OTG REPORT No. 1/09

UNDERWATER ACOUSTIC IMAGING: EXACT GEOMETRICAL-ACOUSTICS TREATMENT OF THE IMAGE DUE TO A SPECULAR REFLECTOR[1]

David G. BLAIR

Ocean Technology Group,[2] F09
The University of Sydney
otg@otg.usyd.edu.au

September 2009

ISSN 1442 8075

[1] *Address correspondence to:* D.G. Blair (D.Blair@civil.usyd.edu.au) or D.G. Blair, Ocean Technology Group, F09, The University of Sydney, NSW 2006, Australia.
[2] The Ocean Technology Group forms part of the School of Civil Engineering.

Abstract

In underwater acoustic imaging, used to produce high-resolution images in turbid waters, a specular reflector in general produces a 'pseudoimage' of the receiving array, located on the reflecting surface; the pseudoimage is of considerable use since it reveals the shape of the surface. A system is considered in which a spherical transmitter together with a 2D receiving array give a 3D image in a single 'ping.' A treatment predicting the shape of the pseudoimage—in particular, its lateral extent—is given that is exact within geometrical acoustics. The surfaces to which the treatment is applied are the paraboloid (with two principal radii of curvature)—provided that the transmitter lies on the paraboloid's axis—the sphere, the cylinder and the plane. The treatment involves a ray-tracing algorithm based on the equation of the surface, and an algorithm to invert that procedure using the Levenberg-Marquardt method. Pseudoimages of lines in the array are graphed and discussed, along with, more interestingly, pseudoimages of squares. While the latter pseudoimages are parallelograms when the square is small, in general they are not parallelograms, since all four sides are curved. Further features found are that an 'object' in the array may produce multiple pseudoimages, no pseudoimage, 'local optima' and/or 'blockage points.' Such an exact determination of the resulting pseudoimage for selected surfaces gives useful insight into pseudoimages that occur in practice. Conditions of validity (arising because wave effects are neglected) are given. The report also contains a preliminary discussion of the extension that would be needed to include wave effects. In addition it is shown that, subject to more restrictive conditions, the results apply also to a general smooth surface. A 'paraxial' approximation (similar to the 'large-range approximation' of an earlier paper but somewhat more general) is described and found to be useful.

Contents

1. Introduction

1.1 Underwater Acoustic Imaging

To produce high-quality images of objects in water, most often optical means such as a video-camera are used. However, in sediment-laden waters such methods fail. Underwater acoustic imaging (UAI) offers a way around this problem. As outlined by Jones (1996), the Maritime Operations Division of the (Australian) Defence Science and Technology Organisation (DSTO) initiated an innovation program to implement UAI. In this endeavour, DSTO was joined by Thales Underwater Systems (TUS) and the CSIRO (Commonwealth Scientific and Industrial Research Organisation). As a result, an operational prototype was produced in 2004.

Accounts of the UAI project have been given by Maguer *et al.* (2000), Jones (2000) and Vesetas and Manzie (2001). Particular aspects of the program have been discussed as follows: a simulation of the image-forming process (Blair and Anstee, 2000), near-field beam patterns (Blair, 2002), rapid signal processing (Blair and Jones, 1998; Blair, 1997) and one-bit digitisation (Blair *et al.,* 2006). In the present report the 'image-forming' process refers to the equation for adding together the signals received at the various sensors with appropriate time delays to produce a 3D image.

The UAI system has a lateral resolution of the order of a few mm per m of range, and a range resolution of the order of a few mm. Images are obtainable for ranges from 0.5 m to beyond 2 m. The dimensions of the two-dimensional (2D) receiving array are a few hundred millimeters. The elements are distributed at random and the array is very sparsely populated. The system has a spherical transmitter and the operating frequency is a few megahertz. A long chirp pulse is used to achieve good range resolution; to this end a cross-correlation or 'dechirping' process is applied to the received signals (Urick, 1983; Rihaczek, 1985). A 3D image is produced in one 'ping' from the transmitter. Images are produced in real time, provided that the 'partly coherent' mode (defined below) is used; thus a moving image is seen. To achieve the high sampling rate needed, the system uses one-bit sampling preceded by the intentional adding of noise.

Other groups have developed somewhat different implementations of the underwater acoustic imaging concept as follows: E. Belcher (Belcher *et al.*, 2002; Belcher *et al.*, 2001; Belcher *et al.*, 1999), J. Impagliazzo (Chiang *et al.*, 2003; Broadstone *et al.*, 1999; Chiang *et al.*, 2002) and R.K. Hansen (Hansen and Andersen, 1996; Hansen and Andersen, 1998; Hansen, 1993), together with co-workers in each case.

1.2 Specular Reflectors

A specular reflector is a surface that reflects waves as a mirror does in optics. In the acoustics case, a specularly reflected wave is produced whenever a wave is incident on a smooth boundary that is an infinitely thin boundary separating two materials of different acoustic impedance. Considerable work on the acoustic imaging of specular reflectors has been reported in the literature, as will now be discussed. Much of this work is concerned with medical imaging, as follows. It is known that imaging via an array or a focussed aperture, by methods appropriate to point scatterers and extended diffuse scatterers, also produces an

image—a correctly located image—of specular surfaces (called in this report pseudoimages, because of their special relationship to the receiving array). However, such an image is restricted to the portion of the specular surface that is nearly perpendicular to the general direction of propagation[3] (Shapiro *et al*., 2001). The image lies[4] on the specular surface; the axial (or range) resolution is one-half of the spatial pulse length, just as for a point scatterer (Anderson-Dutoit, 2002, p. 7). The systems discussed in the present report often involve a long transmitted pulse followed by the appropriate cross-correlation process: then the 'pulse length' is to be replaced by the length of the autocorrelated pulse, of order $c/2B$, where B is the bandwidth and c is the speed of sound.

As stated by Goldstein and Powis (1999, p. 60), specular reflectors are of relatively little use in medical diagnosis: their use tends to be limited to indicating organ size. Some attention has gone into reducing the contribution to the image from specular reflectors, so as to better detect other features (Wilhjelm *et al*., 2004). It is to be noted that in medical imaging, the surfaces called 'specular' tend to produce both a specularly reflected component and a diffusely reflected component (Goldstein and Powis, 1999, p. 61).

We note that sonar imaging in typical situations differs from medical imaging in at least two respects. First, artificially constructed objects (such as sea mines) are often targeted for viewing, in which case the mapping of specular reflectors can be of considerable interest. Second, it is expected that, with artificial targets, very often the diffuse component is quite small.

As a general point in the context of specular reflectors in ultrasound or acoustic systems, other methods have been devised to complement the image-forming equation. In particular, methods have been reported for the classification of reflections into a few categories, of which one is specular reflection; examples from sonar are given below.

We now turn to sonar imaging. In sonar (as in medical imaging), the image-forming equation and similar techniques not only locate point scatterers and diffusely reflecting surfaces, but also give rise to highlights in the image, of which some are due to specular reflectors but others are due to targets of other kinds. Dealing with highlights has thrown up challenges, which authors have addressed to a considerable degree.

Ray tracing of reflections from specular surfaces have been carried out: for example, Bouxsein *et al.* (2006) dealt with a plane, a sphere and a cylinder. Kuc and Viard (1991) employed Huygens' Principle and linear systems theory to study reflection from specular surfaces. Surfaces that are rough still produce some phase coherence effects in the reflected beam. Such surfaces have been treated using the Kirchhoff approximation by, for example, Culver and McDaniel (1991) and Bosma and Kuc (1994); surfaces of this type were also treated by Williams and Funk (1994) (see also references therein).

Work has been done (in sonar) on the classification of highlights and related features. For example, a method reported by Kleeman and Kuc (1995) classifies targets into planes, corners, edges and unknowns, while Nai-Chyuan Yen and Dragonette (1997) used waveform analysis to classify returns into categories including specular reflectors, creeping waves, chalice waves, Bragg waves and Bloch waves. Hansen (2008) reported a method for analysing acoustic data

[3] This refers to the case where the transmitter and the receiving array are placed close together, so that the 'look direction' is approximately the same for both.
[4] This refers to the position in 'image space.'

to produce a mathematical representation of a large or complete surface, initially made up of triangles but optionally later fitted by a smooth mathematical surface; surface roughness values are also estimated.

Ferguson and Wyber (2005) used techniques of acoustic reflection tomography to image underwater objects that included specular reflectors. The waves insonifying the targets were made approximately plane by keeping a large separation between the transducer and the target.

From the discussion in this Section 1.2 (especially the first paragraph), we may note two ways in which the forming of an image of a specular reflector differs from that of a collection of point scatterers placed randomly. (It is expected that a very rough surface may be regarded as a case of the latter.)[5] First, to trace out a complete specular surface, usually multiple pings from different directions are needed. Second, experience has shown that, for those points on the specular surface that do reflect to the receiving array, the theory of image-forming, developed for point scatterers, continues to locate the points in the image correctly; however, it is not necessarily the case that this theory (including the point spread function) applies *in toto* to specular reflectors. We return to this point in Section 12.

1.3 Context and Goals of the Present Work

In the 1999 'Pyrmont 2' trial of the UAI system, Thales Underwater Systems observed that a flat specular reflector can produce what looks like an image of the receiving array (Manzie, 2000). Such an 'image' has been dubbed a '*pseudoimage*' (Blair, 2006). Manzie noted that such an image can alert the observer to the fact that a specular reflector is present. A pseudoimage is very easy to recognise if the receiving array is operated in the 'partly coherent' mode (defined in Section 2 below). By way of clarification, we point out that, because the pseudoimage lies on the *surface* of the specular reflector, the pseudoimage is quite different from images of the usual kind, encountered, for example, in optics.

Pseudoimages are very useful in underwater imaging because they provide a way of mapping a specular (smooth) surface (usually with multiple pings, as pointed out below in Section 1.4). It was therefore judged important to develop a theoretical understanding of pseudoimages. To this end, two articles have been produced: one of them in 2006 (Blair, 2006, to be called I) and one of them being the present report. The approach adopted in both these articles has been to 'work one's way' outwards from the 'chief reflecting point' [defined in Section 3.1 as a point on the reflecting surface that (geometrically) reflects directly back to the (center of) the transmitter]. At that point the surface possesses two principal radii of curvature. Usually, near that point, the smooth surface is described to a good approximation by a second-order polynomial equation determined by these radii (and their orientation)—in fact the equation of a paraboloid. Then the existence of pseudoimages provides a way to use acoustic imaging to determine, not only the location of specular reflectors, but also their principal radii of curvature, because the magnification(s) of the pseudoimage depend on these radii.

In paper I, formulae were derived describing the pseudoimage for both flat and curved reflectors. Each curved reflector was taken to be a paraboloid, described by two (not necessarily equal) radii of curvature; however (as discussed in Section 11 below) the treatment applies, as an approximation, to other smooth surfaces having the same two radii of curvature

[5] Actually, for the very rough surface and the random point scatterers, some coherence is still present, leading to so-called 'speckle' (Goodman, 1976). In the present discussion speckle is ignored.

at the 'chief reflecting point.' The treatment in paper I was based on the 'geometrical approximation.' In that approximation (for reasons given in I and rounded out in Section 2) a given 'object point' (a point on the receiving array),[6] normally leads to a bright point, or pseudoimage point, on the surface of the reflector[7]; the position of the pseudoimage point is given by the law of reflection. The geometrical approximation should be good in the limit of short wavelengths.[8] In paper I (in the case of a non-planar reflector), the 'large-range approximation' was also assumed.

The present report extends the treatment to points that may lie far from the 'chief reflecting point.' It is recognised that, as one moves further away from that point, the radii of curvature in general change. In order to keep the study within bounds, a class of simple surfaces was selected for simulation. The present report is similar to I in that one of the surfaces treated is the paraboloid (with two radii) (The plane is included as a special case.). In a minor extension, the sphere and the cylinder are also treated. What all these surfaces have in common is that the equation of the surface, in Cartesian coordinates (x, y, z), is given by equating to zero a second-order polynomial function of x, y and z. (Not all such polynomials are treated, however.)

For a general treatment of the above surfaces, two major extensions (of the treatment in paper I) are necessary: first, to drop the 'large-range' assumption and, second, to replace the geometrical treatment by one that takes account of wave effects. In the present system, wave effects arise from two sources. First, acoustic waves are propagated and reflected: these are amenable to a treatment via Huygens wavelets undergoing spherical spreading (*e.g.* Clay and Medwin, 1977). Second, the image-forming (a numerical procedure defined above), since it is an approximation to back-propagation (Ljunggren *et al.*, 1980; Shewell and Wolf, 1968; Lalor, 1968), also introduces wave effects. We return to the second task (waves) at the end of the report; the present report carries out the first task: the dropping of the large-range approximation.

In paper I, the 'large-range approximation' was introduced as a means of simplifying the recovery of shape parameters for a non-planar reflecting surface. The large-range approximation basically requires that the displacement of the object point from the 'chief normal' [defined in Section 3.1 below as the normal to the reflecting surface that passes through the (center of) the transmitter] be small compared to the range. (Precise conditions on the approximation are given in I.) It is important to make the extension to larger displacements for two reasons. First, when the displacements are large, if the principal curvatures are inferred from the image data on the basis of the large-range approximation, the results can be in error by a factor as great as three, as will be found below (at Figure 9.2). Moreover, because the image of a square is no longer a parallelogram, in general no unique value for a given curvature would be obtained, even though the curvature has a definite value.

The second reason is that, at large displacements of the object point, further effects occur (as will be seen later). A single object point can lead to multiple pseudoimage points, or to no pseudoimage point. The object point can also lead to points in the image that share some, but

[6] As noted in a footnote to Section 2, strictly one must consider, not a mathematical point (object point), but a region of the plane of the array that is small, but not too small. The appropriate region contains a considerable number of sensor elements.

[7] That is, on the image of the surface of the reflector.

[8] Subject also to (i) the effective length of the pulse being small, and (ii) the smallness of grating lobe effects (Section 10.3).

not all, of the characteristics of a pseudoimage point, namely 'local optima' and 'blockage points.'

It is of interest to see how well the large-range approximation behaves in a typical situation. A rough measure of the goodness of the large-range approximation is the smallness of $\alpha \equiv (L/2)/r_0$, which is essentially the maximum angle (in radians) of inclination of rays to the 'chief normal' (defined in Section 3.1 below).[9] Here L is the diameter of the array and r_0 is the range. A typical system (such as that considered by DSTO) has $L = 0.5\,\mathrm{m}$ and $r_0 = 1.5\,\mathrm{m}$, giving $\alpha = \frac{1}{6}$. Hence, in this situation, while we expect the large-range approximation to give a good *first approximation*, it is *not accurate.* It must be emphasised that this is for a *typical* case: cases of considerably better behaviour and considerably worse behaviour both occur. It is worth mentioning that in the case considered, the accuracy is actually good (even at large displacements) if the reflector is a near-planar surface (see I, p. 11).

The purpose of the present work has been to develop an algorithm that describes exactly (but within the geometrical approximation) a set of surfaces specifiable by a small number of parameters, but having two distinct radii of curvature. The choice of the paraboloid (as above) achieves this. Thus, first, the present study of the paraboloid gives, exactly, an example of the effects that are produced when one goes beyond the plane to a curved surface. And second, it is believed that the paraboloid gives a good illustration of the *qualitative* features (see the 'other features' in Section 1.4) of the pseudoimage produced by a curved surface—at least in the case where the curvature does not change greatly over the extent of the surface.

Arguably the surfaces chosen for simulation are the ones most likely to occur in practice. However, two further limitations of the algorithm as currently coded need to be mentioned. First, the transmitter is required to lie on the axis of the paraboloid. (There is no corresponding limitation on the sphere or the cylinder.) Second, consider a situation of multiple 'pings' in which the transmitter-receiver system is moved rigidly (relative to the target) between pings. The present set of input parameters is not ideally suited to exploring the effects of such a rigid motion. The second limitation could be removed by a not-too-large amount of work. The removal of the first limitation requires an extension of the analytic calculation on which the ray-tracing algorithm is based; the extension appears to be fairly straightforward but somewhat lengthy.

1.4 Use of the Present Work

It is known that the pseudoimage lies on the reflecting surface, but usually occupies only a part (most often, a small part) of that surface. Thus (Stuart Anstee, private communication) an image of the entire specular surface can be built up from multiple pings taken with the sonar system located at different positions and orientations.[10] One is therefore led to ask why the present work is of use.

---

[9] This quantity α is to be distinguished from the quantity α used from Equation (3.1) onward.

[10] This statement assumes that either: (i) the movement of the sonar system from one ping to another is known, or (ii) registration—the fitting together of an image from one ping and that from another ping—can be carried out. The discussion of these two items lies outside the scope of the present work.

A brief answer is that, in the literature, there appears to have been little discussion of the lateral extent of the pseudoimage. This report helps to fill that gap; at the same time it relates the extent and boundaries of the pseudoimage to the geometry of the receiving array.

Thus first, the report is useful in showing, without vagueness, what some typical pseudoimages look like and how their shape may diverge from the simple shape (*e.g.* a parallelogram) predicted by the large-range approximation. Second, the report is useful in showing (with detailed examples) that a number of other features occur. First among these is that, for a given configuration (of the sonar system together with the target), there may be multiple pseudoimages or no pseudoimage. Second among the 'other features,' there may be a pseudoimage of part of the array only. On occasions this is due to the pseudoimage reaching the edge of the reflector. But on other occasions a more novel mechanism is at work, which produces a pair of part-pseudoimages abutting one another and separated by a curve of local optima. The report is useful also for a third reason: it reveals the presence of local optima and blockage points; their existence is of mild theoretical interest.

The above are the main uses of the report. Two lesser possible uses will be mentioned. First, an acquaintance with pseudoimages may help the sonar operator to steer the sonar system to new positions and orientations that enable the complete reflector surface to be mapped with efficiency. Second, it has been suggested (by Stuart Anstee, private communication) that a program that simulates a paraboloidal reflector might be useful in the design phase of a system, the aim of which is to map specular reflectors (the paraboloid being a good test case).[11]

1.4.1 Use of the 'Pseudoimage' Concept

The concept of a 'pseudoimage,' as developed in the short sequence of articles cited at the start of Section 1.3 (three articles including the present one), emphasises the relationship of the pseudoimage to the receiving array. The sequence as a whole has possible uses not mentioned so far in Section 1.4.

First among these, if the receiving array is operated in the partly coherent mode, the pseudoimage tends to consist of a (more or less) regular array of spots. This gives a strong visual cue to the fact that a specular reflector is present. Second, irrespective of whether the partly or the fully coherent mode is used, the pseudoimage gives rather directly the magnification(s) of the pseudoimage. From these, the principal radii of curvature can be deduced (quite quickly in the case where the large-range approximation is valid). There are thus *two* methods for calculating these radii, the other being a calculation based directly on the image (pseudoimage) of the surface in 3D space (without paying attention to the magnification). The 'magnification' method may have an advantage, in that the calculation of the radii is then based on the values of first spatial derivatives rather than second ones.

Third, consider the reflection from a *general* (non-paraboloidal) smooth surface in the neighbourhood of a reflecting point that reflects directly back to the transmitter. For a sufficiently small neighbourhood, a paraboloid provides a good approximation to the surface; then the algorithm of the present report gives the pseudoimage and its magnification accurately. This point is discussed in more detail in Section 11 below.

[11] In particular, a combined hardware-plus-software system to handle registration might be tested before the hardware is built. The present computer program would first need to be modified by, for example, allowing the transmitter to lie off the paraboloid's axis.

1.5 Outline of the Report

The geometrical approximation is described in Section 2. After Section 3 describes the configuration of the system, Section 4 solves exactly, within the geometrical approximation, the problem of tracing a ray forwards from the reflecting surface to the array. In Section 5, the paraxial approximation—a useful generalisation of the large-range approximation—is introduced; within the paraxial approximation, simple formulae describing the pseudoimage are obtained. Returning to an exact treatment, Section 6 presents results obtained from the ray-tracing algorithm. Section 7 describes an algorithm that inverts the ray-tracing procedure, thus obtaining the pseudoimage of any point on the array. Section 8 discusses in turn three programs that call on the inversion routine and presents numerical results from each program. Section 9 shows graphically how the pseudoimage of a square develops away from the parallelogram shape as the size of the square is increased. The conditions on the geometrical approximation are given in Section 10, while Section 11 shows that, under specified conditions, the algorithm of the present report may be applied to a general smooth surface. Conclusions are presented in Section 12.

2. Geometrical Approximation

In the UAI system, image-forming to produce a 3D image proceeds by combining the analytic (*i.e.* complex) signals (Bellanger, 1984, pp. 244–248) from all the sensor elements by a standard delay-and-add procedure (Steinberg, 1976). Exact path lengths are used for applicability in the near field (*e.g.* Knudsen, 1989; Blair and Anstee, 2000). For each image point $\mathbf{r}$ (each point in the 3D image), the absolute value of the resulting complex image amplitude is calculated. In the UAI project, no attempt is made to show these absolute amplitudes at *all* points $\mathbf{r}$ in 3D space. Instead, along each line of sight from the 'viewing point' (for example the center of the receiving array) the *maximum* amplitude is found and displayed (coded as brightness or colour). The above description is for the *fully coherent* mode of operation. In the *partly coherent* mode, the array is subdivided into *tiles*, and for each tile the image-forming as above is carried out. Then, for each image point $\mathbf{r}$, one adds together the *absolute values* of the image amplitudes due to the various tiles to produce the final image. This mode produces a lower resolution, but has the advantage that the image can be computed much more rapidly. Note that for a *fully* coherent array, 'tile' is interpreted to mean the whole array.

The *geometrical approximation*, introduced in I, consists of assumptions 1 and 2 as follows. Assumption 1 is that the physical reflection from the surface is described by geometrical acoustics (analogue of geometrical optics). Assumption 2 is that, as a result of the image-forming, the nth element R_n produces a bright point (a small spot) in the image at any corresponding *reflecting point* S_n, that is, any point that specularly reflects a ray to R_n. (We shall see that there may be more than one such point—there may be as many as three.)

The reason for the bright point (and hence bright regions) can be explained at a more fundamental level than in I. The explanation is in terms of 'path pairs' that contribute to the image amplitude at a given point $\mathbf{r}$ in the scene being viewed. Constructive interference between many such path pairs leads to an enhanced brightness at $\mathbf{r}$ if incident rays reaching $\mathbf{r}$

are specularly reflected to a point *within* the array. The argument is outlined in Appendix A.[12] (The possibility of spurious images is discussed in Section 10.3.) We return to the above argument in Section 12.

We must be aware that there is always *some* blurring of the pseudoimage due to wave effects; specifically, due to 'numerical diffraction' that arises in the image-forming (see I). Such blurring has two consequences. First, often the elements are so close together that, because of blurring, we may regard them as forming a continuous distribution of element strength over the tile. The result is that the ith tile T_i produces a pseudoimage that is an extended bright region. In this report we assume that the elements effectively do form a continuous distribution. The condition for the validity of this assumption is derived in Section 10.2.

Second, the pseudoimage of the tile—or for that matter the pseudoimage of the whole array—is not a bright region that has sharp, geometrical edges, but is itself blurred to a greater or lesser degree. Let us discuss the case of the partly coherent mode. (The corresponding results for the fully coherent mode then follow as a special case.) Consider the distance b over which blurring of the pseudoimage occurs due to wave effects. The size of the pseudoimage (according to the geometrical approximation) is $L' = |m|L$, where m is the magnification. Let each tile have size a; then the pseudoimage of a tile has size $a' = |m|a$. The geometrical approximation is good (at least, as far as wave effects are concerned) provided b is small on the scale of L': for then all sizes on the scale of L' are predicted with reasonable accuracy by that approximation. In these circumstances, b may or may not be small on the scale of a'. If it is not small, the tile's pseudoimage will have a blurred appearance. For this reason we shall refer to the tile's pseudoimage as an 'extended spot' or simply a 'spot.' An experimental image made up of such spots is shown (in false colour) in Figure 1 of I. In that case it must be stressed that the geometrical approximation still holds (*i.e.* on the scale of L'), even though the pseudoimage of a tile is blurred considerably. In the (quite different) case where b is small on the scale of a', we may say that the geometrical approximation holds 'in the strong sense.'

At this point we draw attention to a possible misconception, arising from the following line of thinking. First, there is an image of the transmitter produced by the specular surface, where 'image' here means 'image' in the usual sense of geometrical optics. Second, the pseudoimage is an out-of-focus version of this image—out-of-focus because it is forced to lie on the specular surface. Hence the pseudoimage is inherently blurred, even in the limit of infinitely small wavelengths. This line of thinking is to be rejected. It is not helpful to introduce the 'ordinary' image. The blurring of the pseudoimage approaches zero as the wavelength approaches zero. We shall return to this point in Section 12.

As in I, we shall place emphasis, not on the fully coherent mode, but on the partly coherent mode combined with a sonar array of tiles whose centres make up a *regular lattice* of points—in particular, a lattice whose unit cell is a square. These conditions hold for the above experimental image. In that image, the lattice of spots also has essentially a square unit cell.[13] More generally (Consider a reflector that is curved and/or obliquely inclined.), the spots would

[12] Incidentally, the argument in Appendix A shows that Assumption 2 cannot be maintained at the level of a mapping from a mathematical point (or an element) to another mathematical point. Rather, when Assumption 2 holds, it is in the sense of a mapping from a not-too-small region to a not-too-small region.

[13] The small, random departure from squareness is due to marine growth on the surface of the otherwise plane reflector.

form a lattice whose unit cell is not a square but a parallelogram, provided that the large-range approximation—or more generally the paraxial approximation, discussed in Section 5—holds. In other words, *consider any rectangle, formed of tiles*: its pseudoimage is an arrangement of spots whose external shape and unit cell are both parallelograms.

This report studies the distortion that occurs when the paraxial approximation does not hold (broadly, when the array size is no longer small compared to the range[14]). As the rectangle of tiles is increased in size, we expect the four sides of the pseudoimage to become increasingly curved, yielding what might be called a *curved quadrilateral*, or *c-quad*. The c-quad underlying the spot positions (positions of spots along the sides of the c-quad)[15] will be studied in Section 9; arguably that section contains the results of this report that are of most interest.

For a sonar array operated in *fully* coherent mode, consider an array whose external shape is square. When the geometry becomes non-paraxial, the pseudoimage changes from a (bright) filled-in parallelogram to a filled-in c-quad. (Under conditions favourable to the geometrical approximation, discussed in I, wave effects produce only slight blurring at the edges of the pseudoimage.)

To recapitulate, the geometrical approximation, in its pure form, yields the same pseudoimage (of the whole array) in the partly coherent mode as in the fully coherent mode. However, when it is subsequently recognised that there are *some* wave effects, it is seen that when these are not too large, the principal effect of changing to the partly coherent mode is towards breaking up the whole pseudoimage into extended spots, one spot for each tile. Such an image is shown in paper I.

It follows that a specular reflector may often be distinguished from a diffuse reflector as follows. When part of the array is 'switched off' (by not including its contribution in the image-forming equation), in the diffuse case the image simply becomes more blurred, while in the specular case part of the image tends strongly to disappear. This effect was observed in the 'Pyrmont 2' trial.

Let us return to the partly coherent mode. On occasions, a rectangle of tiles may have as many as three pseudoimages. The system then produces the superposition of these three pseudoimages; that is, the superposition of three c-quads. (Any sufficiently bright local optima are superimposed as well.) Although we have used the term 'superposition,' a little thought shows that, when the geometrical approximation is valid, no two pseudoimages overlap each other. However, they may abut each other, as will be seen in Section 9.2.

In line with what has been said earlier in this section, the present report on the departures from the paraxial image (or the large-range image) assumes the geometrical approximation: specifically, that that approximation holds on the scale of the *array* (*i.e.* there is little blurring on the scale of the array). The conditions for this are given in I and summarised in Section 10; the above experimental image satisfies the conditions.

---

[14] The paraxial approximation applies up to a significantly larger array size in the case of a near-planar array.

[15] If the tile is sufficiently small, the centre of each spot lies at the reflecting point S_i that reflects an incident ray to the *centre* of the tile. (In general, due to curvilinear distortion on the scale of one tile, there is a further displacement of the centre of the spot.)

3. Geometry; Reflecting Surface

3.1 Coordinate Systems

The *chief reflecting point* S_0 on the reflecting surface σ is defined to be such that the normal to σ at S_0 passes through the point transmitter T (Fig. 3.1). (It can be shown that a *spherical* transmitter is equivalent to a point transmitter located at the spherical centre, as the image amplitude function produced is the same.) The normal S_0T, called the *chief normal*, intersects the plane τ of the array at the *chief receiving point* T_0; the distances T_0S_0 and T_0T are called the *range* r_0 and the (signed) *offset* e respectively. The *chief tangent plane* μ is the tangent plane to the reflecting surface at S_0. Let ν be the plane passing through T_0 parallel to μ. We define a right-handed Cartesian system uvw with the w axis along the chief normal and v along the line of intersection of τ and ν. (If τ and ν are the same plane, v is chosen arbitrarily.) The sense of the v axis is chosen so that the angle δ between τ and ν lies between 0 and $\pi/2$ (τ must end up lying 'above' ν in Fig. 3.1.) (Note that ν, the Greek letter representing the plane, is to be distinguished from v in uvw.)

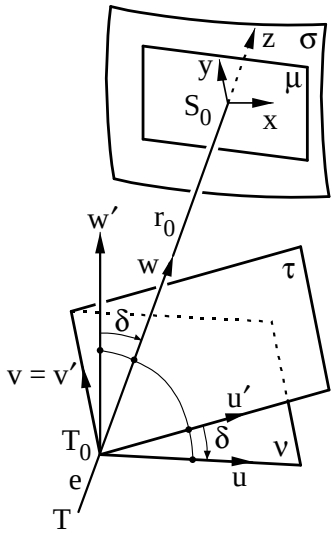

Figure 3.1. Definition of the uvw, xyz and u′v′w′ coordinate systems.

A new frame xyz is obtained by a translation of the uvw axes from T_0 to S_0; the x and y axes lie in the chief tangent plane. One obtains the $u'v'w'$ axes by rotating the uw axes about v through the angle δ; u' and v' lie in the plane of the array. From Figure 3.1 the transformations between the various coordinate systems can be written down. When relating this report to an experimental image, it will be necessary also to transform from the $u'v'w'$ frame to a predetermined frame based on the array of sensor elements.

The portion of the reflecting surface σ sufficiently near S_0 is important because only that portion will reflect energy back *to the array*; energy reaching other parts of σ will not be detected. (Here it is supposed that the array is near the transmitter.) We shall call w', the direction of the array normal, the *broadside* direction. Then the angle δ may be thought of as *the departure of* S_0 *from broadside* when viewed from T_0.

Note that the chief receiving point does not necessarily coincide with the center of the array. However, the chief receiving point is a useful origin to choose [for the *uvw* coordinates and the $u'v'w'$ coordinates] because the results (for the location of the pseudoimage point) then take on the simplest mathematical form. In particular, for small displacements of the object point from that origin, the paraxial approximation (and the large-range approximation) is accurate.[16] Note also that the chief normal is not necessarily normal to the plane of the array.

(Also, it is a mistake to think that the transmitter necessarily lies at the center of the receiving array, or even that the transmitter lies on the perpendicular to the array plane passing through that center.)

3.2 Reflecting Surface

The smooth reflecting surface, of the form $z = f(x, y)$, is, to first order in x and y, simply the tangent plane at S_0. To next order, z is of the form $z = ax^2 + 2hxy + by^2$: the surface is a paraboloid. We study surfaces of three general shapes, the first shape being the (exact) paraboloid. Then, by a rotation

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.1}$$

to new axes X and Y, called the *principal axes*, the equation of the surface may be cast into the form

$$z = \frac{X^2}{2\rho_X} + \frac{Y^2}{2\rho_Y} \tag{3.2}$$

with no term in XY (Perlis, 1952, pp. 84, 186; Guggenheimer, 1963, pp. 209–213; McCrae, 1960, pp. 99–101; Kreyszig, 1959, pp. 124–134). Here $\rho_X = 1/\kappa_X$ and $\rho_Y = 1/\kappa_Y$ are the *principal radii of curvature*. (If the surface is only asymptotic to a paraboloid, ρ_X and ρ_Y are still called the principal radii of curvature of the surface at S_0.) A positive ρ_X means that the reflecting surface is convex along its intersection with the XZ plane. For definiteness we choose $\kappa_X \geq \kappa_Y$ and $-\pi/2 < \alpha \leq \pi/2$. If κ_X and κ_Y are of the same sign the surface is an *elliptic paraboloid*; if of opposite signs, a *hyperbolic paraboloid* (Spiegel, 1968, p. 52). (In the latter case the origin is a saddle point.) If just one of κ_X and κ_Y is zero, the surface is a *parabolic cylinder*; if both are zero it is a plane. A special case of the elliptic paraboloid, occurring when $\kappa_X = \kappa_Y$, is the paraboloid of revolution, to which a *sphere* is a close approximation near S_0. Similarly an ordinary *cylinder* is a close approximation to the parabolic cylinder.

