. However, when the section is not perfectly circular, the definition of centre becomes to some extent a matter of interpretation and in practice the reference centre is



a.



b.

Figure 1.3: Minimum circumscribing radius reference (a), against another circumscribing reference (b).

used.

Considering figure 1.3a, the point **c1** represents the centre position of the minimum radius circumscribing circle (MCC), a standard best-fit reference commonly used for shaft measurement. Point **c2**, in figure 1.3b, represents the centre position of another reference circle circumscribing the profile but not with minimum radius. It can be assumed, for the sake of illustration, that using the MCC as the reference figure the out of roundness is within tolerance but its centre position is eccentric in relation to the datum point by a distance over the allowed tolerance and so it fails inspection. On the contrary, the other reference, figure 1.3b, as it is not minimum radius, has some room for moving around the feature, and so it can be "brought" closer to the datum point, within the tolerance zone. The resulting out of roundness, although larger than the error obtained using the MCC reference, is still lower than its tolerance limit and overall the features pass inspection.

Thus, an optimum reference figure originally defined for form measurement, as is the case of MCC for roundness, may not be appropriate when the position and/or orientation of the feature has also to be inspected.

In order to simulate the graphical procedure outlined above, the general tolerance assessment problem can be formulated as a constrained optimisation problem, as it is sometimes done for shape measurement, with the difference that form, position and orientation constraints of features imposed by design specifications are to be considered in the optimisation process. It is assumed that the ideal design (on a engineering drawing or computer-aided-design (CAD) system, for example) can be represented as a union of geometric elements each represented as a parameterised curve or surface defined in a common reference system. For example, a circle in two dimensions is parameterised by its centre position and radius. The geometric tolerance assessment problem is thus to find a combination of such parameters which defines a reference that does not violate the parameter constraints imposed by dimensional, orientation or position tolerances and at the same time does not violate the form constraints imposed by form tolerances. A set of parameters satisfying these constraints will be a feasible solution, meaning in practical terms that the design tolerance specifications are satisfied and the component passes inspection.

A supplementary problem, derived from the assessment problem as posed above, is to give a measure of the extent to which the tolerance constraints have been met. This remains a different problem from finding optimum reference figures for shape measurement, which does not consider in the optimisation process the parameter and form constraints derived from design specification. As a consequence, the solution of the latter may be outside the feasible region defined by the assessment problem, leading to the situation illustrated in figure 1.3.a.

This work advocates that most of the geometric tolerance inspection problems can be solved by examining the feasibility of an associated optimisation problem, with no need to consider its iteration towards optimality, and thereby simulating the action of a GO-NOGO inspection gauge.

Another inspection situation that well illustrates the point discussed here is the combination of flatness and squareness. This is the case of a precise base and vertical slideway for instance. The slideway surface has to be enclosed by two parallel planes separated from each other by the squareness tolerance value, and perpendicular to the base, which is assumed as the datum surface. The

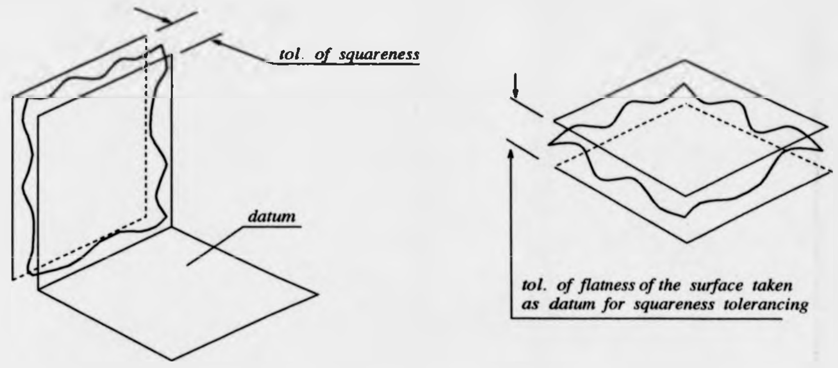

tol. of squareness
datum
tol. of flatness of the surface taken as datum for squareness tolerancing

Figure 1.4: Tolerance zones of squareness and flatness.

flatness error of the slideway is implicitly controlled by the squareness tolerance. In addition to this, the base, which is assumed to be the datum, has also to be flat within a limit defined by its flatness tolerance. These geometric tolerance zones are pictured in figure 1.4.

Flatness measurement is usually done by measuring the peak to valley deviation of the measured data points from a best-fit reference, commonly defined by either least squares or a minimum zone approach. It is unclear how squareness measurement is done in practice and the same sort of comment made before applies to this case: when the surfaces are not perfectly flat, the orientation of the surfaces is rather vague and in practice the orientation of the reference planes are used. Some commercial software for coordinate measuring machines (CMMs) fits least squares planes to the set of measured points representing the surfaces and then measures the angle between them. Although this gives a measure of the perpendicularism of the two surfaces, the result is not accurately comparable with the squareness tolerance specification in design. Moreover, this method gives no information about the flatness of the surfaces. It is also unclear how the datum should be defined.

Instead, it is better that the squareness and flatness errors should be inspected simultaneously for containment within a squareness and flatness *frame.* This procedure can again be formulated as a constrained optimisation, where the objective is to find a combination of parameters for each reference such that the form, position and orientation constraints derived from the design specifications are not violated. In other words the inspection is successful if a feasible solution to the constrained problem is found.

The general inspection problem can be enunciated as: find the parameters of a geometric reference figure (or figures) $\mathbf{C}$ for a set of data points $Y_i, i = 1, \cdots, N$, where N is the number of data points, such that they satisfy the form constraints

$$F(Y_i, \mathbf{C}, \mathbf{t}) \sim 0 \tag{1.1}$$

and the parameter constraints

$$P(\mathbf{C}, \mathbf{t}) \sim 0 \tag{1.2}$$

where $\sim$ *is in* $\{\geq; \leq\}$, and $\mathbf{t}$ are the tolerance specifications from design.

The general behaviour of constrained optimisations is conveniently examined in terms of their geometrical structure in a multi-dimensional "parameter space" in which each parameter is plotted along an orthogonal axis (parameter space representation is further used in chapter 3). The constraints are hyper-surfaces in parameter space, each one dividing the parameter space in two open half spaces, with all the feasible parameter combinations lying to one side and the infeasible ones to the other side. The combination of all the constraints will or will not define a feasible region of solutions in parameter space. The absence of a feasible region indicates that the constraints cannot be satisfied simultaneously. Thus the tolerance assessment problem is to determine whether a feasible region exists. The discovery of any solution that lies in feasible region is sufficient to pass the inspection process.

Depending on the problem in consideration and how it is modelled, the inequality constraints can be linear or non-linear functions of the parameters. If the constraints are linear in their parameters then the optimisation problem can be solved using linear programming techniques. Highly efficient techniques for measurements such as roundness, flatness and straightness have been known for some years in which a local parameter linearisation is used within an "exchange algorithm" (see section 2.4.4). The exchange algorithms ultimately derive from the theory of linear programming, which both guarantees their convergence onto a correct solution and demonstrates the source of their computational efficiency. The geometrical formulation of the fitting problems normally has many constraining equations and few parameters. Optimisations of this "shape" can be solved rapidly by using their dual formulation and the exchange algorithms are also a type of dual programming even though they may be understood geometrically. However, their dual nature becomes a major disadvantage if we attempt to use the exchange algorithms in the simultaneous inspection of several geometric features in order to see whether they simultaneously meet acceptable tolerances.

This work starts by discussing a new approach for the inspection of geometric characteristics of features in which the ultimate position is to provide a set of constraints from which the required inspection information can be obtained by examining only the feasibility of the mathematical programme, with no need to consider its iteration towards optimality. Whenever linearisation of the geometric formulation is adequate, this may be achieved using an existing, though not all that widely known, algorithm to explore feasibility, so guiding the GO-NOGO decision. Methods of tuning the process to maximise computational efficiency are introduced.

Although linearising the constraints allows special algorithms of reasonable efficiency to be applied, this is not always possible or appropriate. For example, when the same approach is extended to the inspection of form, dimension and relative position of several circular features, as is the case of circular mating features, the gain in computer efficiency for using linear functions in the formulation of the inspection problem may not compensate the losses in accuracy of using such a model. In this case the trade-off is unclear between accepting an approximation in the formulation of the constraints or using accurate constraints but taking more effort to obtain a potentially more definitive result. A method that can cope with non-linear models has a much wider application and therefore, non-linear methods are sought that will perform well with the level of computer power likely to be dedicated to an individual measuring machine. Some form of non-linear optimisation will be needed but, since here we are concerned only with feasibility testing, it is not clear which approaches will be most effective. Non-formal methods may be advantageous in this context. Genetic Search methods are explored for applicability to the determination of the feasibility of the inspection problem. Finally, the effectiveness of such non-formal techniques is compared with that of conventional non-linear programming. Two non-linear programming methods are investigated, namely the generalised reduced gradient (GRG) and the Sequential Quadratic Programming (SQP) methods.

1.3 Organisation of the Thesis

Chapter two contains a review of computer aided precision metrology. It starts by presenting some background information about the most common and pertinent measuring equipment and some of the basic rules to be observed in designing software for metrology. Then, it presents a more detailed review of methods and algorithms in the literature for shape measurement as well as to the assessment of geometric tolerances. Finally, tolerancing techniques using solid modelling are briefly reviewed and discussed.

Chapter three introduces the concepts of the new formulations by discussing the problem of inspection of roundness error (in contrast with measurement of roundness error) in combination with position errors, and the idea of GO-NOGO inspection software. Roundness inspection also provides examples where local linearisation can be very effective. The primal and dual formulation of the inspection problem are discussed and a new algorithm is implemented to explore the feasibility of the inspection problem in a more efficient way. The basis of the algorithm is given along with practical examples of how it works. The results of performance tests on these algorithms are presented and discussed.

Chapter four discusses the problem of inspection of combined form and attitude errors, such as squareness or parallelism and flatness. It formulates these inspection problems in a manner applicable to the algorithm discussed in chapter three. The results of tests on these formulations and algorithms are presented and discussed.

Chapter five further explores the approach discussed in the previous chapters to the inspection of dimensions and geometry of circular mating features. It discusses the limitation of linear methods and investigates the use of Genetic Search methods for assessing geometric tolerance errors in cases where a linear approximation is not an adequate model to the truly non-linear formulation of the inspection problem. A Genetic Search model is set up and suitable values for its control parameters are explored experimentally. Two model problems are used to investigate its effectiveness and practicality.

Chapter six investigates the applicability of some formal non-linear programming methods for inspecting geometric tolerance errors. Two methods are considered, the generalised reduced gradient (GRG) and the Sequential Quadratic Programming (SQP) methods. The inspection cases used in the previous chapter are used in this chapter as well, so as to provide a direct comparison in terms of effectiveness between formal and non-formal methods.

Chapter seven presents an overall discussion of the relative merits of the methods considered in this work as well as suggestions for further work.

Chapter 2

A Review of Computer Aided Precision Metrology

2.1 Introduction

Measurement methods and equipment can vary widely depending on the application. However they can be generally classified as: with or without contact; absolute or comparative measurement. Most measurements in mechanical metrology are performed through contact with the workpiece by means of a tip or probe. Instruments in this category have developed from the earliest ones such as vernier calipers, to the sophisticated present generation of Coordinate Measuring Machines (CMMs). In contrast to this, non-contact measurement does not use any type of mechanical probe. All optical instruments are encompassed in this category, ranging from the most common ones such as Profile Projectors to modern Vision Machines and extremely precise instruments such as Laser Interferometers. Moreover, the measurement can be absolute, when the given information is the dimension itself or comparative, when the information is in terms of departure from some reference system. In this case, instruments are usually supported by some amplifying system; this is the case of mechanical comparators or more sophisticated equipment such as roundness, flatness and surface texture measuring instruments.

Every computer aided metrology system comprises a measuring equipment interfaced with computer hardware and software. The software assists the system in performing its operations, such as controlling the positioning of moving parts, data acquisition and evaluation of the measuring data. Evaluation of the measuring data is basically concerned with the mathematical modelling and numerical manipulation of the data. The speed and accuracy of the software in performing these operations affect very much the performance of an instrument and therefore are regarded as matters of great importance.

This chapter presents some background information about the most common and pertinent measuring equipment and some of the basic rules to be observed in designing software for metrology. Then it moves on to a thorough review of methods for shape measurement, with emphasis on roundness and flatness measurements. Finally, it gives some account of tolerancing techniques using solid modelling theory and optimisation techniques for geometric tolerance assessment problems.

2.2 Measuring Equipment

2.2.1 Coordinate Measuring Machines (CMMs)

Coordinate Measuring Machines have gained considerable importance in meeting the inspection requirements of modern manufacturing technology. They bring to inspection advantages such as flexibility and higher throughput. Typically, the basic machine comprises a table with a bridge that traverses its length and a vertical spindle, with a probe holder located at the end of it, which traverses the bridge and in turn moves axially. Thus an orthogonal reference system is so defined by the table and movable components. Different configurations of CMMs exist but they all follow the same principle.

Three-dimensional measurement data is provided by using a probe, usually of the "touch trigger" type, attached to the probe holder, which is brought into contact with the workpiece resting on the table. The probe has as stylus which, when it touches the workpiece, is deflected to switch the probe on/off and a binary signal is sent to the computer control, which in turn records the position of the probe from the axial measuring system, usually by incremental gratings. Other types of probe include a scanning contact probe which senses the workpiece by moving across its surface remaining in contact with it [Jacoby and Lenz, 1980], and more recently, a non-contact optical probe [Treywin and Edwards, 1987].

Once the coordinates of the probe are recorded, the computer software will then manipulate the data. This possibly involves: transforming the data from the machine coordinate system into another system, e.g. a local system referred to the workpiece; compensating for systematic errors [e.g. Burdekin, et al, 1985]; and finally evaluating dimensions, deviations from shape and orientation.

2.2.2 Vision Technology

Machine Vision systems are currently being used in looking for defects in parts, ensuring correct configuration of parts in assembly lines and measurements of dimensions and shapes [Ventura, Chang and Klein, 1988]. Applications of machine vision in automated inspection cells have been demonstrated in the automotive industry [Pastorius, 1989 and Mullins, 1987]. It offers advantages such as high flexibility and high speed of measuring. In a typical arrangement, the camera viewing area is divided into a matrix of picture elements called pixels. Once the image of a part is sampled in each pixel, its signal is sent to the processor and treated by image analysis techniques.

The accuracy and precision of vision systems tends to be limited. The resolution of a machine vision system is defined by the size of one picture element, or pixel of image. As it is the image of a part that is measured, the higher the magnification used, the higher the basic accuracy of the measurement. However, there is a limit on the optical magnification. In addition to this, the correct location of the edges of a part feature is the main factor in determining the accuracy of a system. The influence of these accuracy related problems is quantified in a study presented by Ma et al [1988].

2.2.3 Surface Metrology Instruments Using Stylus Techniques

The assessment of surface form and texture by stylus methods has been in use since the early forties and has gained world wide acceptance due to its advantages such as robustness and high accuracy.

Stylus types of instruments are characterised by the use of a sharply pointed stylus which is rested lightly on the surface of the part and is carefully traversed across it. The stylus, having a radius tip of typically about 1mm for roundness and straightness measurement, is connected via a bar to a transducer which measures the up and down movements of the stylus due to the variation of the distance between the workpiece surface and a datum surface during a relative movement of the gauge and workpiece. Short wavelength variations caused by surface roughness are filtered from the signal to leave the variations due to form errors only, such as straightness, flatness and roundness. For straightness and flatness measurements, the reference datum will be a line or plane nominally parallel to the direction of motion, usually defined by the slideway which carries the stylus and transducer (often called the pick-up unit). Thus, the recorded variations in height of the surface of the workpiece will be perpendicular to the reference datum.

Although the basic principle is the same, instruments for roundness measurement have a mechanical design quite different from the ones for surface texture and straightness assessment and therefore will be considered separately. Most instruments for precision roundness measurement operate by using a precision spindle to generate a relative rotational movement between the component being measured and the stylus and transducer mounted radially with respect to the axis of rotation. A typical configuration comprises a turntable and a column, which carries the pick-up unit. The workpiece is rested on the rotating table and the stylus brought into contact with the workpiece by adjusting its radial and axial position. The achievable accuracy of such instruments is normally limited by the circularity error of the spindle, although in some cases the application of error separation techniques can improve the accuracy to a factor of ten or more [Chetwynd and Siddall, 1976].

The transducer measures the variation of the distance between the workpiece surface and a circular reference datum centred at the instrument axis of rotation and generated in space by the rotational motion between the part and a fixed point in the pick-up. The signal from the transducer represents a combination of the out-of-roundness of the part and the variation of the radial distance of the surface from the instrument axis of rotation caused by relative eccentricity between them.

Traditionally these variations are filtered, magnified and then superimposed on a convenient nominal radius which is totally unrelated to the actual dimensions of the component. The output of the instrument in this form will be referred to as the polar chart. The combination of this radius suppression and magnification often leads to a visually disconcerting effect on the polar chart since the aspect ratio of radial variation to circumference length of the profile has been changed. For example, a cylindrical part slightly flatted at intervals could plot as a figure having convex sides at a low magnification, but appear to have flat sides or even appear star-shaped as the magnification is progressively increased [Reason, 1966]. However in the presence of eccentricity the radius suppression will cause not only a distortion of aspect ratio but a distortion of shape as well [Chetwynd and Phillipson, 1980]. Therefore, every effort must be made to reduce it by keeping the eccentricity small, with various guidelines being suggested [e. g. Reason, 1966, Chetwynd, 1979a].

The straightness of cylindrical parts can be assessed by traversing the stylus vertically up the workpiece, with the turntable held stationary; the reference datum in this case is defined by the slideways of the column which carries the pick-up unit.

The analogue data gathered in these ways by either of these instruments will then be digitised and stored on a computer for further geometric assessment. Methods for shape evaluation using data acquired by this type of instrument will be discussed in the following sections.

2.3 Software for Metrology

The measurement accuracy of computer aided measurement systems is highly influenced by two main factors: how representative is the set of points of the object; and how suitable is the mathematical modelling and numerical manipulation of the data. The rapid and widespread adoption of CMMs over the last decade has led to a great concern about the accuracy of software for metrology. Nowadays, some national and international laboratories and standard institutions are engaged in defining standard procedures for mathematical modelling and testing of software [Cox and Jackson, 1983; Anthony and Cox, 1984]. A British Standard has been published [BSI, 1989], providing information and guidance to software writers and users within the CMM industry.

For any particular measurement, the software implementation comprises the following steps:

- data acquisition;
- mathematical modelling;
- design of algorithm;
- coding in some language.

The strategy used to gather the data points should consider the number of data points to collect and their distribution on the workpiece surface. Clearly, the more data points that are collected, the more reliable the tolerance assessment is likely to be. However, economic considerations will limit the number of data points it is possible to gather and it will be necessary to balance the need for reliable information with these constraints. A British Standard [BSI, 1989] provides some recommendations about the distribution of points as well as the minimum number of points required for each geometric element.

Mathematical modelling is concerned with how basic geometric elements, e. g. lines, circles, planes, can be parameterised, that is represented in terms of a set of algebraic parameters. Reliable parametrisations are required when a computer is used to fit these basic geometries to data gathered using a measuring machine, in order to determine the position, orientation and size of geometric features. It is possible to parametrise each of the geometric elements in more than one way. For example, a line in a specified plane can be defined by either one point on the line and information about its orientation, or two points on the line. The British Standard Institution [BSI, 1989] and a report from the Commission of the European Communities [G. T. Anthony et al., 1991] give some recommendations about ways of parametrising geometric elements elements as well as representing the distance of a point from each of the main geometric elements in terms of the recommended parameters.

Numerical calculations are employed to find the position and orientation of the geometric element that best fits the measured data points. The fit is represented by the value of the parameters of its mathematical model and form errors of the component are measured relative to this figure. Different criteria for specify the best fit are possible. In general algorithms for estimating reference figure fits use either a least squares approach or, based on metrological requirements, an optimisation of a geometric parameter of the figure. The British Standard Institution [BSI, 1989] and the National Physical Laboratory [Forbes, 1989], for example, recommend the least squares technique for computing the best fit geometric element to data.

Important aspects of a software code are modularity and reliability. It is very desirable that software code be well structured, or modular, in the sense that natural divisions be reflected in it by keeping each in a separate module. Practical advantages follow from modularisation such as ease of error detection and flexibility in putting different pieces of code together [Cox and Jackson, 1983]. The reliability of a software is also a matter of greater concern and therefore a validation process should be carried out. Some guidance for metrology software validation is given in Cox and Jackson [1983].

2.4 Linearised Exchange Algorithms for Reference Fitting

2.4.1 Computing Reference Figures

The major task of the computation process is to establish from the measured data the position of the ideal surface of which the true profile is taken to be an approximate realisation. This is the idea of computing a best fit reference figure relative to which profile deviations may be assessed. In this context the deviations are generally referred to as *residuals.*

Common criteria of fit are least squares and minimax. The Least squares best-fit reference is the one which minimises the sum of the squares of the residuals. The minimax reference is the one that generates the minimum value for the maximum deviation.

The definition of the residuals is generally related to instrument geometry and this will influence the algorithmic approach to computing the reference. For example, in some cases, it is possible to define the residuals as a linear function of the reference parameters, and therefore linear optimisation techniques could be used to compute the reference. The implications of various definitions of residuals and fitting criteria for form measurement will be discussed in the next sections. Roundness measurement will be first considered since it well illustrates some further important points.

2.4.2 Roundness Assessment System

Four reference figures are internationally accepted for roundness measurement. They are, in order of preference given by BS 3730 [BSI, 1987a]:

- Least squares circle (LSC),
- Minimum radial zone circles (MZC),
- Minimum radius circumscribing circle (MCC) or ring gauge,

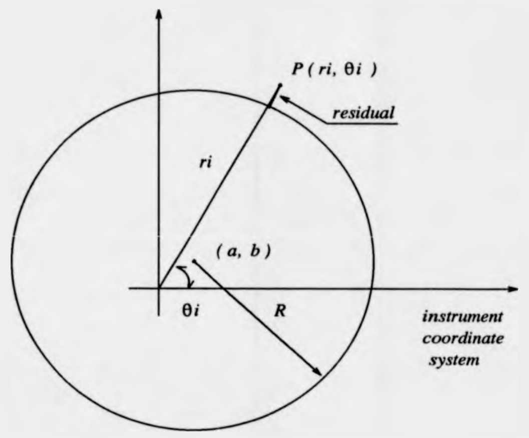


Figure 2.1: Residual of point P_i, (r_i, θ_i) from reference (a, b), R, measured radially from the origin.

- Maximum radius inscribing circle (MIC) or plug gauge.

The minimum radius circumscribing circle and the maximum radius inscribing circle are commonly used for evaluating the roundness of shafts and holes, respectively.

Usually the out-of-roundness information required is stated in terms of the maximum deviation of the profile from the reference figure. For data acquired using an independent spindle type of instrument, the residual can be accurately calculated as the radial separation, measured from the instrument coordinate system origin, figure 2.1, between the workpiece profile and a non-circular reference, the limaçon approximation discussed below. In this case, sound techniques exist for fitting data to the four reference criteria, based upon either linear least squares or linear exchange algorithms and using a limaçon figure as an approximation to the circle. However, the validity of this approach is closely related to the nature of the data set provided by this type of instrument.

In contrast, on a coordinate measuring machine the profile is represented by cartesian ordinate pairs to which a circle should be fitted with the residuals measured radially from its centre, figure 2.2, and not from the instrument coordinate system origin. In this case, unless a coordinate system transformation is made

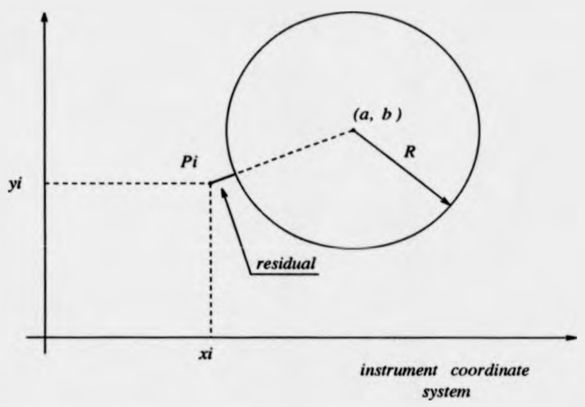


Figure 2.2: Residual of point P_i, (x_i, y_i) from reference (a, b), R, measured radially from the reference centre.

to a suitable local origin, the limaçon reference is not a good approximation, and therefore a truly circular reference fitting will be required.

These different approaches for assessing out-of roundness and other form features are discussed in the following sections.

2.4.3 The Limaçon Reference

It is helpful to visualise the measurement process in terms of three frames of reference as follows [Chetwynd and Phillipson, 1980]. Initially the true component shape exists in what may be termed *component coordinates*, that is, all its points have a fixed relationship to each other independently of its orientation in space. However, to measure the component requires that it is presented to the instrument which effectively expresses its shape in *instrument coordinates.* As it is not possible to position a component perfectly relative to an external frame of reference, an error due to misalignment will be introduced at this stage. The normal instrument operates by first radius-suppressing and then magnifying the profile, so that its output represents a transformation of the instrument coordinate frame into a *chart coordinate* frame, which can be identified with the polar graph. In instrument polar coordinates an eccentric circle, centred at (E, ϕ) and

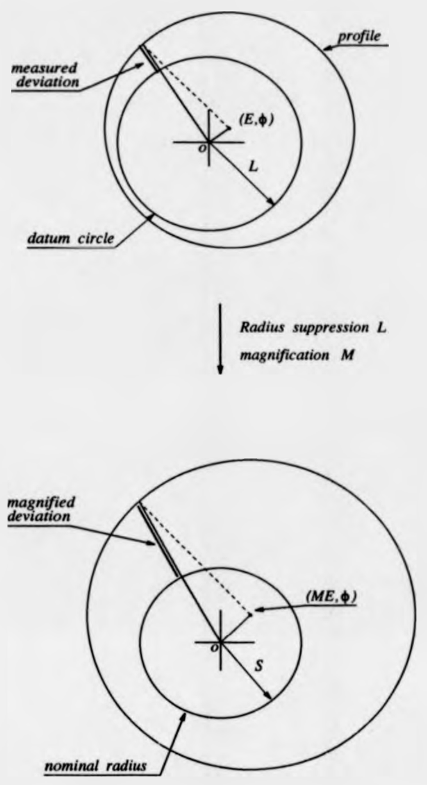


Figure 2.3: Roundness measurement using radius suppression and magnification.

with radius R_o may be expressed [Whitehouse, 1973]

$$k(\theta) = E\cos(\theta - \phi) + (R_o^2 - E^2\sin^2(\theta - \phi))^{1/2} \tag{2.1}$$

If this is then transformed into chart coordinates by applying radius suppression L, magnification M and then adding the arbitrary chart radius S, (see figure 2.3) its polar chart representation will be

$$r(\theta) = M(k(\theta) - L) + S \tag{2.2}$$

Applying the binomial expansion to equation (2.1), gives

$$k(\theta) = E\cos(\theta - \phi) + R_o(1 - (\gamma^2/2)\sin^2(\theta - \phi) - (\gamma^4/8)\sin^4(\theta - \phi)\cdots) \tag{2.3}$$

and if this is expressed in chart coordinates

$$r(\theta) = ME\cos(\theta-\phi) + M(R_o - L) + S + MR_o(-(\gamma^2/2)\sin^2(\theta-\phi) - \cdots) \quad (2.4)$$

where $\gamma = E/R_o$ is termed the eccentricity ratio. Since the ratios of the constant and harmonic terms of the expansions in equations (2.3) and (2.4) have been altered by radius suppression, the figure described by equation (2.4) is distorted from circularity. Therefore, a circle centred on the origin is transformed to another circle but an eccentric circle is transformed to a non-circular shape. The degree of distortion depends, for a given radius suppression, on the amount of eccentricity present. Thus to effectively fit reference circles in instrument coordinates requires that non-circular shapes be fitted on the chart. Ideally, the reference in chart coordinates should be the shape given by equation (2.4). In practice however, the degree of eccentricity present is quite small: an eccentricity of more than 10μm would seriously reduce the measurement range in most measurements and many components have radii exceeding 10 mm, so γ is typically of the order of 10^{-3} or smaller. Under such conditions, it is reasonable to approximate the radius suppressed eccentric circle by just the first two terms of the infinite series in equation (2.4), giving

$$r(\theta) = R + a\cos\theta + b\sin\theta \quad (2.5)$$

where $R = M(R_o - L) + S$, $a = ME\cos\phi$ and $b = ME\sin\phi$.

It was recognised [Whitehouse, 1973, Chetwynd, 1980] that this truncation describes a specific geometric figure, namely a *limaçon* and that its shape represents more closely the radius suppressed form of a circle in chart coordinates than does a true circle. Chetwynd [1980, also Chetwynd and Phillipson, 1980] has brought to attention the fact that a limaçon figure in chart coordinates transforms to another limaçon, though with different parameter values, in instrument coordinate. Furthermore the relationship between a profile and a reference limaçon is unaltered by radius suppression.

Although the true reference shape in chart coordinates is usually better represented by a limaçon than a circle, in instrument coordinates the circle is correct and the limaçon an approximation to it. The "error" between limaçon and circle will be dominated by the second term of the expansion in equation (2.3) giving

$$Error \approx \frac{R_o\gamma^2}{2}\sin^2(\theta-\phi) = \frac{R_o\gamma^2}{4}[1-\cos 2(\theta-\phi)] \tag{2.6}$$

Thus the distortion is essentially elliptical, aligned with the eccentricity vector and the R parameter of the limaçon is a biased estimator of R_o. However, in practice the eccentricity ratio rarely exceeds 0.01, so the radial variation between the limaçon and the circle is at most a fraction of a percent of the total eccentricity.

In his work, Chetwynd has also stressed the point that the limaçon approximation can be understood as a linearisation by the truncation of the Taylor series expansion at the origin of the circle function in instrument coordinates, and not by the simple truncation of the binomial series as shown in equation (2.3). Equivalent procedures are therefore general and not dependent upon specific geometrical features of an instrumental technique. The alternative formulation emphasizes that a choice is made to linearise the circle about a convenient point: the limaçon reference had known advantages over a circle on the chart and being linear in its parameters could be used conveniently for computation. The exploitation of this has led to the development of the exchange algorithms which are discussed next.

2.4.4 Exchange Algorithms for Bounding Limaçon Problems

All the three bounding references (that is MCC, MIC and MZC) can be expressed as problems in the general class of constrained optimisation. That is they can be stated in the form: maximise (or minimise) a function subject to a set of constraints. Thus, for instance, the requirement for finding the plug gauge circle is to find the maximum radius for which a limaçon may be constructed such that the limaçon lies completely inside the data representing the nominal circle being measured (the "radius" of a limaçon is the value of its constant term). This may be expressed mathematically as, given a set of polar data points (r_i, θ_i), where $i = 1, \cdots, N$, the number of data points, maximise R subject to the constraints

$$R + a\cos\theta_i + b\sin\theta_i \leq r_i \tag{2.7}$$

Similarly, for the ring gauge limaçon the requirement will be to minimise R subject to the constraints

$$R + a\cos\theta_i + b\sin\theta_i \geq r_i \tag{2.8}$$

For both of these problems there are three variables a, b and R and so a subset of the constraints consisting of three of them forming a consistent set of simultaneous equations must be exactly satisfied in the optimum solution: thus there will be at least three contact points between the reference and the data.

For the minimum zone limaçons case, a single reference limaçon is defined with a symmetrically placed zone of $\pm h$. The minimum zone problem is then stated as: minimise h subject to the constraints

$$\begin{aligned} R + a\cos\theta_i + b\sin\theta_i + h &\geq r_i \\ R + a\cos\theta_i + b\sin\theta_i - h &\leq r_i \end{aligned} \tag{2.9}$$

again for $i = 1, \cdots, N$. For the minimum zone there are four parameters, a, b, R and h and so at least four contact points between data and reference zone extremities will be expected. Note also that each data point gives rise to two constraints, indicating that finding the zone will need more work than finding the other bounding references.

A consequence of using a limaçon as the reference is that in all three cases the constraints are linear in the parameters, as is the function to be optimised, which makes possible the use of linear programming optimisation techniques. Linear programming is an established technique in operations research with a well developed theory (see, for instance, Hadley [1962] and Wagner [1975]). It offers the advantage that a unique solution may always be found by a finite iterative search.

The implications of the theory of linear programming to the solution of linear bounding references such as the limaçon, including the theoretical justification for exchange algorithms, has been exhaustively explored by Chetwynd [1980] (also [Chetwynd and Phillipson, 1980 and Chetwynd, 1985]) and the relevant points are reviewed below.

In his work, Chetwynd shows that the optimum solution to the minimum circumscribing limaçon problem obeys the following two geometric conditions:

1. All the data points must be circumscribed by the limaçon.

2. There will be at least three data points which lie on the limaçon such that they do not all lie within a subtended angle, at the origin, of less than 180^o.

The second condition is called the 180^o rule. Under the special condition that the three contact points have equal radial values the limaçon becomes a circle centred at the origin.

The optimisation process proceeds by selecting a limaçon that obeys one of the above conditions and iteratively adjusting it , while maintaining that condition, until the other condition is fulfilled. The optimum has then been reached.

The principle underlying an exchange algorithm is as follows. From a set of data points take a number just sufficient to solve the fitting problem exactly and obtain the solution from them. Generally, some other data points will violate some criterion of acceptability of this solution for the complete data set but, by definition, this solution will be optimal for the sub-set that do not cause violation. One of the violating points is then exchanged for one of the original points and a new solution computed. The process iterates until all points satisfy the criteria. Such a process will always converge in a finite number of iterations providing the exchange is performed in a manner which causes the value of the quantity being optimised to vary monotonically throughout the iterations.

Thus, a complete algorithm for the minimum radius circumscribing limaçon is as follows [Chetwynd and Phillipson, 1980]:

1. Choose any three data points such that no two adjacent ones subtend an

angle at the origin of more than 180°.

2. Construct a reference limaçon through these three points.
3. If no data points lie outside this limaçon the solution is found. Otherwise choose the point which violates the reference by the largest amount.
4. Replace one of the reference points by this new point such that the 180° is still obeyed and return to step number two. (Note: this exchange is unique.)

The maximum radius inscribed limaçon problem is solved by changing the sign of all radial data points and applying the ring limaçon algorithm as given above to the modified data. The parameters of the maximum inscribed limaçon are obtained by changing the sign of the parameters found in this way. This ability to use the same algorithm for two problems is a further consequence of the linearity of the limaçon in its parameters.

The conditions for the optimum solution to the minimum radial zone limaçons give rise to the following geometric interpretation [Chetwynd and Phillipson, 1980]:

1. All data points must lie not more than a distance h, measured radially from the origin, from the limaçon.
2. There must be four data points all lying exactly h from the limaçon such that they lie with increasing angle θ_i alternately inside and outside the limaçon.

As with the minimum circumscribing limaçon, these rules may be used to formulate an exchange algorithm, which is given below [Chetwynd and Phillipson, 1980]:

1. Choose, arbitrarily, four data points.
2. Fit a limaçon to these points such that they lie equidistant, radially, from it and obey the alternating point rule with respect to it.

3. If no other data point lies further radially from the limaçon, either inside or outside, the solution is found.

4. Otherwise, substitute the furthest lying point for one of the defining points such that the alternation rule is still obeyed and return to 2. (Again, this exchange is unique.)

The alternation principle has been widely recognised and used intuitively. Exchange algorithms for fitting bounding circle references have been proposed by Anthony and Cox [1985] (see section 2.6.2). In a recent paper [Kaiser and Morin, 1992], it is shown that the alternation property of the minimum zone exchange algorithm can also be derived from the linearised version of a theorem originally due to Bonnesen [1924].

2.4.5 Limaçon Cylindrical Reference

There is no standardisation on how variations of the surface of a cylindrical workpiece from a perfect cylinder should be measured and, therefore, errors of cylindricity are generally measured as an extension of roundness. The same analytical methods are to be expected.

On an independent spindle type of instrument, profiles representing sections of a cylindrical component are produced on planes perpendicular to the z-axis of the instrument coordinate system. However, the axis of the cylindrical component will, in general, be misaligned in relation to the instrument axis due to tilt of the component. The equation of a tilted cylinder in polar coordinates is [Chetwynd, 1980]

$$R(\theta, z) = \left[\frac{(a + a_1 z + ab_1^2 - a_1 b b_1)\cos\theta + (b + b_1^2 + ba_1^2 - aa_1 b_1)\sin\theta}{1 + (b_1\cos\theta - a_1\sin\theta)^2}\right] \tag{2.10}$$

$$+\frac{R(1 + a_1^2 + b_1^2)^{1/2}}{(1 + (b_1\cos\theta - a_1\sin\theta)^2)^{1/2}}\left[1 - \frac{((a + a_1 z)\sin\theta - (b + b_1 z)\cos\theta)^2}{R(1 + (b_1\cos\theta - a_1\sin\theta)^2)}\right]^{1/2}$$

where (a, b) are the coordinates of the intersection of the cylinder axis with the

$z = 0$ plane and R the radius of the cylinder; a_1 and b_1 are the slopes from the z-axis of the projections of the cylinder axis into the xz- and yz-planes The cross section in a plane of constant z is an ellipse with minor semi-diameter R and major semi-diameter $R(1 + a_1^2 + b_1^2)^{1/2}$.

The cylinder is clearly non-linear in its parameters and furthermore can be shown to exhibit non-unique solutions for all the reference conditions. Chetwynd [1980] proposed the linearisation by the Taylor expansion of the tilted cylinder equation in (2.10) about the point of perfect alignment ($a = b = a_1 = b_1 = 0$). This process generates a figure which he named *skew limaçon cylindroid*, given by

$$R(\theta, z) = (a + a_1 z)\cos\theta + (b + b_1 z)\sin\theta + R \tag{2.11}$$

A comparison of equations (2.10) and (2.11) shows how much information is totally disregarded by the linearisation. In particular there is no remaining term concerned with the ellipticity of the cross-section (hence the term skew instead of tilt). The nature of the error terms in the linearisation process is emphasised if they are expressed as [Chetwynd, 1980]

$$Error \approx \frac{tan^2\alpha R}{4}(1 + \cos 2(\theta - \phi_\alpha)) - \frac{E^2(z)}{4R}(1 - \cos 2(\theta - \phi_E(z))) \tag{2.12}$$

where α is the angle of the cylinder axis to the z-axis and ϕ_α and ϕ_E are the directions of tilt and total eccentricity in the xy-plane. The eccentricity terms E and ϕ_E depend upon z whereas the terms due to pure tilt do not. The acceptability of the model depends on the maximum value of the eccentricity ratio and also on the magnitude of the tilt compared to absolute radius. As written above, the first term in the error can be identified with the representation of the tilted cylinder in terms of a skew circular cylindroid while the second term relates to the approximation of the circular cross-sections of that cylindroid by limaçons [Chetwynd, 1980]. The above discussion is naturally also of concern to the measurement of roundness profiles on cylindrical objects as it is quite common for tilt be the major cause of eccentricity in a reading.

The parameter linearisation brought about by the definition of the skew limaçon cylindroid made possible the application of the same techniques as used for roundness measurement. Its behaviour under radius suppression is the same as that of the limaçon since the radius suppression operates in directions perpendicular to the z-axis. The magnification usually associated with the translation to chart coordinates has one extra effect on the cylindroid since generally it would be expected that different values of magnification would be applied in the radial and axial directions. Thus, the slope of the cylindroid axis from the measurement axis is multiplied by the ratio of the magnifications in these directions.

The solution of the boundary limaçon cylindroid is a direct extension of the methods used in two dimensions. However, the motivation for seeking exchange algorithms to replace the general method reduces as the geometrical complexity of the problem increases for the manipulations required to calculate the exchange become more involved. Therefore, the recommended method for solving all the three boundary limaçon cylindroid is by direct solution of the dual linear programme [Chetwynd, 1980].

Given the greater degree of complexity in cylinder formulations than in circle ones, it is to be expected that the shape difference between limaçon cylindroid and cylinder is subject to more sources of variations than is that between limaçon and circle and therefore, there will be less that can be said with certainty concerning bounds to cylinder fits from cylindroid fits. Chetwynd presents in his work a lengthy discussion on the limitations and problems of using the skew limaçon cylindroid model as a approximation to the tilted cylinder, though he ultimately reaches no firm conclusion concerning the limits of its applicability.

2.4.6 Minimum Zone Straight Lines and Planes

There are two definitions of residuals particularly relevant to straightness and flatness measurement: one places the residuals normal to the reference line or plane and the other normal to the instrument datum, figure 2.4. These definitions correspond to residuals aligned in the component and instrument frames respectively.

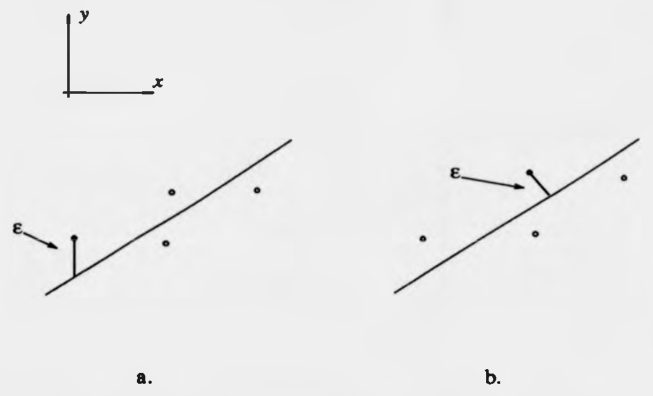


Figure 2.4: Residuals measured parallel to y-axis (**a**) and normal to fitted reference line (**b**).

The residuals in instrument coordinates, ϵ_i, of a set of three-dimensional data points (x_i, y_i, z_i), from a best-fit plane are given by

$$\epsilon_i = z_i - (ax_i + by_i + c) \tag{2.13}$$

for $i = 1, \cdots, N$ where N is the number of points and (a, b) are the slopes of the intersections of the fitted plane with the xz- and yz-planes and c the intercept of the plane with the z-axis. This places the residuals normal to the xy-plane of the instrument coordinate system, thus the residuals are measured parallel to the z-axis. It is this which forces the linearity of the formulae and so guarantees that there is a unique solution to the fit. This model however is only valid if it makes physical sense in terms of the geometry of the instrument to consider the residuals acting in this manner. This is indeed the case on a stylus-based form measurement instrument. The instrument gauge traverses parallel to the datum plane/line and measures the normal distance variation from the datum to the workpiece. Its operation depends on the instrument datum surface being aligned quite closely to the trend of the workpiece surface. Typically the traverse might be 10 mm or longer, while the vertical displacement might be some tens of μm or less. Thus the slope of the fitted line/plane relative to the instrument reference must be very small and only an insignificant approximation is involved in using equation (2.13).

Now consider the use of a coordinate measuring machine to take a set of Cartesian points along the surface of a nearly flat/straight feature. The orientation of the surface may be arbitrary and therefore the residuals are usually considered to be normal to the fitted plane/line: taking them parallel to an instrument axis would generally not be a good approximation. Thus the residuals should be considered as shown in figure 2.4.b. Calculating the residuals according to this definition, the form of equation (2.13) is replaced by

$$s_i = \frac{-z_i + ax_i + by_i + c}{(1 + a^2 + b^2)^{1/2}} \tag{2.14}$$

again for $i = 1, \cdots, N$, where s_i are the residuals measured normal to the fitted plane. This is clearly non-linear in its parameters and consequently fitting according to it involves more computation than does equation (2.13). It is important to note here that by parameterising the equation of a line in a different way, it is possible to express the residuals normal to the fitted line (figure 2.4.b) as a linear function of the parameters [Forbes, 1989]. This however will be discussed in the next sections.

The methods developed for deriving boundary references are not applicable only to the measurement of roundness but to any problem which can be expressed in a similar way. Probably the most commonly applied measurements for straightness and flatness involve the minimum zone criterion. The minimum zone straight lines or planes are a pair of parallel lines/planes so placed that they enclose the profile between them and that their separation is a minimum. By defining the separation normal to the datum axis or plane (and so ensuring linearity) it is consequently possible to solve these by using linear programming techniques.

The solution for the minimum zone straight lines can be found by direct application of the Stiefel exchange algorithm [Osborne and Watson, 1968 and Chetwynd, 1985] for minimax polynomials fitting, that is curves having the smallest possible maximum divergence from the data. In terms of the minimum zone straight lines, there will be three contact points, two contacting one line, and

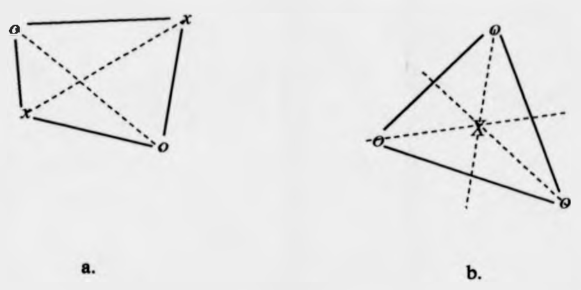


Figure 2.5: Plan view of contact geometries for minimum zone planes. O and X represent contacts with different planes.

one the other in an alternate sequence. The algorithm proceeds by interchanging one of the contact points with the furthest point lying outside the zone, while maintaining the alternate condition, until all the data points are enclosed by the reference lines.

The minimum zone planes can be expressed, in instrument coordinates as: minimise $Z = h$ subject to the constraints

$$\begin{aligned} ax_i + by_i + c + h &\geq z_i \\ ax_i + by_i + c - h &\leq z_i \end{aligned} \tag{2.15}$$

for all $(x_i, y_i, z_i), i = 1, \cdots, N$, a, b and c sign unrestricted and $h \geq 0$.

The geometric interpretation of the dual feasibility of its linear programme led again to the definition of exchange rules [Chetwynd, 1985]. Four contact points are required; there can be two contacts with each of the minimum zone planes, figure 2.5a, or one contact point in one plane and the other three ones in the other minimum zone plane, but obeying a spatial configuration different from the first, figure 2.5b [Chetwynd, 1985]. There is a unique exchange for any new point in order that these relationships are preserved and so a workable exchange algorithm may be based upon these patterns.

2.5 Least Squares Reference Fitting

2.5.1 Computing Least Squares References

Fitting a least squares reference to data amounts to finding the values of the parameters which makes the sum of squares of the deviations d_i of the data points to the reference take on its minimum value, hence the term *least-squares* fit. This is expressed by: minimise

$$\sum_{i=1}^{N} d_i^2 \tag{2.16}$$

where N is the number of data points. In general, modelling the problem of finding the least-squares best fit element to data involves choosing parameters to describe the geometric element and deriving a formula for the deviations of points to the geometric element in terms of these parameters.

Fitting a least square reference is simplified if the deviations d_i are given as a linear function of the parameters as, unlike non-linear least squares problems, it is certain to be directly solvable and the solution to be unique. Some measurement problems are naturally linear and others can be well-approximated. Other problems would involve quite large degrees of approximation and thus non-linear methods are unavoidable.

2.5.2 Least Squares Limaçon

In the case of circle fitting it is first necessary to define what is meant by deviations. The most logical form for the residuals is in terms of their radial distance from the circle centre, figure 2.2, given by

$$\epsilon_i = s_i - R\ , \quad i = 1, \cdots, N \tag{2.17}$$

where N is the number of data points and $s_i = [(x_i - a)^2 + (y_i - b)^2]^{1/2}$ is the distance from the data point (x_i, y_i) to the centre of the circle (a, b). Using equation (2.17), the required minimisation will be non-linear in its parameters a, b and R.

However, for data sets acquired using an independent spindle type of instrument, the variation measured, as mentioned in sections 2.2.3 and 2.4.3, is the distance between the workpiece surface and a circular reference datum, r_i, from the instrument axis of rotation, figure 2.1. Thus, the residuals are more appropriately defined as

$$\epsilon_i = r_i - k(\theta) \ , \quad i = 1, \cdots, N \tag{2.18}$$

where $k(\theta)$ is the equation of an eccentric circle measured from the instrument origin (see equation (2.1)). By using the limaçon approximation (see section (2.4.3)), equation (2.18) becomes

$$\epsilon_i = r_i - (a\cos\theta + b\sin\theta + R) \ , \quad i = 1, \cdots, N \tag{2.19}$$

and this least-squares minimisation will be linear in its parameters.

It is important to observe that when the instrument is of the independent spindle type, the residuals given by equation (2.19) defined radially from the instrument centre are not only more convenient than the ones defined in equation (2.17), but also they are more precisely correct [Chetwynd, 1979b].

A fully general derivation of the least squares limaçon is given by Chetwynd [1980]. The general solution is found by solving the normal equation given by

$$\begin{bmatrix} a \\ b \\ R \end{bmatrix} = \begin{bmatrix} \sum cos^2\theta_i & \sum sin\theta_i cos\theta_i & \sum cos\theta_i \\ \sum sin\theta_i cos\theta_i & \sum sin^2\theta_i & \sum sin\theta_i \\ \sum cos\theta_i & \sum sin\theta_i & n \end{bmatrix}^{-1} \begin{bmatrix} \sum r_i cos\theta_i \\ \sum r_i sin\theta_i \\ \sum r_i \end{bmatrix} \tag{2.20}$$

where n is the number of data points (in lower case to simplify notation) and all sums are over $i = 1, \cdots, n$. Simplifications can be made by selection of specific angular positions θ_i. All off-diagonal terms can be forced to zero simultaneously by choosing a four-fold symmetry of samples around the circle, that is for a sample at θ_i there should also be ones at $\theta_i + \pi/2, \theta_i + \pi$ and $\theta_i + 3\pi/2$. For any four-fold scheme, equation (2.20) reduces to the standard formulae

$$a = \frac{2}{n}\sum r_i cos\theta_i \ ; \quad b = \frac{2}{n}\sum r_i sin\theta_i \ ; \quad R = \frac{1}{n}\sum r_i \tag{2.21}$$

The first derivations of these formulae [see Reason, 1966] were through the integration of continuous profiles using analytic results for the integrals of the trigonometric functions over one cycle. It was assumed that the integral could be replaced by a summation providing a "reasonable" number of samples were used. In most Standards [e. g. BSI, 1987b] this has appeared as a "reasonably large even number" without further justification. The need for four-fold symmetry was found by Chetwynd, 1979b, when a true discrete least squares solution was undertaken.

2.5.3 Least Squares Circle

When using a more general measuring instrument, like a coordinate measuring machine, the profile is represented by Cartesian ordinate pairs to which a circle should be fitted (figure 2.2). In this case, the correct form for the residuals is in terms of their radial distance from the circle centre, as given by equation (2.17). This is non-linear in its parameters and therefore it requires an iterative type of algorithm to find the least squares solution, that is find parameters (a, b) and R so as to minimise

$$\sum_{i=1}^{N}(s_i - R)^2 \tag{2.22}$$

where $s_i = [(x_i - a)^2 + (y_i - b)^2]^{1/2}$.

Note that the limaçon figure would be a valid reference if a coordinate system transformation was made to a suitable local origin, which would allow the linearised form for the residuals (equation 2.19) to be used instead.

An algorithm for fitting circles in a specified plane is described by Forbes, 1989, which is based on the Gauss-Newton algorithm [see e. g. Gill, Murray and Wright, 1981].

An estimate of the solution must be available to start the algorithm. The initial solution is then updated in each iteration until it has converged. The major step of this algorithm is to solve the linear least squares system given by

$$\underline{J}\begin{bmatrix} p_a \\ p_b \\ p_R \end{bmatrix} = -\underline{\epsilon} \tag{2.23}$$

where $\underline{\epsilon}$ is the matrix representation of the deviations given by equation (2.17) and $\underline{J}$ is the Jacobian matrix, whose elements are found from the partial derivatives of ϵ_i with respect to the parameters a, b and R and given by

$$\begin{aligned} \tfrac{\partial \epsilon_i}{\partial x_o} &= \tfrac{-(x_i - x_o)}{r_1} \\ \tfrac{\partial \epsilon_i}{\partial y_o} &= \tfrac{-(y_i - y_o)}{r_i} \\ \tfrac{\partial \epsilon_i}{\partial r} &= -1 \end{aligned} \tag{2.24}$$

The solution of the system given in (2.23) is used to update the parameters estimates as

$$\begin{aligned} a &:= a + p_a \\ b &:= b + p_b \\ R &:= R + p_R \end{aligned} \tag{2.25}$$

This process is repeated until the algorithm has converged.

In order to find a good initial estimate of the circle parameters, Forbes suggests that the function to be minimised be replaced by

$$\sum_{i=1}^{N}(s_i^2 - R^2)^2 \tag{2.26}$$

By expressing $s_i^2 - R^2$ in terms of parameters (a, b) and $\rho = a^2 + b^2 - R^2$, this function is made linear in its new parameters and therefore, in order to minimise (2.26), a linear least squares system is solved as

$$A\begin{bmatrix} a \\ b \\ \rho \end{bmatrix} = b \tag{2.27}$$

where the elements of the i^{th} row of A are the coefficients $(2x_i, 2y_i, -1)$ and the i_{th} element of b is $(x_i^2 + y_i^2)$.

It is worth to mention that a similar type of algorithm, based on the Gauss-Newton method, had been previously proposed, although not presented in detail, by Anthony and Cox [1985]. Similarly, the same linear approximation is suggested so as to find good initial estimates.

Least squares sphere fitting is also discussed by Forbes [1989]. An algorithm for least squares sphere fitting is presented which is the three dimensional version of the algorithm outlined above.

2.5.4 Least Squares Cylindroid and Cylinder

The skew limaçon cylindroid (see section 2.4.5) is linear in its parameters and so, following the same reasoning to use the definition of residuals given by equation (2.19) for the least squares limaçon, its residuals are expressed as

$$\delta = r_i - ((a + a_1 z)\cos\theta_i + (b + b_1 z)\sin\theta_i + R)\ , \quad i = 1, \cdots, N \tag{2.28}$$

where a, b, a_1, b_1 and R are as defined in equation (2.10) and N is the number of data points. Thus the least squares limaçon cylindroid is found by the solution of a linear least squares system [Chetwynd, 1980], which will not be repeated here. Patterns of measurement are also suggested by Chetwynd in order to simplify the least squares computation and make the estimates of the parameters independent of each other. However, there seems to be no convenient simple sampling scheme, as in the two-dimensional case, to justify the adoption of such schemes considering the computing power of present generation computer-aided-measurement system.

Forbes [1989] presents an algorithm for fitting a cylinder to Cartesian data points (x, y, z). In this case, the residuals are defined as the deviations of the data points from the reference, measured radially from the axis of the cylinder. This is expressed as

$$\epsilon_i = r_i - r \tag{2.29}$$

where

$$r_i = \frac{\sqrt{u_i^2 + v_i^2 + w_i^2}}{\sqrt{a^2 + b^2 + c^2}} \tag{2.30}$$

with

$$\begin{aligned} u_i &= c(y_i - y_o) - b(z_i - z_o) \\ v_i &= a(z_i - z_o) - c(x_i - x_o) \\ w_i &= b(x_i - x_o) - a(y_i - y_o) \end{aligned} \tag{2.31}$$

for $i = 1, \cdots, N$, where (a, b, c) represents a vector pointing along the axis of the cylinder and (x_o, y_o, z_o) a point on its axis.

The algorithm to find the least squares best fit cylinder is based on the Gauss-Newton method mentioned in section 2.23. However an assumption is made that for nearly vertical lines, in three dimensions, its direction can be represented by a vector of the form $(a, b, 1)$, and, given the latter, the z-ordinate of a point on the line is specified by $z_o = -ax_o - by_o$. Therefore, equation (2.29) is reduced to a function of the five parameters x_o, y_o, a, b and r. In order to implement the Gauss-Newton algorithm, the partial derivatives of ϵ_i (equation (2.29)) with respect to these five parameters are needed. In order to simplify the derivative expressions, a second assumption is made that the axis is exactly vertical and passes through the origin.

Thus the algorithm iterates as usual, except that at the beginning of each iteration, a copy of the data is translated and rotated so that the trial best-fit element (i. e. the element corresponding to the current estimates of the parameters) has a vertical axis passing through the origin. The special orientation simplifies the calculation of the Jacobian matrix. At the end of an iteration, the inverse translation and rotation transformations are used to update the parameter estimates and thus determine the new position and orientation of the axis.

Unfortunately, there appears to be no straightforward method for obtaining initial estimates of the parameters. If there are estimates of (a, b, c), then we can rotate the data so that the trial axis is vertical and fit a circle to x- and y-coordinates of the rotated points to obtain estimates of (x_o, y_o, z_o) and r. In many measurement situations such estimates of (a, b, c) are available from the approximately known orientation of the workpiece with respect to the coordinate system of the measuring machine.

Algorithms for circle fitting in three dimensions and cone fitting are also discussed by Forbes [1989], and follow the same strategy as for cylinder fitting, although the expressions for the residuals are slightly different in each case.

2.5.5 Least Squares Lines and Planes

As discussed in section 2.4.6, when the residuals are defined normal to the instrument datum, they are expressed as (in two dimensions)

$$\epsilon_i = y_i - (mx_i + l) \, , \quad i = 1, \cdots, N \tag{2.32}$$

where m is the slope of the line and l the intercept of the line with the y-axis. Thus, fitting a line to a set of data points (x_i, y_i) that minimises the sum of the square of the deviations given by equation (2.13) is very straightforward and commonly given in statistics text books as linear regression. The same approach is valid for plane fitting under the same conditions (see equation 2.4).

This definition of residuals is only valid when the geometry of the instrument allows such an assumption be made. In measurements using a coordinate measuring machine for example the residuals are truly defined by the normal distances to the reference line, as given by

$$s_i = \frac{y_i - mx_i + l}{(1 + m^2)^{1/2}} \, , \quad i = 1, \cdots, N \tag{2.33}$$

where s_i is the Euclidian distance of the point (x_i, y_i) to the line of slope m and intercept l. This is non-linear in its parameters and therefore requires non-linear methods.

Forbes [1989] presents an algorithm for fitting a least squares line through a set of points with the residuals measured normal to the fitted line. The residuals are re-expressed as

$$s_i = b(x_i - x_o) - a(y_i - y_o) \ , \quad i = 1, \cdots, N \tag{2.34}$$

where (a, b) are the direction cosines of the line [see e. g. Ayre and Stephens, 1956] and (x_o, y_o) a point on the line.

The algorithm works on the fact that the line passes through the centroid of the points which makes it possible to reduce the problem to a linear least squares system. The direction cosines of the line are defined by finding the Singular Value Decomposition of the matrix of coefficients (the normal equation matrix, see e. g. [Gill, Murray and Wright, 1991]): the direction cosines are given by the singular vector corresponding to the largest singular value.

The algorithm for normal least squares plane fitting is also based on the same assumption, viz that the plane passes through the centroid of the points.

2.6 Minimum Zone and Other Types of Approaches

2.6.1 Minimum Zone Reference Fitting

Minimum zone approaches do provide a smaller form error zone for a given set of data points than commonly used least squares methods (see section 2.5). The minimax reference has the property that the largest absolute residual $| \epsilon_i |$ is as small as possible. The minimum zone methodology is also attractive in the sense that when it is shown diagrammatically it appears similar to a geometrical tolerance zone [e. g. BSI, 1990]. Therefore, there has been interest regarding the evaluation of form tolerances based upon the minimum zone by which features can be described.

Apart from the exchange algorithms, other comprehensive methods have been proposed for fitting a minimax reference to data. Shunmugam [1986] presents a heuristic procedure he calls *median technique* in which the deviations are measured from a *median reference*. In the case of straightness for example, the procedure starts by fitting a trial line passing through the end points. To fix the crest line, the point corresponding to the maximum positive deviation is selected as one of the reference points and the process repeated until the deviations are negative or zero. A similar approach is used to fix the valley line. The median line is then determined by three points selected from the crest and valley points so that the straight zone is minimum. However, the deviation is measured normal to the datum system and not normal to the reference figure. Similar procedures, based on the same technique, are outlined in the paper for finding minimax plane, circular, cylindrical and spherical references and again the deviations are measured from the linearised approximation of the true references. Although this is a simple method, there is no evidence to judge that the reference determined in this way for a given data set is the one that will have the minimum zone.

Murthy and Abdin [1980] present the use of three different methods, namely Monte-Carlo, Simplex and Spiral search techniques, for finding the minimum zone lines, planes, circles and spheres. In the case of Monte-Carlo technique the minimum zone surface is assumed to lie within the zone of deviations obtained by the least squares method. To determine the actual minimum zone reference the parameters are selected randomly. For each randomly selected solution the minimum deviation is calculated. If a value less than the least squares value is found in these trials the reference is shifted to this and the process is repeated. This process continues until the minimum value does not change appreciably. In this method, because of random selection of variables, there is a possibility of missing the actual minimum. A more convenient method, the authors suggest, is the Simplex search technique.

The Simplex search (due to Nelder and Mead [1965] not to be confused with the simplex method of linear programming) gets its name from the regular geometric figure used in the search process. This is a formal sequential gradient search designed to climb up and down non-linear mathematical functions. The basic idea is to compare the values of the objective function at the $n + 1$ vertices of a general simplex and move this simplex gradually towards the optimum point. In the case of circle fitting, for example, the function to be minimised is

$$f = [max\{r_i - r\} - min\{r_i - r\}] \tag{2.35}$$

where $r_i = [(x_i - a)^2 + (y_i - b)^2]^{1/2}$, $i = 1, \cdots, N$, (a, b) and r are the reference parameters and (x_i, y_i) the data set. The least squares solution (see section 2.5.2) is considered a good starting point so as to reduce the number of iterations. A third search technique suggested by the authors, when there are only two or three variables, is the Spiral technique in which a complete scanning for the absolute minimum could be tried in a *spiral* manner around the least squares solution. It is suggested that a combination of spiral and simplex search techniques would yield good results.

It is claimed in the paper that the Simplex search consistently gave the least minimum zone of the methods considered for straightness, flatness, circularity and sphericity and that the results are in the range of 80 % of the least squares zone. One advantage of this method is that the deviations are clearly measured normal to the reference figure (e. g. in the case of circle fitting normal to the centre of the reference). However, no information about the computational efficiency of this method is given in the paper.

Dhanish and Shunmugam [1991] present an algorithm for evaluation of form errors such as straightness, flatness, roundness based on the theory of discrete and linear Chebyshev approximation. The deviations are defined as linearly dependents of their parameters, as in equations (2.32) and (2.13) for deviations from lines and planes respectively, or as in equation (2.5) for deviations from circles. This algorithm is a type of exchange algorithm, in which the Stiefel Exchange Algorithm [Osborne and Watson, 1968] is used for the straightness case. For the flatness case it seems no rule is defined and trial exchanges are made replacing the points in the reference set one by one. Although it is not mentioned, this algorithm is based on the same theory as the exchange algorithms of Chetwynd [Chetwynd, 1985]. In this case however there is no geometric interpretation of the steps to follow and therefore strict mathematical rules are obeyed.

Other minimum zone algorithms are proposed to fit specific reference figures, which will be considered separately.

2.6.2 Minimax Circular References and Other Approaches

The general non-linear minimax circular reference fitting problem is to find a reference (a, b), R so as to minimise

$$max \mid \epsilon_i \mid \tag{2.36}$$

for $i = 1, \cdots, N$, where N is the number of data points and ϵ_i is as in equation (2.17). Equivalently, this is formulated as: minimise h subject to

$$-h \leq \epsilon_i \leq h \,, \quad i = 1, \cdots, N \tag{2.37}$$

Anthony and Cox, 1985, proposed an exchange algorithm for finding minimum zone circles based on a interlacing property [Rivlin, 1979] that there are four points in S, the set of data points, so disposed that, if they are ordered according to the angle the line joining them to the common centre makes with a fixed radius, they lie alternately on the outer and inner circles. The algorithm consists in finding four such points. As initial estimate, it is suggested that the linear least squares system given in (2.27) be used to minimise the approximated function given in (2.26), linear in its parameters. It is argued that the success of the iteration process depends on a good choice for two points in S "close to" and two points in S "far from" the centre coordinates of the current estimate. However, no detail is given about the strategy employed.

Ventura, Chang and Klein [1988] present an algorithm for minimax circle fitting based on the misleading premise that the minimax circle is defined by finding the centre of a circle of constant radius which minimises the zone of deviations. So the number of parameters is reduced to three, the centre coordinates and the maximum deviation. Presumably, the radius is assumed as its nominal value. An exchange algorithm is then proposed based on the statement that there are seven possible solutions for each combination of three points, thus it iterates through the possible combinations of points in order to find the one which does not violate any of the data points and has the minimum value for the deviations. It seems however that the zone so defined is minimum for a circle with a particular radius and not the minimax zone as it is commonly understood, that is when the radius is also allowed to vary. Moreover, for a large data set this algorithm seems to be computationally very expensive.

A different approach, using techniques developed in computational geometry is used by Le and Lee [1991] and Lai and Wang [1988], to define similar algorithms for fitting minimax reference circles to data. The algorithms are based on the concepts of medial axis and farthest neighbour Voronoi edges of a polygon. The medial axis of a polygon G is the set of points q internal to G such that there are at least two points on the boundary of the polygon that are equidistant from q and are closest to q [Lee, 1982]. The set of points q is the collection of the centres of the inscribed circles of polygon G. Associated with each vertex v_i of G there is a convex polygonal region V_i such that v_i is the farthest neighbour of every point in the region. This diagram is denoted as the farthest point Voronoi diagram and its line segments are the farthest point Voronoi edges. Any point on these edges is the centre of a circumscribed circle of the polygon G [Shamos and Hoey, 1975 and Preparata and Shamos, 1985]. The algorithms are based on the statement that the intersection of the medial axis and the farthest neighbour Voronoi edges of the polygon are the centres of concentric circles enclosing the polygon, and the circles with minimum radial separation are the minimum zone circles enclosing the profile. Le and Lee [1991] claim that the time complexity of their algorithm is of the order of $n \log n + k$, where n is the total number of vertices of the polygon, and k is the number of intersections between the medial axis and the farthest neighbour Voronoi diagram.

Exchange algorithms for the minimum circumscribed and maximum inscribed circle references [BSI, 1987a] have also been proposed by Anthony and Cox [1985]. It is stated that the minimum circumscribed circle (MCC) has the property that there is at least one triad of points in the set of data points (S) that lie on the MCC and form the vertices of an acute-angled triangle [Hearn and Vijav, 1982]. It is pointed out that a single exception is when two points in S define a diameter of MCC. The same property is claimed for the maximum inscribed circles, although it is argued there may be a number of locally best solutions. This characterisation is the basis of the proposed algorithms. In order to find an initial estimate, a second algorithm is proposed, which in turn starts from the least squares solution of the linear system given in (2.27). However, no information is given about the performance of these algorithms.