Note that, in the case of the paraboloid, attention has been restricted to the subcase in which the chief reflecting point coincides with the vertex of the paraboloid; in that subcase it follows that the chief normal coincides with the axis of the paraboloid. The algorithm developed on this basis does, however, hold somewhat more generally, as will be discussed in Section 11.

Due to their prevalence in engineering structures, we study two other shapes for the surface, namely the (exact) sphere and the cylinder, given respectively by

[16] Provided also that the point of reflection is close to the chief normal.

$$X^2 + Y^2 + (z - \rho_X)^2 = \rho_X^2 \tag{3.3}$$

$$X^2 + (z - \rho_X)^2 = \rho_X^2 \tag{3.4}$$

Again the radius of curvature ρ_X may have either sign. Only that half of the complete sphere or cylinder is considered present that has $0 \le z/\rho_X \le 1$. The shapes (3.3) and (3.4) each become asymptotic to a paraboloid (3.2) when X and Y both approach zero.

4. Geometrical Solution in the General Case

We consider the following ray-tracing problem: Given a reflecting point $\mathrm{S}(x_s, y_s, z_s)$, find the point $\mathrm{R}(u'_R, v'_R, 0)$ on the array to which a ray, incident at S from the transmitter, is reflected. Except in the early and late stages of the calculation, the mathematics is carried out in the xyz coordinate system.

Consider first the paraboloidal shape (3.2). Given the first two coordinates, x_s and y_s, of S, the corresponding coordinates X_s and Y_s, along with $z_s = Z_s$, are given by Equations (3.1) and (3.2). The tangent plane at S has the equation

$$Z - Z_s = \kappa_X X_s (X - X_s) + \kappa_Y Y_s (Y - Y_s) \tag{4.1}$$

and so the vector

$$\mathbf{b}' = (\kappa_X X_s, \kappa_Y Y_s, -1) \tag{4.2}$$

points along the normal (outward from the reflecting surface).

Let $\hat{a}$, $\hat{b}$ and $\hat{c}$ be unit vectors directed outwards from S along, respectively, the incident ray, the normal and the reflected ray, expressed in xyz (not XYZ) coordinates. Since the transmitter position is $(x, y, z) = (0, 0, -r_0 - e)$, we have

$$\hat{a} = -(x_s, y_s, r_0 + e + z_s)/F \tag{4.3}$$

where

$$F = \left[x_s^2 + y_s^2 + (r_0 + e + z_s)^2\right]^{1/2} \tag{4.4}$$

Let $\mathbf{b}$ be defined as the same vector as $\mathbf{b}'$ but expressed in xyz coordinates. Then from (3.1) we have

$$\begin{bmatrix} b_x \\ b_y \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} b'_X \\ b'_Y \end{bmatrix} \tag{4.5}$$

while

$$b_z = b'_Z = -1 \tag{4.6}$$

Then $\hat{b} = \mathbf{b}/G$, where

$$G = \left[(\kappa_X X_s)^2 + (\kappa_Y Y_s)^2 + 1\right]^{1/2} \tag{4.7}$$

The law of reflection can be written in the form

$$\hat{c} = 2(\hat{a} \cdot \hat{b})\hat{b} - \hat{a} \tag{4.8}$$

so that $\hat{c}$ is now given in terms of x_s and y_s.

A general point $\mathrm{U}(x, y, z)$ on the reflected ray SR is given by

$$(x, y, z) = (x_s, y_s, z_s) + t\hat{c} \tag{4.9}$$

where the parameter t (≥ 0) is the distance of U from S. From Figure 3.1, the equation of the array plane τ is

$$-x\sin\delta + (r_0 + z)\cos\delta = 0 \tag{4.10}$$

The value of t at the particular point U = R is given by substituting (4.9) into (4.10), after writing $\hat{c}$ in (4.9) as $(\hat{c}_x, \hat{c}_y, \hat{c}_z)$. Thus we find

$$t = \frac{(r_0 + z_s)\cos\delta - x_s \sin\delta}{\hat{c}_x \sin\delta - \hat{c}_z \cos\delta} \tag{4.11}$$

When this value is substituted into (4.9), the right-hand side of (4.9) gives (x_R, y_R, z_R), the coordinates of the desired point R on the array.

From Figure 3.1, for any point the *uvw* coordinates are given by the transformation

$$u = x, \quad v = y, \quad w = r_0 + z \tag{4.12}$$

The $u'v'w'$ coordinates are then given by

$$v' = v, \quad \begin{bmatrix} u' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\delta & \sin\delta \\ -\sin\delta & \cos\delta \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix} \tag{4.13}$$

Equations (4.12) and (4.13) apply in particular to the point R; we simply append the subscript R to all the coordinates. Thus finally we have the desired result for u'_R and v'_R. As a check, it may be verified that the resulting value of w'_R is zero.

For the sphere and the cylinder, the corresponding equations are derived by the same method as above. In the case of the sphere, the only differences in the final results are as follows. Equation (3.2) (invoked above Eqn 4.1) is replaced by Equation (3.3). Equations (4.2), (4.6) and (4.7) respectively are replaced by

$$\mathbf{b}' = (\kappa_X X_s, \kappa_X Y_s, -1 + \kappa_X Z_s) \tag{4.14}$$

$$b_z = b'_Z = -1 + \kappa_X Z_s \tag{4.15}$$

$$G = \left[(\kappa_X X_s)^2 + (\kappa_X Y_s)^2 + (1 - \kappa_X Z_s)^2\right]^{1/2} \tag{4.16}$$

(Equation 4.1 is also replaced, but the new equation is omitted here as it does not form part of the final algorithm.)

For the cylinder, Equation (3.2) of the paraboloid is replaced by Equation (3.4). Equations (4.2), (4.6) and (4.7) respectively are replaced by

$$\mathbf{b}' = (\kappa_X X_s, 0, -1 + \kappa_X Z_s), \tag{4.17}$$

Equation (4.15) and

$$G = \left[(\kappa_X X_s)^2 + (1 - \kappa_X Z_s)^2\right]^{1/2} \tag{4.18}$$

The overall result for each of the three shapes is a nest of formulae such that, given the point S, the point R is specified. Our primary interest, however, is in the inverse problem: given R, find S. This is because, given any point R on the array plane (at least if occupied by an element), any corresponding point S is a pseudoimage of R in the geometrical approximation. Hence, to determine the pseudoimage(s), some inversion procedure must be applied to the 'forward' algorithm set out in the present section. Such a procedure is described in Section 7.

5. Paraxial Approximation

Within the geometrical approach, we now introduce an approximation that will be found useful when discussing numerical results—useful because it represents a limiting case. The approximation turns out to be a slight generalisation of the large-range approximation discussed in I.

5.1 The Approximation

To motivate the approximation, we suppose that r_0, e, κ_X and κ_Y are fixed, along with the angles δ and α, and look for the asymptotic form of the relationship between the coordinates (u'_R, v'_R) of R, on the one hand, and the coordinates (x_s, y_s) of S, on the other—the asymptotic form as the two vectors, (u'_R, v'_R) and (x_s, y_s), approach zero. Note that the rays are then *paraxial*, that is, they make a small angle with the chief normal. (Usually the smallness of the one vector implies the smallness of the other, but we shall encounter two exceptions to this: one in Section 5.3 and the other at the point H in Fig. 6.1.)

The calculation proceeds as in Section 4 but with two differences. First, it is convenient to carry out the calculation in all the early stages in the XYZ (not xyz) frame and then switch at a certain stage to the UVW frame. The latter frame is defined to have axes parallel to the XYZ axes but with the origin translated from S_0 to T_0 (Fig. 3.1), so that the transformation between the two frames is

$$U = X, \quad V = Y, \quad W = r_0 + Z \tag{5.1}$$

The second difference is that, due to paraxiality, many terms can be dropped.

Consider first the paraboloidal shape. Then Equations (4.1) and (4.2) remain unchanged. In (4.3) and (4.4), the new equations are obtained by replacing $\hat{a}$, x_s, y_s and z_s by $\hat{a}'$, X_s, Y_s and Z_s. But, in the equation replacing (4.4), X_s, Y_s and Z_s may be dropped, being small compared to $r_0 + e$. Equations (4.5) and (4.6) are not needed, and at (4.7) we have $|\mathbf{b}'| = G = 1$. In (4.8), we simply add a prime to each variable. Let us define σ_X and σ_Y by

$$\frac{1}{\sigma_X} = \frac{2}{\rho_X} + \frac{1}{r_0 + e}, \qquad \frac{1}{\sigma_Y} = \frac{2}{\rho_Y} + \frac{1}{r_0 + e} \tag{5.2}$$

Then the following simple result for $\hat{c}'$ is obtained:

$$\begin{aligned} \hat{c}'_U &= \hat{c}'_X = X_s/\sigma_X, \\ \hat{c}'_V &= \hat{c}'_Y = Y_s/\sigma_Y, \qquad \hat{c}'_W = \hat{c}'_Z = -1 \end{aligned} \tag{5.3}$$

where $\hat{c}'_U$, $\hat{c}'_V$ and $\hat{c}'_W$ are the components of the unit vector $\hat{c}'$ expressed in the UVW system.

The analogue of Equation (4.9) is

$$(U, V, W) = (U_s, V_s, W_s) + t\hat{c}' \tag{5.4}$$

But we immediately see that, for the purpose of calculating the first two components, $U = U_R$ and $V = V_R$, at R, a sufficient approximation for the distance t is $t = r_0$. For there are just two possible sources of error. First, the small paraxial angle causes the ray travelling from S to the uv plane to be a little longer than r_0. But this lengthening is a higher-order correction to t, which is negligible. Second, the ray is shortened because it travels, not to the uv plane, but to

the array plane τ (Fig. 3.1). This shortening is of the order of $\left(u^2+v^2\right)^{1/2}\tan\delta$. For fixed r_0 and fixed $\delta \neq \pi/2$, this shortening becomes negligible compared to r_0 in the limit $u \to 0,\ v \to 0$. Hence we may put $t = r_0$, as claimed. (This will give an incorrect value for the third coordinate W_R—to the first order in small quantities—but that value will not be used, as we shall see.)

With this value of t, it follows from Equations (5.1) to (5.4) that the expressions for U_R and V_R are

$$\begin{aligned} U_R &= r_0\left(\frac{1}{r_0}+\frac{2}{\rho_X}+\frac{1}{r_0+e}\right)X_s \equiv \frac{1}{m_X}X_s \\ V_R &= r_0\left(\frac{1}{r_0}+\frac{2}{\rho_Y}+\frac{1}{r_0+e}\right)Y_s \equiv \frac{1}{m_Y}Y_s \end{aligned} \tag{5.5}$$

Hence the system possesses a pair of *principal magnifications*, namely m_X and m_Y. Equation (5.5) gives expressions for these. In the parentheses in Equation (5.5), the first and third terms may be ascribed to the fact that, within the approximation, r_0 and r_0+e are (to a sufficient approximation) the distances from the reflecting point S to the receiving point R and the transmitter respectively.

It remains to relate the UV coordinates of R, given by (5.5), to the uv coordinates and then the $u'v'$ coordinates of R. The first of these two steps is straightforward, since the rotation matrix in (3.1) that relates xy to XY also relates uv to UV. Then from Figure 3.1, since R lies on the plane τ, the $u'v'w'$ coordinates of R are determined by the uv coordinates (within the paraxial approximation) according to

$$\begin{bmatrix} u_R \\ v_R \end{bmatrix} = \begin{bmatrix} \cos\delta & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} u'_R \\ v'_R \end{bmatrix}, \qquad w'_R = 0 \tag{5.6}$$

(via a simple inversion).

5.2 Discussion

A key result from the above derivation is that, under paraxial conditions, the coordinates (x, y) of S are obtained from the coordinates (u', v') of R by a sequence of *linear* operations. It follows that, under these conditions, the pseudoimage of a square is a *parallelogram*.

5.3 Conditions of Validity

We continue to consider the paraboloidal shape. Then it can be shown that the conditions (5.7) to (5.10) below jointly ensure that the paraxial approximation is valid (subject to the validity of the geometrical approximation)[17]:

$$r_0 + e \gtrsim r_0 \tag{5.7}$$

$$\delta \text{ is not near } \pi/2 \tag{5.8}$$

[17] Here $\gtrsim$ means 'is greater than or of the order of.'

$$|\mathbf{u}'_R| << r_0 \tag{5.9}$$

$$|\rho_X - \rho_0| >> |\mathbf{u}'_R| \quad \text{and} \quad |\rho_Y - \rho_0| >> |\mathbf{u}'_R| \tag{5.10}$$

Here $\mathbf{u}'_R$ is defined as the 2D vector (u'_R, v'_R). ρ_0 (a negative quantity) is defined by the equation

$$\frac{2}{\rho_0} = -\frac{1}{r_0} - \frac{1}{r_0 + e} \tag{5.11}$$

Condition (5.9)—a paraxial or large-range requirement—may be said to be the key condition. The two requirements (5.10) are related to the fact that when (say) ρ_X approaches ρ_0, the magnification m_X, given by (5.5), approaches infinity. In such situations there are rays from the transmitter that, after reflection at a relatively large distance from the 'axis' (the chief normal), return to a position close to the axis at the array plane. Such situations were alluded to earlier, for they are situations in which the smallness of the vector (u'_R, v'_R) does not imply the smallness of the vector (x_s, y_s).

5.4 Spherical and Cylindrical Surfaces

Consider the cylindrical shape. As noted previously, the surface is asymptotically a paraboloid as $X \to 0$. To the next order, from Equation (3.4), we have

$$Z = \tfrac{1}{2}\kappa_X X^2 + \tfrac{1}{8}\kappa_X^3 X^4 + \cdots \tag{5.12}$$

so that the gradient is

$$dZ/dX = \kappa_X X + O\left[(\kappa_X X)^3\right] \tag{5.13}$$

Then the derivation given in Section 5.1 (leading to the results given by Eqns 5.5 and 5.6) remains valid, provided that we ensure that the error introduced by the cubic term in (5.13) is negligible; on this basis the condition of validity given in the next paragraph is derived. For the sphere, the argument proceeds similarly. In fact, in that case it suffices to consider the intersection of the sphere with the plane $Y = 0$; hence Equations (5.12) and (5.13) again apply.

It turns out that the conditions on the paraxial approximation in the cases of the sphere and the cylinder are again (5.7) to (5.10), except that the condition (5.10) must be strengthened to the following:

$$|\rho_X - \rho_0| >> r_0^{1/3} |\mathbf{u}'_R|^{2/3} \tag{5.14}$$

(The corresponding condition on ρ_Y is dropped.)

5.5 The Large-Range Approximation Revisited

In I, the large-range approximation was introduced, based principally on the condition $r_0 >> L$, where L is the diameter of the array (or the side in the case of a square array). In that approximation, the relationship between an object (part of the array) and its pseudoimage was shown to be given by Equations (7) to (9) of I, while Equations (10) to (13) of I give a sufficient set of conditions for the validity of the approximation. We can express the predicted object-pseudoimage relationship as follows: the large-range prediction is the same as the

paraxial prediction, Equations (5.5) and (5.6), except that the (large-range) principal magnifications are

$$m_X = \frac{\rho_X}{2(r_0 + \rho_X)}, \qquad m_Y = \frac{\rho_Y}{2(r_0 + \rho_Y)} \tag{5.15}$$

It is readily verified that these predictions are what the paraxial predictions (the right-hand equalities in Eqn 5.5) become when the transmitter offset e is put equal to zero.

We therefore expect that normally the large-range approximation is valid provided we add to the conditions (5.7) to (5.10) (or Equation 5.14 as the case may be) the further condition $|e| << r_0$. In fact, for the paraboloidal shape, it can be shown that a sufficient set of conditions for the large-range approximation consists of Equations (5.16) to (5.19), as follows.

$$|\mathbf{u}'_R| << r_0 \tag{5.16}$$

$$\delta \text{ is not near } \pi/2 \tag{5.17}$$

$$|e| << r_0 \tag{5.18}$$

$$|\rho_X + r_0| >> |\mathbf{u}'_R|, \quad |\rho_X + r_0| >> |e|, \quad \text{and the two corresponding conditions for } \rho_Y \tag{5.19}$$

Unlike the set of conditions given in I, this set does not contain the array size L. It is therefore a useful alternative set of conditions to that given in I.

We now consider the application of the large-range approximation to the sphere and cylinder. It turns out that, for either of these surfaces, the conditions in (5.19) must be replaced by

$$|\rho_X + r_0| >> r_0^{1/3}|\mathbf{u}'_R|^{2/3}, \qquad |\rho_X + r_0| >> |e| \tag{5.20}$$

(without any corresponding conditions for ρ_Y).

The key requirement of the large-range approximation is condition (5.16). In I, this relationship was thought of as r_0 being large (compared to some fixed value of $|\mathbf{u}'_R|$ or L). But of course the relationship can equally well be thought of as $|\mathbf{u}'_R|$ being small (compared to a fixed r_0), in accordance with the introduction to Section 5.1.

6. The Forward Algorithm; Results

6.1 The Algorithm

The algorithm of Section 4 gives, in effect, a complicated formula by which, given the xy coordinates of a point S on the reflecting surface, one can calculate exactly the coordinates $u'v'$ of the corresponding *object point* R, that is, the point to which a ray, reflected at S, meets the array. We call this the *forward* algorithm. By contrast, the *inverse* procedure for computing S from R requires a more complicated algorithm. Note that any reflecting point S that is related to R as above is a *pseudoimage point* corresponding to the object point R.

A function m-file (a subroutine), called ray.m, has been written in MATLAB, embodying the forward algorithm. (The other routines developed for this report are also written in MATLAB.) The code for ray is included in this report in Appendix C, along with the code for some other routines mentioned in the report. The input variables to ray include shape and Zb,

defined as follows. shape is 1, 2 or 3, for a reflector that is a cylinder, sphere or paraboloid respectively. The reflecting surface may be cut off at a plane $Z = \text{constant}$, in which case the latter is one of the two planes $Z = \pm \text{Zb}$. The motivation for the cutoff, along with further details, are given in Section 6.2.

Two alternative script m-files (main routines), raymain.m and rysuite.m, have been written, each of which makes calls to ray and presents the results via a curve (raymain) or a suite of curves (rysuite). raymain is a relatively simple routine. Its output is intuitively appealing in that it plots object position versus reflecting point position on a *linear* scale; nevertheless it has been found that raymain is used less often than rysuite. An example of the use of raymain (Fig. 6.1) is given in Section 6.3.1.

rysuite produces a suite of curves on the one graph. However, the program has the limitation that its use is confined to systems with a considerable degree of symmetry. Basically, the symmetry requirement is[18]

$$\kappa_X = \kappa_Y, \quad \delta = 0 \tag{6.1}$$

The first of these two requirements means that the reflecting surface σ is a surface of revolution. The double requirement (6.1) entails that the entire system (apart from the detailed placement of the tiles and elements on the array surface) has *circular symmetry* (invariance under rotation about the z or w' axis). Partly because of this, dimensionless or scaled variables can be used to advantage, as will be discussed shortly.

Given the surface σ, each reflecting point $\text{S}(x, y, z)$ is determined by the corresponding *projected* reflecting point $\text{S}'(x, y, 0)$. In rysuite, each curve is generated by taking, as inputs, projected points $\text{S}'(x, y, 0)$ lying along a line through the origin. The various curves describe object points pertaining to the same line of reflecting points S (after projection), but those curves are characterised by different values of the product $\kappa_X r_0$. The other independent dimensionless variables, such as e/r_0, are held constant throughout for a given graph. Due to the symmetry (6.1), all lines through the origin in the xy plane are equivalent, so we may restrict attention to input points lying along the line $y = 0$ without loss of generality. Then each curve is essentially a plot of u' as a function of x.[19] (More correctly, scaled values of u' and x are plotted, as will be discussed shortly.) Because of the circular symmetry (6.1), the x direction is not special, and the curve is simply a plot of *radial* displacement in the $u'v'$ plane versus radial displacement in the xy plane.

Regarding scaling, in the symmetric case (6.1), the reflecting point – object point relationship may be written as

$$\frac{u'}{x} = f\left(\kappa_X r_0, \frac{x}{r_0}, \frac{e}{r_0}, \text{shape}, \frac{\text{Zb}}{r_0}\right) \tag{6.2}$$

For a given graph, the last three arguments of f are held fixed, while $\kappa_X r_0$ (the first argument) specifies a particular curve.

---

[18] An exception to the requirement $\kappa_X = \kappa_Y$ is that a cylindrical surface is allowed, provided that all the input points S considered lie in a common plane perpendicular to the axis of the cylinder.

[19] There is no need to plot v' because the symmetry ensures that $v' = 0$.

The curve itself is a plot of the inverse magnification u'/x versus $\log_{10}(x/r_0)$; an example of such a graph from rysuite is shown in Figure 6.2. (This graph is discussed briefly here and further in Section 6.3.) The interpretation of the vertical axis is as follows. Consider a vector $\mathrm{O'P'}$ in the chief tangent plane, drawn from the chief reflecting point $\mathrm{O'}$ $(=\mathrm{S}_0)$ to a point $\mathrm{P'}(x, y)$. The reflected ray maps $\mathrm{O'}$ to the chief receiving point O $(=\mathrm{T}_0)$ and $\mathrm{P'}$ to a point $\mathrm{P}(u', v')$ in the array plane. Plotted along the vertical axis is the 'inverse magnification' $\mathrm{OP}/\mathrm{O'P'}$. The qualifier 'inverse' is inserted here because our primary interest is in the magnification $\mathrm{O'P'}/\mathrm{OP}$ produced in the mapping of the vector OP to $\mathrm{O'P'}$, which is its pseudoimage (to be precise, the *projection* of the pseudoimage onto the chief tangent plane). Curves of that magnification will be displayed in Section 8.

Because scaled variables are used, a graph from rysuite represents many more situations than might at first be thought. It should be noted from Section 5 that, as $x/r_0 \to 0$ (left-hand end of graph), the inverse magnification must approach the constant value predicted by the paraxial approximation.

The routines rysuite and raymain have limitations. (For example, rysuite requires symmetry.) 'Forward' programs in which these limitations are removed could be written (in particular, a program analogous to the inverse program multigen.m, described below). However, rather than devote time to that task, it was decided to concentrate effort on developing a number of programs for the inverse problem, which is of greater interest.

The reason why the latter problem is of greater interest is as follows. The *array geometry is simple*, because the tile centres form a regular lattice so that the 'array objects' of interest are, for example, squares. A significant question therefore is: given a simple shape in the array—say a square—what is the shape of its pseudoimage? This picks out the 'inverse' problem as being of special interest. By contrast, there is little point in investigating the problem that begins from a *pseudoimage* of simple shape.

6.2 Virtual Object Points; the Cutoff Plane

For a given reflecting point, a corresponding genuine or 'real' object point[20] may fail to exist. This may happen for a variety of reasons. For example, the reflected ray may be travelling away from the array rather than towards it. ray reports the mode of failure, by means of a 'fail code,' which is an integer ranging from 2 to 9. In such cases ray still calculates a notional output pair (u', v'). In many cases these calculated values have some physical significance (*e.g.* the point obtained by extending the reflected ray backwards to the array). In such cases the calculated point is called a *virtual* object point. In the remaining cases the calculated point has no physical significance (and in fact is based on an arbitrary instruction in the routine). Whenever there fails to be a real object point, we shall speak of a *forward failure*. Details of the fail codes, and of the cases where a virtual object point exists, are given in ray in Appendix C.

In raymain and rysuite, optionally the curve of virtual points is output as well as the curve of real object points.

[20] Here 'real' is used in the sense of optics (real versus virtual, not real versus complex or imaginary).

As stated, in ray and the other routines, the reflecting surface may be cut off at one of the two planes $Z = \pm \mathrm{Zb}$. The reasons for introducing this feature are as follows. First, note that if at least one of the two curvatures is negative, the surface, if not cut off, will extend to the array plane and beyond. Second, as portions of the surface nearer to the array are considered, where the surface becomes more steep (becomes more inclined to the chief tangent plane), it becomes more likely that the reflected ray will be reflected a second time. It is not appropriate, in the present work, to cover the calculation of second-reflected rays; rather, such rays are simply regarded as lost to the imaging process. The cutting-off of the surface often restores to the imaging process a ray that would otherwise be second-reflected and lost. For this reason, cutting off the surface often makes the geometry more simple overall. Note that in a real system the reflecting surface is always cut off somewhere.

In ray, as an option, a default value of Zb is available in the programs. The cutoff value has been chosen small enough so that forward failure, when it occurs, is usually due to the cutoff: other types of forward failure tend to be pre-empted. It is therefore believed that the default cutoff produces something like the simplest overall geometry. The numerical results presented in this report are all based on the default cutoff except where otherwise stated.

The details of the default cutoff are as follows. For a paraboloid with both $\kappa_X \geq 0$ and $\kappa_Y \geq 0$, no cutoff is imposed. For all other paraboloids, let $\kappa_0 = \left|\min\left(\kappa_X, \kappa_Y\right)\right|$ and

$$\mathrm{Zb} = \min\left(\frac{1}{2\kappa_0}, \tfrac{1}{2}\kappa_0 r_0^2\right) \tag{6.3}$$

Then we impose a cutoff at $Z = -\mathrm{Zb}$; there is no cutoff at positive Z. For a sphere, let $\kappa_0 = \left|\kappa_X\right|$ and define Zb by Equation (6.3). A cutoff is imposed at either $Z = \mathrm{Zb}$ or $Z = -\mathrm{Zb}$, whichever of the two planes intersects the spherical surface. A cylinder is treated in the same way (the principal curvature that is zero being ignored).

6.3 Results

6.3.1 Results from raymain

Figure 6.1 shows the object position versus reflecting point position, plotted on a linear scale using the program raymain. The case considered involves a paraboloidal reflector and possesses the symmetry (6.1). The parameter values (in SI units) are $r_0 = 0.9$, $e = 0$ and the product $\kappa_X r_0 = -1.375$. As mentioned, due to the symmetry, all lines through the origin in the xy plane are equivalent, so we may consider just projected reflecting points that lie on the x axis without loss of generality. The corresponding object points have $v' = 0$.

As predicted in Section 5, near the origin (incident ray near chief normal), the curve of u' versus x approaches the linear relationship given by the paraxial approximation (shown as a dash-dot line).

At large x the deviation from the paraxial prediction can be large. For the concave surface studied, for positive x, u' changes from a decreasing to an increasing function of x and eventually u' changes sign. The last means that the ray, which for small x is reflected back across the chief normal, eventually upon reflection does not cross that normal.

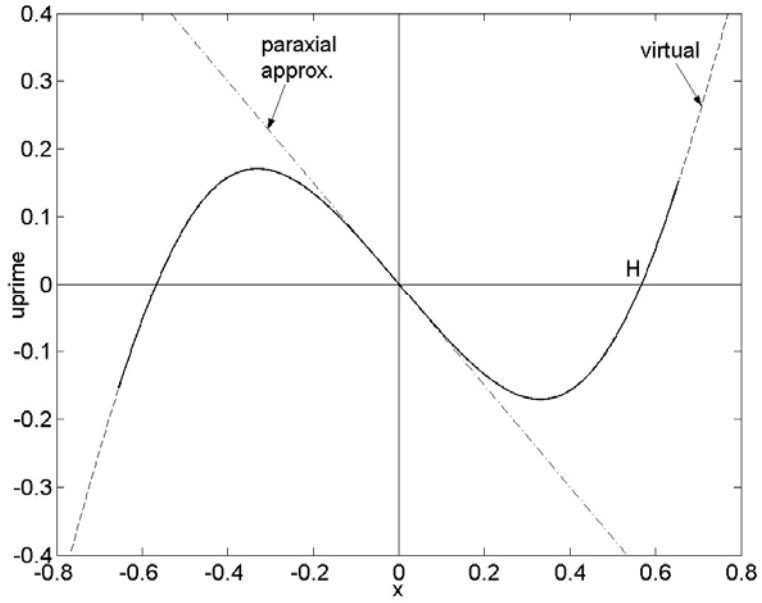

Figure 6.1. Object position plotted against reflecting point position, in a symmetric paraboloidal case. For parameter values see text. Obtained via program raymain. *The dashed portions of the curve represent virtual object points.*

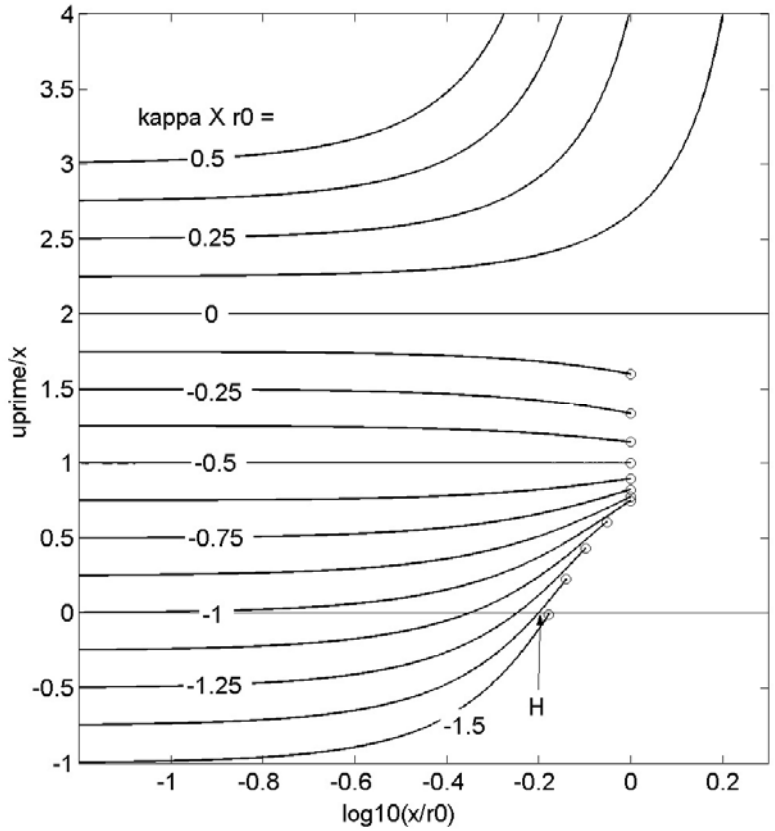

Figure 6.2. Inverse magnification plotted against scaled reflecting point position, for symmetric paraboloids. The values of $\kappa_X r_0$ *cover a small range and are stepped at an interval of 0.125.* $e = 0$. *Obtained via program* rysuite.

A consequence of the curve's having a turning point is that the inverse function, which maps u' to x, is in general multivalued. This has a bearing on results to be given in Section 7.

As $|x|$ is further increased, eventually only a virtual object point is obtained (dashed curve). For this particular case the program then yields a fail code of 3, showing that the reflecting point S has moved beyond the cutoff plane.

6.3.2 Results from rysuite

For a paraboloidal reflecting surface satisfying the symmetry conditions (6.1), with $e/r_0 = 0$, Figure 6.2 gives a suite of curves representing the object point as a function of the reflecting point position, both variables being scaled. (x is taken positive without loss of generality.) The graph covers only the parameter range $-1.5 \le \kappa_X r_0 \le 0.5$. Section 5 tells us that, as x/r_0 approaches zero (left-hand end of curves), the magnification (reciprocal of the ordinate u'/x) approaches the value predicted by the paraxial prediction. It is readily checked that this claim is borne out.[21] The present graph confirms that the inverse magnification differs from a constant value when x increases sufficiently.

In Figure 6.2, only curves for real object points are shown. A circle marker is plotted where the real curve comes to an abrupt end. (For outputs discussed in this report, the convention is adopted that an abrupt end occurring at the left end of the curve is shown by a cross, at the right end a circle. The mode of failure beyond the end is discussed below.)

Two special values of the parameter $\kappa_X r_0$ are worth noting. First, the value zero represents a flat reflecting surface. Then, as simple geometry shows, the inverse magnification u'/x is precisely 2 at all values of x. Second, for $\kappa_X r_0 = -0.5$, from (3.2) simple geometry shows that the transmitter is located at the focus of the paraboloid. By a well-known property of the paraboloid, the reflected ray is then parallel with the chief normal, so that $u' = x$, as in Figure 6.2.

It is of interest to compare the curve of Figure 6.2 for $\kappa_X r_0 = -1.375$ with Figure 6.1. Although the data being plotted are the same for the two figures, the curve has a much more dramatic appearance in the simple plot of Figure 6.1. For example, note that, as x increases in Figure 6.2, the –1.375 curve crosses the horizontal line $u' = 0$ at a point H. This crossing point is to be identified with the right-hand point in Figure 6.1 where u' changes sign (also marked H). This example shows that, when working with plots like Figure 6.2, because the inverse magnification rather than u' itself is plotted, one must bear in mind that the plot downplays the 'drama' that is occurring. On the other hand, the type of plot exemplified by Figure 6.2 has some valuable features: besides displaying a large number of curves simultaneously, it enables an easy comparison with the paraxial approximation.

In Figure 6.3, the input parameters are the same as for Figure 6.2 but cover a wider range of values of $\kappa_X r_0$. The curves that run off the edge at the top and bottom merit discussion. As is

[21] In the present case, since $e = 0$, this prediction reduces to that of the large-range approximation. However, in Section 9.2, graphs are presented that confirm the 'paraxial' claim in the general case $e \neq 0$.

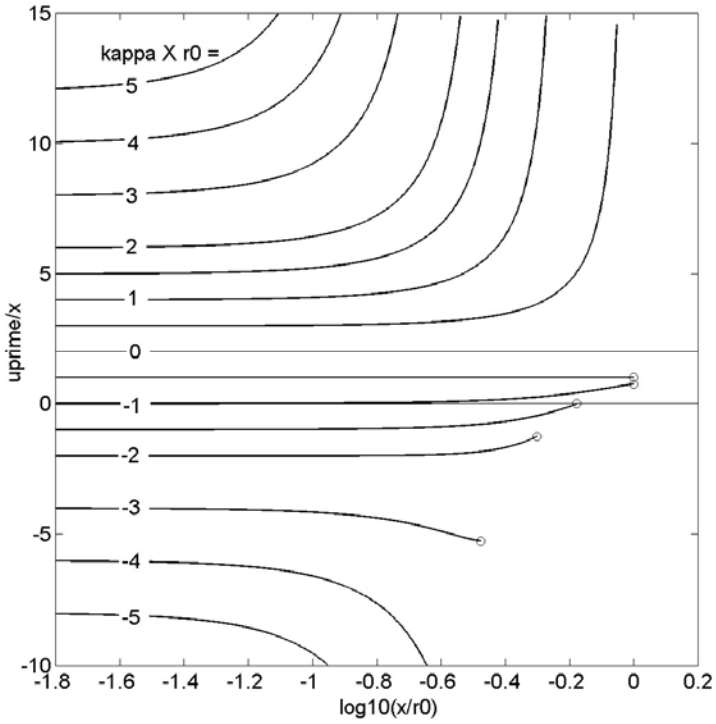

Figure 6.3. As for Figure 6.2, but for a larger range of $\kappa_X r_0$ *values. Each step in the value of* $\kappa_X r_0$ *is either 1 or 0.5.*

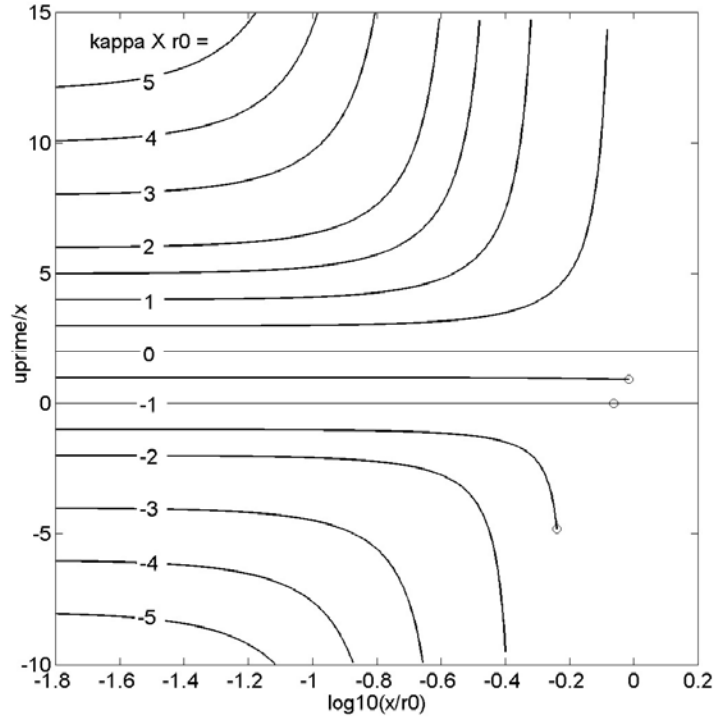

Figure 6.4. As for Figure 6.3, but for a sphere. The results apply also to a cylinder. (The curve for $\kappa_X r_0 = -1$ *coincides with the axis* $u' = 0$ *but does not extend to the right of the circle.)*

common in optics, these curves go asymptotically to $\pm\infty$ following a $(x-c)^{-1}$ singularity, where c is a constant. However, each curve abruptly ceases to represent a real object point at some stage before infinity is reached. (This outcome is assured because the forward algorithm deems that it has failed to deliver a real object point once the direction of the reflected ray comes within about 1/30 radian of being parallel to the array plane.)

A table output by rysuite (not reproduced here) shows that, as x is increased, each curve in Figures 6.2 and 6.3 eventually encounters a forward failure (except in the case of a flat reflector). The reasons for the *first* failure to occur as x increases turn out to be twofold. For all the positive curvature cases, the failure occurs when the reflected ray travels in a direction away from the array, or nearly in such a direction (as judged by the '1/30 radian' criterion above). For the negative curvature cases, the failure occurs because the point S lies beyond the cutoff plane. A diagram (which the user is invited to sketch) shows that these results are not surprising. (The conclusions given in this paragraph and the preceding one are found to apply also in the case of a spherical surface, described below.)

Figure 6.4 is obtained when the inputs are precisely the same as for Figure 6.3 but a sphere rather than a paraboloid is considered. Clearly the results apply also to a cylinder (with the displacement (x, y) of the projected reflecting point S′ at right angles to the axis of the cylinder).

For the sphere, three values of $\kappa_X r_0$ constitute special cases. The value zero yields a plane reflector, as before. The value –1 places the transmitter at the centre of the sphere. Clearly all rays are then reflected back to the transmitter and u' is identically zero. When $\kappa_X r_0 = -0.5$, the transmitter is at the focus of the spherical reflector for paraxial rays (as in geometrical optics). Hence, as an approximation, the reflected ray is parallel to the chief normal and we have $u' = x$. The figure shows that this is a fairly good approximation right out to the abrupt end point.

Comparison with Figure 6.3 shows that the results are similar to those for a paraboloid. The most noticeable difference is for $\kappa_X r_0$ around –2, as follows. Consider x to be (positive and) increasing. Whereas the inverse magnification u'/x becomes less negative in the paraboloidal case, it becomes more negative for the sphere. A trend followed for a wider range of values of $\kappa_X r_0$ is that, for negative $\kappa_X r_0$, for a given x, u' is decreased for the sphere compared to the paraboloid. A simple diagram (which the reader may sketch) explains this as follows. The spherical surface is steeper than the paraboloidal one at a given x. The ray is therefore reflected to a point u' that is more negative than that for the paraboloid. A similar analysis explains the direction of the change when $\kappa_X r_0$ is positive: u' is increased for the sphere.

Figure 6.5 shows some typical virtual curves; the paraboloid shape has been selected. For $\kappa_X r_0 = 0.5$, there are two branches which asymptotically show $a(x-c)^{-1}$ behaviour with the same values of a and c. Similarly for $\kappa_X r_0 = 1.5$. In either case the solid curve on the left represents real object points; the other branch is associated with a ray that appears to emanate from a virtual object point. For $\kappa_X r_0$ equal to –3 and –1.5, the virtual curve (at least at the beginning of the continuation) shows how the real curve would continue if the reflecting surface were not cut off.

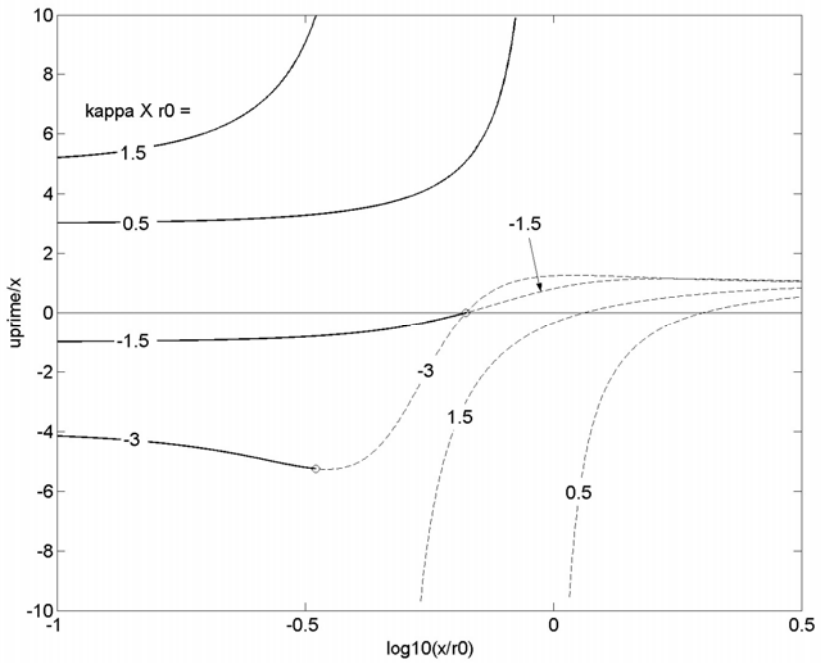

Figure 6.5. As for Figure 6.3, but showing not only the real-object curves (solid) but also the corresponding virtual curves (dashed). For clarity a reduced set of values of $\kappa_X r_0$ *is used.*

7. The Inversion Algorithm

A subroutine imige.m has been written, embodying the solution to the problem that is the inverse of that studied in Section 6. The problem that it must solve is: given an object point lying in the array plane, calculate the position of any corresponding pseudoimage points (and certain related points to be discussed). We put forward a line of reasoning that leads to the algorithm finally selected, the Levenberg-Marquardt method.[22]

Inversion methods in general are discussed by Aster *et al.* (2005). That work (pp. 14, 189) identifies some references on the subject as particularly significant. Inversion methods used in geophysics and remote sensing, but generally having wider application, have been discussed by Twomey (1996), Parker (1994) and Jupp and Vozoff (1975).

[22] After the present work was completed, it was realised that there is an alternative method of obtaining the geometrical pseudoimage of an object point. Given the transmitter location and the object point, by Fermat's principle, a corresponding pseudoimage point exists at any point on the reflecting surface such that the go-and-return path length is stationary with respect to variations of the point within the reflecting surface. There may be more than one way of implementing this idea. One way is as follows. Since the equation of the surface is known, one can write down a pair of equations (since the surface is 2D) that jointly are necessary and sufficient for the point to be a pseudoimage. These two equations could then be solved numerically via the Levenberg-Marquardt method. Note that the 'alternative' method would bypass the forward ray-tracing step.

7.1 Newton's Method

In the present acoustics problem, the forward routine maps a 2D vector $\mathbf{x} \equiv (x, y)$ into a 2D vector $\mathbf{u}' \equiv (u', v')$. (In what follows, the primes will be temporarily dropped for simplicity.) Our primary aim is to obtain an algorithm that calculates the inverse of this functional relationship. (This *initial* problem will be replaced by a somewhat broader problem in Sections 7.2 and 7.3.) An appropriate solution to this initial problem is provided by (a modified version of) Newton's method (*e.g.* Aster *et al.*, 2005, p. 171). In that method, one is given a smooth functional relationship

$$\mathbf{u} = \mathbf{f}(\mathbf{x}) \tag{7.1}$$

between column vectors $\mathbf{x}$ and $\mathbf{u}$, each of length (dimension) m. (The case $m = 1$ is best known; the present acoustics problem has $m = 2$). The problem is to find $\mathbf{x}$, given $\mathbf{u}$. The solution begins from an initial guess[23] $\mathbf{x}^0$ and applies an iterative procedure. In each iterative step, the relationship $\mathbf{f}$ is first linearised via the elements $J_{ij} = \partial u_i / \partial x_j$ of the Jacobian matrix $\mathbf{J}$ evaluated at $\mathbf{x}^0$—or, more generally, at the updated value of $\mathbf{x}$ resulting from the previous iteration. This yields the approximate relationship

$$\delta\mathbf{u} = \mathbf{J}\delta\mathbf{x} \tag{7.2}$$

where $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}^0$, $\delta\mathbf{u} = \mathbf{u} - \mathbf{u}^0$ and $\mathbf{u}^0 = \mathbf{f}(\mathbf{x}^0)$. The Jacobian matrix is then inverted to find the updated value of $\mathbf{x}$. Under suitable conditions, the procedure converges to the solution $\mathbf{x} = \mathbf{x}^s$. The convergence is quadratic, in that twice as many correct significant figures are obtained in each iteration as in the previous one.

Two complications and two limitations on Newton's method are to be noted. The complications are, first, that there may be more then one solution to Equation (7.1), and second, that there may exist no solution at all. Regarding limitations, there are two conditions which, if not fulfilled, make it necessary or advantageous to make modifications—small or large—to Newton's method. The conditions are: (i) $\mathbf{J}$ is 'well-conditioned,' that is, it is not close to being singular; and (ii) $\mathbf{x}^0$ is sufficiently close to the solution $\mathbf{x}^s$.[24] We return to these matters below.

7.2 Local Optima

As a preliminary, for any given values of $\mathbf{x}$ and $\mathbf{u}$, we define the 'error' E to be the magnitude of the vector $\mathbf{f}(\mathbf{x}) - \mathbf{u}$. In particular, we may speak of the error at the end of the nth iteration (and likewise at the end of an internal iteration, see Section 7.3) to be the value of E obtained by inserting the most up-to-date value for $\mathbf{x}$. We also define the relative error to be

$$\text{relerr} = E / |\mathbf{u}| \tag{7.3}$$

For a given $\mathbf{u}$, when there is no solution to Equation (7.1), some interest attaches to finding the $\mathbf{x}$ that minimises the error E. Such a point $\mathbf{x}$ will be called a *local optimum* (*i.e.* a point at which the error is a local minimum but not zero). Recall that a pseudoimage of an array element appears bright in the acoustic image. In some circumstances, namely when the relative error (7.3) is small, we expect that a local optimum for a given element will also appear bright,

[23] The superscripts, 0 and s, are changed to subscripts when the acoustic system is treated. This is done for convenience and is unlikely to cause confusion.

[24] There is a tendency for condition (i) to hold when (ii) holds and to fail when (ii) fails.

though generally less bright than a pseudoimage point. This phenomenon is likely to occur when, as (u', v') is varied along a line, a pseudoimage point is at some stage replaced by a local optimum: some brightness should persist while the local optimum is close to the last pseudoimage point. (Examples occur later, *e.g.* in Figures 8.5 to 8.7.)

The inversion algorithm (embodied in imige.m) has been designed to find local optima (as well as pseudoimage points), even for those $\mathbf{u}$ for which a pseudoimage point exists. Fortunately not much extra work is required to achieve this.

7.3 Levenberg-Marquardt Method

In the initial approach to the present acoustics problem, Newton's method was used but with a modification as follows: if the step specified by the method fails to reduce the error, the step size is repeatedly halved until it does. This modification goes a long way towards overcoming limitation (ii) above. It was found that the modified method works well in finding pseudoimage points and 'blockage points' (defined in Section 7.4), but it does not correctly locate local optima.

Because of the latter failure, the modified Newton method was replaced by the Levenberg-Marquardt (LM) method (described by Aster *et al*., 2005, p. 176; and in more detail by Nash and Sofer, 1996; Nocedal and Wright, 1999; and Björck, 1996). It was found that the LM method both enables the local optima to be found and also provides a more stable way of dealing with limitation (ii). In the LM method, the iterative step to be taken is given, not by (7.2), but by

$$\delta\mathbf{x} = \left(\mathbf{J}^T\mathbf{J} + \lambda\,\mathbf{I}\right)^{-1}\mathbf{J}^T\delta\mathbf{u} \tag{7.4}$$

where $\mathbf{I}$ is the identity matrix and T denotes transpose. (As usual, the Jacobian $\mathbf{J}$ is evaluated at the $\mathbf{x}$ value from the previous iteration.) The positive parameter λ is adjusted during the course of the iterations. One advantage of introducing the $\lambda\,\mathbf{I}$ term is that it ensures that the matrix to be inverted is nonsingular, thus overcoming limitation (i) above.

For $\lambda = 0$, the LM method reduces to the so-called Gauss-Newton (GN) method; the latter method has been discussed by Jupp and Vozoff (1975) and by the above four references given for the LM method.[25] The GN method is frequently used in seeking the point $\mathbf{x}$ that minimises the error[26] when m, the dimensionality of $\mathbf{u}$, exceeds n, the dimensionality of $\mathbf{x}$. The latter situation is common when fitting a model with parameters $\mathbf{x}$ to data values $\mathbf{u}$. Note that, when $m = n$, the GN method reduces further to the Newton method (as is easily seen).[27] For large λ, the LM method reduces to the method of steepest descent, in which the algorithm simply takes a small step down-gradient. Consequently, for large λ the LM method provides

[25] A major modification to the GN method, introduced by Marquardt (1963), based on earlier work of Lanczos (1958), offers a possible alternative to the LM method.

[26] Note that when $m > n$, there is no solution to Equation (7.1), only one or more local minima of the error.

[27] An algorithm based on Newton's method was tried on the present acoustics problem (which has $m = n$), with the following results. For circularly symmetric systems, it produces essentially the same results as the algorithm based on the LM method. Consider now nonsymmetric systems. Then the Newton algorithm is good at locating pseudoimage points; however, it reports 'local optima' in the wrong locations. The latter result is understandable, because neither Newton's method, nor even the GN method, is based, in a sufficiently rigorous way, on minimising the error.

very slow but certain convergence—that is, convergence to a point of minimum error (this minimum error being being zero in the case of a pseudoimage point). Conversely (Aster *et al*., 2005), very small values of λ (essentially the GN method) produce uncertain, but *potentially* much faster, convergence. It is worth noting that the GN and steepest-descent methods produce steps in two different directions, and that the LM step points somewhere between the two.

In setting the value of λ, a 'simple approach' recommended by Aster *et al*. was followed. One starts with a small value of λ (the choice made is 0.1 times the term of maximum absolute value in the determinant of $\mathbf{J}^T\mathbf{J}$). If the LM step reduces the error, the step is taken and λ is halved for the next iteration. If not, that step is not taken. Instead, λ is repeatedly doubled (each doubling leads to a new 'internal iteration') until the error is reduced, at which stage the resulting step is taken and the algorithm proceeds to the next main iteration. (Following the series of doublings, no further change is made in λ in preparation for the next main iteration.) Thus the advantages of both the GN and the steepest-descent methods are obtained.

7.4 Blockage Points

During the running of imige, for each tentative point $\mathbf{x}$ that is generated, imige calls the subroutine ray. After a number of iterations of imige, the point $\mathbf{x}$ may reach a boundary curve beyond which ray always fails (*forward* failure), that is, for these points $\mathbf{x}$ there is no corresponding object point $\mathbf{u}$. Such a boundary curve will be called a *blockage perimeter* and the point reached on that perimeter will be called a *blockage point*; further 'progress' of the point is 'blocked.'[28] The simplest case of a blockage perimeter is a *cutoff perimeter*—the intersection of the cutoff plane (see Section 6.2) with the reflecting surface. If the default cutoff is used, this case is also the most common case of a blockage perimeter; indeed the majority of the blockage perimeters presented in this report are cutoff perimeters. Other types of blockage perimeter occur (i) when the reflected ray becomes almost parallel to the array plane (see '1/30 radian' in Section 6.3.2), and (ii) when the reflected ray begins to be reflected a second time before it meets the array.

A difference between a local optimum and a blockage point is worth noting. Given a particular initial guess $\mathbf{x}_{00}$ that leads to a particular local optimum point P, there is a region surrounding $\mathbf{x}_{00}$ such that any guess $\mathbf{x}_0$ in the region leads to the same local optimum point P. The same is not true for a blockage point: varying $\mathbf{x}_0$ normally leads to a variation in the blockage point. This difference is borne out by numerical results (table output by the program multigen described below).

7.5 Estimates of Derivatives

The nth iteration begins with the input $\mathbf{x}_{n-1} = (x_{n-1}, y_{n-1})$ that was produced by the previous iteration. The derivatives ($\partial u/\partial x$, $\partial v/\partial x$, $\partial u/\partial y$ and $\partial v/\partial y$) at $\mathbf{x}_{n-1}$ are required. The exact

[28] The reported position of a blockage point may (and usually does) depart *very slightly* from the true blockage point (and be very slightly away from the blockage perimeter) because of a failure at a point used to calculate the derivative (Section 7.5). The details are readily gleaned from Appendix B.

derivatives are, however, replaced by estimates using finite differences. The new values of x and y used for this purpose are x1, x2, y1 and y2, given by

$$(\mathrm{x1},\, \mathrm{x2}) = x_{n-1} \mp 5\times 10^{-5}|\mathbf{x}|$$

$$(\mathrm{y1},\, \mathrm{y2}) = y_{n-1} \mp 5\times 10^{-5}|\mathbf{x}|$$

Then a two-sided estimate of each derivative is used, of the form

$$\partial u/\partial x = [u(x+h) - u(x-h)]/2h$$

(two-sided for added accuracy).

7.6 End Points: Discussion

At the end of each internal iteration, the output value of the error and the size of the step taken are tested to determine whether 'convergence' has occurred to a pseudoimage point (PI), a blockage point (BP) or a local optimum (LO); details are given in Appendix B. In any such case the algorithm is terminated.

The criteria for declaring a PI (Appendix B) are straightforward and need no checking. The criteria for declaring a BP or a LO were checked as follows. A number of points declared to be BP were investigated and each was found to lie accurately on a cutoff perimeter. (Thus the criteria used for a BP were confirmed to be valid.) To investigate LO, a program, varerr.m, was written to calculate, for the given value of $\mathbf{u}'$, the error $|\mathbf{f}(\mathbf{x}) - \mathbf{u}'|$ for a 2D grid of points $\mathbf{x}$ in the neighbourhood of the declared LO, $\mathbf{x}_{\mathrm{LO}}$. It was found, for the several such declared LO studied, that the error was indeed minimised at $\mathbf{x}_{\mathrm{LO}}$.

If, despite the tests for PI, BP and LO, the 61st main iteration is commenced, the program declares a maximum-iterations point (MI); the algorithm is terminated without convergence.

The subroutine imige always reaches an 'end point' $\mathbf{x}$—the value of $\mathbf{x}$ obtained in the last iteration that was successful in reducing the error—and this value of $\mathbf{x}$ is output. It is believed that the only types of end point are PI, BP, LO and MI.[29] Table 7.1 summarises the possible types of end point together with the corresponding 'ending code' (90 to 94) used by the suite of programs to report each type. (Code 94 represents a new type of end point that could, *a priori*, occur.) We shall refer to the last two types (93 and 94) as *maximum-iterations points* (93) and 'rare' end points (94) respectively, and the two types collectively as *points of nonconvergence*. From experience, 'rare' end points (94) seem never to occur. Thus in practice it seems that the two italicised terms can be treated as identical.

In the case of any maximum-iterations end point, it is believed that there is an *underlying* end point that is either a PI, a BP or a LO. That is, a superior algorithm would eventually report convergence to one of these types. The underlying types of points (namely BP and LO) that are not pseudoimage points will be called *ghost points*.[30]

[29] Provision is made for imige to report an ending that is other than these (ending code 94 in Table 7.1), but no such endings have been reported.

[30] We use 'ghost reported points' to refer to end points (not *underlying* end points) of types 91 to 94.

Type of end point	Code	Marker
Pseudoimage point	90	plus (+)
Blockage point	91	diamond
Local optimum	92	circle
Maximum-iterations point: maximum main iterations (61) reached. Usually one can identify the underlying end point as 90, 91 or 92 by examining nearby input points	93	[square]
'Rare' end point: other type of ending to imige. Occurs rarely or never	94	[asterisk]

Table 7.1. Showing, for each type of end point reached by the subroutine imige*, the ending code, and the corresponding marker used in the graphical output of* multisym *and succeeding programs. The code is given in a table that is output by the relevant program. Square brackets indicate that the corresponding marker is often suppressed.*

Fortunately maximum-iterations end points occur considerably less often than blockage points or local optima. For almost every MI point, it is clear what the underlying type of end point is (namely PI, BP or LO): the type is identified by noting the type that was actually declared for neighbouring input points.

A single running of imige, based as it is on a single initial guess $\mathbf{x}_0$, will always lead to one and only one end point $\mathbf{x}$. On the other hand, as noted in Section 6.3.1, to any input object point there may be as many as three corresponding pseudoimage points, the functional relationship being multivalued. *A fortiori*, there may be multiple end points. These two observations are reconciled as follows. The three (or two) end points can potentially all be obtained by inputting a number of initial guesses $\mathbf{x}_0$. A grid of such initial guesses is in fact used in the programs multisym and multigen (Sections 8.2 and 8.3 respectively)——programs that seek to find *all* the end points corresponding to a given object point $\mathbf{u}'$.

In the examples studied in the present work, it has been found that for any given object point $\mathbf{u}'$, the underlying end points always total one, two or three, provided one makes a special allowance for blockage points.[31] (The 'misbehaviour' of blockage points is discussed in Section 8.3.1.)

The inversion algorithm used contains some less important features that are omitted in the main text. Interested readers are referred to the code of imige in Appendix C.

8. Results for Pseudoimage Points

In this section we present results for the pseudoimage position—or more generally the end point—as a function of the object position. Three programs are used for this purpose: imsuite.m, multisym.m and multigen.m. These are discussed in turn along with the results obtained from each. A closely related program quad.m is discussed in Section 9.

[31] For blockage points, the statement becomes true if we employ the fiction that end points lying on the same blockage perimeter are identical end points. In actuality, in the case of blockage points, the end point depends on the initial guess (and would do so even if the method of steepest descents were used).

8.1 Program imsuite

The program imsuite, like rysuite, produces a suite of curves on one graph. Its use is confined to systems with considerable symmetry—the same symmetry as described at Equation (6.1). The remarks made about symmetry in Section 6.1 again apply.

Each curve is generated by taking, as inputs, object points $\mathrm{R}(u', v', 0)$ that lie along a line through the origin. The various curves describe pseudoimage points pertaining to the same line of object points R, but those curves are characterised by different values of the product $\kappa_X r_0$. The input points considered lie on the line $v' = 0$ and essentially x is plotted against u'. Due to the symmetry, however, the results also apply to any other radial line, not just $v' = 0$ (compare Section 6.1).

The relationship (6.2) is replaced by

$$\frac{x}{u'} = f\left(\kappa_X r_0, \frac{u'}{r_0}, \frac{e}{r_0}, \text{shape}, \frac{\text{Zb}}{r_0}\right) \tag{8.1}$$

For a given graph, the last three arguments of f are held constant. Each curve (specified by a value of $\kappa_X r_0$) is a plot of the magnification x/u' versus $\log_{10}(u'/r_0)$. Plotted on the vertical axis is the magnification of a vector OP, drawn in the array plane from the chief receiving point, as discussed in Section 6.1. (In some graphs, x/r_0 is plotted instead of x/u', as will be discussed in Section 8.1.1.) Because scaled variables are used, the graph represents many more situations than might at first be thought. From Section 5, as $u'/r_0 \to 0$ (left-hand end of graph), for each curve the magnification must approach the value predicted by the paraxial approximation. This prediction is readily seen to hold in the sample graphs, Figures 8.1 and 8.2.

As discussed in Section 7, the running of imige (called by imsuite) with different initial guesses $\mathbf{x}_0 = (x_0, y_0)$ can lead to more than one pseudoimage point, and can also lead to ghost points. imsuite is designed with the goal of producing just one output, and the output aimed at is the one that lies on the *quasiparaxial* branch (of the pseudoimage relationship). By this we mean the continuous curve of pseudoimage points that reduces to the paraxial solution as a limiting case. In other words, the program seeks the *continuation of the paraxial solution* into the region where the paraxial conditions do not hold. To attain the goal, the initial guess (x_0, y_0) used in imsuite is always the prediction of the paraxial approximation.

Consider any of the above curves: it is a curve traced out as u' increases. (Here and elsewhere in Section 8.1, for convenience we consider r_0 as *fixed*. Indeed the relevant points can be made by considering $r_0 = 1$.) In the cases considered in our simulations, it turns out that, as u' rises out of the paraxial region, imsuite always traces out the *quasiparaxial branch*— to indefinitely large values of u', or, if the quasiparaxial branch terminates, to the termination point. (In principle, the output curve could have jumped to a different pseudoimage branch.) Beyond the termination point the output follows a ghost curve that joins continuously with the quasiparaxial branch.

In Section 7 two types of ghost point (namely, local optimum and blockage point) were identified. For each output ghost point, imsuite identifies the type in a table that is output, but does not show the type on the graph.

8.1.1 Results

Figure 8.1 shows the graph that is output from imsuite for a paraboloidal reflecting surface, with zero offset e and the default cutoff, using selected values of $\kappa_X r_0$. Figure 8.2 shows the corresponding graph for a sphere. (The latter graph applies also to a cylinder, as in Section 6.)

We discuss special cases of the parameter $\kappa_X r_0$. For $\kappa_X r_0 = 0$, the reflecting surface is a plane and the magnification is exactly 0.5, in accord with both the graphs. For the paraboloid with $\kappa_X r_0 = -0.5$, the magnification is exactly one, as in Section 6; for the sphere, the magnification approaches one at low u'. For $\kappa_X r_0 = -1$ (in the context where $e = 0$), it will be recalled that the paraxial approximation predicts an infinite magnification. The upshot is that, at least for u' in some neighbourhood of $u' = 0$, as $\kappa_X r_0 \to -1$ (whether from above or below), the magnification $|x/u'|$ becomes very large, a prediction consistent with each of the

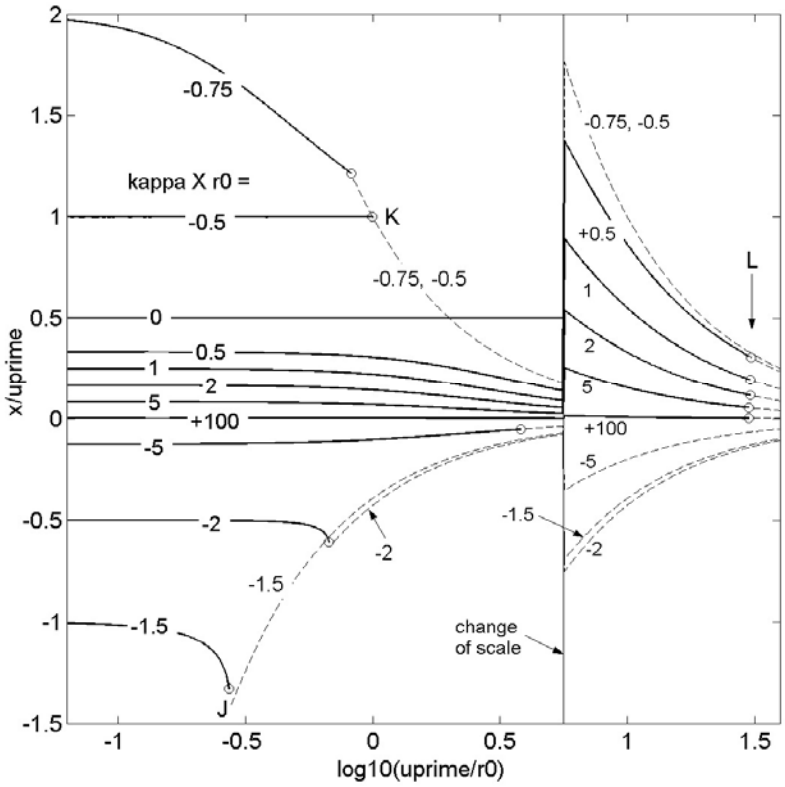

Figure 8.1. Position of pseudoimage points (solid curve) and ghost points (dashed), for a paraboloid. Plotted as magnification against scaled object position, for the values of $\kappa_X r_0$ *marked. Obtained via program* imsuite*. Note the change of vertical scale by a factor of ten at* $\log_{10}(u'/r_0) = 0.75$*. For* $\kappa_X r_0 = 0$ *(plane reflector), the pseudoimage curve (a horizontal line) continues indefinitely far to the right provided the '*$1/30$ *radian' criterion is not invoked. There is a point* J *(approximate position marked) at which the solid curve and the dashed curve actually meet; the graph does not show this because of the nonzero sampling interval used. (The solid curve should continue beyond the circle.) To the right of the point* K*, two dashed curves coincide (see text). The curve for* $\kappa_X r_0 = +100$ *lies barely above the line (not drawn)* $x = 0$.