2.6.3 Minimax Cylindrical References and Other Approaches

Goto and Iizuka [1977] attempted to discover the minimum zone cylinders from the least squares solution. A search method is given but considered to be too inefficient and an alternative is proposed which uses a weighted least squares approach in which the weights relate to the residuals of an unweighted least squares solution so that the major peaks and valleys are emphasised. This method is of course an estimation of the minimum zone cylinders rather than a solution.

One proposed method for defining cylindricity which does not rely on the usual reference figures is due to Goto and Iizuka [1977] when a deformed cylinder is described in terms of an axis consisting of orthogonal polynomials in z and cross-sections of constant z are described as Fourier series using r and θ (r, θ and z being a cylindrical coordinate system). It seems that this method was first adopted more to demonstrate the use of least squares by which the parameters can be found than for metrological reasons but it is a method of describing the surface that can well be used.

Other methods for cylindricity measurement are considered in a paper presented by Murthy [1982] include the spiral tracing method [Tsukada et al, 1977], the multi stylus method [Kakino and Kitazawa, 1978], the surface development method and the method of the orthographic projections of the axis of the cylinder

[Murthy, 1982].

2.6.4 Minimax Planes and Lines

A computational approach to the evaluation of straightness is presented by Lai and Wang, 1988. This procedure is based on the concept of building a convex polygon in a stepwise manner (see e. g. [Preparata and Shamos, 1985]). Using this method, the authors claim that the minimum zone can be determined.

Traband et al [1989] present an algorithm for finding the minimum zone lines and planes based on the theory of convex hulls [Preparata and Hong, 1977]. The convex hull $H(S)$ of S is defined as the smallest convex set containing S, a set of points in E^2, the two-dimensional space. A supporting line of $H(S)$ is a line passing through a vertex of $H(S)$ such that the interior of the convex region lies to one side of the half-plane defined by this line; thus, two supporting parallel lines will define a zone enclosing S and different zones can be defined. The algorithm is then proposed on the grounds that the minimum zone can be defined by searching for such two supporting parallel lines with minimum separation. The effectiveness of the algorithms is tested by using several data sets and comparing the results with the linear least squares method, and some results show that the least zone is obtained by the proposed method. One minor criticism is that the data sets used were well aligned with the x-axis or xy-plane and therefore no information is given about the performance of these algorithms with data sets not aligned with the appropriate axes.

Another approach for finding the minimax lines and planes is presented by Huang, Fan and Wu [1993]. The minimax method proposed uses the concept of the rotations of enclosing planes (the so called control plane rotation scheme) with respect to a particular contact point at each data exchange step. The procedure starts by defining a trial zone parallel to the least squares best-fit plane and containing the extreme points (in the opposite half-spaces defined by the least squares plane). Then a rule of search is proposed based on the rotation of planes so as to get to the condition where four points are in contact with the enclosing planes. The procedure terminates when one of the following two conditions is satisfied: there are two contact points in each plane such that, when projected onto the lower plane, the line linking the two contact points of the upper plane must intersect the line linking the two contact points of the lower plane; or there are three points on one plane and one point on the other plane such that, when projected onto the lower or upper plane, that single contact point must be on the inside of the triangle formed by the other three points. These conditions are also applied by Chetwynd in the exchange algorithm (see section 2.4.6 and figure 2.5) for minimax plane fitting. It seems that while the exchange algorithm iterates from an optimum but unfeasible solution (it starts with a solution that obey one of the two conditions above but does not enclose all the data points), this algorithm starts with a feasible (all the points enclosed by the trial zone) but not optimum minimax plane and then iterates until the optimum solution is found (one of the two conditions above is satisfied).

2.7 Geometric Tolerance Assessment

The National Physical Laboratory (NPL), UK, has recently released a report addressing the problem of assessment of geometric tolerances [Forbes, 1992]. The approach proposed for tolerance assessment is somewhat in line with the work reported in this thesis.

The general tolerance assessment problem is stated as a constrained problem, in which an optimisation procedure is used to find the parameters of the geometric element that do not violate the form and parameter constraints imposed by tolerance specifications. Few examples are considered, such as circularity and tolerance on radius; the case of template matching is also considered.

It recommends the decomposition of multi-component problems, so that it is possible to solve a sequence of optimisation problems and consequently to reduce the complexity of the original problem. This is illustrated by considering the case of two holes with a specified minimum radius and separated from each other by a toleranced distance. It is advocated that first be found the two maximum inscribed circles and then it be checked whether the centre of the circles lie within tolerance. It is observed however that this decomposition changes the original problem.

Another point considered is the use of approximate methods for simplifying the solution of the original problem. Methods considered include: reduction of the number of parameters; substitution of constrained by unconstrained optimisation; and the replacement of non-linear problems by linear problems.

No algorithm is presented for solving the problem as stated, although it is possible to understand that there is a preference for adopting a *Chebyshev* approximation type of method [Osborne and Watson, 1969].

2.8 Towards Automated Inspection in a Computer Integrated Enviroment

2.8.1 Comparison of Measured Data Points with CAD Data Files

Manual drafting has been replaced, over the past ten years, by modern, computerised systems for defining the geometry of mechanical parts. A great variety of computer-aided-design (CAD) systems are available nowadays, embodying a number of geometric representation schemes. In CAD systems, the nominal geometry of the object is constructed and stored in a file. Once the part geometry has been defined, computer-aided-manufacturing (CAM) systems can be used in order to generate the machining part programme which will generate the actual part. After machining, the part is inspected for conformity with design specifications.

Research work reported by Schneeberger [Schneeberger et al, 1983], presents a technique for the determination of errors in part geometry by comparison of a measured part database with a part nominal database. The nominal database represents the desired part geometry generated by the CAD system and transferred into the machining part programme processor. The measured database is obtained by coordinate measurement of surfaces on a manufactured part. A least squares best fit technique is used to calculate surface parameters from measured data, and the parameters of the fitted surface are used to transform the database from the machine coordinate system to a workpiece coordinate system. The nominal part database is also transformed into the same workpiece coordinate system. The errors are calculated by computing the distance from the measured points to the nominal part surface. It is reported that this technique was tested for cone, cylinders and plane fits. However, there is no discussion about the errors introduced by the rotation and translation matrices, in the process of coordinate transformation, which obviously has to be considered, nor about the computational cost involved.

A commercially available software package, Perceval [Sediscad, 1993], apparently works in line with the technique presented above. As described in its advertising leaflet, the error is obtained by projecting the measured point on the theoretical surface stored in the database.

This technique can achieve little more unless used in conjunction with error compensation techniques. A technique for modifying the manufacturing process based on the error information is presented by Duffie [Duffie et al, 1984]. In this paper, the work is concentrated on automated modification of surface geometries using parametric surface patches (see e. g. [Faux and Pratt, 1979]), which are commonly used for sculptured surfaces. A compensation strategy is applied in which a modified surface patch database is created using patch fitting techniques.

2.8.2 Geometric Tolerancing in Solid Modeling Based CAD Data Files

The most powerful computer-aided-design systems are based on solid-geometric modelling systems (solid modellers for short) for geometry definition [Requicha and Volker, 1982]. Solid-geometric modelling systems are expected to be the future medium of communication for geometrical specifications of mechanical product designs. But the solid modellers available today are for the most capable only of representing the nominal geometry of parts. In general, two dimensional projections are derived from a solid model, and these views are annotated with dimensions and tolerances (see [Roy, Liu and Woo, 1991] for a review of automatic dimensioning systems).

Tolerancing annotations contain a great deal of implicit information, which is obvious to the intelligent and experienced production engineer, but is not good for computer implementation. Ideally, the geometric meaning of tolerancing information should be defined mathematically and incorporated in solid modellers for automatic planning and analysis. This would make it possible, for instance, to inspect whether features meet their design specifications by checking whether the data acquired using a measuring machine (after suitable frame transformations) lie within tolerance zones in the solid model, which might be defined as regions of space surrounding the object's nominal geometry.

Several research workers have explored the field of Dimensioning and Tolerancing (D & T) and studied several aspects of its implications for the successful integration of computer-aided design and manufacturing (see e. g. [Roy, Liu and Woo, 1991]). This review concentrates on aspects of the representation of tolerances in solid modelling.

The lack of a formal theory for the representation of tolerances in solid modelling was first identified by Requicha, who developed a theory based on the *variational class* concept [Requicha, 1983, 1984]. Variational classes are families of objects that are similar to a nominal object, are interchangeable in assembly, and are functionally equivalent. By his definition, a variational class is to be represented as a nominal object together with a collection of assertions about the object's features. These assertions define geometric constraints that specify the allowable variations from the nominal object's shape. He proposed the concept of *nonparametric zones* [Requicha, 1984], as opposed to the parametric approach, defined as the set of points that are within a given distance of the nominal features. A tolerance zone is created by the Boolean set difference between two *offset* objects with specified maximum and minimum offset values.

For parameterised objects, the maximal and minimal objects are defined by the maximum and minimum values of the defining dimensions. For non-parameterised objects, the tolerance zone is created by offsetting the object by equal amounts on either side of the nominal. Unlike parameterised objects, the maximal and minimal objects may not be of the same shape. Requicha considers this to be an advantage because this class does not force objects to have perfect shape (same as nominal). The tolerance information is specified as a set of geometric attributes of the surface features of an object boundary, and it dictates the offsetting criteria for the boundary surfaces. A formal theory for *offsetting* operations is discussed by Rossignac [1986].

The parametric approach was first proposed by Hillyard and Braid [1978] and further refined by Lin, Gossard and Light [1981] and Light and Gossard [1982]. In this approach, an object is treated as a parameterised shape where either the parameters are explicitly specified by the user or determined from a set of constraint equations involving geometric relationships. Tolerances are considered to be small changes in the defining dimension parameters. The approach has typically been applied to objects built from planar faces and straight line edges or right circular arcs. Since tolerances can only be supported as attributes of parametric dimensions, the classes of tolerances that can be supported is very limited. Tolerances such as circularity and cylindricity cannot be supported because of the restrictions on geometry. Position tolerances based on resolved entities (axis, mid-plane) cannot be supported as well because the vertices of these entities are not available.

Using his theory of tolerancing, Requicha and Chan [1986] have implemented the representation of geometric features, tolerances and attributes in a CSG-based (Constructive Solid Geometry) modeller, PADL-2 [Brown, 1982]. The basic structure proposed for representing features and attributes is a graph called a variational graph or simply VGraph, which associates the variational information with the solid model. It utilises what is called *2D intersection* set operators that allow one to reference only a portion of a face (called VFace) by intersecting the object with a virtual object. Attributes, such as tolerances, can be attached to VFaces.

Attribute nodes are provided for classes of tolerances. In this respect, Requicha proposed to replace various special-case tolerances used in current practice (e. g. roundness, flatness) with a single form tolerance that applies to all features. In the same way, single surface-orientation tolerance and position tolerance were proposed, with the difference that these two require datum specifications. It also introduced the concepts of master datum system and extended and symmetric features.

One criticism [Roy, Liu and Woo, 1991] is that, as the handling of dimensions and tolerances in the general case requires the ability to access the bounded entities of objects, the tolerance theory of Requicha raises some manipulation problems during implementation; moreover, this kind of representation differs in some respects from the ISO system [e. g. BSI, 1990] for dimensioning and tolerancing. According to Roy, Liu and Woo [1991], this VGraph system has a limited ability to describe design tolerances.

A conceptual framework for tolerance representation and analysis based on CSG solid modellers is presented by Elgabry [1986] somewhat along the lines of Requicha, but without a mechanism to refer to partial faces. Tolerances are attached to whole faces of primitives, which are defined by size, location and orientation vectors with tolerance values. Therefore, users must place appropriate primitives at places where tolerances are to be specified.

Jayaraman and Srinivasan [1989] have examined the issues of representing the geometric tolerances in solid models from the perspective of functional requirements related to the geometry of mechanical parts. Their research is mainly concerned with the positioning of parts with respect to each other in assembly, and with maintaining material bulk in critical portions of parts. They have extended the concept of offset boundaries to adjoining parts in an assembly by means of a *virtual boundary* used as a divider between them. They developed specific *virtual boundary requirements* (VBRs) to reflect the required functional conditions of the assembly, and then discussed the theoretical basis of the interpretation of those virtual boundary requirements with the help of the theory of solid-model-based offsetting, as proposed by Rossignac and Requicha [1986].

However, virtual boundaries have the same limitations as offset boundaries.

A different representation scheme is presented by Johnson [1985] for solid modellers based on boundary representation (B-rep, see e. g. [Requicha, 1982]). However, this representation scheme is applicable only for location and size tolerances, and it is limited to geometric entities such as planar faces, cylindrical faces, conical faces and spherical faces. Roy and Liu [1988] showed the necessity of having a hybrid CSG/B-rep data structure for the tolerance representation so that the advantages of both CSG and B-rep models can be exploited.

Another representation model is proposed by Turner and Wozny [1988], mainly for tolerance analysis problems. A conceptual framework is presented, based on constructive variational geometry, which is intended to be used for evaluation of tolerance variables and design variables as function of the model variables.

Other representation schemes are also reported [Roy, Liu and Woo, 1991]. As this is new field of research, further research is still being carried out and therefore no conclusive and comprehensive theory and software implementation has yet been established. It may turn out that changes in the ISO-tolerance system are necessary in order to accommodate a more comprehensive tolerancing theory. It is also worth noting that a proposal has been put forward for adopting *Vectorial tolerancing* in place of the present ISO-tolerance system [Wirtz, 1992].

Chapter 3

Inspection of Roundness Features

3.1 Introduction

Circularly shaped components are one of the commonest industrial forms, from aerospace and automobile to electro-electronic and white goods industries. As a consequence, inspection of out of roundness of a cross section of a nearly circular component is an economically important area. Also, being a closed figure, circularity is relatively easily expressed mathematically. This combination has often led to it being used to develop and illustrate metrological techniques, a tradition that will be followed here in introducing a novel strategy. This chapter discusses the idea of a GO-NOGO inspection software and presents the implementation of an algorithm for the inspection of combined roundness and centre position or eccentricity errors of nominal circular components [Carpinetti and Chetwynd, 1992]. The extension of this approach to other figures will be covered in later chapters.

The numerical assessment of out of roundness is by measuring the peak to valley deviation of the actual profile from a reference circle fitted to that profile (see chapter 2 for a review of methods and algorithms for roundness measurement).

The centre position and the radius of the reference may themselves be important parameters, for example in measuring the distance between two holes or the eccentricity of two related shafts. In this case a separate concentricity or position tolerance would be required.

Partly for historical reasons and partly because efficient processes have been developed, there is commonly a difference of approach between the specification of tolerances in design standards and the assessment methods specified in metrology standards. This has caused little practical difficulty on simple measurements. However, there may be conflicts when several tolerances are used in combination, for example circularity, centre position and absolute radius.

A highly effective approach to the solution of best-fit roundness references has been used for many years in which radial coordinate data representing the workpiece profile is expressed relative to an origin that lies not far from the best fit centre, a condition that arises naturally from the action of specialised roundness measuring instruments. The circle is substituted by a limaçon having the same parameters, with the errors of so doing normally less than the resolution of the measurement instruments, providing the ratio of eccentricity to radius is kept below about 0.01, [Chetwynd, 1979a] (see chapter 2 for more details). This parameter linearisation offers great computational benefits and will be followed here.

3.2 Boundary Value Roundness References

Standards define three alternatives to least squares as criteria for reference fitting in roundness measurement. These are the minimum radius circumscribing circle (MCC or ring gauge circle), the maximum radius inscribing circle (MIC or plug gauge circle) and the minimum radial zone circles. These fitting problems are identified mathematically with problems of constrained optimisation. Thus, the ring gauge circle fitting problem for instance can be formulated as: find the centre (a, b) of the circle that has minimum radius R while maintaining the condition that the profile, represented by the data points $(r_i, \theta_i), i = 1, \cdots, N$, where N is the number of data points (here given in polar coordinates), is enclosed by the reference. That is to say, minimise R such that each data point be subject to the constraint

$$a\cos\theta_i + b\sin\theta_i + [R^2 - (a\sin\theta_i - b\cos\theta_i)^2]^{1/2} \geq r_i \tag{3.1}$$

which is based on the equation of a eccentric circle [Whitehouse, 1973]. This optimisation process can be well represented using the concept of parameter space. A parameter space, or solution space, is a conceptual frame in which each parameter is plotted along an orthogonal axis. This provides a geometrical view of the problem which is intuitively useful when there are three or fewer parameters and mathematically useful in any number of parameters. The total n-dimensional space described by those axes contains one point for each conceivable combination of parameter values. In such a space both the objective function (that is the function being optimised) and any constraints appear as hyper-surfaces. Each constraint divides the parameter space in two open half spaces, with all the feasible parameter combinations lying to one side and the infeasible ones to the other side. Only the feasible region contains valid combination of parameters.

For the simplest illustration, consider the set of constraints represented in equation (3.1) for the case when $b = 0$. Each data point will generate a constraint whose equality condition produces a boundary line in an (a, R) space that plots as shown in figure 3.1 for an arbitrary number of constraints, dividing the space in feasible and infeasible regions. The feasible region for circumscribing circles lies above the boundary line which, since the boundary is parabolic, will be convex. The objective function is linear and the contours of constant value are lines parallel to the a-axis. Thus it is seen that a convex feasible region is defined, that is any two points within it or on its boundary may be connected by a straight line lying wholy within the region, and therefore a unique solution to the ring gauge exists, corresponding in figure 3.1 to point j.

The equivalent plug gauge circle fit will be equation (3.1) with the direction of optimisation and inequality reversed, so the constraint curves will be identical

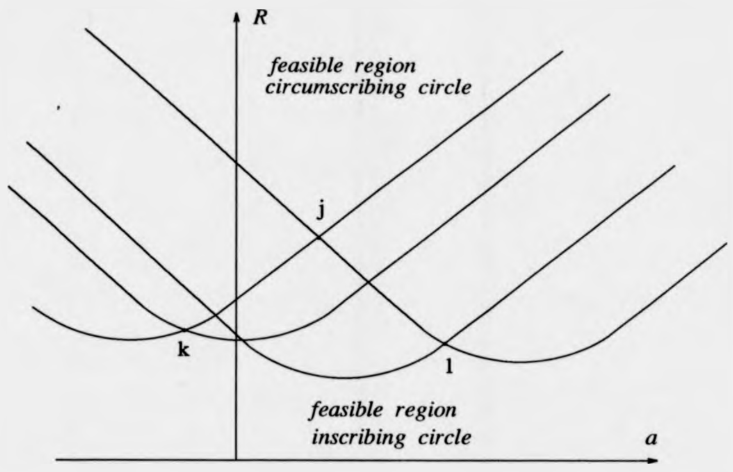


Figure 3.1: Parameter space representation of circumscribing and inscribing circle constraints.

to those of the ring gauge circle, but the feasible region will now lie below the line. This region is non-convex, as is the intersection of the region from all the constraints. The contours of the objective function are, as before, lines parallel to the a-axis. In this case the non-convexity may cause multiple local maxima, illustrated in figure 3.1 by points k and l. If these have to be found by an optimisation algorithm, different alternative maxima can be obtained by alteration of the starting point of the iterative process. There is, however, no guarantee that the local maxima so discovered will include the global maximum.

The minimum zone circle problem has four parameters and hence a four dimensional space, which makes visualisation impossible. However, it can be shown that the feasible region is not necessarily convex and profiles having two distinct solutions have been invented [Chetwynd, 1979a].

On the contrary, all the equivalent limaçon approximations generate linear constraints. For example, the ring gauge limaçon reference to a set of N data points $(r_i, \theta_i), i = 1, \cdots, N$ is expressible as: minimise $Z = R$ subject to

$$a \cos \theta_i + b \sin \theta_i + R \geq r_i \tag{3.2}$$

where (a, b) and R are the reference parameters. It is a geometric consequence

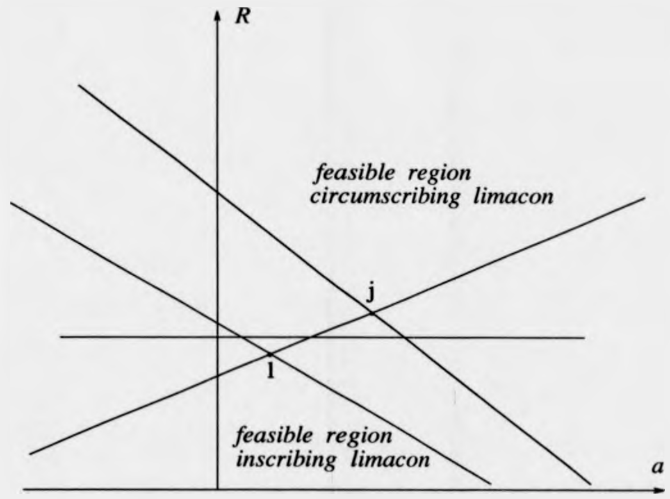


Figure 3.2: Parameter space representation of circumscribing and inscribing limaçon constraints.

of the constraints being linear in the parameters that the feasible region will be convex and that the single best point will always be at an intersection. This can be seen from the two dimensional parameter space representation of the circumscribing and inscribing limaçon references, figure 3.2, where points j and l, represent the optimum for the circumscribing and inscribing limaçon respectively.

Non-linear constraints do not generate plane figures in parameter space and so the search must be over a surface or, worse, if the region is non-convex, over the whole feasible region. This is essentially why it is both more expensive to compute non-linear problems and problematic to guarantee globally optimal solutions.

The limaçon approximation makes possible the use of linear programming techniques, the relevant theory of which is reviewed in the next section.

3.3 Basic Concepts in Linear Programming

A linear programme is an optimisation in which the objective function and all the constraints are linear in the parameters. Using vector notation, it can be expressed as: maximise $z = \underline{c}^T \underline{x}$ subject to

$$\underline{A}\underline{x} \leq \underline{b} \tag{3.3}$$

where for m positive parameters, $\underline{x}$, and n constraints, $\underline{c}$ is an m-vector, $\underline{b}$ an n-vector and $\underline{A}$ an mxn matrix.

Linear programmes are one of the fews classes of constrained optimisation that have guaranteed convergence onto a unique solution. Theorems of linear programming indicate that the optimum solution occurs when each of the constraints which is actively limiting that optimum is satisfied to its limit by one of the parameters (see e. g. [Hadley, 1962]). Hence, only certain combinations of parameter values need be examined. An orderly search through these may be obtained by using the simplex method in which iterations involve only elementary row operations on the matrix-vector representation given by equation (3.3). Simplex organises these vectors as a partitioned matrix (a tableau)

$$\begin{bmatrix} \underline{K} & \underline{b} \\ \underline{c}^T & Z \end{bmatrix} \tag{3.4}$$

where $\underline{K}$ is $\underline{A}$ augmented by an nxn identity matrix and $\underline{c}$ is correspondingly extended by n zero elements. This appends n "slack variables" to the original parameters. If the i^{th} parameter is limiting a particular constraint, the column $\underline{K}_{i}$, in $\underline{K}$, will have value +1 in the row corresponding to that constraint and zero in all other elements. The set of defining parameters so identified form the "basis". Initially the basis is the n slack variables. Iterations attempt to match parameters to constraints in such a way that Z is rapidly maximised. The feasibility of the current iteration is maintained by ensuring that no constraint is ever violated, that is, that no element of $\underline{b}'$ becomes negative. The prime indicates the vector which currently occupies the position originally occupied by $\underline{b}$. At each iteration the largest positive element of $\underline{c}'^T$ is chosen and its column brought actively into the solution. When no positive elements remain in $\underline{c}'^T$, optimality has been achieved and the solution values are readily interpreted from the tableau. Equality constraints, which must be always exactly satisfied, do not take slack variables but equivalent "artificial variables" are used as a device for starting the iterative procedure in an orderly manner. By definition, artificial variables cannot remain present in any feasible solution of the tableau.

At any iteration, the columns which originally consisted of the identity matrix carry a complete and interpretable record of the row transformations carried out on the tableau. Likewise, the columns of the current basis carry the same information in the inverse of their original form. The computationally efficient method of Revised Simplex does not update the full tableau but merely notes what would have been done at each iteration. Elements are only updated when specifically required for calculations.

While the total computation required rises with both m and n, it is particularly sensitive to n, the number of constraints, as the work required relates to that of inverting nxn matrices. It may, therefore, be advantageous to use a dual programme. For any mxn linear programme (termed the primal), an nxm dual may be defined as:

$$\begin{bmatrix} \underline{K} & \underline{c} \\ \underline{b}^T & Z \end{bmatrix} \tag{3.5}$$

where $\underline{K}$ is now the augmented form of $\underline{A}^T$ and the optimisation has changed from minimisation to maximisation or vice versa. It contains exactly the same information as the primal, subject to the correct relative interpretation of specific elements.

Normal simplex methods require sign-definite parameters in order to ensure the sign-definiteness of elements of $\underline{b}^T$ in the feasible region. This cannot be guaranteed with metrological data and so each parameter is replaced by an ordered pair having equal magnitude but opposite sign. Even so, the number of constraints usually dominates the number of parameters. Thus a ring gauge limaçon fit involves six parameters and the minimum zone seven. Typical measurements on specialised roundness measurement instrument involve several hundred profile

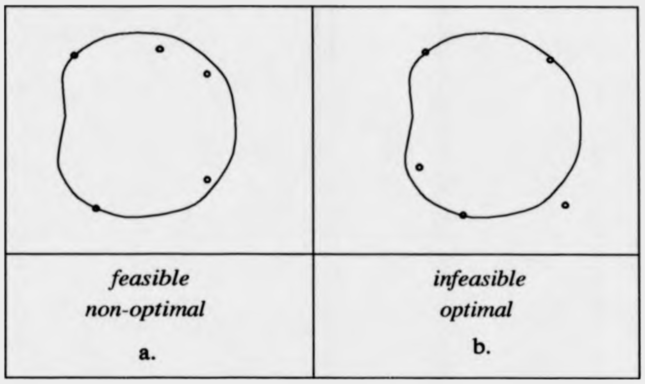


Figure 3.3: Geometric interpretation of primal feasibility (a) and dual feasibility (b).

points each generating a constraint, two in the case of minimum zone. CMM data sets are generally much smaller, but still usually exceed the number of constraints when good measurement practice is followed. Since each constraint generates a slack variable while the variables generate no extra terms in the tableau (each primal variable causes a dual constraint and vice versa), an increase in efficiency can be obtained by solving the dual form of the boundary limaçon problems.

In moving from the primal to the dual, the roles of vectors $\underline{b}$ and $\underline{c}$ are interchanged. At each iteration, any state which is feasible in the dual (that is all elements of $\underline{c}$ positive) corresponds to an optimal condition in the primal but since the dual is not optimum the primal is infeasible. For the ring limaçon case, for example, the geometric interpretation of an optimal, infeasible condition is a limaçon which does not enclose all the points but which is the smallest one that can enclose those within it. The primal feasibility condition amounts to starting with a figure which is too large but which certainly encloses the profile and then shrinking it to the smallest radius which still encloses the profile (figure 3.3a). Dual feasibility would entail choosing initially a figure which is the smallest to enclose some, but not necessarily all, of the data points (in the primal, optimal but infeasible) and then expanding it as little as possible so as to include all the data (figure 3.3b).

The geometric interpretation of the requirements for dual feasibility lead to the definition of the rules that govern the exchange algorithms for the three boundary references presented in chapter 2 [Chetwynd, 1980 and 1985]. In following such rules, the exchange of data points is unique at each iteration since it is identical with that chosen by the simplex method on the dual linear programme, which is known to converge monotonically. This guarantees that cyclical exchanges do not occur and that iterations move monotonically towards an optimum solution.

3.4 Formulation of the Inspection Problem

Inspection of geometric features of components in the context of a manufacturing process requires only that it be established that the measured features are contained within an acceptable tolerance zone. It does not require necessarily that any form of "best fit" be found. However in practice it has been usual, since standard algorithms exist, first to find an optimal reference and then to test its parameters and the deviation of the features from it against the design specification, as discussed in section 1.2 (illustrated in figure 1.2) and in chapter 2.

Considering again the ring limaçon reference, common practice seeks to minimise the reference radius R while maintaining the condition that the profile is enclosed, that is subject to the mathematical constraints represented in (3.2). The design specification for out of roundness actually imposes another set of constraints: the deviation of any data point r_i from the limaçon reference must be less than the tolerance value specified in design, say t_r. This can be written as

$$(a\cos\theta_i + b\sin\theta_i + R) - r_i \leq t_r \qquad (3.6)$$

The combination of this set of constraints with the set in (3.2) may or may not define a region of feasible solutions, that is combinations of parameter values that simultaneously satisfy all the constraints. This situation can again be illustrated graphically using the concept of parameter space.

Consider again the case when $b = 0$. Including the extra constraints of (3.6),

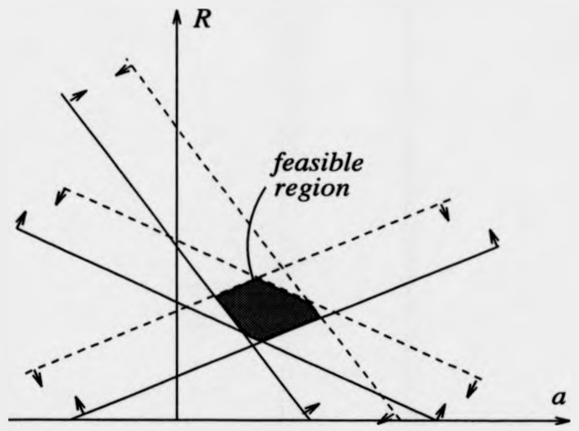


Figure 3.4: Representation of feasible region in parameter space

each data point will generate a constraint pair which produces boundary lines in an (a, R) space as

$$R \geq r_i - a\cos\theta_i \tag{3.7}$$

$$R \leq r_i + t_r - a\cos\theta_i \tag{3.8}$$

that plots as shown in figure 3.4. The inequality direction of the constraint represented in (3.7) (represented as a solid line in figure 3.4 dictates that the feasible region must lie above the lines. The set of boundary lines represented in (3.8), imposed by the form tolerance specification, will be parallel to the boundary line defined by the corresponding constraint (that is, generated by the same data point) and separated from it by the tolerance value. In this case, examination of the inequality directions of the constraints shows that the feasible region must lie below the lines (dashed line in figure 3.4). The combination of the two feasible regions defined by the two set of constraints may define a set of valid parameter values, a single solution (a feasible point) or it may be empty, meaning that no feasible solution exists. For the three-dimensional case, the region, when defined, will be a polyhedron.

Any feasible solution provides an acceptable solution to the problem, since it implies that there are two circles centred at (a, b) and with radius R and $R - t_r$

enclosing the data. In this case the constraints must not be violated, that is to say the profile must be enclosed, but the radius need not be minimal. If there is at least one feasible solution the out of roundness is within tolerance and the workpiece passes inspection without any need to optimise the parameters. Otherwise, the out of roundness is larger than that specified in design and the workpiece fails inspection. This situation is directly comparable to the manual use of an annular template trying to enclose the profile.

The same sort of reasoning may be applied to the other two boundary references. For the plug limaçon reference, because of a change in the inequality, the feasible region is shifted in relation to that of the ring limaçon problem but is otherwise essentially the same. The formulation of the minimum zone problem is: find the limaçon parameters (a, b) and R and separation h such that

$$\begin{aligned} &a\cos\theta_i + b\sin\theta_i + R + h \geq r_i \\ &a\cos\theta_i + b\sin\theta_i + R - h \leq r_i \\ &2h \leq t_r \end{aligned} \tag{3.9}$$

where again t_r is the roundness tolerance value. This requires a four dimensional space and makes visualisation impossible. However, if the parameter b is assumed to be zero for example, these constraints will plot as planes in an (a, R, h) space. Each pair of boundary planes will define a feasible "zone" between them, and the intersection of all of them, if not empty, will define a region of feasible solutions to the simplified problem. Mathematically, the full four-dimensional situation is no more complex. For inspection, it is not necessary for h to be minimum but only for it to be adequately small. Thus, assuming that $h = t_r/2$ and removing h from the constraints and from the parameters set, it yields constraints essentially the same as those of (3.2) and (3.6).

In addition to the inclusion of the form specifications in the assessment problem, there are many situations where not only the out of roundness but also the centre position of the cross section of a circular component is to be controlled. This would be specified by a geometric tolerance of concentricity besides the roundness tolerance. The concentricity tolerance provides a limited zone for the deviation of the position of the centre or axis of the toleranced feature about its ideal position. As shown in chapter 1, there can be difficulties in using the centre of a best-fit reference for this purpose. It is better, therefore, to extend the previous approach with yet a further set of constraints.

To satisfy the geometric tolerance this set should place a circular boundary central on the nominal position within which the trial centre must lie. Here, in keeping with the strategy adopted, we seek an approximation to the circular zone that corresponds to it adequately for practical purposes but which generates only linear constraints in the problem formulation. One alternative could be to approximate the circle piecewise by a series of straight lines, that is the circle is represented as a regular, even-sided polygon. Chetwynd [1980] considers briefly the use of polygons as alternatives to limaçons for approximating the ring and plug circle fits, but not their use in the present context.

Each side of a polygon is a section from a straight line in the measurement plane and so can be described by parameters in which it is linear. So providing that all the sides can be handled together in a simple way the polygon reference is capable of a linear parameterisation and consequently should be relatively easily optimised.

The constraint set then relates to a set of lines of pre-defined slopes each having the same perpendicular distance from an origin corresponding to the nominal position of the feature. The simplest reference polygon consists of a square in which the sides lie parallel to the coordinate axes. The larger the number of sides taken, the closer the representation will become, but at the expense of increased computing since each side generates a constraint.

According to the degree of concern about marginal violations (alternatively, of the relative importance of false positive and false negative decisions in the inspection), the polygon may be chosen to be inscribing to, an average of, or even circumscribing to the true circular tolerance. With the polygon set to give equal interior and exterior deviation from the circle, an octagon would give a maximum error of 4 % of the tolerance value, while figures of 6 and 12 sides would give errors of 7 % and 1.7 % respectively. Table 3.1 presents the errors of

Maximum Error (ϵ) of Polygons (%)			
Sides θ	Circumscribing $\epsilon = \frac{1-\cos\theta}{\cos\theta}$	Inscribing $\epsilon = 1 - \cos\theta$	Average $\epsilon = \frac{1-\cos\theta}{1+\cos\theta}$
4	41	29	17
6	15	13	7
8	8	8	4
12	3.5	3.4	1.7
16	2.0	1.9	1.0
20	1.2	1.2	0.6
32	0.5	0.5	0.25

Table 3.1: Maximum error of approximation of a circle by a polygon.

inscribed, circumscribed and averaged polygons for increasing number of sides. It can be seen that the error of an average polygon with number of sides over 16 is less than 1 % of the tolerance value.

For ease of illustration, a circumscribing square is used here, which has an excessive maximum error of about 41 % at its diagonal. This simplifies discussion without seriously affecting conclusions about the efficiency of the approach since the extra constraints of moving to an octagonal figure, say, will not greatly increase the size of the overall constraint set. In addition, moving to inscribing or averaging fit, so as to reduce the approximation error, has no effect in terms of computational efficiency as it merely requires extra scaling factors.

Thus, assuming that the nominal position is at the origin of the coordinate system used as reference, the square boundary region will be generated by the constraints

$$\begin{aligned} -t_c/2 \le a \le t_c/2 \\ -t_c/2 \le b \le t_c/2 \end{aligned} \tag{3.10}$$

where t_c is the concentricity or centre position tolerance value specified in design and it is assumed that the

Going back to the parameter space representation, a polygon boundary region will define a prismatic figure in a three-dimensional space, with its faces parallel to the R axis. For a square boundary region, the two dimensional representation shows two boundary lines, limiting the maximum and minimum value for the

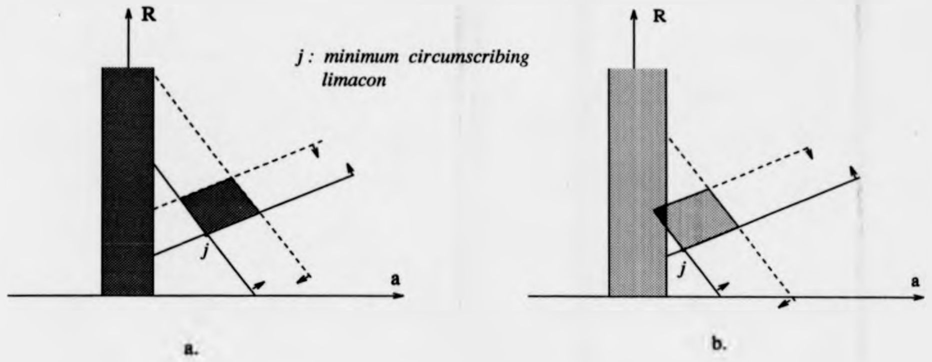


Figure 3.5: Representation of feasible region of a inspection problem: (a) fail inspection and (b) pass inspection.

parameter a.

A further set of constraints is thus added to the original problem, which express the allowed error of the feature position from design specification. Therefore, using the ring limaçon reference, the simultaneous inspection of roundness and centre position against design specification is formulated as: find any set of parameters (a, b) and R such that

$$\begin{aligned} & a\cos\theta_i + b\sin\theta_i + R \geq r_i \\ & (a\cos\theta_i + b\sin\theta_i + R) - r_i \leq t_r \\ & -t_c/2 \leq a \leq t_c/2 \\ & -t_c/2 \leq b \leq t_c/2 \end{aligned} \qquad (3.11)$$

for $i = 1, \cdots, N$, where N is the number of data points and t_r and t_c are the design tolerances of roundness and centre position (nominally at origin) respectively. This will be referred to as the inspection problem.

If the combination of constraints (3.11) results in infeasibility, there is no reference figure with centre limited within the pre-defined position and out of roundness within tolerance. Consequently, the workpiece fails inspection.

It is worth repeating the observation that when the centre coordinate of the optimum reference lies outside the feasible region for the centre position, that is the centre position tolerance zone, two possibilities exist: either no feasible point is defined, in which case there is no reference figure with roundness error and centre position within specifications (represented in figure 3.5a) or, although the optimum reference lies outside the boundaries for centre position, a feasible region is defined (represented in figure 3.5b), in which case at least one reference figure can be found which defines roundness and centre position errors less than the tolerance (as illustrated in figure 1.3). Consequently, the workpiece passes inspection. In this case the component would fail inspection when measured using a conventional best-fit algorithm. This situation is illustrated in figure 1.3.

3.5 The Search for a Feasible Solution

It is frequently necessary to compute an initial feasible solution in order to solve a problem in linear programming. When the solution is by the Simplex Method this is done by adding artificial variables to the set of constraints, and then iterating them out of the solution basis (see, for example, the "two phase method" [Hadley, 1962).

The number of artificial variables depends on the number of equality and $\geq$ constraints and may be as high as the number of constraints. The geometrical formulation of the fitting problems normally has many constraining equations: each data point generates at least one constraint and there may be up to several hundred points in a data set.

The use of dual techniques to solve the inspection problem, as it is done for boundary reference best-fitting for instance, is not appropriate. This is due the fact that all dual based methods relate directly to the real geometry only at the optimal solution (a dual feasible solution is geometrically interpreted as a reference with minimum radius but not enclosing all the data points). The real geometry is described by the primal formulation only. Therefore, only by using primal techniques it is possible to stop an iteractive procedure part-way through on the basis of a geometric condition.

In the revised simplex method, the number of iterations required to eliminate the artificial variables from the basis at least equals the number of artificial variables. If N is the number of constraints, the basis matrix is of order $N + 2$, the solution vector has $N + 2$ elements and the total number of arithmetic operations required to move from one iteration to another will be approximately $(N + 2)^2 + 9(N + 2) + N$. So, it will be required approximately N sets of $(N + 2)^2 + 9(N + 2) + N$ operations to iterate the artificial variables out of the solution basis and get a basic feasible solution, if one exists [Hadley, 1962]. This technique thus seems unsuitable for our purpose: it requires too many computationally intense iterations.

Fletcher [1970a] addressing such difficulties, introduced a relatively straightforward algorithm for obtaining an initial feasible solution that provides a starting point for linearly constrained optimisation problems. This method does not depend on the addition of artificial variables and so is computationally attractive for our purpose. Fletcher's algorithm does not seem to be widely known, nor often used. The reason may be that it requires a different iterative procedure to that used to drive linear programmes to optimality and that on balance implementations based on only one procedure are preferred for most applications. Since we do not seek optimality, this criticism is not relevant: only the Fletcher algorithm iterations will be executed.

If we consider a multi-dimensional parameter space, then every linear constraint will form a hyper-plane in that space, with all the feasible parameter combinations lying to one side and the infeasible ones to the other side. The intersection of a set of such planes equal in number to the number of parameters will define a point, called in this context a vertex. If the parameter values associated with a particular vertex violate any other constraints, then that vertex is non-feasible. Otherwise, by definition, it must lie on the very edge of the feasible region, that is a continuous region of the parameter space that corresponds to values of the parameters that do not violate any of the constraints. The feasible vertices define the points of a convex (hyper-) polyhedron.

The simplex algorithm of linear programming is an iterative procedure for scanning the feasible vertices in an orderly progressive manner so as to move rapidly to the best. The artificial variables approaches set up additional constraints so that there is a chain of vertices from the origin (from where the search can start automatically) to the vertices of the real problem. The expansion of the computational effort arises because the reformulated problem contains many more parameters which generate many additional feasible vertices that may be used in the iterative procedure. Both the work in each iteration and the number of iterations required are increased thereby.

Fletcher's algorithm also seeks vertices, but it can work with non-feasible vertices in the original problem in a way not open to more conventional methods. It requires more computation with the variables defining the current vertex to decide on each move but in problems of the type we have this is more than compensated by the smaller total number of variables involved in that computation. It starts at an arbitrary vertex, which is tested for feasibility. Assuming that this is infeasible, then the search moves to an adjacent vertex. Only moves in directions along the "edges" that define the current vertex need be examined and the method of Lagrange multipliers (well known in other optimisation contexts) provides in effect a measure of how likely it is that the situation would be improved by moving in each direction through the parameter space. The steepest position gradient of the degree of constraint violation with distance indicates the favoured search direction. If the gradient of the sum of violated constraints is negative or zero in all directions then it is proved that no feasible point exists [Fletcher, 1970a], that is the problem as formulated is infeasible. Otherwise the constraint violated in the plane of steepest gradient is removed from the vertex-defining set (called the basis). The reduced set defines the hyper-line in the parameter space which is searched. At some point along this line the constraint will cease to be violated and a new vertex is then defined. However, before this position is reached, another constraint plane may be encountered. If this happens the new constraint is brought into the basis. In either case we obtain a new vertex which is closer (in a numeric sense) to feasibility than the previous one. This forms the starting point for the next iteration, which follows the scheme just described. Iterations continue until either a feasible vertex is reached or the gradient test indicates that the problem is infeasible. One feature of the algorithm of particular interest to the current application is that it provides definitive indications of infeasibility in a small number of iterations. Many methods that optimise efficiently are inefficient at identifying infeasibility.

To prevent a search wandering off across parameter space if a constraint set defines an open polyhedron, all variables should be subjected to upper and lower bound values. Technically, this introduces two extra constraints per parameter compared to the minimal linear programme. However, just as the non-negativity of parameters does not magnify the size of classical linear programming formulations, the bounds make little practical difference to most problems.

The next section provides a brief review in mathematical formulation of the steps in the iteration since the method is not widely documented.

3.6 Algorithm for Calculating Feasible Points for Linearly Constrained Optimisation Problems [Fletcher, 1970a]

We wish to find a feasible solution for the set of m linear constraints in n variables $\underline{x}$ that can be expressed generally as

$$[I, -I, C]^T \underline{x} \geq \begin{pmatrix} \underline{l} \\ -\underline{u} \\ \underline{d} \end{pmatrix} \tag{3.12}$$

where $\underline{l}$ and $\underline{u}$ are the lower and upper bounds respectively, C the coefficients of the general constraints (each column of C corresponds to the coefficients of one constraint), and I the unit matrix. In the present context, $\underline{x}$ corresponds to the set of geometric parameters and each row of the sub-problem $C^T\underline{x} \geq \underline{d}$ corresponds to one constraint of the set of constraints expressed in (3.11). Let $\hat{\underline{x}}$ be a trial vertex which is not feasible. Each iteration is described by the following steps.

step 1. find $\underline{\nabla\phi} = \sum \underline{c}_j$, the gradient of the sum of the constraints which are violated at $\underline{\hat{x}}$, where $\underline{c}_j$ is the j^{th} column of $[I, -I, \bar{C}]$). Note the sum is only over those columns that are violated.

step 2. determine $\lambda_i, i = 1, \cdots, n$, the Lagrange multiplier corresponding to $\underline{\nabla\phi}$

$$\lambda_i = \underline{b}_i \underline{\nabla\phi}(\underline{\hat{x}}) \tag{3.13}$$

where $\underline{b}_i$ is the i^{th} row of B^{-1}, where B (called the basis) is the matrix whose columns are the coefficients of the constraints which define the vertex $\underline{\hat{x}}$ of the current iteration. If all the λ_i are zero or negative, it is proved that no feasible point exists. Otherwise, the direction of search is determined by the vector $\underline{u_s} = \underline{b}_i$, where i is the index of the largest positive λ_i. The vector $\underline{u_s}$ represents in fact the line generated when the inequality constraint corresponding to the i^{th} column of the matrix B is removed.

step 3. now search along direction $\underline{u_s}$ for intersection with other constraints by examining the columns $\underline{c}_j$ of $[I, -I, \bar{C}]$. If $\underline{u_s}^T \underline{c}_j \geq 0$ *and* the constraint corresponding to $\underline{c}_j$ is violated at the vertex, then that constraint will become non-violated at a point along $\underline{u_s}$ distant by

$$D = \frac{d_j - \underline{c}_j^T \underline{\hat{x}}}{\underline{u_s}^T \underline{c}_j} \tag{3.14}$$

where d_j in the present context corresponds to $r_i, i = j$, the radial ordinate of the i^{th} data point. The maximum value of D, α say, over the set satisfying this condition gives the furthest point at which a violated constraint intersects $\underline{u_s}$.

If $\underline{u_s}^T \underline{c}_j \leq 0$ *and* the corresponding constraint is neither violated nor in the basis then that constraint intersects $\underline{u_s}$ at a distance given by equation

3.14 and moving further in that direction will cause it to become violated. The smallest such distance in the set we denote β.

step 4. the new vertex is defined by:

$$\underline{\hat{x}} + min(\alpha, \beta)\underline{u}_s \tag{3.15}$$

that is, we move either as far as possible without violating any currently satisfied constraints or to the most distant intersection with a violated constraint, if that be closer.

If the new vertex is feasible the process terminates. Otherwise, go back to step 1 The new basis B is obtained by using the constraint corresponding to $min(\alpha, \beta)$ to replace the constraint which was dropped from the previous basis and the inverse of the new basis, B^{-1}, is computed from its previous inverse matrix by using the Simplex product formulae [Hadley, 1962, pp. 48].

If the new vertex is feasible the process has ended successfully.

3.7 Algorithm Implementation and Tests

3.7.1 Objectives

The objective of the work discussed here is to test the efficiency of the algorithmic approach described above [Fletcher, 1970a] for inspecting roundness and centre position tolerances for data acquired using conventional roundness measuring machine, as well as to check the result of inspection (in terms of pass/fail) in comparison to the exchange algorithms. This algorithm has been implemented in a customised form (from now on called the gauge algorithm) for two different reference figures, the ring gauge limaçon and the radial zone limaçon. In order to do the tests, the following sequence was obeyed:

- collect data from a set of testpieces using a roundness measuring machine interfaced with a computer by an analogue to digital converter;
- measure their out of roundness and reference centre position using the exchange algorithm for the ring and minimum zone limaçon cases. Estimate the efficiency of the algorithm;

- based on the previous results, define examples of lower and upper limits for the centre position and maximum out of roundness in order to simulate tolerance limits imposed by design;
- feed this tolerance information to the gauge algorithm to check whether the data were contained within tolerance and to measure its efficiency.

Details about data acquisition, algorithm implementations and test procedures are described in the following sections.

The implementation and tests described in the following sections only considered large data sets acquired using a spindle-based roundness measuring machine. Nevertheless, the same method, with some minor modifications, is valid for data acquired using a coordinate measuring machine (or other type of instrument, like a vision system, in which the data points are represented by a set of Cartesian coordinates), provided an origin shift is performed to conform with some conditions discussed in section 2.4.3. However, there remains the question of whether this method remains efficient for relatively small data sets, as is the case for a CMM. This question is addressed in section 3.8.

3.7.2 Algorithm Design and Implementation

This implementation considers only profiles obtained from conventional, spindle based, roundness measuring instruments in which the transducer measures only the radial variation between the surface of the workpiece and a nominal circle represented by a known point in the transducer (see chapter 2 for a brief description of this type of instrument). A fundamentally important feature is that information about the absolute radius of the part is lost, at least at the precision to which the profile is measured. However, by choosing a limaçon reference and making all measurements with respect to it radial from the origin, the analytical difficulties concerning radius suppression may be overcome. As the position of the co-ordinate system origin is unaffected by the radius suppression transformation, the parameter linearity ensures that the geometrical properties of a limaçon are preserved under radius suppression (see section 2.4.3).

In the algorithm proposed by Fletcher [1970a], an initial vertex is usually defined by the lower or upper variable bounds. For the variables corresponding to the centre position, the obvious lower and upper bounds are obtained by interpreting design specifications regarding centre position. However, in the absence of information about the workpiece nominal radius, there is no way of defining bounds around the nominal value for the radius. Consequently, here no specific tolerance band is imposed on the radius of a circular workpiece. Note that a slightly different approach might be taken for data (not radius suppressed) obtained from a coordinate measuring machine for example.

Still considering the same algorithm, it is suggested that an initial trial vertex be found from the variable lower or upper boundaries. However, the geometry of our problem allows a more suitable starting point to be defined by applying a few rules derived from the geometric configuration of the convex region in the parameter space.

In order to accelerate the iteration process, the initial vertex should be chosen as close as possible to the feasible region. Considering the ring limaçon case first, the parameter space is three dimensional, and each constraint is a plane which divides the space in two open half spaces, feasible and infeasible, and the boundaries of the feasible region are formed by the intersection of the equality conditions of all the constraints.

Considering the first set of constraints in (3.11), the equality condition of each constraint defines a plane intersecting the axes of a solution space corresponding to parameters a, b and R. The point of intersection on each axis for each plane will depend on the sign of the sine and cosine functions as well as on the sign of the profile radial data point r_i (considering radius suppressed data [Chetwynd, 1979a]).

Assuming r_i is positive, the points will always intersect the positive section of the R axis. For points in the data array whose angular ordinate is within the first quadrant of the instrument coordinate system, the cosine and sine functions are positive and therefore they define planes intersecting the positive section of the three axes of the parameter space; for points within the second quadrant, the planes will intersect the negative section of the a axis and the positive part of the b and R axes; for points within the third quadrant, the planes will intersect the negative section of the a and b axes and the positive section of the R axis; and for points in the fourth quadrant the planes will intersect the negative part of the b axis and the positive section of the other two axes. For points in the border between two quadrants, the plane will be parallel to the a axis if the point is between the first and the second or the third and the fourth quadrants, and it will intersect the positive section of the b axis if it is between the first and second quadrants. Similarly, the plane will be parallel to axis b if the point is between the second and the third or the fourth and the first quadrants and it will intersect the positive section of the a axis if it is between the fourth and the first quadrants.

When r_i is negative, the signs are inverted and the planes are shifted to the other sections of the axes, having their direction and slope shifted by 180^o, that is unaltered in practical terms. However, as the $\geq$ sign defines the upper half space as the feasible region for each constraint, the constraints with positive r_i will be the relevant ones in order to define the feasible region. In addition to this, and as a consequence of the geometric configuration of the planes defined by these constraints, the optimum point, that is the intersection of three such planes, will be in the quadrant which contains the plane of largest slope. This is derived from the geometric verification that, for the optimum point to be in a quadrant not the same as the one of the plane of largest slope, at least one of the other two planes would have to intersect the plane of assumed largest slope before it intersects the R axis, which would make the latter be the one of largest slope, and therefore in the same quadrant as the optimum point.

The second set of constraints in (3.11) will be shifted by t_r on the R axis, and the feasible region, because of the $\leq$ sign, will be an open convex set on the lower half space. The combination of constraints (3.11) may define a feasible region which, in this case, will be a polyhedron. Taking the variable R on the vertical axis, at least one feasible point (the optimum) will be defined on the quadrant which contains the plane of largest slope (largest r_i). Therefore, when a feasible region exists, the sign of the cosine and sine functions of the constraint with largest slope (and r_i positive) will define in which quadrant the optimum point will be. Based on this information, an initial vertex may be chosen on the same quadrant the optimum point is located, so as to reduce the total number of iterations to approach the feasible region from the initial vertex. This can be structured as the following set of rules:

1. select amongst the data points the one with the largest positive value. The index of this point in the data array will define the index of the (hyper-) plane with largest slope;

2. if this point lies in the first quadrant of the data points array or in the border between the first and the second quadrants, then the planes defined by the upper bounds of variables a and b are used to define the initial vertex. Else, if the point lies in the second quadrant or in the border between the second and the third quadrants, the planes defined by the upper bound of variable b and the lower bound of variable a are used instead. Else, if the point lies in the third quadrant or in the border between the third and the fourth quadrants, the planes defined by the lower bounds of variables a and b are used instead. Finally, if the point lies in the fourth quadrant or in the border between the fourth and the first quadrants , the planes defined by the upper bound of variable a and lower bound of variable b are used to define the initial vertex;

3. define the initial vertex as the intersection of the plane whose index was defined in 1, and the planes corresponding to the lower or upper variable boundaries specified in 2.

For the radial zone limaçon case, although it is four-dimensional parameter space, rules 1 and 2 apply in the same way, that is if the angular ordinate of the point of largest positive radial ordinate, r_i, is within the first quadrant for instance, it will define a hyper-plane crossing the positive sections of axes a and b and therefore the hyper-planes defined by the upper bounds of variables a and

b will be used to define the initial vertex, and so forth. The difference is that in step 3, the initial vertex is defined by the intersection of the plane whose index was defined in step 1, the planes corresponding to the lower or upper variable boundaries specified in step 2 and the plane defined by the upper boundary of the zone variable h, that is half of the roundness tolerance zone. The plane defined by the upper boundary of h is chosen in this case because it is more likely that the vertex so defined is close to the feasible region than the one defined by using the plane corresponding to the lower boundary to h.

These rules will bring the initial vertex close to the feasible region and form the starting point for the implementation of the Fletcher's algorithm described in section 3.6. The routines were written in C language, using single precision floating point. The main function is reproduced in appendix B.

3.7.3 Data Acquisition

A set of 10 profiles were data-logged, and stored in disc files, from nominally circular components which were readily to hand. The parts included ground and turned shafts, in carbon steel and aluminium, with radii between 15 and 40 mm. Measurements were made on a Rank Taylor Hobson Talyrond 200 instrument under typical operating conditions, that is:

- standard hatchet stylus with hatchet radius of 6.3 mm, hatchet width of 1.6 mm and tip transverse radius of 0.38 to 0.51 mm;
- signal magnification of 1000;
- eccentricity of the component within the limits of the instrument chart.

The profile was taken from the instrument amplifier output as a signal of $+-1$ volt, after amplification, representing a displacement of 25.4 mm. This was passed to a standard MINC 12-bit successive approximation analogue to digital converter and then to MINC 11 computer. Although the MINC technology is obsolete it provided a well characterised logging system used solely for data collection. Analysis of the data was performed off-line.

The analogue to digital conversion was controlled by a MINC Basic function. It was programmed to collect 513 data points, with a time interval of $T = 10/512$ s, which corresponds to one complete revolution of the testpiece. The data collection process was triggered by a signal from the Talyrond onto a Schmitt trigger provided by a MINC Clock module.

The data points were placed in a ASCII file and sent to a SUN computer workstation where the tests were actually performed. The data files all consisted of a sequence of 513 sixteen bit words, each containing one twelve bit profile deviation from a reference circle. The angular position of each point is implicitly given as $2\pi i/512$ where i is the point index in the data array. The data sets are listed in appendix A.1. Linear plots of the profiles are reproduced in figures 3.6 to 3.15, and their roundness errors given in tables 3.2 and 3.3.

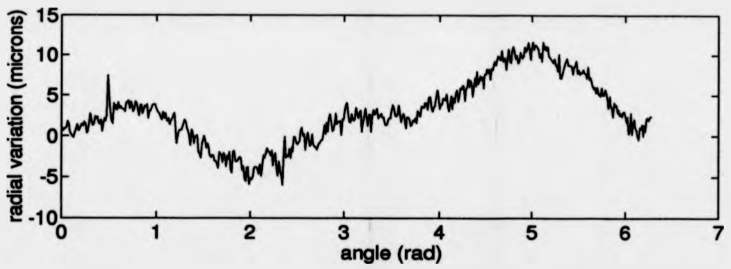


Figure 3.6: Linear plot of circular profile (top/bottom: air/metal): Data set 1.

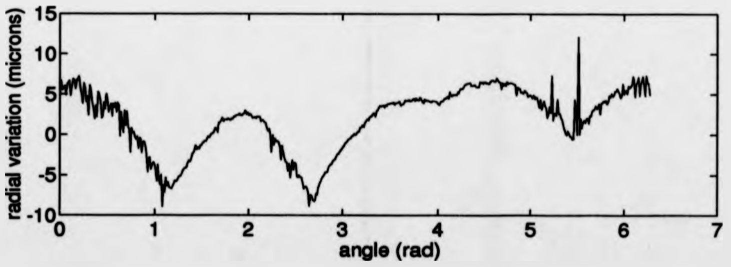


Figure 3.7: Linear plot of circular profile (top/bottom: air/metal): Data set 2.

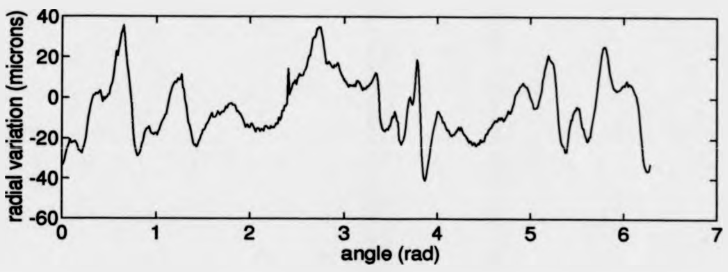


Figure 3.8: Linear plot of circular profile (top/bottom: air/metal): Data set 3.

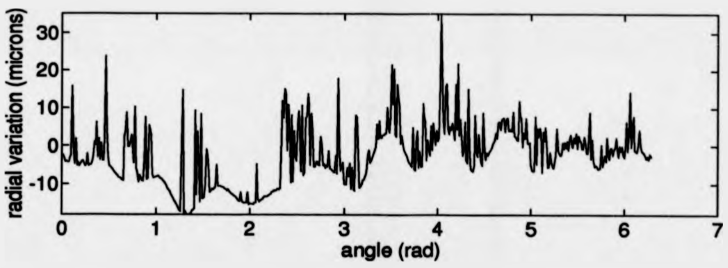


Figure 3.9: Linear plot of circular profile (top/bottom: air/metal): Data set 4.

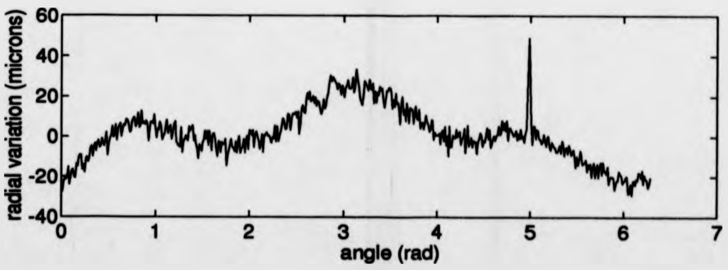


Figure 3.10: Linear plot of circular profile (top/bottom: air/metal): Data set 5.

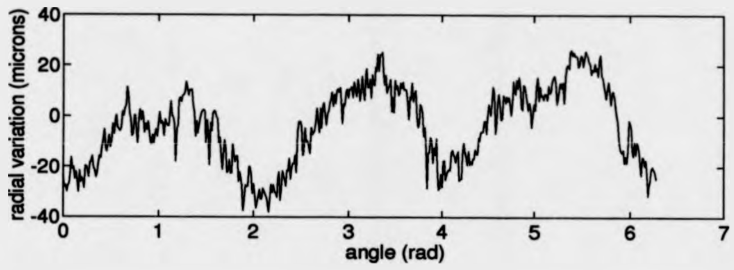


Figure 3.11: Linear plot of circular profile (top/bottom: air/metal): Data set 6.

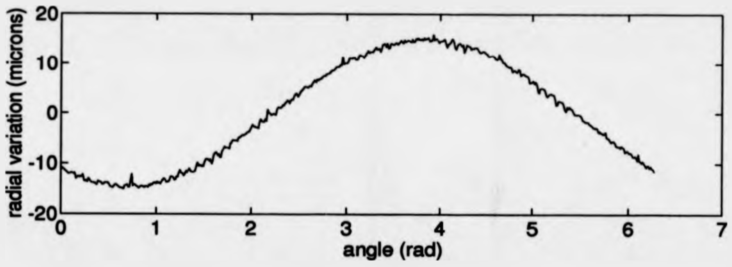


Figure 3.12: Linear plot of circular profile (top/bottom: air/metal): Data set 7.

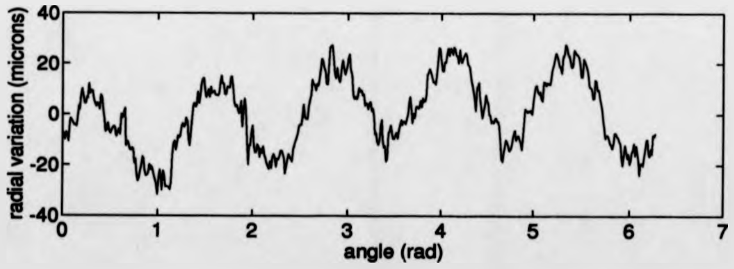


Figure 3.13: Linear plot of circular profile (top/bottom: air/metal): Data set 8.

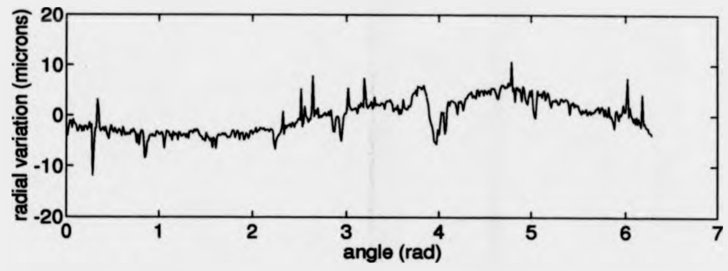


Figure 3.14: Linear plot of circular profile (top/bottom: air/metal): Data set 9.

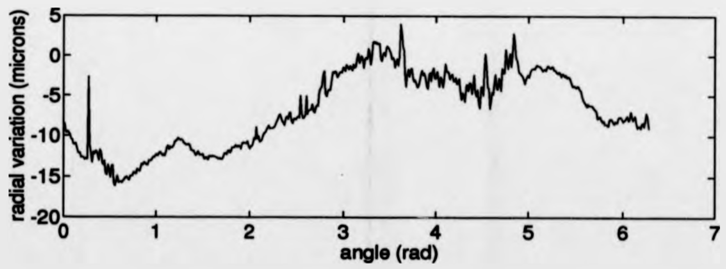


Figure 3.15: Linear plot of circular profile (top/bottom: air/metal): Data set 10.

Data Set	Ring Limaçon			
	Cent Posit		Out of Roundness	Num Iterat.
	a	b		
1	2.54	-4.03	10.53	2
2	4.4	-2.67	15.74	3
3	-.73	9.38	65.83	3
4	3.42	-13.35	47.4	2
5	-5.24	-19.99	60.27	2
6	-1.87	-7.97	57.83	3
7	-11.03	-9.04	3.24	2
8	-6.43	-9.12	51.84	4
9	-1.35	-2.82	17.71	2
10	-.63	-7.37	13.98	4

Table 3.2: Centre position and out of roundness (in microns (μm)) of ring limaçon using the exchange algorithm

3.7.4 Test Procedures and Results

The ring and minimum zone exchange algorithms were implemented as a directed version of the algorithms designed by Chetwynd [1985]. The routines were written in C language, single precision floating point. The main routines are reproduced in appendix B. Tables 3.2 and 3.3 present the centre position of the ring and minimum zone limaçons for each data set, the error of roundness according to these references and the number of iterations to get the references.

In order to test the efficiency of the gauge algorithm to get to a pass/fail result of inspection, simulated tolerance values for the out of roundness and centre position were fed to the gauge algorithm. For each data set and for each fitting, examples of tolerance values for the out of roundness were defined as fractions over and under the measured error of roundness. The centre position tolerance was defined as a maximum deviation from the origin of the instrument coordinate system, assumed for simplicity as the nominal centre position.

Therefore, for each data set, two major sets of tests were performed: first, using the implementation of the gauge algorithm based on the ring limaçon and feeding to it simulated tolerance values based on the results obtained by running the ring gauge exchange algorithm, as indicated in table 3.2; second, following the

Data Set	Minimum Zone Limaçon			
	Cent Posit		Out of Roundness	Num Iterat
	a	b		
1	1.46	-3.57	10.15	4
2	1.35	-4.72	13.23	4
3	-.70	9.46	65.74	8
4	-2.06	-6.59	42.25	6
5	-5.16	-19.92	60.22	6
6	-1.2	-5.55	56.79	5
7	-10.96	-9.3	3.10	4
8	-4.01	-5.82	49.73	4
9	-1.63	-3.53	17.33	4
10	-1.45	-5.31	12.27	4

Table 3.3: Centre position and out of roundness (in microns (μm)) of minimum zone limaçon using the exchange algorithm

same procedure but this time using the implementation of the gauge algorithm based on the zone limaçon reference and calculating tolerance values based on the results obtained by running the minimum zone exchange algorithm, again indicated in table 3.3.

For each data set, a number of examples of tolerance values were defined so as as to consider a range of likely practical situations. Different possible situations were tested, as follows:

- the centre position tolerance and the roundness tolerance are both less than the actual errors (region C in figure 3.16a or b);
- the centre position tolerance is less than the actual error and the roundness tolerance is equal to or greater than the actual out of roundness (region A);
- the centre position tolerance is equal to or greater than the eccentricity error and the roundness tolerance is less than the out of roundness error (region D);
- the centre position tolerance is equal to or greater than the eccentricity error and the roundness tolerance is equal to or greater than actual out of roundness (region B).