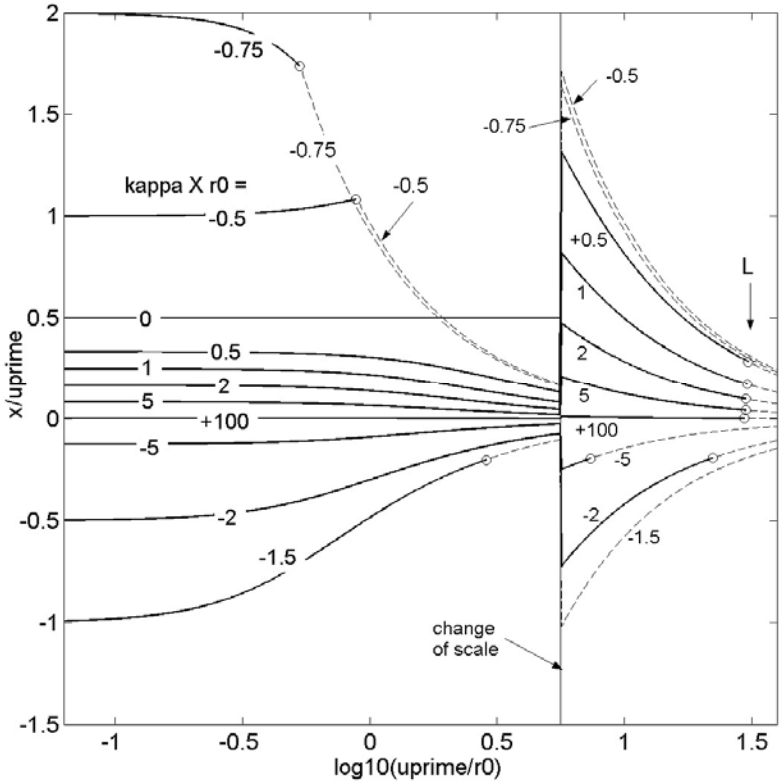

Figure 8.2. As for Figure 8.1, but for a sphere or cylinder.

two graphs. Finally, the curve $\kappa_X r_0 = 100$ represents a case near the limit where $\kappa_X r_0$ approaches infinity. Then the surface approaches a point reflector (at least in the case of a sphere) and x is forced to approach zero (for all values of u'), in accord with each of the graphs. For $\kappa_X r_0$ approaching *minus* infinity (*i.e.* the inside of a hemispherical surface in the case of a sphere), again x must approach zero.[32]

The differences between the curves for the paraboloid and those for a sphere can be accounted for qualitatively via simple diagrams (not presented here). Consider for example the case $-1 < \kappa_X r_0 < 0$. Because the spherical surface is steeper than the paraboloid's surface at a given positive x, the resulting object point displacement u' is less for the sphere. It follows that, *for a given* u', x is greater for the sphere than for the paraboloid. This prediction is borne out by Figures 8.1 and 8.2. A similar argument applies for each of the cases $-\infty < \kappa_X r_0 < -1$ and $\kappa_X r_0 > 0$.

An alternative way, or mode, of plotting the results of Figures 8.1 and 8.2 is more useful for some purposes; the re-plotting produces Figures 8.3 and 8.4 respectively. In this alternative mode, the quantity plotted along the vertical axis is x/r_0. When r_0 is considered fixed, this means that essentially the displacement x, rather than the magnification, is plotted. A

[32] Some of the statements in this paragraph need modification for some choices of the cutoff plane, in that blockage points may be produced for some values of u'.

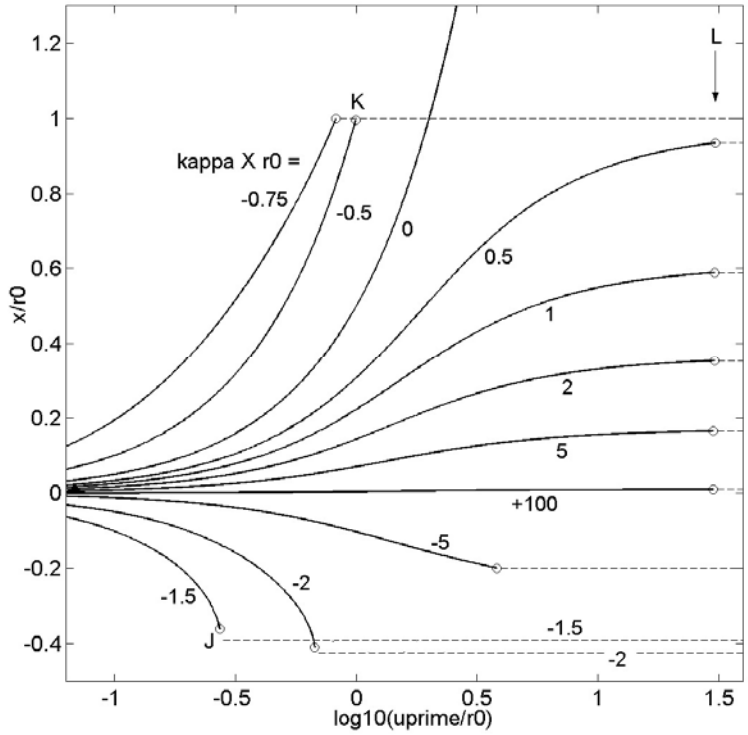

Figure 8.3. The results of Figure 8.1 (paraboloid) re-plotted with x/r_0—essentially the displacement—along the vertical axis. For $\kappa_X r_0 = 0$, the exponential curve continues indefinitely to the right.

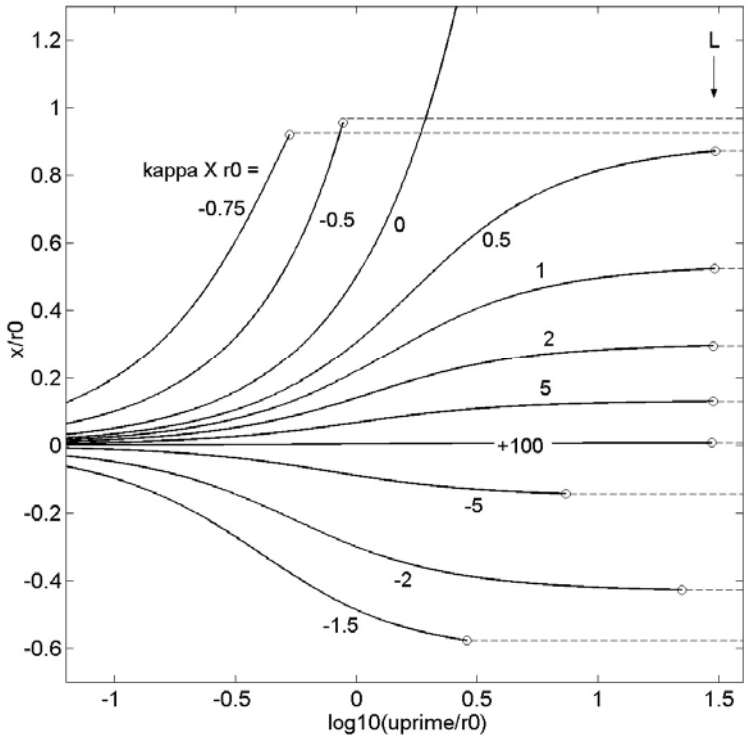

Figure 8.4. The results of Figure 8.2 (sphere) re-plotted with x/r_0—essentially the displacement—along the vertical axis.

disadvantage of the second mode is that a curve that is a constant (horizontal line) in the first mode is replaced by a curve with the exponential shape e^t—due to the horizontal axis being logarithmic. An advantage, however, is that a curve that is exponential (e^{-t}) (in Figures 8.1 and 8.2) becomes a constant (horizontal line). It will turn out that the latter pairing occurs more often, giving an overall advantage to the second mode.

Meanwhile, however, we discuss termination points and ghost curves. Figures 8.1 and 8.2 together (or alternatively, Figs. 8.3 and 8.4) show ten termination points (indicated by circles) (putting aside for now the termination points bunched together in a vertical line labelled by L). A table output by imsuite (not reproduced here) reveals that, beyond the termination point (that is, along the dashed curves), the end-point type is a local optimum in two cases (paraboloid, $\kappa_X r_0 = -2$ and -1.5) and is a blockage point in the other eight cases. Of the eight cases of blockage, six arise from cutoffs. The remaining two cases are covered (as items i and ii) in what follows. Consider, for the sphere, the curves for $\kappa_X r_0 = -5$, -2 and -1.5. Perhaps surprisingly, their termination points arise from three different causes. The respective causes are: (i) the reflected ray meeting the reflecting surface for a second time; (ii) the reflected ray travelling away from the array or coming within about $1/30$ radian of being parallel to it; and (iii) the end point reaching the cutoff plane.

In more detail, consider first the local optimum curves (certain dashed curves in Figs. 8.1, 8.3). Because the case $\kappa_X r_0 = -1.5$ differs little from the case $\kappa_X r_0 = -1.375$ discussed in Section 6.3.1, with the aid of Figure 6.1 it is clear what is happening. Consider what happens at fixed r_0. As x becomes more negative, u' (positive) at first increases (quasiparaxial branch) and then decreases (non-quasiparaxial branch). The larger positive values of u' are never reached. Let u'_t and x_t be the values of u' and x at the turning point. For $u' > u'_t$, there is no corresponding value of x (near x_t) that represents a pseudoimage point, but at $x = x_t$ the error in u' is minimised. Hence, at the point J in Figures 8.1 and 8.3 there is a transition from the quasiparaxial branch to a curve of local optima.

Two further comments regarding the local optima are worth making. First, consider the graph of a function in the neighbourhood of a turning point, as in Figure 6.1. When we invert the function, the resulting curve should have an infinite slope at the termination point (and should asymptotically be a parabola near that point). The two relevant curves in each of Figures 8.1 and 8.3 are consistent with having such an infinite slope.

Second, for all u' along a given curve of local optima, x should be equal to the constant value x_t. Calculations from the output table show that this is the case. (The table—not shown—gives values of x with high precision.) The constancy is seen in Figure 8.3.

Consider now the blockage point curves (Figs. 8.1 and 8.2, or alternatively 8.3 and 8.4). Again consider r_0 fixed. Along each of the eight curves (identified above and again in Table 8.1), it is found that x has a constant value (Figs. 8.3, 8.4). This claim has been verified to high accuracy from imsuite's output table; the results are shown in Table 8.1. For the six curves arising from cutoffs, a further check can be made by calculating the cutoff value of x/r_0 from Equation (6.3) together with (3.2) and (3.3). Table 8.1 shows agreement in this test also. For the paraboloid (Figs. 8.1, 8.3), to the right of the point K, two blockage point curves coincide. In fact, there is a range of values of $\kappa_X r_0$ over which the corresponding curves

coincide: the x values are the same though the z values differ. This is a property that depends on the default cutoff and has no further significance.

For the positive values of $\kappa_X r_0$, (at constant r_0) the pseudoimage curves persist to large values of u' (Figs. 8.1 to 8.4). Furthermore, as u'/r_0 approaches infinity, (for each positive value) x approaches a constant value, equal to the displacement x that produces a reflected ray parallel to the array. [Here we are speaking of the pseudoimage curves (solid), not the ghost (dashed) lines. Thus, in Figs. 8.3 and 8.4, these pseudoimage curves asymptote to a horizontal line.] Note that towards the right-hand end of Figures 8.1 and 8.2, it is the magnification x/u' that approaches zero, not x itself.

As u'/r_0 approaches infinity, eventually the angle of departure from parallelism drops below $1/30$ radian (Section 6.3.2) and the program reports termination of the pseudoimage curve. This produces the bunch of termination points at L on the extreme right; it is no accident that these occur around $u'/r_0 = 30$.

shape	$\kappa_X r_0$	cause of blockage	x/r_0	max. error parts in 10^4
parab	-5	cutoff	-0.20000	0.5
parab	-0.75	cutoff	$+1.00000$	1
parab	-0.5	cutoff	$+1.00000$	1
sphere	-5	(i)	-0.14328	3
sphere	-2	(ii)	-0.42728	3
sphere	-1.5	cutoff	-0.57735	0.6
sphere	-0.75	cutoff	$+0.92702$	1.8
sphere	-0.5	cutoff	$+0.96825$	1.5

Table 8.1. Check on the constancy of x/r_0—and hence the constancy of x at fixed r_0—along each of the eight blockage point curves in Figures 8.1 and 8.2. For each curve (row), values of x/r_0 were extracted from imsuite*'s output table for two input values of u'/r_0 as far apart as possible. These x/r_0 values were compared with each other and, in the 'cutoff' cases, with the known cutoff value. The values of x/r_0 quoted are the known cutoff value in the 'cutoff' cases, and the average of the two simulation values in the other two cases. The maximum error is the maximum departure from the known cutoff value ('cutoff' cases) and the departure of one simulation value from the other ('non-cutoff' cases). Significantly,* all the errors are within the roundoff error *of* imsuite*'s table. For the notations* '(i)' *and* '(ii)' *see the main text.*

8.2 Program multisym

The program multisym again requires a symmetric system. Unlike imsuite, it attempts to find *all* the end points resulting from each given object point ('multiple' end points). To avoid the graph's having too cluttered an appearance, a given graph deals with only one value of $\kappa_X r_0$ rather than a 'suite' of values. To find *all* the end points, many initial guesses $(x_0, y_0) = (x_0, 0)$ are used. The positive values of x_0 are distributed over the interval from

$10^{-2} x_p$ to $10^{+2.5} x_p$—where x_p is the paraxial prediction—with the logarithm of x_0 distributed uniformly. The corresponding values of x_0 having the opposite sign are also used.

A typical graph is shown in Figure 8.5. Each combination of an object point and an end point is shown by a marker. A different marker is used for each type of end point, as shown in Table 7.1. In general a smooth curve—a *branch* of the output—can be drawn joining markers of the same type. This has not been done, as a complex algorithm would be required to do it automatically. However, in each case the position of the curve is clear to the eye. Note that, in the graphs produced by multisym (Figs. 8.5 to 8.8), the second mode of plotting has been followed (*i.e.* x/r_0 along the vertical axis).

8.2.1 Results

Figure 8.5 shows the output from multisym for a paraboloid for the value $\kappa_X r_0 = -1.25$, with zero offset.[33] We are already familiar with the branches AL and LB: they are a repetition of the two curves (pseudoimage and local optimum) for $\kappa_X r_0 = -1.5$ in Figure 8.3, but for a slightly different value of $\kappa_X r_0$. For the smaller values of u' (left end), a second and a third pseudoimage branch are now present, in addition to the quasiparaxial branch AL.

The behaviour of the three pseudoimage branches is explained by referring to Figure 6.1. (Though the two values of $\kappa_X r_0$ are slightly different, the qualitative features are the same.) For u' slightly above zero, from Figure 6.1 clearly there are three corresponding values of x. Furthermore, of the two extreme values of x, one is approximately the negative of the other, in accord with Figure 8.5. As u' is raised, from Figure 6.1 eventually the lowest two values of x coalesce and then disappear. Figure 8.5 shows that upon disappearing, the two pseudoimage branches are replaced by a single curve (LB) of local optima, as anticipated in Section 7.2.

The behaviour of the curves near the point of disappearance, or *critical point*, L, is shown in Figure 8.6. That graph confirms that the two relevant pseudoimage curves form a *single* smooth curve having a vertical tangent at the critical point, as anticipated in Section 8.1.1.

In Figure 8.5, the upper pseudoimage branch gives way, at the point M, to a blockage point curve as u' increases. Along the latter curve, x/r_0 is constant at 0.800. It is known that the blockage point corresponds to (x, y) reaching the *cutoff* perimeter, because, from Equations (6.3) and (3.2), 0.800 is the value of x at the cutoff (the value of Z at the default cutoff being 0.400).

Not only in Figure 8.5, but in all the similar graphs of systems with circular symmetry (Figures 8.7, 8.8, and the previously-discussed Figures 8.3 and 8.4), each local optimum curve and each blockage point curve is characterised by x being a constant.[34] Furthermore

[33] For added clarity, points of nonconvergence have been suppressed in the graphs; similarly in the graphs produced by multigen and quad.

[34] The constancy of x may come as a surprise in the case of the blockage point curve, in view of the discussion in Section 7.4. However, the constancy comes about because in the present case the system

(Fig. 8.5), along each nonparaxial pseudoimage branch, as u'/r_0 approaches zero, x approaches a constant other than zero (as expected from Figure 6.1).

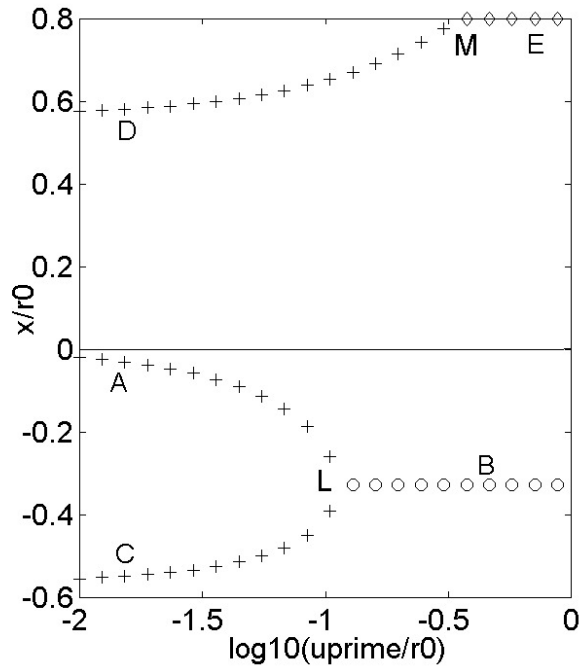

Figure 8.5. Position x (strictly, x/r_0) of pseudoimage points and other end points, for a paraboloid; $\kappa_x r_0 = -1.25$. Obtained by program multisym. *For the code for markers, see Table 7.1.*

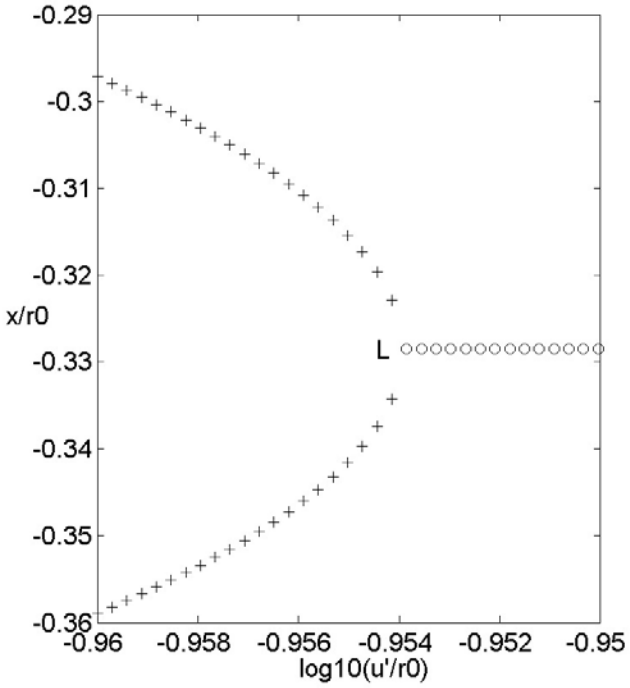

Figure 8.6. Expanded view of Figure 8.5 near the critical point L.

---

is circularly symmetric: the path followed by (x, y) during the iterations is confined to the line $y = 0$. This point is discussed further in Section 8.3.1.

When $\kappa_X r_0$ is changed from -1.25 to -1.5, the resulting graph (not displayed) is qualitatively the same as Figure 8.5, except that now the entire uppermost pseudoimage branch is replaced by a curve of blockage points. At $\kappa_X r_0 = -1.75$, the graph obtained is shown in Figure 8.7. Now most, but not all, of the lowest pseudoimage branch is also replaced by a curve of blockage points.[35]

It is found that the graphs discussed so far capture the qualitative behaviour exhibited for parameter values throughout the interval $-3 < \kappa_X r_0 < -1$. The graphs obtained outside this interval are less interesting than those obtained in this interval. Outside the interval the number of branches becomes less; often there is only a single pseudoimage branch joined end-to-end to a single ghost branch; sometimes there is no ghost branch and the single pseudoimage branch extends from one end of the graph to the other.

The inversion algorithm does not always behave ideally. First, as the initial guess x_0 increases (the input object point u' remaining constant), the end point found often jumps back and forth between one branch and another (*e.g.* between a local optimum and a blockage point curve), thus showing a kind of instability. This is due to the nonzero size of the step that (x, y) takes in the iterative procedure, causing (x, y) to follow a somewhat erratic path. The phenomenon causes no problem in the graphs.

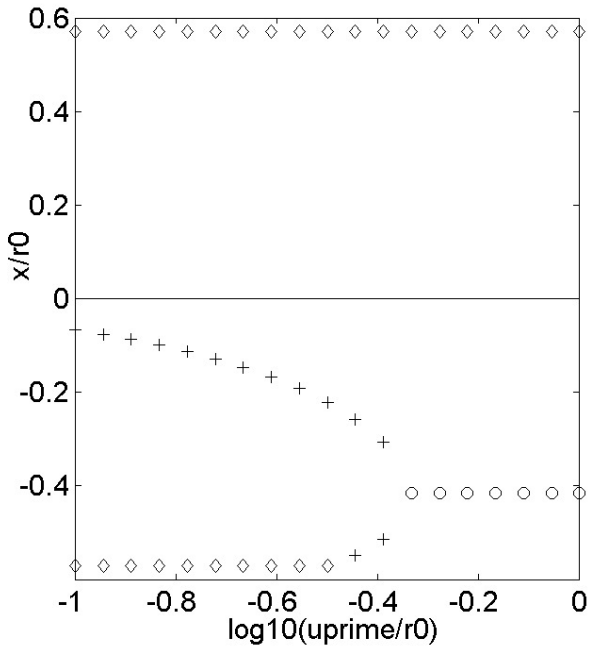

Figure 8.7. As for Figure 8.5, but for $\kappa_X r_0 = -1.75$.

[35] As a check, it is worthwhile to compare the curves in Figure 8.7 (Similar remarks apply to Figure 8.8.) with the corresponding curves obtained by interpolating by eye in Figure 8.3 to $\kappa_X r_0 = -1.75$. For the quasiparaxial branch, the branch of local optima and the critical point, there is seen to be at least semiquantitative agreement.

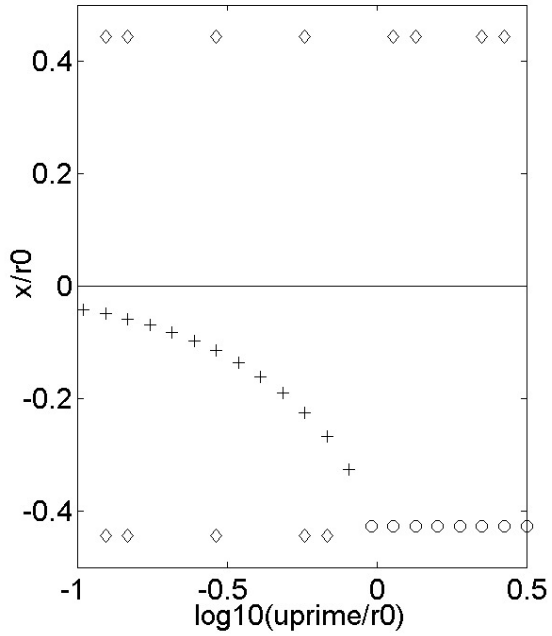

Figure 8.8. As for Figure 8.5, but for $\kappa_X r_0 = -2.25$.

Of somewhat more concern is the behaviour illustrated in Figure 8.8, which is the output for $\kappa_X r_0 = -2.25$. Clearly, in both the upper and lower curves of blockage points, some end points are missing; that is, there are underlying end points that multisym failed to find.

When the underlying end points are pseudoimage points, a happier outcome is obtained. Despite the extensive use to which multisym has been put, no evidence has been found of a missing pseudoimage point, suggesting that such 'misses' do not occur for systems with circular symmetry. Misses for systems *without* such symmetry are discussed in Section 8.3.1.

We return, for systems with circular symmetry, to end points of other types: we note that for blockage points, misses are not infrequent. I am fairly confident that, because u' is varied during the construction of one graph, at least all the *branches* are found, except in rare cases. But there is no obvious way of confirming this; we can only say that there appears to be no reason why there should be more branches than those found.

8.3 Program multigen

The program multigen, like multisym, attempts to produce all the end points (x, y) for each input object point (u', v'). As in multisym, many initial guesses (x_0, y_0) are used; these are spread along a radial line through the point (x_p, y_p) predicted by the paraxial approximation. But multigen differs from multisym in four ways. First, there is no requirement that the system be symmetric; the parameters, including the angles α and δ, may take on any values (hence 'gen' for general in the name). Second, the input points (u', v') may lie along *any* line segment in the $u'v'$ plane. In more detail, the user specifies a 2D vector $(\text{up1}, \text{vp1})$ pointing along the

line segment, and p, the perpendicular offset of the line from the origin.[36] A parameter, dist, is defined as the signed distance along the line, from the offset point, measured in the direction of $(\text{up1}, \text{vp1})$.

Third, no scaling is applied before plotting: the two axes simply represent two of the following three variables: dist, x and y. Fourth, three graphs, not one, are output, as will now be discussed. For each value of dist, the program generates a number of end point vectors (x, y). Each such end point is represented by a marker in each of three graphs: (i) x versus dist, (ii) y versus dist, and (iii) y versus x. These will be called respectively the 'x graph,' the 'y graph' and the 'xy graph.' The meaning of each marker is given in Table 7.1 as before. Clearly the xy graph is a scale drawing of the pseudoimage of the object line. We shall see that the xy graph reveals patterns that are not revealed by the first two graphs.

8.3.1 Results

As an example, multigen was run with the parameter set multigen 1 shown in Table 8.2. (The last seven parameters, I to ht, are matters of detail; they are included so that, by rerunning the program, the reader could reproduce the graphs exactly.[37]) The resulting three graphs are shown in Figures 8.9 to 8.11, to be discussed shortly (beginning with the paragraph after next).

parameter	**value**	**parameter**	**value**	**parameter**	**value**
shape	paraboloid	κ_X	-0.375	N	38
Zb	default	κ_Y	-0.75	showgh	1
r_0	2	up1	0.4	tini	0.75
e	0.1	vp1	0.6	tfin	1.35
δ	18°	p	0	hb	-300
α	32°	I	52	ht	300

Table 8.2. Parameter set multigen 1.

Note (from Table 8.2) that none of the parameters has a special value, such as zero, with one exception: the fact that $p = 0$ means that the line of input points passes through the origin, *i.e.* the chief receiving point. For generality, it is desirable to also treat one or more cases in which $p \neq 0$. In fact, cases of p not equal to zero have been treated, as described in two places below. First, in this Section 8.3.1, following the discussion of the $p = 0$ case, we describe the results obtained when we change the value of p to one while leaving the other parameters in Table 8.2 unchanged. Second, in Section 9, pseudoimages of many squares are computed. For each square, two of the sides have $p \neq 0$; hence that section gives a comprehensive treatment of line segments for which $p \neq 0$.

In each of Figures 8.9 to 8.11, the end points form four branches or clusters: two pseudoimage branches (PI1 and PI2), a local optimum branch (LO); and a blockage point

[36] There is a sign convention associated with p.

[37] tini and tfin specify essentially the initial and final values of dist: for details see the relevant program in Appendix C.

cluster (BP). As in the earlier, symmetric systems, the two pseudoimage curves meet at a point (the critical point) from which a local optimum branch emanates.

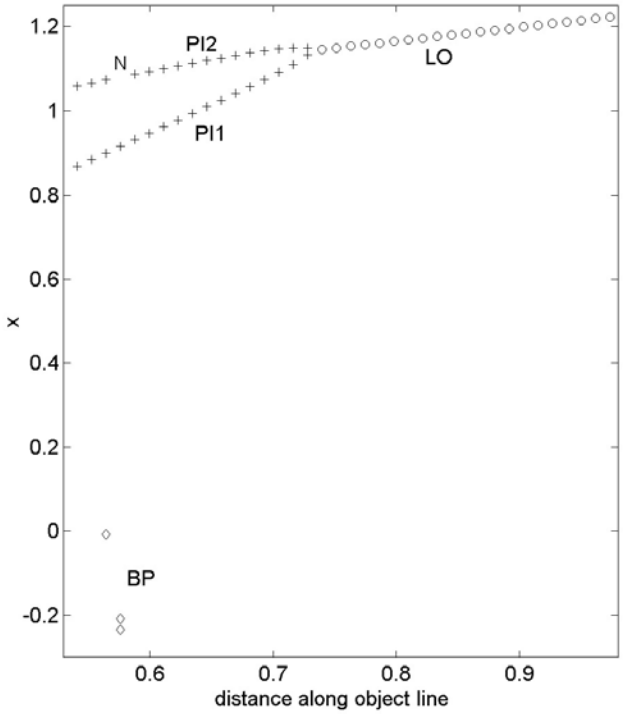

Figure 8.9. x component of end point positions, plotted against object position, for the parameter set multigen 1. *Obtained by program* multigen. *The labels such as* PI1 *are explained in the text.*

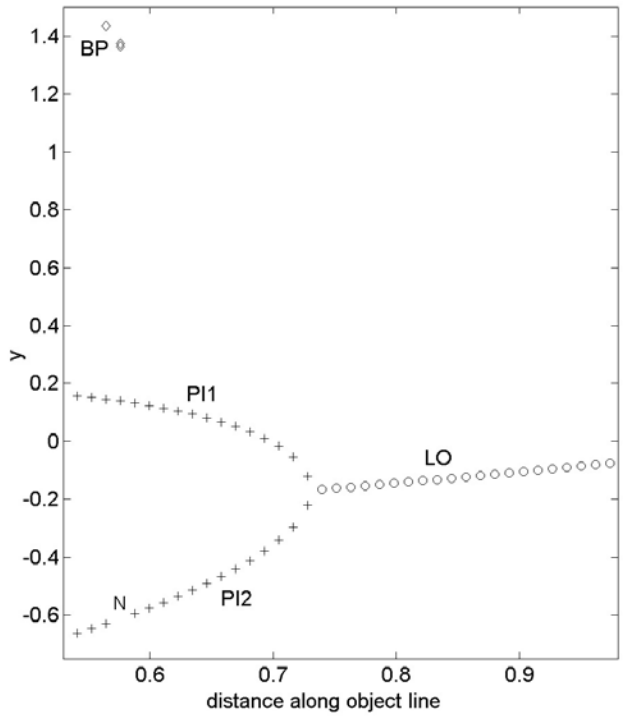

Figure 8.10. As for Figure 8.9, but y component.

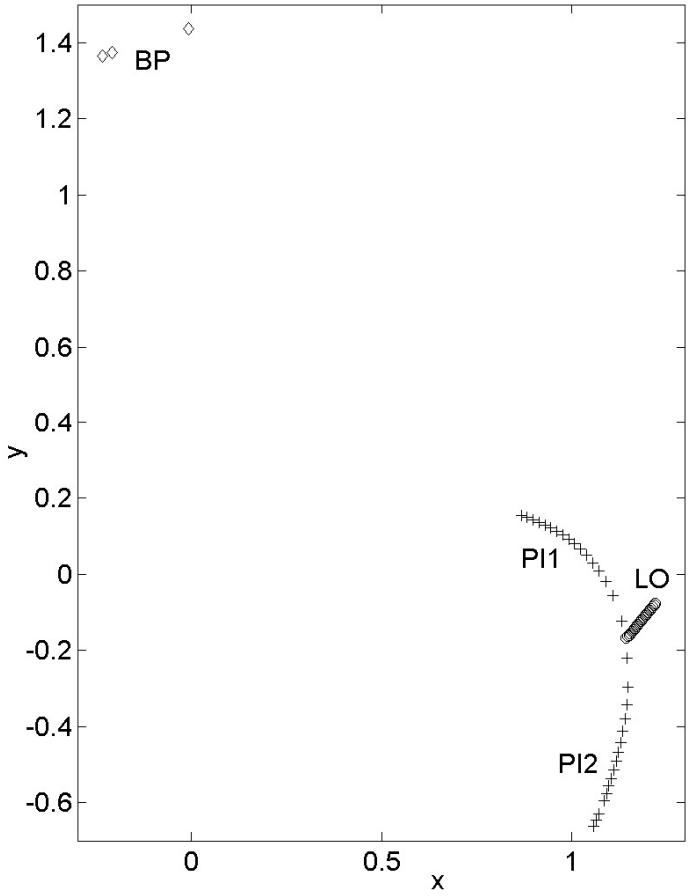

Figure 8.11. Plot of the vector positions (x, y) *of the end points obtained for parameter set* multigen 1. *Hence the curves* PI1 *and* PI2 *together give a scale drawing of the pseudoimage of the object line.*

We turn to blockage points. Because there are only three of them in Figures 8.9 to 8.11, conclusions can hardly be drawn from that avenue. However, when there are many such points, we expect that, *in the xy graph*, the points lie accurately on a single curve, the blockage perimeter. Consider now the separate x and y graphs. Recall, from the discussion of instability in Section 8.2, that, during the iterations, the point (x, y) is thought to follow a somewhat erratic path. Furthermore, because the movement is no longer confined to the line $y = 0$ (circularly symmetric case), but can spread in two dimensions, a change in the initial guess in general leads to a different blockage point (on the same blockage perimeter). Then, on the x and y graphs (graph of x versus dist, for example), the blockage points should not lie on a single curve, but should be scattered.

To test this prediction, multigen was run with the same parameters as in Table 8.2 but with $p = 1$. In the x and y graphs, the very many (about 325) blockage points were found to lie along four roughly parallel curves but with scatter about each of the four: quite small scatter about three curves and very considerable scatter about the fourth. However, in the xy graph, a *single* precise curve embraced all the blockage points, as predicted. Incidentally, these graphs illustrate a further feature of the 'instability' at a blockage point in nonsymmetric cases, as follows. For a given object point—but many initial guesses—the number of output values can be many more than the 'one to three' previously encountered. This happens only with blockage points.

We return to a point made at the end of Section 7.6. Apart from the 'misbehaviour' of blockage points, the plotted points always lie on a small number of smooth curves. In a certain sense, the maximum number of curves encountered in the work is three.[38] In the xy graph, the maximum number is three in a stronger sense.[39]

In regard to pseudoimage points and local optima, it is reasonable conclude that the LM method, as implemented in imige, is good at finding them. Any point claimed to be a pseudoimage point or a local optimum is indeed a point of the type claimed. However, occasionally a 'miss' occurs, that is, no point is 'claimed' even though a pseudoimage point or a local optimum exists. (The iterative process ends with a point of nonconvergence.) For nonsymmetric systems (Symmetric systems may behave still better.), the evidence based on a small number of runs with multigen is that a 'miss' occurs, on average, about once on each graph. Indeed, in the run that produced Figures 8.9 to 8.11, there is exactly one miss (shown by the label N in Figs. 8.9 and 8.10). In regard to *blockage points*, those claimed are indeed blockage points, but the frequency of misses is quite appreciable.

9. Distortion in Two Dimensions

9.1 Program quad

Consider a square on the surface of the array, with one vertex anchored at the $u'v'$ origin (the chief receiving point). When the square is sufficiently small, the paraxial approximation applies and so (Section 5.2) the pseudoimage of the square is a parallelogram. As the square is increased in size, we expect the four sides of the pseudoimage to become increasingly curved, yielding what might be called a *curved quadrilateral*, or *c-quad*—as noted in Section 2. Eventually portions of the curved boundary may be replaced by ghost curves. These general features are shown in Figures 9.1 to 9.4 (to be discussed below).

The program quad.m is designed to display this development. A 'standard square' on the array is specified by a vector $(\text{up0}, \text{vp0})$ along one side (called the 'standard side' and also labelled as side 1), extending from the $u'v'$ origin and ending in the 'standard vertex.' (The square is then fixed by requiring that a second side, called side 2, is obtained by rotating the standard side about the origin by 90° *anticlockwise* in the $u'v'$ plane. The other side adjacent to side 1 is called side 3.) A further $K-1$ squares (to make a total of K squares) are specified by applying linear scaling factors f_k to the standard square, while holding fixed both the vertex at the origin and also the square's orientation. In the present examples the size of the square increases with k (k, which runs from 1 to K, labels the square.), and the standard square, the Kth, is the largest, so that $0 < f_1 < f_2 < \ldots < f_K = 1$.

As with imsuite (Section 8.1), for each input point the program attempts to find the pseudoimage point on the quasiparaxial branch only; this is done by using the paraxial prediction as the initial guess (x_0, y_0) to be input into imige. It is expected that usually, (for

[38] This number is obtained if each blockage perimeter is artificially construed as contributing just one curve, and one restricts attention to a sufficiently small interval of dist.
[39] Here one restricts attention to a sufficiently small interval of dist and does not 'artificially construe' anything.

each input point) when an underlying pseudoimage point on the paraxial branch exists, it is reported as the end point. If such a point does not exist, either a ghost reported point or a pseudoimage point on a different branch must be reported; the experience with imsuite suggests that most often a ghost point will be found. The resulting end points give K attempted pseudoimages of squares. The program plots these K attempted pseudoimages, along with the paraxial prediction for each of these K figures. Pseudoimages are plotted as solid curves; optionally the ghost curves may be plotted (see Section 9.2 for details). The paraxial prediction for each figure is plotted as a dash-dot parallelogram.

For the standard square, each side is labelled with a value of si (the side number) from 1 to 4, as specified at the start of Section 9.1; similarly for each input square obtained from the standard square by scaling. Each side of a c-quad inherits a value of si from the input side of which it is the pseudoimage (see *e.g.* Fig. 9.1).

Markers are placed along some of the sides as follows. A number M is specified by the user such that, if the markers were to be placed along the side of the *input* square (instead of the output), they would subdivide the side into M equal intervals. Details of the markers are given in Table 9.1.[40] The program quad also outputs a table giving the type of end point, using the same code as in Table 7.1.

Which square	Type of point, or which side	Marker
standard or K th square	pseudoimage	plus (+)
	ghost	upward-pointing triangle (Δ)
$(K-1)$ th square	pseudoimage	cross ($\times$)
	ghost	right-pointing triangle
paraxial prediction for standard square	standard side	hexagram (six-pointed star) (large)
	other three sides	pentagram (five-pointed star)

Table 9.1. Details of markers output by program quad.

9.2 Results from Program quad

Examples of results obtained with quad are shown in Figures 9.1 to 9.4. The respective parameter sets, Q1 to Q4, are given in Table 9.2. Note that none of the parameters has a special value such as zero.

For parameter set Q1, Figure 9.1 shows that the relative deviation of the pseudoimage from the parallelogram shape becomes greater as the size of the object square increases. Consider *very* small squares (smaller than the smallest square input into Figure 9.1), such that the paraxial approximation is very good. Then (from Taylor series considerations) we expect that the *absolute* deviation of each vertex from the paraxial prediction is proportional to the square of the side of the object square. This expectation has been tested using the table output by quad (not shown here) and confirmed to high accuracy. Note from Table 9.2 that the offset e is not zero, so that not just the large-range approximation but the (more general) paraxial approximation has been tested and confirmed (confirmed because the *relative* deviation is found to approach zero).

[40] The markers, in particular the triangles, are always plotted, even if they represent points of nonconvergence.

Parameter set **Quantity**	**Q1**	**Q2**	**Q3**	**Q4**
Figure	9.1	9.2	9.3	9.4
shape	parab.	as for Q1	parab.	parab.
r_0	2		2	2
e	0.1		0.1	0.1
δ	18°		18°	18°
α	32°		32°	32°
κ_X	0.25		-0.36	-0.375
κ_Y	0.13		0.77	-0.75
Zb	default			
up0	1.6	6.4	2.4	0.4
vp0	2.4	9.6	-1.6	0.6
M	4	as for Q1	4	4
L	30		30	30
K	4		4	4
$\{f_k\}$	1/4, 2/4, 3/4, 1 (all parameter sets)			

Table 9.2. The parameter sets, Q1 to Q4, used as inputs to quad. Definition of L: The object points used for each square subdivide each side of the square uniformly into LM intervals.

As noted, the smallest square input into Figure 9.1 does not count as 'very small.' It is of mild interest that, nevertheless, the quadratic relationship is seen to hold approximately between all the four c-quads in respect of side 2 (but not side 1), the deviations being close to bearing the ratios $1:4:9:16$.

Note that in Figure 9.1, for the larger squares the sides are markedly curved. Also, some pairs of opposite sides depart markedly from being parallel.

Consider now parameter set Q2 (Figure 9.2). The parameter set is the same as for Figure 9.1 except that the input squares are larger (in linear terms) by a factor of 4. The distortion away from the paraxial parallelogram becomes marked to a much greater degree. In this case, consider the largest pseudoimage: the magnification along the diagonal extending from the origin becomes almost as low as one-third of the paraxial prediction. A qualitative comparison can be made with the $\kappa r_0 = +0.5$ curve in Figure 8.1 (since $\kappa_X r_0$ and $\kappa_Y r_0$ are both positive and their average is about 0.4). In both cases, the magnification decreases as the size of the object increases.

For the remaining parameter sets, Q3 and Q4 (Figs. 9.3 and 9.4), the attempted pseudoimage that is output includes ghost points. In the normal mode of operation of the program, used here, points of nonconvergence are suppressed, and a dashed curve is drawn through the remaining ghost reported points. The latter represent accurately the underlying end points.[41]

[41] In the alternative mode, dashed curves are drawn as before. The points of nonconvergence are also represented, but not as a curve. Instead, either all those points themselves are plotted (as dots), or every Ath such point is plotted, where A is an attrition factor chosen by the user.

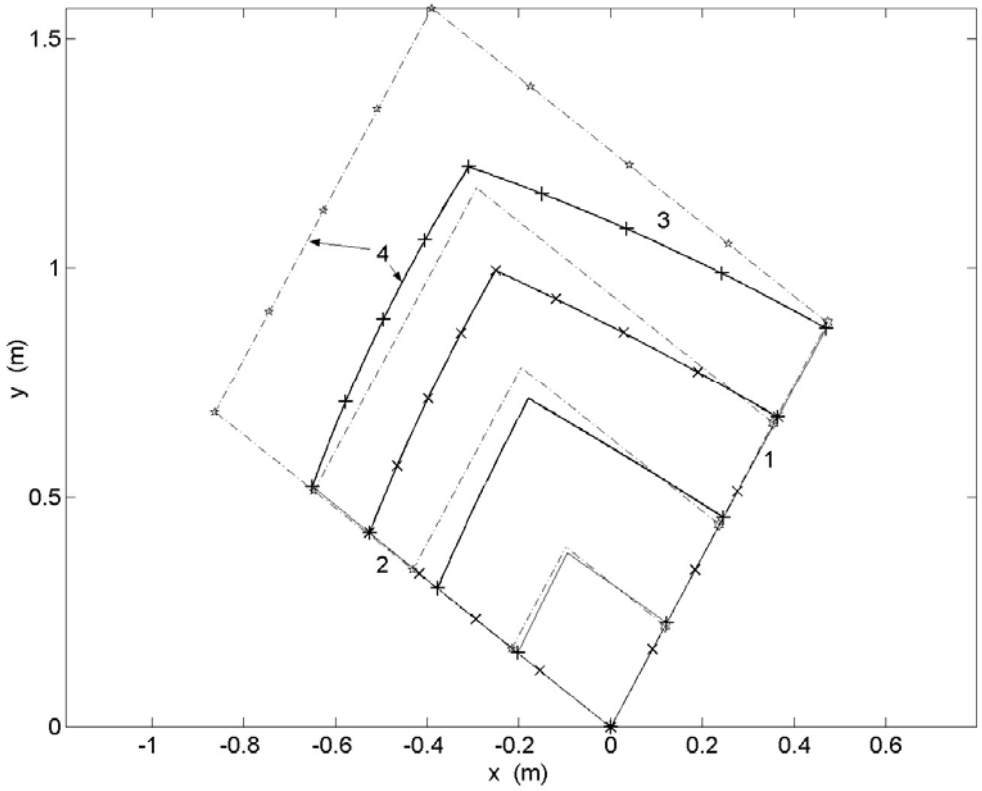

Figure 9.1. Pseudoimages (solid curves) of four squares of different sizes, given by parameter set Q1. Obtained by program quad*. Dash-dot lines: paraxial approximation. The graph shows also the numbering of selected sides with the parameter* si*.*

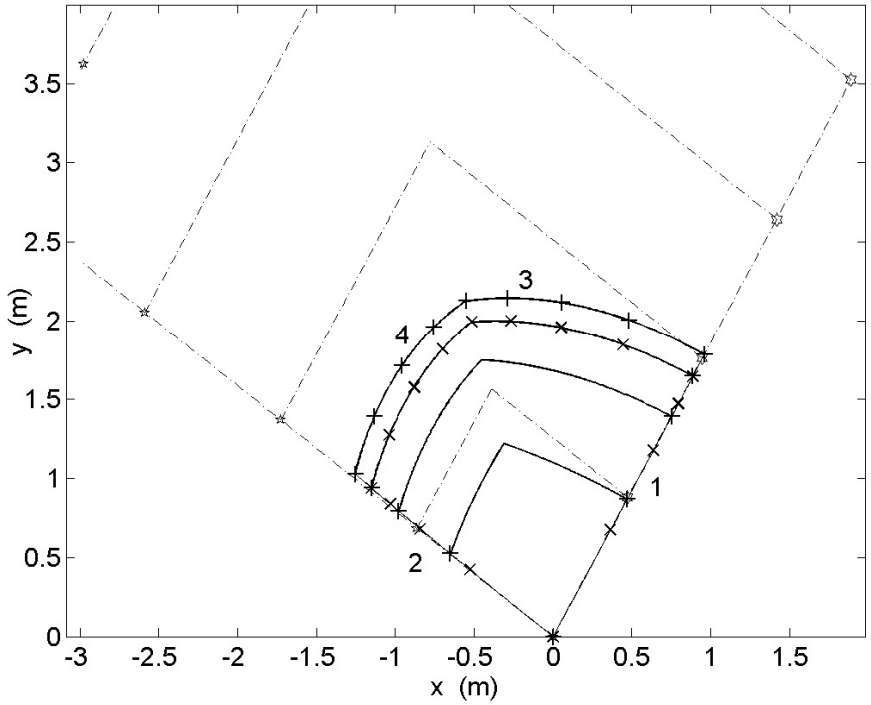

Figure 9.2. As for Figure 9.1, but for parameter set Q2. Note that the paraxial predictions for the c-quads $k = 3$ *and* $k = 4$ *extend beyond the graph's edge, but their positions can be inferred by linear scaling.*

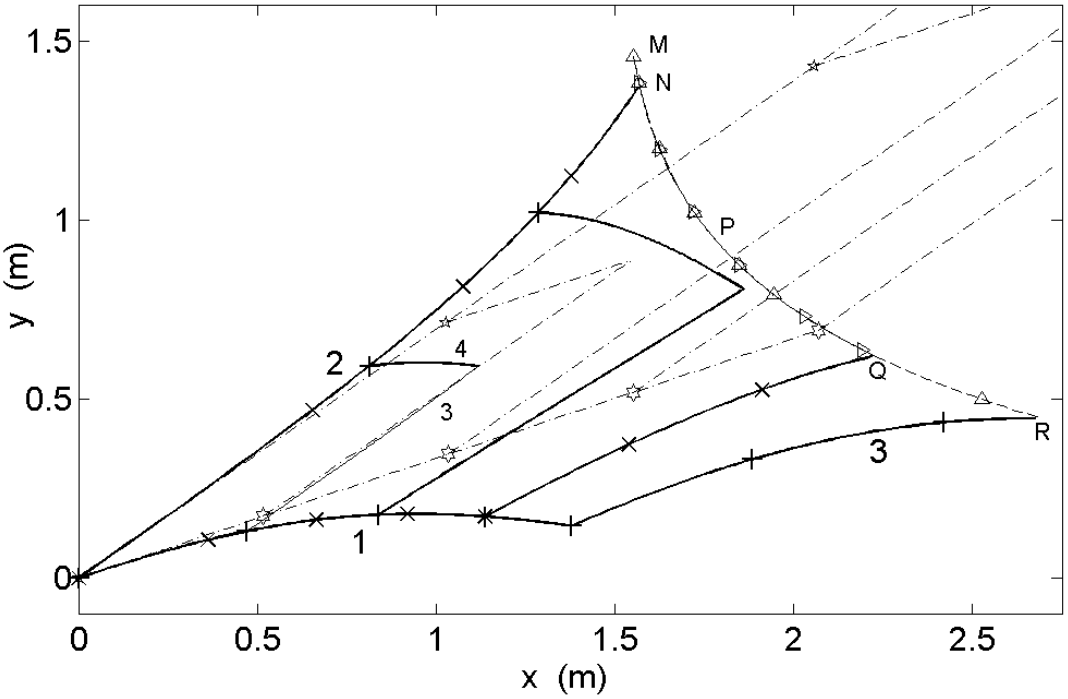

Figure 9.3. As for Figure 9.1, but for parameter set Q3. MNPQR *is a curve of blockage points; although parts of it look like a continuous curve, actually it is a superposition of* dashed *curves.*

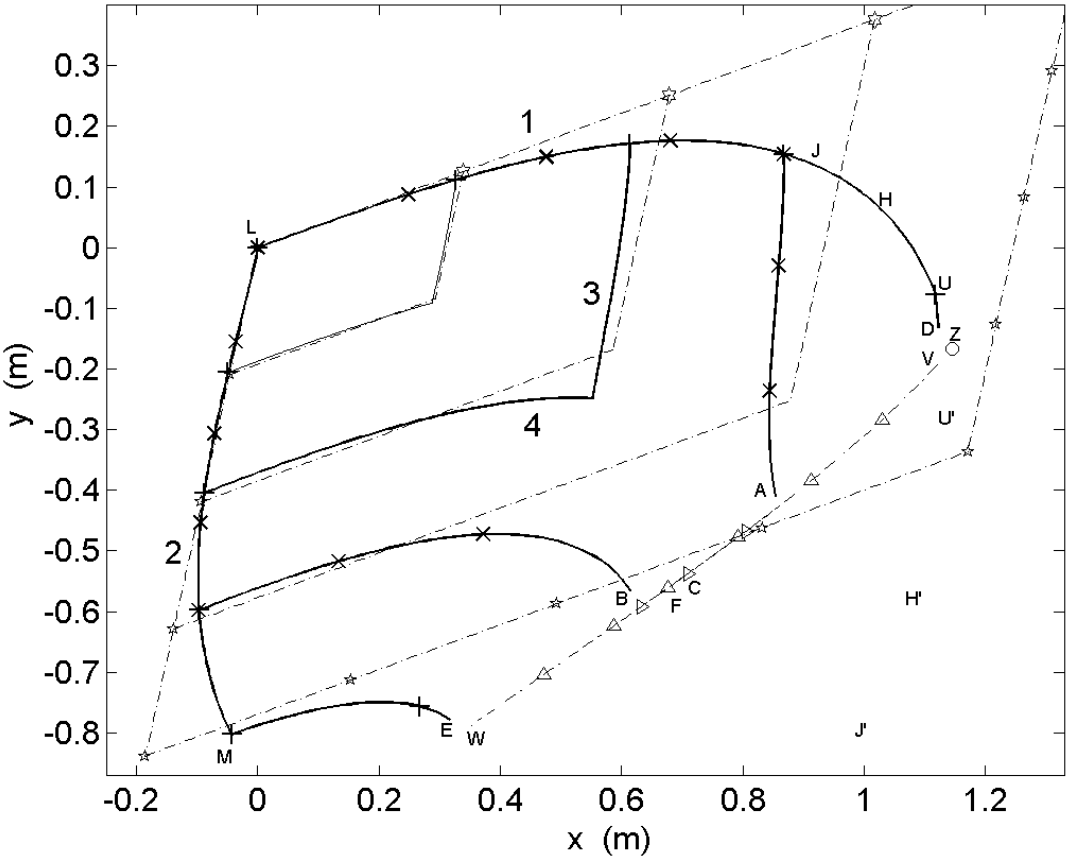

Figure 9.4. As for Figures 9.1 and 9.3, but for parameter set Q4. VCFW *is a curve of local optima.*

In parameter set Q3 (Figure 9.3), one of the two curvatures is negative, so that part of the reflecting surface arches towards the array plane. Already in the pseudoimage of the square $k = 2$ there is severe distortion (compared to the parallelogram).

For the larger squares $k = 3$ and $k = 4$, parts of sides 2 and 3, as well as all of side 4, are missing from the pseudoimage (solid curves). For all these six 'missing' segments (three sides, or part-sides, for each of two squares), each output point is a ghost point; the table shows them all to be blockage points. It is found that each blockage point lies on a curve MNPQR that is *common to all six segments*. (Note that the portion of MPR that presents almost as a continuous curve is in fact made up of coinciding dashed curves.) In fact the dashed curve MPR coincides with the cutoff perimeter, as shown by a calculation, as follows. The default cutoff (Eqn 6.3) places the cutoff plane at $Z = -0.7200$. From the table output by quad, accurate values of (x, y) were noted for two points towards opposite ends of the dotted curve MPR. Using Equations (3.1) and (3.2), the two corresponding values of Z were each found to be -0.7199, in agreement with the prediction. Incidentally, the blockage perimeter MPR is known to be a conic, in particular a hyperbola.

We note that, while the relevant pseudoimage curve ends at N, the blockage curve continues to M. This is readily understood in view of the instability discussed in Section 8.2.1. We also note that at N, Q and R, the underlying pseudoimage curve must end precisely on the blockage perimeter (any slight gap in the drawn figure being due to finite-interval sampling).

We return briefly to parameter set Q1 (Figure 9.1) to raise a suggestion which we then reject. Consider each paraxial prediction given by a five- or six-pointed star on the outermost parallelogram (call any such point P). The corresponding actual pseudoimage point Q is a point marked by a + sign on the $k = 4$ c-quad. In each case, Q is seen to lie very close to the line joining the origin (chief reflecting point) to P. It might be thought that this property holds more generally. However Figure 9.3 shows that it does not. For example, the prediction fails for the sides 1 and 2 (for $k = 4$), since the prediction entails that these actual sides should be close to straight lines. Incidentally, these curved sides are tangential to the respective paraxial lines at the origin, as required.

In parameter set Q4 (Fig. 9.4) both principal curvatures are negative. As a preliminary, note that a *clockwise* rotation is necessary to transform side 1 into side 2; thus the formation of the pseudoimage has involved, among other things, a mirror-image operation.

The ghost points from parameter set Q4 (Fig. 9.4) all turn out (via the table output by quad) to be local optima, in contrast to the blockage points of Figure 9.3. These optimum points come from parts of sides 3 and 4 for each of the input squares[42] $k = 3$ and $k = 4$. The details of the mapping can be deduced from the locations of the triangle markers.[43] A striking result is that the local optimum points from the four part-sides *all lie on the single curve* VCFW. We shall discuss this phenomenon further below.

---

[42] For $k = 4$, a small part of input side 3 generates a portion UD of a pseudoimage curve.

[43] The detailed progress of the optimum point is as follows. For the input square $k = 3$, along side 3, the end point $\mathrm{P}(x, y)$ makes a very small jump from A to a nearby point on the LO curve and then progresses to C (right-pointing triangle). Along side 4, there is a similar jump from B, followed by a similar progression to C. For the square $k = 4$, along side 3, P makes a jump from D to V; it then follows the LO curve to the triangle at F. Along side 4, P jumps similarly from E to W; it then progresses to F.

A fact that is not immediately evident is that there is a *second* pseudoimage of, say, the standard square. To see this, note (from Tables 8.2, 9.2) that the side ($k = 4$, si $= 1$) lies along the same object line as that used to generate Figure 8.11. The side therefore possesses a second pseudoimage branch—a part of PI2. It follows from Figure 8.11 that the portion of the second pseudoimage curve that 'replaces' UHJ is given roughly by U′H′J′. Anticipating, it is roughly a kind of reflection of UHJ about the curve of optima. [That second pseudoimage curve does not appear in Figure 9.4, because quad was designed to produce the quasiparaxial branch only. Interestingly, when quad was run in conjunction with Newton's method, the point (x, y) traced out part of PI1 and then jumped, tracing out part of PI2, extending roughly from H′ to U′.]

The critical point in Figure 8.11 (where LO meets PI) lies approximately at the circle marked Z in Figure 9.4. (To be precise, Z marks the last computed local optimum in Figure 8.11 as the critical point is approached fro`m the right.) It is noteworthy that the curve of optima WFV, when extended to the right, passes through Z (as well as the eye can judge). In view also of the discussion in Section 9.3 below, it is believed that the curve of optima VFW is a smooth extension of the curve of optima in Figure 8.11.[44]

Let us now describe the two pseudoimages of the *interior* of, say, the largest (*i.e.* the $k = 4$) square (the interior being a 2D region). The first pseudoimage of the interior includes the sides ($k = 3$, si $= 3$) and ($k = 3$, si $= 4$) (see Fig.9.4). Note that these two interior curves extend to A and B—and presumably (as underlying curves) they extend slightly further to the curve of optima. Thus there is a first pseudoimage, bounded by the curve of optima and the outermost solid (pseudoimage) curves, that is, DUJLME. And there is a second pseudoimage, extending from the curve of optima to another set of curves, traced out in part by U′H′J′.

For completeness we note some lesser features of the four diagrams. For data set Q3 (Fig. 9.3), sides 1 and 2 diverge (markedly) from each other in the following sense. As the radial distance from the origin (chief reflecting point) increases, the angle between the directions of sides 1 and 2 (*i.e.* the directions of the tangents) becomes greater. By contrast, for data set Q4 (Fig. 9.4), those two sides converge. Again, one can consider, for given directions in the object (u', v') space, how the magnification changes with the size of the input square. For data sets Q1 and Q2, for each direction, the magnification decreases. By contrast, for data set Q4 (Fig. 9.4), it is found that sometimes the magnification increases [*e.g.* compare the cross near $(0.13, -0.53)$ with the plus sign near $(0.26, -0.76)$].

9.3 Common Curve of Local Optima

A striking feature of Figure 9.4 is that the four curves of local optima corresponding to the four relevant sides (si $= 3$ and 4 for each of the squares $k = 3$ and 4) all lie along a common curve. Such coincidence is expected for blockage points (Fig. 9.3), since there is just one cutoff perimeter; but in the case of optima there is no simple explanation. To test this

[44] The fact that the (combined) curve of optima extends *both* ways from Z is, upon reflective thought, not surprising. Consider the input line segment (call it S_1) of Table 8.2 and Figure 8.11. This is collinear with the line segment ($k = 4$, si $= 1$), to be called S_2. For S_1, the curve of optima extends to the right (only) from Z. For the line segment S_2, there is no curve of optima. For the segment S_3 ($k = 4$, si $= 3$), the curve of optima extends from V to F (only). And so on.

coincidence to high accuracy, a section of the curve was chosen near the point C, where *three* sides are represented. Local optimum points from the three sides were plotted on an enlarged scale (Fig. 9.5). The points still lie on a common curve (drawn).[45]

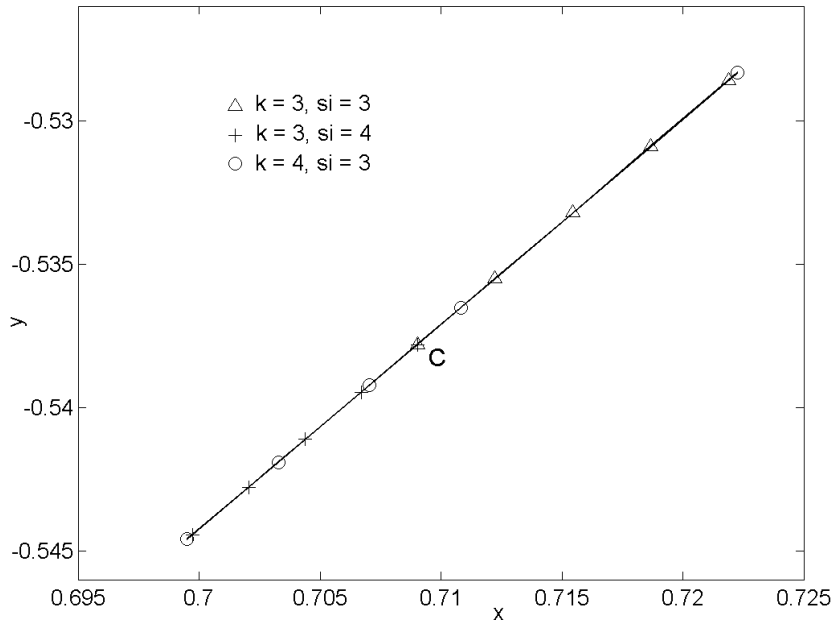

Figure 9.5. Local optimum points generated by three object line segments (sides—see legend) of Figure 9.4. A common smooth curve is fitted.

The fact that two object line segments, emanating from a common point, at right angles to each other, map onto a common curve (as shown in the previous paragraph) is sufficient for us to be confident that *any* line segment emanating from that point maps into the same curve. Thus *the local optimum function maps a 2D region onto a 1D region.* The fact that a third object line (the $k = 4$ one) also yields coincidence gives further confirmation. The italicised result is striking, because normally a function that maps (u', v') to (x, y) maps a 2D region onto a 2D region.

A partial understanding of this result can be given, as there is a precedent (besides the weak precedent of the blockage points). The 2D–1D phenomenon occurs for local optima in the case of circular symmetry (Equation 6.1), as exemplified in Figure 6.1. In that case the curve of optima in the xy plane is a circle, centred on the origin. It is the locus of all the points $\mathbf{x} = (x, y)$ that correspond to the turning point in the function that maps $|\mathbf{x}|$ onto the corresponding object point $|\mathbf{u}'|$.

It seems likely that a fuller explanation of the topology of the local-optimum results can be supplied by catastrophe theory, pioneered by René Thom (Thom, 1975; Zeeman, 1977; Postle, 1980).

[45] This segment is extremely close to a straight line; but Figure 9.4 shows that on a larger scale there is appreciable curvature.

9.4 A Case of Three Pseudoimages

We now describe a case where an extended object has *three* pseudoimages—a case where quite interesting distortion occurs.

For the circularly symmetric system to which Figure 6.1 applies (parameters given in Section 6.3.1), let the coordinates of the turning point on the right be called $(x_t, -u_t)$ (so that $x_t > 0$, $u_t > 0$). Let the x coordinate of the zero-crossing point H be x_h. Consider as the object a small square of side c placed near the $\mathbf{u}'$ origin. In particular, let $c = \frac{1}{10}u_t$ and let a diagonal of the square run from $\left(-\frac{1}{4}c, -\frac{1}{4}c\right)$ to $\left(-\frac{5}{4}c, -\frac{5}{4}c\right)$. By sketching the $\mathbf{u}'$ plane and the $\mathbf{x}$ plane and referring to Figure 6.1, it is straightforward to see what the three pseudoimages look like. (In the sketches, three relevant circles should be drawn, of radii u_t, x_t and x_h respectively.)