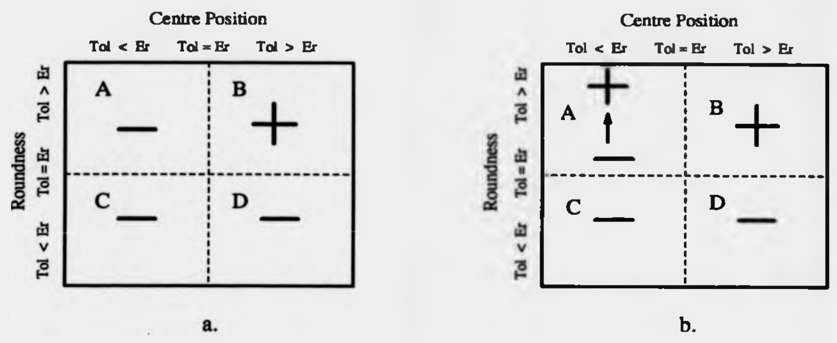


Figure 3.16: Result of inspection (+/-: pass/fail) based on (a) exchange algorithm methods and (b) the gauge algorithm.

Thus, taking Z as the out of roundness, values of $0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3$ times Z were used as roundness tolerances. In the case of the centre position tolerance, a square zone equally spaced around the instrument coordinate system origin was defined. Values of $0.7, 0.3, 0.9, 1.0, 1.1, 1.2, 1.3$ times the maximum reference centre coordinate, $max(a, b)$, were used to define the perpendicular distance from the origin to the upper and lower bounds.

Overall, for each testpiece, and for each reference figure (either ring or zone limaçon), a set of 49 different tolerance zone combinations were fed to the algorithm, which in each case returned with a pass/fail flag.

The routines were run under Unix, on a SUN 4/330, with 48 Mbytes of RAM (Random Access Memory) and speed of 16 MIPS (Million of Instructions Per Second). The efficiency of the algorithms was measured in terms of number of iterations, arithmetic operations and computation time. The execution time was estimated by running the programme under the "time" Unix command, which returns the CPU user time.

The number of iterations of the gauge algorithm was tabulated as shown in tables 3.5 to 3.24 for the ring and zone limaçon references, for each data set. The result of inspection was signaled by appending a minus sign to the number of iterations if it was a fail. Figures 3.17 and 3.18 show graphically the average number of iterations taken, over the whole set of test profiles, for the ring and zone limaçon cases respectively.

The general pattern of results is consistent over all data sets tested and there is no significant increase in the number of iterations for the zone limaçon case over the ring limaçon case, although this formulation generates a slightly larger problem. The number of iterations is generally small but it takes a little longer to evaluate cases of easy acceptance (tolerance greater than error by a large amount) than easy rejection. For data set seven, the number of iterations is distinctly high through the bottom half of the table, that is in cases of acceptance, or rejection because of roundness error exceeding its tolerance zone. This data set has the least roundness error (tables 3.2 and 3.3), and over three times less than the error of any other profile, which means that the examples of tolerance values used for testing this data set were of the same order. Considering that the "size" of the feasible region is affected by the magnitude of the tolerance zone, this may explain why it takes a relatively large number of iterations, although this relation is not clear for the other data sets. Comparing figures 3.9 and 3.12 with tables 3.12 and 3.18, there is no support for claiming that the iterative procedure is sensitive to "spiky" profiles; the same can be said about "rough" profiles, by comparing figures 3.8 and 3.6, 3.10 or 3.11 with tables 3.10 and 3.6, 3.14 or 3.16. Also, in general, the number of iterations rises when the tolerance is close to the actual error. As this condition is approached, the feasible region effectively shrinks towards a point and at the limit the algorithm is required to make an equality decision rather than one based on an inequality. Typically, a floating point comparison will be uncertain at the least significant digit and a further iteration to the next vertex confirms whether the equality comparison failed for this reason. Thus this condition commonly requires one or two extra iterations to establish which side of the boundary the solution lies. Note that this is not the same effect as found in some non-linear optimisation methods (e.g. the Sequential Quadratic Programming method) where an infeasible problem iterates forever because there always remains an infinity of points not searched.

Table 3.4 presents the number of arithmetic operations and the computation time for the gauge and exchange algorithms for typical data sets. The gauge algorithm takes on average approximately $17N$ arithmetic operations per iteration

data set	fit (mcc/mzc)	number iterat.		arith. operations		comp. time (sec)	
		gauge	exchange	gauge	exchange	gauge	exchange
1	mcc	2	2	17206	10576	0.8	0.5
2	mcc	1	3	9751	13167	0.6	0.5
3	mzc	1	8	10933	33192	0.7	0.6
6	mzc	2	5	19741	25362	0.7	0.6
7	mcc	5	2	42191	10576	1.4	0.5
10	mzc	1	4	10889	22752	0.6	0.5

Table 3.4: Number of arithmetic operations and computation time for the exchange and gauge algorithms, for some data sets.

to get to a solution, while the exchange algorithms take approximately $10N$, where N is the number of data points. The computation time represents the user time spent by the CPU to run the programme, accurate to within about one tenth of a second. However, due to variable time-share overheads, these times are more a comparative than an absolute measure of performance. Thus, it is reasonable to assume that a present generation computer dedicated to a measuring machine will perform these computations faster than indicated above. Therefore, although the exchange algorithms are moderately faster than the gauge algorithms, the difference is not very significant as the computation time for both is well within the range normally expected of a computer-aided instrument suitable for on-line applications.

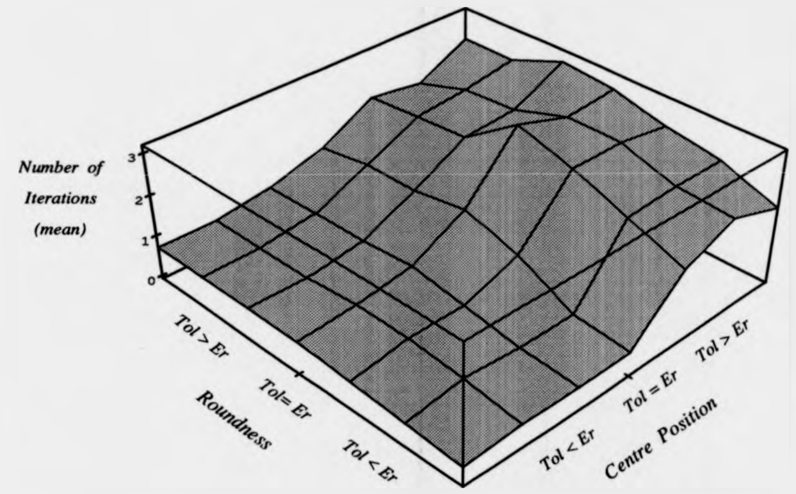


Figure 3.17: Average number of iterations of the gauge algorithm for the ring limaçon over the whole set of profiles.

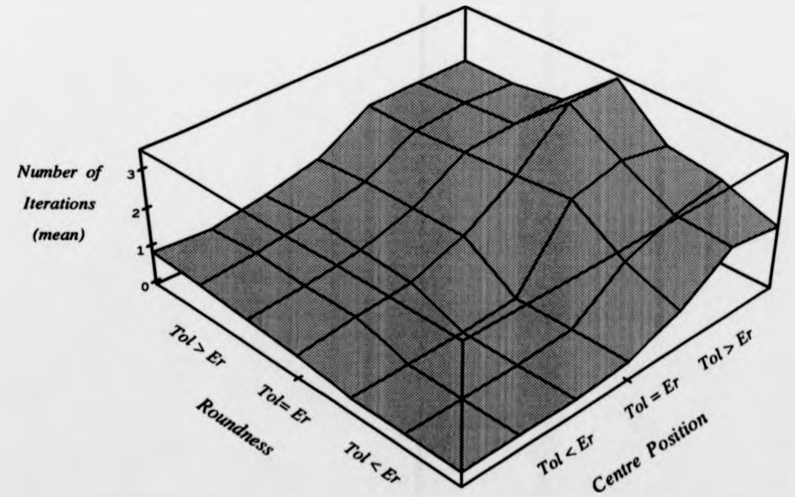


Figure 3.18: Average number of iterations of the gauge algorithm for the zone limaçon over the whole set of profiles.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-1	-1	-2	-0	0	0	0
0.8	-1	-1	-1	-1	0	0	0
0.9	-1	-1	-1	-1	0	0	0
1.0	-1	-1	-4	2	1	1	1
1.1	-1	-2	-3	3	1	1	1
1.2	-2	-3	-5	2	2	2	2
1.3	-2	-5	-4	2	2	2	2

Table 3.5: Ring limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 1.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-0	-0	-0	-0	-0	-0
0.8	-0	-0	-0	-0	-0	-0	1
0.9	-0	-0	-0	-0	-0	0	0
1.0	-1	-1	-0	1	1	4	1
1.1	-1	-1	-1	2	1	1	1
1.2	-2	-2	-2	2	1	1	1
1.3	-2	-2	-2	2	1	1	1

Table 3.6: Zone limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 1.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-0	-0	-0	-0	0	0
0.8	-0	-0	-0	-0	0	0	0
0.9	-0	-0	-0	-0	0	0	0
1.0	-0	-0	-1	1	1	1	1
1.1	-2	-1	1	1	1	1	1
1.2	-1	-2	1	1	1	1	1
1.3	-1	-2	1	1	1	1	1

Table 3.7: Ring limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 2.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-0	-0	-0	-0	1	1
0.8	-0	-0	-0	-0	-0	1	1
0.9	-0	-0	-0	-0	-0	1	1
1.0	-1	-1	-0	0	1	1	1
1.1	-1	-1	-1	2	1	1	1
1.2	-1	-1	-1	2	1	1	1
1.3	-1	-1	-1	2	1	1	1

Table 3.8: Zone limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 2.

Roundness (tolerance /error) / centre position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-0	-0	-0	-2	1	1	1
0.8	-0	-0	-0	-1	1	1	1
0.9	-0	-0	-0	-1	1	1	1
1.0	-0	-0	-0	2	2	2	2
1.1	-0	-0	-3	3	1	1	1
1.2	-1	-1	-2	3	1	1	1
1.3	-1	-1	-1	3	1	1	1

Table 3.9: Ring limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 3.

Roundness (tolerance /error) / centre position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-0	-0	-0	-0	1	1	1
0.8	-0	-0	-0	-0	1	1	1
0.9	-0	-0	-0	-0	1	1	1
1.0	-0	-0	-0	3	3	3	3
1.1	-0	-0	-3	1	1	1	1
1.2	-1	-1	-4	4	1	1	1
1.3	-1	-1	-1	4	1	1	1

Table 3.10: Zone limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 3.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-0	0	1	1	1	1
0.8	-0	-2	1	1	1	1	1
0.9	-0	-1	1	1	1	1	1
1.0	-0	-1	1	1	1	1	1
1.1	-1	-2	1	1	1	1	1
1.2	-1	-2	1	1	1	1	1
1.3	-2	-2	1	1	1	1	1

Table 3.11: Ring limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 4.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-0	-0	-0	0	0	0
0.8	-0	-0	-0	-1	0	0	0
0.9	-0	-0	-0	-1	1	1	1
1.0	-0	-0	-0	1	1	1	1
1.1	-0	-0	-1	1	1	1	1
1.2	-0	-0	-0	2	1	1	1
1.3	-0	-0	-0	2	1	1	1

Table 3.12: Zone limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 4.

Roundness (tolerance /error) / centre position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-1	-1	-1	-1	1	1	1
0.8	-1	-1	-1	-1	1	1	1
0.9	-1	-1	-1	-1	1	1	1
1.0	-1	-1	-1	3	2	2	1
1.1	-2	-3	-2	4	2	2	1
1.2	-2	-2	-2	4	2	2	1
1.3	-2	-2	-2	4	2	2	1

Table 3.13: Ring limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 5.

Roundness (tolerance /error) / centre position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-1	-1	-1	-1	-1	1	1
0.8	-1	-1	-1	-1	-1	1	1
0.9	-1	-1	-1	-1	1	1	1
1.0	-1	-1	-1	1	1	1	1
1.1	-1	-2	-2	4	2	2	1
1.2	-2	-2	-2	4	2	2	1
1.3	-4	-2	-2	4	2	2	1

Table 3.14: Zone limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 5.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-0	-1	-2	2	2	2
0.8	-0	-0	-1	-2	2	2	2
0.9	-0	-0	-2	-3	3	3	3
1.0	-0	-0	-2	3	3	3	3
1.1	-0	-2	-2	3	3	3	3
1.2	-0	-1	-2	3	3	3	3
1.3	-0	-2	-2	4	4	4	4

Table 3.15: Ring limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 6.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-0	-0	-2	2	2	2
0.8	-0	-0	-0	-2	2	2	2
0.9	-0	-0	-0	-2	2	2	2
1.0	-0	-0	-0	2	2	2	2
1.1	-0	-0	-0	4	2	2	2
1.2	-0	-0	-1	5	2	2	2
1.3	-0	-0	-1	5	2	2	2

Table 3.16: Zone limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 6.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-0	-0	-0	-0	-0	-0
0.8	-0	-0	-0	-0	-0	-0	-0
0.9	-0	-0	-0	-0	-0	-0	-0
1.0	-0	-0	-1	1	1	1	1
1.1	-7	-6	-8	8	7	7	7
1.2	-10	-7	-7	6	7	7	4
1.3	-6	-6	-7	7	11	7	9

Table 3.17: Ring limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative),dat set 7.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-0	-0	-0	-0	-0	-0
0.8	-0	-0	-0	-0	-0	-0	-0
0.9	-0	-0	-0	-0	-0	-0	-0
1.0	-0	-1	-1	1	1	1	1
1.1	-4	-7	-7	9	9	8	8
1.2	-10	-8	-8	8	9	8	8
1.3	-8	-10	-8	8	9	7	7

Table 3.18: Zone limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 7.

Roundness (tolerance /error) / centre position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-1	-2	-1	-1	1	1	1
0.8	-2	-2	-2	-1	1	1	1
0.9	-2	-2	-2	-2	2	2	2
1.0	-2	-2	-2	1	1	1	1
1.1	-2	-3	-2	2	2	2	2
1.2	-3	-2	-2	2	2	2	2
1.3	-2	-2	-2	2	3	3	3

Table 3.19: Ring limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 8.

Roundness (tolerance/error) / centre position (tolerance/error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-1	-1	-1	-1	1	1	1
0.8	-1	-1	-1	-1	1	1	1
0.9	-1	-1	-1	-1	1	1	1
1.0	-1	-1	-1	1	1	1	1
1.1	-1	-1	-2	2	2	2	2
1.2	-1	-2	-2	2	2	2	2
1.3	-1	-2	-2	2	2	2	2

Table 3.20: Zone limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 8.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-1	-1	-1	-1	1	1	1
0.8	-1	-1	-1	-1	1	1	1
0.9	-1	-1	-1	-1	1	1	1
1.0	-1	-1	-1	-2	2	2	1
1.1	-1	-2	-3	3	2	2	2
1.2	-1	-2	-3	3	2	2	2
1.3	-1	-2	-3	3	2	2	2

Table 3.21: Ring limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 9.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-1	-1	-1	-1	-1	1	1
0.8	-1	-1	-1	-2	2	2	1
0.9	-1	-1	-1	-3	2	2	1
1.0	-1	-1	-1	3	2	2	1
1.1	-1	-2	-5	3	2	2	2
1.2	-1	-2	-5	3	2	2	2
1.3	-1	-2	-5	3	2	2	2

Table 3.22: Zone limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 9.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-0	-0	-0	0	0	0
0.8	-0	-0	-0	0	0	0	0
0.9	-0	-0	0	0	0	0	0
1.0	-0	-0	1	1	1	1	1
1.1	-1	-1	3	2	2	1	1
1.2	-1	-1	3	3	2	1	1
1.3	-1	-1	3	3	2	1	1

Table 3.23: Ring limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 10.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-0	-0	-0	-0	0	0
0.8	-0	-0	-0	-0	-0	0	0
0.9	-0	-0	-0	-0	0	0	0
1.0	-0	-0	-0	0	0	0	0
1.1	-0	-0	-1	2	1	1	1
1.2	-0	-1	-1	2	1	1	1
1.3	-1	-1	-1	2	1	1	1

Table 3.24: Zone limaçon search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set 10.

3.8 Gauge Algorithm Applied to CMM Data

3.8.1 Objectives

It has been stressed that the limaçon approximation can be applied to data from a coordinate measuring machine (or Cartesian data from a vision machine) as long as the data set is re-expressed relative to an origin that lies not far from the best fit centre. Providing the ratio of the centre eccentricity to the radius is kept below about 0.01, the radial variation between the limaçon and the circle will be at most $5.0\text{x}10^{-5}$ times the reference radius (see section 2.4.3). So, for a radius of say 50 mm, this represent an error of at most 2.5μm, which is in most cases less than the repeatibility of measurement of the machine.

Whenever the linear approximation is a valid resource, the gauge algorithm can be equally applied to the data. Thus, some tests were carried out in order to check experimentally the efficiency of the gauge algorithm for inspecting roundness and centre position tolerance for data acquired using a coordinate measuring machine (CMM) type of instrument as opposed to data acquired using an specialised roundness instrument.

Therefore, the following procedure was obeyed:

- collect data from a set of testpieces using a coordinate measuring machine;
- compute the centroid (arithmetic mean) of the data points, given in Cartesian coordinates;
- define a new Cartesian coordinate system with its origin at the centroid of the points. Re-express the data points in this new coordinate system;
- transform the data points from Cartesian to polar coordinates;
- measure their out of roundness and reference centre position using the minimum zone exchange algorithm;
- define examples of tolerance values and feed this information to the gauge algorithm to check whether the data are contained within tolerance and to

measure its efficiency.

Details about these steps are given in the next sections.

3.8.2 Data Acquisition

A set of 5 profiles were data-logged, and stored in disc files, from nominally circular features of non-circular components which were readily to hand. The parts included bored, milled and drilled holes in aluminium, with radii between 5 and 14 mm.

Measurements were made on a computer controlled coordinate measuring machine, *LK Micro Four*, cantilever type of structure, with a measuring travel of 600, 280 and 450 mm on the x, y and z axes respectively. The specified resolution of the Inductosyn reading system was of 2μm, and the accuracy of measurement claimed to be $\pm 5\mu$m on x-, y- and z-axes. The machine was fitted with a touch trigger probe and operated under manual mode. A set-up procedure was executed in order to identify the probe and to establish a coordinate system local to the components being measured. Following this procedure, for each profile, a set of 30 data points evenly, but not exactly equal, spaced around the circumference were sampled. The decision on the number of data points for each profile was based on the argument that the minimum number of points recommended by standards(that is seven, to detect up to six lobes [e. g. BSI, 1989]) leads to higher uncertainty about the location of the reference and the radial separation than does larger data sets [Odayappan et al, 1992]. In addition to this, here the strategy was not first to define a reference and then measure the deviation of sampled data points from that, as it would normally be the case.

The data points were placed in a ASCII file and sent to a SUN computer workstation where the tests were actually performed. The data files all consisted of a two-dimensional array containing the x-y-cordinates of the data points in relation to a local coordinate system. Figures 3.19 to 3.23 present a linear plot of each profile expressed in polar coordinates and shifted to the centroid of the points. For completeness, the actual data sets are reproduced in appendix A.2.

data	least squares parameters (mm)			round. error	$\lambda = \frac{E}{R}$	
set	a	b	R	(mm)	lstsq	centroid
11	52.946	60.375	13.692	0.031	0.0008	0.047
12	152.120	61.680	13.518	0.244	0.0025	0.024
13	92.604	62.628	5.000	0.100	0.0007	0.039
14	67.846	21.093	8.813	0.021	0.0011	0.0066
15	76.605	41.796	6.485	0.112	0.0033	0.051

Table 3.25: Circle parameters of the least squares fit and respective out of roundness and eccentricity ratios for least squares (lstsq) and centroid centres.

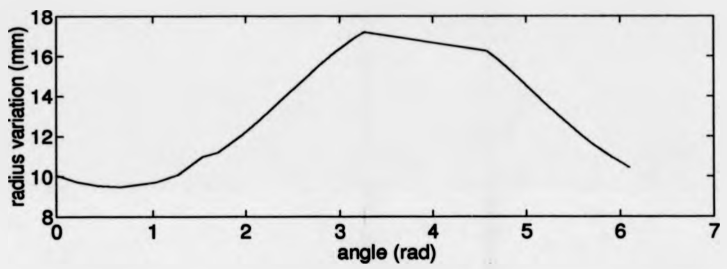


Figure 3.19: Linear plot of circular profile (top/bottom: air/metal): Data set 11.

Table 3.25 presents the parameters of the least-square best fit and out of roundness for each file. The least squares parameters were computed by implementing the algorithm suggested by Forbes [1989] (see section 2.5.3 for details about this algorithm). It was implemented in C language, single precision floating point. The main fragment of the source code is reproduced in appendix B. Table 3.25 also shows for each file the eccentricity ratios based on the eccentricity of the minimum zone circle reference from the least squares centre parameters and from the centroid of the points, each in turn assumed as the origin of the system. The minimum zone circle reference was computed by using the Sequential Quadratic Programming method, implemented by NAG (E04VDF [NAG, 1990]) in order to minimise the zone parameter h subject to the non-linear constraints of 5.11 (as discussed in section 5.5.5).

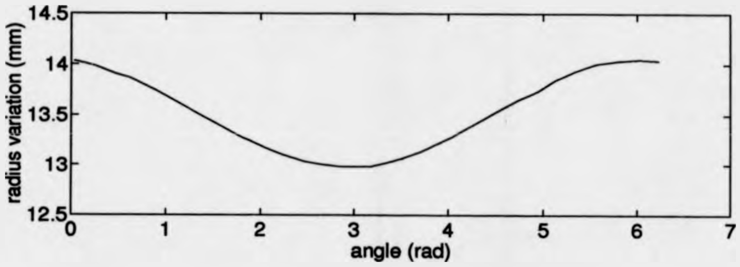


Figure 3.20: Linear plot of circular profile (top/bottom: air/metal): Data set 12.

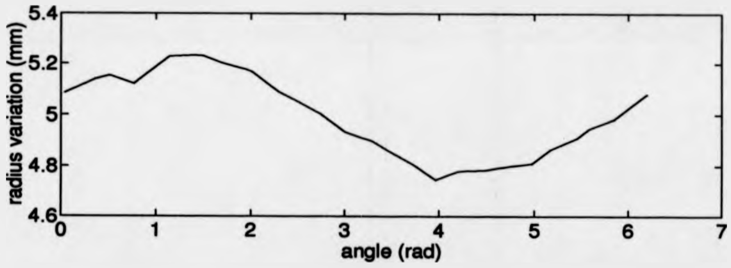


Figure 3.21: Linear plot of circular profile (top/bottom: air/metal): Data set 13.

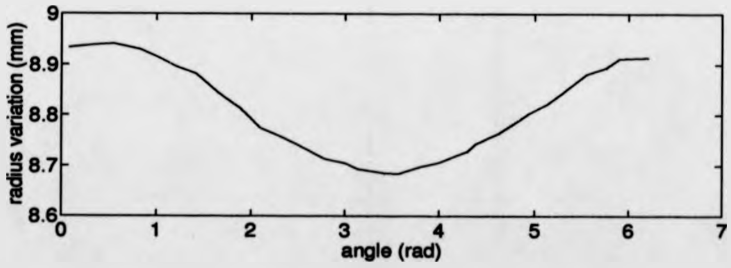


Figure 3.22: Linear plot of circular profile (top/bottom: air/metal): Data set 14.

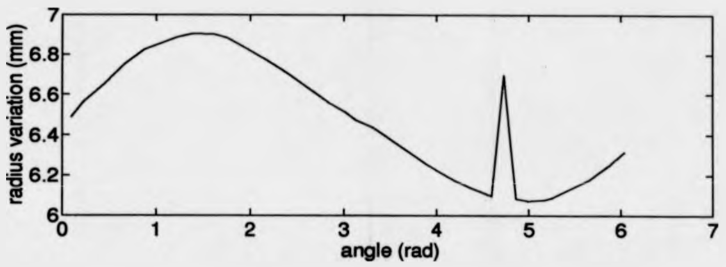


Figure 3.23: Linear plot of circular profile (top/bottom: air/metal): Data set 15.

3.8.3 Test Procedures and Results

The gauge algorithm as implemented in section 3.7.2, using the ring limaçon type of formulation (defined by the set constraints in 3.2 and 3.6) was used for this set of tests. In this case, tolerance on radius might be defined as well, as information on radius is present (differently of the radius suppressed data of spindle based roundness instruments). This represents a small change of the original algorithm, so as to include lower and upper boundaries on the radius parameter. Although it was possible to inspect tolerance on radius, the tests performed here were limited to inspect roundness and centre position, as it simplifies the matrix of results without affecting the efficiency of the gauge algorithm.

The decision on using the centroid as the origin of the local system was also made on the grounds that it simplifies computation without affecting the efficiency of the gauge algorithm. The consequences of such an approximation are discussed in the next section.

After these transformations the data sets were ready for feasibility checking by the gauge algorithm. Simulated tolerance values were calculated based on the out of roundness and centre position of the minimum zone limaçon reference, defined by running the exchange algorithm [Chetwynd, 1985], implemented as described in section 3.7.4. In this case, only the minimum zone reference was used. Examples of tolerance zones for the out of roundness and centre position were defined as described in section 3.7.4. Therefore, for each profile, a set of 49 different tolerance zone combinations were fed to the gauge algorithm, which

data	a	b	R	Z
set	(mm)	(mm)	(mm)	(mm)
11	-3.236	-2.536	13.32	0.57
12	0.0654	-0.0987	13.506	0.191
13	0.184	0.2510	4.982	0.0912
14	0.0350	0.0450	8.810	0.02170
15	0.0718	0.3213	6.4729	0.1030

Table 3.26: Minimum zone limaçon centre position (a, b) (given from the centroid of the points), radius (R) and out of roundness (Z), units in millimeters (mm).

returned with a pass/fail inspection flag.

The tests were run under the same conditions as before, that is, under Unix, on a SUN 4/330, with 48 Mbytes of RAM and speed of 16 MIPS (Million of Instructions Per Second). The efficiency of the gauge algorithm was measured in terms of number of iterations, arithmetic operations and computation time. The computation time was estimated by running the programme under the "time" Unix command, which returns the CPU user time.

The number of iterations of the gauge algorithm was tabulated as shown in tables 3.28 to 3.32, for each data set. The average number of iterations taken over the whole set of profiles is shown graphically in figure 3.24. Table 3.27 shows the number of arithmetic operations and the computation time for some examples of tolerance values for each file.

As in the previous case, the number of iterations is generally small but it takes longer to evaluate cases of easy acceptance than easy rejection; also the number of iterations rises when the tolerance is close to the actual error, for the same reason as discussed before.

A comparison of figures 3.17 or 3.18 and figure 3.24 shows that, although in this case the number of constraints has been drastically reduced, the number of iterations has not been affected by this change. However, the computation time and the number of arithmetic operations per iterations are reduced and therefore the total computation time is sensibly reduced, as shown in table 3.27. An interesting conclusion regarding the number of constraints is that the use of

data set	number iterations	arithmetic operations	computation time (sec)
11	1	727	0.1
12	1	628	0.1
13	4	1958	0.3
14	3	1533	0.2
15	2	1018	0.1

Table 3.27: Number of arithmetic operations and computation time for the gauge algorithms, for some examples of tolerance values.

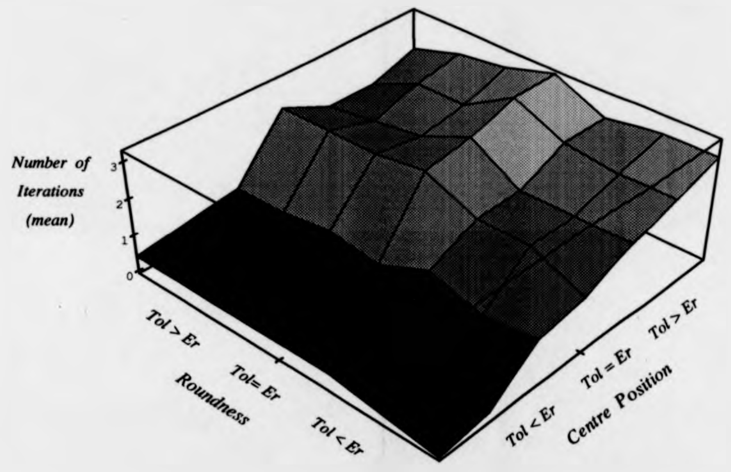


Figure 3.24: Average number of iterations of the gauge algorithm for CMM data over the whole set of profiles.

this method for relatively large data sets for CMM does improve the correctness of the inspection process without bringing any significant additional cost in terms of computation time.

Roundness (tolerance /error) / centre position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-0	-0	-0	-0	-0	-0	-0
0.8	-0	-0	-0	-0	-0	-0	0
0.9	-0	-1	-1	-1	-1	0	0
1.0	-0	-0	-0	-0	2	2	2
1.1	-1	-1	-1	1	1	1	1
1.2	-3	-3	-3	3	1	1	1
1.3	-4	-4	-3	3	3	2	2

Table 3.28: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative),data set 11.

Roundness (tolerance /error) / centre position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-0	-0	-1	1	1	1	1
0.8	-1	-1	-1	1	1	1	1
0.9	-1	-1	-1	1	1	1	1
1.0	-2	-2	-2	2	1	1	1
1.1	-2	-2	-2	2	2	2	1
1.2	-2	-2	-2	2	2	2	1
1.3	-2	-2	-2	2	2	2	2

Table 3.29: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative),data set 12.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-1	-1	-1	-1	-1	-1
0.8	-0	-1	-1	-1	-1	-1	-1
0.9	-2	-1	-1	1	1	1	1
1.0	-2	-3	-3	3	3	3	3
1.1	-3	-3	-3	4	3	2	2
1.2	-3	-3	-3	4	4	4	2
1.3	-3	-3	-3	4	4	4	2

Table 3.30: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative),data set 13.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-0	-0	-0	-0	-0	0
0.8	-0	-0	-0	-0	-0	1	1
0.9	-3	-3	-4	-1	1	1	1
1.0	-2	-4	-3	5	4	4	4
1.1	-3	-3	-3	4	3	3	3
1.2	-3	-3	-3	4	3	3	3
1.3	-4	-4	-3	4	3	3	3

Table 3.31: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative),data set 14.

Roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
centre position (tolerance /error)							
0.7	-0	-0	-0	-0	-0	-0	-0
0.8	-0	-0	-0	-0	-0	-0	0
0.9	-0	-0	-0	-0	-1	1	1
1.0	-1	-1	-2	3	2	2	2
1.1	-1	-1	-2	3	2	2	2
1.2	-1	-1	-2	3	2	2	2
1.3	-1	-1	-2	3	2	2	2

Table 3.32: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative),data set 15.

3.9 Discussion

Figures 3.16a and 3.16b present a graphical illustration of the pass/fail regions according to the exchange and gauge algorithms, where the plus/minus signs indicate pass/fail inspection. In order to check whether the roundness error and centre position are satisfactory by directly using the ring and minimum zone references, tolerance values for both would have to be compared with the reference parameters. Note however that if the inspection fails because the reference centre position eccentricity is larger than the tolerance, there is no routine means to evaluate whether it is possible to define a new reference with its centre and out of roundness within tolerance. This can be seen in figure 3.16a, where the pass status to an inspection procedure using the exchange algorithms is confined to region B only. This reveals a major disadvantage of the exchange algorithms if it is attempted to use them in the inspection of form and position of geometric features.

The formulation of the inspection problem as an extended linear programming problem with a test for feasibility overcomes this limitation. Referring to figure 3.16b, when the eccentricity of the best-fit reference (defined by the exchange algorithm) exceeds the design tolerance, it is possible to find a new reference such that the out of roundness as well as the reference centre position are within tolerance provided that the out of roundness is sufficiently less than the allowed out of roundness, therefore avoiding good components been rejected. This is illustrated by the change in sign in region A, figure 3.16b and confirmed by the results of inspection tabulated in tables 3.5 to 3.24.

The comparison between the roundness error obtained using the ring and the minimum zone limaçon (tables 3.2 and 3.3) shows that, for for the data sets numbered 2, 4 and 10, the error defined is considerably less when using the minimum zone reference. As a consequence, the results of inspection for these data sets, for the ring limaçon reference, tables 3.7, 3.11 and 3.23 respectively, revealed that although the roundness tolerance value is less than the actual error measured using the ring best-fit reference (tolerance/error = 0.9), it is still possible to enclose the profile within the tolerance limits. Therefore, this confirms the general argument that the the ring limaçon best-fit criterion does not define the least deviation.

An eight or sixteen sided polygon, as mentioned before, could be used to define the centre position or concentricity tolerance zone, either inscribing or averaging the tolerance zone. This would give a good approximation to the circular tolerance zone and it is not likely to affect significantly the computational efficiency of the algorithm.

It is clear from the results of inspection that there is no difference in formulating the problem by using the set of constraints in (3.2) and (3.6), that is using a ring limaçon reference as the lower limit, or using the set of constraints in (3.9), that is borrowing it from the minimum zone formulation. The advantage of using the ring limaçon type of formulation is that it results in a smaller problem, with three parameters instead of four which means less computation (although not a great deal). However, as long as the number of parameters is limited to three, it does not matter whether the reference is a circumscribing, inscribing or central to the feasible zone, it will result in a better implementation of the gauge algorithm.

The use of the centroid of the points as the origin to the local system does not result in a good limaçon approximation to the original data set, as it can be seen from the eccentricity ratio λ in table 3.25. However, the approximation is very much improved if the origin of the system is defined by the centre of a least squares reference fitted to the profile. Thus, from table 3.25, the eccentricity ratios for this case are well within the limits considered satisfactory. Therefore, the use of this strategy allows the use of the limaçon reference, which in turn makes possible the use of the gauge algorithm.

It must be stressed that the comparison between the computational cost of the gauge algorithm and the exchange algorithms is only valid because of the special conditions that arise in the problem discussed. The simple exchanges relevant to roundness reference fitting are extremely fast and so provide a target for other approaches to aim at. However they apply only to references that can be linearised and their efficiency, while remaining better than other optimisation techniques, reduces as the geometric complexity increases. The gauge algorithm presented here also requires a linear parameterisation since Fletcher's algorithm is based on linear programming. Nevertheless, for inspection problems that are naturally linear and some others that can be well-approximated, the approach implemented by the gauge algorithm is both usable and useful.

Chapter 4

Inspection of Squareness Features

4.1 Introduction

This chapter extends the algorithmic approach presented in the previous chapter to the inspection of form and orientation of related features of prismatic components. Amongst the geometric errors of orientation classified in design standards [e. g. BSI, 1990], perpendicularity is representative of the typical requirements imposed by this type of geometric tolerancing. Therefore, although the algorithmic approach here presented is generally valid for any class of geometric orientation inspection, the discussion is based on a specific case of perpendicularity of two related planar surfaces. Specifically, an algorithm is proposed for the combined inspection of squareness and flatness of two related surfaces.

Although prismatic workpieces are as common and important in manufacturing industry as circularly shaped ones, the inspeçtion of orientation of geometric features such as squareness, parallelism and angularity has received relatively little attention from standards committees over the past decades.

Standards exist [e. g. BSI, 1989] that bring recommendations for the assessment of geometric features. The general idea of best-fit geometric element

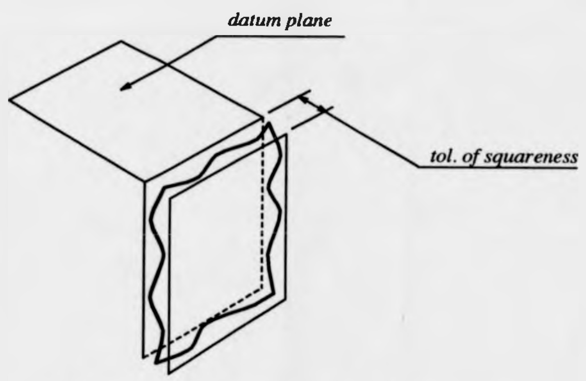


Figure 4.1: Tolerance zone of squareness.

[Forbes, 1989] is used in order to evaluate position, size and form of geometric features. Following this approach, a perpendicularism (or angularity) assessment would be carried out by measuring the relative angular position between reference lines or planes fitted to the feature and datum.

On the other hand, according to design specifications, a feature has an acceptable error of perpendicularism (squareness) for instance, if its axis or surface is within a tolerance zone in general defined by two parallel planes perpendicular to the datum feature, which in turn may be a plane or an axis. The tolerance value defines the width of the zone. Figure 4.1 illustrates the squareness tolerance zone for surfaces. Alternatively, the tolerance zone may be the space within a cylinder, when the squareness of a cylinder axis to a datum plane is to be controlled.

As the tolerance of squareness of two related surfaces requires the surface to lie between two parallel planes, errors of flatness of the feature are implicitly controlled. Consequently, what is suggested by design standards is that inspection of squareness should proceed by checking whether the data set representing the feature is contained within a square template or framework defined by the squareness tolerance zone and datum.

4.2 Flatness and Squareness Inspection

Of the many possible ways a plane can be parameterised, for metrology it is recommended that it be parameterised by the direction cosines (a, b, c) of a line normal to it and a point (x_o, y_o, z_o) on the plane [Anthony et al, 1991]. So, it can be written as

$$ax + by + cz + d = 0 \tag{4.1}$$

where $d = -ax_o - by_o - cz_o$ is the normal distance of the origin from the plane. Following this parameterisation, the Euclidian (normal) deviation of a point (x_i, y_i, z_i) from the plane is given by

$$\epsilon_i = ax_i + by_i + cz_i + d \tag{4.2}$$

which is linear in its parameters. Fundamentally, this definition of residual places it normal to the reference plane, making it suitable for measuring residuals of data points acquired using a CMM.

Based on this parameterisation, it is possible to enclose data points representing a plane surface within a planar zone using only linear constraints in its formulation. Note that on using the general equation of the plane or the slope intercept form of equation, the residual is defined as a non-linear function of its parameters (see section 2.4.6), which would result in non-linear constraints in the formulation of the planar zone problem.

Thus, in case of inspection of flatness, a set of N sampled data points (x_i, y_i, z_i), $i = 1, \cdots, N$ representing a plane surface is within a planar zone of width t_f, that is the tolerance of flatness, if there are parameters a, b, c and d such that

$$\begin{aligned} ax_i + by_i + cz_i + d &\geq 0 \\ ax_i + by_i + cz_i + d - t_f &\leq 0 \end{aligned} \tag{4.3}$$

This formulation does not put the reference plane at the centre of the reference zone (although alternatively, the reference could be placed at the centre of the zone and the constraints be defined by shifting the reference a distance equivalent to $t_f/2$). If a feasible region is defined from the combination of the constraints in (4.3), the flatness error is less than the tolerance specification. This is the same type of formulation as for roundness inspection, namely of finding a feasible point for a linearly constrained problem. It involves inequality constraints only, as before, and the upper and lower variable limits can be easily defined. Therefore, the same algorithmic approach [Fletcher, 1970a] can be used.

In the case of perpendicularity of two related surfaces, one of the surfaces may be a general, functional datum of a particular component, defined at the design stage for both machining and inspection purposes. Otherwise, it is usual to take one of the surfaces as a local datum, just for inspection purposes.

First of all, it is assumed that one of the surfaces is a general datum and that the quality of the datum surface is so good (in comparison to the quality of the other surface to be inspected for squareness) that any best-fit plane has no appreciable possibility of deviating from the design orientation. Then, the surface to be inspected for squareness will be within tolerance if it is possible to enclose the data points between two parallel planes that are exactly perpendicular to the datum plane (a best fit to the points representing it) and separated from each other by no more than the squareness tolerance value. Thus, the proposed procedure for inspecting squareness features in such situations is:

1. find a best-fit plane to the set of points representing the datum. The reference plane so computed has orientation parameters (a_d, b_d, c_d), the direction cosines of the line normal to the plane;

2. find a feasible point $[a, b, c, d]$ to the constraints

$$\begin{aligned} ax_i + by_i + cz_i + d &\geq 0 \\ ax_i + by_i + cz_i + d - t_s &\leq 0 \end{aligned} \tag{4.4}$$

and

$$aa_d + bb_d + cc_d = 0 \tag{4.5}$$

for $i = 1, \cdots, N$, where (a, b, c) are the direction cosines of the planes and t_s the value of the tolerance of squareness. The perpendicularity of the planar zone

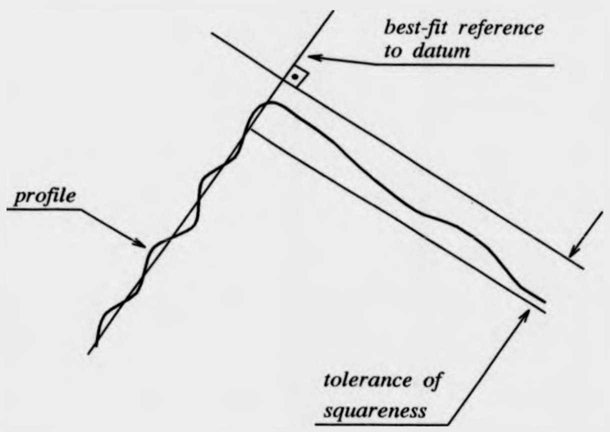


Figure 4.2: Planar frame perpendicular to the best-fit reference to datum feature.

enclosing the data points is imposed by the equality constraint in (4.5). Note that the datum feature might be an axis, in which case the equation given in 4.5 would be replaced by (see e. g. [Ayre and Stephens, 1956])

$$aa_d + bb_d + cc_d = 1 \tag{4.6}$$

The set of constraints defined in (4.4) and (4.5) tests whether it is possible to contain the data points in a frame defined by parallel planes perpendicular to another plane of known orientation, as depicted in figure 4.2 for the two-dimensional case.

This problem involves only linear constraints and therefore the algorithm presented by Fletcher [1970a] can still be used. However, in this case one equality constraint exists. Equality constraints must be satisfied by any solution throughout iterations and therefore are always active. The initial vertex is defined by the intersection, in parameter space, of the hyper-plane defined by the equality constraint given in (4.5) with three other hyper-planes defined by lower or upper variable bounds. Fletcher [1970a] presents a way of defining the bounds so as the constraints in the initial vertex are independent and the basis matrix is nonsingular. In this case, one column of the initial basis matrix is $B = [a_d \;\; b_d \;\; c_d \;\; 0]^T$, that is the normal vector of the plane defined by the equality constraint. The other three columns (the sequence of the columns in the basis matrix is irrelevant) will be vectors $\pm e_i$ of the unit matrix ($i = 1, \cdots, 4$), that is the normal vectors of the planes defined by the bound constraints. As in this case there is only one equality constraint, the method presented by Fletcher [1970a] can be simplified to the following rule: the vector e_i not to enter the basis will be the i^{th} corresponding to the largest element (i^{th}) of the vector $[a_d \;\; b_d \;\; c_d \;\; 0]^T$. Apart from the way in which the initial vertex is defined, the basic algorithm has already been discussed in section 3.6 and its C code implementation is given in appendix B for the roundness and centre position inspection case.

It is important to observe that if the datum is not officially toleranced, and not very smooth, the orientation of the least squares or minimax best fits plane may deviate considerably from the required datum orientation. To illustrate this case, consider the situation when, in order to inspect squareness of a surface in relation to a datum, the datum face is placed onto another (flat) basal surface and the squareness is measured in relation to that base. This does not require the datum be flat, merely that the contact points define a suitable plane. Thus, the way in which the orientation of the datum is defined depends on functional requirements. However, in most cases the least squares or minimax best fit will be an adequate way of defining the datum orientation, provided the datum surface is adequately smooth. Therefore, it is important that a datum feature be fully specified.

Another case to be considered is the inspection of squareness of two related surfaces, where one of the surfaces is assumed as a local datum for convenience at inspection In this case, it is also important that the flatness of the surface assumed as a datum be specified. In this case. the inspection should test whether it is possible to contain the data points in a frame defined by two planar zones with widths equivalent to the value of the flatness and squareness tolerances, as depicted in figure 4.3 for the two-dimensional case.

One advantageous consequence of imposing a flatness tolerance zone on the local datum surface is that, when the squareness error is over the limit imposed by a zone defined as in figure 4.2, it may be possible to contain the data points

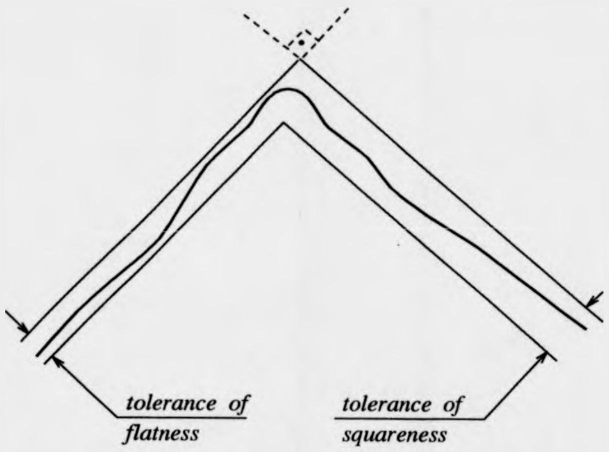


Figure 4.3: Squareness frame enclosing the profile.

within the frame defined by the squareness and flatness tolerance zones, provided that the flatness error is less than the maximum allowed error, as in figure 4.3. Therefore, when considering the combination of the errors, a reduction in the error of flatness allows an increase in the affordable limit for the error of squareness.

This inspection problem can then be formulated as: test whether there is a feasible solution to the constraints

$$\begin{aligned} a_s x_i + b_s y_i + c_s z_i + d_s &\geq 0 \\ a_s x_i + b_s y_i + c_s z_i + d_s - t_s &\leq 0 \\ a_d x_i + b_d y_i + c_d z_i + d_d &\geq 0 \\ a_d x_i + b_d y_i + c_d z_i + d_d - t_f &\leq 0 \end{aligned} \tag{4.7}$$

and

$$a_s a_d + b_s b_d + c_s c_d = 0 \tag{4.8}$$

again for $i = 1, \cdots, N$, where (a_s, b_s, c_s) and (a_d, b_d, c_d) are the orientation parameters of the planes enclosing the surface to be measured for squareness and the local datum respectively, and t_s and t_f the tolerance values for squareness and flatness respectively.

This problem, as it is formulated, involves an equality constraint which is not linear in its parameters and therefore, non-linear programming techniques

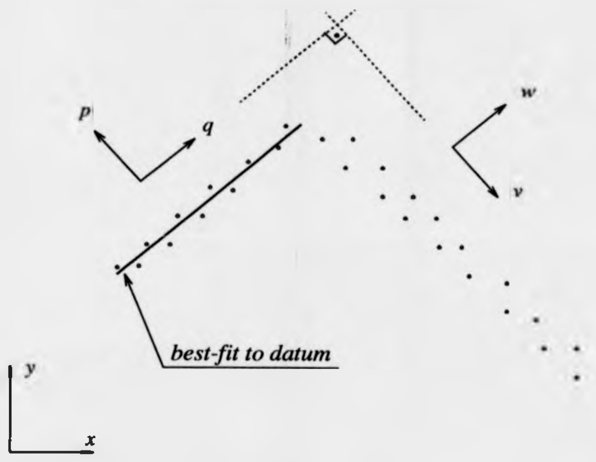


Figure 4.4: Local coordinate systems.

would be required. However, it is possible, by a change of coordinate systems, to reformulate the problems using only linear constraints. This is discussed in the next section.

4.3 An Algorithm for Combined Squareness and Flatness Inspection

The problem of inspecting the squareness of two related surfaces and the combined flatness of the other surface, assumed as a local datum, as formulated in (4.7) and (4.8), can be expressed by a linear model by using coordinates system transformations in the representation of the data points.

The data points representing the datum surface can be re-written in a local coordinate system, which has one axial plane nominally parallel to the datum surface. In the same way, the data points representing the surface to be inspected for squareness can be transformed into another coordinate system, orthogonal to the first local coordinate system. These transformations are illustrated in figure 4.4, for the two-dimensional case. The data points, originally expressed in the (x, y) coordinate system are split and rotated to two local coordinate systems,

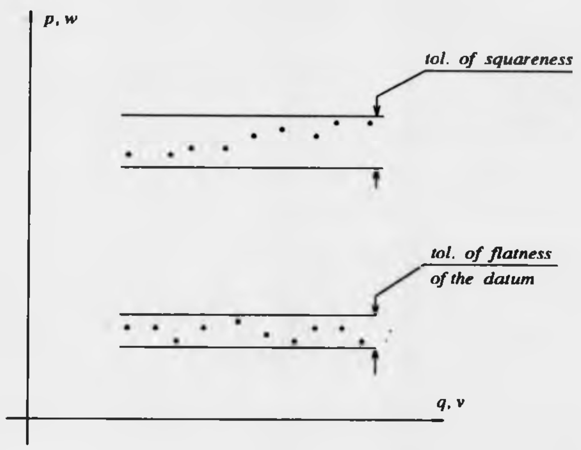


Figure 4.5: Two parallel planar zones plotted on two aligned coordinate systems (p, q) and (w, u).

(p, q) and (w, u). The system (p, q) is parallel to a reference line through the datum, and the (w, u) system is perpendicular in relation to the (p, q) coordinate system.

Then, if the two local coordinate systems are aligned and combined together, the inspection problem becomes one of finding whether there exist two parallel planar zones, of widths defined by the tolerance values, enclosing the data points. This strategy is illustrated in figure 4.5. The relative positions of the two zones in the space is arbitrary and unimportant.

For the three-dimensional case, the two original sets of data points, each one representing a surface, are re-expressed in two orthogonal local coordinate systems, say $(p_i, q_i, r_i), i = 1, \cdots, N_1$ and $(w_j, u_j, v_j), j = 1, \cdots, N_2$. Therefore, the formulation of the inspection problem given in (4.7) and (4.8) becomes: find parameters $[a, b, c, d, e]$, if they exist, to satisfy

$$\begin{aligned} ap_i + bq_i + cr_i + d &\geq 0 \\ ap_i + bq_i + cr_i + d - t_f &\leq 0 \\ aw_j + bv_j + cu_j + e &\geq 0 \\ aw_j + bv_j + cu_j + e - t_s &\leq 0 \end{aligned} \tag{4.9}$$

where (a, b, c) are the orientation parameters of the two parallel planar zones, and d and e the distances from the origin of the reference planes (the other parameters are the same as before).

This formulation involves fewer parameters than the formulation in (4.7) and (4.8) (five against eight parameters), and, most important, the set of constraints in (4.9) contains only linear inequalities. Consequently the problem of finding a feasible point to this set of constraints can be solved as before, using the algorithm presented by Fletcher [1970a].

Since the algorithm proposed by Fletcher is more sensitive, in terms of computing time, to an increase in the number of parameters (the amount of computation increases at a rate of mn^2, where n is the number of variables and m of constraints), it is better to re-formulate this problem so as to reduce the number of variables. This is possible by using the slope-intercept form of the equation of a plane.

Thus, the inspection problem formulated in (4.9) is re-expressed as: find a solution $[A, B, C, D]$ such that

$$\begin{aligned} Ap_i + Bq_i + C &\geq r_i \\ Ap_i + Bq_i + C - t_f &\leq r_i \\ Aw_j + Bv_j + D &\geq u_j \\ Aw_j + Bv_j + D - t_s &\leq u_j \end{aligned} \tag{4.10}$$

again for $i = 1, \cdots, N_1$ and $j = 1, \cdots, N_2$ (where N_1 and N_2 are the number of data points of each data set). The parameters (A, B) are the slope of the intersections of the planes with the p-r- and q-r-planes for one coordinate system and w-v- and u-v-planes for the other, and C and D are the intercepts with axes r and u respectively.

Note that, with the problem reformulated in this way, the separation between the planes is now measured parallel to one of the axes of the coordinate systems, and not normal to the planes as before. However, since the planes are nearly parallel to the equivalent x-y-planes of the local coordinate systems used, this

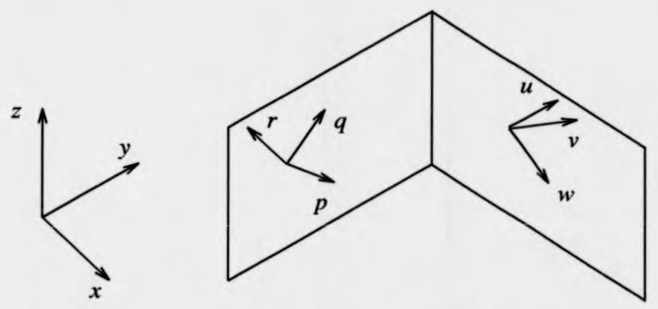


Figure 4.6: Orthogonal three-dimensional local coordinate systems.

approximation, while reducing the computation effort, brings no practical losses in terms of accuracy.

Therefore, the following algorithm is proposed:

1. find a best-fit reference plane through the points representing the datum plane;
2. define a coordinate system, say (p, q, r), such that the p-q-plane is aligned with the reference plane defined in 1 and re-write the data points representing the datum surface in this local coordinate system;
3. define another local coordinate system, say (w, u, v), by rotating the system defined in 2 by 90^o (degrees) in such a way that the second local coordinate system is parallel (nearly) to the surface (as illustrated in figure 4.6). Re-write the data points representing the surface in this local system;
4. apply Fletcher's algorithm to the set of linear constraints in (4.10) in order to find out whether a feasible point is defined or not.

This algorithm is discussed in detail in the next section.

4.4 Algorithm Implementation and Test

4.4.1 Objectives

The objective of the work described here is to test the practicability of the proposed algorithm for use in on-line inspection as well as to check its correctness in terms of pass/fail result of inspection. Therefore, the algorithm described above, for inspecting squareness and flatness of two related surfaces, (from now on called the squareness gauge algorithm), has been implemented and tested experimentally.

In order to do this, the following sequence was undertaken:

- collect data from a set of testpieces using a coordinate measuring machine (CMM);
- measure their squareness errors and the flatness errors of the datum surfaces;
- based on the previous results, define examples of squareness and flatness tolerance values under and over the calculated errors so as to simulate tolerance limits from design.
- input the tolerance information to the squareness gauge algorithm to check whether the data is contained within tolerance and to measure its efficiency.

The efficiency of the squareness gauge algorithm was compared with a NAG [NAG, 1990] implementation of the two-phase method of linear programming [see Hadley, 1962)] (see sections 3.3 and 3.5) by feeding to it the same tolerance information.

As there was no available algorithm in the open literature to compute the squareness error in terms of the planar zones enclosing the surface, as pictured in figure 4.1, this was computed by linear programming, as described in section 4.4.4.

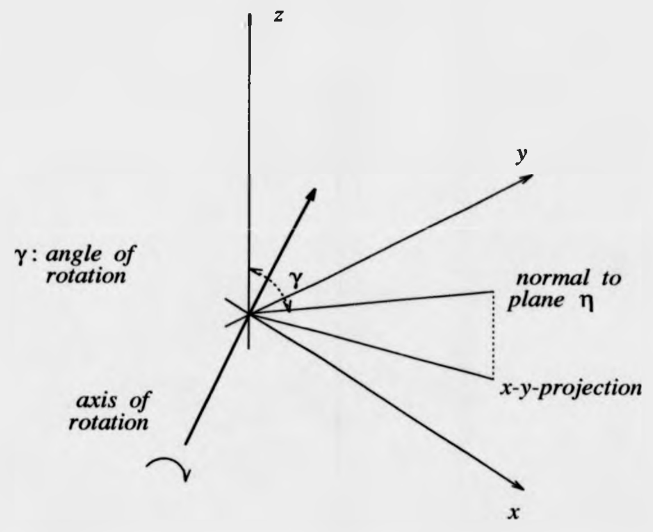


Figure 4.7: Axis of rotation normal to the projection onto x-y of the normal to plane η.

Details about data acquisition, algorithm implementations and test procedures are given in the following sections.

4.4.2 Algorithm Design and Implementation

The first step is to compute a best-fit reference plane through the data points representing the datum surface. A least squares plane fitting algorithm, as proposed by Forbes [1989] (see section 2.5.5), was implemented to determine the best-fit plane (the main fragment of the source code of this algorithm is reproduced in appendix B). This plane, named η for further reference, has its orientation given in terms of the direction cosines of a line normal to the plane, (a_η, b_η, c_η). Thus, the angle between the z-axis and normal to the plane η is given by the inverse cosine of c_η, that is $\gamma = \cos^{-1} c_\eta$.

Therefore, in order to define a coordinate system which has one axial plane nominally aligned with the datum surface, the original coordinate system is rotated by an angle γ around a line normal to the projection, onto the x-y-plane, of a line passing through the origin and with direction cosines (a_η, b_η, c_η), that

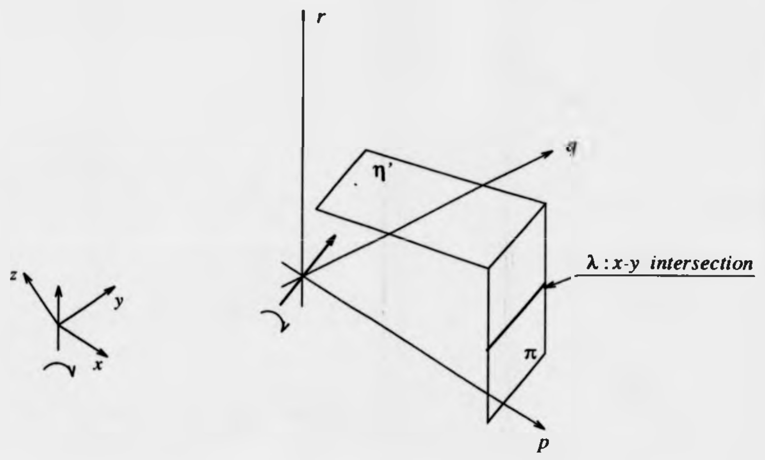


Figure 4.8: Rotation of (p, q, r) around an axis parallel to intersection line l.

is, normal to the plane η. This rotation is represented in figure 4.7, and the new coordinate system, (p, q, r), will have one of its planes, the p-q-plane, parallel to the least squares plane.

All the data points are then re-written in this new coordinate system. Note that in this local coordinate system, the points representing the datum surface will define a plane (η' as illustrated in figure 4.8) nearly parallel (though exactly parallel in the case of least squares fitting) to the p-q-plane, while the other data points, representing the nearly perpendicular surface will define a nearly vertical plane, say π (figure 4.8), with direction cosines (a_π, b_π, c_π). The intersection of the plane π with the p-q-plane will define a line, named λ in figure 4.8.

The second local coordinate system, named (w, u, v), is defined by rotating the (p, q, r) system by 90^o around a line passing through the origin and parallel to λ, represented in figure 4.8. Note that the orientation of line λ is given by the normal to the projection onto p-q-plane of a line with direction cosines (a_π, b_π, c_π), that is normal to the plane π.

Consequently, the two transformations involve rotation of a coordinate system around a general axis. This transformation is given by the following rotation

```
if  (ao bo >= 0 ) then
        $\theta$ = arctan ao/bo ;
        a = cos ($\pi - \theta$ ) ;
else
        $\theta$ = arctan ( 1 - ao ) / bo ;
        a = cos$\theta$ ;
b = cos ($\pi/2 - \theta$ ) ;
c = 0 ;
```

Figure 4.9: Algorithm for computing the orientation parameters (a, b, c) of the axis of rotation.

matrix [Faux and Pratt, 1979, pp. 70]

$$\begin{bmatrix} a^2 + \cos\gamma(1 - a^2) & ab(1 - \cos\gamma) + c\sin\gamma & ca(1 - \cos\gamma) - b\sin\gamma \\ ab(1 - \cos\gamma) - c\sin\gamma & b^2 + \cos\gamma(1 - b^2) & bc(1 - \cos\gamma) + c\sin\gamma \\ ca(1 - \cos\gamma) + b\sin\gamma & bc(1 - \cos\gamma) - a\sin\gamma & c^2 + \cos\gamma(1 - c^2) \end{bmatrix} \tag{4.11}$$

where (a, b, c) are the direction cosines of the axis and γ the rotation angle. The new coordinate system is given by **r'** $=$ **A** . **r**, where **A** is the rotation matrix and **r** and **r'** are vectors whose elements are the coordinates of the original and transformed systems respectively.

In both cases the rotation axis is normal to the projection onto a plane containing the rotation axis of a line passing through the origin and normal to a given plane. Therefore, a procedure was implemented in order to define the orientation of the axis of rotation.

Assuming that (a_o, b_o, c_o) are the orientation parameters of a line normal to a given plane and that $c_o \geq 0$, the orientation (a, b, c) of the rotation axis is computed as described in the algorithm in figure 4.9.

Thus, the complete procedure is as follows:

1. compute the least squares best-fit reference plane η to the data points representing the datum surface. The direction cosines of the normal to η are (a_η, b_η, c_η). If $c_\eta < 0$ then multiply the three parameters by -1.

2. a local coordinate system (p, q, r) is defined by rotating the original sys-

tem by an angle $\gamma = \cos^{-1} c_\eta$ around an axis whose orientation is defined as described in the algorithm in figure 4.9 (given that (a_η, b_η, c_η) are the orientation parameters of the normal to the plane);

3. Re-write all data points in this new coordinate system and compute a best-fit reference plane through the data points representing the surface to be measured for squareness (represented in figure 4.8 by plane π). The least squares algorithm used in 1 is used here as well. The direction cosines of the normal to π is (a_π, b_π, c_π). Again, If $c_\pi < 0$ then multiply the three parameters by -1.

4. a second local coordinate system, (w, u, v), is defined by rotating the (p, q, r) system by 90^o (positive angle) around an axis whose orientation is defined as described in the algorithm above (where now (a_π, b_π, c_π) are the orientation parameters of the normal to the plane);

5. re-write the data points corresponding to the surface to be measured for squareness in the (w, u, v) coordinate system.

6. apply Fletcher's algorithm to the set of constraints given in (4.10). If, for given values of t_s and t_f the tolerances of squareness and flatness, a feasible point is defined, then the feature passes inspection. Otherwise, it fails inspection.

The set of constraints defines no limits for the variables $[A, B, C, D]$. However, as bounds on the variables are required by the method, very large positive and negative values can be chosen for the variable bounds, as this will not affect the number of floating point operations or the number of iterations. Alternatively, limits can be easily estimated from the geometry of the problem. The planes will be nearly parallel to the p-q- and w-u- axial planes, thus slope parameters (A, B) will be close to zero. The intercepts C and D can also be estimated from the r and v ordinates, as they will be nearly the same for all data points of each set. In this implementation, conservative limits of $[-0.1, 0.1]$ were chosen for the parameters (A, B), while for the parameters C and D limits of $[-1.5, 1.5]$ times

z, where z is the average of the r and v ordinates, respectively for the parameters C and D.

It is suggested here that the initial vertex be defined by the the upper parameter bounds and that, during data acquisition, the original coordinate system, local to the testpiece, be defined in a way that the planes be described by positive (x, y, z) coordinate points. This will ensure that, in parameter space, the hyperplanes defined by the constraints intersect the parameter axes in their positive segment, bringing the feasible region (or at least part of it) to the positive region of the parameter space and, therefore closer to the initial vertex so proposed. Otherwise, no strict rule is proposed to choose an initial vertex apart from that proposed by Fletcher [1970a].

The procedure described above was implemented in C language, single precision floating point.

4.4.3 Data Acquisition

A set of 7 prismatic testpieces, presenting squareness and flatness features, were used for these tests. The testpieces included ground and milled surfaces in carbon steel and aluminium, with varying sizes in the order of tens of millimeters.

The relevant surfaces of each testpiece were sampled by using a coordinate measuring machine, *LK Micro Four* (whose characteristics have been described in section 3.8.2), fitted with a touch trigger probe and operated under manual mode. A set-up procedure was executed in order to identify the probe and to establish a coordinate system local to the components being measured. Following this procedure, each surface of each testpiece was evenly sampled by a number of three-dimensional (x, y, z) points recorded by the CMM computer. In order to avoid any processing of the data by the software of the CMM dedicated computer, the data points were transferred by hand to ASCII data files. The data files all consisted of a three-dimensional array containing the x-y-z-coordinates of the data points in relation to a local coordinate system. Table 4.1 presents the number of sampled data points for each data set and their measured flatness and squareness errors. The data sets are listed in appendix A.2.

4.4.4 Test Procedures

The error of squareness of the testpieces was measured as the planar zone enclosing the surface and perpendicular to a minimax reference plane fitted to the points representing the datum feature. Hence, the flatness error of the datum was given by the minimax zone enclosing it. These errors were then computed by solving the following linear programming problem: minimise $Z = h_f$ subject to the constraints

$$\begin{aligned} Ap_i + Bq_i + C + h_f &\geq r_i \\ Ap_i + Bq_i + C - h_f &\leq r_i \\ Aw_j + Bv_j + D + h_s &\geq u_j \\ Aw_j + Bv_j + D - h_s &\leq u_j \\ h_f &\geq 0 \\ h_s &\geq 0 \end{aligned} \tag{4.12}$$

for $i = 1, \cdots, N_1$ and $j = 1, \cdots, N_2$, where N_1 and N_2 are the number of sampled data points for the datum and surface. A NAG implementation (function EO4MBF, [NAG, 1990]) of the Simplex method of linear programming was used for solving this problem. In order to call this function, define constraints, objective function and input the data sets to it, a piece of code as written in Fortran 77 language. The error of flatness was given by $2h_f$ and the error of squareness by $2h_s$. Table 4.1 presents the flatness and squareness error measured for each data set.

In order to test the efficiency of the squareness gauge algorithm to get to a pass/fail result of inspection, simulated tolerance values were defined, for each testpiece, as fractions over and under the measured errors of flatness and squareness.

Following the same strategy adopted in chapter 3, a number of examples of tolerance values were defined so as as to consider a range of likely practical situations, as follows:

data set (datum)	sampled points	flatness (mm)	data set (surface)	sampled points	surf. squareness (mm)
square-1-1	12	0.0476	square-1-2	25	0.08
square-2-1	12	0.19	square-2-2	25	1.036
square-3-1	6	0.017	square-3-2	6	0.075
square-4-1	6	0.0069	square-4-2	6	0.0212
square-5-1	6	0.0086	square-5-2	6	0.0514
square-6-1	6	0.00694	square-6-2	6	0.0132
square-7-1	6	0.012	square-7-2	6	0.0156

Table 4.1: Error of flatness of the datum and squareness of the related surface for each data set.