One pseudoimage, corresponding to the paraxial branch, is approximately a square, located in the first $\mathbf{x}$ quadrant and lying close to the origin (compared to the x_t circle). A second pseudoimage (branch near H in Fig. 6.1) lies again in the first quadrant, but at a radial distance just short of x_h. But this pseudoimage is elongated in the circumferential direction, so that it extends from fairly near the $+x$ axis around to fairly near the $+y$ axis, following a curved path 'approximately parallel' to the circle of radius x_h. The third pseudoimage is similar to the second, being circumferentially stretched. However it lies in the third $\mathbf{x}$ quadrant at a radial distance just beyond x_h.

It is worth noting what happens as the side c is increased. Eventually the first and second pseudoimages abut one another, the curve of separation being the curve of local optima $|\mathbf{x}| = x_t$. This helps to explain how the similar situation in Figure 9.4 comes about.

10. Conditions on the Geometrical Approximation

The conditions on the geometrical approximation have been discussed at some length in I. Here we recall the main results from I and (in Section 10.1) add some new comments on those results. Actually it has been realised that two further conditions are necessary, given in Sections 10.2 and 10.3 respectively.

Suppose that it is desired that the approximation gives a good description of the pseudoimage of an array, the outer boundary of which is approximately a circle of diameter L. Consider the case where $m_X = m_Y \equiv m$ and the array plane is roughly parallel to the reflector. Let S_1 be the pseudoimage of the array center, and let D_1 be the distance from S_1 to the nearest edge of the reflector. The condition for assumption 1 (reflection treated by geometrical acoustics) to hold is that

$$D_1 \geq \tfrac{1}{2}|m|L + \alpha|mr_0\lambda|^{1/2} \tag{10.1}$$

as shown in I. Here λ is the wavelength at the center of the chirp, and α is a purely numerical constant, expected to be somewhat greater than one. A reasonable guess is that α should be taken as 5/2 for a fairly good description and 15/2 for quite good accuracy.[46]

Assumption 2 is concerned with the effects of numerical diffraction as discussed in Section 2. Due to this diffraction, a square tile (i.e. a square subarray that is processed coherently) of side a has its pseudoimage blurred over some distance b. We discuss only the case where $|m|$ is of order unity, since it appears impossible to discuss the case of a more general magnification until the corresponding wave theory has been developed. Then assumption 2 is good (on the scale of L) if $b << L$. As shown in I, this condition on assumption 2 reduces to

$$r_0 \lambda / a << L \tag{10.2}$$

We seek a criterion for the predicted geometrical pseudoimage to closely resemble the actual pseudoimage. Equation (10.2) yields a final criterion of the form $r_0 \lambda / a \leq \beta L$, where it is suggested that β is equal to something like 1/30. A further point—a positive one—can be made: as paper I points out, for one important purpose a weaker condition should suffice, in which $\beta \sim 1/3$. That purpose is the use of a pseudoimage to make a not-too-bad estimate of the magnification (and hence the curvature) in cases where there are two spots which, though blurred considerably, can be resolved as two separate spots.

10.1 Comments

Some comments on the above conditions are in order. First, in the case of an elongated array, with two characteristic lengths, L_1 and L_2, sufficient conditions should be obtained by requiring that each condition above holds for both L_1 and L_2. Second, in cases of non-parallelism ($\delta \neq 0$), the above conditions still apply provided that one first replaces the original array by its projection onto the uv plane.[47]

Third, in cases where part of the geometrical pseudoimage lies near (or beyond) the edge of the reflector, clearly condition (10.1) is violated. However in some circumstances one can find a *part* of the array for which the pseudoimage is geometrical. In that case S_1 is taken to be the pseudoimage of the center of the *part* array, and L is replaced by L', the size of the part array, before applying Equations (10.1) and (10.2). Finally, the present Section 10 assumes that the large-range approximation is at least an order-of-magnitude guide to the geometrical pseudoimage. Hence in the cases of more severe distortion (*e.g.* magnification severely reduced) the present conclusions require modification.

[46] For pseudoimage points near the center of the reflector, the image amplitude may differ from the expected value, due to the edge of the reflector following closely an edge of a so-called Fresnel zone (*e.g.* Ditchburn, 1952).

[47] In that case, a is replaced by two projected values, a_1 and a_2. Then Equation (10.2) must be applied to the pair (L_1, a_1) and also to (L_2, a_2).

10.2 Density of Elements

The receiving elements have been treated as though they formed a continuum of element density, denoted by $g(\mathbf{u})$, the (local) number of elements per unit area. [Here we drop the primes from $\mathbf{u}'$ and $\mathbf{u}'_0$.] This should be a good approximation provided that the element density is high enough. We now find the detailed condition required.

Consider, on the reflecting surface, the point $\mathbf{r}_0$ that, in geometrical acoustics, reflects to a point $\mathbf{u}_0$ in the array plane. From Section 2 and especially Appendix A, the points on the array plane that contribute, by constructive interference, to the image amplitude at $\mathbf{r}_0$ are the points $\mathbf{u}$ such that Equation (A.3) holds, that is,

$$|\mathbf{u} - \mathbf{u}_0| < |m|^{-1} s$$

where s is the distance defined by Equations (A.1) and (A.2). Let $g_0 = g(\mathbf{u}_0)$; thus g_0 is the element density in the neighbourhood of $\mathbf{u}_0$ (initially a sum of delta functions), but averaged over a region containing several (ideally many) elements.[48] Then

$$h = g_0^{-1/2}$$

is a measure of the distance between neighbouring elements.

Consider the physical (acoustic) wave as it reaches the plane of the receiving array. In a small region near a given point on the array, over a short time-interval, the complex pressure of the wave (versus position and time) may be written as the pressure of a plane wave multiplied by what we shall call the 'envelope,' where the envelope varies slowly on the scale of one wavelength (and one period). The envelope varies in the lateral directions on the scale of $|m|^{-1} s$. And the Fourier components of the envelope are appreciable only over wave numbers of the order of the reciprocal of $|m|^{-1} s$. For a reconstruction of the envelope that is close to what would be reconstructed from continuous sampling, sampling must occur at spacings down to $\theta_3 |m|^{-1} s$, where $\theta_3 \sim 1/10$ (say). The required condition is therefore[49]

$$h < \theta_3 |m|^{-1} s$$

This condition should apply irrespective of whether the array is random or regular.

When the left-hand side rises above the right-hand side, all is not lost. The resolution (as a distance *on the reflecting surface*, in order-of-magnitude terms) simply rises in proportion to the left side; thus, instead of s, the resolution becomes

$$|m| h$$

[48] The region is never taken to extend beyond the edge of the array (to where the element density is strictly zero). Instead a region that terminates at the edge is taken for the purpose of averaging.

[49] An investigation strongly suggests that s itself contains a factor $|m|^{1/2}$.

10.3 Grating Lobes

A wave treatment is believed to produce, due to the elements in a small region[50] centered on $\mathbf{u}'$, a contribution to the image amplitude that is peaked at the geometrical pseudoimage of $\mathbf{u}'$. However, in the case of a point target, it is known that periodicities in the array can lead, via grating lobe effects, to spurious images at points other than that geometrical pseudoimage. These periodicities correspond to a high Fourier amplitude, at high wave-number, in the spectrum of the element strength distribution. It is expected that similar effects occur with specular surfaces. Hence a requirement on the geometrical approximation is that the Fourier amplitudes at such wave-numbers be small.

11. Application to a General Smooth Surface

For the specified surfaces such as the paraboloid,[51] the conditions on the 'present algorithm' (by which we mean the algorithm of the present report) have been stated above. Consider now a general smooth reflecting surface. In the neighbourhood of any point on the surface, the surface is well approximated by a paraboloid. Based on this fact, it can be shown that the algorithm also gives a good approximation to the pseudoimage for a general smooth surface, subject however to further conditions, now to be derived.

We consider the case—which is rather common—in which two conditions are fulfilled. The first condition is that there is a chief reflecting point S_0, that is, a point on the surface such that a ray from the transmitter to S_0 is reflected back to the (point) transmitter. The second condition is that the transmitter (or its projection onto the array plane) lies within or near the array.[52] As before, the ray reflected from S_0 coincides with what is called the 'chief normal,' and the point where the latter intersects the array plane is called the 'chief receiving point' T_0. We consider the reflecting surface to be replaced by a paraboloid with its vertex at S_0, having the same two principal curvatures at S_0 as the reflecting surface. For points on the receiving array sufficiently close to T_0 (or equivalently, for points on the reflecting surface sufficiently close to S_0), the present algorithm gives the pseudoimage to a good approximation. Note that the departure of the predicted pseudoimage point from the true one is due, not to a failure of the paraxial approximation (because that approximation is not assumed), but due to the surface's not being a paraboloid.

We define the coordinates X, Y and Z as before, so that, on the paraboloid, Z is a homogeneous quadratic function of X and Y. For simplicity we consider the case where the absolute values of the two principal curvatures are of the same order of magnitude as each other, say of order $\kappa_1 = 1/\rho_1$. (κ_X and κ_Y may be of either sign; κ_1 is positive.) We consider the (quite common) case where the departure of the true surface from the paraboloid is

---

[50] Essentially the 'small region' must be of extent somewhat greater than $|m|^{-1}s$.

[51] The 'specified' surfaces are the paraboloid (with the transmitter lying on the paraboloid's axis), the sphere, the cylinder and the plane.

[52] A good practical criterion is that the distance of the (projected) transmitter from the center of the array is less than $1.5(L/2)$, or $0.75L$, where L is the length or diameter of the (square or circular) array.

qualitatively the same as the departure of a sphere from a paraboloid of revolution, or more generally, the departure of an ellipsoid from an elliptic paraboloid. More precisely, we consider the case where each of Z and its first and second derivatives (with respect to X and Y) are given accurately by the paraboloid provided that both

$$|X| << \rho_1, \qquad |Y| << \rho_1 \tag{11.1}$$

(Typically then, the error first becomes appreciable at displacements of order ρ_1.) The mapping from $\mathbf{u}' = (u', v')$ to its pseudoimage point $\mathbf{X} = (X, Y)$—and likewise the inverse mapping—is then given accurately by the present algorithm provided

$$|\mathbf{X}| << \rho_1 \tag{11.2}$$

(To be precise, we note that there are rare combinations of parameters for which an appreciable, or even large, error arises due to a local magnification approaching infinity. The phenomenon is essentially the same as when a radius of curvature approaches ρ_0 in the situation discussed in Section 5.3.) Equation (11.2) is the condition we seek. Equation (11.2) can be rewritten as a condition on $\mathbf{u}'$ (instead of $\mathbf{X}$), but such a condition would be considerably more complicated.

As Equation (11.2) entails that the gradients ($\partial Z/\partial X$ and $\partial Z/\partial Y$) are small, it might be thought that (11.2) entails that the rays (both forward and reflected) are paraxial, or equivalently, that the paraxial approximation holds. But this is not so, because the chief condition on the paraxial approximation, namely Equation (5.9), is not guaranteed to hold by (11.2). Rather, in the subcase

$$|\mathbf{X}| << r_0, \qquad |\mathbf{X}| << \rho_1 \tag{11.3}$$

both the present algorithm and the paraxial approximation are valid.[53] There remains the subcase

$$|\mathbf{X}| \gtrsim r_0, \qquad |\mathbf{X}| << \rho_1 \tag{11.4}$$

in which the present algorithm, but not the paraxial approximation, is valid.[54]

12. Conclusions; Future Work

The geometrical approximation has been applied to 'specified' surfaces, namely the paraboloid with two principal curvatures—with the transmitter lying on the paraboloid's axis—, as well as the sphere, the cylinder and the plane. Within that approximation, the exact general solution for the forward problem has been given, in the form of a nest of formulae, implemented in the routine ray. For the inverse problem—the problem of determining the pseudoimage when given an object in the array plane—an algorithm has been described and implemented in the routine imige. Starting from a given object point, the algorithm can lead to multiple pseudoimage points, and also to blockage points and local optima, the natures of which have been discussed. Pseudoimages of extended objects have also been discussed.

Among other topics discussed, the report shows that the present algorithm applies to a general smooth surface, subject to further conditions.

---

[53] Subject also to the 'rare combinations' proviso above.

[54] In this subcase, the reflecting surface is always near-planar, in the sense that $\rho_1 >> r_0$. As a result it can be shown that, within the region (11.4), there is a subregion of considerable size in which the exact planar solution, given in I, is a good approximation.

We have developed the paraxial approximation, a useful generalisation of the large-range approximation in which the pseudoimage of a square is still a parallelogram.

A suite of programs with graphical outputs has been produced. These show how the pseudoimage point gradually diverges from the paraxial prediction as the object point moves further from the chief receiving point. Of the programs, multigen is the most general, as it attempts to find *all* the end points for object points lying along a *general* line in $u'v'$ space. The program quad shows how the pseudoimage of an object square develops away from the paraxially-predicted parallelogram.

In future work, it is intended to dispense with the two assumptions of the geometrical approximation and instead use wave theory. As discussed in Section 1.3, in the present system, wave effects arise in two ways. First, acoustic waves are propagated and reflected: these are amenable to a treatment via Huygens wavelets undergoing spherical spreading. Second, the image-forming also introduces wave effects. A wave treatment would not only deal with the wave-induced blurring of the pseudoimage; it would also predict the *image amplitude distribution* $A(\mathbf{r})$; such a prediction is not attempted by the geometrical approximation. The 'bright points' calculation (Section 2 and Appendix A) is suggestive of how such a wave treatment would proceed. A preliminary account of such a wave treatment for the large-range case has been given by Blair (2004).

By reasoning as in Section 11, it may prove possible to develop such a wave theory, not only for the specified surfaces, but, at the same time, for a general smooth surface, by concentrating on one small part of the surface at a time.

A wave treatment would provide an answer to a question raised in Section 1.2: Does the theory of the image-forming for a collection of randomly placed point scatterers apply *in toto* to specular reflectors? A preliminary calculation suggests that the answer is no, and that the theory of the point spread function is different, and more complex, for the specular reflector.

In more detail, the preliminary calculation just mentioned is an extension of the 'bright points' calculation (Section 2 and Appendix A). Tentatively, the results are as follows. One can define an appropriate point spread function (PSF)[55] showing, for a given point $\mathbf{u}'$ in the receiving array, the extent to which the element strength density $g(\mathbf{u}')$ at $\mathbf{u}'$ contributes to the image amplitude $A(\mathbf{r})$ at a given point $\mathbf{r}$. This PSF depends for its existence on there being a point $\mathbf{r}_0$ that (according to geometrical acoustics) reflects to a point $\mathbf{u}'_0$ in (or near the edge of) the receiving array. The PSF depends on the values of certain parameters such as the principal curvatures at $\mathbf{r}_0$, but does not depend on the function $g(\mathbf{u}')$. The formula thus obtained for the PSF is valid only for $\mathbf{r}$ near $\mathbf{r}_0$ and $\mathbf{u}'$ near $\mathbf{u}'_0$.

The 'preliminary calculation' yields further results. Because the PSF is not a delta function, $g(\mathbf{u}')$ is not *sharply* reproduced in the pseudoimage, but is subject to blurring ('spreading') in the lateral directions. For a given $\mathbf{u}'$, the spread in $\mathbf{r}$ is essentially over the region S (Eqn A.2); for a given $\mathbf{r}$, the spread of the contributing $\mathbf{u}'$ values is over the region R (Eqn A.3). In particular, consider any edge—an edge of the array in the case of the fully coherent mode, or

[55] Warning: More than the usual care must be taken in defining this function, due to effects associated with the distinction between coherence and incoherence.

an edge of a tile in the partly coherent case. It is at such edges that the spreading will be most apparent.

In addition, if the calculation of Appendix A is correct, it is easy to show that the dimensions of each of S and R are proportional to $\lambda^{1/2}$, where λ is a wavelength typical of the Fourier components of the transmitted signal. Hence each spread is also proportional to $\lambda^{1/2}$.

A final point concerns what happens as the typical wavelength λ approaches zero. Here we may consider any extended region[56] R′ of the receiving array. It is believed (particularly as the spread is suggested to be proportional to $\lambda^{1/2}$) that the wave treatment would confirm the following: that, as λ approaches zero, the lateral spreading of the pseudoimage of R′ (and, in particular, the spread of the pseudoimage of any edge) approaches zero, and that the edges of the resulting sharp pseudoimage are identical to those predicted by the geometrical approximation.

Acknowledgements

The imaging apparatus was constructed by TUS with the assistance of the Division of Telecommunications, CSIRO. I thank Dr. Ian S.F. Jones for his advice and for his strong encouragement. Dr. Stuart Anstee is thanked for his comments on the draft, which were both extensive and helpful.

References

Anderson-Dutoit, B. (2002). *Echocardiography: The Normal Examination of Echocardiographic Measurements*, Illustrated Ed. Blackwell, Oxford, UK.

Aster, R.C., Borchers, B. and Thurber, C.H. (c. 2005). *Parameter Estimation and Inverse Problems*. Elsevier Academic Press, Amsterdam.

Belcher, E., Dinh, H., Lynn, D. and Laughlin, T. (1999). Beamforming and Imaging with Acoustic Lenses in Small, High-Frequency Sonars. In *Proceedings of Oceans 1999 Conference*, pp. 1495–1499 (conf. held 13–16 September, Seattle, Washington).

Belcher, E., Hanot, W. and Burch, J. (2002). Dual-Frequency Identification Sonar (DIDSON). In *Proceedings of the 2002 International Symposium on Underwater Technology* (IEEE Cat. No. 02EX556), pp. 187–192 (conf. held at Piscataway, New Jersey).

Belcher, E., Matsuyama, B. and Trimble, G. (2001). Object Identification with Acoustic Lenses. In *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings* (IEEE Cat. No. 01CH37295) (Marine Technol. Soc. Part Vol. 1, 2001), pp. 6–11 (conf. held 5–8 November, Honolulu, Hawaii).

Bellanger, M. (c. 1984). *Digital Processing of Signals: Theory and Practice*. Wiley, Chichester, UK.

Björck, Å. (1996). *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia.

[56] Here the region R′ must be large enough so that the effects of constructive interference are dominant. This condition is satisfied if the size of R′ is approaching the size of R (Eqn A.3) or is larger.

Blair, D.G. (1997). *Underwater Acoustic Imaging: A Computing Hardware Approach to Rapid Processing* (DSTO Technical Note DSTO-TN-0099). Aeronautical and Maritime Research Laboratory, Melbourne.

Blair, D.G. (2002). *Theory Pertaining to Comparison and Calibration in an Experiment to Measure Acoustic Attenuation Coefficients* (DSTO Technical Note DSTO-TN-0417). Systems Sciences Laboratory, Melbourne.

Blair, D.G. (2004). Image due to a Curved Specular Reflector in Acoustic Mine Imaging. In: *Proceedings of Mine Countermeasures and Demining Conference* (conf. held at the Australian Defence Force Academy, Canberra, Australia, February 9–11, 2004). Organised by the Defence Science and Technology Organisation, Sydney.

Blair, D.G. (2006) (referred to as I). Underwater Acoustic Imaging: Image due to a Specular Reflector in the Geometrical-Acoustics Limit. *J. Mar. Sci. Technol.* **11**, 123–130.

Blair, D.G. and Anstee, S.D. (2000). *Underwater Acoustic Imaging: A Simulation Program and Related Theory* (DSTO Technical Note DSTO-TN-0274). Aeronautical and Maritime Research Laboratory, Melbourne.

Blair, D.G. and Jones, I.S.F. (1998). *Underwater Acoustic Imaging: Rapid Signal Processing* (DSTO Technical Note DSTO-TN-0098). Aeronautical and Maritime Research Laboratory, Melbourne.

Blair, D.G., Jones, I.S.F. and Madry, A. (2006). *Underwater Acoustic Imaging: One-Bit Digitisation* (OTG Report No. 1/06). Ocean Technology Group, University of Sydney.

Bosma, O. and Kuc, R. (1994). A Physical Model-Based Analysis of Heterogeneous Environments Using Sonar-ENDURA Method. *IEEE Trans. Pattern Anal. Machine Intell.* **16**, Issue 5, 497–506.

Bouxsein, P., An, E., Schock, S. and Beaujean, P.-P. (2006). A SONAR Simulation Used to Develop an Obstacle Avoidance System. In *Oceans 2006 – Asia Pacific* (IEEE, Piscataway, New Jersey), pp. 884–890 (conf. held in Singapore).

Broadstone, S.R., Chiang, A.M. and Impagliazzo, J. (1999). Low-Power, High-Resolution 3D Sonar Imaging System. *Proceedings of the SPIE – The International Society for Optical Engineering*, Vol. 3711, pp. 57–67. USA.

Chiang, A.M., Broadstone, S.R. and Impagliazzo, J.M. (2002). A Portable, Electronic-Focusing Sonar System for AUVs Using 2D Sparse-Array Technology. In *Oceans 2002 Conference and Exhibition. Conference Proceedings* (IEEE Cat. No. 02CH37362) (Part Vol. 4, 2002), pp. 2129–2134 (conf. held at Piscataway, New Jersey).

Chiang, A., Broadstone, S. and Impagliazzo, J. (2003). Development of a Handheld Bistatic Imaging Sonar System for Underwater Search and Survey. *J. Acoust. Soc. Am.* **114**, 2368.

Clay, C.S. and Medwin, H. (1977). *Acoustical Oceanography: Principles and Applications.* John Wiley, New York.

Culver, R.L. and McDaniel, S.T. (1991). Bistatic Ocean Surface Reverberation Simulation. In *ICASSP 91: International Conference on Acoustics, Speech and Signal Processing* (IEEE Cat. No. 91CH2977-7), pp. 1453–1456 (conf. held at Toronto, Ont., Canada).

Ditchburn, R.W. (1952). *Light*. Blackie, London.

Ferguson, B.G. and Wyber, R.J. (2005). Application of Acoustic Reflection Tomography to Sonar Imaging. *J. Acoust. Soc. Am.* **117**, No. 5, 2915–2928.

Goldstein, A. and Powis, R.L. (1999). Medical Ultrasonic Diagnostics. In Papadakis, E.P. (ed.), *Ultrasonic Instruments and Devices: Modern Instrumentation, Techniques and Technology*, Illustrated Ed., pp. 46–193. Academic Press, New York.

Goodman, J.W. (1976). Some Fundamental Properties of Speckle. *J. Opt. Soc. Am.* **66**, No. 11, 1145–1150.

Guggenheimer, H.W. (1963). *Differential Geometry.* McGraw-Hill, New York.

Hansen, R.K. (1993). An Acoustic Camera for 3D Underwater Imaging. In *International Conference on Acoustic Sensing and Imaging* (Conf. Publ. No. 369), pp. 99–102 (conf. held 29–30 March 1993). IEE, London.

Hansen, R.K. (2008). *Method of Constructing Mathematical Representations of Objects from Reflected Sonar Signals.* USA Patent, Pub. No. US 2008/0043572 A1 (Pub. 2008).

Hansen, R.K. and Andersen, P.A. (1996). A 3D Underwater Acoustic Camera—Properties and Applications. In Tortoli, P. and Masotti, L. (eds), *Acoustical Imaging*, pp. 607–611. Plenum, New York.

Hansen, R.K. and Andersen, P.A. (1998). The Application of Real Time 3D Acoustic Imaging. In *IEEE Oceanic Engineering Society. Oceans '98. Conference Proceedings* (IEEE Cat. No. 98CH36259) (Part Vol. 2, 1998), pp. 738–741. New York.

Jones, I.S.F. (1996). *Underwater Acoustic Imaging Innovation Program* (DSTO Technical Note DSTO-TN-0065). Aeronautical and Maritime Research Laboratory, Melbourne.

Jones, I.S.F. (2000). Dimensional Images from a High-Resolution Underwater Imager. In *Proceedings of the 32nd Offshore Technology Conference*, held at Houston, Texas, 1–4 May 2000, paper 12110. Offshore Technology Conference, Richardson, Texas.

Jupp, D.L.B. and Vozoff, K. (1975). Stable Iterative Methods for the Inversion of Geophysical Data. *Geophys. J. R. Astron. Soc.* **42**, 957–976.

Kleeman, L. and Kuc, R. (1995). Mobile Robot Sonar for Target Localization and Classification. *Int. J. Robotics Res.* **14**, No. 4, 295–318.

Knudsen, D.C. (1989). A New Beamformer for Acoustic Imaging. In *Oceans '89: An International Conference Addressing Methods for Understanding the Global Ocean.* IEEE Press, New York.

Kreyszig, E. (1959). *Differential Geometry*. University of Toronto Press, Toronto.

Kuc, R. and Viard, V.B. (1991). A Physically Based Navigation Strategy for Sonar-Guided Vehicles. *Int. J. Robotics Res.* **10**, No. 2, 75–87.

Lalor, E. (1968). Inverse Wave Propagator. *J. Math. Phys.* **9**, 2001–2006.

Lanczos, C. (1958). Linear Systems in Self-Adjoint Form. *Am. Math. Monthly* **65**, 665–679.

Ljunggren, S., Lovhaugen, O. and Mehlum, E. (1980). Seismic Holography in a Norwegian Fiord. In Metherell, A.E. (ed.), *Acoustic Imaging. International Symposium on Acoustical Holography and Imaging, 1978*, Vol. 8, pp. 299–315. Plenum, New York.

Maguer, A., Vesetas, R. and Azemard, F. (2000). 3D Acoustic Imaging of Objects in Water. In *Acoustics 2000: Proceedings of Australian Acoustical Society Annual Conference*, held at Joondalup Resort, Western Australia, 15–17 November 2000, pp. 87–93. Australian Acoustical Society, Perth, WA.

Manzie, G. (2000). High Resolution Acoustic Mine Imaging. In *UDT Pacific 2000: Undersea Defence Technology*, Darling Harbour, NSW, Australia, 7–9 February 2000, pp. 356–359. Nexus Information Technology, Swanley, Kent, UK.

Marquardt, D.W. (1963). An Algorithm for Least-Squares Estimation of Non-Linear Parameters. *J. SIAM* **11**, 431–441.

McCrae, W.H. (1960). *Analytical Geometry of Three Dimensions.* Oliver and Boyd, London.

Nai-Chyuan Yen and Dragonette, L.R. (1997). Wave Packet Decomposition for Acoustic Target Recognition. In *Proceedings of the SPIE – The International Society for Optical Engineering*, Vol. 3069, pp. 436–445. USA.

Nash, S.G. and Sofer, A. (1996). *Linear and Nonlinear Programming.* McGraw-Hill, New York.

Nocedal, J. and Wright, S.J. (1999). *Numerical Optimisation.* Springer-Verlag, New York.

Parker, R.L. (1994). *Geophysical Inverse Theory*. Princeton University Press, Princeton, New Jersey.

Perlis, S. (1952). *Theory of Matrices.* Addison-Wesley, Cambridge, Mass.

Postle, D. (1980). *Catastrophe Theory: Predict and avoid personal disasters*. Fontana, London.

Rihaczek, A.W. (1985). *Principles of High-Resolution Radar*, Revised Version. Peninsula, Los Altos, Calif.

Shapiro, R.S., Simpson, W.L., Rautsch, D.L. and Hsu-Chong Yeh (2001). Compound Spatial Sonography of the Thyroid Gland. *Am. J. Roentgenol.* **177**, 1195–1198.

Shewell, J.R. and Wolf, E. (1968). Inverse Diffraction and a New Reciprocity Theorem. *J. Opt. Soc. Am.* **58**, 1596–1603.

Steinberg, B.D. (1976). *Principles of Aperture and Array System Design—Including Random and Adaptive Arrays*. Wiley, New York.

Spiegel, M.R. (1968). *Mathematical Handbook of Formulas and Tables.* McGraw-Hill, New York (Schaum's Outline Series).

Thom, R. (1975). *Structural Stability and Morphogenesis: An Outline of a General Theory of Models* (transl. by D.H. Fowler). W.A Benjamin, Reading, Mass.

Twomey, S. (1996). *Introduction to the Mathematics of Inversion in Remote Sensing and Indirect Measurements.* Dover, Mineola, New York.

Urick, R.J. (1983). *Principles of Underwater Sound*, 3rd Ed. McGraw-Hill, New York.

Vesetas, R. and Manzie, G. (2001). AMI: A 3-D Imaging Sonar for Mine Identification in Turbid Waters. In *Oceans, 2001: MTS: IEEE Conference and Exhibition*, Honolulu, 5–8 November 2001, Vol. 1, pp. 12–21. IEEE Press, New York.

Wilhjelm, J.E., Jensen, M.S., Jespersen, S.K., Sahl, B. and Falk, E. (2004). Visual and Quantitative Evaluation of Selected Image Combination Schemes in Ultrasound Spatial Compound Scanning. *IEEE Trans. Med. Imag.* **23**, No.2, 181–190.

Williams, K.L. and Funk, D.E. (1994). High-Frequency Forward Scattering from the Arctic Canopy: Experiment and High-Frequency Modeling. *J. Acoust. Soc. Am.* **96**, No. 5, Pt. 1, 2956–2964.

Zeeman, E.C. (1977). *Catastrophe Theory: Selected Papers 1972–1977*. Addison-Wesley, Reading, Mass.

Appendix A: Bright Points

Let $\mathbf{r}_0$ be a point on the specular surface—a point which, in geometrical acoustics, reflects to a point $\mathbf{u}_0$ on the array. (Here we drop the primes from $\mathbf{u}_0'$, $\mathbf{u}'$, etc.) We consider points $\mathbf{u}$ on the array near $\mathbf{u}_0$. By the theory of Huygens wavelets, the pressure at $\mathbf{u}$ is the sum of the contributions from many paths, where each go-and-return path consists of two straight lines that meet at a point $\mathbf{r}'$ on the surface. For a monofrequency signal, the phase shift of any contribution is proportional to the corresponding go-and-return path length $\Lambda(\mathbf{r}', \mathbf{u})$. The (complex) image amplitude $A(\mathbf{r})$ at a point $\mathbf{r}$ on the surface is obtained via the image-forming equation; $A(\mathbf{r})$ thus consists of a phase-shifted sum of the pressures over the elements. Formally this sum can be written as an integral over $\mathbf{u}$, in which $g(\mathbf{u})$, the element strength per unit area, appears as a factor in the integrand. Due to the time delays of the image-forming equation, the contribution to $A(\mathbf{r})$ from the *path pair* $(\mathbf{r}', \mathbf{u})$ has an associated total phase shift proportional to the path difference $\Lambda(\mathbf{r}', \mathbf{u}) - \Lambda(\mathbf{r}, \mathbf{u})$ associated with the path pair.

We recall Fermat's principle (*e.g.* Ditchburn, 1952, p. 216), which states (in the case of reflections) that, for a given $\mathbf{u}$, *the point* $\mathbf{r}(\mathbf{u})$ *of (geometrical) reflection* to $\mathbf{u}$ is such that the path length $\Lambda(\mathbf{r}', \mathbf{u})$ is stationary with respect to variations in $\mathbf{r}'$ (on the surface) about $\mathbf{r}(\mathbf{u})$. Wave theory justifies this principle by pointing out that, because of the stationarity, constructive interference between many more Huygens wavelets than usual, with values of $\mathbf{r}'$ near $\mathbf{r}(\mathbf{u})$, occurs at $\mathbf{u}$. In other words, the values of $\mathbf{r}'$ that interfere constructively occupy a much larger region than usual.

Similarly, in the present system, a bright point will appear at $\mathbf{r}$ when, in the 4D integration over $\mathbf{r}'$ and $\mathbf{u}$, there is a 4D region, of rather large extent in all four directions, such that all the path pairs picked out by that region interfere constructively at $\mathbf{r}$.

At the beginning of this appendix we considered points $\mathbf{r}_0$, $\mathbf{u}_0$ and $\mathbf{u}$. For such a point $\mathbf{u}$, consider the Huygens paths that reach $\mathbf{u}$. A path via any point $\mathbf{r}'$ interferes constructively with the path via the fixed point $\mathbf{r}_0$ provided that

$$\left|\Lambda(\mathbf{r}', \mathbf{u}) - \Lambda(\mathbf{r}_0, \mathbf{u})\right| < \lambda \qquad \text{(A.1)}$$

(where $\lambda =$ wavelength). More correctly, a coefficient of λ *of order unity*—call the coefficient θ_1—should be inserted on the right-hand side; the value of θ_1 depends on the largest phase difference that is counted as giving constructive interference. Now, in the first instance, consider the case $\mathbf{u} = \mathbf{u}_0$. ($\mathbf{u}_0$ is defined at the start of the appendix.) Because of the stationarity associated with Fermat's principle, Equation (A.1) holds for a larger-than-usual region S of $\mathbf{r}'$ space. Ignoring anisotropic cases for simplicity, that region S may be written as

$$\left|\mathbf{r}' - \mathbf{r}_0\right| < s \tag{A.2}$$

(for some s, the calculation of which is postponed to a later article).

Now consider the region R of $\mathbf{u}$ space defined by

$$\left|\mathbf{u} - \mathbf{u}_0\right| < \left|m\right|^{-1} s \tag{A.3}$$

where m is the *local* magnification (as given by geometrical acoustics). At any point $\mathbf{u}$ in R, $\mathbf{u}$ is still in the 'coherent beam' reflected from the region S, and so Equation (1) should still hold.[57] (Adjustments may have to be made to the coefficient θ_1 and to a similar coefficient θ_2 to be inserted in Eqn A.3. From here on, the inclusion of such coefficients θ is to be understood.)

(Actually, the present argument suffices only for the case where the reflected rays are roughly normal to both the array and the reflecting surface. Treatment of the general case is postponed to a later article, and may require the definition of different kinds of magnification. The latter would depend, for example, on whether the 'size' of the 'object' is measured in the array plane or measured via the projection onto the plane perpendicular to the rays.)

Since the region (A.2) is large in the relevant sense, so also is the region (A.3) (at least for a fixed m that is neither zero or infinite). Consider now the combinations $(\mathbf{r}', \mathbf{u})$ such that $\mathbf{r}' \in \mathrm{S}$ and $\mathbf{u} \in \mathrm{R}$: these generate a *large* region in 4D space for which the inequality (A.1) holds. Hence at $\mathbf{r} = \mathbf{r}_0$ the point is bright.

We note in passing that the argument strongly suggests the following: that essentially it is the elements within R (possibly with an altered value of θ) that contribute to, and suffice to produce, the bright point at $\mathbf{r} = \mathbf{r}_0$.

The argument (predicting a bright point at $\mathbf{r}_0$) may be repeated at any other point $\mathbf{r}$ by calling it $\mathbf{r}_0$. Note that the argument strongly suggests a certain 'blurring' of the pseudoimage as follows. Given an initial $\mathbf{r}_0$, as $\mathbf{r}$ varies away from $\mathbf{r}_0$, $A(\mathbf{r})$ differs from $A(\mathbf{r}_0)$ by a rather small amount until the condition

$$\left|\mathbf{r} - \mathbf{r}_0\right| < s \tag{A.4}$$

(possibly with a different value of θ to that in Eqn A.2) is violated.

Appendix B: Termination of the Inversion Algorithm

The decision whether to terminate the algorithm is taken at the end of each internal iteration. The decision depends on the value of relerr, the relative error (Eqn 7.3), and also on the value of relstep, the relative step most recently tried. For any tentative step $\delta\mathbf{x}$ (taken from a

[57] After all, surely when the '<' sign in (A.3) is replaced by '<<,' all the relevant phases are changed by an amount that is small compared to $\pi/2$.

position $\mathbf{x} = \mathbf{x}_{n-1}$), relstep is defined as[58]

$$\text{relstep} = |\delta\mathbf{x}|/|\mathbf{x}|$$

At the decision point, several tests are performed. Termination at a pseudoimage point is declared if relerr is less than 10^{-10}; it is also declared if both $\text{relerr} < 10^{-5}$ and $\text{relstep} < 10^{-5}$. If a pseudoimage point is not found, a blockage point or a local optimum may be found, as follows. First, the step size is considered to be small enough to warrant termination of the algorithm if either $\text{relstep} < 10^{-10}$ or the 60th *internal* iteration has been performed. When that combined condition holds, consider the most recent running of ray at the latest (central) $\mathbf{x}$ value and at the four associated points used in estimating the derivatives: in each case ray may have failed (to find a corresponding value of $\mathbf{u}'$). If ray (having succeeded at the central point $\mathbf{x}$) failed at an associated point, a blockage point is declared. If, on the other hand, it succeeded at all four associated points, a local optimum is declared.

Appendix C. Printout of Selected Routines

In the present work about 16 routines (main programs and subroutines) are used, each being an m-file written in MATLAB. From these, the most useful collection of four routines has been selected for reproduction here; they are imparax.m, ray.m, imige5.m and multigen.m. (Strictly, the routine called imige.m in the body of the report possesses the name imige5.m.) These four routines suffice to run the program multigen. Several of the other routines can be regarded as modelled on multigen: these consist of all the routines that call either ray or imige5 and produce graphical output.

multigen speaks of 'fail codes'; these are what the body of the report calls—more appropriately—'ending codes.' imige5 speaks of 'failure codes': such a code is a more primitive description of the ending. The fail code (ending code) in multigen and similar programs is deduced from a combination of two parameters, namely, imige5's 'failure code' and another output from imige5.

C.1 imparax.m

```
% Subroutine imparax.m
%
% DESCRIPTION: For a sonar array viewing a specular reflector, this routine
% accepts a description of the system and a point on the sonar array.  It
% outputs the point on the reflecting surface that is, according to the
% PARAXIAL APPROXIMATION, the pseudoimage of that point
%
% Author of routine: David Blair, from December 2006 to July 2008

function xve2 = imparax(r0, e, delta, alpha, kappaX, kappaY, upri, vpri)

% This function m-file does the same as imige5.m , except that
% it assumes also the PARAXIAL APPROXIMATION.  This approximation has wider
% conditions of validity than the large-range approximation
%
```

[58] The use of the relative error and the relative step (in place of absolute quantities) in the convergence criteria is particularly appropriate when investigating small departures from the paraxial approximation, and does no harm when used more generally.

```
% INPUT VARIABLES: See the routine imige5
%
% OUTPUT VARIABLES:
% xve2       row vector of two elements, namely:
% x, y            coordinates of the geometrical pseudoimage point S of
%            the point R.  R lies on the receiving array and S lies on the
%            reflecting surface.

uprive2 = [upri, vpri];
Qalpha = [cos(alpha), sin(alpha); -sin(alpha), cos(alpha)];
magX = 1/(2*kappaX*r0 + 1 + r0/(r0 + e));
magY = 1/(2*kappaY*r0 + 1 + r0/(r0 + e));
Qm = diag([magX, magY]);
Qiproj = diag([cos(delta), 1]);

xve2 = (Qalpha'*Qm*Qalpha*Qiproj*uprive2')';
```

C.2 ray.m

```
% Subroutine ray.m
%
% DESCRIPTION
% For a sonar array viewing a specular reflector, this routine accepts the
% description of the system and the geometry of the outgoing or forward
% ray.  It outputs the point where the reflected ray meets the receiving
% array
%
% Author of routine: David Blair.  June 2005 to July 2008

function uprive2f = ray(shape, Zb, r0, e, delta, alpha, kappaX, ...
    kappaY, x, y)

% This function m-file deals with the pseudoimage of the receiving array
% produced when an underwater acoustic image device "views" a specular
% reflector.  Within the geometrical approximation, given a point S on the
% reflecting surface, the routine determines the point R such that a ray
% emanating from the transmitter and the meeting the reflector at S, upon
% reflection meets the array plane at R.  Strictly, what is input is the x
% and y coordinates, which usually determine the point S(x,y,z).  However,
% a corresponding valid reflecting point S does not always exist.
% Furthermore, even when S does exist, a corresponding valid object point R
% does not always exist.  However, whenever R does exist, it is unique.
%
% An alternative description (which however, assumes the existence of S and
% R) is as follows.  When a point S on the pseudoimage is input into the
% routine, the routine calculates the point R on the array of which the first
% point is the pseudoimage.
%
% Image amplitudes do not enter into the calculation, which is merely
% concerned with the functional relationship that maps the one point, S, onto
% the other, R.
%
% It is assumed that the reflecting surface is described by an equation of
% one of the three forms
% cylinder:        X^2 + (Z - 1/kappaX)^2 = 1/kappaX^2
% sphere:        X^2 + Y^2 +(Z - 1/kappa)^2 = 1/kappa^2
% paraboloid:         Z = 0.5*kappaX*X^2 + 0.5*kappaY*Y^2
% In the paraboloid, kappaX and kappaY are the principal curvatures.  For the
% cylinder and the sphere, kappaX (kappa respectively) is a curvature.  For
% all three forms, the kappas concerned can be positive or negative.
% For all three forms, the chief
% normal to the surface is defined as the normal that passes through the
% spherical centre of the transmitter.  Z is the coordinate measured along
% the chief normal, starting from the chief reflecting point.  Z increases
% as the point concerned moves away from the array towards the reflecting
% surface; the direction of that movement is called the 'forward'
```

```
% direction.  [For more on the kappas and similar parameters, see below under
% "Input Variables."]
%
% INPUT VARIABLES:
%
% shape      1 for cylinder, 2 for sphere, 3 for paraboloid
%            For a plane, input 3 (or 1 or 2 now allowed) with
%            kappaX = kappaY = 0
% Zb         must be > 0; BUT, as a code, it can equal 0 or -1 (see end of
%            the description of Zb for the codes); AND it can also equal 0
%            in an "EXCEPTIONAL" case described just five lines down.
%            The reflecting surface is taken to be cut off at Z = Zb or
%            -Zb; the details follow.
%            The range of Z values retained (i.e. not cut off) always
%            includes Z=0
%            For a paraboloid with both kappas >= 0, there is
%            no cutoff and Zb is ignored; in that case it is recommended to
%            input Zb as zero (this is an EXCEPTION to the above rule).
%            For other paraboloids, the
%            boundary (cutoff) is taken at -Zb; only the portion of the
%            quadric surface with Z > -Zb is retained.
%            For a cylinder or a sphere with positive
%            curvature, the reflector extends from Z = 0 to Z = +Zb.  For a
%            cylinder or sphere with negative curvature, the reflector
%            extends from Z = -Zb to Z = 0.
%            For a cylinder or a sphere, Zb must not exceed abs(radius).
%         CODES
%            Zb = 0 (code applicable to a cylinder or a sphere):  This
%            instructs the program ray to put Zb equal to abs(radius).
%            Zb = -1:  The program ray replaces the value -1 by a
%            default (positive) value of Zb.  For details, see the program.
% r0         range, as measured along the chief normal, from the plane of
%            the array to the reflecting surface
% e          transmitter offset, measured along the chief normal, from the
%            array plane to the spherical centre of the transmitter
%            (positive if centre is behind the array plane)
% delta      angle that the chief tangent plane makes with the array plane
% alpha      Preliminary: An xyz system is set up as in the paper below.  Its
%            origin is at the chief reflecting point.  The xy plane is the
%            chief tangent plane.  The y axis is parallel to a line in the
%            array plane.  Then:
%            For a paraboloid, alpha is the angle between the x axis and
%            the principal direction X of the reflecting surface.
%            (For a paraboloid of revolution or a flat reflector, recommended
%            to put alpha = 0)
%            Similarly for a cylinder, where the Y axis is parallel to the
%            axis of the cylinder.
%            For a sphere, the input value of alpha is ignored and
%            internally alpha is put equal to zero.
% kappaX, kappaY
%            For a paraboloid: the principal curvatures, as above
%            For a flat reflector: input with both kappas equal to zero;
%            shape may be 1, 2 or 3.  Internally, shape is equated to 3
%            (paraboloid)
%            For a cylinder: kappaX (positive or negative but not zero) is
%            the curvature; a convention is enforced whereby the input kappaY
%            is required to be zero
%            For a sphere: kappaX (positive or negative) is the
%            curvature, and kappaY must equal kappaX
%            For any of the kappas, a positive value means that the relevant
%            centre of curvature lies forward of the chief reflecting point.
% x, y       coordinates of the selected input point S(x, y, z); the latter
%            lies on the reflecting surface
%
% All inputs are in SI units (radian, metre, metre^(-1))
%
% SIGN, AND OTHER, CONVENTIONS, including the definitions of the xyz and
% (upri, vpri, wpri) coordinates systems: See the paper, Blair, D.:
% "Underwater acoustic imaging: Image due to a specular reflector in the
```

```
% geometrical-acoustics limit," J. Marine Science & Technol., vol 11,
% pp. 123-130 (2006).  (In the descriptions of programs, e.g. the present
% description, uprime and vprime are often abbreviated to upri and vpri.)
%
% OUTPUT VARIABLES:
%
% uprive2f   row vector of three elements, as follows.
%     upri, vpri          coordinates of the point R in the array plane
%            to which a ray incident at S is reflected.  Geometrically, S
%            is the pseudoimage of R.  The upri and vpri axes are in the
%            plane of the array.
%     wpri   the third coordinate of R, always equal to zero
% fail       a fail code
%            = 0 if, for the given (x,y), a valid reflecting point S and a
%                valid object point R exist
%            = a higher integer otherwise (SEE DETAILS AT END OF CODE)

fail = 0;

% check input values
if shape ~= 1 & shape ~= 2 & shape ~= 3
    disp('shape must equal 1, 2 or 3')
    pause
end

% shapi is the value for shape that the routine uses internally
if shape ~= 3 & kappaX == 0 & kappaY == 0 % adjust shapi for flat reflector
    shapi = 3;
else
    shapi = shape;
end
if shapi == 1 | shapi == 2
    if kappaX == 0
        disp('in ray, for a cylinder or sphere, kappaX must not be zero ')
        disp('unless kappaY is also zero')
        pause
    end
    if shapi == 1 & kappaY ~= 0
        disp('in ray, for a cylinder, kappaY must be zero')
        pause
    end
    if shapi == 2 & kappaY ~= kappaX
        disp('in ray, for a sphere, kappaY must equal kappaX')
        pause
    end
end
if Zb < 0 & Zb ~= -1
    disp('in ray, Zb must be > zero or else equal to 0 or -1')
    pause
end
if shapi == 3 & (kappaX < 0 | kappaY < 0)
    if Zb <= 0 & Zb ~= -1
        disp(['in ray, for this shape Zb must strictly exceed zero', ...
        ' or else equal -1'])
        pause
    end
end
if shapi == 1 | shapi == 2
    if Zb > 1/abs(kappaX)
        disp('in ray, for cylinder or sphere, Zb must not exceed the radius')
        pause
    end
end

% set default value of Zb if appropriate
if Zb == -1
    if shapi ==1 | shapi == 2
        kap = abs(kappaX);
        Zb = min(1/(2*kap), 0.5*kap*r0^2);
```

```
%       Zb = min((1 - 1/sqrt(2))/kap, 1/kap - sqrt(1/kap^2 - r0^2));
    elseif shapi ==3
        if kappaX >= 0 & kappaY >= 0
            Zb = 0;
        else
            kap = abs(min(kappaX, kappaY));
            Zb = min(1/(2*kap), 0.5*kap*r0^2);
        end
    end
end

% ************************************************************************
% initial processing
if shapi == 2
    alpha = 0;
end
if shapi == 1 | shapi == 2
    kappa = kappaX;
end
if shapi == 1 | shapi == 2
    if Zb == 0
        Zb = (1./abs(kappa))*(1 - 1.e-14);
    end
end

% initial processing continued
%    -Zm is the minimum value of Z on the retained surface
%    Zp [defined only when (surface is a cylinder or a sphere) and kappa > 0]
%       is the maximum value of Z on the retained surface

if shapi == 3
    if kappaX >= 0 & kappaY >= 0
        Zm = 0;
    else
        Zm = Zb;
    end
end
if shapi == 1 | shapi == 2
    if kappa > 0
        Zp = Zb;
        Zm = 0;
    end
    if kappa < 0
        Zm = Zb;
    end
end

% cylinder
if shapi == 1
    sigrad = 1./kappa;
    xve2 = [x, y];
    Qalpha = [cos(alpha), sin(alpha); -sin(alpha), cos(alpha)];
    temp = (Qalpha*xve2')';
    X = temp(1);
    Y = temp(2);
    if kappa > 0
        sign = -1;
    end
    if kappa < 0
        sign = +1;
    end

    if sigrad^2 - X^2 < 0
        fail = 2;
    end
    if fail == 0
        Z = sigrad + sign*sqrt(sigrad^2 - X^2);
    else
        Z = 0;
```

```
    end
    z=Z;

    if kappa > 0 & Z > Zp
        if fail == 0
            fail = 3;
        end
    end
    if kappa < 0 & Z < -Zm
        if fail == 0
            fail = 3;
        end
    end

    side4 = -x*sin(delta) + (z + r0)*cos(delta);
    if side4 < 0
        if fail == 0
            fail = 4;
        end
    end

    xve3 = [xve2, z];
    bprve2 = [kappaX*X, 0];
    bve2 = (Qalpha'*bprve2')';
    bve3 = [bve2, kappa*Z - 1];
    F = sqrt(x^2 + y^2 + (z+r0+e)^2);
    ahat = -(1/F)*[x, y, z+r0+e];
    G = sqrt((kappaX*X)^2 + (kappa*Z - 1)^2);
    bhat = (1/G)*bve3;

% sphere
elseif shapi == 2
    sigrad = 1./kappa;
    xve2 = [x, y];
    Qalpha = [cos(alpha), sin(alpha); -sin(alpha), cos(alpha)];
    temp = (Qalpha*xve2')';
    X = temp(1);
    Y = temp(2);
    if kappa > 0
        sign = -1;
    end
    if kappa < 0
        sign = +1;
    end

    if sigrad^2 - X^2 - Y^2 < 0
        fail = 2;
    end
    if fail == 0
        Z = sigrad + sign*sqrt(sigrad^2 - X^2 - Y^2);
    else
        Z = 0;
    end
    z=Z;

    if kappa > 0 & Z > Zp
        if fail == 0
            fail = 3;
        end
    end
    if kappa < 0 & Z < -Zm
        if fail == 0
            fail = 3;
        end
    end

    side4 = -x*sin(delta) + (z + r0)*cos(delta);
    if side4 < 0
        if fail == 0
```

```
            fail = 4;
        end
    end

    xve3 = [xve2, z];
    bve3 = [kappa*x, kappa*y, kappa*z - 1];
    F = sqrt(x^2 + y^2 + (z+r0+e)^2);
    ahat = -(1/F)*[x, y, z+r0+e];
    G = sqrt((kappa*x)^2 + (kappa*y)^2 + (kappa*z - 1)^2);
    bhat = (1/G)*bve3;

% paraboloid
elseif shapi == 3
    xve2 = [x, y];
    Qalpha = [cos(alpha), sin(alpha); -sin(alpha), cos(alpha)];
    temp = (Qalpha*xve2')';
    X = temp(1);
    Y = temp(2);
    Z = 0.5*kappaX*X^2 + 0.5*kappaY*Y^2;
    z=Z;

    if Z < -Zm
        if fail == 0
            fail = 3;
        end
    end

    side4 = -x*sin(delta) + (z + r0)*cos(delta);
    if side4 < 0
        if fail == 0
            fail = 4;
        end
    end

    xve3 = [xve2, z];
    bprve2 = [kappaX*X, kappaY*Y];
    bve2 = (Qalpha'*bprve2')';
    bve3 = [bve2, -1];
    F = sqrt(x^2 + y^2 + (z+r0+e)^2);
    ahat = -(1/F)*[x, y, z+r0+e];
    G = sqrt((kappaX*X)^2 + (kappaY*Y)^2 +1);
    bhat = (1/G)*bve3;
end

% ********************************************************************
% resume calculation for all three shapes together
abdot = sum(ahat.*bhat);
chat = 2*abdot*bhat - ahat;

if chat(1,3) >= -1/30
    if fail ==0
        fail = 5;
    end
end

fail6 = 0;
denR = chat(1,1)*sin(delta) - chat(1,3)*cos(delta);
if denR == 0
    fail6 = 11;
    % reflected ray is parallel to the array; see override to "9" below
    if fail == 0
        fail = 6;
    end
end
if fail6 == 11
    coeffR = 0;
else
    coeffR = (-x*sin(delta) + (z+r0)*cos(delta)) / denR;
end
```

```
if coeffR/r0 > 30
    if fail == 0
        fail = 6;
    % reflected ray is approaching being parallel to array
    end

elseif coeffR < 0
    if fail == 0
        fail = 6;
    % reflected ray is directed away from the array
    end
end

if fail6 == 11 & fail ~= 2 % "9" overrides 3 to 6
    fail = 9;
end

Rxve3 = xve3 + coeffR*chat;
if Rxve3(3) >= -Zm
    if fail == 0
        fail = 7;
    end
end

if fail == 0 & (shapi == 1 | shapi == 2) & kappa < 0
    tH = -(Zm + z)/chat(1,3);
    Hxve3 = xve3 + tH*chat;
    Hxve2 = [Hxve3(1), Hxve3(2)];
    temp = (Qalpha*Hxve2')';
    XH = temp(1);
    YH = temp(2);
    ZH = Hxve3(3);
    if shapi == 1
        if XH^2 + (ZH - 1./kappa)^2 - 1./kappa^2 > 0
            fail = 8;
        end
    elseif shapi == 2
        if XH^2 + YH^2 + (ZH - 1./kappa)^2 - 1./kappa^2 > 0
            fail = 8;
        end
    end
end

if fail == 0 & shapi == 3 & (kappaX < 0 | kappaY < 0)
    tH = -(Zm + z)/chat(1,3);
    Hxve3 = xve3 + tH*chat;
    Hxve2 = [Hxve3(1), Hxve3(2)];
    temp = (Qalpha*Hxve2')';
    XH = temp(1);
    YH = temp(2);
    ZH = Hxve3(3);
    if 0.5*kappaX*XH^2 + 0.5*kappaY*YH^2 - ZH < 0
        fail = 8;
    end
end

Qdelta = [cos(delta), sin(delta); -sin(delta), cos(delta)];
Ru = Rxve3(1);
Rv = Rxve3(2);
Rw = Rxve3(3) + r0;
vpri = Rv;
uwprive2 = (Qdelta*[Ru, Rw]')';
uprive2f =[uwprive2(1), vpri, fail];

% FAIL CODES
%
%           NOTE: A "forward" direction is a direction in which Z increases,
%                 or a direction moving outwards from the array.  The context
```

```
%                determines which of these two descriptions is relevant
% fail = as follows
% 0    no failure, a REAL (i.e. GENUINE) object point exists
% 2    given x and y, there is no corresponding point on the hemisphere
%      or half-cylinder
% 3    point S on the quadric is not on the reflecting surface by virtue
%      of its lying behind or beyond the cutoff plane
% 4    reflecting point S is backward of the array
% 5    reflected ray travels in a direction for which Z is increasing, or
%         nearly in such a direction
%    NOTE re 5 and 6: The criterion for "nearly" is approximately "being
%    within 1/30 radian of such a direction"
% 6    reflected ray travels away from array, or nearly so
% 7    either (1) the receiving point R is forward of the chief tangent plane,
%      or     (2) there is a cutoff plane and R is forward of it.
% 8    reflected ray meets the reflecting surface again before it meets the
%      array
% 9    reflected ray is parallel to the array plane

% NOTE: Each numerical fail code overrides those that are numerically
% greater, in the sense that if there are two reasons for failure, the
% lesser code is reported.  However, as an exception, the code "9"
% is regarded as lying numerically between 2 and 3

% VIRTUAL OBJECT POINTS
% When the fail code is in the interval 3 to 8 inclusive, a virtual object
% point exists and is output by the function ray
% IF AND ONLY IF THE FAIL CODE IS 2 OR 9, there exists neither a real nor a
% virtual object point.  Then the output coordinates have no physical
% significance whatever (and in fact are based on an arbitrary step in the
% computation).
```