- the flatness tolerance and the squareness tolerance are equal to or greater than the actual errors (region B in figure 4.10, compare to figure 3.16a);
- the flatness tolerance and the squareness tolerance are less than the actual errors (region C);
- the flatness tolerance is equal to or greater than the actual error and the squareness tolerance is less than actual error (region A);
- the flatness tolerance is less than the actual error and the squareness tolerance is equal to or greater than the actual error (region D).

Thus, values of $0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3$ times the actual errors of flatness and squareness were used as tolerances of flatness and squareness respectively.

In addition to this, in order to compare the efficiency and result of inspection of the squareness gauge algorithm with that of the two-phase method of linear programming [Hadley, 1962], the same NAG function used for calculating the errors was used. This time however a different piece of code was written so as to input to the function the inspection problem as given by the set of constraints in (4.10).

Therefore, overall, for each testpiece, a set of 49 different tolerance zone combinations were fed to the squareness algorithm, as well as to the NAG two-phase implementation, which in each case returned with a pass/fail flag.

The routines were run under Unix, on a SUN 4/330, with 48 Mbytes of RAM (Random Access Memory) and speed of 16 MIPS (Million of Instructions Per Second). The efficiency of the algorithms was measured in terms of number of iterations, arithmetic operations and computation time. The computation time was estimated by running the programme under the "time" Unix command, which returns the CPU user time.

4.5 Results and Discussion

Tables 4.3 to 4.16 present the number of iterations to get to a solution for each testpiece, respectively for the gauge algorithm and for the NAG implementation of the two-phase method. The result of inspection was signaled by appending a minus sign to the number of iterations if it was a fail. The average number of iterations taken, over the whole set of testpieces, for the gauge and NAG algorithms are shown graphically in figures 4.11 and 4.12 respectively. Table 4.2 presents a comparison between the number of iterations and computation time of the squareness gauge algorithm and the NAG two-phase algorithm.

Considering the squareness gauge algorithm, the pattern of results is typical of the behaviour seen in all the tests carried out, including the ones shown in chapter 3, for roundness and centre position inspection. Again, it takes a little longer to evaluate cases of easy acceptance (tolerance greater than error by a large amount) than easy rejection. Also the number of iterations rises when the tolerance is close to the actual error. The reason for this, as discussed in chapter 3, is that when this condition is approached, the feasible region effectively shrinks towards a point and the algorithm is required to make an equality decision rather than one based on an inequality. Thus, one or two extra iterations are required to establish which side of the boundary the solution lies.

The average number of iterations is however higher in this case than for the ring or zone gauge algorithms, although it has the same size in terms of number of variables as the zone gauge. The reason for this could be that the special start procedure devised for circularity cases is more effective than that used for squareness cases.

The squareness gauge algorithm took an average of 8 iterations, with a computation time of approximately 0.1 seconds, running under the conditions already mentioned. The use of a larger data set, as expected, made no no appreciable difference in terms of number of iterations, as can be seen from tables 4.3 and 4.5, for two larger data sets.

The NAG implementation of the two phase method of solution took an average of 13 iterations, with a computation time of approximately 0.3 seconds. In this case however, an increase in the number of data points does reduce the efficiency of the algorithm, making it less appropriate for cases in which larger data sets are required, as it can be seen from tables 4.4 and 4.6.

This approach brings to evidence the importance of specifying the tolerance of flatness of a local datum, in the case of inspection of squareness of two related surfaces. When the flatness error of the datum is less than the allowed value, the effect of an angular error from perpendicularity of the related surface over the allowed limit (when expressed the squareness tolerance from design as a maximum angular error from perpendicularity), is compensated by the fact that the datum surface is smother than expected. Therefore, when the flatness error is sufficiently less than the allowed error, the result of inspection will be positive even when the squareness tolerance is slightly smaller than the actual error. This is demonstrated by the results of inspection, in tables 4.3 to 4.16, and illustrated in figure 4.10, region A.

On the other hand, in cases where the datum is not only for the purpose of squareness specification of one surface related to another, but a more general datum, used for specifying other features as well (e. g. the squareness or flatness of other surfaces of the same component), then the situation may be not quite the same as discussed in the previous paragraph, as the squareness constraint may be required in relation to a very well defined orientation. In such cases, the squareness inspection problem may be formulated as discussed in section 4.2.

Finally, the formulation of the inspection problem as in (4.9) further explores

data set	iterations		arith. operations	comp. time (sec)	
	gauge	two-phase	gauge	gauge	two-phase
square-1	8	23	14028	0.2	0.6
square-2	8	14	7683	0.2	0.4
square-3	8	9	3357	0.1	0.2
square-4	9	13	3619	0.1	0.3
square-5	10	10	4147	0.1	0.2
square-6	9	10	3757	0.1	0.2
square-7	9	12	3766	0.1	0.3

Table 4.2: Number of arithmetic operations and computation time for the squareness gauge algorithm and the NAG two-phase method implementation, for some examples of tolerance zones.

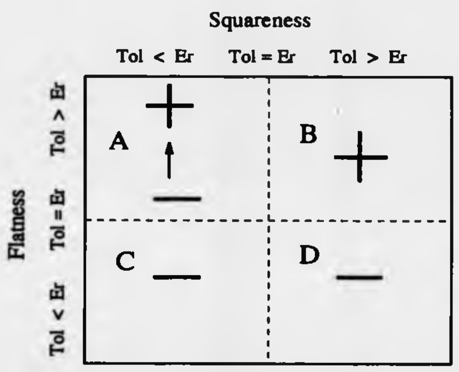


Figure 4.10: Result of inspection (+/-: pass/fail) based on the squareness gauge algorithm.

the linear parameterisation of lines and planes as presented by Forbes [1989]. Transformation of this formulation as in (4.10) reduces the number of parameters but approximates the non-linear deviation, of this type of parameterisation, given by $\epsilon = h\sqrt{A^2 + B^2 + 1}$ to h. However, as the parameters A and B, the slopes, are very close to zero, the error will be, in general, negligible.

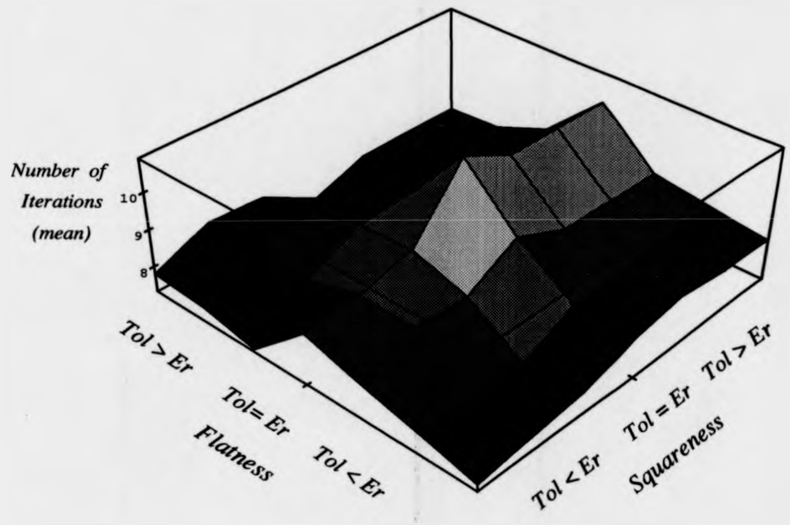


Figure 4.11: Average number of iterations of the squareness gauge algorithm over the whole set of profiles.

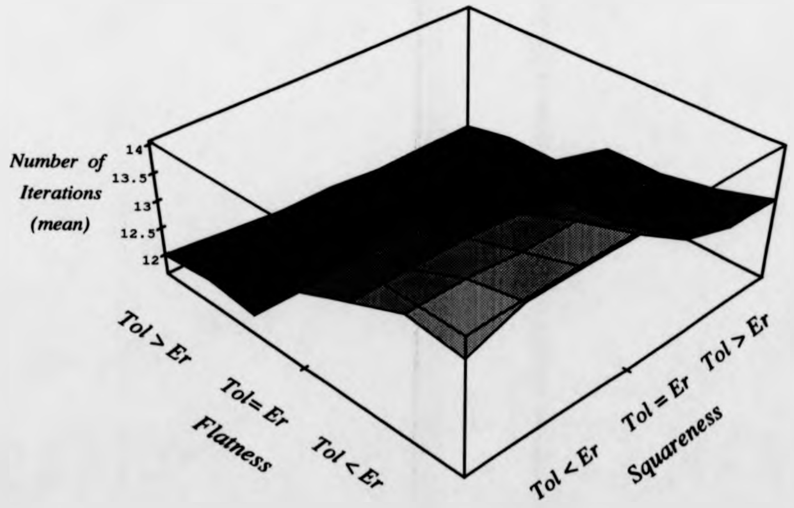


Figure 4.12: Average number of iterations of the NAG two-phase implementation over the whole set of profiles.

datum flatness (tolerance /error) / squareness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-7	-7	-7	-8	7	6	6
0.8	-7	-7	-7	-8	7	6	6
0.9	-7	-7	-8	-8	7	6	6
1.0	-8	-8	-10	12	8	6	6
1.1	-8	-9	-8	10	8	6	6
1.2	-8	-8	-8	9	8	6	6
1.3	-8	-8	-8	9	8	6	6

Table 4.3: Squareness gauge algorithm search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-1-1 and square-1-2.

datum flatness (tolerance /error) / squareness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-18	-22	-22	-22	22	23	23
0.8	-22	-22	-22	-22	22	23	23
0.9	-23	-23	-23	-23	23	23	23
1.0	-25	-25	-23	23	23	23	23
1.1	-24	-25	-23	23	23	23	23
1.2	-24	-25	-23	23	23	23	23
1.3	-24	-24	-23	23	23	23	23

Table 4.4: NAG two-phase search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-1-1 and square-1-2.

datum flatness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
squareness (tolerance /error)							
0.7	-6	-7	-7	-7	7	6	6
0.8	-6	-7	-7	-8	7	7	5
0.9	-5	-5	-7	-8	7	7	6
1.0	-5	-6	-8	14	7	6	6
1.1	-5	-5	-8	14	7	6	6
1.2	-5	-6	-8	14	7	6	6
1.3	-5	-6	-8	14	7	6	6

Table 4.5: Squareness gauge algorithm search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-2-1 and square-2-2.

datum flatness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
squareness (tolerance /error)							
0.7	-18	-18	-17	-17	14	14	14
0.8	-18	-18	-17	-17	14	14	14
0.9	-18	-17	-17	-17	14	14	14
1.0	-18	-17	-15	14	14	14	14
1.1	-15	-14	-14	14	14	14	14
1.2	-15	-14	-14	14	14	14	14
1.3	-15	-14	-14	14	14	14	14

Table 4.6: NAG two-phase search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-2-1 and square-2-2.

datum flatness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
squareness (tolerance /error)							
0.7	-7	-8	-9	-9	9	8	8
0.8	-8	-8	-9	-9	9	8	8
0.9	-8	-8	-9	-9	9	8	8
1.0	-8	-8	-9	9	9	8	8
1.1	-8	-8	-9	9	9	8	8
1.2	-8	-8	-9	9	9	8	8
1.3	-8	-8	-9	9	9	8	8

Table 4.7: Squareness gauge algorithm search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-3-1 and square-3-2.

datum flatness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
squareness (tolerance /error)							
0.7	-11	-9	-9	-9	9	9	9
0.8	-9	-9	-9	-9	9	9	9
0.9	-9	-9	-9	-9	9	9	9
1.0	-9	-9	-9	9	9	9	9
1.1	-9	-9	-9	9	9	9	9
1.2	-9	-9	-9	9	9	9	9
1.3	-9	-9	-9	9	9	9	9

Table 4.8: NAG two-phase search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-3-1 and square-3-2.

datum flatness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
squareness (tolerance /error)							
0.7	-8	-8	-8	-9	-9	-8	-10
0.8	-8	-8	-8	-9	-9	-8	-10
0.9	-8	-8	-8	-9	-9	-8	10
1.0	-8	-8	-8	9	8	8	8
1.1	-8	-8	-8	9	8	8	8
1.2	-8	-8	-8	9	8	8	8
1.3	-8	-8	-8	9	8	8	8

Table 4.9: Squareness gauge algorithm search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-4-1 and square-4-2.

datum flatness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
squareness (tolerance /error)							
0.7	-14	-13	-13	-13	-11	-11	-11
0.8	-14	-13	-13	-13	-11	-11	-11
0.9	-14	-13	-13	-13	-11	-11	11
1.0	-12	-11	-11	11	11	11	11
1.1	-12	-10	-10	10	10	10	10
1.2	-10	-10	-10	10	10	10	10
1.3	-10	-10	-10	10	10	10	10

Table 4.10: NAG two-phase search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-4-1 and square-4-2.

datum flatness (tolerance /error) / squareness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-8	-8	-8	-8	-8	-8	-8
0.8	-8	-8	-8	-8	-8	-8	-8
0.9	-8	-8	-8	-8	-9	-9	9
1.0	-12	-12	-12	11	11	11	9
1.1	-11	-11	-11	10	10	10	9
1.2	-11	-11	-11	10	10	10	9
1.3	-11	-11	-11	10	10	10	9

Table 4.11: Squareness gauge algorithm search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-5-1 and square-5-2.

datum flatness (tolerance /error) / squareness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-11	-11	-11	-10	-10	-10	-10
0.8	-11	-11	-11	-10	-10	-10	-10
0.9	-11	-11	-11	-10	-10	-10	9
1.0	-11	-11	-11	10	10	10	9
1.1	-11	-11	-11	10	10	10	9
1.2	-11	-11	-11	10	10	10	9
1.3	-11	-11	-11	10	10	10	9

Table 4.12: NAG two-phase search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-5-1 and square-5-2.

datum flatness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
squareness (tolerance /error)							
0.7	-9	-9	-9	-9	-9	-9	-9
0.8	-9	-9	-9	-9	-10	-10	-11
0.9	-9	-9	-9	-9	-11	-11	11
1.0	-9	-9	-9	11	10	10	10
1.1	-9	-9	-9	11	10	10	10
1.2	-10	-10	-9	11	10	10	10
1.3	-10	-10	-9	11	10	10	10

Table 4.13: Squareness gauge algorithm search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-6-1 and square-6-2.

datum flatness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
squareness (tolerance /error)							
0.7	-12	-12	-12	-10	-10	-10	-10
0.8	-12	-12	-12	-10	-10	-10	-10
0.9	-12	-12	-12	-10	-10	-10	10
1.0	-12	-12	-12	10	10	10	10
1.1	-12	-12	-11	10	10	10	10
1.2	-12	-12	-11	10	10	10	10
1.3	-11	-11	-10	10	10	10	10

Table 4.14: NAG two-phase search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-6-1 and square-6-2.

datum flatness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
squareness (tolerance /error)							
0.7	-8	-8	-10	-11	-9	-9	-9
0.8	-8	-8	-10	-11	-11	-12	12
0.9	-8	-8	-10	-11	11	11	9
1.0	-8	-8	-10	10	11	9	9
1.1	-9	-9	-9	10	9	9	9
1.2	-9	-9	-9	9	9	9	9
1.3	-9	-9	-9	9	9	9	9

Table 4.15: Squareness gauge algorithm search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-7-1 and square-7-2.

datum flatness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
squareness (tolerance /error)							
0.7	-12	-12	-10	-10	-10	-10	-9
0.8	-12	-12	-10	-10	-10	-10	9
0.9	-12	-12	-10	-10	9	9	9
1.0	-12	-12	-10	10	9	9	9
1.1	-12	-12	-10	9	9	9	9
1.2	-12	-10	-10	9	9	9	9
1.3	-12	-10	-10	9	9	9	9

Table 4.16: NAG two-phase search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data sets square-7-1 and square-7-2.

Chapter 5

Genetic Search Methods for Inspecting Geometric Tolerances

5.1 Introduction

In this chapter, the approach to formulate inspection problems discussed so far is extended to the inspection of mating features. One of the most important tolerance assessment problems is to decide whether two parts will fit together. For example a hole plate with two holes and corresponding plug plate with two corresponding circular plugs. In practice, separated tolerances are specified for each part so that any part satisfying these tolerances will mate. Tolerances of position of the holes and plugs are specified in this case, using the principle of maximum material condition, that is the tolerance dependent on the size of the feature [e. g. BSI, 1990]. During inspection, the main concern is to test whether the features are within tolerance. This inspection problem is addressed here.

The formulation of the equivalent constrained problem involves non-linear constraints in its parameters and therefore, in order to find a feasible point, two possible ways exist: either a linear model is used as an approximation to the problem and linear methods are used for that or non-linear methods are directly applied to the truly non-linear geometrical model.

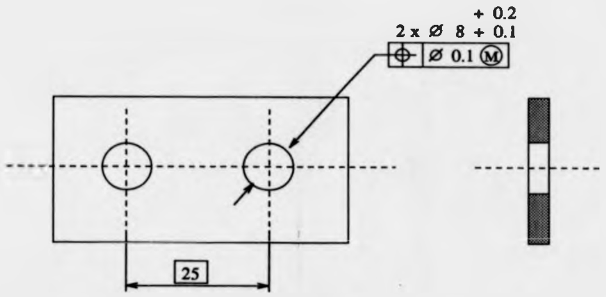


Figure 5.1: Dimension and positional tolerances of two holes.

Although linearising the constraints allows a special algorithm of reasonable efficiency to be applied, as discussed in chapter 3 and 4, the reward for accepting an approximation in the formulation of the constraints is smaller in this case, as the relation between gain in computer efficiency and losses in accuracy is not so favourable.

This chapter discusses the use of some non-linear optimisation methods for assessing geometric tolerance errors as an alternative to linearisation. Specifically, the application of Genetic Search methods to the determination of the feasibility of the inspection problem is explored experimentally as an alternative to formal methods [Carpinetti and Chetwynd, 1993]. Genetic Search methods have been applied successfully in many engineering problems and may have practical relevance to our problem.

5.2 Inspection of Related Circular Features

Consider the inspection of mating features, as is the case of holes on a plate having to match studs on another plate in assembly (the fixed fastener case). Separated tolerances are specified for each part so that any part satisfying these tolerances will mate. Usually, position tolerances are specified using the principle of maximum material condition (MMC) [BSI, 1990], as illustrated in figure 5.1 for a plate with 2 holes.

For the case illustrated, the MMC principle means that the maximum allow-

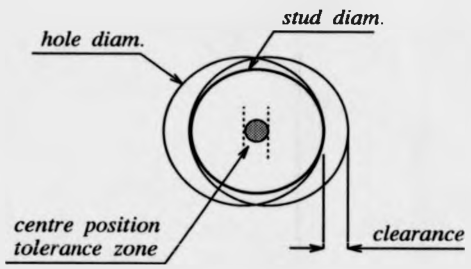


Figure 5.2: Clearance between hole and stud.

able deviation of the theoretically exact position of the hole is calculated based on the minimum clearance between the stud and hole when assembled, which occurs when both stud and hole are at their maximum material limits of size (minimum diameter for the hole and maximum diameter for the stud), figure 5.2. The set of tolerances and nominal dimensions defines, for each mating part, a template or frame, as illustrated in figure 5.3.

Therefore, the inspection of mating features may be done by checking, on each part separately, whether the data points representing the surface of the features may be contained in the frame of figure 5.3. That is, it is checked whether it is possible to fit reference circles with radius within lower and upper limits and respective centres separated from each other by a distance within the limits defined by the nominal distance and position tolerances. Thus, for each part, a constrained problem is defined in which the centre of the reference circles, $(a_k, b_k), k = 1, 2$ (if any of them exists), must be such that

$$\begin{aligned} R^2_{min} \leq (x_i - a_k)^2 + (y_i - b_k)^2 \leq R^2_{max} \\ (L - t_{cp})^2 \leq (a_1 - a_2)^2 + (b_1 - b_2)^2 \leq (L + t_{cp})^2 \end{aligned} \qquad (5.1)$$

for $i = 1, \cdots, N_k, k = 1, 2$, where N_k is the number of data points representing each profile, R_{min} and R_{max} are the radius lower and upper limits, L is the nominal distance between centres and t_{cp} is the centre position tolerance.

It is important to note that if the conventional approach of first finding a best-fit reference to data and then measuring the distance between centres is used (as suggested in standards, e. g. [BSI, 1989]), the benefit brought to inspection

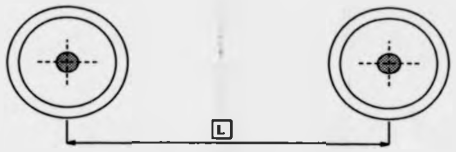


Figure 5.3: Annular template containing data points and reference centres within position tolerance.

and assembly of designing using the principle of maximum material condition is missed.

If the dimension of either the hole or the stud is less than that of MMC, the clearance between them will be larger and hence, even if one of the holes or studs is out of position tolerance, they will fit together. This is the same as saying that if the radius of a hole, measured using a best-fit technique, is found to be larger than the minimum radius but eccentric by more than the position tolerance, it may be possible to fit another circle such that its radius and centre position are within tolerance.

Thus, for the case in consideration, the actual maximum tolerance of position for each feature, say a hole of the hole plate, is defined by the tolerance of position plus the difference between the actual and the MMC diameter of the hole, if the diameter is larger than its minimum. Therefore, even if the centre position of one of the holes is eccentric by more than its maximum tolerance, it may still be possible to fit the two parts together. This is in fact considered by testing the data points for containment within the tolerance template of figure 5.3.

Forbes [1992] also discusses the problem of inspecting mating circular features. The problem is initially formulated as: find a separating surface of parameters $\underline{a}$ such that each of the surfaces of the mating parts lies on one side of the separating surface. This information is encoded by separation constraints of the form

$$D(X;\underline{a}) \geq 0 \geq D(Y;\underline{a}) \tag{5.2}$$

where X and Y represent measurements on the surfaces of the parts. The general part mating problem is then formulated by considering parameter and form (if any) constraints in addition to the constraint in (5.2).

This problem requires one or two frame transformations to relate X and Y to the same frame of reference. It is then pointed out that by setting separate tolerances for the parts (so that any part satisfying the tolerances will mate) the original assessment problem is decomposed into two template matching problems. Forbes' report does not consider methods of solving this problem, although it suggests that in general geometric assessment problems of this type can be solved by using a type of minimax Chebyshev approximation [Osborne and Watson, 1968 and 1969].

Turner [1990] discusses the problem of relative positioning of parts in assembly, particularly the hole and stud problem. By describing each part position in terms of relationships between various features of the part and mating features of its neighbouring parts, it is possible for a solid modelling system to compute the modelling transformations needed to simulate the desired assembly configuration.

In this context, non-interference constraints are specified such that each clearance be non-negative. Optimum alignment is achieved if the two parts are positioned so that the minimum clearance will be maximised over all the mating sites. So the objective is to maximise the minimum clearance. Mathematical programming is then used to establish an optimum position.

This approach is particularly relevant in cases where the nominal part positions are known, and where the aim is to determine the effect of small variations applied to each of the parts on the positions of the other parts. Further details as to the application of this method to tolerancing problems are discussed by Turner and Wozny [1987] and Turner, Wozny and Hoh [1987].

5.3 The Search of Feasible Solutions to Non-Linearly Constrained Problems

There are situations where a linear model is not only convenient in terms of computation effort, but it is a more precise description of the geometrical problem. This is the case for instance of the limaçon approximation for describing circular profiles obtained from conventional, spindle based, roundness measuring instruments (see section 2.4.3). On the other hand, when a coordinate measuring machine is used for sampling a circular feature (of a non-circular component for instance), the data points are given originally by Cartesian ordinate pairs (see section 2.2.1) and the residuals expressed as non-linear functions of the circle parameters (see figures 2.1 and 2.2, chapter 2), as formulated in (5.1).

The set of constraints in (5.1) are distinctly non-linear and therefore, in order to find whether this set of constraints define a feasible point, either the constraints are replaced by those of a linear approximation model of the geometric problem so as to use linear methods or non-linear methods of solution are directly applied. As discussed in section 3.8, in case of circular features, linearisation is possible by transforming the coordinate system from Cartesian to polar after translating the origin of the coordinate system to a point close to the reference centre. The new origin may be, for example, simply the centroid (arithmetic mean) of the points, or, generally better (as discussed in section 3.9), the centre of the least squares best-fit circle to the set of points. This transformation is likely to enable the limaçon reference to be used instead as a good approximation to the circle reference.

Considering the case illustrated in figure 5.1 and formulated in (5.1), linearisation is possible by defining two local polar coordinate systems. The centre position tolerance zone is as before approximated by a series of straight lines defining a regular, even-sided polygon. In this case however, for one of the holes, the polygon is not centred at the origin of the local coordinate system but at the nominal centre position.

However, when such a linear model for two related circular features is compared with the equivalent linear model of a single circular feature, as far as the centre position tolerance is concerned, the inaccuracy of the approximation involved in this linearisation process is doubled, as there are two related features in consideration, and the computer efficiency has dropped to half as two different successive problems will have to be solved. This relation between accuracy of approximation and efficiency of computation becomes even less attractive if more features have to be considered simultaneously, as it would be the case of three, four or more holes on a plate, for instance. Therefore, as the computational gain of linear approximations reduces with problem size, the additional computational effort for handling accurate non-linear constraints may be compensated by the improved accuracy of the model and generality of application.

As has been said, non-linear constraints do not generate plane figures in parameter space and so the search must be over a surface or, worse, if the region is non-convex, over the whole feasible region. This is essentially why it is both more expensive to compute non-linear problems and problematic to guarantee globally optimal solutions.

There is no general agreement on the best approach for solving non-linearly constrained problems and much research is still to be done (see e. g. [Fletcher, 1987] and [Gill, Murray and Wright, 1981]). In general this type of problem is solved either by transforming the constrained problem to an unconstrained problem or by locally linearising the constraints. Typical of the former group are the penalty or barrier methods or the augmented Lagrangian methods [Gill, Murray and Wright, 1981]. These methods however suffer from some computational disadvantages and are not entirely efficient [Fletcher, 1987]. In the latter group, is found the projected Lagrangian methods in which the original problem is transformed into a sequence of linearly constrained subproblems. When the subproblem is a quadratic programme, such methods are called QP-based methods or sequential quadratic programming methods (SQP). The SQP methods are implemented by NAG [NAG, 1990] and further discussed in chapter 6.

An attractive approach to the non-linear inspection problem is the use of feasible direction methods. These methods attempt to maintain feasibility by searching from one feasible point to another along feasible arcs. One of the great advantages of these methods is that it is possible to determine an initial feasible solution (if one exists) by the introduction of artificial variables in a manner quite similar to that used for the simplex calculations [Hadley, 1964]. The application of these methods in geometric tolerance inspection problems is also investigated in chapter 6.

In a totally different approach, non-formal optimisation techniques have gained acceptance over recent years and have been applied successfully in many engineering problems. The following sections introduce the idea of genetic search methods in optimisation and investigate the viability of using genetic search algorithms in inspecting geometric tolerances.

5.4 Genetic Search Methods

5.4.1 Genetic Search in Optimisation

Genetic search algorithms were developed by John Holland at the University of Michigan [Holland, 1975]. These methods have since been adapted for a large number of applications in game theory, induction systems, and other aspects of human cognition, such as pattern recognition and natural language processing. The potential of genetic algorithms as function optimisers has been demonstrated in quite a number of different fields. Current applications include, neural networks, machine learning, structural design, gas pipeline control, electronic filter design and job shop scheduling. The use of genetic search in non-convex, non-linear constrained optimisation problems is reported by Hajela [1990].

Genetic search methods have their philosophical basis in Darwin's theory of survival of the fittest [Sinnott, Dunn and Dobhansky, 1950]. A set of design alternatives representing a population in a given generation (that is at a given iteration) is allowed to reproduce and cross among the alternatives, with bias allocated to the most fit members of the population. Combination of the most desirable characteristics of mating members of the population results in progeny that are more fit than the parents. Hence, if a measure that indicates the fitness of a generation is also the desired goal of a design process, successive generations produce better values of the objective function.

Genetic algorithms are different from typical search methods in three ways [Goldberg, 1989]:

- they work with an encoding of the parameter set rather than with the actual parameters;
- they search from a population of points, not a single point;
- they use probabilistic transition rules, not deterministic rules.

An obvious advantage in this approach is that the search is not based on gradient information, and has, therefore, no requirements on the continuity or convexity of the design space. The corresponding disadvantage is that many iterations may be required as special features of the decision surface are not exploited. Since the inspection problem requires only to find a feasible, and not an optimal, solution they may be relatively efficient for them. There appear to be no clear-cut ways of checking whether this is so other than by experimentation.

5.4.2 Elements of Genetic Search

There are three basic components necessary for the implementation of a genetic algorithm. At the outset, there must be a code or scheme that allows for a bit string representation of possible solutions to the problem. Next, a suitable function must be devised that allows for a ranking or fitness assessment of any solution. The final and most significant component is the development of transformation functions that mimic the biological evolution process when applied to a population of (chromosomal representations of) solutions to the problem.

The design variables are represented by a fixed length string of 0's and 1's that comprise components of a binary coded number. Other representation codes are possible. For the purposes of discussion, let us choose a 10 bit binary number with the maximum and minimum of the design variable corresponding to the maximum and minimum of the binary number. As many binary strings as the number of variables to a defined problem are then placed end to end. This enlarged string represents one solution. A sequence of such strings can be introduced to construct a population of solutions (designs), with each solution having a corresponding fitness factor. This fitness factor is defined according to the value of the objective function to be optimised. In case of constrained optimisation a penalty is associated with any constraint violation in the solution.

Once a population of designs is generated the genetic search can proceed to produce new designs with a higher level of fitness than members of the current population. The first concept in this process is the one of reproduction, which is meant to bias the population to contain more fit members and to eradicate the less fit ones. Let the fitness associated with the i^{th} solution string be denoted by f_i. We can obtain a sum of these fitness values as $f_{sum} = \sum f_i$. The ratio of individual fitness to the fitness sum denotes a ranking of that string in the population. This ratio is used to construct a weighted roulette wheel, with each string occupying an area on the wheel in proportion to this ratio. The wheel is then employed to determine the strings that participate in the reproduction. A random number generator that determines a pseudorandom number between 0 and 1 is invoked to determine the location of the spin on the roulette wheel. Pairs of strings selected in this manner (mating pairs) are then subjected to operations of crossover and mutation to produce pairs of offsprings. Other selection schemes are examined by Brindle [1981].

The second component of genetic search is referred to as crossover, and corresponds to allowing select members of the population to exchange characteristics of the design among themselves. Crossover entails selecting a start and end position on a pair of mating strings at random, and simply exchanging the string of 0's and 1's between these positions on one string with that from the mating string.

Mutation is the third concept in the genetic refinement process, and is one that safeguards the process from a complete premature loss of valuable genetic material during the reproduction and crossover. This corresponds to selecting a few members of the population, determining at random a location on the strings, and switching the 0 or 1 at that location.

The steps described above are repeated for successive generations of the population, until no further improvement in the fitness is attainable. The member in this generation with the highest level of fitness is the nearest to optimal design.

Central to these components are questions related to optimal population sizes, lengths of strings (chromosome), and frequency with which the transformation functions are invoked, that is probabilities of crossover and mutation, p_c and p_m, in the evolution process. Some aspects of these problems are presented in Goldberg [1989]. These issues are best discussed in section 5.5.3, by relating them to the physical problems of concern here.

5.4.3 Genetic Search in Constrained Problems

The previous section describes only the process for unconstrained optimisation and must be modified to account for constraints. At first, it would appear that inequality constraints pose no particular problem. A genetic algorithm generates a sequence of parameters to be tested using the system model, objective function, and the constraints. The model is simply run, the objective function evaluated, and the constraints are checked to see if there is any violation. If not, the parameter set is assigned the fitness value corresponding to the objective function evaluation. If constraints are violated, the solution is infeasible and thus has no fitness. However many practical problems are highly constrained and finding any feasible point is almost as difficult as finding the best. As a result, we usually want to get some information out of infeasible solutions, by degrading their fitness ranking in relation to the degree of constraint violation. Penalty function methods have been used successfully in this case in a number of problems (see e. g. [Hajela, 1990] and [Goldberg, 1987]).

In a penalty method, a constrained optimisation problem is transformed to an unconstrained problem by associating a cost or penalty with all constraint violations. This cost is included in the objective function evaluation.

Let the basic optimisation problem be of the form: find X which minimises $f(X)$ subject to

$$g_i(X) \leq 0\ , \quad i \in I \tag{5.3}$$

This problem is converted into an unconstrained minimisation problem by constructing a function of the form:

$$\phi = \phi(X, r) = f(X) + r\sum_{i=1}^{m}[G_i(X)]^q \tag{5.4}$$

where r is a positive penalty parameter, the exponent q is a non-negative constant, and the bracketed function is defined as

$$[G_i(X)] = max(g_i(X), 0) = \begin{cases} g_i(X) & g_i(X) > 0 \\ 0 & g_i(X) \leq 0 \end{cases} \tag{5.5}$$

The second term on the right of equation (5.4) is called the penalty term. A number of alternatives exist for the penalty function $[G_i(X)]^q$. It is common to square the violation of the constraints for all violated constraints i, that is to set $q = 2$. Under certain conditions, the unconstrained solution converges to the constrained solution as the penalty coefficient r approaches infinity (see e. g. [Fletcher, 1987]).

5.4.4 Reproduction, Crossover and Mutation

The three operators of the genetic search method are implemented in straightforward code segments presented by Goldberg [1987]. Each operator depends on random choice. In the explanation that follows, the existence is assumed of three random choice routines:

- *random*: returns a real pseudorandom number between zero and one (a uniform random variable on the real interval [0, 1]).
- *flip*: returns a boolean true value according to specified probability. That is successive calls generate a random sequence of true and false values with an average of true values proportional to the specified probability.
- *rnd*: returns a random integer value between specified lower and upper limits (a uniform random variable over a subset of adjacent integers).

```
begin
    partsum := 0
    j := 0
    rand := random * sumfitness
    repeat  j := j + 1
        partsum := partsum + fitness[j]
    until  (partsum >= rand)  or  (j := popsize)
    return  j
end
```

Figure 5.4: Algorithm for wheel selection procedure.

Selection of the fittest individuals in a population is implemented as a linear search through a roulette wheel with each individual (solution) occupying an area on the wheel in proportion to the ratio of its fitness value to the sum of the fitness values of all the other individuals (solutions). This is done by the algorithm shown in figure 5.4. The location where the ball has landed after a random spin of the wheel is determine by

$$rand := random * sumfitness$$

where *sumfitness*, the sum of the population fitness, is multiplied by the normalised pseudorandom number generated by *random.* In order to find the solution that occupy the slot where the ball has landed, the repeat-until loop searches through the weighted roulette wheel until the partial sum (*partsum* in figure 5.4) is greater than or equal to the stopping point *rand.* In the algorithm shown in figure 5.4, the function returns the population index value corresponding to the selected individual.

The basic algorithm for the crossover routine is given in figure 5.5. It takes two parent strings and generates two new ones called *child1* and *child2.* The probabilities of crossover and mutation, *pcross* and *pmutation* are passed to the routine along with the string length *lchrom*, a crossover count accumulator *ncross*, and a mutation count accumulator *nmutation.*

At the beginning of the routine, it is determined whether crossover is performed on the current pair of parent chromosomes, that is solutions. A biased

```
begin
    if flip(pcross) then
        jcross := rnd(1 , lchrom - 1)
        ncross := ncross + 1
    else
        jcross := lchrom
    end
{first exchange , 1 to 1 and 2 to 2}
    for j := 1 to jcross do
        child1[j] := mutation(parent1[j])
        child2[j] := mutation(parent2[j])
    end
{second exchange , 1 to 2 and 2 to 1}
    for j := jcross + 1 to lchrom do
        child1[j] := mutation(parent2[j])
        child2[j] := mutation(parent1[j])
    end
end
```

Figure 5.5: Algorithm for crossover procedure.

coin is tossed that comes up heads (true) with probability *pcross.* The coin toss is simulated in the boolean function *flip*, where *flip* in turn calls on the pseudorandom number routine *random.* If a cross is called for, a crossing site is selected between 1 and the last cross site. The crossing site is selected in the function *rnd*, which returns a pseudorandom integer between specified lower and upper limits (between 1 and $lchrom - 1$). If no cross is to be performed, the cross site is selected as *lchrom* (the full string length) so a bit-by-bit mutation will take place, despite the absence of a cross. Finally, the partial exchange of crossover is carried out in the two *for-do* loops at the end of the algorithm. The first *for-do* handles the partial transfer of bits between *parent1* and *child1* and between *parent2* and *child2.* The second *for-do* loop handles the transfer and partial exchange of material between *parent1* and *child2* and between *parent2* and *child1.* In all cases, a bit-by-bit mutation is carried out by the boolean function *mutation.*

Mutation at a point is carried out by an implementation of the algorithm shown in figure 5.6. The function *flip* is used to determine whether or not to change a true to a false (a 1 to a 0) or vice versa. The function *flip* will only come up heads (true) *pmutation* percent of the time as a result of the call to the

```
begin
    mutate := flip(pmutattion)    {flip the biased coin}
    if mutate then
        nmutation := nmutation + 1
        mutation := change_bit_value
    else
        mutation := no_change
    end
end
```

Figure 5.6: Algorithm for mutation procedure.

pseudorandom number generator within *flip* itself. The number of mutations is counted in *nmutation*. It is possible to avoid much random number generation if it is decided when the next mutation should occur rather than calling *flip* each time.

These three algorithms form the main part of the genetic algorithm. The other segments of the algorithm are discussed in the following section.

5.5 Genetic Search Model Implementation and Tests

5.5.1 Objectives

Although software implementations of the genetic algorithm are available, there is no obvious theoretical way of predicting its efficiency in different situations. Hence a Genetic Search model was set up and suitable values for its control parameters were checked experimentally.

In order to evaluate the efficiency of this model under practical conditions, two inspection problems were considered, as follows:

1. inspection of roundness and centre position of a circular feature, formulated

as: find (a, b) and R such that

$$\begin{aligned}(x_i - a)^2 + (y_i - b)^2 &\leq (R + t_r)^2 \\ (x_i - a)^2 + (y_i - b)^2 &\geq R^2 \\ (x_o - a)^2 + (y_o - b)^2 &\leq t_{cp}\end{aligned} \qquad (5.6)$$

for $i = 1, \cdots, N$, where N is the number of data points and (x_o, y_o) the nominal position. This is similar to that examined by local linearisation methods in sections 3.4 and 3.7.

2. template matching: inspection of position and dimension of four circular features (holes or studs), of the same dimensions and tolerances, on a plate forming a square frame. This can be formulated as: find reference centres $(a_k, b_k), k = 1, \cdots, 4$ such that

$$R_{min}^2 \leq (x_i - a_k)^2 + (y_i - b_k)^2 \leq R_{max}^2 \,, \quad i = 1, \cdots, N_k \,, \quad k = 1, \cdots, 4 \qquad (5.7)$$

and

$$\begin{aligned}(L - t_{cp})^2 \leq (a_1 - a_2)^2 + (b_1 - b_2)^2 \leq (L + t_{cp})^2 \\ (L - t_{cp})^2 \leq (a_1 - a_3)^2 + (b_1 - b_3)^2 \leq (L + t_{cp})^2 \\ (L - t_{cp})^2 \leq (a_2 - a_4)^2 + (b_2 - b_4)^2 \leq (L + t_{cp})^2 \\ (L - t_{cp})^2 \leq (a_3 - a_4)^2 + (b_3 - b_4)^2 \leq (L + t_{cp})^2\end{aligned} \qquad (5.8)$$

where N_k are the number of data points representing each circular feature, R_{min} and R_{max} are the radius lower and upper limits, L is the nominal distance between centres and t_{cp} is the centre position tolerance. The references are numbered such that $k = 1, 2$ are the bottom, respectively left and right, and $k = 3, 4$ are the top, respectively left and right ones, as illustrated in figure 5.7. This is an extended version of the problem introduced in section 5.2, for which no local linearisation appears to be satisfactory and the algorithmic choice lies between the method examined here and non-linear optimisation methods such as those explored further in chapter 6.

These problems were implemented by adapting a version of GENESIS, a public domain software implementation of genetic search techniques.

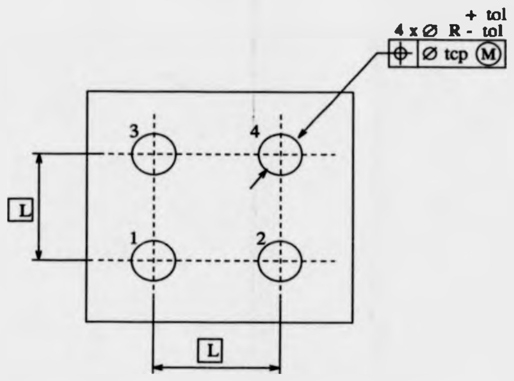


Figure 5.7: Specifications of four circular features on a plate.

In order to do the tests the following sequence was obeyed:

- collect data from a coordinate measuring machine. Simulated data is also used;
- measure the characteristics of the features, that is dimensions, form and location by a Fortran implementation of the sequential quadratic programming (SQP) method (routine E04VDF [NAG, 1990], see sections 6.4 and 6.5.3);
- based on the previous results, define examples of tolerance values so as to simulate tolerance limits from design;
- feed the tolerance information to the genetic search model and perform preliminary tests so as to define best values for a number of control parameters such as population size, crossover and mutation rates and then using such values to perform more definitive tests.

Details about the GENESIS software and the modifications to it, data acquisition and test procedures are described in the next sections.

5.5.2 Genesis Software

Genesis (Genetic search implementation system, version 5.0) is a public domain software for function optimisation developed by Grefenstette [1990]. The system is written in C language. Being a public domain software, it is possible to alter it so as to customise it to any particular need.

Genesis has three levels of representation for the structures it is evolving. The lowest level, or "packed" representation, is used to maximise both space and time efficiency in manipulating structures. In general, this level of representation is transparent to the user. The next level, or "string" representation, represents structures as null-terminated arrays of character variables. This level is provided for users who wish to provide an arbitrary interpretation on the genetic structures, for example, non-numeric concepts. The third level, or "floating point" representation, is the appropriate level for many numeric optimisation problems. In this case, a number of values in a range are automatically translated to binary representation by the programme. The user specifies the floating point representation by interacting with the "setup" programme. For each parameter the user specifies its range in floating point representation and the length of the binary string representing the parameter. So, for example, for a string of 4 bits and a range of floating point values between 20 and 22, the number 20 will be represented by the binary number "0000" and the number 22 by "1111", that is the minimum and maximum binary numbers. In between these two limits, 14 values can be represented, with a step length of 0.125 in floating point representation. If the string length is increased, for the same interval, the number of values that can be represented is increased and therefore the "resolution" or "granularity" of the floating point representation is improved.

By default, the initial population is chosen at random. Alternatively, the initial population may contain heuristically chosen initial points. If the chosen initial points are fewer than the population needs, the remaining structures will be initialised randomly.

One generation comprises the following procedures: selection, crossover, mutation and evaluation. The three main procedures are variations of the basic algorithms described in section 5.4.4. The selection procedure is based on an algorithm by Baker [1987], and is a variation of the wheel roulette procedure described before. In addition to this, the user may opt for the "elitist" selection strategy. The elitist selection strategy stipulates that the best performing structure always survives intact from one generation to the next. In the absence of this strategy, it is possible that the best structure disappears, due to crossover and mutation transformations. The crossover procedure also differs from the algorithm described before, as it chooses randomly two crossover points (instead of one point). The segments between the crossover points are then exchanged, provided that the parents differ somewhere outside of the crossed segment.

The evaluation procedure computes the fitness value of each structure or solution based on the objective function to be optimised. To use GENESIS, the user must write an evaluation procedure, which takes one structure as input and returns a double precision value. In the case of the inspection problems, the objective function is the penalty term of equations (5.4) and (5.5), so that the fitness of a solution is a function of its degree of constraint violation or infeasibility. The functions implemented were based on the algorithm described in figure 5.8. The genetic algorithm searchs through infeasible solutions in an attempt to minimise (minimisation process is the default option) the penalty term. The iteration process terminates when the penalty term is reduced to zero, meaning that a feasible solution has been found. However, in cases where no feasible point exists the penalty term is never reduced to zero and hence the iteration process never terminates. Even though there is no formal proof of infeasibility in such cases, the method may still be of practical use since if no feasible point is found within a specified number of generations, the probability of a false negative by terminating as a failure will be acceptably low.

In order to adapt GENESIS to our problems, a few modifications were introduced. These are:

- read data set and store it in a two-dimensional array variable. Use this variable to construct the set of constraints, as described in (5.6) for the

```
{ N : number of constraints }
{ r : penalty parameter }
{ x : structure (solution) }
{ gi(x) : ith constraint }
{ [gi(x)]^q : penalty function }

G(x) := 0 ;
for i := 1 to N do
    if gi(x) > 0 then
        G (x) := G(x) + [gi(x)]^q ;
    end
end
G(x) := [G(x)]* r ;
return G(x)
```

Figure 5.8: Algorithm for evaluation procedure.

first inspection problem and (5.7) for the second problem;

- input the inspection parameters. Tolerance of roundness, tolerance of position and nominal centre position for the first problem, and tolerance on radius, nominal radius, tolerance of position and nominal distance between features for the second problem;
- set the penalty parameter, r, and the exponent q of the penalty function;
- interrupt the process if the evaluation function returns a null penalty term.

These modifications were made by altering some parts of the original source programme and introducing new pieces of code.

Before starting the search, the "setup" programme is executed, which prompts the user for a number of input parameters relevant to the genetic search itself and how the evaluation is made. All of this information is stored in a file for future use, so it is run only once. Pressing the return key to any prompt gets the default value shown in brackets. The prompts are as follows:

- floating point representation [y]: unless this is declined the user is asked to specify the number of genes (parameters). Each gene takes on a range of floating point values, with a user-defined granularity and output format. The user is asked to specify, for each gene, its maximum and minimum value, the number of values (that is the string length) and the desired output format.

- the number of experiments [1]: this is the number of independent optimisations of the same function.
- the number of trials per experiment [1000]: this is the number of generations multiplied by the population size.
- the population size [50].
- the length of the structures in bits [30]: when the floating point representation is selected, this number is computed automatically from the information collected above (number of values).
- the crossover rate [0.60].
- the mutation rate [0.001].
- the generation gap [1.0]: the generation gap indicates the fraction of the population which is replaced in each generation.
- the scaling window [5]: when minimising a numerical function with a genetic algorithm, it is common to define the fitness value, $f(x)$ of a structure x as $u(x) = f_{max} - f(x)$, where f_{max} is the maximum value that $f(x)$ can assume in the given search space. This transformation guarantees that the value $u(x)$ is positive, regardless of the characteristics of $f(x)$. Often, f_{max} is not available a priori, in which case $u(x)$ is define as $u(x) = f(x_{max}) - f(x)$, where $f(x_{max})$ is the maximum value of any structure evaluated so far. Either definition of $u(x)$ has the unfortunate effect of making good values of x hard to distinguish. For example, suppose $f_{max} = 100$. After several generations, the current population might contain only structures x for which $5 < f(x) < 10$. At this point, no structure in the population has a performance which deviates much from the average. This reduces the selection pressure toward the better structures, and the search stagnates. One solution is to define a new parameter F_{max} with a value of, say, 15, and

rate each structure against this standard. For example, if $f(x_i) = 5$ and $f(x_j) = 10$, then $u(x_i) = F_{max} - f(x_i) = 10$, and $u(x_j) = F_{max} - f(x_j) = 5$; the performance of x_i now appears to be twice as good as the performance of x_j. The scaling window W allows the user to control how often the baseline performance is updated. If $W > 0$ then the system sets F_{max} to the greatest value of $f(x)$ which has occurred in the last W generations. A value of $W = 0$ indicates an infinite window (i. e. $u(x) = f(x_{max}) - f(x)$).

- the seed for the random number generator [123456789].
- the options [cefgl]: GENESIS allows a number of options which control the kinds of output produced, as well as certain strategies employed during the search. Each option is associated with a single character. The default options are: *c*: collect statistics concerning the convergence of the algorithm; *e*: use the "elitist" selection strategy; *f*: use the floating point representation; *g*: use Gray code. A Gray code is sometimes useful in representing integers in genetic algorithms. Gray codes have the property that adjacent integer values differ at exactly one bit position. The use of Gray codes avoid unfortunate effects of "Hamming cliffs" in which adjacent values, say 31 and 32, differ in every position of their fixed point binary representations (01111 and 10000, respectively). This option has no effect unless option *f* is also set; *l*: log activity (starts and restarts) in the "log" file. Other options are available, as the *M* option, for maximisation processes. These are described in the software user guide [Grefenstette, 1990].

In addition to these input parameters, the user is prompted for information on how to output results, which are described in the software's user guide [Grefenstette, 1990] and not repeated here. Also in the user guide is how to install, compile and run the software, under either DOS or Unix operating systems.

The definition of the control parameters described above is discussed in the next section.

5.5.3 GS Control Parameters

A number of the input parameters mentioned in the last section influence the number of generations that is likely to be needed to solve particular types of problems. These are as follows:

- population size: a larger population increases the chance of "good" solutions among the population and therefore reduces the total number of generations. However, it increases the computation time for each generation. There is a region in which an increase in the population leads to a decrease in the number of generations, with a relatively insignificant increase in the computation time. However, beyond that region no significant reduction in the number of generations is obtained, due probably to an increase in the number of repeated solutions in the population.

- structure length: larger strings increase the number of real values that can be represented within a defined interval, which increases the chance of "good" solutions in the population and consequently reduces the total number of generations. However, the length of each variable string is limited by the length of the binary word that a particular computer can handle.

- probabilities of crossover, p_c, and mutation, p_m: the frequency with which the genetic transformations are performed also alters the total number of generations. Studies by DeJong [1975] show that values of p_c and p_m of 0.6-0.8 and 0.01-0.02, respectively, perform adequately for most problems.

- range over which a variable is defined: as mentioned before, when working with floating point representation, each variable is considered over a range of real values. By reducing the range over which a variable is considered, for the same string length, the gap between two consecutive values (resolution or granularity) is narrowed, and consequently the chance of a "good" solution being missed in between two sampled values is reduced, thus reducing the total number of generations to get a feasible solution. The ranges should be chosen around the expected solution. The problem

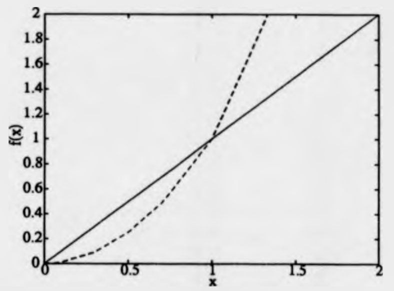


Figure 5.9: Penalty functions $f(x) = x^2$ (–) and $f(x) = x$ (-).

of narrowing too much the range of candidate solutions is that the solution may be missed out of the chosen range. For the problems in consideration, a good estimate of the solution is given by the least squares method. The "width" of the range of candidate solutions should be reduced to a minimum, without further reducing or even eliminating the feasible region by the upper and lower variable bounds so defined.

- penalty function and parameter: with constrained optimisation processes, the penalty function used and the penalty parameter also affects the total number of generations. The penalty parameter has to magnify the penalty term so as to "penalise" the objective of the original problem. Recommended parameters values are from 100 onwards [Hajela, 1990]. Regarding the penalty function, it is common to square the violation of the constraints for all violated constraints i, that is setting $q = 2$ in equation (5.4). However, although this rapidly eliminates from the population of solutions those with high infeasibility, for small violations (less than 1), the resulting contribution to the penalty term of a infeasible solution is less than the amount by which the solution violates any constraint and the violation. This is illustrated in figure 5.9. One alternative is to set $q = 2$ if the constraint violation is greater than one, ($if \;\; g_i(x) > 1$ in figure 5.8), and otherwise, $if \;\; 0 < g_i(x) < 1$, then $q = 1$.

Although recommended values for some of these parameters were available (the default options), tests were performed so as to identify the best values (or range of values) for the population size, probabilities of crossover and mutation and penalty function and parameter. The string length and the range over which the variables were defined were set on theoretical grounds, since there was a clear trend for these for which the efficiency of the genetic search was improved. In pre-fixing these, tests were simplified without compromising its validity. Thus, the string length was set to the maximum that the computer used could operate upon. The way in which the variable ranges were defined and details about the procedures followed for the tests are described in section 5.5.5.

5.5.4 Data Acquisition and Generation

Three circular profiles, originally data-logged for the tests described in section 3.8, were used again for this set of tests. The data points were acquired by a coordinate measuring machine, $LK4$, as described in section 3.8.2. Each set consisted of 30 points evenly but not exactly spread along the circular features. The data sets were numbered as 11, 12 and 13 in section 3.8 and the same numbering will be used here. Linear plots of these profiles, expressed in polar coordinates and shifted to the centroid of the points, are shown in figures 3.19 to 3.21. These data sets are listed in appendix A.2.

In addition to real data, simulated data were generated as well. A roundness profile can be represented by a general periodic wave of the form [Damir, 1979]

$$r(\theta) = A \sin N\theta + r_o \tag{5.9}$$

where N is the number of lobes, A the amplitude and r_o the nominal radius. In order to generate data sets, a function was written in MATLAB [The MathWorks, 1992] to implement equation (5.9). An element of noise was introduced by adding to the signal a random number (uniformly distributed in the interval [0.0, 1.0]) multiplied by a amplitude parameter. This was obtained by using a random generator function, implemented in MATLAB and described in Forsythe, Malcolm and Moler [1977].

The data set for each profile consisted of a two-dimensional array representing

data set	lobes	amplitude	noise amplitude	radius (mm)	$L_x = L_y$ (mm)
s-1	3	0.02	0.01	50	100
s-2	5	0.02	0.002	10	200
s-3	7	0.003	0.003	2	5

Table 5.1: Characteristics of simulated data sets of single circular features.

Cartesian ordinates pairs obtained by the transformation of the polar coordinates from equation (5.9), as

$$\begin{aligned} x[i] &= r(\theta_i)\cos\theta_i + L_x \\ y[i] &= r(\theta_i)\sin\theta_i + L_y \end{aligned} \tag{5.10}$$

where (L_x, L_y) are the Cartesian ordinates of the origin of the polar representation.

For the first inspection problem under consideration, three data sets were generated, each one representing a single circular feature sampled by thirty points around 360°. The magnitude of the figures used for the amplitude and noise magnification were chosen so as to simulate dimensions in millimeters (mm) and errors of roundness due to:

- error of form and some error of roughness (data set s-1);
- mainly error of form (data set s-2);
- small error of form relative to the error of roughness (data set s-3).

Table 5.1 shows the characteristics of the profiles generated. The number of lobes (3, 5 and 7) were chosen on the grounds that these are generally the most common in real profiles.

For the second inspection problem, three sets of four data sets were generated, simulating four circular features on a plate, each sampled by fifteen points around 360°, again with three five and seven lobes. The magnitude of the figures used for the amplitude and noise magnification were again chosen so as to simulate dimensions in millimeters (mm) but in this case to simulate errors of roundness due to, mainly, error of form. Table 5.2 shows the characteristics of the profiles

data set	lobes	amplitude	noise amplitude	radius (mm)	$L_x = L_y$ (mm)	centre distances (mm)
s4h-1	3	0.06	0.005	10	50	100
s4h-2	5	0.1	0.001	20	50	300
s4h-3	7	0.02	0.002	3	5	30

Table 5.2: Characteristics of simulated data sets of four circular features on a plate.

generated for the second inspection problem. The characteristics for each set of four profiles were kept the same. Therefore, the resulting profiles for each set are nearly the same, apart from some variations introduced by random noise. The definition of the same characteristics for the four profiles may be a slight oversimplification but was based on the fact that this would be expected, for example, of a workpiece with circular profiles machined by the same process.

For the purpose of graphical illustration of the sort of profiles generated, linear plots of the profiles with 200 points are shown in figures 5.10 to 5.15. The data sets were saved in ASCII files for further manipulation and are listed in appendix A.3.

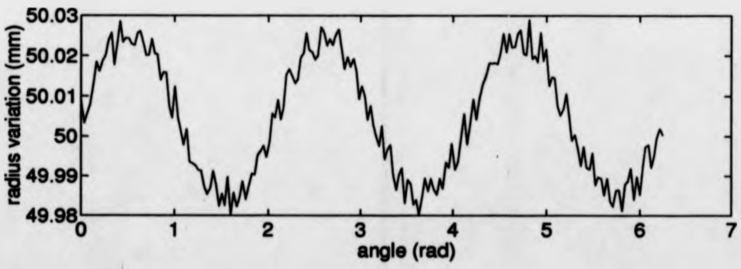


Figure 5.10: Linear plot of simulated profile (top/bottom: air/metal metal): data set s-1.

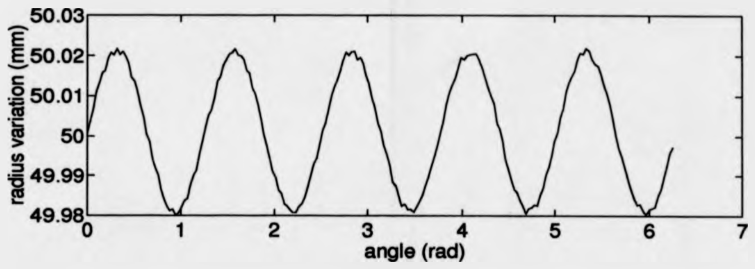


Figure 5.11: Linear plot of simulated profile (top/bottom: air/metal): data set s-2.

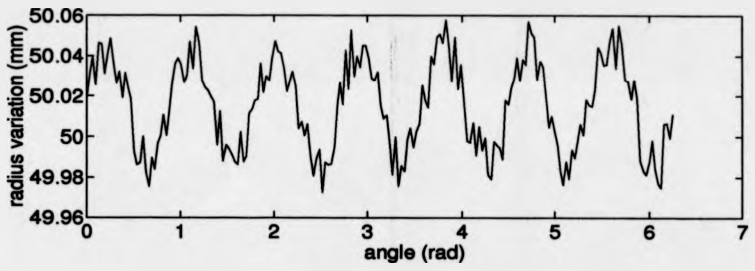


Figure 5.12: Linear plot of simulated profile (top/bottom: air/metal): data set s-3.

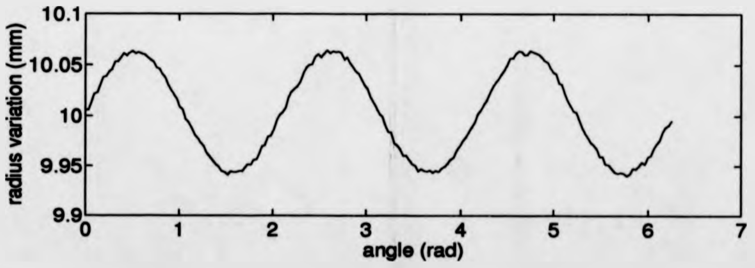


Figure 5.13: Linear plot of simulated profile (top/bottom: air/metal): data set s-4h01.

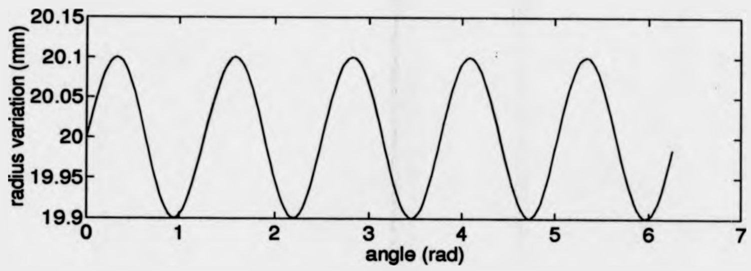


Figure 5.14: Linear plot of simulated profile (top/bottom: air/metal): data set s-4h02.

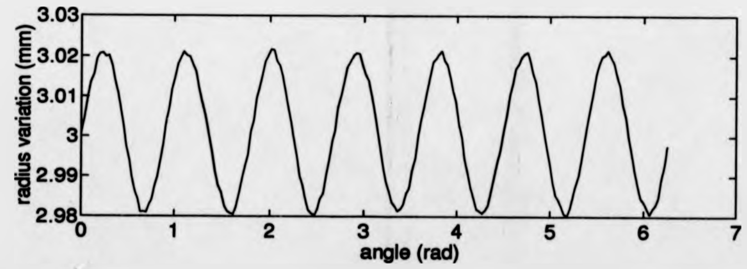


Figure 5.15: Linear plot of simulated profile (top/bottom: air/metal): data set s-4h03.

data set	a	b	R	$(2h)$
s-1	100.0016	99.9990	50.0050	0.04740
s-2	199.9998	199.9999	10.0009	0.0365
s-3	5.0005	5.0001	2.0012	0.0075

Table 5.3: Parameters of MZC reference and out of roundness of roundness of simulated profiles: units in millimeters (mm).

5.5.5 Test Procedures

In order to measure the dimension, location and out of roundness of the circular features, a minimum zone reference (MZC) was used as best-fit criterion. A NAG Fortran implementation of the sequential quadratic programming (SQP) method (E04VDF [NAG, 1990], see sections 6.4 and 6.5.3) was used to minimise the zone parameter h subject to the non-linear constraints

$$\begin{aligned} (x_i - a)^2 + (y_i - b)^2 &\leq (R + h)^2 \\ (x_i - a)^2 + (y_i - b)^2 &\geq (R - h)^2 \end{aligned} \tag{5.11}$$

for $i = 1, \cdots, N$, where (a, b) are the centre of the reference and R its radius.

A piece of Fortran code was written to read the data sets, formulate the constraints, call the function and input the data to it. The parameters of the best fit and the error of roundness for the data sets related to the first inspection problem are shown in tables 5.3 and 5.4, for simulated and real profiles respectively. For the data sets related to the second inspection problem, tables 5.5, 5.6 and 5.7, respectively for each set of four circular profiles, show the parameters of the best fits and the distances between centres. The circular features are numbered in such a way that profiles $1, 2$ are the bottom, respectively left and right, and $3, 4$ are the top, respectively left and right ones, as in figure 5.7.

Tests were performed so as to consider the cases when both errors (that is position and roundness for the first case and position and radius for the second) are less than respective tolerances, or when the position error is greater than its tolerance and the out of roundness or radius (either for the first or second cases)

data set	a	b	R	$(2h)$
11	52.9540	60.3820	13.6855	0.0310
12	152.0440	61.6562	13.5070	0.1947
13	92.6014	62.6240	4.9845	0.0955

Table 5.4: Paremeters of MZC reference and out of roundness of roundness of real profiles: units in millimeters (mm).

data set s-4h01						
profile	a	b	R	h	distance between centres	
					1-	4-
1	50.0098	50.0012	10.0008	0.0580	-	-
2	150.0003	49.9999	10.0027	0.0590	99.9993	100.0011
3	50.0000	149.9977	10.0044	0.0587	99.9965	99.9990
4	150.0012	149.9990	10.0031	0.0580	-	-

Table 5.5: Parameters of MZC references of the four profiles and distances between their centre positions, data set s-4h01: units in millimeters (mm).

data set s-4h02						
profile	a	b	R	h	distance between centres	
					1-	4-
1	50.0003	50.0002	20.0004	0.0870	-	-
2	350.0003	50.0001	20.0005	0.0870	300.0	300.0001
3	50.0000	350.0000	20.0005	0.0870	299.9998	299.9990
4	350.0002	350.0001	20.0005	0.0867	-	-

Table 5.6: Parameters of MZC references of the four profiles and distances between their centre positions, data set s-4h02: units in millimeters (mm).

data set s-4h03						
profile	a	b	R	h	distance between centres	
					1-	4-
1	5.0005	5.0029	3.0012	0.0199	-	-
2	35.0022	4.9993	3.0031	0.0190	30.0017	30.0016
3	5.0033	35.0003	3.0040	0.0195	29.9970	30.0018
4	35.0050	35.0012	3.0056	0.0198	-	-

Table 5.7: Parameters of MZC references of the four profiles and distances between their centre positions, data set s-4h03: units in millimeters (mm).

data set	nominal position		eccentricity of MZC
	x	y	
s-1	100.0016	100.0240	0.025
s-2	199.9998	200.0199	0.02
s-3	5.0005	5.0051	0.005
11	52.9540	60.3970	0.015
12	152.0440	61.7562	0.1
13	92.6014	62.6740	0.05

Table 5.8: Simulated nominal position of profiles and eccentricity of respectives MZC reference centres: units in millimeters (mm).

are within their tolerance limits. The latter was considered so as to realise the point in which the problem becomes infeasible, that is fails inspection. The situation when both errors are greater than respective tolerances was not considered because it is definitely a case of empty feasible region and the iteration process would have to be stopped anyway, as discussed in the next section. Thus, for the first inspection problem, examples of tolerance values of out of roundness were defined following the same strategy used for the tests described in chapter 3. Values of $1.0, 1.1, 1.2, 1.3$ times the out of roundness were used as roundness tolerances. In order to define examples of centre position tolerance, a simulated error of position was defined by fixing the nominal centre position of the feature such that the measured centre position of the MZC was eccentric by an amount of the same magnitude of the roundness error, again simulating dimensions in millimeters. The nominal centre position for each data set and eccentricity of respectives MZC references are given in table 5.8. Following this, values in the region 0.5 to 1.3 times twice the eccentricity of the MZC centre position (since the tolerance value is a diametrical zone) were used as centre position tolerances.

For the second inspection problem, the nominal radius (R_{nom}) was defined by considering the features as holes and, therefore defining the lower limit as the maximum material condition (MMC) dimension. Thus, for each simulated feature, R_{nom} was defined so as to be larger than $R_{min} = min(R_k - h_k)$ ($k = 1, \cdots, 4$), according to the MZC references given in tables 5.5 to 5.7. The values

data set	R_{min}	R_{nom}	$R_{nom} - R_{min}$	L_{min}	t	L_{nom}
s-4h01	9.9428	10.0928	0.15	99.9965	0.2	100.1965
s-4h02	19.9134	20.0634	0.15	299.9998	0.2	300.1998
s-4h03	2.9813	3.0313	0.05	29.9970	0.05	30.047

Table 5.9: Nominal radius and distance between centres simulating four holes on a plate; units in millimeters (mm).

for R_{nom} were computed so as to be larger than R_{min} by an amount of the order of 10^{-2} to 10^{-1}, simulating dimensions in millimeters. The values of R_{min} and R_{nom} for each data set are given in table 5.9. Following this, values of $1.0, 1.1, 1.2, 1.3$ times $R_{nom} - R_{min}$ were defined as the tolerance on radius, where in this case the tolerance is indicated as $R_{nom} \pm tolerance$. Thus, in the limiting case, when the tolerance is $\pm(R_{nom} - R_{min})$, the MMC dimension will be the minimum radius, amongst the four features, defined by the respective MZC references. The nominal distance between centres was defined by selecting the minimum distance between MZC centres of any two related features,L_{min}, and adding to it a value of about the same magnitude as the tolerance on radius (simulating dimensions in millimeters). Thus, $L_{nom} = L_{min} + t$, where t is the value added. The values for t and L_{nom} are given in table 5.9. Following this, values in the region of 0.5 to 1.3 times t were used as the position tolerance, such that when the tolerance is t, the distance between the MZC centres for at least one of the combinations will be on the limit, that is on the edge of the tolerance zone.