C.3 imige5.m

```
% Subroutine imige5.m
%
% DESCRIPTION
% For a sonar array viewing a specular reflector, this subroutine accepts a
% description of the system and the point where the reflected ray meets the
% sonar array.  When successful, it outputs a point on the reflecting
% surface that is either a pseudoimage of the input point, a local optimum
% or a blockage point
%
% Author of program: David Blair, from 27 April 2005 to July 2008 (Replace-
% ment of Newton's method by the Levenberg-Marquardt method commenced 10
% April 2008)
%
% NOTE: This program is based on the Levenberg-Marquardt method.  Thus it
% differs markedly from the older version, imige5nwt.m, which is based on
% Newton's method
%
% GENERAL
%
% This function m-file deals with the pseudoimage of the receiving
% array produced when an underwater acoustic imaging device "views" a specular
% reflector.  Within the geometrical approximation, when a point R on the
% array is input into the routine, the primary aim of the routine is to find a
% point S, on the reflecting surface, that is the pseudoimage of R.  Image
% amplitudes do not enter into the calculation, which is merely concerned with
% the functional relationship that maps the one point onto the other.
%
% There may be as many as three pseudoimage points for a given object
% point; a single running of imige finds at most one.  Every running of
% imige performs an iterative process, in which (x, y) progresses from an
% initial guess through a sequence of values, with the "error" [in (uprime,
% vprime) space] always decreasing.  [For the definition of "error" and
```

```
% uprime, vprime, x and y, see the description of the program ray.m or see
% under "Levenberg-Marquardt Method" below.]
% Upon the completion of the routine, (x, y) has reached some "endpoint,"
% which is reported (i.e. is output).  The endpoint may be a pseudoimage point
% but is otherwise said to be a "ghost reported point."
%
% A ghost reported point may be:
%   - a blockage point [(x, y) can progress no further because it has
%     reached, for example, the cutoff perimeter]
%   - a local optimum [the "error" has reached a local minimum], or
%   - a point of nonconvergence [i.e. any other endpoint]
% A point of nonconvergence is:
%   - a maximum-iterations point [the number of main iterations performed
%     has reached the maximum allowed value], or
%   - a 'rare' endpoint [this type of endpoint occurs rarely and perhaps
%     never]
% We now presume that 'rare' endpoints do not occur.  Then it is believed
% that, whenever a point of nonconvergence is reached, there is an
% UNDERLYING endpoint--a pseudoimage point, a blockage point or a local
% optimum--that would eventually be reached by a superior algorithm allowing
% many more iterations.  Blockage points and local optima--whether found or
% underlying--are collectively called "ghost points."
%
% CODES for endpoints: imige does not directly report a code specifying
% which of the above endings has occurred.  The determination of that code is
% done in multigen.m and in certain other programs that call imige; for
% a description of these codes see the program multigen.m  Instead, imige
% reports according to a more primitive classification of endings, with a
% corresponding code [output variable xve(5) = failim; see below under "Output
% Variables"].  From that code, together with xve(6) = flrycen, the type
% of ending as above (more sophisticated scheme) can be deduced.
%
% LEVENBERG-MARQUARDT METHOD
%
% Described in Aster, RC, Borchers and Thurber, "Parameter estimation and
% inverse problems" (Elsevier Academic Press, Amsterdam and Boston,
% c. 2005), p. 176.
%
% An iterative method of solving a system of nonlinear equations
%                          u = f(x)
% where x and u are column (or row) vectors.  If there is no solution, the
% method minimises the "error" between the two sides.  [In the
% acoustics problem, the
% vector x, when spelt out, is (x, y), and u is (uprime, vprime).  For details,
% see program ray.m, or see under "Input Variables" below]  An initial guess
% for the vector x is required.  In each iteration, a step is calculated via
% the inversion of a matrix
%                  J_tran * J  +  lambda * I
% Here J is the Jacobian matrix (essentially du/dx); tran is the transpose;
% and I is the identity matrix.  lambda is a scalar whose value depends on
% the detailed algorithm.  A small value of lambda makes the method
% approximate to the Gauss-Newton method (also called the Gauss method); the
% latter reduces to Newton's method when the Jacobian is square (i.e. vectors u
% and x have the same length).  This lambda gives an uncertain but potentially
% fast convergence (i.e. to a pseudoimage point, a local optimum or a blockage
% point).  A large value of lambda makes the method
% approximate to the method of steepest descent.  This lambda gives certain
% but slow (or very slow) convergence.
%
% Regarding the setting of lambda, the present program follows the
% suggestion of Aster et al. One starts off with a low value of lambda.  Let
% the "error" be distance of the input or "target" u vector from the f(x)
% calculated from the value of x reached so far (in the iterations).   If the
% tentative step (i.e. the step calculated as above) reduces the
% error, the step is taken and lambda is halved for the next iteration.  If
% it does not reduce the error, lambda is repeatedly doubled until a step
% is found that does reduce the error.  That step is taken (and lambda is
% not further changed as the next iteration is entered).
%
```

```
% GENERAL--MORE
%
% The reflecting surface has one of three shapes: cylinder, sphere and
% paraboloid.  For details, see the routine ray.m and also below under “Input
% Variables.”
%
% The result output by imige depends on the parameter parax.
%
% Case parax = 1:
% Then the INITIAL GUESS is equal to the pseudoimage point predicted by the
% paraxial approximation (the values of the input parameters xin and yin
% are ignored.)  A consequence is that, in cases where the pseudoimage curve
% has multiple branches, what is normally found is the pseudoimage lying on
% THAT BRANCH (call it B1) that is the continuation of the PARAXIAL
% PREDICTION (where the latter is construed as existing only where it is
% valid).  [Here a 'branch' means a continuous curve
% or continuous surface that essentially maps the object point into the
% pseudoimage point (or endpoint of some other type).]
% A branch may come to an end.  When the input object point lies
% beyond the end of the branch B1, the routine normally finds the endpoint
% (blockage point or local optimum) that is on the branch that is most
% naturally regarded as the continuation of the branch B1.
%
% Case parax = 0:
% Then the initial guess is specified in the input parameters xin and yin
% (values of x and y).  Then the result
% of the iterative procedure may lie on a branch other than B1.  If a
% suitable LARGE COLLECTION OF initial guesses are used in separate calls to
% imige, very often all the branches are found; that is, all the pseudoimages
% and all the endpoints corresponding to the given object point are found.
% Note the qualification 'very often': sometimes some of the endpoints are not
% found.
%
%
% INPUT VARIABLES:
%
% shape      1 for cylinder, 2 for sphere, 3 for paraboloid
%            For a plane, input 3 (or 1 or 2 now allowed) with
%            kappaX = kappaY = 0
% Zb         must be > 0; BUT, as a code, it can equal 0 or -1 (see end of
%            the description of Zb for the codes); AND it can also equal 0
%            in an "EXCEPTIONAL" case described just five lines down.
%            The reflecting surface is taken to be cut off at Z = Zb or
%            -Zb; the details follow.
%            The range of Z values retained (i.e. not cut off) always
%            includes Z=0
%            For a paraboloid with both kappas >= 0, there is
%            no cutoff and Zb is ignored; in that case it is recommended to
%            input Zb as zero (this is an EXCEPTION to the above rule).
%            For other paraboloids, the
%            boundary (cutoff) is taken at -Zb; only the portion of the
%            quadric surface with Z > -Zb is retained.
%            For a cylinder or a sphere with positive
%            curvature, the reflector extends from Z = 0 to Z = +Zb.  For a
%            cylinder or sphere with negative curvature, the reflector
%            extends from Z = -Zb to Z = 0.
%            For a cylinder or a sphere, Zb must not exceed abs(radius).
%        CODES
%            Zb = 0 (a code applicable only to a cylinder or a sphere):  This
%            instructs the program ray to put Zb equal to abs(radius).
%            Zb = -1:  The program ray replaces the value -1 by a
%            default (positive) value of Zb.  For details, see the program
%            code in ray.m.
% r0         range, as measured along the chief normal, from the plane of
%            the array to the reflecting surface
% e          transmitter offset, measured along the chief normal, from the
%            array plane to the spherical centre of the transmitter
%            (positive if centre is behind the array plane)
% delta      angle that the chief tangent plane makes with the array plane
```

```
% alpha      Preliminary: An xyz system is set up as in the paper below.  Its
%            origin is at the chief reflecting point.  The xy plane is the
%            chief tangent plane.  The y axis is parallel to a line in the
%            array plane.  Then:
%            For a paraboloid, alpha is the angle between the x axis and
%            the principal direction X of the reflecting surface.
%            (For a paraboloid of revolution or a flat reflector, recommended
%            to put alpha = 0)
%            Similarly for a cylinder, where the Y axis is parallel to the
%            axis of the cylinder.
%            For a sphere, the input value of alpha is ignored and
%            internally alpha is put equal to zero.
% kappaX, kappaY
%            For a paraboloid: the principal curvatures, as above
%            For a flat reflector: input with both kappas equal to zero;
%            shape may be 1, 2 or 3.  Within the subroutine ray.m that is
%            called, shape is equated to 3 (paraboloid)
%            For a cylinder: kappaX (positive or negative but not zero) is
%            the curvature; a convention is enforced whereby the input kappaY
%            is required to be zero
%            For a sphere: kappaX (positive or negative) is the
%            curvature, and kappaY must equal kappaX
%            For any of the kappas, a positive value means that the relevant
%            centre of curvature lies forward of the chief reflecting
%            point.
% upob, vpob          coordinates upri, vpri (i.e. uprime, vprime) of the point
%            R on the receiving array (of which the pseudoimage point S, or
%            other endpoint, is sought).  (In the descriptions of programs,
%            uprime and vprime are often abbreviated to upri and vpri.)  The
%            upri and vpri axes are in the plane of the array and the third
%            coordinate wpri of R is zero.  ('ob' is for 'object.')
% parax      (logical) 1 if the initial guess for (x, y) is to be that given
%            by the paraxial approximation (see above)
% xin, yin     the initial guess for (x, y) (ignored if  parax = 1)
%
% All inputs are in SI units (radian, metre, metre^(-1))
%
% SIGN, AND OTHER, CONVENTIONS, including the definitions of the xyz and
% (upri, vpri, wpri) coordinates systems: See the paper, Blair, D.:
% "Underwater acoustic imaging: Image due to a specular reflector in the
% geometrical-acoustics limit," J. Marine Science & Technol., vol 11,
% pp. 123-130 (2006).
%
% OUTPUT VARIABLES:
%
% xve        row vector of eleven elements, namely:
%   (1), (2)       x, y.  In turn these are coordinates of the pseuoimage
%              point S(x, y, z), or other endpoint (reported point), which lies
%              on the reflecting surface.  z, which is equal to Z, is given
%              by the three equations (C) above [together with the transfor-
%              mation from (x, y) to (X, Y) specified by the angle alpha].
%   (3)        value of iter upon exit.  This is the number of main iterations
%              carried out, not counting the initial iteration, except that an
%              unsuccessful 60 is rendered as 61.  The initial iteration is
%              labelled as iter = 0
%   (4)        value, upon exit, of iterini if iter = 0, otherwise of iter3.
%              In either case, this is the number of internal iterations
%              carried out within the last iteration, except that an
%              unsuccessful 60 (internal) is rendered as 61
%   (5)        failure code, equal to:
%        10      imige finds a pseudoimage point
%        11      maximum number of iterations (60) performed without
%                convergence
%        13      failure at calculation of derivative (should not occur if
%                coding done correctly)
%        16      [iter3 reached 60, OR step size reached its lower limit]
%                AND, in the last internal step, ray failed at one of five
%                points.  (iter3 is the number of internal iterations.  Of
%                the five points, four are associated with the
```

```
%                calculation of the derivatives.)
%        17      [iter3 reached 60, OR step size reached its lower limit]
%                AND, in the last internal step, ray produced five valid
%                points but error not reduced
%        21      iterini (number of internal iterations in the initial
%                iteration) reached 60.  This implies that the initial guess
%                for S did not produce five valid ray-pairs, even after 60
%                halvings of the x and y coordinates.  (Five,
%                because of calculation of derivatives, see 16 above.)
%   (6)        fail code, of ray, produced at central point reached in last
%              (60th) internal iteration
%   (7)        maximum of 5 fail codes, of ray, produced at 5 points reached
%              in last (60th) internal iteration
%              [Not sure re the following two statements: The codes in
%              elements (6) and (7) cannot exceed 8.  Also, if
%              iter upon exit equals zero, element (7) cannot be zero.]
%   (8)        relerrsuc, i.e. the relative error in vector (u, v) at end
%              of last iteration that succeeeded in reducing the error.
%              Here "relative" means "relative to the modulus of (u, v)"
%   (9)        relerr, i.e. the relative error in (u, v) obtained in the
%              last internal iteration
%  (10)        relstepsuc, i.e. the relative step in the vector (x, y)
%              carried out in the last iteration that succeeded in reducing
%              the error.  Here "relative" means "relative to the modulus
%              of (x, y)"
%  (11)        relstep, i.e. the relative step in the vector (x, y) tried
%              in the last internal iteration

function xve = imige5(shape, Zb, r0, e, delta, alpha, kappaX, kappaY, ...
    upob, vpob, parax, xin, yin)

% check for errors in input
if shape ~= 1 & shape ~= 2 & shape ~= 3
    disp('shape must equal 1, 2 or 3')
    pause
end

% shapi is the value for shape that the routine uses internally
if shape ~= 3 & kappaX == 0 & kappaY == 0
    shapi = 3;
else
    shapi = shape;
end

if shapi == 1 | shapi == 2
    if kappaX == 0
        disp('in imige, for a cylinder or sphere, kappaX must not be zero')
        disp('unless kappaY is also zero')
        pause
    end
    if shapi == 1 & kappaY ~= 0
        disp('in imige, for a cylinder, kappaY must be zero')
        pause
    end
    if shapi == 2 & kappaY ~= kappaX
        disp('in imige, for a sphere, kappaY must equal kappaX')
        pause
    end
end
if Zb < 0 & Zb ~= -1
    disp('in imige, Zb must be > zero or else equal to 0 or -1')
    pause
end
if shapi == 3 & (kappaX < 0 | kappaY < 0)
    if Zb <= 0 & Zb ~= -1
        disp(['in imige, for this shape Zb must strictly exceed zero', ...
        ' or else equal -1'])
        pause
    end
```

```
end
if shapi == 1 | shapi == 2
    if Zb > 1/abs(kappaX)
        disp('in imige, for cylinder or sphere, Zb must not exceed the radius')
        pause
    end
end

% ************************************************************************
% initial processing
xve = zeros(1, 11);
if upob == 0 & vpob == 0
    xve = [0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0];
else % A*** this loop is completed at end of routine
iter = 0;
iterout = -1;

flrycen = 0;
flryder = 0;
failim = 0;

% initial guess of x, y
if parax == 1
    xve2 = imparax(r0, e, delta, alpha, kappaX, kappaY, upob, vpob);
    x = xve2(1);
    y = xve2(2);
else
    x = xin;
    y = yin;
end

% BEGIN INITIAL ITERATION (also called iteration zero) to produce initial
% guess for the sequence of main iterations
% ****************************************************************

% In detail, if the routine ray.m fails with the very initial guess,
% repeatedly halve the vector [x, y] until that routine succeeds.  Success
% means that a point Sg = Sguess has been found that produces a ray meeting
% the array.  Success is required both at [x, y] itself and at the four
% points used in estimating the derivatives.
exitini = 0;
iterini = 0;
while exitini == 0
    fail1 = 0;
    iterini = iterini + 1;
    distlarge = sqrt(x^2 + y^2);
    upve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, x, y);
    up = upve2f(1);
    vp = upve2f(2);
    fail10 = upve2f(3);
    errsq = (up - upob)^2 + (vp - vpob)^2;

    instep = 0.00005*distlarge;
    x1 = x - instep;
    x2 = x + instep;
    y1 = y - instep;
    y2 = y + instep;
    upve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, x1, y);
    fail1 = max(fail10, upve2f(3));
    upve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, x2, y);
    fail1 = max(fail1, upve2f(3));
    upve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, x, y1);
    fail1 = max(fail1, upve2f(3));
    upve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, x, y2);
    fail1 = max(fail1, upve2f(3));

    if fail1 == 0
        exitini = 1;
    else
```

```
        x = x/2;
        y = y/2;
    end

    if iterini >= 60
        xyvetrial  = [x, y];%^%
        iterout = 0;
        failim = 21;
        flrycen = fail10;
        flryder = fail1;
    end
end

if failim ~= 21 % D*** 'if' loop is completed at start of main output

% BEGIN MAIN ITERATIONS ************************************************

idem = [1 0; 0 1];
jac = zeros(2, 2);
exit = 0;
exitbad = 0;
while exit == 0
    iter = iter + 1;

    % test for end of iterations

    if iter >= 61
         disp('number of main iterations reached 61');
         %^%disp('61 main iterations reached; press return')
         failim = 11;
         %pause
         exit3 = 1;
         exit = 1;
         exitbad = 1;
    end

    if exitbad == 0 % B*** loop is completed just before start of
                    % internal iterations
    fail2 = 0;
    fail3 = 0;
    dist = sqrt(x^2 + y^2);
    delu = upob - up;
    delv = vpob - vp;

    % calculate estimated derivatives ux, vx, uy, vy and the Jacobian
    % matrix, jac
    instep = 0.00005*dist;
    x1 = x - instep;
    x2 = x + instep;
    y1 = y - instep;
    y2 = y + instep;
    upve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, x1, y);
    uatx1 = upve2f(1);
    vatx1 = upve2f(2);
    fail3 = upve2f(3);
    upve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, x2, y);
    uatx2 = upve2f(1);
    vatx2 = upve2f(2);
    fail3 = max(fail3, upve2f(3));
    upve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, x, y1);
    uaty1 = upve2f(1);
    vaty1 = upve2f(2);
    fail3 = max(fail3, upve2f(3));
    upve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, x, y2);
    uaty2 = upve2f(1);
    vaty2 = upve2f(2);
    fail3 = max(fail3, upve2f(3));

    if fail3 ~= 0
```

```
    % disp('fail3 > 0; failure at calculation of derivative')
    failim = 13; % This instruction is probably a red herring,
                 % i.e. probably it never gets carried out
end

ux = (uatx2 - uatx1)/(2*instep);
vx = (vatx2 - vatx1)/(2*instep);
uy = (uaty2 - uaty1)/(2*instep);
vy = (vaty2 - vaty1)/(2*instep);
jac = [ux uy; vx vy];

% Set initial lambda
if iter == 1
    kac = jac'*jac;
    lam = 0.1*max(abs(kac(1,1)*kac(2,2)), abs(kac(1,2)*kac(2,1)));
end

end % B *** end of initial loop re main iterations
    % NOTE: It is important not to collapse B and E into one loop

if exitbad == 0 % E *** loop is completed just before end of
                % code-for-the-iteration

iter3 = 0;
exit3 = 0;

% BEGIN INTERNAL ITERATIONS ****************************************
% i.e. begin while loop (internal iterations), during which lambda may
% be repeatedly doubled; normally doubling is performed if fail2 > 0.
% lambda is halved if the first internal iteration is successful.
% iter3 and exit3 refer to internal iterations

exitinbad = 0;
while exit3 == 0
    iter3 = iter3 + 1;

    if iter3 >= 61
        if fail2 ~= 0
            failim = 16;
        else
            failim = 17;
        end
        %pause
        exit3 = 1;
        exit = 1;
        exitinbad = 1;
    elseif iter3 > 1 & relstep < 1.e-10
        if fail2 ~= 0
            failim = 16;
        else
            failim = 17;
        end
        exit3 = 1;
        exit = 1;
        exitinbad = 1;
    end

    if exitinbad == 0 % C*** 'if' loop ends at end of internal iter'n loop
    flrycen = 0;
    flryder = 0;

    % Calculate vector step given by the Levenberg-Marquardt method,
    % then the tentative new values of x and y, namely xt and yt.
    % In xt, yt, etc., "subscript" t is for "tentative"

    matri = jac'*jac + lam*idem;
    if det(matri) == 0
        disp('matri is a singular matrix.  Press any key to try to')
        disp('remedy this (by changing lambda)')
```

```
            iter
            iter3
            pause
            lam = lam*1.1;
            matri = jac'*jac + lam*idem;
            if det(matri) == 0
                disp('matri is doubly a singular matrix')
                pause
%    NOTE: It is not clear whether matri can even singly be a singular
%    matrix.  Such an ending has never occurred, let alone a "doubly"
%    ending.  If either does occur, it might signal an exact local optimum.
%    In any case, such an ending can be treated as a rare type of
%    non-convergence.
            end
        end
        inma = inv(matri);
        delu2 = [delu; delv];
        delxvet = inma*jac'*delu2;
        delx = delxvet(1);
        dely = delxvet(2);
        xt = x + delx;
        yt = y + dely;

        uptve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, xt, yt);
        upt = uptve2f(1);
        vpt = uptve2f(2);
        fail20 = uptve2f(3);
        errsqt = (upt - upob)^2 + (vpt - vpob)^2;

        % logic1 and logic2 are two tests, each of which is
        % sufficient for finding a pseudoimage point
        obdist = sqrt(upob^2 + vpob^2);
        relerr = sqrt(errsqt)/obdist;
        logic1 = (relerr < 1.e-10);
        step = sqrt(delx^2 + dely^2);
        dist = sqrt(x^2 + y^2);
        relstep = step/dist;
        logic2 = (iter3 == 60) & (relstep < 1.e-5) & (relerr < 1.e-5);

        % When testing (below) for successful end of sequence of internal
        % iterations, ray may fail at the points intended to
        % be used for calculation of the derivative.  Hence tests must be
        % performed at these points
        distfut = sqrt(xt^2 + yt^2);
        instep = 0.00005*distfut;
        x1 = xt - instep;
        x2 = xt + instep;
        y1 = yt - instep;
        y2 = yt + instep;
        upve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, x1, y);
        fail2 = max(fail20, upve2f(3));
        upve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, x2, y);
        fail2 = max(fail2, upve2f(3));
        upve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, x, y1);
        fail2 = max(fail2, upve2f(3));
        upve2f = ray(shapi, Zb, r0, e, delta, alpha, kappaX, kappaY, x, y2);
        fail2 = max(fail2, upve2f(3));

        % test for "PSEUDOIMAGE FOUND"
        if (fail20 == 0) & (logic1 | logic2)
            %^%disp('pseudoimage found; answer follows')
            failim = 10;
            %pause
            x = xt;
            y = yt;
            up = upt;
            vp = vpt;
            errsq = errsqt;
            relerrsuc = relerr;
```

```
            relstepsuc = relstep;
            exit3 = 1;
            exit = 1;

        % test for successful end of sequence of INTERNAL iterations
        elseif (fail2 == 0) & (errsqt < errsq)     % success
            x = xt;
            y = yt;
            up = upt;
            vp = vpt;
            errsq = errsqt;
            relerrsuc = relerr;
            relstepsuc = relstep;
            stepsuc = step;
            exit3 = 1;

            % HALVE lambda if first internal iteration was successful
            if iter3 == 1
                lam = lam/2;
            end

        % DOUBLE lambda (for next internal iteration)
        else
            flrycen = fail20;
            flryder = fail2;
            lam = 2*lam;
        end
        end % C*** end of loop re internal iterations completed
    end                                              % end of while of internals
    end % E*** end of loop re main iterations completed
end                                            % end of while of main iterations
end % D*** completion of 'if' loop re fail in initial iteration

% ********************************************************************
% MAIN OUTPUT (for case where object point is not at origin)

xve(1) = x;
xve(2) = y;

xve(3) = iter;
xve(4) = iter3;
if iterout == 0
    xve(3) = 0;
    xve(4) = iterini;
end
xve(5) = failim;
xve(6) = flrycen;
xve(7) = flryder;
xve(8) = relerrsuc;
xve(9) = relerr;
xve(10) = relstepsuc;
xve(11) = relstep;

end % A*** completion of 'if' loop involving coordinates both zero
```

C.4 multigen.m

```
% Program multigen.m
%
% DESCRIPTION
% This main routine concerns a sonar array viewing a specular reflector.
% The input includes a sequence of points on the array, spaced equally
% along a line.  The graphical output attempts to show, for each such
% point, all the corresponding "end points" on the reflecting surface,
% being points reached in the search for a pseudoimage point
%
```

```
% Author of program: David Blair, from 3 January 2007 to July 2008
%
% produced from imsurfmn2.m, the earlier version of multisym.m
% short for "multiple endpoints", "general" (as opposed to symmetric)
%
% this routine was once called optmult
%
% BACKGROUND
%
% This program deals with the pseudoimage of the receiving
% array produced when an underwater acoustic imaging device "views" a specular
% reflector.  It assumes the geometrical approximation (but makes no other
% approximation).  It is a main routine, which repeatedly calls imige5.m (note
% spelling, i.e. not image).
%
% Within the geometrical approximation, each call to imige (i.e. imige5) ac-
% cepts a point R(uprime, vprime) on the array and an initial guess (xin, yin).
% imige proceeds to calculate a point S(x, y), on the reflecting surface,
% that is either a pseudoimage of R or is some other endpoint.  [Strictly
% speaking, the call calculates the first two coordinates (x, y) of S.]  For
% each of a number of object points (uprime, vprime), a number of initial
% guesses (xin, yin) are generated; and for each combination (uprime, vprime,
% initial guess) a corresponding endpoint (x, y) is calculated.
%
% The coordinates just referred to are described as follows.
% The uprime and vprime axes lie in the plane of the array, and a point in
% the array plane is specified by these coordinates. The x and y axes lie
% in the chief tangent plane with the origin at the chief relecting point.
% (For more on these coordinates, see the subroutine ray.m)  Throughout the
% program, all parameters of the array are held constant.
%
% The endpoint may be a pseudoimage point but is otherwise said to be a "ghost
% reported point."  FOR THE DEFINITION of, and a discussion of, blockage
% point, local optimum, "error," underlying endpoint and related terms, see the
% routine imige5.m.
%
% multigen (and certain other main routines that call imige) generates a
% "fail code" telling which of the above endings has occurred.  This is a
% more sophisticated and user-friendly code than that generated by imige,
% but the code is determined by the output values of two variables from imige.
% For details of the code, see under "Fail Codes" below.
%
% GENERAL
%
% The program generates endpoints for a sequence of points (uprime, vprime)
% lying along a line, which need not pass through the origin.  The user
% specifies:
% (i)  a 2-D vector (up1, vp1) pointing along the line
% (ii) the offset p, that is, the perpendicular distance from the origin to
%      the line.  p is taken positive if, after stepping from the origin to
%      the line, a right-hand turn is required in order to then move in the
%      direction of the vector (up1, vp1).  Negative if a left-hand turn
%
% Each value used for the vector (uprime, vprime) is specified by a
% parameter t, the relationship being
%         (upri, vpri) = (offset vector) + (up1,vp1)*t.               (1)
% t (which may be called the multiplier) is a scalar.  The values of t are
% uniformly spaced along a specified interval (tini, tfin).  If the offset is
% zero there is no need to consider the sign in (ii).  In that case only
% values of t of one sign need be input, because the system possesses
% symmetry.
%
% For each value of the multiplier [i.e. for each point (upri, vpri)], the
% program generates a number of values (usually of order 50) of the initial
% guess (xin, yin).  Each combination produces an endpoint (x, y).  But the
% number of DIFFERENT endpoints is normally much less than the "50".
% Indeed, the number of different endpoints has been found (so far) to always
% equal one, two or three, provided that:
% (1) the points of nonconvergence are put aside, and
```

```
% (2) all the blockage points produced are deemed (contrary to fact) to
% coincide
%
% **** The parameter dist:
% For display purposes, the program does not use t but instead uses dist.
% dist is the same as t but is an unscaled (but signed) distance.  Thus dist
% is the distance of the object point from the offset point, measured along
% the line of object points, counted positive in the direction of (up1, vp1).
%
% **** The output graphs and table:
%
% Consider the vector (x, y) as a function of dist (or of t). In general the
% function is not single-valued, but has multiple (up to three--or more if
% there are blockage points) values.  The program outputs three graphs.  Of
% these, the first graph, the "x" graph, plots, as points, the various
% combinations (dist, x) found.   The second graph, or "y" graph,
% is described similarly.  The third graph, or "xy" graph, plots the
% combinations (x, y) obtained as endpoints.  It reveals patterns not evident
% in the first two plots.
%
% Consider the case (see "Notes re Output" below) where the output of points
% of nonconvergence is suppressed.  Apart from the "misbehaviour" of
% blockage points, in each graph the points plotted lie on a small number
% of smooth curves (also called "branches"); furthermore, at each dist, the
% number of branches is one, two or three.  Significantly, on the xy graph, the
% blockage points become "well-behaved" like the pseudoimage points and the
% local optima: then indeed the points plotted lie on a small number of smooth
% curves.  (The number may be as large as four or five, because a range of
% values of dist is involved.  But if we confine attention to a
% sufficiently small interval of dist, the maximum number drops
% to three.)
%
% The program also outputs a table giving, for each combination of dist and
% initial guess, the value of the pair (x, y) obtained as the endpoint and
% also the fail code.  The table enables the user to identify which curve on
% the y graph is to be paired with a given curve on the x graph.
%
% **** The initial guesses
% For each value of dist, the prediction of the PARAXIAL approximation
% (xpar, ypar) for the pseudoimage position (x, y) is first calculated.  As
% the initial guesses, a number I (of order 50) of values of the vector
% (xin, yin) are generated.  Each of these is a scalar times the vector
% (xpar, ypar).  Thus they are arranged along a line.  They are arranged along
% the line in two grids, one with the scalar always positive
% and one with the scalar negative.  In each of the two cases, the logarithms
% of the scalars are equally spaced.
%
% INPUTS
%
% e deltad alphad    Standard variables describing the array system
% up1, vp1      a vector in the (uprime, vprime) plane pointing along the line
%              of points to be input as object points
% p            offset of the line of object points.  Its sign is positive if,
%              after moving from the origin along the offset, one then needs
%              to make a RIGHT-hand turn to move in the (up1, vp1)
%              direction
% tini, tfin   initial and final values of t [see Equation (1) above]
% N            number of values of t
% shape        whether the surface is a cylinder (= 1), sphere (= 2), or
%              paraboloid (= 3)
% Zb           specifies where the reflecting surface is cut off (details
%              in ray.m, also imige.m) (= -1 for default)
% r0           range
% kappaX, kappaY   the principal curvatures of the reflecting surface (see
%              ray.m for details)
%        Note: The various values of the vector (uprime, vprime), used as
%              inputs, are given by the parameter t through the equation
%                     vector = (offset vector) + t*(up1, vp1)
```

```
% ratiolo (positive)     minimum value of the ratio r of the guess for the
%              (x, y) vector to the paraxial prediction for the (x, y) vector
%              (suggest 10^(-2))
% ratiohi (positive)     maximum value of the ratio r (suggest 10^(2.5))
% I            I is the number of initial guesses (x, y), for a given object
%              point.  I must be even. I should be big enough to cover
%              positive and negative values of the ratio r.
%        Thus I = 2*h+2 where h is number of intervals on the positive
%              side of the r scale (suggest I = 52)
%        The LOGARITHMS of the sizes of the guesses are uniformly spaced.
%
% SPECIAL INPUTS
% hb           minimum ordinate allowed to appear on the graph
% ht           maximum ordinate allowed to appear on the graph
% showgh       (logical) is 1 if the graphs are to show also 'ghost' reported
%              points, i.e. not just pseudoimage points
% recordall    (logical):  (Note: At present only recordall = 1 is supported.)
%              The value 1 means that the table output to a
%              file is to give information for ALL input guesses.  The value
%              0 means that the table is to give information only for input
%              guesses that fail to yield a pseudoimage point
%
% Note re Inputs:
% deltad and alphad are in degrees
% All distances are in metres
%
% NOTES RE OUTPUT
%    GRAPHICAL
% As discussed above, three graphs are produced.  The first two are plots of x
% versus dist, and y versus dist, respectively, where dist is defined above.
% Points, not curves, are plotted.  The third graph plots all the endpoints
% (x, y), irrespective of dist.  In each graph, each point is plotted as
% a marker representing the fail code (see "Fail Codes" below).
%
% The program as it stands suppresses the plotting of points of
% nonconvergence.  If these are to be plotted, the user must slightly
% modify the program by commenting, uncommenting and copying appropriate
% instructions.  (In the end, only three instructions need to be changed.)
% For most purposes, suppression produces a superior, more clean-cut graph.

%
%    TABLE
% A table multigen.txt is produced giving, for each combination of an input
% point and an initial guess (x, y), the output (x, y) and the fail code.
%
%    FAIL CODES:
% 90 (in table) or a plus sign (in graph)
%       a pseudoimage point
% 91 or a diamond
%       blockage point.
%       (The point reached is not unique, in the sense that that point reached
%       depends on details of the algorithm; also in the sense that the point
%       reached depends on the initial guess)
% 92 or a circle
%       local optimum point. The "error" in the calculated object
%       point is a local minimum with respect to changes in (x, y).
%
%    *** NOTE: In the program as it stands, the plotting of endpoints with
%    either of the two fail codes below is suppressed (see subheading
%    "Graphical" above)
% 93 or a square
%       maximum-iterations point.
%       Normally one can identify the underlying type of endpoint as a
%       pseudoimage point, a blockage point or a local optimum by examining
%       nearby input points
% 94 or an asterisk
%       a 'rare' endpoint, i.e. other type of end to imige.m  Expected to
%       occur rarely or never
%
```

```
% *** NOTE: For the exact criteria determining the fail code, given the
% output from imige, see the program code under "set fail code"
%
% Note: Due to a bug in the version of MATLAB used, when the program is run
% on a subsequent occasion, the previous version of multigen.txt
% is not erased but is overwritten, beginning at the start of the
% file.  If the new version
% of the table to be output is SHORTER than the previous one, the later
% parts of the previous table are retained; the user needs to be aware of
% this.  From time to time, the user may wish to make a fresh start by
% deleting the .txt file at the command prompt.  (For the command to
% take full effect, it may be necessary to exit from MATLAB.)

format compact
format long
more off

% inputs ************************************************

showgh = 1;
recordall = 1;
ht = 300; % 300
hb = -300; % -300

ratiolo = 10.^(-2);
ratiohi = 10.^(2.5);
I = 52; % is 2*h+2 where h is number of intervals on positive side
%         of the r or ratio scale
up1 = 0.4;
vp1 = 0.6;
p = 0; % remember sign
tini = 0.75;
tfin = 1.35;
N = 38;

shape = 3;
Zb = -1;
r0 = 2;
kappaX = -0.375;
kappaY = -0.75;
e = 0.1;
deltad = 18;
alphad = 32;
% ****************************************** end of inputs

delta = deltad*(pi/180);
alpha = alphad*(pi/180);

if floor(I/2) == floor((I-1)/2)
    disp('I must be even')
    pause
end

fulcou = 0;
reccou = 0;
failcou = 0;
dgecou = 0;
bigdist = zeros(I*N, 1);
output = zeros(I*N, 7);

xp = NaN*ones(I*N, 1);
xo = NaN*ones(I*N, 1);
xd = NaN*ones(I*N, 1);
xs = NaN*ones(I*N, 1);
xa = NaN*ones(I*N, 1);
yp = NaN*ones(I*N, 1);
yo = NaN*ones(I*N, 1);
yd = NaN*ones(I*N, 1);
ys = NaN*ones(I*N, 1);
```

```
ya = NaN*ones(I*N, 1);

mag = sqrt(up1^2 + vp1^2);
up1unit = up1/mag;
vp1unit = vp1/mag;
pvecu = -p*vp1unit;
pvecv = p*up1unit;
stept = (tfin - tini)/(N-1);

for j = 1:N

% Calculate input uprime and vprime and the paraxial approximation
% to (x, y)
    t = tini + (j-1)*stept
    upri = pvecu + t*up1;
    vpri = pvecv + t*vp1;
    dist = t*mag;

    xve2 = imparax(r0, e, delta, alpha, kappaX, kappaY, upri, vpri);
    xpar = xve2(1);
    ypar = xve2(2);

    logstep = (log10(ratiohi) - log10(ratiolo))*2/(I-2);
    for i = 1:I
        [j i]
% Calculate the next initial guess (xin, yin) for start of iterations
        if  i <= I/2
            if i == 1
                loog = log10(ratiohi);
            else
                loog = loog - logstep;
            end
            xin = - xpar*10^loog;
            yin = - ypar*10^loog;

        else
            if i == I/2 + 1
                loog = log10(ratiolo);
            else
                loog = loog + logstep;
            end
            xin = xpar*10^loog;
            yin = ypar*10^loog;
        end

        xyout(1:11) = imige5(shape, Zb, r0, e, delta, alpha, kappaX, ...
          kappaY, upri, vpri, 0, xin, yin);

% A "guessed point" is an initial guess, but the number of guesssed points
% continues to accumulate when the input object point is changed.
%
% failcou counts the number of guessed points for which the endpoint is other
%    than a pseudoimage point                       (numnonpi = total found)
% dgecou counts the number of guessed points for which the end point is not
%    a pseudoimage point, a local optimum or a blockage point.  (Almost
%    always, and perhaps always, this means the maximum number of iterations
%    was reached.)  ("dge" for diverge)           (numnoncge = total found)
% reccou counts the number of guessed points for which output is to be
%    recorded in the output file                    (numrecs = total found)
% fulcou is the serial number of the combination (object point, initial
% guess)
        fulcou = fulcou + 1;
        reccou = reccou + 1;
        if xyout(5) ~= 10
            failcou = failcou + 1;
        end

        xx = xyout(1);
        yy = xyout(2);
```

```
        fim = xyout(5);
        frc = xyout(6);

        f = fulcou;
% set fail code
        if fim == 10 % pseudoimage point
            fd = 90;
            xp(f) = xx;
            yp(f) = yy;
        elseif fim == 16 & frc == 0 % blockage point
            fd = 91;
            xd(f) = xx;
            yd(f) = yy;
        elseif fim == 14 | fim == 17 % local optimum
            fd = 92;
            xo(f) = xx;
            yo(f) = yy;
        elseif fim == 11 % maximum-iterations point
            fd = 93;
            xs(f) = xx;
            ys(f) = yy;
        else % 'rare' endpoint
            fd = 94;
            xa(f) = xx;
            ya(f) = yy;
        end

        if fd == 93 | fd == 94
            dgecou = dgecou + 1;
        end

% truncate graphs at top and bottom
        if xp(f) >= ht | xp(f) <= hb | yp(f) >= ht | yp(f) <= hb
            xp(f) = NaN;
            yp(f) = NaN;
        end
        if xo(f) >= ht | xo(f) <= hb | yo(f) >= ht | yo(f) <= hb
            xo(f) = NaN;
            yo(f) = NaN;
        end
        if xd(f) >= ht | xd(f) <= hb | yd(f) >= ht | yd(f) <= hb
            xd(f) = NaN;
            yd(f) = NaN;
        end
        if xs(f) >= ht | xs(f) <= hb | ys(f) >= ht | ys(f) <= hb
            xs(f) = NaN;
            ys(f) = NaN;
        end
        if xa(f) >= ht | xa(f) <= hb | ya(f) >= ht | ya(f) <= hb
            xa(f) = NaN;
            ya(f) = NaN;
        end
        bigdist(fulcou) = dist;

% set variables to be output to table
        output(reccou, 1) = j;
        output(reccou, 2) = dist;
        output(reccou, 3) = i;
        output(reccou, 4) = loog;
        output(reccou, 5) = fd;
        output(reccou, 6) = xx;
        output(reccou, 7) = yy;
    end
end

% Plot graphs
more on
if showgh == 1
    plot(bigdist, xp, '+k', bigdist, xo, 'ok', bigdist, xd, 'dk')
```

```
%    plot(bigdist, xp, '+k', bigdist, xo, 'ok', bigdist, xd, 'dk', ...
%        bigdist, xs, 'sk', bigdist, xa, '*k')
else
    plot(bigdist, xp, '+k')
end
hold on
xlabel('distance along object line')
ylabel('x')
title('x-component of Pseudoimage Position')
hold off
disp('opportunity to view and save "x" graph')
pause

if showgh == 1
    plot(bigdist, yp, '+k', bigdist, yo, 'ok', bigdist, yd, 'dk')
else
    plot(bigdist, yp, '+k')
end
hold on
xlabel('distance along object line')
ylabel('y')
title('y-component of Pseudoimage Position')
hold off
disp('opportunity to view and save "y" graph')
pause

if showgh == 1
    plot(xp, yp, '+k', xo, yo, 'ok', xd, yd, 'dk')
else
    plot(xp, yp, '+k')
end
axis equal % ***
hold on
xlabel('x')
ylabel('y')
title('2-D Pseudoimage Position')
hold off
disp('opportunity to view and save "xy" graph')
pause

format short
numrecs = reccou
numnonpi = failcou
numnoncge = dgecou

% output to file 'multigen.txt'
fid = fopen('multigen.txt', 'w');
fprintf(fid, ' j      dist     i  loog  fail       x           y \n');
fprintf(fid, ...
    '%3u %11.4e %3u %7.3f %3u %12.5e %12.5e \n', ...
    output');
fclose(fid);

format long
more off
disp('end of main routine reached')
```