Preliminary tests were carried out in order to set best values for population size, crossover and mutation rates and penalty function and parameter. These tested consisted of setting values for the genetic search parameters and evaluating the variation in performance of the algorithm to get a feasible solution to the profiles and tolerances fed to it. Only tolerance of roundness of single circular profiles were checked at this stage, hence simplifying the inspection problem to:

find parameters (a, b) and R such that

$$\begin{aligned} (x_i - a)^2 + (y_i - b)^2 &\leq (R + t_r)^2 \\ (x_i - a)^2 + (y_i - b)^2 &\geq R^2 \end{aligned} \tag{5.12}$$

for $i = 1, \cdots, N$, where N is the number of data points and t_r is the roundness tolerance.

For each data set, the roundness tolerance value was defined such that the ratio of tolerance to ratio was equal to 1.1. Four data sets representing single circular profiles were used, two being of real profiles (data sets 11 and 12) and the other simulated simulated ones (data sets s-2 and s-3).

Regarding the genetic search model, as mentioned in section 5.5.3, the length of the binary strings representing each variable was fixed to its maximum for the computer used, that is 31 bits. The range of real values, from which candidate solutions were translated to the binary notation of the genetic search, was defined with width equal to twice the roundness tolerance value such that:

- for the radius variable, R, the least squares radius value lies on upper limit of the range;
- for the centre position, (a, b), the least squares values of the centre position lie centrally between the lower and upper limits.

For the radius variable, the least squares value is shifted to the upper limit of the range because the inspection problem (constraints in (5.12)) uses an inscribing reference, and for this reason any feasible reference will have its radius less than the least squares one. The intention in defining the ranges proportional to the roundness tolerance value was to reduce them to a "safe" minimum, without, by adding such constraints, eliminating the feasible region of the inspection problem. The least squares values for the variables were used on the grounds that any feasible solution ought to lie "close" to the least squares solution, which may itself be a feasible solution. The least squares solution was computed by running a direct implementation of the algorithm proposed by Forbes [1989] (see section 2.5.3), written in C language, single precision floating point (the main fragment of its source code is reproduced in appendix B). In case a position tolerance is also specified, the range for the centre position variables could be set equal to the tolerance value such that the coordinates of the nominal centre position lie centrally in between the upper and lower limits. These ranges define a square region around the circular region defined by the position tolerance, therefore ensuring that the feasible region is not further constrained.

Having defined that, the best values for the population size and rates of crossover and mutation were checked by fixing two of them in the range of recommended values and running the process for different values for the other, varying it around the range of recommended values. The exponent of the penalty function was set to one ($q = 1$) and the penalty parameter to $r = 100$ (set in the evaluation function, see figure 5.8). These values were checked in latter tests. Therefore, the following procedure was undertaken:

- set the crossover and mutation rates to $p_c = 0.8$ and $p_m = 0.01$ and run the process for population sizes from 50 to 100 in intervals of 10;
- set the population size to 50, the mutation rate to $p_m = 0.01$ and run the process for crossover rates from 0.55 to 0.85 in intervals of 0.05;
- set the population size to 50, the crossover rate to $p_c = 0.8$ and run the process for mutation rates from 0.001 to 0.021 in intervals of 0.004.

These parameters and ranges were defined by running the "setup" programme before starting the search process. The other parameters were set as follows:

- the generation gap equal to 1.0;
- the scaling window equal to 2;
- the seed for the random number generator from 1 to 9;

- options "efg", for elitist selection (e), floating point representation (f) and gray code (g).

The setup programme saves the parameter values in a file called *in.ext* and the variable ranges in a file called *template.ext*, where *ext* is a user defined extension. At the start of a search process, this information is retrieved so as to set up the genetic algorithm. Thus, for each data set under test, the setup programme was run once, in the beginning of the experiments, so as to set initial values for the parameters and ranges. Any further modification was done by editing either the *in.** or the *template.** files. The results of this set of tests are presented in figures 5.18, 5.19 and 5.20 and discussed in the next section.

After running these tests, another set of tests was run, this time altering the exponent of the penalty function and penalty parameter. The population size and crossover and mutation rates were set to respectively 50, 0.8 and 0.01, and the exponent of the penalty function and penalty parameter were set alternately as follows:

- to $q = 2$ when the violation is greater than one ; otherwise, when the violation is less than one, $q = 1$; set the rest as before;
- to $q = 1$ and the rest as before;
- set values of the penalty parameter, r, equal to $10, 50, 100, 300$ and 500; set $q = 1$ and the rest as before.

The evaluation function was altered, as indicated in figure 5.16, so as to cope with the change of value of the exponent of the penalty function. The process was run again and its efficiency recorded. Table 5.10 presents the results of this test.

One additional test was performed in order to demonstrate and quantify the importance of defining the range of real values for each variable as narrowly as

```
{ N : number of constraints }
{ r : penalty parameter }
{ x : structure (solution) }
{ gi(x) : ith constraint }
{ [gi(x)]^q : penalty function }

G(x) := 0 ;
for i := 1 to N do
    if gi(x) > 0 then
        if gi(x) > 1.0 then
            q := 2 ;
        else
            q := 1 ;
        end
        G (x) := G(x) + [gi(x)]^q ;
    end
end
G(x) := [G(x)] * r ;
return G(x)
```

Figure 5.16: Evaluation procedure with floating penalty function.

possible. To do this, the genetic search was run for the same inspection problem and same data sets as before, using the default values for all the parameters but different ranges for the variables, namely: 1, 2, 3 and 4 times the roundness tolerance value. The ranges were placed as before, that is having the least squares radius as the upper limit and the least squares centre position value central in between the limits. The variation on the number of generations was recorded and it is presented in figure 5.21.

Having done these sets of preliminary tests, the best values for such parameters (or region of values) were defined, as indicated in table 5.11 and discussed in the next section. Based on the results of these tests, suitable parameter values were chosen (as indicated in table 5.11) to set up a genetic search model which was used to perform more definitive tests for the two inspection problems in consideration and for the data sets available.

The same procedure was followed, that is:

- run the setup programme to input the genetic search parameters and ranges of real values for each variable, for each data set. The other parameters, apart from the ones given in table 5.11, were set as before. The range for each variable was set as described below;

- run the process for different tolerance values and record its efficiency.

Examples of tolerance values were chosen as described in the beginning of this section. The specific values for the different ratios of centre position error to tolerance are given in tables 5.13 to 5.18 for the first problem and tables 5.19 to 5.21 for the second.

Regarding the ranges of real values, for the first inspection problem, and for the variables corresponding to the reference centre position coordinates, the ranges were set equal to the centre position tolerance value, such that the nominal values lay centrally in between the upper and lower limits. For the variable corresponding to the reference radius, the range was set equal to twice the roundness tolerance value such that the ratio of error to tolerance was equal to 1.0. So for the largest example of roundness tolerance, this range was still adequate. For the second inspection problem, the ranges for the centre position variables were defined as equal to the tolerance on radius, since the amount by which a feasible reference can be shifted from a best fit reference is proportional to the tolerance on radius and will never exceed it. Considering that there is no nominal centre position but only relative position, the ranges were placed such that the least squares value lay centrally in between the limits. Again, the least squares values were chosen on the grounds that it ought to be close to any acceptable solution.

Furthermore, given that infeasibility can never be proved by this approach, it was chosen to allocate a failure condition if some number of iterations does not yield a feasible solution. As the feasible region shrinks towards a point, there is an increasing probability that the search will need an increasing number of iterations to find a feasible solution. In the limiting case, when the error equals the tolerance value, it may happen that, for some cases, the number of generations to find the feasible point exceeds the failure condition, leading to a false negative

```
{ N : number of constraints }
{ r : penalty parameter }
{ x : structure (solution) }
{ gi(x) : ith constraint }
{ z : buffer zone }
{ [gi(x)]^q : penalty function }

G(x) := 0 ;
for i := 1 to N do
   if gi(x) > z then
      if gi(x) > 1.0 then
         q := 2 ;
      else
         q := 1 ;
      end
      G (x) := G(x) + [gi(x)]^q ;
   end
end
G(x) := [G(x)] * r ;
return G(x)
```

Figure 5.17: Evaluation procedure with floating penalty function and buffer zone.

result. Therefore, ways of reducing the number of generations at the limiting case were sought.

One alternative is to define a "buffer zone" around the feasible region such that any solution violating any constraint by no more than the defined value for the zone is an acceptable solution. Although this is an approximation to the true feasible region, as long as the zone so defined is much smaller than the geometric tolerances involved, this approximation is not likely to bring any practical harm in terms of accuracy. Alternatively, making the geometric tolerance zones a little smaller than those specified would compensate for worries about false positive results. This procedure was adopted by altering the evaluation function, as indicated in figure 5.17. Therefore, the evaluation functions implemented for the two inspection cases incorporate the floating penalty function and the buffer zone, as described in figure 5.17 (the source code of these functions are reproduced in appendix B). The buffer zone was set by inputing its value through the main function.

For the first inspection problem, the genetic search was run twice, with and without a buffer zone. For the second inspection problem however, the genetic search was run only with a buffer zone defined. For both cases the zone was of $1.0\text{x}10^{-5}$ mm (that is when the data set is given in millimeters), as shown in the next section.

The tolerance values for which the problems become infeasible were confirmed by running the NAG Fortran implementation of the sequential quadratic programming (SQP) method (routine E04VDF [NAG, 1990], see sections 6.4 and 6.5.3), previously used for calculating the MZC references. Two pieces of Fortran code were written to call the function and input the parameters of the inspection problems to it.

Tests were run on a SUN 4/330, with 48 Mbytes of RAM (Random Access Memory) and speed of 16 MIPS (Million of Instructions Per Second). For all cases tested, the efficiency was measured in terms of execution time and number of iterations. The execution time was estimated by running the programme under the "time" Unix command, which returns the CPU user time.

5.6 Results and Discussion

Figures 5.18 to 5.20 show the number of generations that the genetic search took to yield a feasible solution for different combinations of population size, crossover and mutation rates. Although the tests were performed for a small number of data sets, the results seemed quite consistent for the data sets tested and showed in most cases how the genetic search can be improved. For the profiles tested, there was a clear indication that the genetic search was improved when the population size was of about 60, and performed relatively well for a population size in the range of the range of 60 to 80. Variation of the crossover rate showed that a value of $p_c = 0.75$ improved the genetic search for three of the data sets tested, while for the other one a slightly lower value of $p_c = 0.65$ to 0.7 performed better. However, as far as the mutation rates is concerned, no clear trend could be observed when varying it in the region of recommended values. For data sets s-1 and 12, two minimum points are defined, around 0.001 and 0.013. For data set 11, the minimum seems to be around 0.001, while for data set s-2, it seems to be around 0.017. A rate of 0.001, the default value used by GENESIS, seems to be a reasonable value for all the data sets tested.

Table 5.10 shows the number of generations required to yield a feasible solution for two different penalty functions. The use of a piecewise penalty function showed that the genetic search performs better in this case than keeping the exponent q of the penalty function equal to one. Also, tests made for different penalty parameters ($r = 10, 50, 100, 300$ and 500) showed no variation in the number of generations. This shows that variation of the penalty parameter towards infinity is important only when searching for optimal solutions and not just for feasible ones.

Figure 5.21 shows that the number of generations to yield a feasible solution increases as the width of the range of real values used is increased. This can also be realized in figures 5.18 to 5.20 by comparing the curve of data set 12 with the others. The range used for data set 12 was between four and six times larger than was used for the other data sets. It also shows, indirectly, that the same effect will happen if the length of the binary strings is reduced. Therefore, it confirms the importance of keeping such ranges as small as possible. For the single feature problem, the range for the radius variable was set equal to twice the roundness error. This was done so as to make sure that for all examples of tolerance values the feasible region was not further constrained. However, in practice, the width of the range may well be set equal to the roundness tolerance value, with the least squares value as the upper limit, since the distance from the least squares radius to the reference radius has to be less than the tolerance zone. The ranges for the centre position parameters were set exactly around the position tolerance and so they are as narrow as they can be without interfering with the feasible region. For the second inspection problem, the ranges for the centre position variables were defined as equal to the tolerance on radius. These ranges can also be reduced if it is considered that the amount by which a best fit reference can be shifted is no more than the smaller distance of the minimum zone circles to either the upper or lower limits defined by the tolerance on radius. This distance can be estimated by using the least squares reference and the width of the ranges can be made proportional to that.

Based on the results obtained, table 5.11 presents ranges of suitable values for such parameters, for the type of problem under consideration. The values of the parameters used for the subsequent tests are also given.

In order to evaluate the efficiency of the genetic search, the number of iterations to get a feasible solution for different tolerance zone combinations was plotted as shown in figures from 5.22 to 5.30 and tabulated as shown in tables 5.13 to 5.21. The figures and tables for each data set contains essentially the same information, however, while the tables give the exact figure used for the ratio of centre position error to tolerance and correspondent number of generations, the graphs emphasise the trend towards an increased number of generations required to get a feasible solution as the problem approaches infeasibility. For the first inspection problem, plots A, B, C and D represent cross sections of a three-dimensional graph having ratio of roundness error to tolerance and ratio of centre position error to tolerance on its horizontal axes. For the second inspection problem the ratio of roundness error to tolerance is replaced by the ratio of the difference between the MMC radius defined by the MZC reference and the nominal radius to the tolerance on radius, that is ($R_{nom} - R_{min}$) (using the notation of table 5.9) over tolerance on radius. The vertical dashed lines indicate the point at which the problem becomes infeasible. However, they represent an approximate indication, since that the SQP method also presents some numerical problems when the problem approaches infeasibility (as discussed in the next chapter).

Graphs B to D of the figures related to the first inspection problem (figures 5.22 to 5.27) confirm what has already been discussed in chapter 3: it may be possible to get a pass inspection of a feature whose best fit is eccentric by more than its tolerance allows, provided that the roundness tolerance is sufficiently larger than its actual error. The same effect can be seen again from graphs B to D of the figures related to the second inspection problem (figures 5.28 to 5.30): when the actual diameter is larger than its MMC size, the combination of features can be acceptable in terms of interchangeability even though the eccentricity of some of the features (when measured using a best-fit criterion) is larger than its corresponding tolerance. Consequently, this is a clear advantage of this approach when compared with the standard method of fitting best-fit geometric elements to data and measuring their characteristics.

For the graphs relative to the first inspection problem (figures 5.22 to 5.27), the dashed curves indicate the number of generations when a buffer zone of 1.0×10^{-5} mm is used (i. e. about 0.1 % of the tolerance zone), whereas the solid curves indicate the number of generations with no tolerance zone defined. For the second problem, the curves plotted are for a buffer zone of 1.0×10^{-5} mm.

The centre position tolerance values were chosen such that tests were made around a region "close" to the limit between feasible and infeasible regions. For each data set, and for each plot, the ratio of centre position error to tolerance "closest" to the limit represent the maximum ratio for which the genetic search could find a solution within a certain number of generations. In general, the distance from the test limit to the estimated limit was of the order of 2 % of the estimated limit for the first inspection case and up to 5 % for the second. The general pattern of results is consistent over all data sets tested. For the first inspection problem (figures 5.22 to 5.27), the results show that in the limit tested, for most cases, a feasible solution is found within about 60 to 80 generations. Data sets 12 and 13, as expected, required more generations to get a feasible solution, as the width of the ranges used in these cases were considerably larger than for the other data sets. Data set 11 also required a few more generations than data sets s-1, s-2 and s-3, perhaps indicating that the simulated data sets "behave" better than the real ones. For the second inspection case, in the tested limit, a solution is found in most cases within about 100 to 150 generations. Although the number of generations in the limit for data set s-4h03 is not fewer than for the others, as might be expected (since the chosen ranges are narrower than for the other two simulated data sets), the points at which the tests were made for this case were closer to the feasibility limit than for the other two data sets.

Table 5.12 presents, for some data sets and for some tolerance combinations, the number of generations and computation time to get a feasible solution. The number of generations for cases of easy inspection, either for the first or second problem, are well bellow 50, representing a computation time of the order of 1 and 3 seconds for the first and second problems respectively. For the tested limits, the computation time goes up to about 4 and 15 seconds for the first and second problems respectively. For the second problem, although it has as many as three times the number of parameters and as many as twice the number of constraints, the number of generations to get a feasible solution does not increase proportionally. Considering that the speed of execution of the computer used is affected by variable time-share overheads, this execution speed can be considered slightly slower than that of a late generation microcomputer dedicated to a measuring machine, for example a 486, 33 MHz micro-computer.

However, when the error becomes closer to the tolerance value, that is, as the feasible region shrinks, the number of generations tends to increase exponentially. In the limiting case, when the feasible region is a point in space, the number of generations to find this point is unaffordably high. This can be overcome by using a buffer zone on the boundaries of the feasible region, such that the feasible region is effectively enlarged by a certain amount. For the first inspection problem, it can be seen that when a buffer zone of $1.0\text{x}10^{-5}$ mm is used, the number of generations for the tested limits go down to about 40 generations. For the second problem the number of generations plotted already considers a buffer zone of $1.0\text{x}10^{-5}$ mm, and therefore, in order to further reduce the number of generations at the limiting cases, a larger zone would have to be used. The approximation introduced by defining a buffer zone around the feasible region leads to a degree of uncertainty about whether an inspection that gave a positive result really meant that the part was acceptable or not. In practice, introducing a buffer zone implies a slight widening of the geometric tolerance zone. Typically the increase might be one part in a thousand, so the error introduced by this method will be no more than 0.1 %. Provided the buffer is small compared to the tolerance band the likelihood of falsely accepting bad components will be tolerably low. Of course, the buffer zone could be taken off the tolerance, with a slight possibility then of false negative results in marginal cases.

When there is no feasible solution, the search would continue for ever and therefore it has eventually to be stopped artificially. However, it is in fact possible to predict the cases of clear rejection by monitoring the rate at which the objective or fitness function of the problem (as in equation 5.4) changes, that is by monitoring the evolution towards feasibility of the best candidate solutions from generation to generation. This is shown in figures 5.31 to 5.34. In each graph, the curves show the evolution of the best solutions towards feasibility for cases of: clear acceptance (dashed curve); close to the limit of feasibility (dash/dot and dotted curves, respectively above and below the limit), and clear rejection (solid curves). The values used for the ratios of centre position error to tolerance are indicated in the captions. It can be seen that after few tens of generations, it is possible to identify the bad components, that is the cases of clear rejection. As the range of values that the fitness function assumes changes for different features, this might be normalised by the maximum value that the function assumes, that is its initial value.

Therefore, a strategy for inspecting using this method might involve the definition, during a setup phase, of two threshold points, such that:

- if the normalised value of the fitness function for the best solution has not reached an expected value after a threshold on the number of iterations or computation time, it is possible to say with high confidence that no feasible point exists.

- if it is not possible to say at an early stage whether the problem is infeasible or not just by monitoring the evolution of the fitness function, then the process is stopped after reaching a second threshold point on the number of generations or computation time. The higher the threshold, the smaller is the chance of rejecting marginally good components.

However, considering that the maximum number of generations is limited by the computation time acceptable for the inspection cycle, it may happen that the process has to be stopped before such marginally good components can be identified. For example, for the first inspection case, for centre position errors of about .98 of their tolerance limits, the number of generations to get a feasible solution was in the region of 60 to 80, which represented a computation time of the order of 3 to 5 seconds on the computer used. In these cases, the number of generations was reduced if a buffer zone was defined around the feasible region. Thus, if the process had to stopped at that stage, geometric errors under their tolerance limits by less than those figures quoted above would result in false negative reports. Therefore, although it is not possible to be categoric about the best choice of threshold on the basis of the tests shown here, since that efficiency of genetic search methods also depends on data set, there needs to be a trade off between the maximum computation time and the chance of false negative reports. Further discussion and conclusions about this method are deferred until the last chapter, after examining other non-linear techniques, which is done in the next chapter.

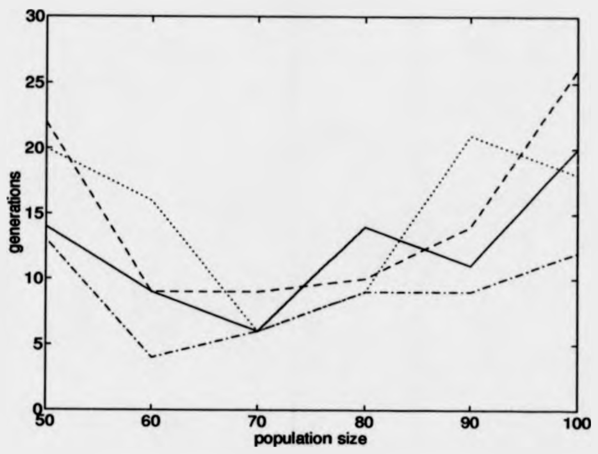


Figure 5.18: Number of generations to get a feasible solution for different population sizes:$p_c = 0.8$, $p_m = 0.01$, $q = 1$ and $r = 100$. Data sets: s-1 (-.), s-2 (..), dt11 (-) and dt12 (- -).

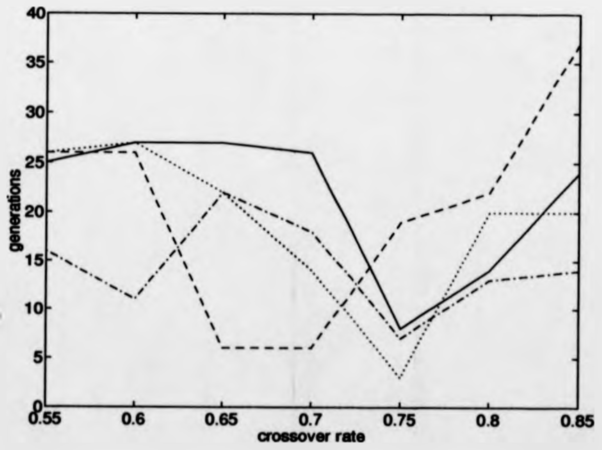


Figure 5.19: Number of generations to get a feasible solution for different crossover rates: population size of 50, $p_m = 0.01$, $q = 1$ and $r = 100$. Data sets: s-1 (-.), s-2 (..), dt11 (-) and dt12 (- -).

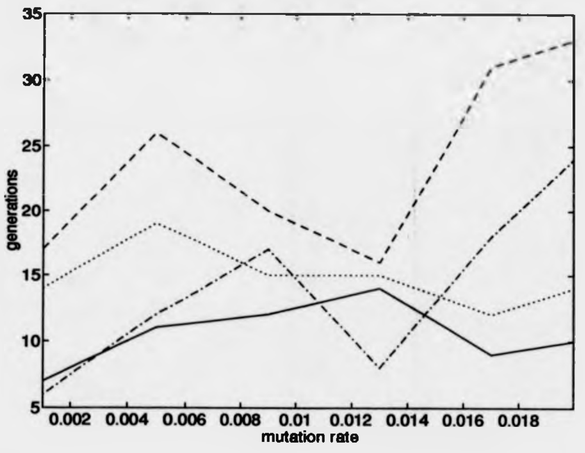


Figure 5.20: Number of generations to get a feasible solution for different mutation rates: population size of 50, $p_c = 0.8$, $q = 1$ and $r = 100$. Data sets: s-1 (-.), s-2 (..), dt11 (-) and dt12 (- -).

data	penalty function q	
set	1 or 2	1
11	5	14
12	20	22
s-1	13	13
s-2	13	20

Table 5.10: Number of generations to get a feasible solution for different values of the exponent q of the penalty function: population size of 50 and $p_c = 0.8$, $p_m = 0.01$ and $r = 100$.

parameter	recommended values	value set
population size	60 - 80	60
crossover rate	0.65 - 0.75	0.75
mutation rate	-	0.001
exponent q	$if g_i > 1.0$ $q = 2$, $else$ $q = 1$	as recommended
penalty par. r	from 10	100

Table 5.11: Recommended range of values and values set for genetic search control parameters.

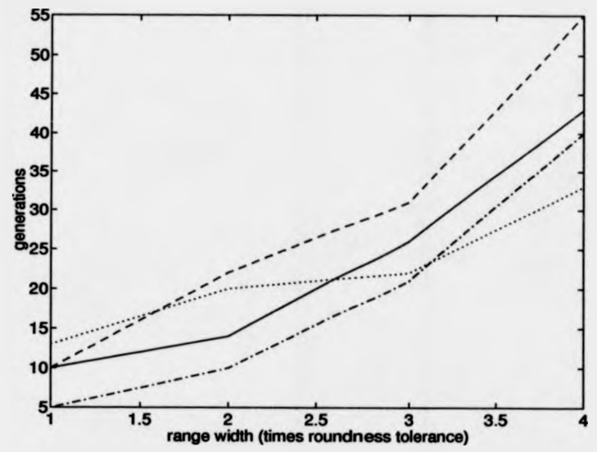


Figure 5.21: Number of generations to get a feasible solution for different widths of the range of real values for each variable, for data sets: s-1 (-.), s-2 (..), dt11 (-) and dt12 (- -).

data set	gen	comput. time (sec)	data set	gen	comput. time (sec)
s-1	23	1.3	12	85	4.9
s-1	72	4.2	13	20	1.6
s-2	51	2.6	13	97	5.2
s-2	17	0.9	s-4h01	17	2.7
s-3	60	3.5	s-4h01	71	10.5
s-3	35	2.0	s-4h02	10	1.8
11	18	1.0	s-4h02	106	15.8
11	108	5.3	s-4h03	30	4.6
12	25	1.7	s-4h03	116	16.9

Table 5.12: Computation time of the genetic search, for some examples of tolerance combinations for different data sets.

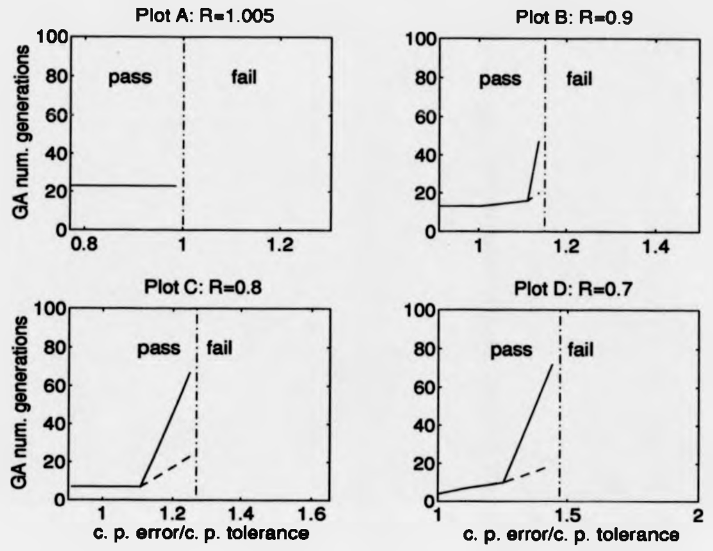


Figure 5.22: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance and roundness error to tolerance (R); without buffer zone (-), and with a buffer zone of 1.0x10^{-5} mm (- -): data set s-1.

$R = 1.005$		$R = 0.9$		$R = 0.8$		$R = 0.7$	
C	gen.	C	gen.	C	gen.	C	gen.
0.77	23	0.91	13	0.91	7	1.0	4
0.83	23	1.0	13	1.0	7	1.11	7
0.91	23	1.11	16	1.11	7	1.25	10
0.985	23	1.136	47	1.25	67	1.44	72

Table 5.13: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance (C) and roundness error to tolerance (R), without buffer zone: data set s-1.

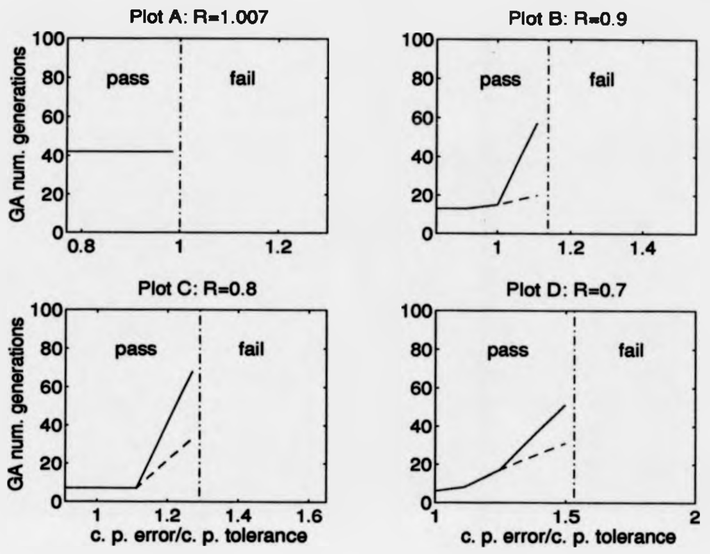


Figure 5.23: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance and roundness error to tolerance (R); without buffer zone (-), and with a buffer a zone of 1.0x10^{-5} mm (- -): data set s-2.

$R = 1.007$		$R = 0.9$		$R = 0.8$		$R = 0.7$	
C	gen.	C	gen.	C	gen.	C	gen.
0.77	42	0.83	13	0.91	7	1.0	6
0.83	42	0.91	13	1.0	7	1.11	8
0.91	42	1.0	15	1.11	7	1.25	17
0.985	42	1.11	57	1.27	68	1.495	51

Table 5.14: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance (C) and roundness error to tolerance (R), without buffer zone: data set s-2.

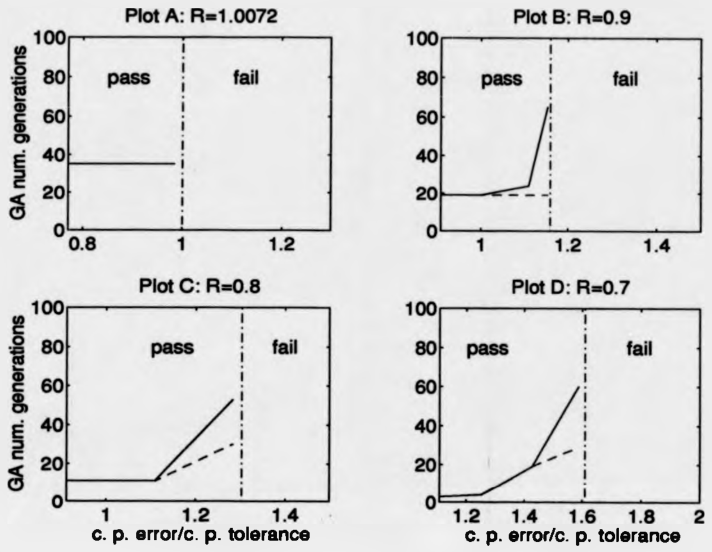


Figure 5.24: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance and roundness error to tolerance (R); without buffer zone (-), and with a buffer a zone of $1.0\text{x}10^{-5}$ mm (- -): data set s-3.

$R = 1.0072$		$R = 0.9$		$R = 0.8$		$R = 0.7$	
C	gen.	C	gen.	C	gen.	C	gen.
0.77	35	0.91	19	0.91	11	1.11	3
0.83	35	1.0	19	1.0	11	1.25	4
0.91	35	1.11	24	1.11	11	1.428	19
0.985	35	1.154	55	1.28	53	1.587	60

Table 5.15: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance (C) and roundness error to tolerance (R), without buffer zone: data set s-3.

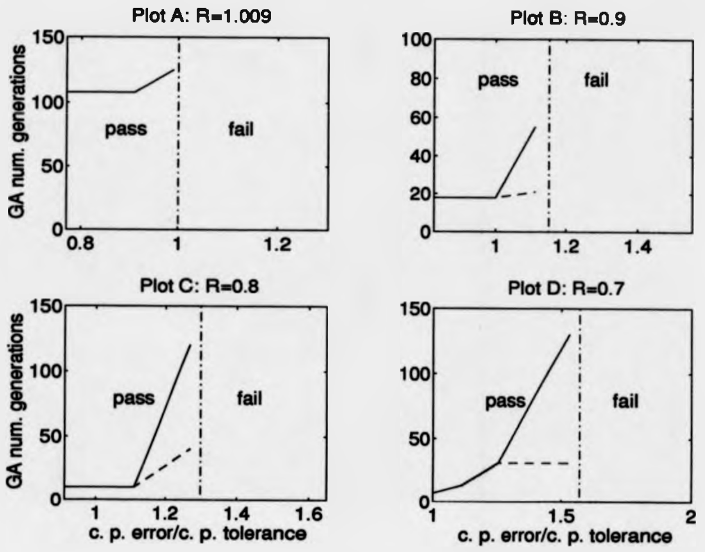


Figure 5.25: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance and roundness error to tolerance (R); without buffer zone (-), and with a buffer a zone of 1.0x10^{-5} mm (- -): data set 11.

$R = 1.009$		$R = 0.9$		$R = 0.8$		$R = 0.7$	
C	gen.	C	gen.	C	gen.	C	gen.
0.77	108	0.83	18	1.0	10	1.11	7
0.83	108	0.91	18	1.11	10	1.25	13
0.91	108	1.0	18	1.25	11	1.428	53
0.96	125	1.1145	55	1.31	46	1.53	74

Table 5.16: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance (C) and roundness error to tolerance (R), without buffer zone: data set dt11.

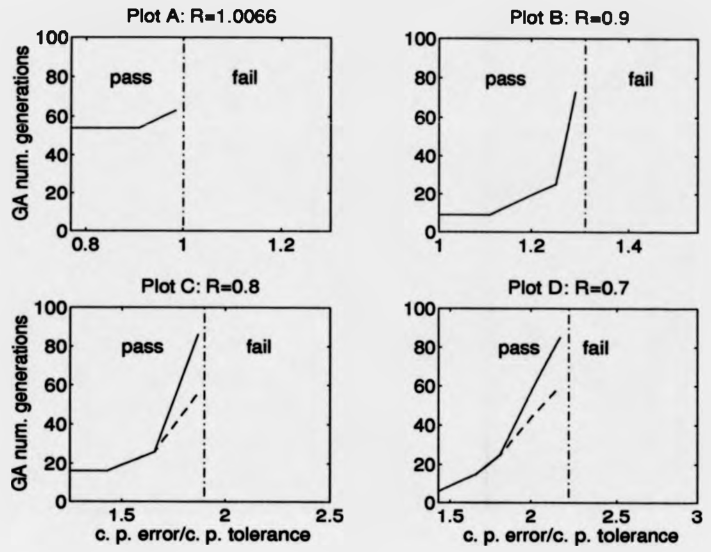


Figure 5.26: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance and roundness error to tolerance (R); without buffer zone (-), and with a buffer a zone of 1.0×10^{-5} mm (- -): data set 12.

$R = 1.0066$		$R = 0.9$		$R = 0.8$		$R = 0.7$	
C	gen.	C	gen.	C	gen.	C	gen.
0.77	54	1.0	9	1.25	16	1.428	6
0.83	54	1.11	9	1.428	16	1.66	15
0.91	54	1.25	25	1.66	26	1.81	25
0.985	63	1.29	73	1.87	86	2.17	85

Table 5.17: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance (C) and roundness error to tolerance (R), without buffer zone: data set dt12.

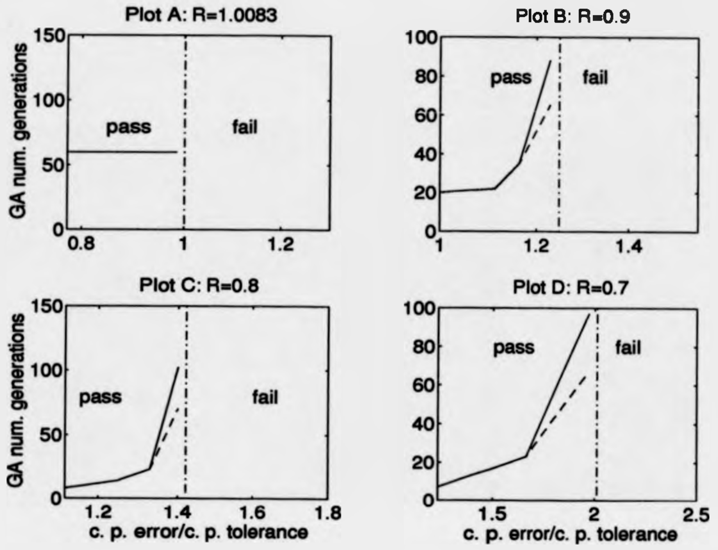


Figure 5.27: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance and roundness error to tolerance (R); without buffer zone (-), and with a buffer a zone of 1.0×10^{-5} mm (- -): data set 13.

$R = 1.0083$		$R = 0.9$		$R = 0.8$		$R = 0.7$	
C	gen.	C	gen.	C	gen.	C	gen.
0.77	60	1.0	20	1.11	8	1.25	7
0.83	60	1.11	22	1.25	14	1.428	14
0.91	60	1.162	35	1.33	23	1.66	23
0.985	60	1.23	88	1.4	102	1.97	97

Table 5.18: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance (C) and roundness error to tolerance (R), without buffer zone: data set dt13.

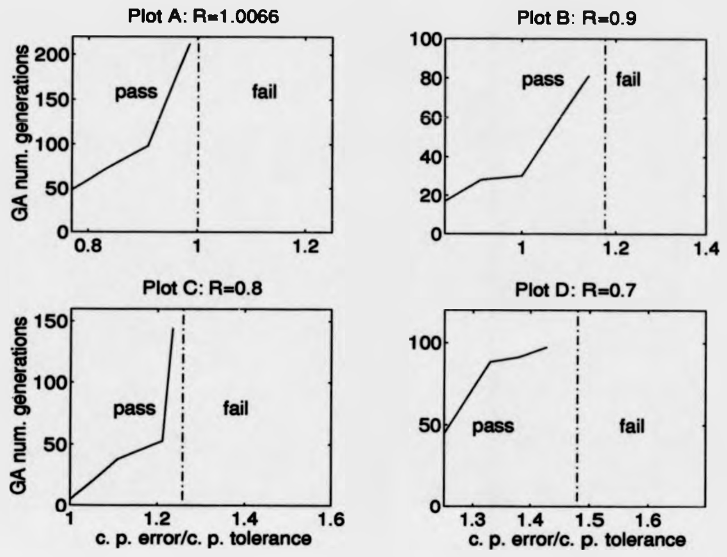


Figure 5.28: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance and $(R_{nom} - R_{min})$ to tolerance on radius (R), with a buffer zone of 1.0x10^{-5} mm: data set s-4h01.

$R = 1.0066$		$R = 0.9$		$R = 0.8$		$R = 0.7$	
C	gen.	C	gen.	C	gen.	C	gen.
0.77	48	0.83	17	1.0	5	1.25	45
0.83	71	0.91	28	1.11	38	1.33	88
0.91	98	1.0	30	1.212	53	1.379	91
0.985	212	1.1428	81	1.2345	144	1.428	97

Table 5.19: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance (C) and $(R_{nom} - R_{min})$ to tolerance on radius (R): data set s-4h01.

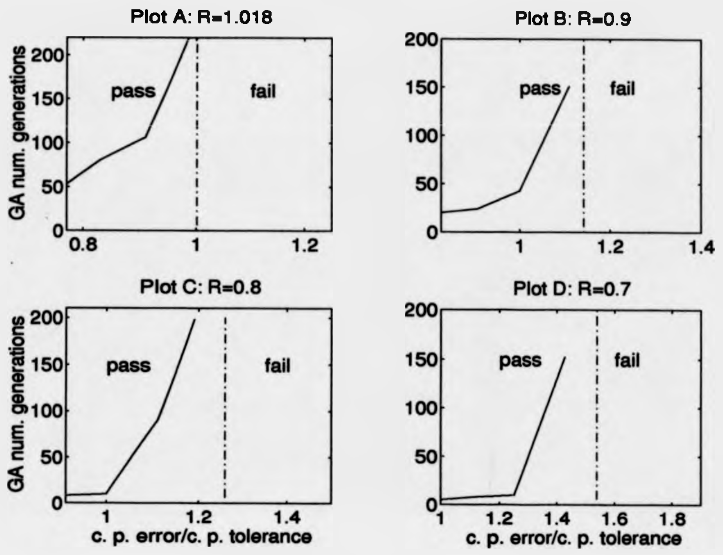


Figure 5.29: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance and $(R_{nom} - R_{min})$ to tolerance on radius (R), with a buffer zone of 1.0x10^{-5} mm: data set s-4h02.

$R = 1.018$		$R = 0.9$		$R = 0.8$		$R = 0.7$	
C	gen.	C	gen.	C	gen.	C	gen.
0.77	53	0.83	20	0.91	8	1.0	5
0.83	80	0.91	24	1.0	10	1.11	8
0.91	106	1.0	43	1.11	90	1.25	10
0.985	219	1.11	151	1.19	198	1.428	152

Table 5.20: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance (C) and $(R_{nom} - R_{min})$ to tolerance on radius (R): data set s-4h02.

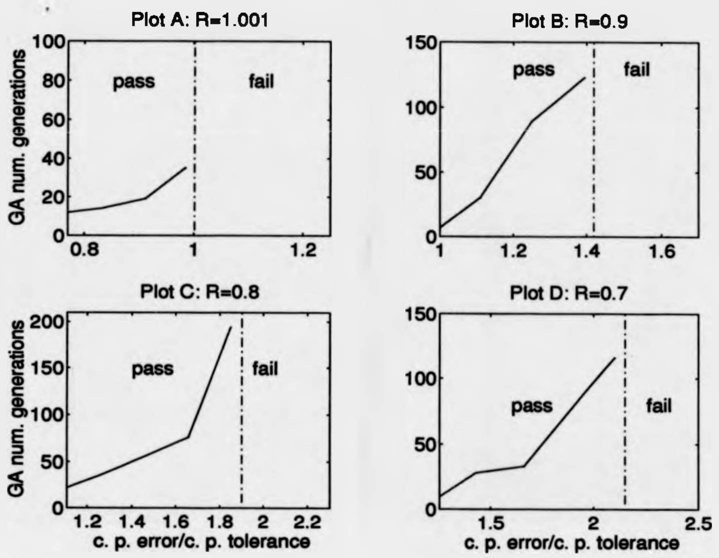


Figure 5.30: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance and $(R_{nom} - R_{min})$ to tolerance on radius (R), with a buffer zone of $1.0\text{x}10^{-5}$ mm: data set s-4h03.

$R = 1.0066$		$R = 0.9$		$R = 0.8$		$R = 0.7$	
C	gen.	C	gen.	C	gen.	C	gen.
0.77	12	1.0	7	1.11	22	1.25	10
0.83	14	1.11	30	1.25	35	1.428	28
0.91	19	1.25	89	1.66	76	1.66	33
0.985	35	1.395	123	1.85	194	2.1	116

Table 5.21: Number of generations to get a feasible solution for different ratios of: centre position error to tolerance (C) and $(R_{nom} - R_{min})$ to tolerance on radius (R): data set s-4h03.

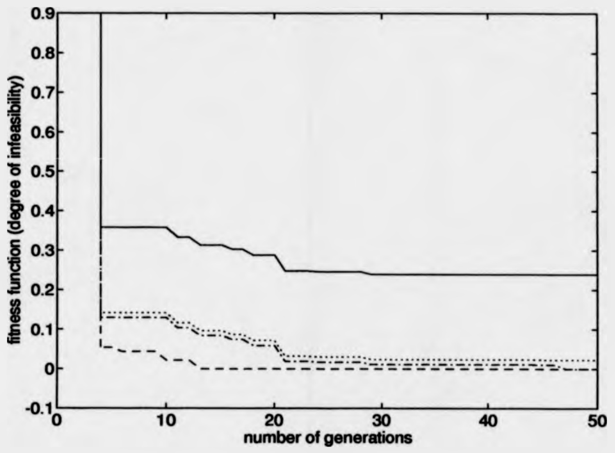


Figure 5.31: Evolution of solutions towards feasibility for data set s-1, for ratio of roundness error to tolerance $R = 0.9$ and for ratios of centre position error to tolerance $C = 1.0$ (- -), 1.136 (-.), 1.19 (..) and 1.6 (-): data set s-1.

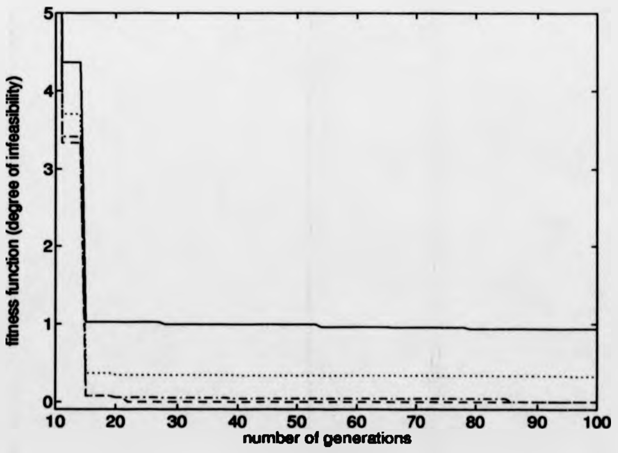


Figure 5.32: Evolution of solutions towards feasibility for data set 13, for ratio of roundness error to tolerance $R = 0.9$ and for ratios of centre position error to tolerance $C = 1.11$ (- -), 1.23, (-.), 1.26 (..) and 1.66 (-): data set 13.

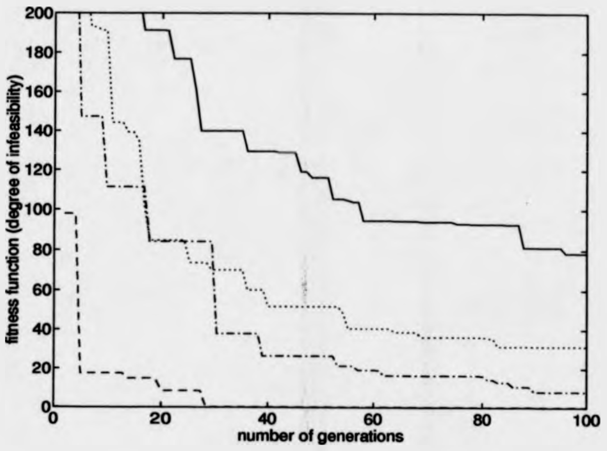


Figure 5.33: Evolution of solutions towards feasibility for data set 13, for ratio of roundness error to tolerance $R = 0.9$ and for ratios of centre position error to tolerance $C = 1.11$ (- -), 1.23, (-.), 1.26 (..) and 1.66 (-): data set s-4h01.

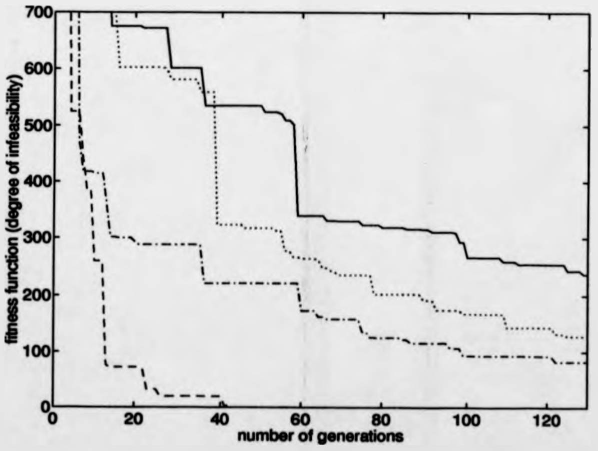


Figure 5.34: Evolution of solutions towards feasibility for data set 13, for ratio of roundness error to tolerance $R = 0.9$ and for ratios of centre position error to tolerance $C = 1.11$ (- -), 1.23, (-.), 1.26 (..) and 1.66 (-): data set s-4h02.

Chapter 6

Non-Linear Programming Methods for Inspecting Geometric Tolerances

6.1 Introduction

In the previous chapter, the approach discussed in chapter 3 and 4 for inspecting geometric features was extended to the inspection of circular mating features, a typical as well as important example of a situation that involves non-linear constraints in the formulation of the inspection problem. The advantages and disadvantages of using a linear model as an approximation to the non-linear model were also discussed and alternatives to formal non-linear programming techniques, namely genetic search techniques, were investigated. In this chapter, the use of formal non-linear programming techniques for inspecting geometric tolerances are further investigated and compared with the genetic search techniques discussed in the previous chapter.

The optimisation of a non-linear function subject to non-linear constraints is considerably more complicated than that with only linear constraints, and the methods reflect this increase in complexity. With linear constraint methods, an initial feasible point is computed, and all iterates thereafter are feasible. This is possible because the search direction can be constructed so that the constraints in the current working set are automatically satisfied at all trial points computed during the iteration. By contrast, when even one constraint function is non-linear, it is not straightforward to generate a sequence of iterates that exactly satisfy a specified sub-set of the constraints. Although driving the solution of a non-linear programme to optimality may present difficulties, the position may be more satisfactory when only the discovery of feasibility is of concern, as is the case of the inspection problems proposed here.

Despite the above comments, reliable commercial software implementations of non-linear programming methods exist, for example Sequential Quadratic Programming methods, or reduced gradient methods. Reduced gradient methods are particularly interesting because, similarly to linear programming methods, they can be used to determine whether a problem is feasible before starting the optimisation process.

This chapter investigates the applicability of such methods in the determination of the feasibility of the inspection problem. Specifically, it investigates the efficiency of commercial software implementations of two of such methods, namely the NAG implementation of the Sequential Quadratic Programming (SQP) [NAG, 1990] and the GINO implementation of the Generalised Reduced Gradient (GRG) algorithm [The Scientific Press, 1992], for testing the feasibility of the model problems formulated in chapter 5 (as in (5.6), (5.7) and (5.8), section 5.5.1).

6.2 Non-Linear Programming Methods

An optimisation problem of a non-linear function subject to non-linear constraints has the form of: minimise (or maximise) $F(X)$ subject to

$$\begin{aligned} 0 \le c_i(X) \le ub(n+i)\ , &\quad i = 1, \cdots, n_{ineq}\ , \\ c_i(X) = 0\ , &\quad i = n_{ineq}+1, \cdots, m\ , \\ lb(i) \le X_i \le ub(i)\ , &\quad i = 1, \cdots, n\ , \end{aligned} \tag{6.1}$$

where X is the solution vector of n variables, c_i are equality and inequality non-linear constraints and lb and ub are the lower and upper variable and constraint bounds. The number of inequality constraints, n_{ineq}, or equality constraints, $m - n_{ineq}$ may be zero. The objective function $F(X)$ and the constraints c_i are assumed differentiable. In our inspection problems (formulated as in (5.6), (5.7) and (5.8)) there are no equality constraints and the inequality constraints are all differentiable. There is no objective function explicitly defined, since our problems are primarily concerned with feasibility.

Generally, optimisation methods generate a sequence of iterates leading to the solution. For example, in the methods for unconstrained and linearly constrained optimisation, an iteration is defined by the calculation of a search direction and of a step length. In general, non-linear programming methods generate the next iterate by solving a complete general unconstrained or linearly constrained sub-problem. This is the case of penalty and barrier function methods (see e. g. [Gill, Murray and Wright, 1981] and [Fiacco and McCormick, 1968]) and augmented Lagrangian methods (see e. g. [Gill, Murray and Wright, 1981] and [Hestenes, 1969]), which solve the original problem by formulating a sequence of unconstrained sub-problems related in some way to the original problem. Still in this class are the projected Lagrangian methods [Gill, Murray and Wright, 1981 and Fletcher, 1987], which transform the original problem to a sequence of linearly constrained sub-problems based on the Lagrangian function. However, some methods are based on extending methods for linear constraints to the non-linear

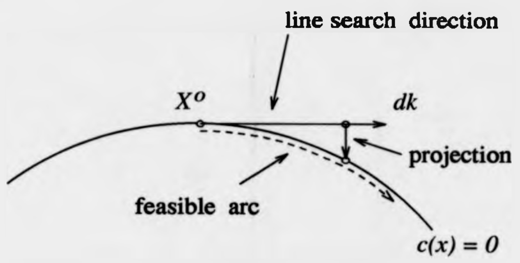


Figure 6.1: Feasible direction search.

case and, therefore, display the same structure as linearly constrained methods, namely, an iteration is composed of the calculation of a search direction followed by the calculation of a step length. The presence of non-linear constraints does not guarantee that a line search maintains feasibility so a simplistic explanation of what is done is that at any feasible point X^o a search direction d_k in the tangent plane is calculated and a feasible arc is then obtained by projecting any point in the feasible region, as illustrated in figure 6.1. This is the case of gradient projection [Rosen, 1961] or reduced gradient methods [Abadie and Carpentier, 1969] (these are often called feasible direction methods).

The next sections further review and present algorithmic details of the generalised reduced gradient method, and the Sequential Quadratic Programming, a particular method of projected Lagrangian methods.

Before moving on to the next section, the optimality conditions for a non-linear problem as in (6.1) are stated, since they apply to any method of solution. The necessary conditions for a solution X^* to be a local minimum of (6.1) are [Gill, Murray and Wright, 1981]:

1. $c(X^*) \geq 0$, with $\hat{c}(X^*) = 0$, where $\hat{c}$ denote the subset of nb constraints that are active at X^*;
2. $g(X^*) = \hat{A}(X^*)^T \lambda^*$, where $g(X^*)$ is the gradient vector of the objective function, $\hat{A}(X^*)$ is the matrix whose rows are the transposed gradient vectors of the active constraints and λ^* is a nb-vector of Lagrange multipliers

of the corresponding active constraints;

3. $\lambda_i^* \geq 0$, $i = 1, \cdots, nb$; and

4. $Z(X^*)^T W(X^*, \lambda^*) Z(X^*)$ is positive semi-definite, where $Z(X^*)$ denotes a matrix whose columns form a basis for the set of vectors orthogonal to the rows of $\hat{A}(X^*)$ and $W(X^*, \lambda^*)$ denotes the Hessian matrix, that is the matrix of second partial derivatives, of the Lagrangian function L given by

$$L(X, \lambda) = F(X) - \lambda^T \hat{c}(X) \tag{6.2}$$

When zero valued Lagrange multipliers λ_i^* are present, extra restrictions are necessary on the Hessian matrix of the Lagrangian function, as discussed in [Gill, Murray and Wright, 1981]. These optimality conditions for constrained optimisation are often called the Kuhn-Tucker conditions [Kuhn and Tucker, 1951].

6.3 Generalised Reduced Gradient Methods

Generalised Reduced Gradient (GRG) algorithms for non-linearly constrained optimisation problems were first developed by Abadie and Carpentier [1969], who also designed the first GRG software. In a computational study of Colville [1968], GRG methods were found to be among the most reliable and efficient at that time. Subsequent work on reduced gradient methods has been performed, amongst others, by Sargent and Murtagh [1973], Abadie [1978] and Lasdon, Waren et al. [1978], which resulted in further improvements to the original method.

There are many possible GRG algorithms. Basically, they are motivated by the same idea as are active set methods for linear constraints: to stay "on" a subset of the non-linear constraints while reducing the objective function. However, when non-linear constraints are involved, some sort of iterative "correction" process is required to follow a curving constraint boundary. The differences among reduced gradient type of algorithms arise from the variety of techniques used to achieve the aims of staying feasible and reducing the objective function. Their underlying concepts are described by Abadie and Carpentier [1969], Abadie [1978] and Lasdon, Fox and Ratner [1973]. This section briefly describes GRG2, the algorithm developed by Lasdon, Waren et. al. [1978], and implemented by GINO [The Scientific Press, 1992].

The problem as stated in 6.1 is converted to the following equality form by adding slack variables $X_{n+1}, \cdots, X_{n+m}$: minimise $F(X)$ subject to

$$\begin{aligned} c_i(X) - X_{n+i} = 0 \,, \quad i = 1, \cdots, m \,, \\ lb(i) \leq X_i \leq ub(i) \,, \quad i = 1, \cdots, n+m \,, \end{aligned} \tag{6.3}$$

where $lb(i) = 0$ for $i = n+1, \cdots, n + n_{ineq}$ and $lb(i) = ub(i) = 0$ for $i = n + n_{ineq} + 1, \cdots, n+m$. The variables $X_1, \cdots, X_n$ are called natural variables.

Let X^o be a feasible solution to the constraints given in 6.1, and assume that nb of the c_i constraints are binding (i. e. hold as equalities) at X^o. A constraint g_i is taken as binding if $|c_i - ub(n+i)| < \epsilon$ or $|c_i - lb(n+i)| < \epsilon$, that is if it is within ϵ of one of its bounds (ϵ is the parameter "EPNEWT" or "EPINIT" in GINO [The Scientific Press, 1992], see section 6.5.2).

GRG algorithms use the nb binding constraint equations to solve for nb of the natural variables, called the basic variables, in terms of the remaining $n - nb$ natural variables and the nb slacks associated with the binding constraints. These n variables are called non-basic. Let $\underline{y}$ be the vector of nb basic variables and $\underline{x}$ the vector of n non-basic variables, with their value corresponding to X^o denoted by $(\underline{y}^o, \underline{x}^o)$. Then the binding constraints can be written

$$\hat{c}(\underline{y}, \underline{x}) = 0 \tag{6.4}$$

where $\hat{c}$ is the vector of nb binding constraint functions (the definition of c is extended here to include the slack variables). The basic variables must be selected so that the nb-by-nb basis matrix $B = (\partial \hat{c}_i / \partial y_j)$ $i = j = 1, \cdots, nb$ is non-singular at X^o (that is has linearly independent columns). Then the binding constraints in 6.4 may be solved for $\underline{y}$ in terms of $\underline{x}$, yielding a function $y(\underline{x})$ valid for all $(\underline{y}, \underline{x})$ sufficiently near $(\underline{y}^o, \underline{x}^o)$. This reduces the objective to a function of $\underline{x}$ only

$$F(y(\underline{x}), \underline{x}) = \psi(\underline{x}) \tag{6.5}$$

and, in a neighbourhood of $(\underline{y}^o, \underline{x}^o)$, reduces the original problem to a simpler reduced problem: minimise $\psi(\underline{x})$ subject to

$$l \leq \underline{x} \leq u \tag{6.6}$$

where l and u are the bound vectors for $\underline{x}$. The function $\psi(\underline{x})$ is called the reduced objective and its gradient, $\nabla\psi(\underline{x})$, the reduced gradient.

The original problem is solved by solving a sequence of reduced problems. The reduced problems are solved by a gradient-based iterative method, whose general descent algorithm is as follows:

1. Compute the gradient of the reduced objective function at the current point $(\underline{y}^o, \underline{x}^o)$, $\nabla\psi(\underline{x}^o)$;
2. If the current solution is close enough of being optimal, stop;
3. Compute a one-dimensional search direction $\underline{d}$ from $\nabla\psi(\underline{x}^o)$;
4. Determine how far to move along this search direction, starting from $(\underline{y}^o, \underline{x}^o)$, and move this distance to a new point. Replace $(\underline{y}^o, \underline{x}^o)$ by this new point and return to step 1.

At a given iteration k, with non-basic variables $\underline{x}^o$ and basic variables $\underline{y}^o$, the inverse of the basis matrix, B^{-1}, is computed and the gradient $\nabla\psi(\underline{x}^o)$ is evaluated as follows

$$\begin{aligned} \pi &= (\partial F/\partial \underline{y})^T B^{-1} \\ \partial\psi/\partial \underline{x}_k &= (\partial F/\partial \underline{x}_k) - \pi(\partial \hat{c}/\partial \underline{x}_k) \end{aligned} \tag{6.7}$$

If $(\underline{y}^o, \underline{x}^o)$ is not optimal, then a search direction $\underline{d}$ is formed from $\nabla\psi(\underline{x}^o)$ and a one-dimensional search is initiated to solve the problem

$$\text{minimise} \quad \psi(\underline{x}^o + \alpha\underline{d}) \tag{6.8}$$

for $\alpha > 0$.

There are several ways in which a search direction may be determine. GRG2 uses the Broyden-Fletcher-Shanno (BFS) variable metric algorithm [Fletcher, 1970a, and Shanno et. al., 1974], modified to accommodate upper and lower bounds on the variables as suggested by Goldfarb [1969]. Alternatively, conjugate gradient (CG) methods [Fletcher, 1987] may be used (five CG methods are available in GINO [The Scientific Press, 1992], as described in section 6.5.2).

The minimisation of 6.8, that is to determine how far to move along $\underline{d}$, is done only approximately to a first local minimum by choosing a sequence of positive values $\{\alpha_1, \alpha_2, \cdots\}$ for α. For each value α_i, $\psi(\underline{x}^o + \alpha_i\underline{d})$ must be evaluated. By equation 6.5, this is equal to $F(y(\underline{x}^o + \alpha_i\underline{d}), \underline{x}^o + \alpha_i\underline{d})$ so that basic variables $y(\underline{x}^o + \alpha_i\underline{d})$ must be determined. These satisfy the system of equations defined by the binding constraints

$$\hat{c}(y, \underline{x}^o + \alpha_i\underline{d}) = 0 \tag{6.9}$$

where $\underline{x}^o$, $\underline{d}$ and α_i are known and y is to be found. This system is solved by a variant of Newton's method [Abadie and Carpentier, 1969].

In determining how far to move along the search direction $\underline{d}$, the initial step size is determined in a similar way to that described by Lasdon, Fox and Ratner [1973a]. GRG2 then operates in two phases, halving the initial step size (if necessary) until an improved point is found, or doubling the step size until the minimum is bracketed. If, in calculating $\psi(\underline{x}^o + \alpha_i\underline{d})$, the Newton's method fails to converge, then, if this occurs on the first step, the step size is halved and it is tried again. Otherwise, if an improved point has already been found, the search is terminated. The search may continue until for three successive values of α_i,

A, B and C, the objective value is such that they satisfy

$$\begin{gathered} 0 \leq A \leq B \leq C \\ \psi(\underline{x}^o + A\underline{d}) \geq \psi(\underline{x}^o + B\underline{d}) \leq \psi(\underline{x}^o + C\underline{d}) \end{gathered} \tag{6.10}$$

Then the interval $[A, C]$ contains a local minimum of $\psi(\underline{x}^o + \alpha\underline{d})$. An approximation to this minimum is located by fitting a quadratic function to the set of points.

If, in searching for such three successive points, some c_i constraints (which were previously not binding) or basic variable bounds may be violated. Then, a new (reduced) α value is determined, such that at least one such new constraint or variable is at its bound and all others are satisfied. To determine this new constraint, an estimate is made of α using linear interpolation between the current and previous values of the violated constraints. If, at this new point, the objective is less than at all previous points, the new constraint is added to the set of binding constraints, the one-dimensional search is terminated and solution of a new reduced problem begins. It is worth to mention that this is an important feature of this algorithm, as it attempts to return to the (non-linear) constraint surface at each step in the one-dimensional search. This differs from early strategies suggested by Abadie [1972], which involves linear search on the plane tangent to the constraint surface prior to returning to that surface.

The current solution is considered optimal if either of two tests is met. The first test checks whether the Kuhn-Tucker optimality conditions are satisfied to within a small positive number which can be controlled by the user. The second optimality test checks whether the fractional change in the objective is less than a certain value for a certain number of consecutive iterations, again set by the user (respectively "EPSTOP" and "NSTOP" in GINO [The Scientific Press, 1992], see section 6.5.2).

When the basis constructed, B, is degenerate, that is it has one or more basic variables at bounds, the search direction produced may cause some of these variables to violate a bound immediately, i. e. d may not be a feasible direction. Then, GRG2 computes a usable feasible direction or proves that the current point satisfies the Kuhn-Tucker conditions.

The algorithm assumes that a feasible solution X^o is available. In the absence of an initial feasible solution, such a solution can be obtained (provided it exists) by the introduction of artificial variables in a manner quite similar to that used for the simplex calculations [Hadley, 1964]. Referring to the problem formulated in 6.1, assume that no feasible solution is available. Select an initial point X_{init} and compute $c_i(X_{init}),\ i = 1, \cdots, m$. For convenience assume that only the first l inequality constraints are violated, so that $c_i(X_{init}) < 0,\ i = 1, \cdots, l$. Then, $l+1$ new variables $x_{n+i},\ i = 1, \cdots, l+1$ are introduced and a region in a $(n+l+1)$ parameter space is defined by the constraints

$$\begin{aligned} & c_i(X) + x_{n+i} \geq 0\ , \quad i = 1, \cdots, l \\ & c_i(X) \geq 0\ , \quad i = l+1, \cdots, n_{ineq} \\ & c_i(X) = 0\ , \quad i = 1 + n_{ineq}, \cdots, m \\ & -\textstyle\sum_{i=1}^{l} x_{n+i} - x_{n+l+1} \geq 0 \\ & lb(i) \leq X \leq ub(i)\ , \quad i = 1, \cdots, n+l \end{aligned} \tag{6.11}$$

where $lb(i) = 0$ for $i = n+1, \cdots, n+l$.

The new solution vector defined by $[X_{init}, x_{n+1}, \cdots, x_{n+l+1}]$, where $x_{n+i} = -g_i(X_{init}),\ i = 1, \cdots, l$ and $x_{n+l+1} = -\sum_{i=1}^{l} x_{n+i}$ is a feasible solution to the set of constraints above. We now solve the problem

$$maximise\ \ Z = x_{n+l+1} \tag{6.12}$$

subject to the set of constraints in 6.11. If $max\ \ Z = x_{n+l+1} = 0$, then, the corresponding point X is a feasible solution to the set of constraints originally defined in 6.1. Otherwise, if $max\ \ Z = x_{n+l+1} < 0$, it can be proved that no feasible solution exists to the original set of constraints in 6.1.

6.4 Projected Lagrange Methods

Fletcher [1974, 1977 and 1987], Murray [1976] and Gill, Murray and Wright [1981] present an overview of projected Lagrangian methods (or Lagrange-Newton methods according to Fletcher). Within Projected Lagrange methods, this review considers a class of methods in which the objective function is specialised to be a quadratic function, so that the subproblem of interest is a quadratic programme (QP). The first suggestion of using a QP sub-problem to solve a non-linearly constrained problem was made by Wilson [1963]. This was followed by innumerable works on this method, as reported by Gill, Murray and Wright [1981].

When optimality conditions 2, 3 and 4 hold, the optimum point X^* is a stationary point, that is minimum of the Lagrangian function. Based on this property, the optimum point X^* of (6.1) can be defined as the solution of a linearly constrained sub-problem, whose objective function is related to the Lagrangian function, and whose linear constraints are chosen so that minimisation occurs only within the desired sub-space. This suggests that the solution of a non-linearly constrained problem can be obtained by solving a sequence of linearly constrained sub-problems based on the Lagrangian function. Since a linearly constrained sub-problem is itself a constrained optimisation problem, the Lagrange multipliers of the sub-problem provide estimates of the multipliers of the original problem.

Thus, the following iterative method is suggested [Gill, Murray and Wright, 1981 and Fletcher, 1987]: given an initial point X_{init} and an initial Lagrange multiplier vector λ_{init}, set $k \leftarrow 0$ and repeat the following steps:

1. Check termination criteria. If X_k satisfies the optimality conditions for the problem in (6.1), the algorithm terminates with X_k as the solution;
2. Solve the quadratic programming sub-problem. Let p_k denote the solution

of the quadratic programme

$$\begin{aligned} \text{minimise} \quad & \Phi_k(p_k) \\ \text{subject to} \quad & I_k(p_k) \geq 0 \end{aligned} \tag{6.13}$$

3. Update the estimate of the solution. Set $X_{k+1} \leftarrow X_k + p_k$, set λ_{k+1} to the Lagrange multiplier vector associated with the k^{th} sub-problem (6.13), set $k \leftarrow k + 1$ and go back to step 1. Note that the solution of the quadratic subproblem is the step from X_k to X_{k+1} rather than X_{k+1} itself.

During the k^{th} major iteration, the linear constraints I_k of the sub-problem in (6.13) are obtained by replacing the active set of non-linear constraints, $\hat{c}(X_k) = 0$, by their first order Taylor series approximation about X_k given

$$I_k(p_k) \equiv \hat{A}_k p_k + \hat{c}_k \tag{6.14}$$

where $\hat{c}(X_k)$ (denoted by $\hat{c}_k$) is the vector of active constraint values at X_k, and $\hat{A}(X_k)$ (denoted by $\hat{A}_k$) is a matrix whose rows are the transposed gradient vectors of the active constraints.

Likewise, the objective function $F(X)$ in (6.1) is replaced by the quadratic function

$$\Phi_k(p_k) \equiv g_k^T p_k + \frac{1}{2} p_k^T H_k p_k \tag{6.15}$$

where $g(X_k)$ (denoted by g_k) is the gradient of $F(X_k)$, and H_k is an approximation of the Hessian matrix of the Lagrangian function (given in (6.2)).

The solution of (6.13) is given by solving the system of linear equations in p_k and λ_k, namely

$$\begin{pmatrix} H_k & -\hat{A}_k^T \\ \hat{A}_k & 0 \end{pmatrix} \begin{pmatrix} p_k \\ \lambda_k \end{pmatrix} = \begin{pmatrix} -g_k \\ -\hat{c}_k \end{pmatrix} \tag{6.16}$$

where λ_k is used as estimates of the Lagrange multipliers for the $(k+1)^{th}$ sub-problem.

The derivation of QP-based methods is based on conditions that hold only in a small neighbourhood of the optimum, and hence the significance of the sub-problem as in (6.14) and (6.15) is questionable when X_k is not close to the optimum solution X^*. Gill, Murray and Wright [1981], briefly consider formulations of the QP sub-problem that may differ from (6.13) when X_k is far from optimal. For a discussion of the merits of different formulations of the QP sub-problem, see Murray and Wright [1980]. In the NAG implementation of this method (the Sequential Quadratic Programming (SQP) function E04VCF [NAG, 1990]), the Hessian matrix of the quadratic approximation of $F(X)$ (H_k in (6.15)) is an approximation of the Hessian matrix of an augmented Lagrangian function (see [Gill, Murray and Wright, 1981]).

In order to ensure that X_{k+1} is a better point than X_k, the solution of the QP sub-problem can be used as a search direction. The next iterate is then defined as

$$X_{k+1} = X_k + \alpha_k p_k \tag{6.17}$$

where p_k is the result of the QP sub-problem, and α_k is a step length chosen to yield a "sufficient decrease" in some suitable chosen merit function that measures progress toward X^*. In the SQP algorithm implemented by NAG [1990], the chosen merit function is an augmented Lagrangian function.

Strategies for determining the active set of constraints at each major iteration and initial estimates of the Lagrange multipliers are briefly discussed by Gill, Murray and Wright [1981]. No information is given in the NAG documentation about such details.

One disadvantage of this method is that if no feasible point exists, the algorithm keeps iterating, in an attempt to reduce the constraint violations.

6.5 Algorithm Implementation and Tests

6.5.1 Objectives

The objective of the work described here has been to test the practicability of using some standard non-linear programming techniques in the context of on-line geometric tolerance inspection. In order to do that, commercial software implementations of the GRG and SQP methods (see sections 6.4 and 6.3), respectively GINO and NAG, were used for checking, for some data sets, whether the tolerance constraints were violated. The data sets used were related to the inspection cases formulated in chapter 5, that is:

1. inspection of roundness and centre position of a circular feature, formulated as: find (a, b) and R such that

$$\begin{aligned} (x_i - a)^2 + (y_i - b)^2 &\leq (R + t_r)^2 \\ (x_i - a)^2 + (y_i - b)^2 &\geq R^2 \\ (x_o - a)^2 + (y_o - b)^2 &\leq t_{cp} \end{aligned} \tag{6.18}$$

   for $i = 1, \cdots, N$, where N is the number of data points and (x_o, y_o) the nominal position.

2. template matching: inspection of position and dimension of four circular features (holes or studs), having the same dimensions and tolerances, on a plate forming a square frame. Formulated as: find reference centres $(a_k, b_k), k = 1, \cdots, 4$ such that

$$R^2_{min} \leq (x_i - a_k)^2 + (y_i - b_k)^2 \leq R^2_{max}\ , \quad i = 1, \cdots, N_k\ , \quad k = 1, \cdots, 4 \tag{6.19}$$

and

$$\begin{aligned}
(L - t_{cp})^2 &\leq (a_1 - a_2)^2 + (b_1 - b_2)^2 \leq (L + t_{cp})^2 \\
(L - t_{cp})^2 &\leq (a_1 - a_3)^2 + (b_1 - b_3)^2 \leq (L + t_{cp})^2 \\
(L - t_{cp})^2 &\leq (a_2 - a_4)^2 + (b_2 - b_4)^2 \leq (L + t_{cp})^2 \\
(L - t_{cp})^2 &\leq (a_3 - a_4)^2 + (b_3 - b_4)^2 \leq (L + t_{cp})^2
\end{aligned} \tag{6.20}$$

where N_k are the number of data points representing each circular feature, R_{min} and R_{max} are the radius lower and upper limits, L is the nominal distance between centres and t_{cp} is the centre position tolerance. The references are numbered such that $k = 1, 2$ are the bottom, respectively left and right, and $k = 3, 4$ are the top, respectively left and right ones, as illustrated in figure 5.7.

The sequence of tasks obeyed for performing the tests was generally the same as for the tests described in the previous chapters. Details about the software packages, data acquisition and generation, and test procedures are described in the next sections.

6.5.2 GINO Software

GINO (General Interactive Optimiser) [The Scientific Press, 1992] is a modeling programme for solving optimisation problems which combines an interactive interface with GRG2, an implementation of the generalised reduced gradient (see section 6.3).

The version of GINO available for the tests was capable of handling up to 100 variables and up to 50 constraints, requiring 512 kbytes of RAM (Random Access Memory) and appropriate for running on IBM PC compatibles or Apple Macintosh.

Details about how to enter the objective and constraint functions and run the programme are described in the software user guide and will not be considered here. In order to reduce the computational effort, an initial solution must be supplied, whether feasible or not, close to the estimated optimal solution. In addition to that there are a number of parameter options within GINO and GRG2 that allow you to set various print levels, set limits on iterative processes, set numerical tolerances used to determine if constraints are adequately satisfied and if various processes have converged, and choose alternative methods that can be used to perform different tasks. The parameters that are relevant to the iteration process are listed below. These parameters have default value assigned to them, which are indicated in square brackets. The values used in our problems, if different from the default ones, are indicated. The options are as follows:

- final binding constraint tolerance (EPNEWT) [10^{-4}]: in GRG2 a constraint is assumed to be binding if it is within "EPNEWT" of its bound (either greater or less than the bound by no more than this amount). If a constraint is not binding, then it is either within its bounds or it is a violated constraint. Increasing it can sometimes speed convergence, while decreasing it occasionally yields a more accurate solution. Values larger than 0.01 should be treated cautiously, as should values smaller than 10^{-6}. A value "EPNEWT" $= 10^{-5}$ was chosen, similar to the "buffer zone" defined by the genetic search model discussed in chapter 5.
- initial binding constraint tolerance (EPINIT) [10^{-4}]: if it is desired to solve a given problem, first with a relatively large value of "EPNEWT", then with "EPNEWT" set to a smaller value, this can be achieved by assigning the initial constraint tolerance to "EPINIT" and the final one to "EPNEWT".
- fractional objective change tolerance (EPSTOP) [10^{-4}]: if the fractional change in the objective function is less than "EPSTOP" for "NSTOP" (see below) consecutive iterations, and the Kuhn-Tucker optimality conditions are not satisfied within "NSTOP", then GRG2 terminates.
- pivot rejection tolerance (EPSPIV) [10^{-3}]: if, in constructing the basis inverse, elements that are too small in absolute value are selected for pivot elements, it is possible that numerical inaccuracies in the inverse matrix will

occur that will prevent finding the solution. Therefore, if the absolute value of a prospective pivot element is less than "EPSPIV", then that element will be rejected as a pivot candidate.

- phase I objective augmentation (PH1EPS) [0.0]: sometimes an initial approximation to the optimum point is available that yields a good value for the objective function but is infeasible. On the other hand, the initial feasible point found by phase I may yield a much poorer value of the objective function, sometimes so far from the approximate solution that GRG2 never finds the proper point. A non-zero value of "PH1EPS" will cause phase I to incorporate a multiple of the true objective, along with the sum of infeasibilities, in the phase I objective. The multiple is selected so that, at the initial point, the ratio of the true objective and the sum of the infeasibilities is equal to "PH1EPS". The larger the value of "PH1EPS", the more emphasis given to the actual objective function.

- iteration limit if no change in objective (NSTOP) [3]: if the fractional change in the objective function is less than "EPSTOP" for "NSTOP" consecutive iterations, the programme will try some alternative strategies. If these do not produce a fractional function change greater than "EPSTOP", then the optimisation process is terminated.

- Newton iteration limit (ITLIM) [10]: if after "ITLIM" iterations of the subroutine "NEWTON", it has not converged satisfactorily (in its attempt to solve the binding constraint equations for the basic variable values), the iterations are stopped and corrective action is taken.

- one dimensional search iteration limit (LIMSER) [10,000]: if the number of completed one dimensional searches equals "LIMSER", then optimisation is terminated.

- basic variable estimation (IQUAD) [0]: The default method (option 0) for estimating initial values for basic variables uses linear extrapolation based on a calculated tangent vector. Quadratic extrapolation can often speed computations by providing better initial values of the basic variables for the

Newton calculations. Quadratic extrapolation (option 1) has been selected in our case.

- derivatives (KDERIV) [0]: partial derivatives of the objective and constraint functions can be approximated by numerical forward differences (default option, 0) or central differences (option 1). Central differences are exact for linear and quadratic functions, while forward differences are exact only for linear functions. However, each central difference computed requires two function evaluations, while each forward difference requires only one.

- conjugate gradient method (MODCG) [1]: this parameter, together with "MAXHES" (see below) controls which algorithm is used to generate search directions. If a conjugate gradient algorithm is to be used, then "MODCG" specifies which one. The options are listed below, where option "1", the Fletcher-Reeves formula, is the default one. For information about these methods, see Fletcher [1987] and Dennis and Schnable [1983].

"MODCG" value	Name of CG method
1	Fletcher-Reeves
2	Polak-Ribiere
3	Perry
4	1-step DFP
5	1-step BFGS

  Using a conjugate gradient (CG) algorithm instead of the default "BFGS" method (by setting "MAXHES" = 0, as indicated below) is primarily useful in problems with too many variables for the amount of computer memory available. Using one of the "CG" options reduces storage requirements by about $n(n+1)/2$ where n is the number of variables. However, the "CG" options often require more iterations and more computer time than the default "BFGS" option.

- search direction (MAXHES) [-1]: the default option calls the "BFGS" to compute the search direction. This option is the Broyden-Fletcher-Shanno

(BFS) variable metric algorithm [Fletcher, 1970a, and Shanno et. al., 1974], modified to accommodate upper and lower bounds on the variables as suggested by Goldfarb [1969]. Otherwise, if "MAXHES" is set to "0", it uses a conjugate gradient formula for generating search directions. The specific formula to be used is determined by "MODCG".

6.5.3 SQP NAG Function

The NAG library has two optimisation functions available that uses a sequential quadratic programming (SQP) algorithm: E04VDF and E04VCF. The latter is a comprehensive routine, that allows more user supplied parameter options than the other one, and was used in our tests.

E04VCF is a Fortran routine designed to minimise an arbitrary smooth function subject to constraints, which may include simple bounds on the variables, linear constraints and smooth non-linear constraints (as formulated in 6.1). It uses a SQP algorithm on the lines briefly described in section 6.4, but standard documentation reveals little how it does some of the calculations.

This function is called by a piece of Fortran code in which a number of its input parameters such as number of variables and constraints, variable and constraint bounds, and others are declared and defined. The user must supply an initial estimate of the solution close to the optimum. The user must also supply sub-routines that define the objective and constraint functions and their first derivatives. Details about how to write such a programme are given in the software user guide and will not be considered here. In addition, there are a number of parameters that can assume different values according to the problem or user needs. These parameters and the values used are described below:

- EPSAF: this must specify a bound on the absolute error in computing the objective function $F(X)$ at the initial point. It is recommended that

"EPSAF" be of the order of $\epsilon_M|F(X)|$, where ϵ_M is the machine precision. This was done by calling the NAG routine X02AJF, which returns ϵ_M. The value returned was 10^{-16}, which was then multiplied by the absolute value of $F(X)$.

- ETA: this must specify how accurately the scalar α_k (that is how far to move from X_k to X_{k+1} along the search direction p_k, see section 6.4) should approximate a univariate minimum of the merit function along p_k. The recommended value of ETA for non-linearly constrained problems is 0.9, which corresponds to a relaxed line search.

- FEATOL(j): this is an array that must contain a set of positive tolerances that define the maximum permissible absolute violation in each constraint in order for a point to be considered feasible. As the elements of "FEATOL(j)" increase, the algorithm is less likely to encounter difficulties with ill-conditioning and degeneracy. However, larger values of "FEATOL(j)" mean that the constraints could be violated by a significant amount. It is recommended that "FEATOL(j)" be set equal to the square root of the machine precision value ϵ_M returned by the function X02AJF. A value of 10^{-5} was chosen for all the constraints, since this was compatible with the "buffer zone" set by the genetic search model.

- COLD - logical: "COLD" must indicate whether the user has specified an initial estimate of the active set of constraints. If "COLD" is "TRUE", the initial working set is determined by the first QP sub-problem. The "warm" start option is particularly useful when the function is restarted at the point where an earlier run terminated. The "cold" start option (by setting "COLD" to ". TRUE. ") was used.

6.5.4 Data Acquisition and Generation

The data sets used in this set of experiments were essentially the same as the ones used for the genetic search experiments, described in section 5.5.4, so as it was possible to make comparisons in terms of efficiency and result of inspection between the non-formal approach of genetic search and the non-linear programming methods discussed in this chapter. However, considering that the version of GINO available could handle no more than 50 constraints, the data sets had to be sub-sampled, as described below, so as to comply with this limitation.

Circular profiles originally data-logged for the tests described in section 3.8 were used again. The data points were acquired by a coordinate measuring machine, as described in section 3.8.2. Each set originally consisted of 30 data points evenly but not exactly spread along the circular features. For this set of experiments, the data sets were sub-sampled so that every other data point of the original data set formed the sub-sampled data sets. The data sets were numbered $11, 12, \cdots, 15$ in section 3.8.2 and will be called here sub-11, sub-12 and so on, indicating that these are sub-sampled data sets. Linear plots of these profiles representing the original data sets (not sub-sampled ones), expressed in polar coordinates and shifted to the centroid of the points, are shown in figures 3.19 to 3.23.

Simulated data sets were also used. These were generated as described in section 5.5.4, having the same characteristics of the data sets used for the genetic search experiments but with fewer data points. Thus, for the first inspection problem (formulated as in (6.18)), three data sets representing a single circular feature were generated, each one sampled by fifteen points. The characteristics of these profiles are shown in table 5.1. The files are called sub-s1, 2 and 3. For the second inspection problem (four circular features on a plate, formulated as in (6.19) and (6.20)), five sets of four data sets were generated, simulating four circular features on a plate, each sampled by five points. Three of them were generated having the same characteristics of the ones used for the genetic

data set	lobes	amplitude	noise amplitude	radius (mm)	$L_x = L_y$ (mm)	centre distances (mm)
sub-4h01	3	0.06	0.005	10	50	100
sub-4h02	5	0.1	0.001	20	50	300
sub-4h03	7	0.02	0.002	3	5	30
sub-4h04	5	0.08	0.002	50	100	600
sub-4h05	7	0.15	0.002	5	10	50

Table 6.1: Characteristics of simulated data sets of four circular features on a plate.

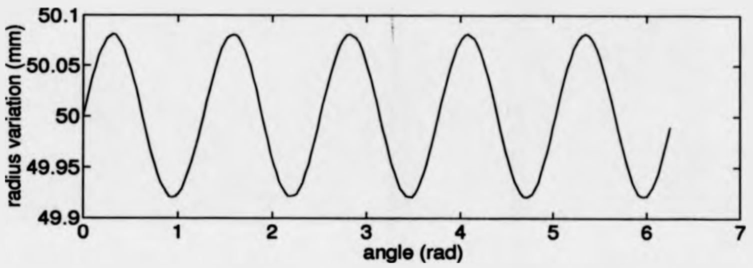


Figure 6.2: Linear plot of simulated profile (top/bottom: air/metal): data set sub-4h04.

search experiments (namely s4h-1, 2 and 3) but with fewer data points. The other two were generated following the same reasoning, that is, the magnitude of the figures were chosen so as to simulate dimensions in millimeters (mm) and errors of roundness due to, mainly, errors of form. Again the characteristics of each set of four profiles were kept the same. Table 6.1 shows the characteristics of the profiles generated for the second inspection problem, including those already described in table 5.2.

The data sets were saved in ASCII files for further manipulation and are listed in appendix A.3. Figures 5.10 to 5.15 present linear plots of the profiles previously generated (but with 200 sampled points). Linear plots of the additional profiles sampled in sub-4h04 and sub-4h05 are illustrated in figures 6.2 and 6.3.

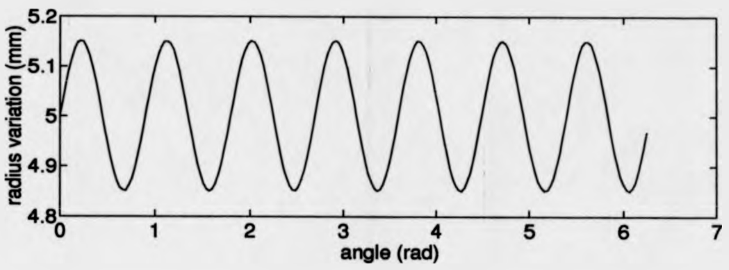


Figure 6.3: Linear plot of simulated profile (top/bottom: air/metal): data set sub-4h05.

6.5.5 Test Procedures

The dimension, location and out of roundness of the circular features were measured by fitting a minimum zone reference (MZC) to the data points. Similarly to the procedure described in section 5.5.5, the SQP NAG function E04VDF was used to minimise the zone parameter h subject to the set of constraints in (5.11). Computation of the MZC references was also carried out by GINO, and the results shown to be quite consistent with those obtained by using the SQP algorithm.

The parameters of the best fit and the error of roundness for the data sets related to the first inspection problem are shown in tables 6.2 and 6.3, for simulated and real profiles respectively. For the data sets related to the second inspection problem, tables 6.4 to 6.8, for each set of four circular profiles, show the parameters of the best fits and the distances between centres. The circular features are numbered in such a way that profiles $1, 2$ are the bottom, respectively left and right, and $3, 4$ are the top, respectively left and right ones, as in figure 5.7.

In the set of tests using the GRG2 algorithm, and unlike the tests performed using genetic search, all the possible situations were considered, that is:

- when both errors (position and roundness for the first case and position and radius for the second) are less than respective tolerances;

data set	a	b	R	$(2h)$
sub-s1	100.002	99.9990	50.0070	0.0391
sub-s2	199.9994	200.0000	10.0010	0.0355
sub-s3	5.0016	4.9998	2.0023	0.0067

Table 6.2: Parameters of MZC reference and out of roundness of roundness of simulated profiles: units in millimeters (mm).

data set	a	b	R	$(2h)$
sub-11	52.9586	60.3766	13.6854	0.0183
sub-12	152.1365	61.6507	13.529	0.0563
sub-13	92.6307	62.645	4.997	0.0495
sub-14	67.8450	21.0954	8.8093	0.01835
sub-15	76.6165	41.79125	6.4743	0.09264

Table 6.3: Parameters of MZC reference and out of roundness of roundness of real profiles: units in millimeters (mm).

data set sub-4h01						
profile	a	b	R	h	distance between centres	
					1-	4-
1	49.9993	50.0123	10.0027	0.0472	-	-
2	150.0041	50.0014	10.0038	0.0485	100.0048	100.0105
3	50.0024	150.0141	10.0027	0.0491	100.0018	99.9984
4	150.0008	150.0119	10.0029	0.0482	-	-

Table 6.4: Parameters of MZC references of the four profiles and distances between their centre positions, data set sub-4h01: units in millimeters (mm).

data set sub-4h02						
profile	a	b	R	h	distance between centres	
					1-	4-
1	50.0000	50.0004	20.0003	0.0003	-	-
2	349.9998	50.0002	20.0005	0.0003	299.9998	299.9998
3	50.0000	349.9998	20.0004	0.0003	299.9994	299.9996
4	349.9996	350.0000	20.0006	0.00026	-	-

Table 6.5: Parameters of MZC references of the four profiles and distances between their centre positions, data set sub-4h02: units in millimeters (mm).

data set sub-4h03						
profile	a	b	R	h	distance between centres	
					1-	4-
1	4.9995	4.9956	3.0016	0.0164	-	-
2	35.0053	4.9948	3.0015	0.0167	30.0058	30.0004
3	4.9997	34.9961	3.0012	0.0163	30.0005	29.9992
4	34.9989	34.9952	3.0006	0.0161	-	-

Table 6.6: Parameters of MZC references of the four profiles and distances between their centre positions, data set sub-4h03: units in millimeters (mm).

data set sub-4h04						
profile	a	b	R	h	distance between centres	
					1-	4-
1	99.9998	100.0002	50.0006	0.00015	-	-
2	700.0000	100.0001	50.0009	0.00056	600.0002	599.9999
3	99.9998	700.0004	50.0009	0.00044	600.0002	600.0002
4	700.0000	700.0000	50.0011	0.00042	-	-

Table 6.7: Parameters of MZC references of the four profiles and distances between their centre positions, data set sub-4h04: units in millimeters (mm).

data set sub-4h05						
profile	a	b	R	h	distance between centres	
					1-	4-
1	10.0004	9.9641	5.0011	0.1220	-	-
2	60.0007	9.9652	5.00074	0.1222	50.0003	49.9996
3	10.0000	59.9642	5.0004	0.1218	50.0001	50.0004
4	60.0004	59.9648	5.0001	0.1216	-	-

Table 6.8: Parameters of MZC references of the four profiles and distances between their centre positions, data set sub-4h05: units in millimeters (mm).

data set	nominal position		eccentricity of MZC
	x	y	
sub-s1	100.002	100.019	0.02
sub-s2	199.9994	200.020	0.02
sub-s3	5.0016	5.0048	0.005
sub-11	52.9586	60.3866	0.01
sub-12	152.1365	61.6757	0.025
sub-13	92.6307	62.6700	0.025
sub-14	67.8450	21.1054	0.010
sub-15	76.6165	41.8412	0.050

Table 6.9: Simulated nominal position of profiles and eccentricity of respectives MZC reference centres: units in millimeters (mm).

- when the position error is greater than its tolerance and the out of roundness or radius (either for the first or second cases) are within their tolerance limits.
- when the roundness error or radius are out of their tolerance and position error is less than its tolerance;
- when the position error and roundness error or radius are outside their tolerance limits.

For the first inspection problem, following the same strategy used for the tests described in chapter 3 (see section 3.7.4), values of $0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3$ times the out of roundness were used as roundness tolerances. For the centre position tolerance, values of $0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3$ times the twice the eccentricity of the MZC centre position from the nominal centre position (since the tolerance value is a diametrical zone) were used as examples of centre position tolerance. For each data set, the nominal centre position was fixed such that the measured centre position of the MZC reference was eccentric by an amount of the same magnitude of the roundness error, simulating dimensions in millimeters. The nominal centre position for each data set and eccentricity of respectives MZC references are given in table 6.9.

data set	R_{min}	R_{nom}	$R_{nom} - R_{min}$	L_{min}	t	L_{nom}
sub-4h01	9.9536	10.0536	0.1	99.9984	0.1	100.0984
sub-4h02	20.0000	20.1500	0.15	299.9994	0.2	300.1994
sub-4h03	2.9845	3.0348	0.05	29.9992	0.05	30.0492
sub-4h04	50.00034	50.10034	0.1	599.9999	0.1	600.0999
sub-4h05	4.8785	5.0785	0.2	49.9996	0.2	50.1996

Table 6.10: Nominal radius and distance between centres simulating four holes on a plate; units in millimeters (mm).

For the second inspection problem, following the same strategy adopted for the tests described in chapter 5 (see section 5.5.5), the nominal radius (R_{nom}) was defined by considering the features as holes, and defining the lower limit as the maximum material condition (MMC) dimension. For each simulated plate, R_{nom} was defined so as to be larger than $R_{min} = min(R_k - h_k)$ $(k = 1, \cdots, 4)$, according to the MZC references given in tables 6.4 to 6.8. The values for R_{nom} were computed so as to be larger than R_{min} by an amount of the order of 10^{-1} to 10^{-2}, simulating dimensions in millimeters. The values of R_{min} and R_{nom} for each data set are given in table 6.10. Values of $0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3$ times $R_{nom} - R_{min}$ were defined as the tolerance on radius, where in this case the tolerance is indicated as $R_{nom} \pm tolerance$. In the limiting case, when the tolerance is $\pm(R_{nom} - R_{min})$, the MMC dimension will be the minimum radius, amongst the four features, defined by the respective MZC references. The nominal distance between centres was defined by selecting the minimum distance between MZC centres of any two related features, L_{min}, and adding to it a value of about the same magnitude as the tolerance on radius (simulating dimensions in millimeters). Thus, $L_{nom} = L_{min} + t$, where t is the value added. The values for t and L_{nom} are given in table 6.10. Values of $0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3$ times t were used as the position tolerance, such that when the tolerance is t, the distance between the MZC centres for at least one of the combinations will be on the limit, that is on the edge of the tolerance zone.

Following this pattern, for each data set, related to either the first or second problem, a set of 49 different tolerance zone combinations were fed to the GRG2 algorithm implemented in GINO. In the case of the SQP algorithm, preliminary tests had shown that when no feasible point was available, the algorithm kept iterating indefinitely and it had to be stopped eventually. The strategy adopted in this case, similarly to the genetic search, was to assume infeasibility after a "certain" number of iterations (as discussed in the next section). Therefore, the number of tolerance zone combinations tested, per data set, either for the first or second problems, were reduced to those in which infeasibility was not expected, based on the results obtained using the GRG algorithm.

For both algorithm implementations, an initial solution, close to the optimum, was supplied. For the data sets representing real profiles, that is, data sets $sub-11$ to $sub-15$, the values of the least squares fits to the data sets were used as the estimate. The least square references were again computed by using the algorithm suggested by Forbes [1989] (see section 2.5.3 and appendix B). For the simulated profiles, the nominal values for the centre position and radius, that is as defined in equations 5.10 and 5.9 respectively, were used as initial estimates.

The NAG routine was run under Unix, on a SUN 4/330, with 48 Mbytes of RAM (Random Access Memory) and speed of 16 MIPS (Million of Instructions Per Second). The efficiency of the NAG function algorithm was measured in terms of number of major iterations and computation time. The computation time was estimated by running the programme under the "time" Unix command, which returns the CPU user time. The GINO programme was run under DOS, on a IBM PC computer, microprocessor 386DX, 20MHz clock, with a math co-processor. The efficiency of this programme was measured in terms of number of iterations of the GRG algorithm and computation time. The computation time was estimated by timing "manually" the iteration process.

6.6 Results and Discussion

The number of iterations of the GRG algorithm to get a pass/fail result of inspection was tabulated as shown in tables 6.12 to 6.14 and 6.16 to 6.17 for the first inspection problem and tables 6.22 to 6.26 for the second problem. The least squares solution was used as initial estimate to start the iteration process. The result of inspection was indicated by appending a minus sign to the number of iterations if it was a fail. It is worth mentioning that for data sets sub-4h01 and sub-4h05, related to the second inspection problem, the result of inspection was positive when the tolerance on radius was set smaller than its limit to avoid rejection, that is in which the radius is outside the tolerance zone for that, at a value of $0.9(R_{nom} - R_{min})$, as defined in table 6.10. This can be explained by the relatively large minimum circumscribing zones defined by very few data points, 5 for each profile. A higher number of data points would reduce the uncertainty about this (pass) inspection decision. The average number of iterations of the GRG algorithms, in the case of the first inspection problem, are shown graphically in figures 6.4 and 6.5, respectively for the data sets representing simulated and real profiles. Figure 6.6 shows it for the data sets related to the second inspection problem. The computation time of the GRG algorithm for the data sets tested is shown in table 6.11, for some tolerance zone combinations.

Regarding the tests using the GRG algorithm, the results show that the general pattern in terms of number of iterations is consistent over all data sets tested. They also show that the algorithm takes longer to determine that it is definitely impossible to reduce to zero the sum of infeasibilities than to find a feasible solution to a particular problem. This is probably due to the fact that the optimality conditions are not entirely satisfied and therefore the algorithm tests whether there is any significant improvement of the objective function over a certain number of iterations.

A comparison between the average number of iterations for the tests using simulated and real profiles, for the first problem (figures 6.4 and 6.5) shows that the results are generally consistent apart from a peak caused by data set sub-s2, and confirms that the simulated data sets (generated in the way described in sections 5.5.4 and 6.5.4) are valid for such tests.

Using the least squares solution as initial estimate has shown to be adequate, guaranteeing that a solution was found (when one existed) in a reasonable number of iterations. Thus in practice it is recommended that a least squares computation be performed before applying this method.

Despite the increase in complexity of the second inspection problem, the number of iterations required is generally the same as for the first one. Note that for the region in which the centre position error is greater than its tolerance, and radius deviation (from its nominal value) is less than its tolerance, the result of inspection is predominantly positive, therefore resulting in lower number of iterations than for the first case in the corresponding region (roundness error smaller than its tolerance). However, by comparing the computation time between both, it can be seen from table 6.11 that although the number of iterations is not altered, the amount of computation in each iteration is sensibly higher for the second case. The computation times tabulated are for a 386SX, 20MHz microprocessor. By comparison, these figures would be reduced by a factor of 1/5 when running the algorithm on a 486, 33MHz microprocessor. Therefore, the efficiency of this method represents no problem for its use in a context of on-line inspection, even for more complex problems.

Regarding the SQP algorithm, the number of major iterations to get an optimal solution, in cases where the feasible region was not empty, are shown in tables 6.15 and 6.21 for the first inspection problem, and 6.27 for the second problem. Again the least squares solution was used as initial estimate to start the iteration process. It can be seen that the number of major iterations had little variation all over the different combinations that resulted in feasibility. The computation time was very fast, and varied little, in the region of 0.5 to 1.0 second for the cases tested. The least squares solution has again shown to be adequate, guaranteeing that a solution was found (when one existed) in a reasonable number of iterations. For infeasible problems the algorithm kept iterating and therefore had to be stopped eventually. One strategy to overcome this limitation might be, as suggested in the case of genetic search methods, to assume as infeasible any problem that takes more than a specified limit of major iterations. From tables 6.15, 6.21 and 6.27, conservative number of iterations for the sort of problems tested might be in the region of 20 iterations, which represents a computation time of the order of 5 seconds on the computer used. However, when the geometrical errors are just under their tolerance limits, the iteration process may present problems due some numerical instabilities, which may be caused by Lagrange multipliers becoming extremely small, or because a direction of descent could not be found or even because the tolerances on the constraints are too tight [NAG, 1990]. If the latter situation occurs, the instability may be resolved by relaxing the tolerances on the constraints. These numerical instabilities may cause the process to terminate without a solution or to keep iterating up to the maximum number of iterations previously defined. For the problems under test, such instabilities started when the centre position error was about 0.99 of its tolerance limit (when the combined error was equal to its tolerance limit). In such marginal cases, the number of iterations required for the algorithm to come up with a solution may exceed a pre-defined threshold on the number of iterations, resulting in false negative reports, similarly to what was observed when using genetic search techniques. Therefore, in this case there again needs to be acceptable trade off between the maximum computation time and the chance of false negative reports. Further discussion and conclusions about these methods are deferred until the next chapter.

data set	number of iterations	computation time (sec.)	data set	number of iterations	computation time (sec.)
sub-s1	11	3.3	sub-15	10	5.9
sub-s2	17	6.8	sub-4h01	12	12.1
sub-s3	10	4.1	sub-4h02	13	11.2
sub-11	6	2.9	sub-4h03	9	10.0
sub-12	12	6.5	sub-4h04	13	10.4
sub-13	10	4.1	sub-4h05	12	9.9
sub-14	17	7.5			

Table 6.11: Longest execution times of the GRG2 algorithm for the data sets tested.

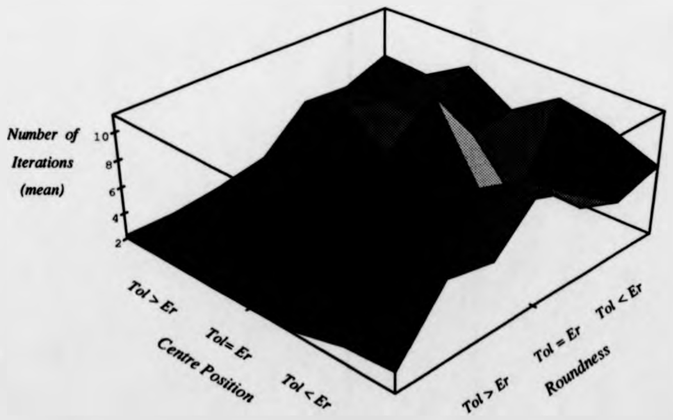


Figure 6.4: Average number of iterations of the GRG2 algorithm for inspection of roundness and centre position, over the whole set of simulated profiles.

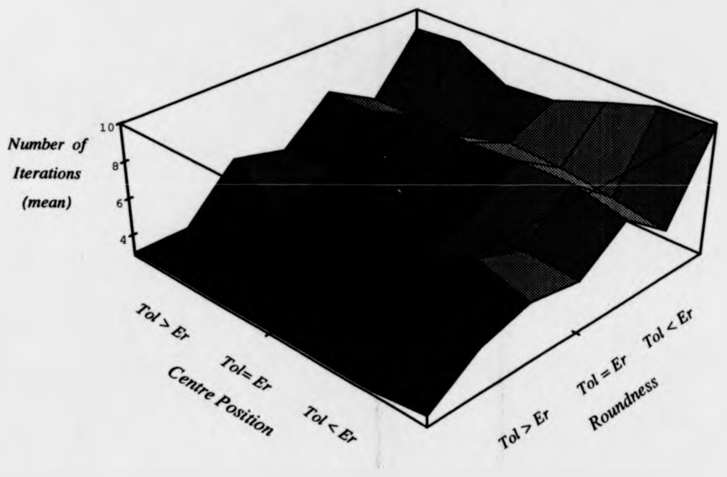


Figure 6.5: Average number of iterations of the GRG2 algorithm for inspection of roundness and centre position, over the whole set of real profiles.

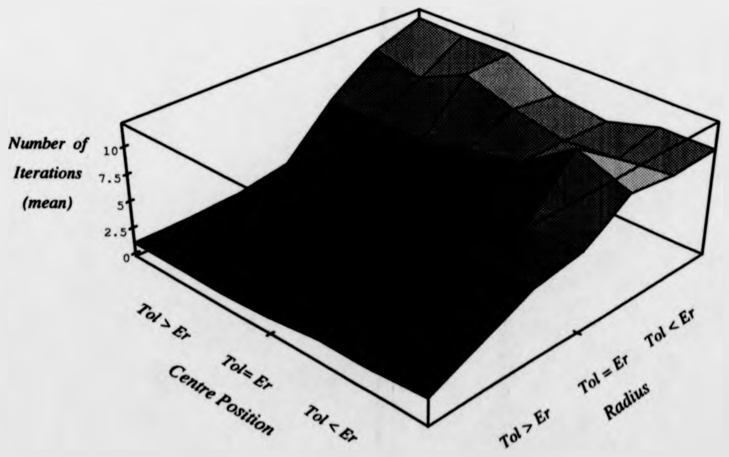


Figure 6.6: Average number of iterations of the GRG2 algorithm over the whole set of data sets for inspection of radius dimension and centre position of four holes on a plate.

roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
center position (tolerance /error)							
0.7	-3	-3	-4	-8	-8	-8	4
0.8	-3	-3	-4	-8	-8	4	2
0.9	-11	-2	-11	-2	2	2	2
1.0	-9	-5	-9	3	2	2	2
1.1	-7	-3	-4	3	2	2	2
1.2	-6	-3	-4	2	2	2	2
1.3	-6	-3	-4	2	2	2	2

Table 6.12: Search for feasible solution: number of iterations of the GRG2 algorithm to pass inspection (positive) or to fail inspection (negative); single circular feature, simulated data set sub-s1.

roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
center position (tolerance /error)							
0.7	-6	-6	-6	-10	-11	-15	6
0.8	-6	-6	-6	-11	-11	7	7
0.9	-5	-4	-4	-4	-12	4	4
1.0	-7	-9	-17	13	13	13	4
1.1	-11	-9	-9	8	6	4	4
1.2	-6	-6	-7	6	6	4	4
1.3	-6	-6	-6	5	5	4	4

Table 6.13: Search for feasible solution: number of iterations of the GRG2 algorithm to pass inspection (positive) or to fail inspection (negative); single circular feature, simulated data set sub-s2.

roundness (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
center position (tolerance /error)							
0.7	-13	-10	-13	-14	-3	1	1
0.8	-17	-7	-14	-2	1	1	1
0.9	-12	-3	-2	-2	1	1	1
1.0	-5	-6	-7	1	1	1	1
1.1	-9	-6	-3	4	1	1	1
1.2	-9	-10	-11	3	1	1	1
1.3	-10	-9	-9	3	1	1	1

Table 6.14: Search for feasible solution: number of iterations of the GRG2 algorithm to pass inspection (positive) or to fail inspection (negative); single circular feature, simulated data set sub-s3.

data set	roundness (tolerance/error)	centre position (tolerance / error)						
		0.7	0.8	0.9	1.0	1.1	1.2	1.3
sub-s1	1.0	-	-	-	-	2	2	2
	1.1	-	-	5	3	2	2	2
	1.2	-	5	5	3	2	2	2
	1.3	6	5	5	3	2	2	2
sub-s2	1.0	-	-	-	-	2	2	2
	1.1	-	-	-	3	2	2	2
	1.2	-	3	2	3	2	2	2
	1.3	4	3	2	3	2	2	2
sub-s3	1.0	-	-	-	-	2	2	2
	1.1	5	4	2	2	2	2	2
	1.2	5	4	2	2	2	2	2
	1.3	7	4	2	2	2	2	2

Table 6.15: Search for feasible solution: number of major iterations of the SQP NAG algorithm to pass inspection, for single circular features, simulated data sets.

roundness (tolerance /error) / center position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-7	-6	-5	-6	-6	4	4
0.8	-7	-6	-6	-6	-6	4	4
0.9	-7	-6	-7	-7	7	4	4
1.0	-7	-6	-7	7	7	4	4
1.1	-7	-6	-7	7	7	4	4
1.2	-7	-6	-7	7	7	4	4
1.3	-7	-6	-7	7	7	4	4

Table 6.16: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set sub-11.

roundness (tolerance /error) / center position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-13	-9	-9	-5	-4	3	2
0.8	-12	-11	-11	-5	5	2	2
0.9	-15	-10	-10	-5	4	4	2
1.0	-12	-10	-10	5	3	3	2
1.1	-10	-9	-10	5	3	2	2
1.2	-10	-9	-10	5	3	2	2
1.3	-10	-8	-10	5	3	2	2

Table 6.17: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set sub-12.

roundness (tolerance /error) / center position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-10	-3	-5	-5	-10	6	6
0.8	-10	-3	-5	-5	5	5	5
0.9	-10	-3	-5	-5	5	5	5
1.0	-10	-3	-5	5	6	5	5
1.1	-10	-3	-5	5	5	5	5
1.2	-10	-3	-5	5	5	5	5
1.3	-10	-3	-5	5	5	5	5

Table 6.18: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set sub-13.

roundness (tolerance /error) / center position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-16	-7	-15	-7	-5	5	3
0.8	-17	-7	-15	-8	-6	3	2
0.9	-6	-7	-15	-8	5	3	2
1.0	-7	-8	-15	5	5	3	2
1.1	-8	-8	-16	7	3	3	2
1.2	-15	-8	-11	5	3	2	2
1.3	-16	-8	-11	5	3	2	2

Table 6.19: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set sub-14.

roundness (tolerance /error) / center position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-3	-3	-3	-5	-5	-8	2
0.8	-3	-3	-3	-5	-12	5	2
0.9	-6	-6	-3	-5	-7	2	2
1.0	-3	-6	-3	4	3	2	2
1.1	-3	-3	-3	3	3	2	2
1.2	-3	-3	-3	3	3	2	2
1.3	-3	-3	-3	3	2	2	2

Table 6.20: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set sub-15.

data set	roundness (tolerance/error)	centre position (tolerance / error)						
		0.7	0.8	0.9	1.0	1.1	1.2	1.3
sub-11	1.0	-	-	-	-	5	5	5
	1.1	-	-	5	5	5	4	2
	1.2	6	6	5	5	5	5	4
	1.3	6	6	5	5	5	5	4
sub-12	1.0	-	-	-	4	4	4	4
	1.1	-	4	4	4	4	4	4
	1.2	4	4	4	4	4	4	4
	1.3	4	4	4	4	4	4	4
sub-13	1.0	-	-	-	-	2	2	2
	1.1	-	2	2	2	2	2	2
	1.2	2	2	2	2	2	2	2
	1.3	2	2	2	2	2	2	2
sub-14	1.0	-	-	-	-	2	2	2
	1.1	-	-	5	5	4	4	3
	1.2	6	6	5	5	5	4	4
	1.3	6	6	6	6	5	5	5
sub-15	1.0	-	-	-	-	4	4	4
	1.1	-	-	-	4	4	4	4
	1.2	-	5	5	4	4	4	4
	1.3	5	5	5	4	4	4	4

Table 6.21: Search for feasible solution: number of major iterations of the SQP NAG algorithm to pass inspection, for single circular features, real data sets.

radius (tolerance $/R_{nom} - R_{min}$) / center position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-9	-8	-9	6	7	3	2
0.8	-8	-8	-12	6	4	2	2
0.9	-8	-8	-8	4	3	2	2
1.0	-12	-8	8	2	2	2	2
1.1	-12	-8	5	2	2	2	1
1.2	-12	-8	5	2	2	1	1
1.3	-11	-9	5	2	2	1	1

Table 6.22: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set sub-4h01.

radius (tolerance $/R_{nom} - R_{min}$) / center position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-11	-9	-12	-6	-8	-10	2
0.8	-13	-9	-10	-6	-8	2	2
0.9	-8	-9	-8	-6	2	2	2
1.0	-9	-10	-9	2	2	2	2
1.1	-15	-17	-9	2	2	1	1
1.2	-15	-11	-9	2	2	1	1
1.3	-15	-11	-8	2	1	1	1

Table 6.23: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set sub-4h02.

radius (tolerance $/R_{nom} - R_{min}$) / center position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-9	-8	-8	-9	4	3	2
0.8	-8	-8	-9	7	3	2	1
0.9	-8	-7	-8	5	2	2	1
1.0	-6	-6	-5	2	1	1	1
1.1	-6	-6	-5	2	1	1	1
1.2	-6	-6	-5	2	1	1	1
1.3	-6	-6	-5	2	1	1	1

Table 6.24: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set sub-4h03.

radius (tolerance $/R_{nom} - R_{min}$) / center position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-11	-13	-12	-8	-8	3	3
0.8	-11	-10	-12	-8	-12	2	2
0.9	-9	-9	-9	-8	2	2	2
1.0	-9	-11	-8	2	2	2	1
1.1	-10	-11	-8	2	2	1	1
1.2	-10	-11	-8	2	1	1	1
1.3	-12	-11	-9	2	1	1	1

Table 6.25: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set sub-4h04.

radius (tolerance /$R_{nom} - R_{min}$) / center position (tolerance /error)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
0.7	-8	-9	-9	7	5	2	2
0.8	-8	-9	-10	8	5	2	2
0.9	-8	-9	-9	6	4	2	2
1.0	-8	-12	8	2	2	2	1
1.1	-12	-12	5	2	1	1	1
1.2	-12	-12	5	2	1	1	1
1.3	-12	-11	5	2	1	1	1

Table 6.26: Search for feasible solution: number of iterations to pass inspection (positive) or to fail inspection (negative), data set sub-4h05.

data	radius	centre position (tolerance / error)						
set	(tolerance/$R_{nom} - R_{min}$)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
sub-4h01	0.9	-	-	-	2	2	2	2
	1.0	2	2	2	2	2	2	2
	1.1	2	2	2	2	2	2	2
	1.2	2	2	2	2	2	2	2
	1.3	2	2	2	2	2	2	2
sub-4h02	1.0	-	-	-	1	1	1	1
	1.1	-	-	2	2	2	2	2
	1.2	-	2	2	2	2	2	2
	1.3	2	2	2	2	2	2	2
sub-4h03	1.0	-	2	2	2	2	2	2
	1.1	3	2	2	2	2	2	2
	1.2	3	3	2	2	2	2	2
	1.3	3	3	3	3	2	2	2
sub-4h04	1.0	-	-	-	2	3	3	3
	1.1	-	-	3	2	3	3	3
	1.2	4	4	3	3	3	3	3
	1.3	4	4	4	3	3	3	3
sub-15	0.9	-	-	-	3	2	2	2
	1.0	-	3	2	2	2	2	2
	1.1	3	3	2	2	2	2	2
	1.2	3	3	3	3	3	3	3
	1.3	3	3	3	3	3	3	3

Table 6.27: Search for feasible solution: number of major iterations of the SQP NAG algorithm to pass inspection, for four holes on plate, simulated data sets.

Chapter 7

Conclusions and Suggestions for Further Work

7.1 Introduction

This work has proposed an approach to the inspection of geometric tolerances by which the data points representing the features to be inspected are checked for containment within a zone defined by the design geometric tolerance specifications, in contrast to the usual approach of first defining best-fit references to data and then checking whether the deviations from the references and their parameters satisfy the design tolerance constraints. A selection of important cases have been considered explicitly, including roundness, position of circular features, flatness, squareness and the combination of several circular features.

It has been shown that determining whether sets of data points representing some features meet their tolerance specifications is equivalent to determining whether a feasible region is defined by the combination of sets of linear or non-linear constraints derived systematically from the data points and the geometric characteristics under consideration. This approach is superior to that based on defining best-fit references when the combination of two characteristics of a feature is to be inspected, such as roundness and centre position for example.

Several numerical methods for assessing problem feasibility have been thoroughly examined. Linear methods were investigated in chapters 3 and 4, that are applicable if the constraints are naturally linear or if there are valid approximate mathematical models to the inspection problems. Non-linear methods were then examined for inspection situations in which linearisation is not appropriate. Chapter 5 investigated the use of genetic search techniques as opposed to formal non-linear techniques such as the generalised reduced gradient (GRG) and sequential quadratic programming (SQP), which are explored in chapter 6.

It remains here to bring together the various strands of these investigations and hence to evaluate the extent to which each may influence the next generation of computer aided metrology software. First a comparison is made in terms of practicability and accuracy between the non-linear methods looked at in chapter 6 and genetic search techniques. There follows some further comparison between linear and non-linear techniques. The relative overall merits of the methods are then considered and final conclusions presented in the penultimate section, followed by some suggestions for further work.

7.2 Comparison of Non-linear Techniques

All the types of genetic search algorithm and non-linear programming techniques under consideration seem from theoretical evidence and from the tests performed in sections 5.5 and 6.5, to offer valid techniques for geometric tolerance inspection. As far as practicability and accuracy of results are concerned, the tests indicate that each has good and bad features that must be considered and balanced.

In terms of computation times, the three algorithms performed very well for cases of easy acceptance for the sort of problems under test. The tests using the GRG algorithm were run on a 386, 20 MHz micro-computer, and therefore the figures quoted in table 6.11 have to be multiplied by a factor of about 1/5 to obtain their equivalent execution times running on the SUN computer used for testing the other two algorithms. The GRG and the SQP algorithms produced a result (in cases of easy acceptance) in about the same time, while the genetic algorithm was a bit slower. However, the tests of the genetic algorithm were more severe, as larger data sets were used in this case: 30 and 60 data points for the first and second problems respectively, as opposed to 15 and 20 data points used for the tests of the GRG and SQP algorithms. The reason for using smaller data sets was that the GRG GINO implementation available [The Scientific Press, 1992] was able to process up to 50 constraints only, and consequently the data sets had to be sub-sampled for such tests. Considering that each data point generates two constraints, the number of constraints in the genetic search was as many as two and three times that used for the GRG and SQP tests. The number of parameters of the problems tested was increased from 3 to 8 (for the first and second problems respectively). The computation time for the different sizes of the problems tested increased approximately by a factor of 3 for the genetic algorithm, while for the GRG and SQP algorithms it increased approximately by a factor of 3 and 2.5 respectively. For the SQP and GRG algorithms, the number of operations performed in each iteration is roughly proportional to (perhaps increasing with a power slightly greater than unity) the number of parameters and constraints (which determine the size of the matrices to be manipulated), and the degree of non-linearity of the constraint functions (which determines the difficulty of the line search process). Unlike formal non-linear techniques, the genetic search is not so affected by the size of the problem or the degree of non-linearity of the constraints, except that it is generally more "difficult" to find a feasible solution as the dimension of the parameter space increases and it is more constrained. Although the stochastic processes inherent to genetic search make it difficult to predict behaviours theoretically, a combination of general arguments and experimental evidence suggests that the computational effort of the genetic algorithm may increase less rapidly than those for GRG and SQP algorithms as problem complexity increases.

The main advantage of the GRG algorithm over the genetic and SQP algorithms is that it yields a definitive answer of whether the problem is infeasible or not in a reasonable computation time and presents negligible instability or uncertainty of result when the geometric errors are just under their tolerance limits. The uncertainty of result of the GRG algorithm would be caused essentially by the tolerance or buffer zone defined around the constraints. However, for typical buffer values needed here, of the order of 10^{-5} of the tolerance, the error introduced would be of the order of 0.001 %, which in practice has no significance. On the other hand, both the SQP and the genetic search algorithms had to be halted after a "certain" number of iterations when there was no feasible solution, since the algorithms could not detect such condition and kept iterating indefinitely. The problem with such a measure is that, if the computation time has to be under "certain" limits, there may be some marginal feasible cases for which the allowed computation time is not enough to differentiate them from the truly infeasible cases, resulting in acceptable components being rejected by such inspection methods. The genetic algorithm has an advantage over SQP in the sense that it is possible to detect cases of clear rejection at an early stage, therefore saving computation time.

For the cases tested, when limiting the computation time to about 15 and 5 seconds (on the computer used) for the genetic and SQP algorithms respectively, large majority of tests failed to solve in the time available when the geometric error was within 2 and 1 % of its tolerance zone from limit for the genetic and SQP algorithms respectively, while such failures were very rare at greater than 5 and 2 % respectively. On a tolerance zone of 0.1 mm, the uncertainty region is no more than a few micrometers and directly comparable with typical measuring machine accuracies (e. g. the CMM used for these tests has an accuracy of 5μm on each axis). Since the error due to algorithmic approximation is independent of and generally similar to or smaller than the error due to the machine inaccuracies, the resulting vectorial summation of these two errors (the square root of the sum of the square of the errors) will not be a great deal larger than the error due to machine inaccuracy alone. Thus the number of false decisions will not be much different from that which would occur with a "perfect" algorithm and a real (not perfect) CMM. Moreover, the algorithmic errors can be controlled to lie almost entirely in the false positive (or false negative) region while those of the instrument will be randomly scattered in both regions. Although these algorithms lead to some marginally good components being rejected, it is reasonable to expect that there exists, from the engineering and economics point of view, an acceptable trade off between the maximum computation time and the risk of false negative reports resulting from adopting such techniques.

Further conclusions about these methods will be drawn after considering a comparison between linear and non-linear techniques.

7.3 Comparison of Linear and Non-linear Techniques

The main disadvantage of linear techniques is that they are only applicable when the formulation of the inspection problems can be made linear in its parameters. On the other hand, non-linear techniques can be applied to any situation, no matter whether the formulation of the problem results in linear or non-linear constraint functions. However, the extra computational effort resulting from using such techniques is unnecessary when the formulation of the inspection problem can be made linear without significantly compromising the precision of the inspection process, as is the case of inspection of some important characteristics such as roundness, flatness, squareness and parallelism amongst others. In addition, as the size of data sets grows, the computational effort resulting from formal non-linear techniques increases at a greater rate than with linear ones, and consequently they are less affordable.

The Simplex method of linear programming is sensitive to the size of the data sets, mainly due to artificial variables added to the problem (as many as the number of constraints), which increase the number of iterations and the number of operations per iteration. By contrast, for the Fletcher algorithm, the number of iterations is not affected by the size of the data sets (neither are the sizes of the matrices to be processed). For example, the tests in section 3.7, for inspection of roundness and centre position, were performed using large data sets (513 data points), from a roundness measuring machine. Using specially derived start-up procedures, computation times hardly exceeded one second, considerably shorter than the ones recorded for the tests performed in chapter 5 and 6, using non-linear techniques, for data sets with few tens of data points.

Large data sets result from a number of surface probing techniques, such as profilometry, as is the case of roundness measuring machines or CMMs fitted with analogue probes, and vision systems, when the surface or profile is sampled by image analysis. The use of such techniques is expected to increase and with it the need to process large sets of inspection data in an efficient way. The Fletcher algorithm provides a valid, rapid and significant approach for cases in which linearisation is possible and large data sets are to be processed. It far out-performs GRG and SQP under such conditions. With minimal data sets from CMMs, there is still a speed advantage, although its practical significance is smaller since the other methods are fast enough for many practical applications.

7.4 Conclusions

Table 7.1 summarises in general form some of the advantages and disadvantages of the methods looked at. Referring to the headers on table 7.1, the generality of use of a method is whether it can cope with linear as well as non-linear functions. The ease of operation of a software implementation of a method is understood as whether it needs, at an initial stage, to set parameters, apart from the geometric tolerancing information, which will interfere with the performance of the algorithm. The certainty of result of a method is an indication of the chances of not having false "pass" or "fail" inspection results. Another important feature of a method is whether it can cope well with large data sets.

A particular algorithmic implementation is considered here generally suitable for on-line inspection if it can produce a result within about 5 seconds, running on a computer of moderate speed expected to be dedicated to a measuring machine, for example a 486, 33 MHz micro-computer. This execution time is generally less than that of data acquisition cycles of, for example, a coordinate measuring machine or a roundness measuring instrument. When a vision machine is used, although the data acquisition time itself is very low, the time for loading/unloading the component from the measuring site (or clearing the field for image acquisition in case of "on-site" inspection) can generally be assumed to be in the region of few seconds. The information about errors is whether it is possible using the same software implementation or the same method to measure the errors of the workpiece being inspected, in addition to checking whether they satisfy tolerance requirements.

Of the methods considered, the Simplex and Fletcher algorithms are for linear programming and in this sense are limited to those cases in which linearisation is possible. On the other hand, genetic search and non-linear programming methods have general use, as they can be applied to linear as well as non-linear inspection problems.

method	generality of use	ease of operation	certainty of result	large data sets	on-line inspec.	information about errors
Simplex	R	G	G	P	R/G	G
Fletcher	R	G	G	G	G	P
Gen. S.	G	R	R	R	P/R	P
GRG	G	G	G	P	R/G	G
SQP	G	G	R	P	R	G

Table 7.1: General appreciation of the methods investigated: G - good; R - reasonable ; P - poor.

Any software implementation of the linear programming methods would be quite easy to operate, as long as interfaced with a friendly front-end, since no decision has to be made about any parameter that might affect the efficiency or accuracy of the algorithm. This is not quite the same when genetic search or SQP methods are used. Apart from the parameters intrinsic to the algorithms that have standard recommended values which could be pre-set by the manufacturer, other parameters would have to be defined such as threshold values and buffer zones, which are in general application dependent. However, as the definition of these parameters is determined by quality control policies and production planning schemes, the definition of such parameters could be taken at a managerial level. Therefore, although there would be a more complex initial set-up phase probably requiring professional input, no greater skill than usual would be required of a machine operator.

Non-linear programming methods such as generalised reduced gradient (GRG) methods also need a number of parameters to be set by the user. However, there exist recommended values for this parameters which perform well for most cases, including the ones under consideration. Therefore such parameters could be preset to default values, making any software implementation of such methods quite easy to operate, again provided that it is interfaced by a suitable front-end.

Regarding implementation of these methods, a common problem is how to automate the process of formulating and feeding a range of inspection problems to such algorithms. The details of constraint sets that must be set up depend critically upon the geometry of the features and will vary somewhat even between closely related problems such as inspecting position and size of two, three or more circular features positioned in different ways, or combining different characteristics of features, to be inspected simultaneously. If each particular inspection case had to be treated at the level used in this work, the methods would be impractical for all but research or special, high-cost applications. The sort of features and characteristics that need to be inspected are defined in standards [BSI, 1990], and for these, the type of constraint function generated by each data set or by the tolerance constraints could, in principle, be held in the "library" of a compiler or knowledge based system which would "compose" the inspection problem as defined by the user. Although this is not within the scope of the work presented in this thesis, it must be addressed if a more general and automated inspection system is considered. General evidence of the structure of inspection problem formulation, as examined here, and of for example, the rapid recent progress in implementing algebraic manipulation on small computers all reinforces the plausibility of useful "features inspection compilers" being developed.

As discussed, in marginal cases when the errors are very close to the tolerance values, there exists an uncertainty about a "pass" or "fail" result of inspection. The certainty of a "pass/fail" result from a linear programming method, will be dependent on two factors: the machine precision and the level of approximation of the linear formulation of a non-linear inspection problem. In general numerical errors will be very small, while linearisation errors are deemed tolerable a priori, so the results of such algorithms may be regarded as very accurate. For the genetic and SQP algorithms, the main factor of marginal uncertainty is the limit on the number of iterations to find a feasible solution. In some cases, there may be a feasible solution which is not found because it would take more computation time to find it than has been allowed by the defined threshold. The level of uncertainty will depend on the the maximum computation time affordable for inspection (and so on the computation speed that can be afforded) and on the width of the tolerance or buffer zone on the constraints. The buffer zone could be taken off the geometric tolerance zone, with a slight possibility then of false negative results in marginal cases. Although both algorithms present such a problem, the tests have shown that the chances of false result for the SQP algorithm are slightly less than for the genetic algorithm, making the former a better choice in this sense.

In the case of the GRG algorithm, although a number of approximations are made by the algorithm, as described in section 6.5.2, in practice the main factor of uncertainty will be the buffer or tolerance zones on the constraints, which, as discussed, will be of very little significance in practice.

The Simplex method, in contrast with the Fletcher algorithm, is expensive computationally when processing large data sets. The GRG and SQP algorithms are by nature more complex than linear programming and also directly affected by an increase in the size of data sets. The genetic algorithm also leads to an increase of the computational time (by increasing the number of iterations), although not by as much as formal non-linear programming techniques.

Considering the computation speed of computers presently likely to be dedicated to a measuring machine, and from the considerations made above, the Simplex method can be considered adequate for on-line inspection, if small data sets (up to a few tens of points) are used (for example the cases tested in section 4.4). The same can be said about the GRG algorithm. The Fletcher algorithm is adequate for on-line inspection, no matter what size of data set is used. In the case of the genetic and SQP algorithms, their adequacy for on-line inspection is mainly dependent on the acceptable level of uncertainty of result of marginal cases. For the cases tested, and for the level of uncertainty considered, they are on the edge of what can be regarded as adequate for on-line inspection. The SQP method can be considered slightly better than the genetic algorithm. Ideally, faster computers would be required for the genetic and SQP algorithm, so as to minimise the risk of false "fail" results. It is, of course, likely that such computers will be available at suitable costs in the near future.

Of the methods considered, the Fletcher algorithm has the disadvantage that it does not indicate an optimal point in cases where a feasible region exists. As a consequence of that, no information can be obtained about the errors generated by the manufacturing process. The genetic search method has the same limitation, not because it is not possible to obtain an optimum point but because it would take too long. The other methods under consideration do not present such limitation and therefore have the advantage of being suitable not only for inspection purposes but for checking for drifts of the manufacturing process.

To conclude, the Fletcher algorithm offers a very useful tool for inspecting the combined effect of form, position and orientation errors of geometric features, especially when the features are sampled by large data sets, as long as the formulation of the inspection problem involves only linear functions. There are inspection problems which are naturally linear and others which can be well-linearised, and for these this algorithmic approach is very attractive. The GRG algorithm brings together the advantages of generality of use, ease of operation, certainty of result and information about errors. It is also adequate for on-line inspection with small to moderate data sets. Therefore, the combination of the GRG and the Fletcher algorithm in one software package can adequately cope with a large range of inspection problems and different sizes of data sets.

The main drawback of genetic and SQP algorithms is that some uncertainty of result in marginal cases may have to be accepted, in order to keep the inspection cycle as short as required. The computation times for the SQP algorithm are shorter than those of the genetic algorithm, and in this respect, the SQP algorithm can be considered the better. Although for the sort of problems and data sets used for tests, the computational times of the GRG and SQP were shorter than those of the genetic algorithm, it has been seen that the rise in computational effort of the genetic algorithm in response to an increase in problem size and data sets is less than those of formal non-linear techniques. Therefore, it may well be that, for more complex problems and large data sets, the genetic algorithm offers a better alternative. At present there is insufficient data to do more than speculate on this point. As faster low-cost computers become available genetic search will become more practicable but also the size of the problems that can be handled by GRG in a reasonable time will increase. If constraint feasibility methods become widely used for inspection, then problems of greater geometrical complexity will be addressed and a trend towards increasing number of constraints and larger sets will continue. Thus, genetic search should certainly not be discounted although for the immediate future GRG is recommended as the preferred method for situations where the linear techniques developed here cannot be used.

7.5 Suggestions for Further Work

It is envisaged that the work presented in this thesis can be followed up on three levels. On a first level, the approach discussed here can be generalised to include a wide range of inspection situations. Thus, in addition of the geometric features and characteristics already discussed, others could be included, such as: cylindricity, straightness of a cylinder axis, straightness of a general axis, squareness or parallelism of a cylinder axis related to plane, sphericity and so forth. Considering that the number of parameters required to represent for example a cylinder (8), a sphere (4) or a line in three-dimensions (4) are in the same region of those of the inspection problems discussed in this thesis, it seems likely that such implementations would be satisfactory for on-line inspection.

In addition, software implementations of the algorithmic methods considered here could be developed in a form appropriate for routine application. Such software would include an interface with a measuring machine (e. g. a coordinate measuring machine) for data acquisition, and a user friendly front-end for tolerance information input, any other input required, and output of results. The crucial aspect is to devise ways to automate the process of formulating different inspection problems (constraint generation) based on the input, by a machine operator, of the basic features and tolerances to be inspected.

On a second level, work can be carried out on whether other numerical methods not considered in this work can be applied and, if so, on their relative merits when compared to the ones already discussed in this work. This is the case, for example, of linear and non-linear minimax Chebyshev approximation, as suggested by Forbes [1992] or other methods, such as penalty function methods, that transform the constrained problem into a sequence of non-constrained problems.

On a third and more complex level, an investigation can be pursued on possible ways of integrating the inspection activity with the design stage, such that the geometric errors of different features can be inspected by direct comparison with their CAD data files.

Based on the formalisms developed here and on the experimental results obtained, the likelihood of practically useful systems being produced is high enough to justify further work in all of these areas.

Appendices

Appendix A

Data Set Listings

A.1 Talyrond Data Sets

Data sets 1 to 10 represent profiles from a roundness measuring instrument. They should be read horizontally, row by row. Each element in the array (of 513 elements) corresponds to the radial deviation of the profile, where the angular position of each point is implicitly given as $2\pi i/512$ where i is the element index in the data array.

Data Set 1

1.0795	0.6350	0.8255	0.9525	1.0160	1.7780	1.3970	0.5080	0.1905
-0.0635	-0.1905	0.3810	0.8255	1.3970	0.6350	0.8255	1.1430	1.4605
1.2065	1.7145	1.3970	0.3810	0.5715	1.3970	2.7305	1.9050	1.2700
1.4605	2.4130	2.0955	2.3495	1.6510	1.4605	1.9685	0.5715	1.5875
2.1590	1.8415	2.5400	7.4295	5.0165	3.0480	1.8415	1.5240	4.0005
3.2385	3.7465	3.1750	4.1275	3.3655	3.6195	3.3655	3.3020	3.2385
2.9845	4.0640	4.1910	4.3180	3.3655	4.1910	3.6195	2.9845	3.3020
4.2545	3.9370	3.1750	2.4765	3.7465	3.8100	2.7305	3.1115	3.6830
3.4925	3.8735	2.1590	3.1115	3.0480	3.4290	3.8100	3.8735	3.8735
3.4290	3.0480	2.7940	2.7305	2.2860	1.9685	2.9210	2.1590	2.7940
2.0320	1.7780	2.1590	1.3970	1.8415	2.0955	2.4765	2.7940	0.7620
-0.8890	-0.0635	0.5715	0.7620	0.6985	1.0795	1.5875	1.9685	1.5875

0.9525	0.6350	-0.0635	-1.1430	0.7620	0.3175	-0.8890	-0.1905	-0.0635
0.0635	-1.9050	-2.2225	-2.6670	-0.9525	-0.5715	0.0000	-0.6350	-1.4605
-0.9525	-1.4605	-2.3495	-2.6035	-2.3495	-2.3495	-3.0480	-3.0480	-3.3655
-3.9370	-1.9685	-2.2225	-2.2860	-3.2385	-2.0955	-2.4765	-3.7465	-2.4130
-2.7940	-1.9685	-3.9370	-4.5085	-1.9685	-1.8415	-3.0480	-3.1115	-3.0480
-2.7940	-2.6035	-2.7940	-4.1910	-3.6830	-5.2705	-5.5245	-3.8735	-3.8100
-5.9055	-5.1435	-5.3340	-5.0165	-3.9370	-3.8100	-3.8100	-4.0005	-4.6355
-3.4925	-4.8895	-3.4925	-2.3495	-2.6670	-2.1590	-1.5240	-1.9685	-1.9685
-2.4130	-1.7780	-3.6195	-2.4765	-2.0955	-3.8735	-4.8895	-3.1115	-4.2545
-4.8895	-5.9690	-3.0480	-0.1270	-2.6670	-2.3495	-2.5400	-1.5875	-2.7305
-1.3970	-1.9050	-0.1270	-0.1905	-0.8255	-1.0160	-0.8255	-0.5080	0.6350
1.0160	0.0000	0.2540	-1.7145	-0.3810	0.3810	-0.3810	-1.2065	-1.0160
0.1905	-1.0795	-1.2700	-1.3335	-1.6510	-1.3970	-0.9525	-0.8255	-0.5080
-0.5080	0.1905	1.3335	0.2540	1.5875	0.5080	2.1590	2.6670	1.7145
0.5715	0.7620	1.3335	2.7305	1.4605	1.7145	1.7145	1.8415	0.1270
2.0320	2.0320	2.9210	3.7465	4.0640	2.8575	1.7145	1.6510	2.6670
2.0320	1.9050	2.0320	3.4290	2.7305	2.1590	2.9210	3.1115	0.5080
1.5875	2.2225	3.3655	2.3495	1.2065	2.6035	2.6035	0.5080	2.3495
3.1750	2.6035	2.8575	3.3655	2.1590	2.2860	2.5400	2.7305	2.2225
2.2860	1.4605	1.3335	1.4605	1.9050	2.2225	2.1590	3.1750	3.8100
3.3020	2.4765	2.4130	2.6670	3.7465	2.8575	1.7145	2.4130	2.7305
1.9050	0.9525	2.2860	2.2860	2.0320	1.2700	1.4605	1.9050	1.7780
2.2225	1.5875	3.0480	2.7940	2.4130	3.6195	3.4290	3.4925	4.0640
4.0640	3.2385	4.7625	4.8260	3.4925	2.9845	3.5560	3.6830	4.6990
5.6515	4.8260	3.3020	3.9370	3.8735	4.6355	4.5720	4.3815	5.1435
3.7465	3.0480	4.0005	4.9530	3.3020	4.6990	4.1275	5.2705	5.6515
4.2545	5.0165	5.9055	5.7785	6.2230	6.0325	4.3180	5.9055	6.0960
6.9850	6.0325	6.8580	6.5405	4.8895	6.0960	7.1755	6.1595	7.9375
6.4135	6.7310	7.4295	6.9215	7.3025	5.9055	7.6200	7.6200	8.2550
7.8105	8.2550	6.9850	8.3185	8.1280	8.8265	9.2075	8.7630	9.4615
9.9060	9.5250	9.3980	10.5410	9.2075	9.3980	9.6520	8.1915	9.5250
10.7315	9.2710	9.0170	10.4140	11.0490	10.4140	9.5250	9.9695	9.0170
9.1440	10.3505	10.0330	9.5885	9.9060	10.9855	10.4140	10.7950	11.5570
10.9220	9.4615	10.7315	11.6205	10.6680	11.2395	11.1760	10.8585	10.7315
9.7155	10.2235	9.2075	11.6205	10.7315	11.1125	10.6045	11.0490	9.9060
9.7155	9.4615	9.2710	9.0170	9.5250	8.6995	8.7630	8.0010	10.0965
7.1755	7.1120	7.7470	8.3185	8.9535	9.2075	9.5885	8.6995	8.7630
8.6995	8.5090	7.6200	8.5725	8.6360	9.2710	7.7470	8.1280	7.6200

7.8740	7.8105	8.1915	7.7470	7.6835	7.1120	8.5090	6.9850	7.2390
6.7945	6.2230	5.8420	6.0325	5.3340	6.0325	5.2070	5.0800	4.9530
5.9690	5.0165	5.0165	5.7150	5.2705	4.5085	3.4290	4.8895	3.6195
3.4925	2.3495	4.1275	3.6830	3.8100	2.8575	3.2385	2.5400	2.7940
2.4130	2.9210	2.9845	2.4130	1.7145	2.3495	0.3175	0.4445	2.7940
0.1905	1.6510	2.0320	1.0160	0.7620	0.0000	-0.4445	1.0160	0.8890
1.3970	0.0000	1.3335	0.5715	2.0955	2.2225	1.9050	2.3495	2.4130

Data Set 2

4.8895	6.6675	5.5245	5.7785	6.0325	4.8895	6.2865	6.3500	6.6675
4.8260	5.9055	6.8580	6.7310	5.6515	6.7945	6.7945	7.2390	5.9055
5.0165	3.8735	6.3500	5.7150	4.0640	4.6990	2.4130	6.0960	5.2705
3.9370	3.2385	1.9685	3.5560	2.1590	5.2705	4.8895	4.1275	2.0320
2.3495	4.0005	3.1750	5.0165	4.3180	2.7940	3.3020	3.8100	2.2860
3.8100	3.3655	3.9370	2.7940	3.9370	3.5560	-0.1905	2.7940	1.1430
0.2540	2.9845	1.1430	2.8575	0.9525	0.5715	-2.1590	0.8255	0.2540
0.3175	-0.3175	-0.5715	-0.2540	-1.7145	-0.4445	-0.5715	-2.2225	-0.8890
-0.8890	-1.3970	-2.6035	-4.6355	-2.9845	-2.8575	-4.3180	-3.7465	-3.4290
-3.6830	-4.4450	-5.7785	-3.7465	-5.3975	-6.2230	-4.9530	-8.8900	-6.2865
-7.1120	-5.3340	-6.3500	-5.9055	-6.5405	-6.6040	-6.6675	-6.3500	-6.0325
-5.7150	-5.7785	-5.6515	-5.2705	-5.1435	-5.0800	-4.7625	-4.4450	-4.1910
-4.1275	-4.0640	-3.5560	-3.2385	-3.3655	-2.9845	-2.4130	-2.2860	-1.9685
-1.3335	-1.8415	-1.5875	-1.7780	-1.3335	-1.3335	-1.0795	-0.9525	-0.8255
-0.8255	-0.4445	-0.2540	-0.3810	-0.0635	0.0635	0.1905	0.3810	0.8255
0.6985	0.2540	1.4605	1.4605	1.7145	2.0955	1.6510	1.8415	1.8415
1.8415	1.9685	1.9050	1.7780	1.9685	1.9050	2.2225	2.2225	2.3495
2.4130	2.4130	2.6035	2.3495	2.4130	2.5400	2.6670	2.9845	2.5400
2.7305	2.4130	2.5400	2.3495	2.0320	2.0955	2.1590	2.0320	2.2225
2.1590	2.2225	1.6510	1.7780	1.7780	1.5240	1.4605	0.8890	0.4445
0.6985	-0.1270	-2.1590	-0.0635	-1.2700	-0.5080	-0.8890	-0.8255	-0.9525
-1.5240	-3.1115	-1.3970	-2.1590	-2.8575	-3.3655	-3.4290	-4.5085	-3.6195
-5.3340	-4.3815	-3.1750	-4.3815	-3.4925	-3.6195	-5.2070	-5.2705	-5.0165
-5.2705	-5.2705	-5.9055	-6.6675	-6.4770	-6.4135	-6.7310	-7.1755	-8.8265
-7.4295	-7.5565	-8.0645	-8.1915	-8.0645	-7.3025	-7.4295	-6.4770	-6.0960
-5.7785	-5.3975	-5.1435	-4.9530	-4.7625	-4.6355	-4.6355	-4.1275	-3.8100
-3.6830	-3.5560	-3.3020	-3.1115	-3.0480	-2.7305	-2.4130	-2.4130	-2.0955
-1.9050	-1.7145	-1.5240	-1.3335	-1.0795	-0.9525	-0.6985	-0.5715	-0.5080

-0.3175	-0.1905	-0.1270	0.0635	-0.0635	0.1905	0.3810	0.3810	0.8890
0.8890	1.1430	0.9525	1.4605	1.7145	1.9050	1.7780	2.1590	2.2225
2.3495	2.5400	2.6670	2.2225	2.9210	2.9845	3.3020	3.5560	3.4925
3.5560	3.6195	3.3655	3.6830	3.5560	3.8100	3.8100	3.8735	3.8735
3.8735	4.0005	3.7465	3.9370	3.9370	4.0005	4.1275	3.6195	3.7465
3.7465	4.0640	4.2545	4.1910	4.1910	4.3180	4.1275	4.1910	4.3815
4.4450	4.1910	4.5085	4.2545	4.2545	4.5720	4.3815	4.0640	3.9370
4.3815	4.1910	4.2545	4.1910	4.1275	4.0005	4.1910	4.1910	4.1275
4.0640	4.0640	3.8100	3.6830	4.0640	4.1275	4.1275	4.3815	4.5085
4.5085	4.6990	4.4450	5.2705	4.8895	4.7625	4.9530	4.8895	4.9530
4.8895	5.2705	5.1435	5.5880	5.7150	5.6515	5.8420	6.1595	6.2230
6.2865	6.3500	6.3500	6.4135	6.0325	6.0960	6.0960	6.1595	6.0325
5.9055	5.9055	5.9055	6.5405	6.4135	6.5405	6.3500	6.4770	6.5405
6.3500	6.4135	6.2865	6.4770	6.6675	6.7310	6.7310	6.6040	6.6040
6.9850	6.7310	6.4770	6.4770	5.6515	6.4135	6.0960	6.3500	6.0325
6.5405	6.2865	6.3500	6.4135	6.3500	6.2865	6.2230	6.1595	4.8895
5.8420	5.6515	5.7150	5.2070	5.2705	5.0800	5.4610	5.2705	5.2070
4.6990	4.8260	4.8895	4.7625	4.5085	4.8895	4.4450	4.6355	4.5720
4.1275	3.3655	4.3815	3.1115	3.6830	4.1910	3.6195	3.2385	2.0955
2.4765	3.4925	2.7305	7.3025	1.7780	2.6670	2.6035	2.3495	4.3815
2.6035	2.6035	2.0320	1.8415	1.0795	0.6985	0.2540	-0.0635	0.4445
0.1270	-0.2540	-0.2540	-0.5715	-0.5080	4.3180	2.9210	0.0000	12.1285
0.0635	1.9685	0.8255	1.5240	1.7780	1.4605	1.9685	2.0320	1.3970
2.3495	2.2860	2.9210	3.0480	3.1750	3.5560	3.2385	3.4290	3.8100
2.5400	3.8735	4.0005	4.1910	4.0640	4.2545	3.8100	4.1910	4.8260
4.6355	5.0800	4.7625	4.9530	5.3975	5.3975	5.7150	5.3975	5.0800
4.5720	4.8260	6.2865	5.0165	5.3975	5.5245	5.0800	6.0325	5.8420
5.9690	6.1595	7.2390	6.8580	4.6990	6.2865	6.5405	7.1120	4.8260
6.2865	6.7310	7.2390	6.1595	4.8260	7.3025	6.4135	6.3500	4.9530

Data Set 3

-33.4645	-32.9565	-31.4325	-28.4480	-25.7810	-24.0665	-22.5425	-20.5740
-22.2885	-21.9075	-22.0345	-21.4630	-22.2250	-23.4315	-25.9715	-26.0985
-26.7970	-27.3050	-25.4635	-23.6220	-20.8915	-16.7640	-12.8270	-8.6995
-5.7785	-2.6670	-0.5080	1.0795	1.0795	2.0320	2.0955	1.6510
2.6035	3.3655	0.6350	-1.9685	0.1270	-1.0160	0.1270	0.1905
1.5240	1.4605	5.4610	6.4770	9.2075	12.3190	17.4625	22.8600

20.5105	24.4475	27.7495	31.5595	32.3215	35.6235	31.3055	23.9395
17.1450	12.6365	8.3185	1.9685	-3.2385	-11.3030	-20.1295	-24.1300
-26.9240	-28.7655	-28.1940	-27.2415	-24.6380	-22.3520	-21.3995	-17.5895
-16.1925	-15.1765	-15.0495	-14.1605	-15.8750	-17.3355	-17.7165	-18.0975
-17.4625	-17.2085	-18.3515	-16.5100	-15.6845	-13.9065	-12.1920	-10.8585
-10.0330	-8.3820	-7.6200	-5.0165	-1.8415	1.0795	3.4290	5.3340
6.5405	7.8105	7.2390	8.0010	9.5250	8.6360	9.0805	11.3665
4.8895	3.9370	-0.6350	-4.1910	-7.9375	-12.4460	-13.3350	-16.1925
-19.0500	-22.4155	-23.3680	-23.0505	-24.1300	-23.4950	-21.4630	-20.5105
-18.3515	-17.5260	-15.8115	-15.8750	-14.6050	-13.0175	-12.3190	-11.1125
-10.0965	-8.3185	-7.4295	-9.4615	-9.5250	-9.2710	-8.3820	-6.5405
-7.5565	-8.3820	-7.3025	-6.7945	-7.0485	-6.0960	-4.2545	-5.0165
-3.4290	-3.0480	-2.7940	-2.5400	-3.4925	-3.4925	-3.5560	-5.9055
-6.6675	-6.5405	-7.7470	-8.0645	-9.3345	-10.7950	-12.3825	-14.4145
-13.6525	-13.0810	-14.6050	-14.6050	-12.7000	-12.7000	-14.2240	-14.9225
-16.7640	-15.2400	-14.8590	-15.2400	-16.5735	-15.3670	-15.4940	-16.3195
-14.0335	-13.8430	-14.4145	-15.1130	-15.1765	-13.7795	-14.7320	-14.4145
-14.6685	-13.3350	-12.5095	-12.8905	-9.6520	-10.9220	-9.3980	-6.2230
-6.2865	-3.9370	-4.0640	-2.4765	14.2875	1.6510	3.5560	4.6990
6.7945	7.3660	8.5090	6.5405	9.5885	10.4140	11.7475	10.8585
10.9855	11.1125	13.2715	14.0970	15.8750	18.7325	20.3835	20.8915
24.0665	28.3845	29.8450	29.6545	31.4325	33.7185	34.2900	34.8615
34.7980	32.0675	28.6385	24.3205	20.3200	16.8275	16.8275	17.7800
17.2085	16.0020	15.0495	14.7320	16.0655	15.4940	16.4465	17.2085
14.8590	12.8270	11.3665	10.2235	8.8900	8.3820	7.6835	6.6040
5.4610	6.4135	5.9055	6.0325	5.8420	6.5405	5.7785	8.2550
8.2550	7.4295	6.2865	4.8895	3.8100	4.7625	4.4450	4.8895
5.6515	5.7785	6.8580	7.0485	8.2550	9.3980	10.6045	12.1285
11.7475	7.8105	0.7620	-9.3980	-12.8270	-14.9860	-15.8750	-16.2560
-15.6210	-15.0495	-15.4940	-14.0335	-13.5890	-11.3665	-8.9535	-9.3980
-6.6040	-8.3185	-10.6680	-12.2555	-20.3200	-21.5265	-22.9235	-21.5900
-20.4470	-17.9705	-14.0970	-9.2075	-4.1910	-0.7620	0.5080	-2.1590
-3.2385	-0.3175	5.6515	11.4935	18.7960	16.3830	4.9530	-9.4615
-27.5590	-35.3060	-39.3065	-40.9575	-39.6875	-36.2585	-32.1945	-28.9560
-24.3205	-20.3835	-17.7165	-14.9225	-12.4460	-10.0330	-6.9850	-6.6040
-7.4930	-8.6995	-9.3345	-11.8745	-11.6840	-13.2715	-15.0495	-15.9385
-15.8115	-17.0815	-18.2245	-18.7325	-17.7800	-18.0340	-16.9545	-16.1925
-15.6210	-14.2240	-15.6845	-14.2240	-15.7480	-17.3355	-18.0975	-18.3515

-19.5580	-20.3200	-21.7170	-21.0820	-21.5900	-23.1140	-21.9710	-22.0980
-23.5585	-23.1140	-22.1615	-22.6695	-20.4470	-21.3995	-20.9550	-21.2090
-19.1135	-17.5260	-17.5895	-15.7480	-15.1130	-14.5415	-13.4620	-12.8905
-11.6840	-11.0490	-11.3665	-10.9220	-9.5885	-10.4140	-9.5885	-9.7155
-11.1125	-10.4140	-9.2710	-8.3820	-6.2230	-7.3025	-7.4295	-6.0325
-3.8735	-2.6035	1.2700	2.2860	3.3655	4.0005	5.5245	5.7150
6.7310	7.3660	6.2230	5.6515	5.5880	4.4450	2.5400	0.6985
-0.7620	-1.9685	-4.8260	-5.0165	-4.6990	-4.6355	-3.6830	-0.8890
1.3970	3.1750	9.1440	10.7315	14.1605	16.8275	18.6055	21.2725
19.3675	18.4785	17.7165	17.2720	14.2875	11.1125	4.9530	-3.8735
-11.9380	-17.3990	-21.5265	-22.9235	-23.7490	-24.3205	-26.7970	-26.4795
-22.6695	-16.5100	-13.3985	-10.4140	-10.6045	-7.4930	-6.1595	-5.7150
-4.1910	-5.0165	-4.8260	-8.3820	-11.4935	-14.3510	-14.9860	-17.7165
-19.2405	-21.2725	-20.7010	-19.1135	-18.7960	-14.6050	-10.9855	-7.8105
-4.8260	0.1270	4.6355	9.0170	12.5730	17.6530	21.9710	24.8920
25.4635	25.3365	23.6855	20.0660	16.5100	12.2555	10.1600	7.4295
5.1435	4.5085	4.1275	3.5560	3.8735	4.0640	4.2545	5.3975
5.8420	6.5405	6.0325	7.4930	8.4455	6.1595	7.0485	6.5405
6.1595	4.9530	3.7465	2.2860	1.2700	-1.7145	-4.7625	-11.9380
-18.0975	-23.9395	-29.5275	-33.0200	-34.7345	-36.1315	-36.1315	-35.6870
-32.7025							

Data Set 4

-2.2225	-2.4765	-3.1115	-4.1910	-4.0005	-4.2545	-4.3180	-3.2385
-2.4130	15.7480	0.1905	-4.7625	1.9050	-5.1435	-4.3815	-5.3340
-4.8260	-4.1910	-3.8735	-4.6990	-5.3975	-4.8895	-2.0955	-5.2070
-4.8895	-5.0165	-4.3180	-2.4765	1.0160	0.2540	6.2865	-2.9845
1.9050	-3.6830	0.6985	-3.8100	-3.8735	1.9050	23.8760	4.1910
-2.9845	-5.3340	-5.5245	-5.9055	-6.2865	-6.7310	-7.0485	-7.4295
-7.7470	-8.2550	-8.3185	-8.5725	-8.9535	-9.0805	2.4765	4.0005
8.8265	-0.2540	-0.1905	1.0795	2.7940	0.1905	-4.5085	10.2870
-5.9055	-7.0485	-9.5250	-7.6835	-8.6360	-9.1440	-9.2710	-3.8100
7.6200	-8.8265	-8.5725	5.4610	4.4450	2.7940	-7.4930	-8.3820
-8.2550	-8.1280	-8.0645	-7.8105	-8.1915	-8.5725	-8.5725	-9.0805
-9.4615	-9.8425	-10.3505	-11.0490	-11.4935	-11.8745	-12.7000	-13.3350
-13.9700	-14.6050	-15.0495	-15.4940	-16.1290	-16.8910	-17.1450	0.6350
14.7955	-17.7800	-16.9545	-17.7800	-17.9705	-17.9705	-17.7800	-17.6530

-16.9545	-16.7005	-16.4465	9.3345	-12.7635	3.8100	-3.4290	-14.6685
8.4455	-13.7795	-13.6525	-13.4620	-9.8425	-0.8890	-2.6670	-6.9850
-12.0650	-11.6840	-11.2395	-11.4300	-10.9855	-10.6680	-4.9530	-10.5410
-10.5410	-10.4775	-10.6045	-10.7950	-10.7950	-11.2395	-11.3030	-11.6840
-11.3665	-11.8110	-12.1920	-12.5095	-12.7000	-12.9540	-13.1445	-13.5890
-13.8430	-14.2875	-14.4780	-12.1285	-14.5415	-14.7955	-14.9225	-14.8590
-14.9225	-12.1920	-15.2400	-15.0495	-15.1130	-15.1130	-15.2400	-14.8590
-14.8590	-4.7625	-13.9065	-14.5415	-14.2875	-14.0335	-13.9065	-13.5255
-13.2080	-13.2715	-13.1445	-12.9540	-12.7000	-12.5730	-12.0015	-11.9380
-11.6205	-11.6840	-11.4935	-11.1760	-11.4300	-11.0490	2.0320	11.8745
7.4295	15.0495	13.4620	-1.2065	10.8585	3.4925	-9.5885	8.1915
-9.3980	3.9370	-2.9845	-4.0005	5.5880	9.2075	3.6195	-6.7310
10.7315	-7.7470	-7.0485	7.4295	7.8740	13.7795	10.8585	-5.0165
8.5725	4.2545	-5.5245	-5.5245	-4.3180	-4.1275	-5.0165	-5.1435
2.4130	0.3175	-5.0800	-4.7625	-5.8420	-5.1435	-5.5245	-1.3335
-6.0325	-6.0960	-6.4770	-6.8580	-6.9850	-2.5400	-7.3025	17.8435
0.0635	-6.6040	-5.3340	-4.3815	-9.8425	-10.0965	-10.1600	-5.7785
-8.0010	-4.5720	-11.2395	-5.6515	-11.3030	-11.7475	8.1280	7.4295
-2.5400	-10.9220	-10.6680	-9.9695	-8.9535	-8.8265	-7.8740	-6.9215
-2.6670	-1.1430	-4.5085	-3.7465	-2.9845	-2.2225	-1.2065	-0.3175
5.7150	1.8415	6.6675	2.0955	2.0955	2.8575	2.9210	3.3020
3.4290	10.1600	4.8895	3.6195	7.6200	21.4630	11.1125	20.3835
2.9845	2.4765	16.2560	12.0015	10.3505	0.6350	2.7305	0.3810
-1.0795	-1.7780	-2.2860	-2.8575	-4.0005	-4.5085	-4.8895	-5.5245
4.8895	-6.0325	-6.4135	0.9525	4.0005	-5.7150	-1.4605	-5.4610
-5.3975	11.2395	7.6835	6.5405	-3.8100	1.5875	-1.0160	-2.4765
5.3340	-1.1430	6.0325	3.3655	0.2540	4.0640	1.2700	2.2225
16.8275	35.1155	20.0660	6.7310	3.2385	15.6210	16.4465	4.6355
3.1115	5.1435	4.2545	6.0960	4.1275	1.5875	16.3195	4.4450
21.7170	2.7305	7.1120	2.4130	-2.4765	-0.9525	6.4770	-3.6195
-4.5085	15.0495	-0.3175	-6.0325	-6.3500	-6.6675	-5.0165	8.1915
-1.3970	-2.7940	3.1115	-1.2065	-4.2545	-5.1435	8.7630	-1.4605
-3.5560	-1.7780	-2.4765	-1.7145	-1.2065	-0.6985	0.3175	0.8890
1.4605	3.4290	2.0320	6.0960	6.4135	4.3180	7.0485	7.2390
7.4295	3.8100	7.0485	7.4295	4.0005	4.3180	4.0005	4.0005
8.9535	2.5400	2.4130	1.6510	1.4605	11.8745	7.9375	7.4295
-0.2540	3.4290	2.0320	7.3025	0.7620	0.6350	0.9525	-4.8260
-6.2230	-6.5405	-6.6040	7.8105	-4.7625	4.6355	2.4130	4.7625

3.3020	-6.9215	3.7465	1.3335	4.9530	-0.8890	-5.5880	-5.4610
-1.5875	-4.1275	-5.3975	-5.7785	-3.6830	-2.5400	4.6355	-0.9525
-1.9050	-2.4130	-2.0320	0.8890	-0.7620	1.0160	-0.5080	1.2700
-1.0795	1.9050	2.9845	0.0635	-0.3810	2.2860	0.8255	3.3655
-0.6350	2.6670	1.5240	1.1430	-0.7620	-0.8255	2.4765	0.8255
-1.9050	-1.9050	4.1275	8.8265	-3.4290	1.6510	-3.7465	-5.3975
-5.1435	-5.3975	-6.0325	-5.8420	-5.7785	0.3175	-5.3340	-4.0005
-2.4130	-4.0640	2.2225	-2.6670	-3.4925	-0.5080	-0.6350	-2.5400
-0.7620	-0.3810	2.1590	-0.8890	-2.8575	0.0000	-1.3335	-0.1270
-2.4130	-2.9845	0.7620	6.4135	-1.3970	3.2385	14.2240	-0.3175
4.8260	7.8740	-1.1430	-1.3335	-1.6510	0.2540	4.2545	-0.0635
-1.3970	-1.5240	-1.9050	-2.5400	-3.3655	-2.7305	-3.5560	-2.0955
-2.7940							

Data Set 5

-20.0025	-27.2415	-22.6060	-20.8915	-21.4630	-19.7485	-15.1765	-23.9395
-21.0185	-17.1450	-16.3830	-20.1295	-20.7010	-13.5890	-14.9225	-10.7315
-9.1440	-9.3345	-16.5735	-14.0970	-17.9705	-11.0490	-10.2235	-12.7635
-11.0490	-6.0325	-4.7625	-10.0330	-5.0800	-3.4925	-7.3025	-3.1115
-1.9685	-5.5880	-4.9530	-1.0795	-3.4925	-4.8260	-1.9685	1.9685
-9.3980	-3.6195	-0.9525	0.3175	5.0800	-0.4445	0.4445	5.7785
-1.9685	3.6195	5.2705	6.5405	-0.0635	4.5085	5.7785	5.2705
8.4455	8.0645	5.9690	2.0955	2.7305	7.4930	8.7630	9.5250
4.6990	6.6675	11.5570	4.3815	5.2705	12.6365	6.3500	7.2390
5.9055	7.8105	5.5880	-2.2225	2.8575	7.2390	7.3660	6.2865
10.7950	4.0005	-0.7620	3.8100	3.0480	7.1755	4.8895	-1.7145
0.1905	1.3335	5.5880	8.0645	1.4605	3.8100	5.5245	-1.3970
1.9685	6.2865	7.4930	-2.6670	-3.8100	-4.6355	-5.3340	4.5085
-7.9375	-2.2860	-1.6510	0.4445	-1.1430	-2.0955	4.8895	2.8575
4.3815	5.3975	-2.1590	0.5080	-3.7465	-5.9055	-10.9220	-4.0640
-5.3340	-7.3660	-3.1750	-1.6510	2.7305	-1.3970	2.5400	-4.4450
2.6670	-1.4605	-7.0485	1.2065	1.5875	-1.2065	-6.4135	-9.9695
-2.3495	-2.8575	-7.9375	-2.8575	-4.0640	-3.8735	-1.2065	-14.5415
-9.3980	-7.4930	-4.6990	-1.3335	-5.2070	-6.0325	-7.4930	-2.5400
-8.0645	-3.7465	0.1270	-8.3185	-6.1595	-5.3340	-1.2700	-8.1280
-2.0955	0.1905	-6.4770	3.0480	2.6670	0.4445	3.0480	-5.4610
1.2700	2.4765	0.1270	-2.1590	-6.2865	-2.8575	1.1430	-7.8740

0.2540	1.0160	1.9685	0.1905	0.5080	-1.3970	-3.2385	-0.1270
-1.1430	5.2705	-2.6035	5.0800	-2.4130	0.5715	2.8575	5.2070
8.8265	7.2390	3.4925	4.2545	11.2395	10.9220	3.0480	6.3500
10.9220	11.6205	10.9855	12.1920	9.1440	12.4460	1.2700	5.9690
13.7795	15.6845	12.0650	11.6840	19.4945	21.8440	18.7960	22.4790
17.5260	18.6690	20.3200	16.6370	19.5580	15.2400	17.7165	17.2085
13.9065	11.1760	13.3350	18.7325	13.9065	14.9225	19.5580	21.4630
27.0510	30.5435	27.3685	29.5910	28.7020	27.1145	26.6065	23.6220
26.5430	25.2730	20.9550	25.5905	22.4790	19.6215	24.5110	25.4000
28.0670	23.9395	23.9395	24.7650	29.4640	22.0345	24.7650	33.5280
28.3845	25.2095	19.0500	17.4625	15.8115	20.3200	27.6860	26.3525
24.9555	22.9235	28.3845	23.9395	23.3045	26.5430	16.6370	22.6060
16.9545	22.2250	20.8280	24.9555	22.9870	24.1300	18.6690	14.4780
18.7960	22.9870	22.7330	24.3840	24.3840	22.5425	14.0970	18.6055
14.7955	21.5900	16.5100	17.0815	13.5890	6.1595	16.8910	19.9390
10.9220	13.5255	7.8105	16.8910	13.4620	6.7310	14.4145	13.4620
13.6525	6.4770	12.1920	15.2400	9.0170	8.2550	13.5890	4.0005
6.8580	6.9850	3.6830	5.8420	10.5410	7.3025	6.0325	2.4130
3.3020	8.1915	7.8740	-1.7145	1.9685	0.6350	0.5715	0.6350
1.5875	-3.8735	1.9050	3.0480	-2.3495	5.0165	-4.2545	0.2540
-9.6520	5.1435	-5.0800	-5.0165	0.7620	-0.6985	-0.6985	3.6195
-0.2540	1.2065	2.5400	-3.9370	-1.2700	-0.0635	4.1275	-7.9375
-0.5080	5.5245	-4.6355	-2.9210	-0.3810	-5.2070	-2.0320	-2.4130
-2.5400	-8.3820	-2.4130	-1.1430	1.7780	0.1905	-4.8260	-2.0320
-2.7305	0.6350	4.4450	-1.1430	3.4290	-5.4610	-3.5560	-1.2700
0.4445	-1.9050	0.5715	0.7620	5.1435	0.9525	1.7780	7.9375
3.4290	6.8580	2.8575	7.9375	-1.8415	3.6830	4.3180	2.7940
1.6510	0.3175	1.1430	2.9845	-1.7145	1.3335	-1.0795	3.6830
-0.8255	0.3175	-2.7940	1.8415	4.0640	21.6535	48.9585	34.9885
6.6675	-4.0005	5.3340	5.0165	-1.0795	3.4925	2.9210	-1.2700
-1.2065	-3.6195	-2.4765	1.2065	-1.7780	-4.8895	-0.6350	0.3810
-4.9530	-6.7945	-2.7305	-4.3180	-5.3975	-5.3975	-7.7470	-7.6835
-6.2865	-4.3815	-8.2550	-2.0955	-4.2545	-9.4615	-6.1595	-5.2070
-8.5090	-4.2545	-6.4135	-9.2710	-8.3820	-6.5405	-9.7790	-15.4940
-9.9060	-14.2240	-11.4300	-7.1120	-9.0170	-13.8430	-13.4620	-12.6365
-11.4935	-17.5260	-13.7795	-15.5575	-13.5255	-20.2565	-14.8590	-12.9540
-20.3835	-15.3035	-13.6525	-20.4470	-15.0495	-12.8905	-18.9865	-12.8270
-17.2720	-17.2085	-18.6690	-14.9225	-23.5585	-21.2725	-19.8120	-20.1295

-20.2565	-26.4795	-23.5585	-22.1615	-21.9075	-17.2720	-18.7960	-19.3675
-23.3680	-22.1615	-21.9075	-20.3200	-19.8120	-28.3845	-27.8130	-23.6855
-29.1465	-22.3520	-21.9710	-24.1300	-21.3360	-22.0345	-17.8435	-18.4150
-20.5740	-16.8275	-23.6220	-19.1770	-21.3360	-21.5265	-25.4635	-23.7490
-20.5105							

Data Set 6

-24.5110	-28.0670	-27.3685	-29.9085	-27.3685	-26.6700	-21.8440	-16.4465
-20.9550	-20.1295	-24.7650	-22.0980	-25.5270	-29.9720	-21.6535	-26.0985
-24.3840	-25.4000	-28.5115	-19.6850	-19.8755	-21.9075	-24.0665	-18.4785
-17.9070	-15.8115	-20.6375	-21.4630	-24.1935	-22.2250	-19.0500	-15.8115
-16.5100	-12.2555	-9.4615	-5.3340	-12.7635	-13.9700	-11.4935	-7.6200
-12.1920	-9.8425	-3.2385	-7.0485	-1.9050	0.3175	-4.4450	-5.3340
-5.7785	-4.6355	1.7780	0.1905	2.6670	4.5085	11.3030	8.6995
1.0795	-1.3335	-7.9375	-6.9215	0.1905	0.1905	-5.9690	-10.6045
-0.8255	2.5400	-1.6510	1.7780	-1.6510	-6.4770	-3.1750	-5.3975
-2.2225	-6.4135	-10.5410	-11.0490	-7.4295	-10.6680	-7.6200	-5.2070
-8.0010	-3.2385	1.4605	-3.0480	-2.0320	-2.9845	-7.8105	-1.3335
-3.1115	-3.7465	-3.8735	0.1270	5.7785	0.1270	-0.7620	-3.3655
-17.9070	-4.8895	-3.6195	6.0960	7.3025	5.6515	8.1280	8.2550
9.4615	13.3985	11.4300	8.1915	10.0965	10.3505	10.2870	4.3815
1.1430	-8.0645	-2.0955	-6.7310	-3.8100	-1.0795	1.2065	2.0320
2.4130	1.7780	-10.4775	-1.6510	-6.4770	-19.2405	-2.7305	-5.7785
0.8255	1.9685	1.6510	-0.1905	-1.3335	-10.5410	-8.4455	-16.1925
-21.0185	-15.4940	-9.4615	-12.7635	-22.2250	-21.8440	-17.2720	-22.6060
-22.4790	-11.1760	-15.0495	-18.0340	-15.8115	-15.8115	-25.7810	-22.2250
-24.5745	-27.4320	-37.4015	-32.7660	-27.3685	-27.9400	-27.6225	-19.8755
-20.3200	-22.9235	-30.6070	-26.8605	-31.3055	-36.4490	-34.0360	-32.3215
-29.0830	-32.8930	-32.8930	-28.6385	-29.9085	-31.1150	-32.2580	-34.1630
-37.9730	-33.4645	-27.1145	-30.3530	-27.2415	-30.1625	-30.2260	-32.1945
-25.2095	-22.1615	-33.3375	-28.5115	-21.7805	-24.0030	-25.5905	-27.6860
-21.4630	-21.6535	-16.8275	-24.8285	-20.2565	-18.7325	-18.7325	-15.5575
-22.2250	-6.1595	-2.1590	-2.0955	-4.1910	-15.4940	-13.9700	-4.7625
-10.2235	-9.7790	-11.7475	-14.7320	-4.6355	-7.2390	-6.7310	-5.7785
-2.1590	2.9845	-2.2860	-6.4770	3.0480	-4.6990	3.2385	5.2070
0.5715	-3.4290	0.8890	2.1590	5.5245	6.4135	9.3345	3.9370
8.5725	5.0165	9.9695	10.7315	9.5250	11.1760	6.1595	-2.9210

4.7625	4.2545	9.2075	10.5410	7.2390	10.5410	10.7950	11.4935
7.3660	13.2715	10.0330	7.8105	14.4145	12.3190	6.1595	9.5250
15.6210	9.7155	6.4135	16.1290	18.6055	11.3030	11.4935	14.5415
7.3025	16.2560	12.7000	12.1920	19.3040	17.7165	24.3840	17.8435
24.2570	24.1935	25.1460	18.1610	12.5095	16.0020	9.2710	10.9855
12.0650	14.4145	1.8415	7.2390	1.5875	14.0335	10.8585	11.9380
8.3185	9.1440	13.0175	11.1125	13.3350	10.3505	10.4775	10.1600
5.3340	-1.2700	7.6200	6.7945	13.0810	5.8420	8.5725	10.0965
-0.4445	-4.3815	-2.4765	4.6990	3.8735	1.0160	-7.8105	-4.4450
-13.0810	-28.5750	-14.7955	-14.2875	-8.3820	-10.0330	-18.4785	-12.5095
-15.3035	-8.8900	-20.4470	-29.1465	-28.3845	-23.0505	-28.1305	-17.9070
-17.5895	-25.2095	-23.3680	-21.7170	-21.1455	-21.4630	-14.3510	-13.3985
-15.3670	-15.7480	-18.2245	-15.3670	-13.9065	-25.4000	-24.8920	-24.6380
-15.4305	-10.5410	-14.4145	-16.2560	-19.2405	-13.8430	-19.1770	-17.7800
-18.9230	-17.0180	-16.7640	-13.2080	-14.8590	-15.0495	-8.6995	-9.6520
-6.4770	-6.1595	-6.0960	-2.1590	-1.9050	-6.7945	-0.8255	4.6355
2.9845	0.4445	5.7785	11.4935	8.6995	-6.7310	-2.0955	5.1435
8.6360	4.8260	-1.2700	-0.6985	8.2550	7.3660	5.2705	7.8105
6.0325	7.5565	4.9530	0.6350	5.3340	13.7160	9.6520	8.7630
14.2875	10.9855	15.3670	11.3665	3.4290	2.6035	12.3190	11.8745
8.8265	5.5880	2.3495	5.2705	2.2225	-4.3180	1.7780	4.7625
2.5400	11.7475	15.3035	6.2230	11.6205	8.1280	8.3820	6.6675
8.9535	8.7630	6.4770	11.9380	9.2075	9.8425	13.4620	14.9225
15.8750	13.3985	4.8895	7.7470	11.6205	13.1445	11.1125	13.2715
5.3340	15.4305	19.6215	20.2565	18.8595	25.1460	26.0985	24.0665
25.2095	23.7490	23.8760	23.2410	19.4945	20.4470	24.0030	23.8125
19.7485	22.5425	25.7175	24.3205	23.6855	21.3995	20.5740	16.6370
18.3515	17.9070	20.7645	18.0340	19.2405	15.8750	22.1615	24.1935
15.8115	15.0495	11.5570	10.2870	7.0485	10.5410	13.5890	9.5250
4.7625	9.1440	9.7790	7.1755	4.0005	0.3810	-2.2860	-9.2710
-12.9540	-13.9700	-14.4780	-13.2715	-18.4785	-15.9385	-18.4150	-13.4620
-1.7145	-3.1750	-5.5245	-13.6525	-15.8115	-10.1600	-15.1765	-10.3505
-11.5570	-14.5415	-18.6055	-23.3045	-16.1925	-16.5735	-19.5580	-17.4625
-23.1140	-31.5595	-27.1780	-25.7175	-20.8915	-20.0660	-21.1455	-22.0345
-24.7650							

Data Set 7

-11.3665	-10.7950	-11.0490	-11.2395	-11.4935	-11.6840	-12.0015	-12.0015
-11.9380	-11.8110	-11.8745	-12.0015	-11.8745	-12.9540	-12.6365	-12.5095
-12.7000	-12.5095	-12.6365	-12.1920	-13.3985	-13.5255	-13.3350	-13.6525
-12.9540	-13.3350	-13.7160	-13.7795	-13.8430	-13.9065	-14.0335	-14.2240
-13.4620	-13.6525	-13.7160	-13.9700	-14.1605	-14.0335	-13.8430	-13.8430
-13.7795	-14.0970	-14.5415	-14.6685	-14.7955	-14.6050	-14.2240	-14.1605
-14.4145	-14.6050	-14.8590	-14.8590	-14.8590	-15.0495	-14.4145	-14.2875
-14.1605	-14.6685	-14.6685	-13.3985	-12.1285	-14.4145	-14.1605	-14.2240
-14.4780	-14.6050	-14.8590	-14.6685	-14.4780	-14.4780	-14.6050	-14.6685
-14.5415	-14.5415	-14.7320	-14.2240	-14.4145	-14.0970	-14.2240	-13.9065
-13.7795	-14.0335	-13.6525	-13.9065	-13.8430	-14.0335	-13.5890	-13.2715
-13.0175	-13.2080	-13.7160	-13.6525	-13.5890	-13.3350	-12.5730	-12.5730
-12.7635	-13.2715	-13.1445	-12.5730	-12.7000	-11.8745	-11.3030	-11.8745
-11.9380	-12.3825	-12.1285	-11.4935	-11.7475	-11.3665	-10.9855	-11.5570
-11.6205	-11.6840	-11.6205	-11.4935	-10.5410	-9.8425	-10.4775	-10.2870
-10.5410	-10.3505	-10.6045	-9.7155	-9.2075	-9.0805	-9.2710	-9.4615
-9.2075	-10.0330	-9.5250	-8.2550	-7.9375	-7.8105	-8.3820	-8.1280
-8.8265	-8.7630	-7.5565	-7.1755	-6.6675	-6.9215	-6.8580	-6.8580
-7.1120	-6.0960	-5.9055	-5.3340	-5.3975	-5.4610	-5.3975	-5.5880
-4.8895	-4.8260	-4.6990	-4.3815	-4.4450	-4.2545	-4.1910	-3.9370
-3.7465	-3.6830	-3.3655	-2.7305	-2.2860	-2.7305	-2.6670	-2.8575
-2.8575	-3.0480	-1.9050	-1.2065	-1.5875	-1.7145	-1.7780	-1.1430
-1.5875	0.6985	0.0000	0.0000	-0.3810	-0.2540	-0.1270	-0.1905
-0.3810	0.6350	1.2065	1.1430	1.0160	1.2700	1.6510	1.5240
1.9685	2.1590	2.6035	2.4765	1.9685	2.6035	2.7940	2.7305
3.3020	3.8735	3.9370	3.3655	3.6830	4.2545	4.5085	4.7625
5.0800	5.3975	5.2705	5.1435	5.3340	5.5245	5.3340	5.9055
6.3500	6.6040	6.2230	6.7945	6.6040	6.8580	6.9215	7.1120
7.3660	7.4930	7.4930	7.8105	7.8740	8.0010	8.1280	8.4455
8.6360	8.6360	8.8265	9.2710	9.3345	9.3345	9.3980	9.6520
9.6520	11.3030	9.9060	10.0965	10.0330	10.0330	10.4775	11.1125
11.0490	11.0490	11.1760	11.1760	10.7950	10.7950	10.8585	11.6205
11.7475	11.5570	11.5570	11.1760	11.4935	11.8745	12.2555	12.6365
12.2555	12.3825	12.4460	12.3825	12.7000	13.0175	13.3350	12.8905
12.9540	13.0810	12.4460	12.9540	13.9065	13.9065	14.0335	13.3985

14.0970	13.1445	13.2080	14.0335	14.0335	13.8430	13.9065	14.1605
14.2240	13.9065	14.1605	14.4145	14.6685	14.4145	14.1605	14.7955
14.4145	14.5415	14.7320	14.6685	14.5415	14.3510	14.9225	14.3510
14.4780	15.1130	14.9225	15.1765	14.8590	14.8590	15.0495	14.7320
15.2400	14.9225	15.1130	15.1130	15.2400	14.5415	14.7955	14.6685
15.8750	14.7955	15.0495	15.0495	14.6685	14.2875	14.4780	14.6050
14.6050	14.7955	14.9225	14.0335	13.3350	13.9700	14.0970	14.4145
14.9225	14.9860	14.4145	13.2080	12.9540	13.7795	14.0335	14.1605
13.9700	13.5255	13.9065	12.3190	12.8905	13.8430	13.6525	13.5890
13.3985	12.9540	13.1445	13.0810	13.2080	13.2715	13.2715	13.0175
12.5730	12.2555	12.3190	12.3825	12.5095	12.4460	12.1285	12.0650
11.6205	11.4300	11.5570	11.5570	11.6205	11.4935	11.1760	10.9220
10.8585	10.9220	11.9380	11.0490	10.8585	10.2870	10.2870	10.0330
9.7790	9.5250	9.4615	9.3345	9.3980	8.9535	8.3185	8.6995
8.6360	8.4455	8.2550	8.8900	8.7630	7.6200	7.8105	8.0010
7.8740	7.6835	7.5565	7.4930	7.1120	6.1595	6.2230	6.3500
6.5405	6.0325	6.3500	5.7150	4.4450	4.1910	5.0165	5.0800
5.0165	4.9530	4.3815	3.4290	3.3655	3.8100	3.8735	3.9370
3.7465	3.7465	2.9845	1.6510	1.7780	2.5400	2.4130	2.3495
2.2860	1.8415	0.7620	0.3810	1.3335	0.9525	1.3335	0.8890
0.8890	0.3175	-0.1905	-0.1905	-0.2540	-0.1905	-0.3175	-0.3810
-0.6350	-1.2700	-1.4605	-1.5875	-1.7145	-1.6510	-1.7780	-2.1590
-2.2860	-2.4765	-2.7940	-2.7940	-2.9845	-2.8575	-3.1115	-3.8100
-3.5560	-3.9370	-3.9370	-4.1275	-4.1275	-3.7465	-4.8260	-4.8895
-5.0800	-5.4610	-5.7150	-5.5245	-5.5880	-5.7785	-6.3500	-6.6040
-6.6040	-6.6675	-6.8580	-6.5405	-6.7310	-6.8580	-7.7470	-7.5565
-7.4930	-7.7470	-7.9375	-7.8740	-8.0645	-8.5725	-8.6360	-8.5090
-8.8900	-9.3980	-8.1280	-9.4615	-9.5885	-9.8425	-9.5885	-9.9695
-10.3505	-10.7315	-10.4140	-10.3505	-10.8585	-10.5410	-10.9220	-11.0490
-11.4935							

Data Set 8

-6.8580	-7.8105	-9.9695	-6.9850	-8.2550	-10.7315	-3.4290	-1.7780
-2.4130	-3.4290	-4.2545	-4.0640	-4.8260	-1.3335	3.3655	1.7145
7.1755	10.0330	7.3660	4.2545	4.3180	6.7310	6.4135	12.0650
8.2550	9.2710	9.2710	5.6515	4.1275	3.1115	4.3815	4.0005
1.9050	7.6200	6.4770	1.5875	-2.4765	-6.8580	-6.1595	-3.4925

-6.6675	-7.5565	-6.2230	-5.2070	-4.7625	-5.4610	-8.7630	-7.3025
-5.7785	-8.0010	0.0000	-1.4605	-0.0635	2.3495	-12.3190	-10.5410
-11.4935	-13.6525	-14.1605	-13.1445	-13.8430	-19.1770	-17.5260	-23.4950
-26.2890	-25.0825	-22.2885	-19.9390	-16.7005	-17.5260	-22.2885	-24.2570
-24.1300	-23.1775	-19.8120	-21.9075	-20.4470	-21.9710	-24.1935	-25.7810
-27.8130	-31.6230	-25.0825	-21.9710	-24.4475	-29.9085	-22.4155	-24.0030
-28.5115	-28.3845	-28.5115	-28.7020	-29.7815	-26.9875	-16.0655	-11.5570
-10.4775	-14.0970	-11.2395	-8.4455	-9.1440	-8.5090	-4.6990	-3.6830
-5.0800	-4.3815	-1.4605	-3.1115	-9.3980	-12.3190	-6.7310	-1.4605
3.6830	4.3815	2.9210	3.8100	8.9535	4.3815	7.2390	12.3190
13.9700	14.2240	10.2870	6.6675	7.7470	9.8425	6.7945	10.7950
8.3820	9.1440	9.3345	7.6835	7.7470	7.4930	8.1280	12.6365
15.1765	12.2555	12.1285	7.9375	7.8105	10.9855	10.5410	10.5410
14.9225	14.7955	10.8585	4.8260	3.1115	3.6830	1.9685	0.3810
1.9050	5.2705	-4.8260	1.2065	6.2230	1.5875	-10.2235	-19.7485
-12.3825	-12.1920	-6.8580	-4.4450	-12.7000	-15.0495	-14.8590	-11.8745
-12.8905	-16.8275	-13.6525	-15.3035	-12.6365	-16.2560	-17.3355	-18.9230
-20.3835	-21.5900	-18.4785	-21.2725	-16.7005	-15.6845	-16.8275	-20.7010
-17.5895	-17.9070	-16.7640	-14.3510	-14.3510	-15.8115	-17.6530	-23.3680
-20.6375	-17.9705	-15.6210	-16.8910	-17.7165	-18.6690	-13.2715	-12.7000
-10.7315	-9.3980	-7.6835	-4.5720	-4.0005	-4.1275	-1.5240	3.9370
-1.0160	-1.9050	-1.9050	3.4290	5.9690	8.1280	13.7160	11.9380
11.6205	9.5250	12.0650	11.3665	14.6685	17.4625	21.8440	20.8915
18.0340	17.1450	11.9380	11.8110	20.5740	26.2255	26.9875	27.2415
22.3520	15.6845	14.2875	18.2880	15.1130	12.8270	14.0335	18.5420
21.3995	18.0340	15.6845	17.3990	19.4945	21.9710	23.6220	17.1450
15.9385	8.1280	6.1595	6.9850	6.0960	7.3025	10.4775	9.0805
9.5885	7.6835	8.1280	3.4925	4.4450	4.1910	2.0955	4.2545
4.9530	1.3970	2.2860	-1.2065	-7.8740	-7.0485	-10.8585	-11.9380
-10.0965	-4.3180	-2.1590	-10.2870	-12.8270	-12.3825	-18.5420	-17.2720
-7.3025	-6.5405	-4.3815	-6.6675	-9.5250	-9.7155	-8.2550	-7.1755
-9.4615	-6.1595	-4.6990	-2.2225	-4.1910	-2.7940	-2.7940	-3.8735
-1.7780	0.1270	4.7625	6.9850	2.9845	-3.4925	0.3175	-1.3335
2.9210	3.1750	2.3495	2.8575	5.5245	2.9210	6.4770	5.0800
5.2705	8.3820	15.5575	16.5735	17.9705	16.5100	14.8590	12.8270
12.9540	11.8745	15.1765	19.8120	26.2890	25.7810	20.8915	17.1450
17.0815	20.7010	20.4470	23.8760	26.1620	23.2410	26.1620	25.3365
23.6220	26.5430	25.9080	23.2410	23.1140	20.2565	25.5905	21.8440

20.4470	21.8440	23.2410	24.2570	23.5585	19.9390	19.2405	18.2880
13.4620	11.8745	9.3345	6.0325	1.5240	10.6045	11.4300	12.6365
5.9055	2.4130	-1.3335	1.0160	3.6195	7.7470	5.7785	4.1910
0.0000	-0.3810	-3.3020	-2.6670	-3.3655	-1.8415	-1.4605	-2.7940
-2.7940	-4.6355	-10.2235	-17.4625	-18.5420	-14.7320	-12.1285	-10.0965
-10.7950	-12.3825	-14.2240	-13.6525	-12.1920	-9.1440	-7.8740	-6.2230
-10.0965	-8.9535	-8.3185	-9.9060	-11.1125	-13.8430	-10.0330	-4.0005
2.2225	1.5240	0.9525	1.4605	1.0795	1.7780	0.0635	3.9370
7.9375	11.0490	11.4935	8.8265	8.1915	9.0170	6.4135	5.3340
12.9540	12.8270	11.4300	12.6365	12.3190	11.8745	13.7160	17.5260
19.2405	21.7805	24.1935	23.0505	21.0185	19.1135	18.6055	16.3195
22.8600	23.4315	27.6225	26.4160	23.4950	23.5585	21.0820	18.6690
19.1770	25.4635	24.3840	23.6220	23.4950	22.6060	20.0660	14.9225
14.7955	21.9075	20.7010	19.3040	18.1610	15.6845	10.9220	10.6045
10.0330	6.3500	6.6040	13.5890	12.9540	9.5885	9.8425	6.5405
4.3815	1.5240	-3.2385	-5.2705	-9.3980	-8.2550	-8.2550	-9.7790
-11.4935	-12.1285	-13.3350	-11.0490	-7.3660	-7.4295	-8.6360	-9.2710
-10.6680	-11.6840	-17.3355	-20.0660	-16.4465	-12.0650	-14.3510	-17.2720
-15.4940	-19.4945	-17.9070	-14.9860	-14.1605	-11.3665	-16.2560	-12.3190
-15.3670	-19.0500	-23.9395	-19.3675	-19.2405	-20.1930	-16.8910	-13.8430
-10.1600	-10.0330	-15.4305	-16.5735	-13.5890	-15.6210	-8.1915	-8.8265
-7.5565							

Data Set 9

-3.4290	-3.9370	-2.2860	-1.0795	-1.0795	-2.4765	-0.8255	-1.4605	-1.7780
-1.9685	-3.1115	-2.9845	-2.0320	-2.2225	-3.1750	-2.2225	-3.0480	-2.0955
-2.1590	-2.9210	-2.4130	-1.3970	-2.5400	-1.8415	-11.8745	-6.1595	-2.6670
-2.4765	3.1750	0.8890	-2.4765	-3.1115	-2.7305	-2.4130	-3.1115	-2.2860
-3.4925	-3.9370	-4.6355	-2.7940	-2.0320	-2.9845	-3.3655	-2.5400	-3.2385
-3.5560	-2.9845	-2.7940	-3.2385	-4.0005	-2.3495	-3.3020	-3.8735	-3.5560
-3.7465	-3.5560	-3.7465	-2.9845	-3.3020	-2.7305	-3.3020	-3.4925	-5.3975
-3.7465	-5.8420	-3.6195	-2.9845	-3.1115	-5.4610	-8.3820	-7.8105	-5.2705
-4.9530	-4.0005	-3.8100	-4.0640	-3.7465	-4.1275	-4.1910	-3.1750	-3.2385
-3.6830	-3.7465	-3.4290	-3.3655	-4.0640	-6.4135	-4.3815	-3.0480	-3.2385
-3.3655	-4.1275	-3.8100	-4.8895	-3.2385	-3.9370	-3.6195	-3.8100	-4.8260
-3.3655	-3.4290	-4.1275	-4.0640	-3.6830	-3.7465	-4.3815	-3.6830	-4.5085
-3.3020	-3.1115	-3.1115	-3.4925	-2.8575	-3.1750	-2.7940	-2.9845	-3.4925

-4.1275	-4.1275	-4.7625	-4.6990	-5.0800	-3.4925	-4.2545	-4.4450
-5.0165	-4.5720	-6.2865	-4.4450	-4.8260	-6.4770	-4.3815	-3.7465
-3.8735	-3.9370	-3.6830	-3.6195	-3.8100	-3.6195	-3.8735	-4.3815
-4.2545	-4.3815	-4.0005	-3.3020	-2.9845	-3.0480	-4.3815	-3.6830
-3.0480	-3.3655	-4.7625	-3.6830	-3.4925	-3.4290	-4.7625	-4.5720
-3.4925	-3.7465	-4.5085	-4.1275	-3.1115	-2.9845	-3.6830	-3.2385
-3.2385	-2.4765	-3.3020	-2.9845	-2.9845	-2.4765	-2.6035	-2.7305
-2.5400	-3.0480	-2.8575	-2.8575	-2.9210	-3.2385	-2.7305	-2.8575
-3.1115	-5.1435	-6.6040	-5.0165	-4.3815	-3.9370	-3.6830	-3.2385
-2.3495	0.6985	-3.6830	-1.9685	-2.5400	-2.0320	-2.0320	-1.7145
-2.3495	-1.3970	-1.6510	-2.3495	-1.4605	-1.3970	-0.8890	-0.6350
-0.8890	5.2070	-2.2225	-1.3335	1.7780	-0.1905	-0.5080	-1.3335
-0.7620	-0.8255	0.6985	7.8105	-0.2540	0.9525	0.3175	1.3970
-0.4445	-0.1270	-0.3175	0.0000	0.1905	0.5080	-0.0635	0.0635
0.6985	-0.0635	1.0160	-1.0160	-3.1750	-3.2385	-0.6985	-0.3175
-0.1905	-1.0795	-3.8735	-5.0165	-2.2225	-0.0635	0.3175	0.4445
1.2700	5.3340	1.8415	1.4605	1.0795	1.2700	1.2700	1.4605
1.2700	0.4445	2.0320	1.6510	2.4765	2.0320	1.6510	7.3025
4.7625	1.8415	1.4605	1.9050	2.0955	2.7305	1.3970	1.6510
3.6195	2.0320	2.2860	2.1590	1.9685	1.9685	2.4765	2.4130
2.2225	1.3335	2.6035	2.8575	2.8575	2.7940	3.1115	2.6670
2.2225	0.8255	1.2700	1.5240	1.1430	2.0320	0.3175	1.2065
0.3810	1.1430	3.1750	1.5240	1.3335	1.2065	1.3335	1.9050
1.7780	2.4130	4.0005	4.3815	4.5085	5.0165	3.9370	5.7785
5.6515	5.6515	5.2070	5.5245	5.9690	5.0165	4.3815	3.2385
2.0955	-0.4445	-1.3970	-2.0955	-3.3020	-5.0165	-5.2705	-5.5880
-2.7940	-3.6830	-1.6510	0.6985	0.3175	0.5715	-3.5560	-2.7305
1.5240	2.7305	2.0320	2.7940	2.9210	2.8575	2.6670	2.5400
1.7780	0.3175	2.7940	2.6035	2.9210	2.3495	1.0795	2.4765
3.4290	3.4925	3.9370	4.3815	4.3180	4.7625	3.6195	4.0005
3.6830	3.4925	4.9530	5.0800	3.9370	4.8260	4.9530	4.9530
4.0640	5.0165	4.7625	4.3180	4.9530	5.0165	4.1910	5.3975
5.3340	5.3975	5.1435	4.4450	3.5560	3.3020	4.5085	5.5245
6.0325	6.0325	5.9055	5.5245	5.7785	5.6515	6.5405	5.9690
6.2865	10.6680	4.8895	5.4610	2.7940	5.5880	5.5880	5.0165
5.3340	4.6355	2.9210	5.8420	5.2705	3.3655	2.6670	2.2860
3.6830	4.9530	3.9370	5.1435	1.6510	-0.1270	-0.4445	2.7940
4.9530	4.0640	4.1910	4.3815	3.6195	4.8260	4.9530	3.1750

3.8735	4.5720	4.0005	2.4765	1.7145	2.9845	2.9845	2.8575
3.2385	2.9210	2.2860	2.3495	3.2385	3.3655	2.9210	2.8575
3.2385	3.2385	2.9845	2.4130	0.2540	2.4130	2.2860	2.6670
1.7780	2.2860	1.9685	2.0955	1.2700	0.7620	0.3810	1.1430
0.6985	0.8890	1.3335	0.1905	0.6985	0.9525	0.3810	0.6985
0.6350	0.8890	1.2065	1.4605	1.7780	0.6350	1.2065	1.7780
1.3335	0.0635	1.3335	-0.0635	-0.0635	1.7780	-0.2540	-0.0635
-0.6350	-1.3970	-2.5400	0.8890	1.0160	-0.1270	-0.8890	0.4445
0.0635	0.5715	2.1590	0.7620	0.8255	4.3180	7.3660	-0.1905
1.6510	0.0635	-0.3175	-1.1430	-0.4445	-1.5240	-0.2540	-1.5875
-0.5715	-1.5875	-2.2225	4.1275	-1.6510	-1.4605	-2.0955	-2.4765
-2.5400	-3.3020	-3.3655	-3.8100				

Data Set 10

-8.7630	-8.5090	-9.5885	-9.5885	-9.4615	-9.7155	-9.9695	-10.4140
-10.4775	-11.0490	-11.0490	-10.9220	-11.4300	-12.0650	-12.1285	-12.0650
-12.6365	-12.5730	-12.6365	-12.8905	-12.7635	-12.8905	-12.8270	-2.6670
-11.1760	-12.1285	-13.2715	-12.0015	-12.2555	-11.8745	-11.9380	-12.3190
-13.0175	-11.7475	-12.1285	-13.3985	-13.5890	-13.0810	-14.6050	-14.7955
-13.5890	-14.2240	-15.1130	-13.7160	-13.6525	-15.6845	-16.1290	-15.6210
-14.9225	-15.7480	-15.6210	-15.6210	-15.7480	-15.4940	-15.3035	-15.1130
-15.0495	-15.3035	-15.4305	-14.9225	-14.9225	-14.9225	-14.4780	-14.5415
-14.6685	-14.0970	-14.2875	-14.3510	-13.9065	-13.4620	-13.5890	-13.2715
-13.3985	-13.5890	-13.5890	-13.0175	-13.0810	-13.0175	-12.8270	-12.5730
-12.6365	-12.3825	-12.2555	-12.4460	-12.1920	-12.0015	-12.1920	-12.4460
-12.2555	-12.4460	-12.0015	-11.4935	-11.2395	-11.6840	-11.3665	-11.4935
-11.4300	-11.1760	-10.6680	-10.6680	-10.5410	-10.2870	-10.4140	-10.5410
-10.4775	-10.6045	-10.8585	-10.9855	-11.1125	-11.1125	-11.0490	-11.1760
-11.7475	-11.8110	-11.8110	-12.0650	-12.3825	-12.2555	-11.9380	-12.3190
-12.3825	-12.3190	-12.2555	-12.8270	-12.7635	-12.6365	-12.9540	-12.5095
-12.4460	-12.5730	-12.4460	-12.5730	-12.5730	-12.6365	-12.7635	-12.8270
-12.7635	-12.8905	-12.8905	-12.6365	-12.1920	-12.1285	-12.3190	-11.8745
-11.8110	-12.0650	-11.9380	-11.4300	-11.4300	-11.4300	-11.1760	-11.3665
-11.3665	-11.2395	-11.3030	-11.6205	-10.9855	-10.8585	-10.8585	-10.9220
-10.9220	-10.6680	-11.1760	-11.5570	-11.3030	-11.2395	-10.8585	-10.2870
-10.4140	-8.9535	-10.3505	-10.0330	-10.7315	-10.6680	-10.4140	-10.0330
-9.7155	-9.5885	-9.3345	-9.0170	-8.7630	-9.1440	-9.0805	-9.0805

-9.0170	-8.9535	-8.6360	-8.2550	-7.9375	-7.4930	-7.3660	-7.7470
-8.1915	-8.4455	-8.6995	-7.8105	-7.6200	-7.9375	-7.1755	-7.1120
-8.0645	-8.4455	-7.9375	-7.8740	-8.0010	-7.6835	-7.6200	-7.7470
-5.0165	-7.2390	-7.8105	-7.7470	-7.3660	-5.0800	-7.3025	-6.8580
-6.7945	-6.7310	-6.0960	-6.9215	-7.1120	-6.5405	-6.2230	-6.2865
-4.5085	-4.0640	-3.8100	-2.4130	-1.9050	-4.7625	-5.0800	-4.4450
-4.2545	-4.3180	-3.1115	-2.0320	-2.1590	-1.9685	-2.0320	-2.6670
-2.4765	-2.0955	-2.0320	-2.0955	-1.4605	-1.5875	-1.9050	-1.1430
-1.4605	-1.7145	-1.2700	-1.5240	-1.8415	-1.0795	-1.4605	-1.3335
-0.3810	0.3175	-0.8890	-0.4445	0.0000	-0.3175	-1.5240	-0.3810
-0.1905	-0.3810	0.8890	0.2540	-1.0795	-0.3175	1.6510	1.7780
1.5875	1.7145	1.4605	1.5240	1.5240	-0.3175	-0.6350	0.2540
0.4445	0.0000	1.0795	1.2065	1.0160	0.6985	0.3810	-0.3810
-0.1905	-0.5080	-1.4605	-0.6985	-0.3810	-1.2065	-0.0635	3.9370
2.9210	1.0795	0.7620	-2.1590	-3.8735	-2.3495	-1.7145	-2.6035
-1.7145	-1.5875	-2.8575	-3.1750	-3.0480	-3.4925	-3.9370	-3.7465
-3.0480	-2.2860	-1.8415	-2.4130	-2.1590	-2.7305	-3.7465	-4.1275
-2.4130	-2.0955	-3.1750	-2.0320	-1.5240	-2.8575	-2.7940	-1.9685
-2.9845	-3.8735	-2.6670	-2.9845	-3.9370	-2.7305	-1.0160	-1.9685
-1.9685	-2.0955	-2.9210	-2.4765	-2.2860	-3.1750	-3.1750	-2.7305
-3.4925	-3.4290	-2.8575	-4.4450	-5.4610	-4.6355	-5.0800	-5.7150
-3.6830	-3.6195	-4.3815	-3.3655	-3.4925	-4.7625	-3.4925	-2.4130
-4.5720	-5.0800	-4.2545	-5.9055	-6.5405	-3.9370	-3.8735	-4.9530
-2.0320	0.2540	-1.3335	-3.9370	-4.0640	-6.6040	-5.4610	-4.6355
-4.8260	-3.4290	-2.3495	-4.1275	-4.1910	-2.7940	-3.3655	-4.3815
-1.5875	-2.1590	-2.0955	0.8255	-0.5080	-1.5875	-0.5715	0.3175
-1.3335	0.1270	2.7940	1.7780	-0.2540	-0.5080	-1.0795	-2.0955
-2.1590	-2.4130	-2.5400	-2.9845	-3.4925	-2.9845	-2.5400	-2.4130
-2.6670	-1.9685	-1.6510	-1.6510	-1.5875	-1.6510	-1.3335	-1.1430
-1.2700	-1.2700	-1.3970	-1.6510	-1.8415	-1.7145	-1.6510	-1.8415
-1.2700	-1.2065	-1.5240	-1.5240	-1.3335	-1.3335	-1.7780	-1.7780
-2.0320	-2.1590	-2.0320	-2.2225	-2.6035	-2.2225	-2.2860	-2.6035
-2.4130	-2.7305	-2.6035	-2.7305	-2.9845	-3.4290	-3.8735	-3.7465
-3.8735	-4.0005	-3.8100	-3.9370	-4.2545	-4.3180	-5.0800	-5.5245
-5.7785	-6.0325	-6.3500	-6.0325	-6.2230	-6.2230	-6.2230	-6.3500
-6.6040	-6.7310	-7.0485	-7.8105	-7.6200	-7.5565	-8.0010	-8.2550
-8.1280	-8.5725	-8.5725	-8.6360	-8.7630	-8.0010	-7.8740	-7.9375
-8.2550	-8.5090	-8.1915	-8.3185	-8.3185	-7.7470	-7.6200	-7.6200

-7.8740	-8.1915	-8.0010	-8.0010	-7.7470	-7.4930	-8.0645	-7.9375
-6.9850	-7.5565	-8.0010	-7.8740	-7.4930	-8.3185	-8.9535	-8.8265
-8.7630	-9.0170	-8.5090	-8.3820	-8.8265	-8.1280	-7.1755	-7.6835
-9.0170							

A.2 CMM Data Sets

Data sets 11 to 15 represent circular profiles, sampled by 30 points by a coordinate measuring machine. They consist of two-dimensional arrays, where each line corresponds to a data point (x- and y-coordinates). Data sets sub-11 to sub-15 are sub-sampled from these data sets (by taking each other data point, starting from the first one) and are not reproduced separately.

Data Sets					
11		12		13	
x	y	x	y	x	y
39.3110	59.4620	138.6370	61.3910	87.6160	62.3800
39.3410	61.6460	138.7020	63.1600	87.9060	64.3390
40.0850	64.9930	139.1990	65.6040	89.9010	66.8570
41.1440	67.2880	140.2200	68.0740	91.9110	67.5970
42.6340	69.3700	141.4800	69.9990	94.1680	67.4050
45.0170	71.5270	143.5020	72.0880	96.2210	65.9810
46.7170	72.5660	145.8570	73.6560	97.3530	64.2020
48.5500	73.3370	148.5720	74.7250	97.6120	62.6170
50.5130	73.8490	151.7760	75.2030	96.6390	59.6200
52.7150	74.0510	154.4770	74.9980	94.7110	58.0800
54.2110	74.0080	157.2170	74.2050	93.1810	57.6790
55.8140	73.7810	160.1330	72.5750	90.8750	57.9400
57.9140	73.1350	162.9170	69.8300	89.3080	58.9460
60.2620	71.9580	164.0480	68.0270	87.6640	61.8180
63.1430	69.5120	165.1790	65.1970	87.6650	63.3640
64.4550	67.8000	165.6350	62.2620	88.6820	65.7100
65.5180	65.8080	165.4780	59.5830	91.6200	67.5430
66.1580	63.9200	164.6180	56.9960	93.2970	67.6070
66.5270	62.0910	163.0820	53.7390	96.6760	65.5390
66.6330	60.2090	160.6330	51.1390	97.5360	63.4870
66.5370	58.7360	158.3260	49.5890	97.0440	60.3220
66.2390	57.1190	155.9460	48.6910	94.8570	58.1590
65.7760	55.5790	153.2640	48.1950	92.8330	57.6380
65.0440	53.9690	150.4290	48.2550	91.4570	57.7690

data set 11 (cont.)		data set 12 (cont.)		data set 13 (cont.)	
x	y	x	y	x	y
63.7780	52.0000	147.8200	48.8510	89.6710	58.6040
62.1850	50.2680	144.9600	50.2100	88.2590	60.1640
59.8150	48.5270	141.7930	52.9740	88.4380	65.3860
58.7080	47.9530	139.9760	55.7870	92.9200	67.6460
57.1800	47.3490	139.1390	57.9830	95.8430	58.8080
55.4100	46.9050	138.7340	59.9360	90.5320	58.0820

Data Sets							
14		15		continuation			
x	y	x	y	x	y	x	y
59.0320	21.0610	70.1900	40.9140	65.4810	12.6070	77.5330	35.3720
59.5570	24.0890	70.1740	42.2590	62.9030	13.7960	76.0560	35.3120
61.6910	27.3860	70.6600	44.2310	60.8000	15.7940	74.6280	35.5880
64.8780	29.3970	71.7060	45.9420	59.3750	18.6670	73.3170	36.1670
68.0730	29.9110	73.8170	47.6000	59.0290	21.1920	72.0700	37.1090
70.9040	29.3640	75.4240	48.1640	60.5890	26.0970	70.2680	40.4010
74.3400	27.0590	77.9370	48.1660	63.1320	28.5310	70.2230	42.6860
75.6560	25.1410	79.7840	47.4970	67.0100	29.8700	72.3140	46.5670
76.4110	23.1120	82.2720	45.0370	71.0650	29.3060	76.0200	48.2570
76.6320	20.4460	82.8690	43.5810	75.3860	25.6650	78.9180	47.8900
75.9420	17.6390	83.0910	41.9950	76.0090	17.7680	81.9310	45.5720
74.4080	15.2050	82.7560	39.8540	72.5890	13.6700	82.6480	39.5620
72.0720	13.3560	82.0190	38.3080	64.6590	12.8730	79.3470	35.9450
70.4540	12.6720	79.7100	36.1390	60.3210	16.5050	77.5210	35.3650
68.0230	12.2810	78.8390	35.7330	59.1280	19.8630	72.2830	36.9220

Data sets square-1-1 and square-1-2 to square7-1 and square-7-2 represent planar surfaces sampled by a coordinate measuring machine, where each pair of data sets (e. g. square-1-1 and square-1-2) represent two related surfaces. They consist of three-dimensional arrays, where each line corresponds to a data point (x-, y- and z-coordinates). Data sets square-1-1 and square-2-1 contain 12 data points, while data sets square-1-2 and square-2-2 are equal and contain 25 points. All the other data sets contain 6 data points each.

Data Sets					
square-1-1			square-1-2 / square-2-2		
x	y	z	x	y	z
58.455	66.310	8.197	72.416	64.594	25.748
59.090	49.890	8.188	65.570	61.092	25.737
59.575	36.709	10.512	72.577	55.092	25.740
59.024	51.638	9.944	71.752	48.506	25.737
58.422	68.467	11.234	72.742	38.383	25.727
58.608	62.611	14.044	78.967	39.364	25.739
58.980	52.691	12.882	80.727	52.295	25.749
59.447	40.085	16.582	82.367	62.354	25.754
59.519	37.950	18.499	86.700	56.603	25.750
59.225	45.891	19.530	88.650	49.086	25.755
58.860	55.268	17.796	88.472	38.130	25.703
58.500	64.824	19.464	96.005	38.009	25.761
square-2-1			98.032	51.200	25.769
x	y	z	98.248	64.416	25.772
141.432	70.876	7.821	113.622	64.167	25.786
141.805	61.350	8.853	114.193	56.960	25.791
142.171	51.787	12.085	114.997	51.007	25.788
142.450	49.201	9.685	115.571	41.904	25.787
142.693	37.789	11.053	120.410	40.425	25.793
142.300	48.132	11.489	128.835	49.782	25.802
142.125	52.681	10.522	135.241	38.664	25.818
141.900	58.662	12.648	133.240	49.982	25.811
141.629	65.646	16.894	130.834	58.376	25.799
142.051	54.523	21.451	134.987	63.322	25.809

square-2-1 (cont.)			square-1-2 / square-2-2 (cont.)		
x	y	z	x	y	z
142.314	47.702	20.323	130.867	57.198	25.808
142.747	36.455	21.678	-	-	
square-3-1			square-3-2		
x	y	z	x	y	z
89.3980	43.3350	24.9190	25.6000	41.0420	17.2610
84.0720	35.0420	27.7680	37.9670	55.4860	22.1510
80.0290	24.2990	27.7670	40.4740	71.7390	19.7100
74.6850	28.3410	36.4810	54.7730	65.7380	31.1750
78.0150	37.2630	36.4840	56.2180	45.4330	37.4160
77.9850	47.2290	41.2560	41.5740	24.3980	32.6360
square-4-1			square-4-2		
x	y	z	x	y	z
43.4860	46.9580	51.8590	84.6440	39.9690	33.7100
39.0110	57.7180	51.9350	86.3010	56.2830	36.0870
58.8860	72.0740	51.9770	88.1190	74.1670	33.8600
57.8170	59.4690	51.9110	87.9820	72.8640	40.8700
56.3660	42.3670	51.8160	85.4370	47.7410	40.8690
71.6630	41.0690	51.7850	83.8290	31.9530	38.7180
square-5-1			square-5-2		
x	y	z	x	y	z
38.8240	47.2810	51.8890	36.0780	91.5520	34.7720
55.1210	42.9320	51.8180	53.5320	89.9810	34.7740
51.7150	56.9900	51.9060	70.8830	88.4110	35.5070
66.5200	56.2980	51.8770	57.2040	89.6450	40.3900
67.5840	68.8190	51.9400	47.5650	90.5150	44.0370
39.3710	71.6340	52.0220	78.1170	87.7550	44.0400
square-6-1			square-6-2		
x	y	z	x	y	z
40.3990	46.3580	61.5190	99.4000	37.1120	44.0310
41.6040	60.5650	61.5330	100.7280	51.3670	46.0780
54.9930	66.6290	61.5430	101.9100	64.1130	43.8290
65.1440	47.8320	61.5250	101.6410	61.1000	54.9770
79.2680	63.1730	61.5350	100.8080	52.3220	50.5260
94.0900	43.5740	61.5110	99.2390	35.3910	55.6290

square-7-1			square-7-2		
x	y	z	x	y	z
75.5510	-47.8690	77.0380	78.1810	102.2570	89.0000
107.9810	-27.0730	95.5250	117.7700	101.5100	90.4540
157.7900	-2.9340	116.8790	154.1110	87.9990	106.0780
153.7800	-50.2500	74.4420	74.9990	71.4260	123.2730
121.8490	16.7820	134.7880	94.5640	49.8340	147.6550
71.0300	17.9960	136.1980	154.1960	47.9740	150.6600

A.3 Simulated Data Sets

Data sets s-1, s-2 and s-3 contain 30 sampled data points each, representing circular profiles generated as described in section 5.5.4. They consist of two-dimensional arrays, where each line corresponds to a (x, y) data point. Data sets sub-s1 to sub-s3 are sub-sampled from these data sets (again by taking each other data point, starting from the first one) and are not reproduced separately.

Data Sets					
s-1		s-2		s-3	
x	y	x	y	x	y
150.0081	100.0000	210.0002	200.0000	7.0017	5.0000
148.9221	110.3987	209.7985	202.0827	6.9614	5.4169
145.6999	120.3469	209.1526	204.0750	6.8299	5.8147
140.4714	129.4042	208.0907	205.8782	6.6164	6.1744
133.4690	137.1711	206.6800	207.4189	6.3391	6.4872
125.0038	143.3079	204.9916	208.6458	6.0022	6.7358
115.4502	147.5508	203.0904	209.5114	5.6190	6.9051
105.2245	149.7080	201.0472	209.9635	5.2091	6.9890
94.7747	149.7153	198.9527	209.9642	4.7910	6.9887
84.5505	147.5487	196.9097	209.5109	4.3806	6.9064

s-1 (cont.)		s-2 (cont.)		s-3 (cont.)	
x	y	x	y	x	y
74.9999	143.3014	195.0077	208.6469	3.9983	6.7350
66.5324	137.1696	193.3203	207.4186	3.6612	6.4869
59.5285	129.4042	191.9093	205.8782	3.3822	6.1754
54.3017	120.3462	190.8485	204.0745	3.1716	5.8141
51.0791	110.3985	190.1997	202.0831	3.0406	5.4165
49.9942	100.0000	189.9990	200.0000	2.9973	5.0000
51.1014	89.6063	190.2337	197.9241	3.0465	4.5848
54.3392	79.6705	190.8790	195.9391	3.1728	4.1865
59.5638	70.6214	191.9086	194.1212	3.3784	3.8218
66.5501	62.8501	193.2960	192.5544	3.6600	3.5118
74.9961	56.6920	194.9907	191.3236	4.0007	3.2692
84.5453	52.4355	196.9095	190.4884	4.3818	3.0973
94.7712	50.2512	198.9564	190.0707	4.7906	3.0078
105.2294	50.2456	201.0435	190.0720	5.2094	3.0080
115.4552	52.4337	203.0904	190.4886	5.6181	3.0977
125.0035	56.6927	205.0091	191.3240	5.9988	3.2700
133.4488	62.8514	206.7030	192.5556	6.3394	3.5125
140.4402	70.6185	208.0907	194.1218	6.6223	3.8213
145.6653	79.6685	209.1203	195.9394	6.8269	4.1866
148.9032	89.6053	209.7648	197.9244	6.9560	4.5842

Data sets s4h-1, s4h-2 and s4h-3 contain 60 sampled data points each, representing 4 four related circular profiles, where each of the profiles was sampled by 15 points and generated as described in section 5.5.4. Each data sets consist of a two-dimensional array, where each line corresponds to a data point (x- and y-coordinates). The data points representing each profile are listed in sequence such that the first, second, third and forth quarters of the arrays correspond respectively to the bottom left and right and top left and right profiles.

Data Sets					
s4h-1		s4h-2		s4h-3	
x	y	x	y	x	y
60.0051	50.0000	70.0008	50.0000	8.0005	5.0000
59.1903	54.0918	68.3507	58.1703	7.7448	6.2221
56.7170	57.4599	63.3250	64.7989	7.0025	7.2240
53.0801	59.4796	56.1805	69.0216	5.9311	7.8658
48.9606	59.8895	47.9003	69.9769	4.6878	7.9700
45.0000	58.6603	40.0431	67.2458	3.4907	7.6141
41.8632	55.9118	33.8190	61.7562	2.5870	6.7531
40.1838	52.0865	30.3520	54.1763	2.0447	5.6282
40.2476	47.9271	30.5210	45.8596	2.0849	4.3804
41.9529	44.1534	33.8196	38.2443	2.5575	3.2254
44.9983	41.3368	39.9565	32.6041	3.5083	2.4163
48.9487	39.9977	47.9185	30.1957	4.6849	2.0016
53.1011	40.4558	56.1806	30.9781	5.9238	2.1570
56.6692	42.5932	63.4411	35.0722	7.0130	2.7644
59.0837	45.9557	68.1926	41.9001	7.7378	3.7811
160.0048	50.0000	370.0003	50.0000	38.0018	5.0000
159.1913	54.0922	368.3506	58.1702	37.7459	6.2226
156.7176	57.4607	363.3251	64.7990	37.0030	7.2246
153.0801	59.4797	356.1804	69.0212	35.9311	7.8655
148.9602	59.8934	347.9003	69.9775	34.6878	7.9701
145.0000	58.6603	340.0432	67.2456	33.4905	7.6146
141.8601	55.9140	333.8192	61.7560	32.5876	6.7527
140.1807	52.0872	330.3523	54.1762	32.0461	5.6279
140.2490	47.9274	330.5211	45.8596	32.0831	4.3800
141.9556	44.1554	333.8192	38.2440	32.5561	3.2244

s4h-1 (cont.)		s4h-2 (cont.)		s4h-3 (cont.)	
x	y	x	y	x	y
144.9974	41.3352	339.9564	32.6040	33.5081	2.4159
148.9483	39.9934	347.9185	30.1957	34.6848	2.0010
153.1027	40.4510	356.1806	30.9781	35.9236	2.1575
156.6714	42.5906	363.4407	35.0726	37.0135	2.7638
159.0842	45.9555	368.1926	41.9002	37.7371	3.7814
60.0020	150.0000	70.0003	350.0000	8.0013	35.0000
59.1923	154.0927	68.3509	358.1703	7.7446	36.2220
56.7163	157.4592	63.3251	364.7990	7.0029	37.2244
53.0803	159.4803	56.1806	369.0218	5.9313	37.8662
48.9606	159.8889	47.9003	369.9770	4.6880	37.9688
44.9972	158.6652	40.0433	367.2456	3.4906	37.6144
41.8599	155.9141	33.8195	361.7558	2.5876	36.7527
40.1789	152.0875	30.3519	354.1763	2.0457	35.6280
40.2497	147.9275	30.5209	345.8596	2.0840	34.3802
41.9527	144.1533	33.8196	338.2443	2.5562	33.2245
44.9983	141.3368	39.9563	332.6038	3.5086	32.4168
48.9487	139.9974	47.9185	330.1956	4.6848	32.0013
53.1012	140.4555	56.1804	330.9787	5.9237	32.1573
56.6701	142.5922	63.4409	335.0724	7.0138	32.7634
59.0875	145.9540	68.1922	341.9003	7.7382	33.7809
160.0025	150.0000	370.0007	350.0000	38.0020	35.0000
159.1920	154.0925	368.3507	358.1703	37.7445	36.2220
156.7189	157.4621	363.3251	364.7990	37.0028	37.2243
153.0801	159.4795	356.1806	369.0221	35.9307	37.8645
148.9606	159.8895	347.9003	369.9773	34.6878	37.9701
144.9976	158.6645	340.0430	367.2461	33.4912	37.6132
141.8617	155.9128	333.8189	361.7562	32.5868	36.7533
140.1796	152.0874	330.3521	354.1763	32.0451	35.6281
140.2501	147.9276	330.5216	345.8597	32.0844	34.3803
141.9524	144.1531	333.8190	338.2438	32.5561	33.2244
144.9972	141.3349	339.9565	332.6041	33.5082	32.4160
148.9482	139.9931	347.9184	330.1949	34.6847	31.9998
153.1020	140.4531	356.1803	330.9789	35.9236	32.1576
156.6689	142.5935	363.4408	335.0725	37.0130	32.7644
159.0867	145.9544	368.1925	341.9002	37.7380	33.7810

Data sets sub-4h01 to sub-4h03 are sub-sampled from the previous similar data sets (again by taking each other data point, starting from the first one) and are not reproduced separately. Data sets sub-4h04 and sub-4h05 contain 20 sampled data points each, representing four related circular profiles sampled by 5 points each and generated as described in section 6.5.4. The sequence in which the data points are listed is the same as before.

Data Sets			
sub-4h04		sub-4h05	
x	y	x	y
150.0006	100.0000	15.0010	10.0000
115.4511	147.5536	11.5726	14.8401
59.5485	129.3897	6.0701	12.8552
59.5485	70.6103	5.8386	6.9765
115.4509	52.4470	11.5183	5.3272
750.0003	100.0000	65.0010	10.0000
715.4512	147.5540	61.5729	14.8410
659.5479	129.3902	56.0703	12.8551
659.5489	70.6106	55.8394	6.9772
715.4513	52.4459	61.5178	5.3286
150.0013	700.0000	15.0000	60.0000
115.4511	747.5537	11.5723	64.8392
59.5477	729.3903	6.0702	62.8551
59.5483	670.6101	5.8391	56.9769
115.4509	652.4471	11.5181	55.3278
750.0014	700.0000	65.0011	60.0000
715.4512	747.5539	61.5726	64.8401
659.5485	729.3897	56.0696	62.8556
659.5479	670.6098	55.8394	56.9771
715.4510	652.4466	61.5182	55.3275

Appendix B

Extracts of Source Codes

This appendix reproduces the main fragments of the C codes written for some of the algorithms used in this thesis. The codes reproduced below are limited to those which are not standard or commercially available implementations and are listed in the following sequence:

- the iteration process of the gauge algorithm [Fletcher, 1970a];
- the genetic search evaluation functions for the first and second inspection problems;
- the ring and zone exchange algorithms [Chetwynd and Phillipson, 1980];
- the least squares circle and plane algorithms [Forbes, 1989].

gauge.c

```c
/* This function performs the iterations of the Fletcher algorithm [1970a] */

int gauge(data, vector, Amin, Amax, Bmin, Bmax, Z)

   struct DATA   *data;
   struct VECTOR *vector;
   float         Amin, Amax, Bmin, Bmax, Z;

#define N          (int)(data->nb_elt)
#define r(i)       (float)(data->radius[i])
#define B(i)       (float)(vector->b[i])
#define X(i)       (float)(vector->x[i])
#define A(i,j)     (float)(vector->a[3*i+j])
#define V(i)       (int)(v[i])
#define AT(i,j)    (float)(vector->at[(2*N+4)*i+j])
#define C(i,j)     (float)(vector->c[3*i+j])
#define Cinv(i,j)  (float)(vector->ci[3*i+j])
#define Cel(i)     (int)(vector->cel[i])
#define Na(i)      (float)(vector->na[i])
#define La(i)      (float)(vector->la[i])
#define U(i)       (float)(vector->u[i])
#define Y(i)       (float)(vector->y[i])
#define E(i,j)     (float)(vector->e[3*i+j])
#define Temp(i,j)  (float)(vector->temp[3*i+j])
#define Eta(i)     (float)(vector->eta[i])

{
   int         i, j, f, fl, *v, m, K, oci, nci, nca, ncb;
   float       theta, alpha, spl, sp2, beta, D;

   vector->b=fl_mem_alloc(2*N+4);
   vector->a=fl_mem_alloc(3*N);
   v=int_mem_alloc(2*N);

/* setup vector BV(i) made up of variable boundaries and data set elements */

   set_vect_b(vector,Amax,Amin,Bmax,Bmin,Z);

/* calculate cos sin matrix: A(i,0)=cos(2*pi*i/(N-1))
                             A(i,1)=sin(2*pi*i/(N-1)) and  A(i,2)=1  */
   sin_cos_matrix(vector, N);

/* set up matrix AT(i,j) = [I -I A], whose columns are the normal
     vectors of the constraints  */
   set_matrix(vector, N);

/* set basis matrix C(i,j) (whose columns are the coefficients of the
     constraints which define the vertex), and inverse of C, Cinv(i,j)
     for initial vertex as X=[Amax, Bmax, Rmax] */
```

gauge.c

```c
   set_basis(vector);

/* define initial vertex and new inverse of basis matrix */

   init_vertex(vector, data, Amin, Amax, Bmin, Bmax, &nci);

/* check for violation of the set of constraints */

   constr_viol(vector, v, &m);

/* if there isn't constraint violations, the initial solution is feasible*/
   if(m==0) {
      cfree(vector->a);
      cfree(vector->b);
      cfree(v);
      return(OK);
   }
   else {            /* start iterations of the Fletcher algorithm */
      while(m!=0) {
/* set nabla as sum of the violated constraints */
         for(i=0;i<3;i++)
            Na(i)=0;
         for(j=0;j< m;j++) {
            K=V(j);
            for(i=0;i<3;i++)
               Na(i)=Na(i)+AT(i,K);
         }
/* calculate lambda as Cinv(i,j)*nabla(i) */
         for(i=0;i<3;i++) {
            La(i)=0;
            for(j=0;j<3;j++)
               La(i)=La(i)+Cinv(i,j)*Na(j);
         }

         theta=0;
         oci=0;
         for(i=0;i<3;i++)
            if(La(i) > theta) {
               theta=La(i);
               oci=i;
            }
/* if the largest lambda (theta) is larger than zero, then there is a
   direction to search                                                 */

         if(theta > 1.0e-6) {
            for(i=0;i<3;i++)
               U(i)=Cinv(oci,i);

/* find the furthest violated constraint to the vertex and store this value
   in alpha and the index of the corresponding constraint in nca          */
```

```
            alpha=0;
            nca=0;
            for(j=0;j<m;j++) {
               sp1=0;
               sp2=0;
               K=V(j);
               for(i=0;i<3;i++)
                  sp1=sp1+U(i)*AT(i,K);

               if(sp1 > 0) {
                  for(i=0;i<3;i++)
                     sp2=sp2+AT(i,K)*X(i);
                  D=(B(K)-sp2)/sp1;
                  if(D > alpha) {
                     alpha=D;
                     nca=K;
                  }
               }
            }
/* check if there are constraints not in the basis neither amongst the
   violated ones, which are intersected by the direction u. In case there
   are, find the distance of the closest constraint to the vertex and store
   this value in beta and the index of the corresponding constraint in ncb.*/

            ncb=0;
            beta=1.0e+20;
            for(j=0;j<2*N+4;j++) {
               f=0;
               f1=0;
               for(i=0;i<3;i++)
                     if(j == Cel(i)) f=1;
               for(i=0;i<m;i++)
                     if(j == V(i))    f1=1;
               if(f != 1 && f1 != 1) {
                  sp1=0;
                  for(i=0;i<3;i++)
                     sp1=sp1+U(i)*AT(i,j);
                  if(sp1 < 0) {
                     sp2=0;
                     for(i=0;i<3;i++)
                        sp2=sp2+AT(i,j)*X(i);
                     D=(B(j)-sp2)/sp1;
                     if(D < beta) {
                        beta=D;
                        ncb=j;
                     }
                  }
               }
            }
```

```
            if(beta < alpha) {
               for(i=0;i<3;i++)
                     X(i)=X(i)+beta*U(i);
               nci=ncb;
            }
            else {
               for(i=0;i<3;i++)
                     X(i)=X(i)+alpha*U(i);
               nci=nca;
            }

            for(i=0;i<3;i++) {
               C(i,oci)=AT(i,nci);
               Cel(oci)=nci;
            }

            constr_viol(vector, v, &m); /* check violation of constraints */

            if(m != 0)
/* calculate the inverse by the product form of the inverse [Hadley, pp 48]*/
            inverse(vector, oci);
         }
         else
            return(NOTOK);
      }
   }
   cfree(vector->a);
   cfree(vector->b);
   cfree(v);
   return(OK);
}
#undef  N
#undef  r(i)
#undef  B(i)
#undef  X(i)
#undef  A(i,j)
#undef  V(i)
#undef  AT(i,j)
#undef  C(i,j)
#undef  Cinv(i,j)
#undef  Cel(i)
#undef  Na(i)
#undef  La(i)
#undef  U(i)
#undef  Y(i)
#undef  E(i,j)
#undef  Temp(i,j)
#undef  Eta(i)
```

ga_eval.c

```
/* This function evaluates  whether the constraints are violated and return
   the penalty term, for the first inspection problem formulated as in
   (5.6)  */

double eval(str, length, vect, genes, penalty,tol_zone)

  char str[];      /* string representation                        */
  int length;      /* length of bit string                         */
  double vect[];   /* floating point representation                */
  int genes;       /* number of elements in vect                   */
  float penalty;     /* penalty parameter                            */
  float tol_zone;    /* tolerance zone  */


  {
    int   i,c,K;
    float vu,vl,vcp,sum;

    sum=0;
    c=0;

 /* check contraint violations */

    for(i=0;i<data->nb_elt;i++) {
       vl=((data->x[i]-vect[0])*(data->x[i]-vect[0])+(data->y[i]-vect[1])*
          (data->y[i]-vect[1]))-vect[2]*vect[2];
       vu=((data->x[i]-vect[0])*(data->x[i]-vect[0])+(data->y[i]-vect[1])*
          (data->y[i]-vect[1]))-((vect[2]+tol_zone)*(vect[2]+tol_zone));
       if(vu>tol) {
         if(vu>1.0)
           sum += (vu*vu);
         else
           sum += vu;
         c++;
       }
       if(vl<-1*tol) {
         vl *= -1;
         if (vl>1.0)
            sum += (vl*vl);
         else
            sum += vl;
         c++;
       }
    }

    vcp=((xo-vect[0])*(xo-vect[0])+(yo-vect[1])*(yo-vect[1])-1*
        (tol_cp*tol_cp/4));
    if(vcp > tol) {
      if (vcp>1.0)
        sum += (vcp*vcp);
```

ga_eval.c

```
      else
        sum += vcp;
      c++;
    }

/* if there is any violation return 'sum', the penalty parameter */

    sum  *= penalty;

/* otherwise, Feasflag is ste to 1, to terminate the search */

    if(c == 0)  Feasflag=1;

    return((double)(sum));
}
```

ga_eval_4h.c

```
double eval(str, length, vect, genes, penalty,tol_zone)

/* This function evaluates whether the constraints are violated and return
   the penalty term, for the second problem formulated as in (5.7) and
   (5.8)      */


  char str[];     /* string representation                        */
  int length;     /* length of bit string                         */
  double vect[];  /* floating point representation                */
  int genes;      /* number of elements in vect                   */
  float penalty;    /* penalty parameter                            */
  float tol_zone;   /* tolerance zone  */


{
    int   i,c,K;
    float vu1,vl1,vu2,vl2,vu3,vl3,vu4,vl4,vcpl1,
          vcpl2,vcpl3,vcpl4,vcpu1,vcpu2,vcpu3,vcp
          u4,l1,l2,l3,l4,sum;
    FILE  *f;


    sum=0;
    c=0;


    for(i=0;i<N/4;i++) {
       vu1=((data->x[i]-vect[0])*(data->x[i]-vect[0])+(data->y[i]-vect[1])*
           (data->y[i]-vect[1]))-((nom_rad+tol_zone)*(nom_rad+tol_zone));
       vu2=((data->x[i+N/4]-vect[2])*(data->x[i+N/4]-vect[2])+
           (data->y[i+N/4]- vect[3])*(data->y[i+N/4]-vect[3]))-
            ((nom_rad+tol_zone)*(nom_rad+tol_zone));
       vu3=((data->x[i+N/2]-vect[4])*(data->x[i+N/2]-vect[4])+
           (data->y[i+N/2]-vect[5])*(data->y[i+N/2]-vect[5]))-
           ((nom_rad+tol_zone)*(nom_rad+tol_zone));
       vu4=((data->x[i+3*N/4]-vect[6])*(data->x[i+3*N/4]-vect[6])+
           (data->y[i+3*N/4]-vect[7])*(data->y[i+3*N/4]-vect[7]))-
           ((nom_rad+tol_zone)*(nom_rad+tol_zone));
       vl1=((data->x[i]-vect[0])*(data->x[i]-vect[0])+(data->y[i]-vect[1])*
           (data->y[i]-vect[1]))-((nom_rad-tol_zone)*(nom_rad-tol_zone));
       vl2=((data->x[i+N/4]-vect[2])*(data->x[i+N/4]-vect[2])+
           (data->y[i+N/4]-vect[3])*(data->y[i+N/4]-vect[3]))-
           ((nom_rad-tol_zone)*(nom_rad-tol_zone));
       vl3=((data->x[i+N/2]-vect[4])*(data->x[i+N/2]-vect[4])+
           (data->y[i+N/2]-vect[5])*(data->y[i+N/2]-vect[5]))-
           ((nom_rad-tol_zone)*(nom_rad-tol_zone));
       vl4=((data->x[i+3*N/4]-vect[6])*(data->x[i+3*N/4]-vect[6])+
           (data->y[i+3*N/4]-vect[7])*(data->y[i+3*N/4]-vect[7]))-
           ((nom_rad-tol_zone)*(nom_rad-tol_zone));
```

ga_eval_4h.c

```
       if(vu1>tol) {
         if(vu1>1.0)
            sum += (vu1*vu1);
         else
            sum += (vu1);
         c++;
       }
       if(vu2>tol) {
         if(vu2>1.0)
              sum += (vu2*vu2);
         else
              sum += (vu2);
         c++;
       }
       if(vu3>tol) {
         if(vu3>1.0)
            sum += (vu3*vu3);
         else
            sum += (vu3);
         c++;
       }
       if(vu4>tol) {
         if(vu4>1.0)
            sum += (vu4*vu4);
         else
            sum += (vu4);
         c++;
       }
       if(vl1+tol<0) {
         vl1 *= -1;
         if(vl1>1.0)
           sum += (vl1*vl1);
         else
           sum += (vl1);
         c++;
       }
       if(vl2+tol<0) {
         vl2 *= -1;
         if(vl2>1.0)
            sum += (vl2*vl2);
           else
            sum += (vl2);
         c++;
       }

       if(vl3+tol<0) {
         vl3 *= -1;
         if(vl3>1.0)
           sum += (vl3*vl3);
```

```
      else
        sum += (vl3);
      c++;
    }
    if(vl4+tol<0) {
      vl4 *= -1;
      if(vl4>1.0)
          sum += (vl4*vl4);
        else
          sum += (vl4);
      c++;
    }
  }

  vcpl1=(vect[0]-vect[2])*(vect[0]-vect[2])+(vect[1]-vect[3])*
         (vect[1]-vect[3])-(L-tol_cp)*(L-tol_cp);
  vcpu1=(vect[0]-vect[2])*(vect[0]-vect[2])+(vect[1]-vect[3])*
        (vect[1]-vect[3])-(L+tol_cp)*(L+tol_cp);
  vcpl2=(vect[0]-vect[4])*(vect[0]-vect[4])+(vect[1]-vect[5])*
        (vect[1]-vect[5])-(L-tol_cp)*(L-tol_cp);
  vcpu2=(vect[0]-vect[4])*(vect[0]-vect[4])+(vect[1]-vect[5])*
        (vect[1]-vect[5])-(L+tol_cp)*(L+tol_cp);
  vcpl3=(vect[2]-vect[6])*(vect[2]-vect[6])+(vect[3]-vect[7])*
        (vect[3]-vect[7])-(L-tol_cp)*(L-tol_cp);
  vcpu3=(vect[2]-vect[6])*(vect[2]-vect[6])+(vect[3]-vect[7])*
        (vect[3]-vect[7])-(L+tol_cp)*(L+tol_cp);
  vcpl4=(vect[4]-vect[6])*(vect[4]-vect[6])+(vect[5]-vect[7])*
        (vect[5]-vect[7])-(L-tol_cp)*(L-tol_cp);
  vcpu4=(vect[4]-vect[6])*(vect[4]-vect[6])+(vect[5]-vect[7])*
        (vect[5]-vect[7])-(L+tol_cp)*(L+tol_cp);

  if(vcpl1+tol < 0) {
    vcpl1 *= -1;
    if(vcpl1>1.0)
      sum += (vcpl1*vcpl1);
    else
      sum += (vcpl1);
    c++;
  }
  if(vcpu1-tol > 0) {
    if(vcpu1>1.0)
      sum += (vcpu1*vcpu1);
    else
      sum += (vcpu1);
    c++;
  }
  if(vcpl2+tol < 0) {
    vcpl2 *= -1;
    if(vcpl2>1.0)
       sum += (vcpl2*vcpl2);
```

```
      else
        sum += (vcpl2);
      c++;
    }
    if(vcpu2-tol > 0) {
      if(vcpu2>1.0)
        sum += (vcpu2*vcpu2);
      else
        sum += (vcpu2);
      c++;
    }
    if(vcpl3+tol < 0) {
      vcpl3 *= -1;
      if(vcpl3>1.0)
        sum += (vcpl3*vcpl3);
      else
        sum += (vcpl3);
      c++;
    }
    if(vcpu3-tol > 0) {
      if(vcpu3>1.0)
        sum += (vcpu3*vcpu3);
      else
        sum += vcpu3;
      c++;
    }
    if(vcpl4+tol < 0) {
      vcpl4 *= -1;
      if(vcpl4>1.0)
        sum += (vcpl4*vcpl4);
      else
        sum += vcpl4;
      c++;
    }
    if(vcpu4-tol > 0) {
      if(vcpu4>1.0)
        sum += (vcpu4*vcpu4);
      else
        sum += vcpu4;
      c++;
    }

    sum  *= penalty;

    if(c==0)
      Feasflag=1;

    return((double)(sum));
  }
```

ring_exchange.c

```
/*   This program implements the ring exchange algorithm as presented by
     D. G. Chetwynd, 1980.                                                  */


void ring_exchange(data, AA, BB, RR, ZZ)

struct DATA *data;
float       *AA, *BB, *RR, *ZZ;

#define N         (int)(data->nb_elt)
#define r(i)      (float)(data->radius[i])
#define C(i)      (float)(vector->c[i])
#define S(i)      (float)(vector->s[i])


{
   struct VECTOR *vector;
   float         A, B, R, Z, V, T, S12, S32, C12, C32;
   int           i, J, k, N2, max, I1, I2, I3;

   vector=(struct VECTOR *)calloc(1,sizeof(struct VECTOR));

   vector->c=fl_mem_alloc(N);
   vector->s=fl_mem_alloc(N);


/* calculate sin and cos functions */

   sin_cos(vector, N);

/* first estimate of diameter */
   N2=(N-1)/2;
   J=0;
   V=r(0)+r(N2);
   for(i=0;i<N2;i++) {
       T=r(i)+r((N2+i));
       if(T > V) {
          V=T;
          J=i;
       }
   }
   R=V/2;
   V=(r(J)-r((J+N2)))/2;
   A=V*C(J);
   B=V*S(J);


/* check violation of constraints */

   max=-1;
```

ring_exchange.c

```
   V=1.0e-5;
   for(i=0;i<N;i++) {
       T=r(i)-(A*C(i)+B*S(i)+R);
       if(T > V) {
          V=T;
          max=i;
       }
   }

   if(max == -1) {
      *AA=A;
      *BB=B;
      *RR=R;
      *naop=aop;
      cfree(vector);
      cfree(vector->c);
      cfree(vector->s);
      return;
   }

   I3=J+N2;    /* 3 point contact, order J, J+N2, max */
   if(max < I3) {
      if(max < J) {
         I1=max;
         I2=J;
      }
      else {
         I1=J;
         I2=max;
      }
   }
   else {
      I1=J;
      I2=I3;
      I3=max;
   }

   while(max != -1) {  /* do the exchange procedure */

/* calculate reference based on I1 I2 I3 */

       S12=S(I1)-S(I2);
       S32=S(I3)-S(I2);
       C12=C(I1)-C(I2);
       C32=C(I3)-C(I2);


       A=((r(I1)-r(I2))*S32 - (r(I3)-r(I2))*S12)/(C12*S32-C32*S12);

       if(fabs((double)(S12)) < fabs((double)(S32)))
```

ring_exchange.c

```
      B=(r(I3)-r(I2)-A*C32)/S32;
   else
      B=(r(I1)-r(I2)-A*C12)/S12;
   R=r(I1)-A*C(I1)-B*S(I1);

   max=-1;      /* check for violation */
   V=1.e-5;
   for(i=0;i<N;i++) {
      T=r(i)-(A*C(i)+B*S(i)+R);
      if(T > V) {
         V=T;
         max=i;
      }
   }
   if(max != -1) { /* place max in reference using PI criterion */
      if(max > I1) {
         if(max > I2) {
            if(max > I3) {
               if((max-I2) < N2) {
                  I3=max;
               }
               else {
                  I1=I2;
                  I2=I3;
                  I3=max;
               }
            }
            else {
               if((max-I1) < N2) {
                  I2=max;
               }
               else {
                  I3=max;
               }
            }
         }
         else {
            if((I3-max) > N2) {
               I1=max;
            }
            else {
               I2=max;
            }
         }
      }
      else {
         if((I2-max) < N2) {
            I1=max;
         }
         else {
```

ring_exchange.c

```
               I3=I2;
               I2=I1;
               I1=max;
            }
         }
      }
   }
   Z=0;
   for(i=0;i<N;i++) {
      T=r(i)-(A*C(i)+B*S(i)+R);
      if(T < Z)
         Z=T;
   }

   *AA=A;
   *BB=B;
   *ZZ=(float)(fabs((double)(Z)));
   *RR=R;
   cfree(vector);
   cfree(vector->c);
   cfree(vector->s);
   return;
}

#undef N
#undef r(i)
#undef C(i)
#undef S(i)
```

zone_exchange.c

```
/* This program implements the minimum  zone exchange algorithm as
   presented by D. G. Chetwynd, 1980 and described in section 2.4.4. */

void zone_exchange(data, AA, BB, RR, HH)

struct DATA *data;
float       *AA, *BB, *RR, *HH;

#define N          (int)(data->nb_elt)
#define r(i)       (float)(data->radius[i])
#define C(i)       (float)(vector->c[i])
#define S(i)       (float)(vector->s[i])
#define Id(i)      (int)(id[i])

{
   struct VECTOR *vector;
   float         A, B, R, H, EJ;
   double        Dmax, YR, z;
   int           i, id[4], d[4], J, k, n1, n2, n3;

/* memory allocation  */

   vector=(struct VECTOR *)calloc(1,sizeof(struct VECTOR))

   vector->c=fl_mem_alloc(N);
   vector->s=fl_mem_alloc(N);

/* calculate sin and cos functions */

   sin_cos(vector, N);

/* first estimate of diameter */

   n1=(N-1)/4;
   n2=(N-1)/2;
   n3=3*((N-1)/4);

   Id(0)=0;
   Id(1)=n1;
   Id(2)=n2;
   Id(3)=n3;

   A=(r(0)-r(n2))/2;
   B=(r(n1)-r(n3))/2;
   R=(r(0)+r(n1)+r(n2)+r(n3))/4;
   H=(r(0)+r(n2)-r(n1)-r(n3))/4;

/* find largest non enclosed point */

   J=-1;
```

zone_exchange.c

```
   Dmax= fabs((double)(H)) + 1.e-5;
   for(i=0;i<N;i++) {
       YR=fabs((double)(r(i)-(R+A*C(i)+B*S(i))));
       if(Dmax < YR) {
          Dmax=YR;
          J=i;
       }
   }
   if(J == -1) {
      *AA=A;
      *BB=B;
      *RR=R;
      *HH=2*(float)(fabs((double)(H)));
      cfree(vector);
      cfree(vector->c);
      cfree(vector->s);
      return;
   }

/* start exchange procedure  */

   while(J!=-1) {
      EJ=r(J)-(R+A*C(J)+B*S(J));
      k=0;
      for(i=0;i<4;i++) {
          if(J < Id(i)) {
             d[k]=i;
             k++;
          }
      }

      if(k == 0) k=4;
      else       k=d[0];

      z=1;
      if((copysign(z,(double)(H))*pow(-1*z,(double)(k))) !=
          copysign(z,(double)(EJ)))
         k=k-1;

      if(k < 0) {
         Id(3)=Id(2);
         Id(2)=Id(1);
         Id(1)=Id(0);
         Id(0)=J;
      }
      else {
         if(k < 4)
            Id(k)=J;
         else {
            Id(0)=Id(1);
            Id(1)=Id(2);
            Id(2)=Id(3);
            Id(3)=J;
         }
      }

/* find new reference */

      A=((r(Id(0))-r(Id(2)))*(S(Id(1))-S(Id(3)))-(r(Id(1))-r(Id(3)))*
         (S(Id(0))-S(Id(2))))/((C(Id(0))-C(Id(2)))*(S(Id(1))-S(Id(3)))-
         (C(Id(1))-C(Id(3)))*(S(Id(0))-S(Id(2))));

      if(fabs((double)(S(Id(0))-S(Id(2)))) <
                                    fabs((double)(S(Id(1))-S(Id(3)))))
         B=((r(Id(1))-r(Id(3)))-A*(C(Id(1))-C(Id(3))))/(S(Id(1))-S(Id(3)));
      else
         B=((r(Id(0))-r(Id(2)))-A*(C(Id(0))-C(Id(2))))/(S(Id(0))-S(Id(2)));

      R=(r(Id(0))+r(Id(1))-A*(C(Id(0))+C(Id(1)))-B*(S(Id(0))+S(Id(1))))/2;

      H=r(Id(0))-(R+A*C(Id(0))+B*S(Id(0)));

 /* find largest non enclosed point */

      J=-1;
      Dmax= fabs((double)(H)) + 1.e-5;

      for(i=0;i<N;i++) {
         YR=fabs((double)(r(i)-(R+A*C(i)+B*S(i))));
         if(Dmax < YR) {
            Dmax=YR;
            J=i;
         }
      }
   }

   *AA=A;
   *BB=B;
   *RR=R;
   *HH=2*(float)(fabs((double)(H)));

   cfree(vector);
   cfree(vector->c);
   cfree(vector->s);
   return;
}

#undef N
#undef r(i)
#undef C(i)
```

```
#undef S(i)
#undef id(i)
```

lstsq_circle.c

```
/* This function implements the algorithm proposed by Forbes [1989]
   and described in section 2.5.3 */

lstsq_circle(data,xo,yo,r)
   struct CARTDATA  *data;
   float            *xo,*yo,*r;

#define N       (int)(data->nb_elt)
#define A(i,j)    (float)(a[3*i+j])
#define J(i,j)    (float)(mj[3*i+j])

{
    float    *a, *b, *mj, *d, *ri;
    float    pxo, pyo, pr, dpxo, ppxo;
    int      i;

    a=fl_mem_alloc(3*N);
    mj=fl_mem_alloc(3*N);
    b=fl_mem_alloc(N);
    d=fl_mem_alloc(N);
    ri=fl_mem_alloc(N);

/*  get initial solution */

    for(i=0;i<N;i++) {       /* set matrix A and vector b */
       A(i,0)= 2*data->x[i];
       A(i,1)= 2*data->y[i];
       A(i,2)= -1;
       b[i]= ((data->x[i]*data->x[i]) + (data->y[i]*data->y[i]));
    }

/* solve linear least squares system by Cholesky */

    linear_lstsq(a, b, xo,yo,r,N);

    *r=(float)(sqrt((double)((*xo)*(*xo)+(*yo)*(*yo)-(*r))));

/* set matrix J and vector d  */

    for(i=0;i<N;i++) {
       ri[i]= sqrt((data->x[i]- *xo)*(data->x[i]- *xo) +
              (data->y[i]- *yo)*(data->y[i]- *yo));
       d[i]= -1*(ri[i] - *r);
       J(i,0)= -1*(data->x[i]- *xo)/ri[i];
       J(i,1)= -1*(data->y[i]- *yo)/ri[i];
       J(i,2)= -1;
    }

    dpxo=1;
    ppxo=0;
```

lstsq_circle.c

```
     while(dpxo > 1.e-4) {

     linear_lstsq(mj, d, &pxo,&pyo,&pr,N);

     dpxo=(float)(fabs((double)(pxo-ppxo)));

     ppxo=pxo;

     *xo= *xo + pxo;
     *yo= *yo + pyo;
     *r = *r  + pr;
     }
     cfree(a);
     cfree(mj);
     cfree(b);
     cfree(d);
     cfree(ri);
     return;
}


#undef A(i,j)
#undef J(i,j)
#undef N
```

lstsq_plane.c

```
/* This function implements the algorithm proposed by Forbes [1989],
 for least squares plane fitting                                    */

void plane_least_square(data,pa,pb,pc)

     struct DATA *data;
     double *pa, *pb, *pc;

{
  float **a, *w, **v, sing_val, sing_vect, div;
  int   i,j, idx;

/* memory allocation  */


  a=flp_mem_alloc(data->nb_elt+1);
  v=flp_mem_alloc(4);

  for(i=0;i<=data->nb_elt;i++)
     a[i]=fl_mem_alloc(4);

  for(i=0;i<=3;i++)
     v[i]=fl_mem_alloc(4);

  w=fl_mem_alloc(4);

/* calculate the centroid of the points   */

  centroid(data);

/* shift the origin of the coordinate system to the centroid */

  for(i=1;i<=data->nb_elt;i++)  {
    a[i][1]=data->x[i-1] - data->xcentr;
    a[i][2]=data->y[i-1] - data->ycentr;
    a[i][3]=data->z[i-1]  -data->zcentr;
  }

/* calculate the singular value decomposition of a[i][k]  */

  svdcmp(a,data->nb_elt,3,w,v);

/* choose the singular vector corresponding to the smallest singular
   value, sing_val                                                    */

  sing_val=1.0e+21;
  for(i=1;i<4;i++) {
    if(w[i] < sing_val) {
      sing_val=w[i];
      idx=i;
```

lstsq_plane.c

```
    }
  }

  *pa=(double)(v[1][idx]);
  *pb=(double)(v[2][idx]);
  *pc=(double)(v[3][idx]);


  cfree(a);
  cfree(v);
  for(i=0;i<=data->nb_elt;i++)
     cfree(a[i]);
  for(i=0;i<=3;i++)
     cfree(v[i]);
  cfree(w);
  return;
}
```

References

Abadie, J. and Carpentier, J., 1969. Generalization of the Wolfe reduced gradient method to the case of non-linear constraints. In *Optimization*, R. Fletcher (ed.), pp. 37-47, Academic Press, New York.

Abadie, J., 1972. Application of GRG algorithm to optimal control problems. In *Nonlinear and Integer Programming*, J. Abadie (ed.), pp. 191-211, North-Holland Pub. Co., Amsterdam.

Abadie, J., 1978. The GRG method for non-linear programming. In *Implementation of Optimization Software*, H. J. Greenberg (ed.), pp. 335-363, Sijthoff and Noordhoff.

Anthony, G. T. and Cox, M. G., 1985. Reliable algorithms for roundness assessment according to BS3730. In: *Proceedings Conference on Software for Coordinate Measuring Machines*, Teddington, UK: National Physical Laboratory, pp. 30-37.

Anthony, G. T.; Anthony, H. M.; Cox, M. G. and Forbes, A. B., 1991. *The parametrization of fundamental geometric form.* Technical Report EUR 13517EN, Comission of the European Communities (BCR Information), Luxembourg.

Anthony, G. T. and Cox, M. G., 1984. *The design and validation of software for dimensional metrology.* Teddington, UK: National Physical Laboratory, NPL report DITC 50/84.

Ayre, H. G. and Stephens, R., 1956. *A First Course in Analytic Geometry.* D. Van Nostrand Company: New Jersey. USA.

Baker, J. E., 1987. Reducing bias and inefficiency in the selection algorithm. *Genetic Algorithms and their Applications: Proc.* 2nd *Int. Conf.*, J. J. Grefenstette (ed.), pp. 14-21, Cambrige, MA.

Bonnesen T., 1924. Über das isoperimetrische Defizit ebener Figuren. *Math. Ann.*, vol. 91, pp. 252-268.

BSI, 1987a. *Assessment of departures from roundness*, Part 1. London: British Standard Institution. BS 3730:1987.

BSI, 1987b. *Determining departures from roundness by measuring variations in radius.* London: British Standard Institution. BS 6740:1987.

BSI, 1989. *British standard guide to assessment of position, size and departure from nominal form of geometrical features.* London: British Standard Institution. BS 7172: 1989.

BSI, 1990. *Engineering drawing practice. Part 3: Recommendations for geometric tolerancing.* London: British Standard Institution. BS 308:Part3:1990.

Brindle, A., 1981. *Genetic algorithms for function optimization.* PhD thesis, University of Alberta, Edmonton, USA.

Brown, C. M., 1982. PADL-2: A technical summary. *IEEE Comput. Graph. Appl.*, vol. 2, no. 2, pp. 69-84.

Burdekin, M.; Di Giacomo, B. and Xijings, Z., 1985. Calibration software and application to Coordinate Measuring Machines. In: *Proceedings Conference on Software for Coordinate Measuring Machines*, Teddington, UK: National Physical Laboratory, pp. 1-7.

Carpinetti, L. C. R. and Chetwynd, D. G., 1992. A new strategy for inspecting roundness tolerances. *ASPE* 7th *Annual Meeting*, Florida, October 18-23, pp.

281-284.

Carpinetti, L. C. R. and Chetwynd, D. G., 1993. Genetic search methods for inspecting geometric tolerances. *ASPE* 8^{th} *Annual Meeting*, Seattle, Washington, November 7-12.

Chetwynd, D. G., 1979a. Roundness measurement using limaçons. *Precision Engineering*, vol. 1, pp. 137-141.

Chetwynd, D. G., 1979b. Dimensioning nominal circles: the resolution of conflicting ideas. *Acta IMEKO*, pp.683-690.

Chetwynd, D. G., 1980. *A unified approach to the measurement analysis of nominally circular and cylindrical surfaces.* PhD thesis, University of Leicester.

Chetwynd, D. G. and Phillipson, P. H., 1980. An investigation of reference criteria used in roundness measurement. *J. Phys. Sci. Instrum.*, vol. 13, pp. 530-538.

Chetwynd, D. G. and Siddal, G. J., 1976. Improving the accuracy of roundness measurement. *J. Phys. Sci. Instrum.*, vol. 9, pp. 537-544.

Chetwynd, D. G., 1985. Applications of linear programming to engineering metrology. *Proc. Instn. Mech. Engrs.*, vol. 199, no. B2, pp. 93-100.

Colville, A. R., 1968. *A comparative study on non-linear programming codes.* IBM New York Scientific Center, report no. 320-2949.

Cox, M. G. and Jackson, K., 1983. *Algorithms and software for metrology.* Teddington, UK: National Physical Laboratory, NPL report MOM 65.

Damir, M. N. H., 1979. Approximate harmonic models for roundness profiles.

Wear, vol.57, pp. 217-225.

DeJong, K. A., 1975. *Analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, Univ. of Michigan, Ann Harbor, MI.

Dennis, J. E. and Schnable, R. B., 1983. *Numerical Methods for Unconstrained Optimization*, Prentice-Hall, Englewood Cliffs, NJ.

Dhanish, P. B. and Shunmugam, M. S., 1991. An algorithm for form error evaluation - using the theory of discrete and linear Chebyshev approximation. *Computer methods in applied mechanics and engineering*, vol. 92, pp. 309 - 324.

Duffie, N. A. ; Bollinger, J.; Piper, R. and Kroneberg, M., 1984. Error compensation of CAD/CAM data bases by comparison with measured data bases for increased part accuracy capability. *NAMRC-XII*, Houghton, MI, May, pp. 424-430.

Elgabry, A. K., 1986. A framework for a solid-based tolerance analysis. *ASME Int. Computers in Engineering Conf.*, Chicago, USA.

Faux, I. D. and Pratt, M. J., 1979. *Computational geometry for design and manufacture.* Chichester, Ellis Horwood.

Fiacco, A. V. and McCormick, G. P., 1968. *Non-linear Programming: Sequential Unconstrained Minimization Techniques.* John Wiley and Sons, New York and Toronto.

Fletcher, R., 1970a. *The Calculation of Feasible Points for Linearly Constrained Optimization Problems.* Harwell, UK: Atomic Energy Research Establishment Report AERE-R 6354.

Fletcher, R., 1970b. A new approach to variable metric algorithms. *Comptr. J.*, vol. 13, pp. 317-322.

Fletcher, R., 1974. Methods related to Lagrangian functions. In *Numerical Methods for Constrained Optimisation*, P. E. Gill and W. Murray (eds.), pp. 219-240, Academic Press, London and New York.

Fletcher, R., 1977. Methods for solving nonlinearly constrained optimisation problems. In *The State of the Art in Numerical Analysis*, D. Jacobs (ed.), pp. 365-448, Academic Press, London and New York.

Fletcher, R., 1987. *Practical Methods of Optimization.* Second edition. John Wiley & Sons: Chichester, UK.

Forbes, A. B., 1989. *Least square best fit geometric elements.* Teddington: National Physical Laboratory. NPL Report DITC 140/89.

Forbes, A. B., 1992. *Geometric tolerance assessment.* Teddington: National Physical Laboratory. NPL Report DITC 210/92.

Forsythe, G. E. ; Malcolm, M. A. and Moler, C. B., 1977. *Computer methods for mathematical computation.* Prentice Hall.

Gill, P. E.; Murray, W. and Wright, M. H., 1981. *Practical Optimization.* Academic Press: London.

Gill, P. E.; Murray, W. and Wright, M. H., 1991. *Numerical Linear Algebra and Optimization*, vol. 1. Addison-Wesley Publishing Company: California, USA.

Goldberg, D. E., 1987. Computer-aided gas pipeline operation using genetic algorithms and rule learning. Part 1: Genetic algorithms in pipeline optimization. *Engineering with Computers*, vol. 3, pp. 35-45.

Goldberg, D. E., 1989. *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA: Addison-Wesley Publishing Company.

Goldfarb, D., 1969. Extension of Davidon's variable metric method to maximization under linear inequality and equality constraints. *SIAM J. Appl. Math.*, vol. 17, no. 4. pp. 739-764.

Goto, M. and Iizuka, K., 1977. An analysis of the relationship between minimum zone deviation and the least squares deviation in circularity and cylindricity. *Proceedings of the International Conference on Production Engineering*, New Delhi, India.

Grefenstette, J. J., 1990. A user's guide to genesis. version 5.0: public domain software.

Hadley, G., 1962. *Linear Programming.* Reading, Massachusetts: Addison-Wesley Pub. Company.

Hadley, G., 1964. *Nonlinear and Dynamic Programming.* Reading, Massachusetts: Addison-Wesley Pub. Company.

Hajela, P., 1990. Genetic search - an approach to the nonconvex optimization problem. *AIAA journal*, vol. 28, no. 7, pp. 1205-1210.

Hearn, D. W. and Vijay, Y., 1982. Efficient algorithms for the (weighted) minimum circle problem. *Oper. Res.*, vol. 30, pp. 777-795.

Hestenes, M. R., 1969. Multiplier and gradient methods. *J. Opt. Th. Applics.*, vol. 4, pp. 303-320.

Hillyard, R. C. and Braid, I. C., 1978. Characterizing non-ideal shapes in terms of dimensions and tolerances. *Comput. Graph.*, vol. 12, no. 3, pp. 234-238.

Holland, J. H., 1975. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Harbor, MI.

Huang, S. T.; Fan, K. C. and Wu, J. H., 1993. A new minimum zone method for evaluating flatness errors. *Precision Engineering*, vol. 15, no. 1, pp. 25-32.

Jacobi, H. D. and Lenz, K. G., 1980. Automatic continuous scanning with Multi-Coordinate Measuring Machines. *Proceedings* 5^{th} *International Conference on Automated Inspection and Product Control*, pp. 465-472.

Jayraman, R. and Srinivasan, V., 1989. Geometric tolerancing: I. Virtual boundary requirements. *IBM J. Res. Develop.*, vol. 33, no. 2, pp. 90-104.

Johnson, R. H., 1985. *Dimensioning and tolerancing-Final report.* Report R-84-GM-02.2. Computer Aided Manufacturing International.

Kaiser, M. J. and Morin, T. L., 1992. A remark on a characterization of out of roundness measures. *Meas. Sci. Technol.*, vol. 3, pp. 341-342.

Kakino, Y. and Kitazawa, J., 1978. In situ measurement of cylindricity. *Ann. CIRP*, vol. 27, no.1, pp. 371-375.

Kuhn, H. W. and Tucker, A. W., 1951. Non-linear programming. *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pp.481-492, Berkeley, University of California Press.

Lai, K. and Wang, J., 1988. A computational geometry approach to geometric tolerancing. 16th *North American Manufacturing Research Conference*. University of Illinois. pp. 376-379.

Lasdon, L. S. ; Fox, R. and Ratner, M., 1973. Non-linear optimization using the generalized reduced gradient method. Tech. Memo. 325, Dept. of Oper. Res., Case Western Reserve Univ., Cleveland, Ohio.

Lasdon, L. S. ; Fox, R. and Ratner, M., 1973a. An efficient one-dimensional search procedure for barrier functions. *Math. Programming*, vol. 4, pp. 275-296.

Lasdon, L. S. ; Waren, A. D.; Jain, A. and Ratner, M., 1978. Design and testing of a generalized reduced gradient code for non-linear programming. *ACM Transactions on Mathematical Software*, vol. 4, pp. 34-50.

Le, V-B. and Lee, D. T., 1991. Out of roundness problem revisited. *IEEE Trans. Pattern Anal. Machine Intel.*, vol.13, pp. 217-223.

Lee, D. T., 1982. Medial axis transformation of a planar shape. *IEEE trans. Pattern Anal. Machine Intell.*, vol. PAMI-4, no.4, pp. 363-369.

Light, R. A. and Gossard, D. C., 1982. Modification of geometric models through variational geometry. *Computer Aided Design*, vol. 14, no. 2, pp. 209-214.

Lin, V. C.; Gossard, D. C. and Light, R. A., 1981. Variational geometry in computer-aided design. *Comput. Graph.*, vol. 15, no. 3.

Ma, Q. H.; Wu, S. M. and Hu, G. L., 1988. Lightining and accuracy in machine vision inspection. *SPIE Automotive Displays and Industrial Illunination*, vol. 958, pp. 181-187.

Mullins, P., 1987. European quality in production. *Production*, vol. 99, no.

12, pp. 50-53.

Murray, W., 1976. Constrained optimization. In *Optimization in Action*, L. C. W. Dixon (ed.), pp. 217-251, Academic Press, London and New York.

Murray, W. and Wright, M. H., 1980. *Computation of the search direction in constrained optimization algorithms.* Report SOL 78-23, Department of Operations Research, Stanford University.

Murthy, T. S. R. and Abdin, S. Z., 1980. Minimum zone evaluation of surfaces. *Int. J. Mach. Tool Des. Res.*, vol. 20, pp. 123-136.

Murthy, T. S. R., 1982. A comparison of different algorithms for cylindricity evaluation. *Int. J. Mach. Tool Des. Res.*, vol. 22, no. 4, pp. 283-292.

NAG, 1990, *NAG fortran library.* Oxford, UK: The Numerical Algorithms Group Ltd.

Nelder, J. A. and Mead, R., 1965. A simplex method for function minimization. *Comput. J.*, vol. 7, pp. 308.

Odayappan, O. Raja, J. Hocken, R. J. and Chen, K., 1992. Sampling strategies for circles in coordinate measuring machines. *ASPE* 7th *Annual Meeting, Florida, October 18-23*, pp. 149-152.

Osborne, M. R. and Watson, G. A., 1968. On the best linear Chebyshev approximation. *Comput. J.*, vol.10, pp. 172-177.

Osborne, M. R. and Watson, G. A., 1969. An algorithm for minimax approximation in the nonlinear case. *Computer Journal*, vol. 12, pp. 63-69.

Pastorius, W. J., 1989. Vision measures up in automated assembly. *Manufacturing Engineering*, april, pp. 81-83.

Preparata, F. P. and Hong, S. J., 1977. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM*, vol. 20, no. 2, pp. 87-93.

Preparata, F. P. and Shamos, I. M., 1985. *Computational Geometry: an Introduction.* Springer-Verlag. New York.

Reason, R. E., 1966. *Report on the measurement of roundness.* Leicester: Rank Taylor Hobson.

Requicha, A. A. G. and Volker, H. B. 1982. Solid modeling: A historical summary and contemporary assessment. *IEEE Comput. Graphics Appl.* 2(**2**), pp. 9-24.

Requicha, A. A. G., 1983. Toward a theory of geometric tolerancing. *Int. J. Robotics and Research*, vol. 2, no. 4, pp. 45-60.

Requicha, A. A. G., 1984. Representation of tolerances in solid modeling: issues and alternative approaches. In: *Solid modeling by computers: from theory to applications*, J. W. Boyse and M. S. Pickett, Eds. New York: Plenum, pp. 3-22.

Requicha, A. A. G. and Chan, S. C., 1986. Representation of geometric features, tolerances, and attributes in solid modelers based on constructive geometry. *IEEE J. Rob. Autom.*, vol. RA-2, no. 3, pp. 156-166.

Rivlin, T. J., 1979. Approximation by circles. *Computing*, vol. 21, pp. 93-104.

Rosen, J. B., 1961. The gradient projection method for non-linear programming, Part II - non-linear constraints. *SIAM J. Appl. Math.*, vol. 9, pp. 514-532.

Rossignac, J. R. and Requicha, A. A. G. 1986. Offsetting operations in solid modeling. *Comput. Aided Geom. Des.* vol. 3, pp. 129-148

Roy, U. and Liu, C. R., 1988. Feature based representational scheme of a solid modeler for providing dimensioning and tolerancing information. *Robotics and Computer Integrated Manufacturing*, vol. 4, no. 3/4.

Roy, U.; Liu, C. R. and Woo, T. C., 1991. Review of dimensioning and tolerancing: representation and processing. *Computer Aided Design*, vol. 23, no. 7, pp. 466-483.

Sargent, R. W. H. and Murtagh, B. A., 1973. Projection methods for non-linear programming. *Math. Prog.*, vol. 4, pp. 245-268.

Schneeberger, J.; Bollinger, J.; Duffie, N. and Yamazaki, K., 1983. Computerized three-dimensional error analysis for automated part inspection. *NAMRC-XI*, Madison, Wi, May, 24-26. pp. 423-427.

Sediscad, 1993. *Perceval+ inspection and CAD/CAM.* FR: Sediscad and Groupe Renault Automation.

Shamos, M. I. and Hoey, D., 1975. Closest-point problems. *Proc.* 16th *IEEE Symp. Foundations of Comput. Sci.*, pp. 151-162.

Shanno, D. F.; Berg, A. and Cheston, G., 1974. Restarts and rotations of quasi-Newton methods. *Information Processing 74*, North-Holland Pub. Co., Amsterdam, pp. 557-561.

Shunmugam, M. S., 1986. On assessment of geometric errors. *Int. J. Prod. Res.*, vol. 24, no. 2, pp. 413-425.

Sinnott, E. W.; Dunn, L. C. and Dobhansky, T., 1950. *Principles of genetics.* McGraw Hill, NY.

The MathWorks, 1992. *Pro-Matlab for SUN workstations*, version 4.0a-SUN. Sherborn, MA, USA: The MathWorks, Inc.

The Scientific Press, 1992. GINO: General Interactive Optimizer, version 1.0. San Francisco, CA, USA: The Scientific Press.

Traband, M. T.; Joshi, S.; Wysk, R. A. and Cavalier T. M., 1989. Evaluation of straightness and flatness tolerances using the minimum zone. *Manufacturing Review*, vol. 2, no. 3, pp. 189-195.

Treywin, E. T. and Edwards, D. B., 1987. Automatic inspection and control for quality. *Proceedings* 8th *International Conference on Automated Inspection and Product Control*, pp. 121-144.

Tsukada, T.; Anno, Y,; Yanagi, K. and Suzuki, M., 1977. An evaluation of form errors of cylindrical machine parts by spiral tracing method. *Proc.* 18th *Mach. Tool Des. Res.*

Turner, J. U. and Wozny, M. J., 1987. Tolerances in computer-aided geometric design. *Visual computer*, vol. 3, no. 4, pp. 214-226.

Turner, J. U. ; Wozny, M. J. and Hoh, D. D., 1987. Tolerance analysis in a solid modeling environment. *Proc. ASME Computers in Engineering Conf.*, New York.

Turner, J. U. and Wozny, M. J., 1988. A framework for tolerances utilizing solid models. 3rd *Int. Conf. Computer-Aided Production Engineering*, University of Michigan, Ann Arbor, MI.

Turner, J. U., 1990. Relative positioning of parts in assemblies using mathematical programming. *Computer-aided design*, vol. 22, no. 7, pp. 394-400.

Ventura, J. A.; Chang, C. A. and Klein, C. M., 1988. Automated inspection of circular parts. *Proceedings of the* 10th *Annual Conference on Computers and Industrial Engineering*, vol. 15, pp. 349-354.

Wagner, H. M., 1975. *Principles of Operations Research.* Prentice Hall.

Whitehouse, D. J., 1973. A best fit reference for use in partial arcs. *J. Phys. Sci. Instrum.*, vol. 6, pp. 921-924.

Wilson, R. B., 1963. *A simplicial algorithm for concave programming.* PhD thesis, Harvard University.

Wirtz, A., 1992. Vectorial tolerancing for production quality control and functional analysis in design. ISO/TC 10/SC 5/WG 1, document no. 26 E.